Journal of Information Processing Vol. 18 88–95 (Mar. 2010)

Regular Paper

Base Station Placement for Effective Data Dissemination in Sensor Networks

Aoi Hashizume,[†1] Hiroshi Mineno[†2] and Tadanori Mizuno[†3]

Reprogramming sensor nodes is important for managing sensor networks. The latest reprogramming protocols use radio communication to distribute software data. In multi-base station sensor networks, the placement of the base stations affects several wireless reprogramming performance metrics. We developed a method for placing base stations, and we evaluated the features of software dissemination for multi-base station sensor networks. Simulations showed that the placement and number of base stations and the number of data segments were the key parameters in software dissemination.

1. Introduction

Advances in MEMS and low power wireless communication technology have led to the development of wireless sensor networks (WSN). A typical WSN consists of a number of small battery-powered sensor nodes that sense, collect, and transfer various data autonomously. There are many WSN applications and services, including structural monitoring, security, position tracking, and so on. These networks include state-of-the-art technologies (ad-hoc network routing, data processing, position estimation, etc.), and these technologies are implemented as specific codes on the sensor nodes. These technologies are highly advanced and still developing. Therefore, these codes will be modified or extended in the future for long-running applications using WSNs. Thus, a method to efficiently reprogram many deployed sensor nodes is necessary.

Wireless reprogramming has been extensively researched[1)–4)]. Wireless reprogramming distributes the new code to a lot of sensor nodes using wireless multihop communication. **Figure 1** illustrates one wireless reprogramming process. The purpose of general protocols in WSNs is data acquisition: the base station aggregates lots of small amounts of data from the edge nodes. In contrast, the purpose of wireless reprogramming protocols is data dissemination: the base station disseminates large data packets to the entire network. Because WSNs are becoming larger, more than one base station is needed to disseminate software. However, most of the reprogramming protocols that have been discussed assume a single-base station environment, and multi-base station environments have not been discussed.

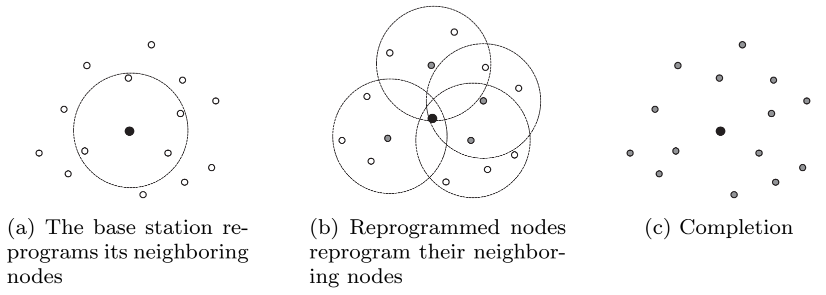

(a) The base station reprograms its neighboring nodes
(b) Reprogrammed nodes reprogram their neighboring nodes
(c) Completion

Fig. 1 Wireless reprogramming.

Here, we present methods of placing base stations in large-scale WSNs. First, we treat software dissemination as a facility location problem. Second, we treat multi-base station placement as a circle packing problem. Third, we evaluate the features of software dissemination for multi-base station WSNs.

This paper is organized as follows. In Section 2, we explain several issues related to wireless reprogramming and the facility location problem. An overview of our proposed technique, packing, is introduced in Section 3. We describe the simulation environment, evaluate packing, and analyze several characteristics of segmented data dissemination in Section 4. Finally, Section 5 summarizes the paper and mentions future work.

2. Related Issues

2.1 Challenges of Reprogramming

Many wireless reprogramming protocols share design challenges. Here, we deal

†1 Graduate School of Informatics, Shizuoka University
†2 Faculty of Informatics, Shizuoka University
†3 Graduate School of Science and Technology, Shizuoka University

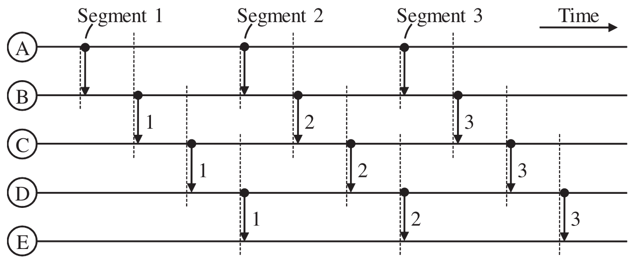


Fig. 2 Three pipelining segments in a four-hop network.

with the three most important challenges [1]:

(1) **Completion time:** The reprogramming completion time affects services using WSNs. When we reprogram the network, we have to stop the services and wait until the code update is completed. Thus, we have to minimize the reprogramming completion time.

(2) **Energy efficiency:** Sensor nodes are usually battery-powered and the sensor node battery provides the energy used in reprogramming. This battery also supplies energy for computing, communication, and sensing functions. Therefore, reprogramming must be energy-efficient.

(3) **Reliability:** Reprogramming requires the new code to be delivered throughout the entire network, and the delivered code must be executed correctly on the sensor nodes.

In the next section, we explain two techniques used in several reprogramming protocols to resolve the challenges listed above.

2.2 Reprogramming Approaches

2.2.1 Pipelining

Pipelining was developed to accelerate reprogramming in multihop networks [5),6)]. Pipelining allows parallel data transfer in networks. In pipelining, a program is divided into several segments, and each segment contains a fixed number of packets. Instead of receiving a whole program, a node becomes a source node after receiving only one segment. **Figure 2** shows the process of software distribution in pipelining. There are five sensor nodes deployed linearly. A dashed line represents the interference range, and a solid arrow represents the reliable communication range. To avoid the hidden terminal problem, the parallel data should be transferred with at least three-hop spacing. In Fig. 2, while (D) is sending segment 1 to (E), (A) is sending segment 2 to (B) simultaneously. Thus, pipelining can transfer program codes fast by overlapping the segments. Particularly for transmitting a large program, pipelining can reduce the completion time significantly.

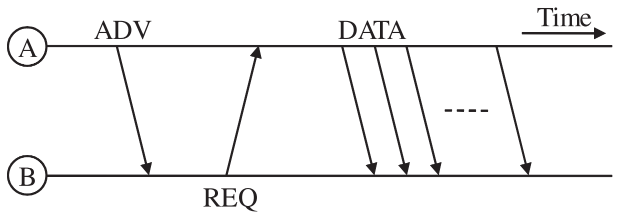


Fig. 3 Three-stage handshake.

2.2.2 Negotiation

Negotiation is used to avoid data redundancy and improve reprogramming reliability. As explained above, pipelining is done through segmentation. After the data is segmented, it is necessary to avoid broadcast storms that are caused by dealing with a large number of segments. A negotiation scheme was developed in SPIN [7)]. This scheme uses three-stage handshakes between senders and receivers. **Figure 3** shows the three-stage handshake. There are three types of messages (ADV, REQ, and DATA) in simple negotiation protocols. First, the source node (A) sends an ADV message, which includes its received segment information, to neighboring nodes (B). Second, if the destination node receives the ADV message, it compares its own segment with the received segment information and decides whether it needs the segment sent by the source node. If the segment is needed, the receiver requests the segment from the source node by sending an REQ message. Finally, if the source node receives the REQ message from the destination node, it forwards the requested DATA message. By using this scheme, the source node knows which segment is requested before sending it out. This reduces data redundancy. Although the negotiation scheme has advantages for pipelining, the control packets cause some problems. A three-stage handshake needs at least two control packets (ADV and REQ messages) to send each DATA message. Thus, if we increase the number of segments to accelerate code distribution, the number

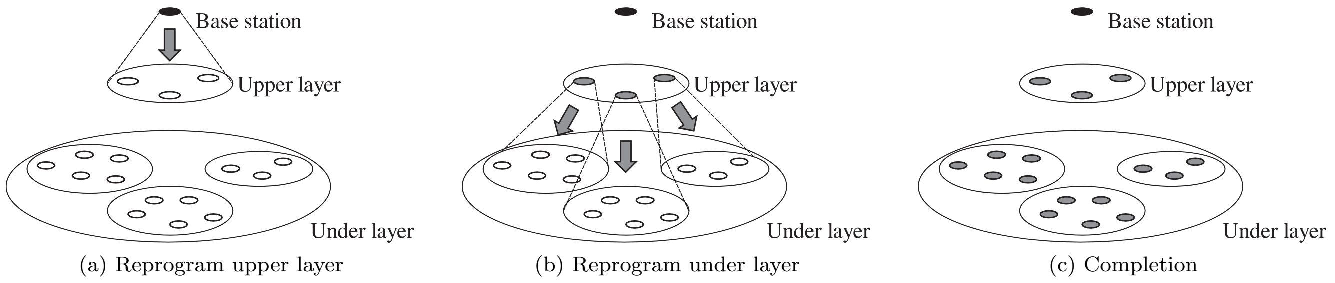


Fig. 4 Example of hierarchical reprogramming protocol behavior.

of control packets increases. This reduces energy efficiency and reliability for two reasons. First, sending messages is one of the most energy-consuming actions for sensor nodes. Second, when many messages are sent, message collisions may occur. Therefore, it is necessary to control the number of divided segments.

2.3 Hierarchical Reprogramming

Hierarchical reprogramming accelerates software distribution and reduces the number of control packets. Firecracker [8)] and Sprinkler [9)] are known as hierarchical reprogramming protocols. **Figure 4** illustrates their operation. First, the base station sends program codes to nodes in the upper layer of the node hierarchy (i.e., pseudo-base stations). Then the pseudo-base stations distribute codes to other nodes in their local areas. Except for Firecracker and Sprinkler, most reprogramming protocols start distributing software from a single base station in the network and assume no hierarchy.

Hierarchical reprogramming protocols improve reprogramming efficiency, but the decision method for the placement of pseudo-base stations has not been discussed. Deploying base stations in suitable places greatly improves the efficiency of the reprogramming.

2.4 Facility Location Problem

Constructing an efficient network is one of the challenges for sensor networks. Although there are several definitions of efficiency, we consider an efficient network to be a network that can accelerate software distribution, reduce network traffic, or reduce power consumption of sensor nodes. Network construction refers to the placement of base stations. The placement of base stations is treated as a facility location problem. A facility location problem is a mathematical problem that concerns deciding the most suitable location for facilities (e.g., educational facilities, medical centers, department stores, etc.) in terms of economic efficiency or convenience. These problems can be treated as packing problems, covering problems, or problems involving Voronoi diagrams. In this paper, we treat the placement of base stations as a circle packing problem.

3. Modeling and Analysis of Placement of Base Stations

Here, we assume program codes disseminate concentrically at constant speeds from each base station in a planar network. Placement order minimizes interference between the propagating waves of the software distribution from each base station. In other words, placement should maximize the dimensions of the concentric circles centered on each base station when propagating waves contact each other. Then, the placement decision becomes a circle packing problem. This problem has been discussed both as a theoretical geometrical problem as well as a hard test for global optimization methods. The circle packing problem is replaced by a problem that maximizes a minimum value of the distance between two circles and the distance between the circle and the boundary of the unit square. An optimized solution to the problem is expressed using the following symbols [10)].

S: Unit square
SB: Boundary of S

n: Number of equal circles that are packed in S
x_i: Center of a number i circle
D_n: Delaunay triangulation with $x_i (i = 1, \ldots, n)$
$d(x_i, x_j)$: Euclidean distance between x_i and x_j
$d(x_i, SB)$: Euclidean distance between x_i and SB

$$max \quad min[\min_{(i,j)\in D_n} d(x_i, x_j), \min_{i=1,\ldots,n} d(x_i, SB)]$$
$$s.t. \quad x_i \in S, \quad i = 1, \ldots, n$$

The best known packings of equal circles in the unit square are already known. **Figure 5** shows some examples of circle packings [11]. We used a packing approach for multi-base station placement. In this approach, base stations are placed in the center of each circle.

To evaluate the packing approach, we developed other methods called the random approach and the edge approach.

- **Random approach:** Placement of base stations is decided by uniform random numbers. This method is similar to the decision method of pseudo-base stations in Sprinkler [9].
- **Edge approach:** Base stations are placed at the edges of networks. This method is similar to the pseudo-base station selection in Firecracker [8]. Additionally, assuming networks are dealt with using Voronoi diagrams, all dimensions of the base stations' Voronoi cells are nearly equal.
- **Packing approach:** Base stations are placed at the centers of circles that are packed in the unit square.

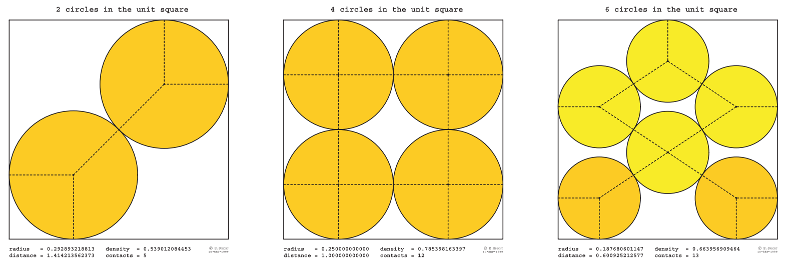

2 circles in the unit square
radius = 0.292893218813 density = 0.539012084453
distance = 1.414213562373 contacts = 5
© E. Specht 10-SEP-1999
4 circles in the unit square
radius = 0.250000000000 density = 0.785398163397
distance = 1.000000000000 contacts = 12
© E. Specht 10-SEP-1999
6 circles in the unit square
radius = 0.187680601147 density = 0.663956909464
distance = 0.600925212577 contacts = 13
© E. Specht 10-SEP-1999
(a) 2 circles (b) 4 circles (c) 6 circles

Fig. 5 Packings of equal circles in the unit square.

4. Simulation Evaluation

4.1 Environments

In this section, we describe an evaluation of the packing approach using the TinyOS [12] network simulator (TOSSIM [13]). The purpose of this evaluation is to show that the packing approach is superior to other approaches in terms of completion time, network traffic, and power consumption.

First, we explain the implementation and evaluation parameters of the simulation. To evaluate the proposed approaches, we adopted MNP [6] as a reprogramming protocol. This state-of-the-art protocol includes pipelining and negotiation. In TOSSIM, completion time and network traffic were observed, but the battery or power consumption of sensor nodes was not duplicated. Then, we calculated power consumption on the basis of the typical power consumption of Mica2 Motes in **Table 1** [1),14]. When node i sends S_{it} packets and receives R_{it} packets during t ms, the power consumption of node i $P_i(t)$ is expressed as follows.

$$P_i(t) = 20 \cdot S_{it} + 8 \cdot R_{it} + 1.25 \cdot t \qquad (1)$$

If there are k nodes in the network and the protocol needs T ms to reprogram an entire network, power consumption of the entirety P_{total} is as follows.

$$P_{total} = \sum_{i=1}^{k} P_i(T) \qquad (2)$$

Next, we describe the simulation environment. **Table 2** shows the simulation parameters. We assume each node has a transmission radius of 50 feet, so it can receive messages within a 25-foot radius. Nodes are deployed in a reticular pattern 10 × 10 or 20 × 20. Each node has 40 feet of spacing. Eighty packet-sized program codes are divided into 10 segments and distributed. **Figure 6** illustrates sample topologies when there were four base stations in the network. In Fig. 6, colored squares represent base stations and heavy lines express Voronoi

Table 1 Typical power consumption of Mica2 Motes.

Operations	Power consumption (nAh)
Send one packet	20.000
Receive one packet	8.000
Idle listen for 1 ms	1.250

Table 2 Simulation parameters.

Network topology	10 × 10 grid, 20 × 20 grid
Node spacing	40 feet
Radio transmission radius	50 feet
Program code size	80 packets
Number of segments	10 segments

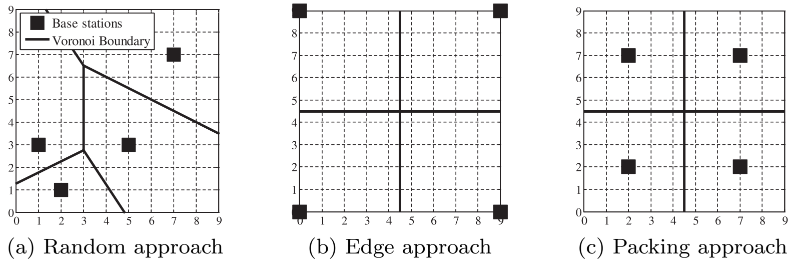


Fig. 6 Topologies with four base stations.

boundaries.

4.2 Base Station Placement

In this section, we compare three approaches in two networks of different sizes. **Figure 7** shows simulation results for a 10 × 10 network. These graphs show that every approach had shorter completion time and reduced network traffic and power consumption with more base stations. The packing approach obtained almost all the best scores compared with the others, regardless of the number of base stations.

Figure 8 shows simulation results for a 20 × 20 network. As in Fig. 7, all approaches improved reprogramming efficiency. Again, packing was the most efficient placement approach.

4.3 Network Size and Number of Base Stations

Next, we evaluate the relationship between the number of base stations and the network size. We used only the packing approach and increased the number of base stations from 5 to 25. **Figure 9** compares the performance of the entire network for a 10×10 network and a 20×20 network. Every result shows that in a 20 × 20 network, the characteristics of the evaluation parameters monotonically

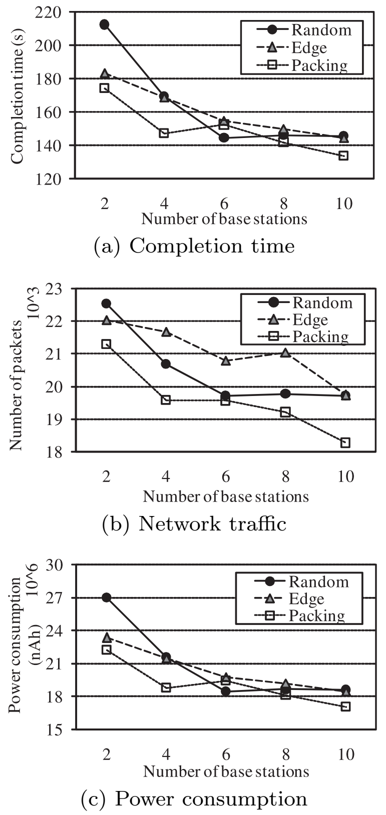


Fig. 7 Comparison of three approaches in a 10 × 10 network.

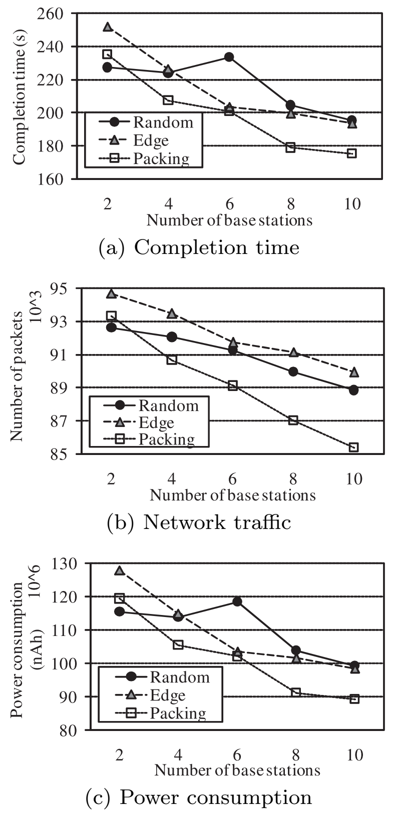
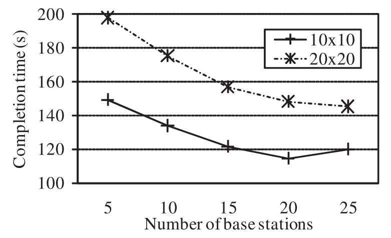


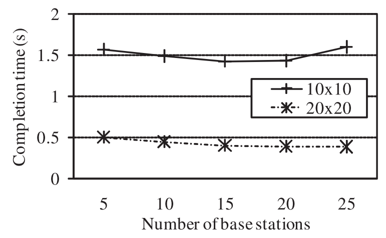
Fig. 8 Comparison of three approaches in a 20 × 20 network.

decreased as the number of base stations increased. On the other hand, in a 10 × 10 network, increasing the number of base stations caused poor results.

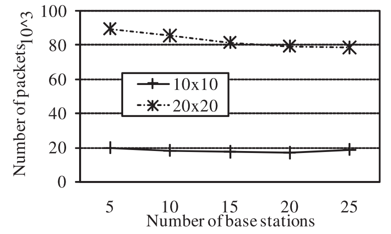
Figure 10 shows performance per node in the two networks. The values in Fig. 9 divided by the number of its own nodes gives the values in Fig. 10. A 10 × 10 network's reprogramming efficiency clearly deteriorated as a result of an increase in the number of base stations. Thus, it is conceivable that there are upper limits to the number of base stations for each network size that can help improve reprogramming performance efficiently. If so, exceeding the upper limit



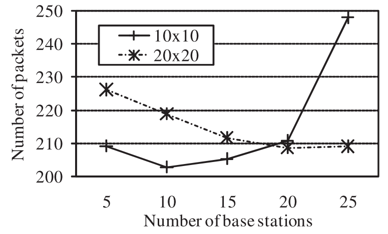
(a) Completion time



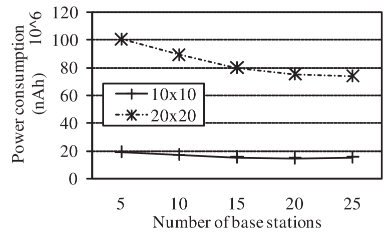
(a) Completion time



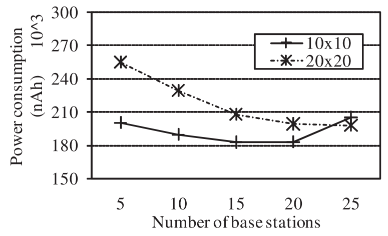
(b) Network traffic



(b) Network traffic



(c) Power consumption



(c) Power consumption

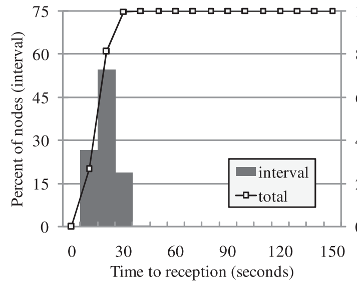
Fig. 9 Comparison of 10 × 10 network with 20×20 network using packing approach (entire network).

Fig. 10 Comparison of 10 × 10 network with 20×20 network using packing approach (per node).

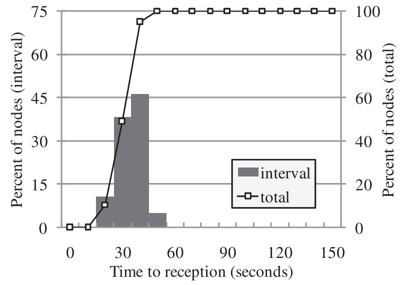
would have adverse effects on reprogramming.

4.4 Analysis of Adverse Effects

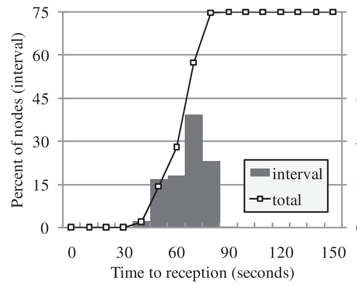
The adverse effects are due to the inherent characteristics of pipelining. Pipelining works well when there is a large number of hops between base stations and farthest destination nodes. In contrast, a small number of hops causes delay in code distribution. In the 10 × 10 network in Figs. 9 or 10, when there are five base stations, each base station has to send segments to the farthest four-hop



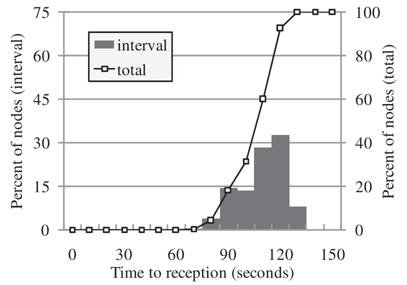
(a) 1 segment



(b) 2 segments



(c) 5 segments



(d) 10 segments

Fig. 11 Propagation time for four sample segmentation numbers.

nodes. Furthermore, each base station only has to send segments to one-hop nodes when there are 25 base stations. Therefore, increasing the number of base stations shortens the required number of hops needed to reprogram the entire network and causes inefficient pipelining.

We also evaluate the influence of the segmentation number on reprogramming performance. In this section, we use only the packing approach and change the segmentation number from one to ten in a 20 × 20 network. **Figure 11** shows the percentage of nodes that were programmed in a given 10-second interval. These graphs indicate the reason for the adverse effects. From Figs. 11 (a) to (d), program reception times are delayed due to the increased numbers of segments. Furthermore, entire completion times are also delayed. Therefore, it is clear that segmentation and pipelining distribution did not work well.

Next, we take notice of the control message and the power consumption. **Fig-**

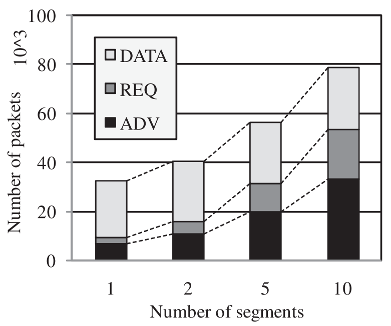


Fig. 12 Details about control messages.

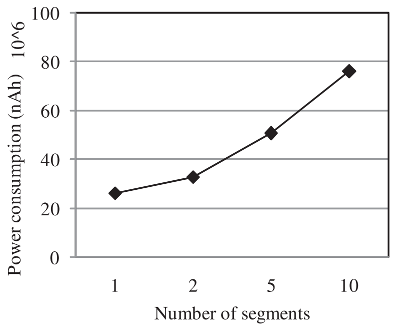


Fig. 13 Power consumption.

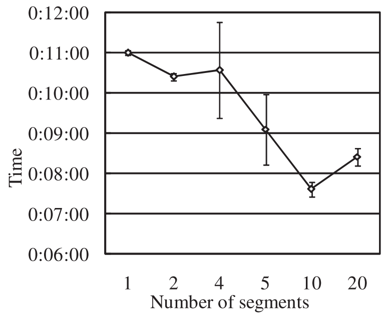


Fig. 14 Completion time of entire network.

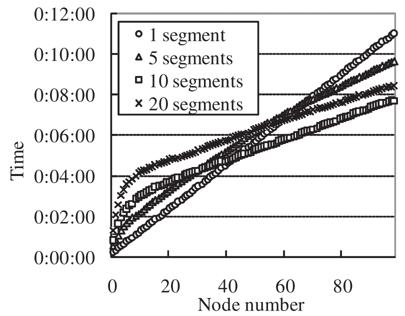


Fig. 15 Completion time per node.

ure 12 illustrates details of the control messages. In contrast to the constancy of the DATA messages (program data), ADV messages and REQ messages increased with the number of segments. This shows that even with a small number of program data retransmissions, there may be any number of control messages used to negotiate with neighboring nodes. Increasing the number of control messages (packets) causes data collisions and deteriorates data transfer reliability. **Figure 13** illustrates the power consumption of the entire network related to completion time and control messages.

We also analyzed the characteristics of pipelining in linear networks. Nodes are deployed in a 1×100 linear pattern. We changed the number of segments from 1 to 20. **Figure 14** shows the completion time for the entire network. Division number 10 led the fastest transfer, and division number 20 led a poor result. **Figure 15** shows the completion time per node. In 1 segment and 5 segments, nodes placed near the base station were reprogrammed faster than 10 segments. This suggests that when short paths are needed to disseminate data, having fewer segments is desirable.

Therefore, due to the pipelining characteristics, when a base station has to distribute a program code to only a few-hop nodes, segmented data dissemination is not suitable. Segmentation and pipelining work well when long paths are needed to propagate program data in wireless reprogramming. For fast transfer, the packing approach is useful. If few hops are needed to disseminate software data, we do not have to divide the data into segments. If many hops are needed, segmentation is required.

5. Summary and Future Work

We presented a packing approach that can reprogram large-scale sensor networks efficiently by increasing the number of base stations and placing them in an appropriate manner. Simulations showed that the packing approach can shorten completion time and reduce network traffic and power consumption effectively. In addition, we found that the optimal number of base stations is determined by the network size. Furthermore, we characterized segmented data dissemination in multi-base station sensor networks.

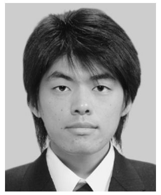
In future work, we will try to change simulation topologies and evaluate them in various environments. For instance, we will scale network size, place barricades, and deal with node irregularities. Then, we will search for the numbers of base stations, placements, and the numbers of segments that are best suited to the targeted networks.

References

1) Wang, Q., Zhu, Y. and Cheng, L.: Reprogramming wireless sensor networks: Challenges and approaches, *IEEE Network*, Vol.20, No.3, pp.48–55 (2006).
2) Thanos, S., Tyler, M., John, H., et al.: A remote code update mechanism for wireless sensor networks, *CENS Technical Report* (2003).
3) Pradip, D., Yonghe, L. and Sajal, K.D.: ReMo: An Energy Efficient Reprogramming Protocol for Mobile Sensor Networks, *Proc. IEEE PerCom*, pp.60–69 (2008).

4) Leijun, H. and Sanjeev, S.: CORD: Energy-efficient Reliable Bulk Data Dissemination in Sensor Networks, *Proc. IEEE INFOCOM*, pp.574–582 (2008).
5) Jonathan, W.H. and David, C.: The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale, *Proc. ACM SenSys*, pp.81–94 (2004).
6) Sandeep, S.K. and Limin, W.: MNP: Multihop Network Reprogramming Service for Sensor Networks, *Proc. IEEE ICDCS*, pp.7–16 (2005).
7) Kulik, J., Heinzelman, R.W. and Balakrishnan, H.: Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks, *Wireless Networks*, Vol.8, No.2–3, pp.169–185 (2002).
8) Philip, L. and David, C.: The Firecracker Protocol, *Proc. ACM SIGOPS European Workshop* (2004).
9) Vinayak, N., Anish, A. and Prasun, S.: Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices, *Proc. IEEE RTSS*, pp.277–286 (2005).
10) Nanzan University: *Academia, Mathematical Sciences and Information Engineering*, Vol.2, pp.55–60 (2002).
11) Eckard Specht: Packing of circles in the unit square (52C17) (online). http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html (accessed 2009-02-10).
12) UC Berkeley: TinyOS Community Forum (online). http://www.tinyos.net/ (accessed 2009-05-25).
13) Philip, L., Nelson, L., Matt, W., et al.: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, *Proc. ACM SenSys* (2003).
14) Crossbow Technology, Inc.: *Mica2 Wireless Measurement System Datasheet* (2003).

(Received May 25, 2009)
(Accepted December 17, 2009)
(Released March 10, 2010)



Aoi Hashizume received his B.I. degree in Informatics from Shizuoka University, Japan in 2009. He is currently working towards the M.I. degree at the Graduate School of Informatics, Shizuoka University, Japan. His research interests include computer networks, mobile computing, protocol engineering and reprogramming in wireless sensor networks. He is a member of the Information Processing Society of Japan.

Hiroshi Mineno received his B.E. and M.E. degrees from Shizuoka University, Japan in 1997 and 1999, respectively. In 2006, he received the Ph.D. degree in Information Science and Electrical Engineering from Kyushu University, Japan. Between 1999 and 2002 he was a researcher in the NTT Service Integration Laboratories. In 2002, he joined the Department of Computer Science of Shizuoka University as an Assistant Professor. His research interests include mobile computing, sensor networks and heterogeneous network convergence. He is a member of the Information Processing Society of Japan, IEICE, IEEE, ACM and the Informatics Society.

Tadanori Mizuno received his B.E. degree in Industrial Engineering from Nagoya Institute of Technology in 1968 and received the Ph.D. degree in Engineering from Kyushu University, Japan, in 1987. In 1968, he joined the Mitsubishi Electric Corporation. Since 1993, he has been a Professor of Shizuoka University, Japan. Now, he is a Professor of the Graduate School of Science and Technology of Shizuoka University. His research interests include mobile computing, distributed computing, computer networks, broadcast communication and computing, and protocol engineering. He is a member of the Information Processing Society of Japan, IEICE, the IEEE Computer Society, ACM and the Informatics Society.