Name-based Location Service for Mobile Agents in Wireless Sensor Networks

Shunichiro Suenaga
The Graduate University for Advanced Studies,
Nihon Unisys Ltd.
suenaga@nii.ac.jp

Shinichi Honiden
National Institute of Informatics,
The University of Tokyo
honiden@nii.ac.jp

ABSTRACT

Mobile agent technologies are effective ways of deploying applications in wireless sensor networks (WSN). There are currently several different approaches for enabling communication between agents operating on a WSN, but none of them allows for dynamic agent location determination. This introduces some restrictions which severely limit the usefulness of mobile agents on WSNs. In this paper, we propose a name-based location service for mobile agents in a WSN. The approach is largely implemented through the regular placement of *landmark* nodes across the WSN. We show that our approach can enable communication with the nearest target agent and maintains a constant level of communication among mobile agents over the whole WSN.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Design, Performance, Experimentation

Keywords

sensor networks, mobile agent, location service, middleware

1. INTRODUCTION

Wireless sensor networks (WSN) are progressing towards technical maturity, and their widespread utilization will soon be a reality. Object tracking and monitoring of the natural environment are expected applications where WSNs will be particularly useful[10]. For an application to run on a WSN, software must be installed on each node, and the nodes must exist in a particular physical location. Moreover, WSN-based applications are just like any other application in that they require maintenance, e.g., software patches. Given that the application software is installed on each individual node, it is largely infeasible to conduct such updates manually. Two general solutions currently exist: network reprogramming(Deluge[8] and MNP[14]) and mobile agent technology (SensorWare[2], ActorNet[11] and Agilla[4]). Reprogramming updates applications all at once on all nodes and mobile agent technology updates software gradually. Mobile agent technology is more suitable than reprogramming for adding and changing applications because it allows existing applications to continue to execute.

However, explorations into inter-agent communication in the domain of WSNs are still young. Currently, several approaches to establish communication between mobile agents in WSNs do exist, yet none of them manages the agents' locations. This means that communication is either achieved through hard-coding the location of a shared tuple space or through broadcasting. These restrictions are detrimental to the real-world application of mobile agent WSN systems because mobile agents can move and WSN nodes have limited resources.

This paper introduces a communication-enabling location service for mobile agents across an entire WSN. The method enables agents to communicate with the nearest target agent on demand without hard-coding a node location, broadcasting or periodic polling. The approach has been implemented in middleware and it divides a WSN into multiple clusters, generating a *landmark* node in each cluster. The *landmark* nodes keep track of local agent locations and respond to dynamic requests for such information. When an agent needs to communicate with another agent, it sends a request to the *landmark* for the type of agent it needs and obtains the nearest target agent's location.

Section 2 discusses related work and current problems, while Sections 3 and 4 describe our approach and how it is implemented. Section 5 shows an evaluation of our approach. In Sections 6 and 7 we enter into some discussion of the limitations and implications of our method and finish off with a discussion of possible directions for future work.

2. RELATED WORK

SensorWare[2] enables mobile agent migration between nodes in WSNs, the mobile agents are written as Tcl scripts. Agents are able to communicate, but the specific node to

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2802

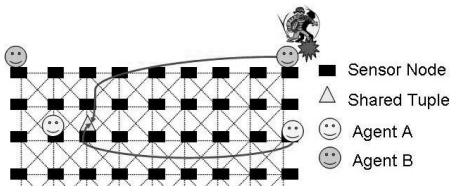

Figure 1: Problem of decoupled communication

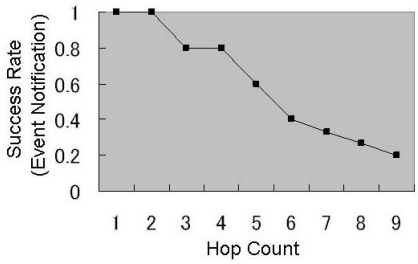

Figure 2: Success rate of event notification using decoupled communication

communicate with must be specified by the programmer. This approach does not allow for the migration of agents between nodes, because programmers must always be able to know and specify the physical location of the agents.

ActorNet[11] is an agent platform for WSN in which mobile agents are written in Scheme. ActorNet supports message passing via broadcasting. Any single broadcast in a large WSN requires a large number of messages to complete and repetitive broadcasting, as would be the case when two agents are communicating, is largely infeasible given the limited resources of WSN nodes.

In Agilla[4], communication between agents is performed through a shared tuple space. Using a shared tuple space is more suitable than the previous two approaches because agents can communicate without knowing the location of other agents. However, there are still two drawbacks. The first is that the location of the shared tuple space (i.e. a node) must be specified by the programmer. This leads to the second drawback, which is that agents often need to poll distant shared tuple spaces in order to check for new data. These restrictions can cause situations where two agents are polling a specific tuple space across network when they could talk directly with significantly less cost, as shown in Figure 1. Furthermore, communicating across network in a lossy environment means that the success rate of communications deteriorates the further agents get from their message's destination (i.e. the specified shared tuple space). Figure 2 shows the success rate of a notification event between two agents through a shared tuple space in a simulated lossy environment (TOSSIM[12][1]). It shows that maintaining reliable communications between mobile agents across an entire WSN requires a lot more resends the more hops there are between the message originator and destination. In this paper, we refer to shared tuple space communications as "decoupled communication" because there is a middle-man sandwiched between the sender and receiver.

There are several approaches proposed in wired environments and MANET, which are not suitable for WSNs. Feng et al. [3] propose an approach which establishes communication between mobile agents by using a moving mailbox in wired environments. Although this approach is reliable and effective, each node must keep track of the address of mobile agents. When applying this approach to WSNs, maintaining the consistency of address information on each node becomes a problem due to packet loss and messaging costs.

[1]TOSSIM's "lossy model" makes it possible to carry out a simulation of the asymmetrical link between nodes and the packet loss in WSN. We use 12-foot grid lossy model in this paper

ZRP[6] and LANMAR[13] are routing protocols related to MANET. They propose a method to maintain routing information within groups and zones, enabling proactive message delivery within those same groups and zones. This research focuses on the routing behaviour below the application layer, and does not focus on the utilization of application specific knowledge such as the type of agent or its location.

3. APPROACH

Given the problems stated in Section 2, we set out to create a solution that meets the following requirements.

1. The agent can obtain the location of the closest target agent dynamically in order to send messages directly to the target agent without hard-coding, broadcasting or polling.

2. There is a guaranteed worst-case level of communication reliability that does not depend on the size of the network.

The middleware we propose enables an agent to find the closest target agent among the ones who provide the desired service, and it also enables communication between agents by using the local shared tuple space on the node that the agent is located. Moreover, clusters are formed in order to keep the quality of communication between agents in the whole WSN constant. That is, each cluster can be considered its own mini-WSN and if a guaranteed level of communication reliability is maintained in all clusters, then the overall level of communication reliability for the WSN is equal to that of any individual cluster.

We choose to shy away from centralized search services and instead have implemented a spatially distributed search service, since communication quality and distance between nodes are more closely related in a WSN than they are in a wired network. In our method, the names of the agents that together form an application are defined according to their function or type. Clusters which divide up the WSN geographically are formed and *landmark* nodes are introduced to the center of each cluster in order to manage the types and locations of all agents local to the cluster. Agents registered in the middleware will have their type and location stored at the nearest *landmark*. When an agent needs to communicate with a specific type of agent, it sends a request to the *landmark*. The *landmark* replies with the

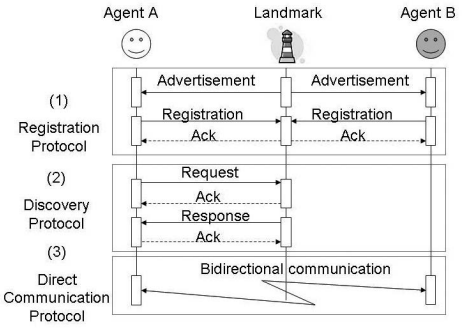

Figure 3: Communication protocol

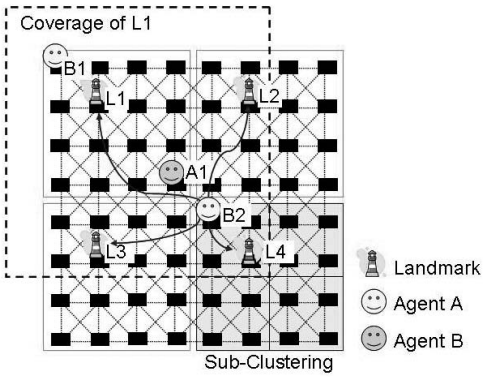

Figure 4: Registration to multiple landmarks

location of a nearby suitable target agent. After the originating agent knows the location of its target agent, it can start communicating with the target agent directly.

Although the locations and types of agents are registered in the *landmark*, the programmer does not need to specify a static location when writing the program. Locations are dynamically determined and obtained via the middleware.

There is one important assumption made in our approach that requires clarification up front: all clusters always have at least one instance of each possible "target agent". This is necessary to maintain the "mini-WSN" status of each cluster as well as to ensure that a *landmark* will be able to reply to any request (barring physical malfunction).

3.1 Landmark

The initialization, discovery and communication protocols of a *landmark* are shown in Figure 3.

In the *RegistrationProtocol*, the *landmark* broadcasts a message advertising that it is a *landmark*[2] to the surrounding nodes, and the middleware uses these messages to determine its *landmark* node. When an agent migrates to the node, the middleware transmits a registration message to the *landmark*,which registers the type and location of the agent. When the agent moves, the node makes sure to notify the *landmark* as well. In the *DiscoveryProtocol*, an agent specifies the type of the target agent in a request and sends it to the *landmark*. The *landmark* responds with the location, and communication between agents can be performed via the local tuple space. Thus, the *landmark* is out of the communicative loop once the agents enter the *DirectCommunicationProtocol*.

3.2 Boundary Solution

When an agent is deployed near the boundary of a cluster as shown in Figure 4, it is not always possible for a *landmark* to provide the nearest target agent. The agent searches for and finds a target agent deployed within its cluster at a distant position. In this example, $Agent - A$ (A1) does not communicate with the nearest agent, $Agent - B$ (B2), and instead communicates with B1. To avoid this, the locations and names of agents can be registered at two or more *landmarks* by the middleware. B2 deployed in the cluster of L4 can register at neighboring *landmarks* (L1, L2, L3) based on the position of the sub-cluster. The registration to the nearest *landmark* (in this case L4) is always carried out.

Sub-clustering is conducted by calculating locations using the information from the nearest *landmark*. For instance, when the B2's node receives the advertisement message from L4, the node knows that L4 is the nearest *landmark* and the size of the cluster is 4x4. The node also knows that it belongs to the upper-left sub-cluster of L4, and thus it can register with L1, L2, and L3. This means that the actual coverage of L1 is larger than the 4x4 cluster size as shown in Figure 4. In this way, *landmarks* are able to provide the agent with the location of the nearest target agent.

4. IMPLEMENTATION

The functions described in Section 3 were implemented as an extension of Agilla as shown in Figure 5. An agent cooperates with this middleware through the *Instructionset*. The *InstructionSet* has many functions, for example, searching for a target agent and moving agents to other nodes. The *InstructionSetHandler* interprets this *InstructionSet* and performs processing which distributes the extended functions of this approach, and the functions offered by Agilla.

The functions of the *landmark* and clustering are implemented in the *LandmarkEngine* and *ClusterManager*, respectively. The *LandmarkEngine* handles *landmark* initialization and agent type registration, communication between *landmarks*, and finding target agents. The programmer merely has to register the agent type and the middleware will handle low-level communications such as those between nodes and the *landmark*. The characteristic functionalities of clusters (sub-clustering, initialization, etc.) are implemented in the *ClusterManager*.

The *MappingTable* is used to store the locations and names of agents when the node is a *landmark* and it is used as a cache table when the node is not a *landmark*. In addition, the amount of information storable in the *MappingTable* can be specified in a configuration file.

[2]landmarks are initialized by flooding messages with the relevant topological information. Each node uses this information and its location to determine if it is a landmark. Changes to this algorithm allow us to adapt to other physical topologies.

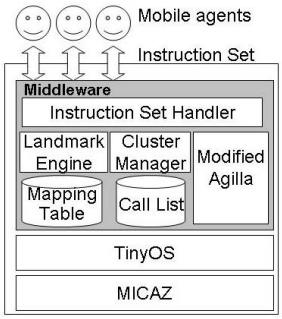

Figure 5: Architecture

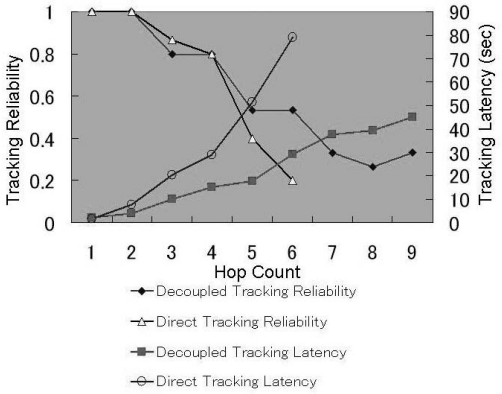

Figure 6: Tracking latency and reliability (TOSSIM, 12-foot, lossy)

5. EVALUATION

5.1 Evaluated Example and Desired Value

In this paper, we present an "Intruder Detection" example for performance evaluation. This application consists of two kinds of agents: one that detects the intruder and another that tracks the intruder. The former is called a *Detector* and the latter is called a *Tracker*. The *Detector* notifies the *Tracker* when it detects an intruder. The *Tracker* moves to the node in which the *Detector* exists and starts tracking the intruder. We evaluate the delay and accuracy of the intruder detection and tracking application implemented with the *landmark* middleware. We define "the duration from when the *Detector* detects an intruder to when the *Tracker* moves to the node in which the *Detector* exists and starts tracking" as the "tracking latency", and the success rate of this process as the "tracking reliability".

We aim to guarantee 80% reliability to start tracking within 10 seconds of receiving a notification from *Detectors*, therefore we set 0.8 as the desired average tracking reliability and 10 seconds[3] as the average tracking latency.

5.2 Simulation Environment

We assume a sensor[1] that can scan 5 m in diameter and can be equipped on MICAz. To cover the space with the scan field of the sensor, we need to arrange the nodes with sensors at intervals of 3.5 m[4]. In addition, the node knows its own location on a two-dimensional grid network.

We simulated this environment in TOSSIM[12], the evaluation used the lossy model (12-foot) with sensor nodes placed at intervals of 3.5 m. Communication between nodes was performed by simple greedy forwarding[9] using geographic routing as the routing protocol. Moreover, only adjoining nodes and nodes adjoining each other by diagonal lines can communicate with each other.

Note that all specific numerical values in this evaluation are determined by the application, and should not be taken as a limitation of our method.

5.3 Tracking latency and reliability

To achieve the desired values described in Section 5.1, average tracking latency and average tracking reliability were measured for every distance between agents in a 10x10 WSN. Figure 6 shows the tracking reliability and tracking latency of "decoupled communication" and "direct communication" when the *Detector* and *Tracker* are deployed at distances of 1 to 9 hops in a diagonal line [5]. Direct communication is the communication style proposed in this paper. Figure 6 shows the average tracking reliability and latency (measured 15 times for each hop distance). The tracking reliability for more than seven hops in our approach was 10% or less, and tracking latency was 120 seconds or more. Therefore, Figure 6 only shows the evaluation from one to six hops for our approach. The tracking reliability and latency are 80% and 28.9 s in our approach at 4 hops, and 53% and 29.29 s in "decoupled communication" at 6 hops. This result means our approach using 5x5 clustering can assure better tracking reliability and latency compared with "decoupled communication" when the WSN is 6x6 or larger. Also, as WSN sizes increase, the effectiveness of our approach also increases, because we can increase the number of clusters and still maintain the quality of communication within each cluster. Comparatively, the expected quality of "decoupled communication" only decreases as the network gets larger.

Figure 6 shows the result for agents deployed on a diagonal line. In this case, the distance between agents and the migration length of the *Tracker* are always the same. However, agents are only assumed to be deployed in a two-dimensional environment. We need to evaluate statistical data in consideration of other spatial arrangements and compare the performances of both approaches for the same WSN size. Table 1 shows the average tracking latency and average tracking reliability at deployment time for the *Tracker* at all the nodes three times for WSN sizes of 4x4, 5x5, 6x6

[3]Human intruders, such as thieves, are said to be able to gain entry in 10 - 30 seconds

[4]The diagonal lines between nodes on a grid are 5 m, the grid interval is approximately 3.5 m, 3.5 m corresponds to 12 feet

[5]The landmark of the proposal and the tuple space of the "decoupled communication" are located in the middle of the initial nodes where the *Detector* and *Tracker* were deployed. When the distance between agents is one hop, the *Tracker* is deployed to the same node as the landmark or another specified node.

Table 1: Spatial average of tracking latency and reliability (TOSSIM, 12-foot, lossy)

Evaluated Item	4x4	5x5	6x6	8x8
direct tracking latency	7.61 s	9.52 s	13.23 s	N/A
direct tracking reliability	0.90	0.83	0.77	N/A
decoupled tracking latency	7.27 s	8.86 s	12.24 s	16.17 s
decoupled tracking reliability	0.96	0.88	0.81	0.61

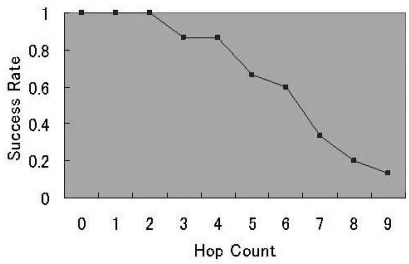

Figure 7: Success rate of registration (TOSSIM, 12-foot, lossy)

and 8x8 nodes. The *Detector* is always fixed at the corner of the WSN. Table 1 shows that the desired value (average tracking reliability: 80%, average tracking latency: 10 seconds) can be achieved for 4x4 and 5x5 clusters. In addition, we can achieve the same quality in an 8x8 WSN by dividing it into four 4x4 clusters. Thus, our approach not only keeps the communication quality constant in a large-scale WSN by creating multiple clusters, but also maintains the same quality as "decoupled communication" in a small-scale WSN.

5.4 Evaluation of Boundary Solution

We allow registration and deletion at two or more *landmarks* as discussed in Section 3.2. In this case, the success rate of registration and deletion was evaluated since the reliability of the information stored in the *landmark* is important. The relation between the distance to a *landmark* node and the success rate of registration is shown in Figure 7. Data shows the average value for 15 measurements. Given 4x4 clusters, the maximum number of hops for registration is four, and the average success rate is 87%. However, given 5x5 clusters, the average success rate rapidly falls to 65% in our simulated environment.

6. DISCUSSION

6.1 Programmability and Scalability

The proposed approach provides agents with a dynamic search mechanism for locating the closest target agent, therefore the programmer does not need to understand exactly where the agents will be at any given point in time. In addition, agents can obtain the location information of other similarly-typed agents from *landmarks*, enabling individual agents to autonomously limit their numbers in any particular area of a WSN. These primitives make it easy for programmers to control the spatial deployment of agents and to establish communication between them.

Although communication between *landmarks* is implemented in this middleware communication between *landmarks* is not used for communication between agents. When a large-scale WSN is composed of multiple clusters, creating the clusters does not become a restriction on the quality of communication between agents. In other words, as long as all kinds of agents are deployed within the coverage of each individual *landmark*, our proposal guarantees a constant level of communication quality between agents in the whole WSN. For example, consider a 30x30 WSN. From the results in Table 1, a 83% tracking reliability and a 9.52 seconds tracking latency can be guaranteed in the whole WSN by creating 36 5x5 clusters. On the other hand, we can not expect effective tracking if we use "decoupled communication".

6.2 Limitations of our proposal

Our proposal is not suitable for the following conditions.

1. Agents that move regularly with a short cycle.
2. Applications that cannot work within the coverage of a *landmark*.

In the first case, an agent needs to delete its name from *landmarks* before migration, and register its name again after migration. Figure 8 shows the duration and success rate between deletion and registration when the agent migrates. "Hop Count" is the distance between a *landmark* and an agent and the result displays the case when the agent moves to a different node with the same hop count. Figure 8 shows that agents need about 10 seconds to migrate between nodes given 5x5 clusters. This implies that if the target agents move every 10 seconds between nodes, the probability to establish communication may be 50%. Our middleware does generate an exception, allowing for the message to be resent, but the process causes some unignorable delays. We are considering message forwarding as a method to overcome this problem.

In the case of 2, our proposal can not enable communication between agents because it assumes the agents composing the applications always exist within the coverage of a *landmark*. For example, consider a warehouse in which nodes are attached to all the stock, and the WSN is constituted of multiple clusters and it manages the total inventory. In this case, the total inventory cannot be counted within the coverage of one *landmark*.

6.3 Target Application

We assume that applications consist of several agents whose relations can be defined by some events and communication between agents is only conducted when these events are detected. In addition, the agents migrate and behave autonomously before receiving these events. These assumptions fit event-based applications like "intruder detection"

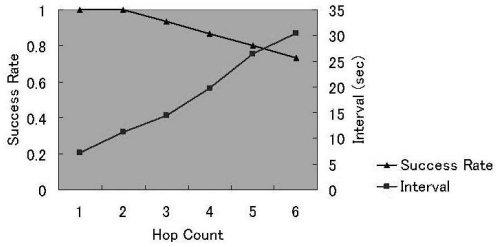

Figure 8: Registration and deletion overhead to a landmark (TOSSIM, 12-foot, lossy)

and "object tracking". On the other hand, our approach is not suitable for applications which require frequent communication between agents that are almost constantly moving.

6.4 Future Work

In our approach, since a *landmark* needs to communicate frequently, the battery of the node which performs the *landmark* function will generally be drained before the batteries of other nodes. Therefore, there appears to be room for dynamically changing the *landmark* node based on remaining battery power. GRA[5] defined the relation between the node's role and its attributes (battery, network connection, a sensing function, etc.), and has proposed a technique of rotating roles according to the nodes' variable attributes. LEACH[7] is an algorithm which rotates cluster heads to save the node's battery energy. We believe that *landmarks* can be rotated by using similar techniques. However, *landmark* rotation causes problem of optimal cluster formation. If a node at the edge of the cluster is chosen as a *landmark*, the quality of location service within the clusters can be variable (the cluster and surrounding clusters may need to be formed again). Optimal cluster formation is one of the future works. In addition, *landmark* cooperation might become necessary to provide redundancy and robustness when another *landmark* fails.

7. CONCLUSION

We proposed a middleware which provides a name-based location service for mobile agents in a WSN and divides a WSN into multiple clusters. By using our approach, the programmer needs not specify and care about the locations of mobile agents and nodes to enable communications between mobile agents. Our approach can also assure constant-quality communications over the whole large-scale WSN. *Landmark* cooperation and rotation of *landmarks*, optimal cluster formation are future work.

8. REFERENCES

[1] Panasonic motion sensor. `http://pewa.panasonic.com/pcsd/product/sens/select_motion.html`.

[2] A. Boulis, C.-C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 187–200, New York, NY, USA, 2003. ACM Press.

[3] X. Feng, J. Cao, J. Lü, and H. Chan. An efficient mailbox-based algorithm for message delivery in mobile agent systems. In *MA '01: Proceedings of the 5th International Conference on Mobile Agents*, pages 135–151, London, UK, 2002. Springer-Verlag.

[4] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 653–662, Washington, DC, USA, 2005. IEEE Computer Society.

[5] C. Frank and K. Römer. Algorithms for generic role assignment in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 230–242, New York, NY, USA, 2005. ACM Press.

[6] Z. Haas and M. Perlman. The zone routing protocol (zrp) for ad hoc networks. In *Internet draft, Mobile Ad-Hoc Network (MANET) Working Group*. IETF, 1998.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.

[8] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA, 2004. ACM Press.

[9] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[10] M. Kuorilehto, M. Hannikainen, and T. D. Hamalainen. A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 5(5):774–788, 2005.

[11] Y. Kwon, S. Sundresh, K. Mechitov, and G. Agha. Actornet: an actor platform for wireless sensor networks. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1297–1300, New York, NY, USA, 2006. ACM Press.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM Press.

[13] G. Pei, M. Gerla, and X. Hong. Lanmar: landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc*, pages 11–18, 2000.

[14] L. Wang. Mnp: multihop network reprogramming service for sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 285–286, New York, NY, USA, 2004. ACM Press.