Computer Communications 31 (2008) 3292–3298

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Efficient service discovery mechanism for wireless sensor networks ☆

Jaehoon Jung [a], Seunghak Lee [a], Namgi Kim [b,*], Hyunsoo Yoon [a]

[a] Department of EECS, KAIST, 373-1 Kuseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
[b] Department of Computer Science, Kyonggi University, 94-6 Iui-dong, Yeongtong-gu, Suwon, Gyeonggi 443-760, Republic of Korea

ARTICLE INFO

Article history:
Received 10 July 2007
Received in revised form 14 May 2008
Accepted 19 May 2008
Available online 29 May 2008

Keywords:
Service discovery
Contents discovery
Location-based service
Wireless sensor network

ABSTRACT

A ubiquitous computing environment is a large-scale mobile environment where numerous sensor devices are widely distributed and connected through wireless networks. However, existing service discovery protocols in sensor networks are inappropriate for such environments since they are not scalable. This paper proposes a DHT (Distributed Hash Table) based service discovery protocol including the mechanism that constructs topology-aware overlay networks in ubiquitous environments. Our protocol is scalable since it does not require a central lookup server and does not rely on multicast or flooding. Simulation results show that our protocol is scalable and outperforms traditional flooding-based protocols.

1. Introduction

In ubiquitous computing environments, numerous computing devices, such as portable or embedded devices as well as regular computers, will be widely distributed and connected by large-scale wireless networks. In order to get a desired service in such environments, we need to discover it first. For example, if users want to use a printer or a beam projector service, they first need to discover a device providing that service.

A ubiquitous environment is a large-scale mobile environment where numerous mobile sensor nodes are connected by wireless networks. However, existing service discovery protocols are inappropriate for ubiquitous environments. Centralized approaches using a lookup server [1–3] are not suitable for such ubiquitous environments since the whole infrastructure depends on a central node, which might be unreliable. Moreover, they are not scalable and might cause a bottleneck. Distributed approaches [4–6] appear better suited to ubiquitous environments. But these approaches are also not scalable since they rely on multicast or flooding mechanism which generates too much traffic. Therefore it is desired to investigate a scalable service discovery protocol which does not require a central lookup server and does not rely on multicast or flooding.

We can consider adopting DHTs (Distributed Hash Tables [7,8]) for scalable service discovery in wireless sensor networks. The DHT technique is a scalable solution for distributed data storage in peer-to-peer computing environments. However, recent work [10] explains that DHTs are inappropriate for ubiquitous environments because it is difficult to construct overlay networks reflecting a physical topology in such environments. If overlay networks do not reflect a physical topology, serious network overload due to the increase of the hop stretch[1] occurs.

Therefore, in this paper, we adopt DHTs in wireless sensor networks by suggesting a mechanism that constructs overlay networks reflecting a physical topology. We propose a DHT-based service discovery protocol. The proposed protocol constructs topology-aware overlay networks in large-scale wireless sensor networks. Therefore, our protocol is scalable because it does not require a central lookup server and does not rely on multicast or flooding.

The remainder of this paper is organized as follows: in Section 2, we survey related works. Section 3 describes our DHT-based service discovery protocol in detail. We present our simulation results in Section 4 and conclude in Section 5.

2. Related work

2.1. Previous service discovery protocols

Traditional solutions can be classified into two approaches: centralized protocols and distributed protocols. The centralized protocols, such as Jini [1], UDDI [2], and Salutation [3], rely on a central lookup server. On the other hand, the distributed protocols, such as SLP [4], UPnP [5], and Konark [6], rely on multicast or flooding. The distributed approaches appear more suitable than centralized ones

☆ This work was supported by the GRRC program of Gyeonggi province.
* Corresponding author. Tel.: +82 16 470 7414.
E-mail address: ngkim@kyonggi.ac.kr (N. Kim).

[1] The ratio of the number of physical hops in the overlay path to the number of physical hops in the shortest path.

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.05.015

in ubiquitous environments. However, the distributed approaches are still inappropriate for wireless sensor networks because they generate too much traffic for maintaining network topologies.

In distributed approaches, one recent work is based on the concept of clustering [9]. In the clustering, cluster headers take a role as central lookup servers in their clusters. So, centralized service discovery is locally performed in each cluster at first. And, in order to achieve global discovery over all clusters, cluster headers summarize their cached service descriptions and then deliver the summarized information to all other cluster headers. However, delivering summarized information also causes network-level flooding which generates too much traffic. Another recent work is GSD (Group-based Service Discovery) [10]. The GSD classifies services into several groups according to a service hierarchy. The GSD nodes advertise their service descriptions to their neighbors by broadcast. The nodes search desired services locally in their caches. To achieve global discovery in the GSD, the list of groups that the sender has seen in its vicinity is included in each advertisement. Then, the group information is used to selectively forward a service request. However, exchanging group information in the GSD scheme also generates relatively much traffic by using network-level flooding.

2.2. Distributed Hash Tables

Distributed Hash Tables (DHTs) [7,8] are the results of research efforts on peer-to-peer (P2P) computing networks. In P2P networks, services have service descriptions that consist of a service name, a service provider ID, and so on. The service descriptions are primarily used for identifying service. However, in P2P network adopting DHT, these descriptions are also used for storing and discovering services. When a service is created, the P2P network generates a service description. Then, it stores the service in the node, namely rendezvous node, corresponding to the hash value of the service description. Accordingly, if adopting DHT, we can build a scalable network because the DHT does not require a localized server and does not generate much traffic by flooding. However, in order to adopt DHT, the overlay networks must reflect a physical topology. If the topology of the overlay network mismatches that of the physical network, the hop stretch overhead severely increases. Ratnasamy et al. [13] used landmark clustering scheme to build a topology-aware CAN [7] overlay network. However, to construct a fixed set of landmarks is unsuitable for mobile environments.

There have been two approaches to build topology-aware overlay networks in mobile environments: PNS (Proximity Neighbor Selection) and PIS (Proximity Identifier Selection). In PNS approach, a node's overlay address is chosen randomly, but a proximity node is chosen when performing overlay routing. Cramer and Fuhrmann [11] proposed PNS using randomized Chord [8] with *k*-hop neighbor information. In the Cramer's method, messages are preferentially routed to the eligible node among *k*-hop neighbors. However, the PNS approach works only in dense networks because more than one 2-hop neighbor node should be existed in the node's hash range. In the PIS approach, on the other hand, a node's overlay address is chosen according to the node's current location. Winter et al. [12] proposed PIS which is based on overlay clustering. They suggest random landmarking (RLM), instead of fixed set of landmarks. In the RLM, the landmark keys (overlay IDs) are distributed over all regions and a node gets the closest landmark key in its region. To determine the closest landmark key, a node periodically measures distance to all landmarks. Then, a node shares a common prefix of the overlay ID with the closest landmark. The RLM utilizes the overlay lookup capabilities by reducing traffic overhead within an overlay cluster. However, it generates much control traffic for periodically measuring distances to all landmarks. Moreover, the hop stretch overhead still increases when performing overlay routing among overlay clusters.

3. Proposed scheme

In this section, we propose a scalable DHT-based service discovery protocol that is suitable for large-scale wireless sensor networks. We suggest a mechanism that constructs topology-aware overlay networks in ubiquitous environments. In order to construct the topology-aware network, we adopt the concept of clustering.

3.1. DHT-based service discovery protocol

Fig. 1 shows a basic mechanism of DHT-based service discovery. We adopt CAN [7]-based overlay networks to reflect a physical topology. We will describe how to construct topology-aware overlay networks in Section 3.2. In the CAN-based overlay networks, the service descriptions that consist of a service name, a service provider ID, etc. are stored in a distributed manner using a DHT-based algorithm. The specific service description is stored in the node, namely rendezvous node, corresponding to the hash value of the service name. In our scheme, we use the 2-dimensional Cartesian region algorithm to construct an overlay network and select the rendezvous node for the specific service description. Fig. 1b shows the example of a virtual overlay network which is constructed by the 2-dimensional Cartesian region algorithm with five nodes. Each node takes charge of the assigned region. When the first node is entered in an overlay network, it charges the entire region. After that, whenever the next node is entered, it selects a $\langle cx,cy\rangle$ coordination and divides the region for the $\langle cx,cy\rangle$ in two half with the previously assigned node. The newly entered node takes the responsibility for one of split regions and constructs routing table with the neighbor nodes. For instance, when a new node selecting the coordination $\langle 0.7,0.1\rangle$ is inserted into the Fig. 1(b) overlay network, the node B's region (0.5–1, 0–0.5) is divided into two half and the new node takes the region (0.5–1, 0–0.25) and the node B takes the region (0.5–1, 0.25–0.5).

After constructing an overlay network, the information for service descriptions is delivered to and from the rendezvous node by each node's routing table. When a service provider wants to advertise its service description information into the network, it firstly obtains the $\langle sx,sy\rangle$ hash key corresponding to the service name by using a well known hash function such as MD5 and SHA. After getting the $\langle sx,sy\rangle$ hash key, the service provider finds the rendezvous node belonging to the hash value and stores the service description into that node. For example, in Fig. 1, node E provides a beam projector service and wants to advertise its service to the network. Then, the node E firstly finds the rendezvous node for the beam projector by calculating the hash key of the beam projector service. If the beam projector's hash key is $\langle 0.2,0.3\rangle$, node A is the rendezvous node for beam projector because the node A takes the responsibility of region (0–0.5, 0–0.5). Therefore, node E advertises its beam projector service by storing its service description at node A. The hash key of the beam projector is always $\langle 0.2,0.3\rangle$ at all nodes because they share the same hash function. As a result, a service provider can advertise its service description on the entire overlay network by storing the service description at the rendezvous node.

In the same way, a service seeker discoveries the desired service using the service's rendezvous node. In order to find a specific service, a service seeker also obtains the service's $\langle sx,sy\rangle$ hash key by the same hash function. Using this hash key, the service seeker finds out the service's rendezvous node and sends a query where the service is available. Then, the service's rendezvous node replies

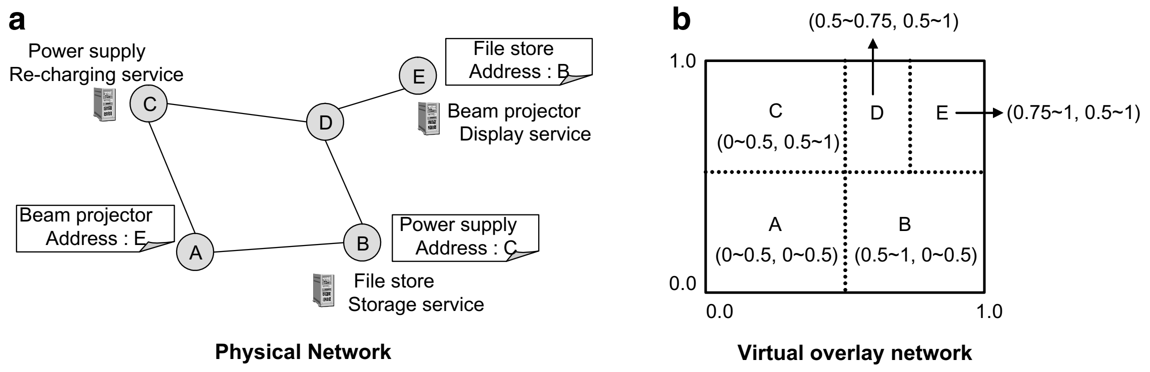

Fig. 1. Basic mechanism of DHT-based service discovery.

which nodes provide the service. For example, in the Fig. 1, if node C wants to know where a beam projector service is available, it has to send a service discovery query to the beam projector's rendezvous node. Because the beam projector's hash key is $\langle 0.2, 0.3 \rangle$, the node C learns that the beam projector's rendezvous node is node A and sends the query to the node A. Consequently, a service seeker also finds out where the service is available on the overlay network by requesting the service description to the service's rendezvous node.

As mentioned in Section 1, recent work [10] explains that it is difficult to adopt these DHT-based algorithms in wireless sensor environments. But, we make it possible to adopt our DHT-based algorithm in large-scale wireless sensor networks by suggesting the mechanism that constructs topology-aware overlay networks.

3.2. Topology-aware overlay network

In this section, we explain the mechanism constructing topology-aware overlay networks in large-scale wireless sensor networks. Our mechanism is based on the concept of clustering. The clustering is performed by binding up physically close nodes into a cluster. Physically close nodes become a cluster using only 1-hop neighbor information. Although a topology of nodes changes frequently, our clustering mechanism maintains the overlay network information to be rarely exchanged. In our mechanism, only cluster headers join overlay networks according to their physical locations. With this concept, our mechanism guarantees topology-aware overlay routing and minimizes the overhead of maintaining topology-aware overlay networks in large-scale wireless sensor networks. We first describe our clustering mechanism and then explain how to construct topology-aware overlay networks. Finally, we explain the revised service discovery protocol using our topology-aware overlay networks.

3.2.1. Clustering

The primary goal of our clustering mechanism is to make a topology of clusters change rare although a topology of nodes changes frequently. Nodes periodically broadcast beacon messages including their own ID and their cluster headers' IDs. They can discover neighbors and identify neighbors' cluster headers. If a node does not have any neighbor, it creates a new cluster and becomes a cluster header. Otherwise, it joins the nearest cluster using neighbors' cluster information. The cluster in which the most neighbors are included is the nearest cluster. Cluster headers limit the maximum number of member nodes, otherwise entire networks may become one cluster. If a node cannot join any cluster, it creates a new cluster. When a node leaves its cluster, it sends leave messages to the current cluster header and then joins a new cluster. There is a backup node in each cluster in case a cluster header leaves. The node with the least mobility and the most battery power is selected as a backup node. If a cluster header leaves its cluster, a backup node becomes a new header and floods header notification messages within its cluster. By this clustering mechanism, a topology of clusters changes rarely although a topology of nodes changes frequently.

3.2.2. Construction of topology-aware overlay networks

As shown in Fig. 2, cluster headers consist in overlay networks according to their physical locations. In order to reflect a physical topology, it is necessary that a new cluster header finds the closest cluster header and splits its key space. Boundary nodes in each cluster can discover neighbor clusters by neighbors' beacon messages. Member nodes in each cluster send header information of

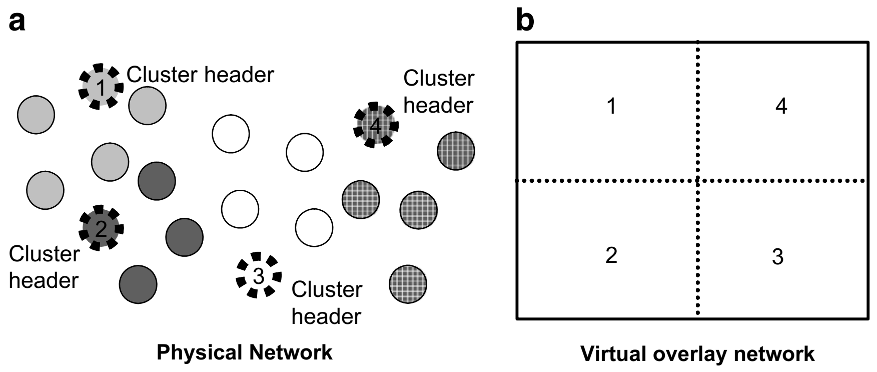

Fig. 2. Construction of topology-aware overlay networks.

neighbor clusters to their header nodes periodically. Using this information, a new cluster header can find the closest cluster. Let a set of cluster a's neighbor clusters $N(a) = c_1, c_2, \ldots, c_k$. Then the cluster that has $MAX_{i=1,\ldots,k}|N(a) \cap N(c_i)|$ is the closest cluster. If there are more than two cluster headers which have the same maximum number of jointed neighbor clusters (i.e. the values of $MAX_{i=1,\ldots,k}|N(a) \cap N(c_i)|$ are same), the closest cluster header is selected by counting hop distance from the current cluster header to the target cluster headers. The hop distance between cluster headers can be calculated by local flooding process during clustering. For the local flooding process, nodes at the cluster border exchange the information of hop count from their cluster headers. Then, the border node notifies the exchanged information of its cluster header. Consequently, the cluster header gets the hop distance up to the neighbor's cluster header by adding the notified hop count and the hop count to its border node. If even the hop distances up to the cluster headers also tie, the closet cluster header is randomly selected within the tied clusters.

After the closest cluster header is selected, it splits its key space into two equal-sized halves, vertically or horizontally. The newly divided key spaces are assigned according to physical locations. Neighbor headers of the closest header are used as reference nodes for a location-aware key space assignment. If key spaces are divided vertically, left or right neighbor headers are selected as reference nodes. Otherwise, upper or lower neighbor headers are selected as reference nodes. New headers can choose key spaces according to their physical locations by comparing hop distances to reference nodes with the closest header. If there are no reference nodes, new headers choose one of the divided key spaces randomly. But, this situation occurs only in the initial phase. Moreover this just makes overlay networks flip, which does not make the hop stretch increase.

In this manner, we can construct overlay networks which reflect a physical topology. Additionally, the overhead of maintaining overlay networks is minimized since the topology of clusters does not change frequently by our clustering mechanism. Finally, we can construct topology-aware overlay networks that guarantee topology-aware overlay routing and minimize the overhead of maintaining overlay networks.

3.2.3. Service discovery with our topology-aware overlay networks

In this section, we explain the revised service discovery protocol with our topology-aware overlay networks using cluster headers. Fig. 3a shows how to advertise services. The service provider sends its service description to its cluster header. The cluster header delivers the service description to the rendezvous node using overlay networks. The rendezvous node stores the service description in its service cache. Header nodes lying on the routing path also store the service description in their caches for faster service discovery. Fig. 3b shows how to lookup services. The service seeker sends a lookup message to its cluster header. The cluster header delivers the lookup messages to the rendezvous node using overlay networks. If the header node lying on the routing path has the desired service description, it immediately sends the service description to the service seeker. Otherwise, the rendezvous node sends the desired service description to the service seeker.

For load-balancing, a cluster header delegates its role to a backup node if its battery is not sufficient or its mobility is too high. In a cluster, the node with the least mobility and the most battery power is selected as a backup node. Eq. (1) shows the backup node selection algorithm within a cluster.

$$node_{backup} = \arg\max_i \left\{ \alpha \cdot \frac{ResidualPower_i}{ResidualPower_{MAX}} - (1-\alpha) \cdot \frac{Velocity_i}{Velocity_{MAX}} \right\} \quad (1)$$

In the equation, α $(0 \leqslant \alpha \leqslant 1)$ is the weighting factor of residual power and velocity components. The $ResidualPower_{MAX}$ and $Velocity_{MAX}$ are the maximum value of residual powers and velocities in the system, respectively. If the value of a backup node based on Eq. (1) is larger than the value of the cluster header or the cluster header leaves its cluster, a backup node becomes a new cluster header. When the cluster header cannot perform its duty any more, it must send its overlay information and service descriptions to the backup node.

4. Performance evaluation

We evaluated the performance of our DHT-based service discovery protocol using NS-2 [15] simulator. Table 1 shows the simulation parameters. In the simulation, we select cluster headers based on Eq. (1) and set α as 0.5 in default. We simply assumed that the power of a cluster header is reduced in proportion to a period of taking the role of the cluster header. Table 1 shows the simulation parameters. We evaluated the performance in terms of message overhead, lookup time, and lookup success ratio. We compared our protocol with a flooding-based protocol in which a service seeker floods a lookup message and then available service

Table 1
Simulation parameters

Total time	120 s
Number of nodes	50, 75, 100, 125, 150
Area	(1000 m × 1000 m) to (1700 m × 1700 m)
Transmission rage	250 m
Limited number of nodes in a cluster	11
Beacon interval	1 s
Cluster update interval	5 s
Request frequency per node	1.5 requests per minute
Mobility model	Random way-point with 3 s pause time
Node velocity	1 m/s–5 m/s
Routing protocol	DSR (Dynamic Source Routing) [14]

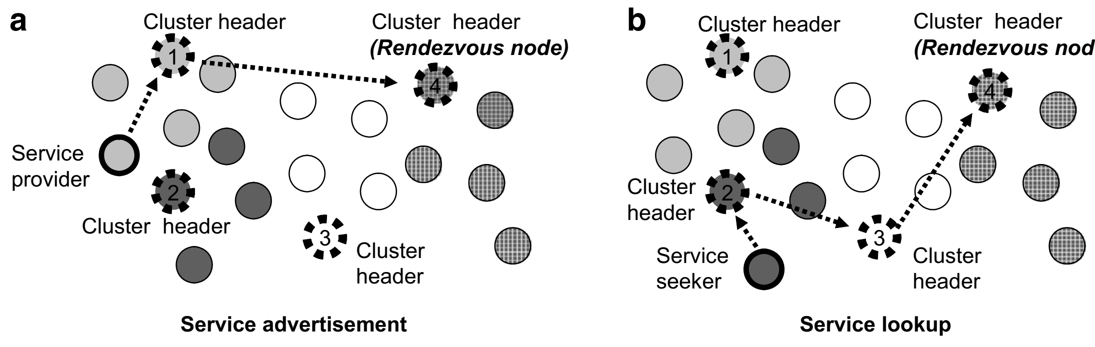

Fig. 3. Service advertisement and lookup using overlay networks.

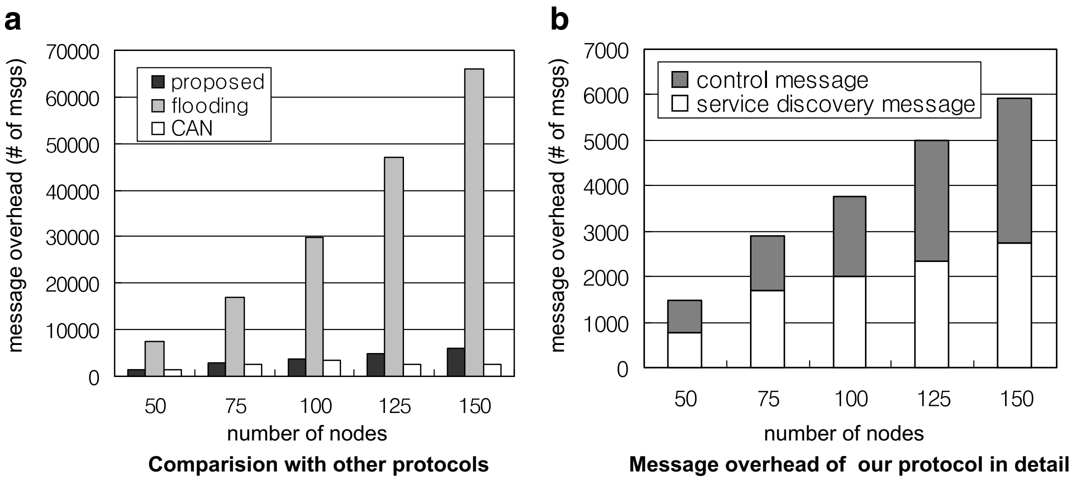

Fig. 4. Message overhead vs. number of nodes.

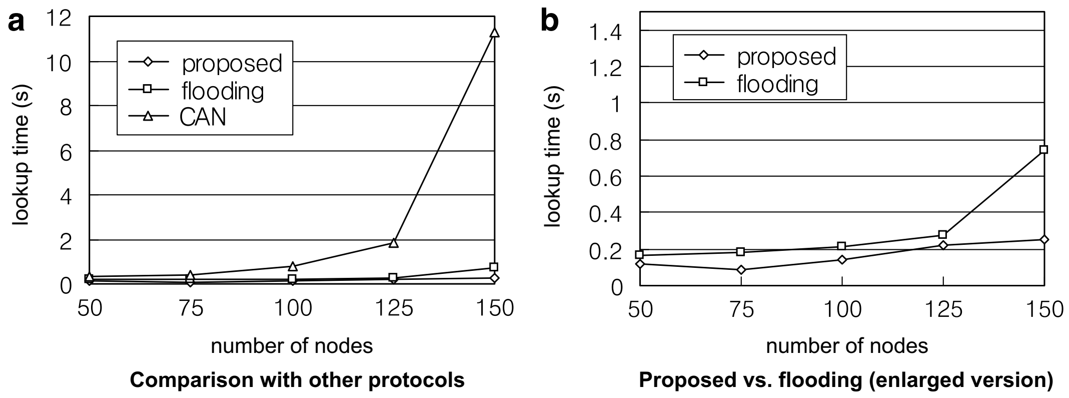

Fig. 5. Lookup time vs. number of nodes.

providers reply unicast messages to the service seeker. We also evaluated a CAN-based protocol, namely a DHT-based protocol which does not consider a physical topology.

4.1. Scalability

We set the number of nodes from 50 to 150 in order to evaluate scalability. To maintain the same density, we vary the size of the area according to the number of nodes. Node's velocity is set as 3 m/s.

Fig. 4a shows the message overheads counted by the number of logically generated messages on the overlay network. As shown in the result, our protocol generates much less messages than the flooding-based protocol. This is because our protocol uses unicast and local flooding only. We also observe that the message overhead of the flooding-based protocol increases more significantly than that of our protocol as the number of nodes increases. In fact, when there are N nodes, $O(N)$ messages per lookup are generated in the flooding-based protocol. But, our protocol generates $O(sqrt(N))$ messages per lookup since messages are unicasted in our protocol. Let the number of nodes per cluster be a constant k. Then, the number of clusters is N/k and the maximum number of physical hop count within a cluster is k. Therefore, on the overlay network, the length of physical hop count within a cluster is $O(k)$. Moreover, in our protocol, the length of physical hop count between neighbor cluster headers is $O(2k)$. This is because we reflect the physical network topology to the overlay network so that the hop stretch is maintained as a constant value. In the overlay network, the total number of physical hop count for service discovery is the product of the length of logical routing path among cluster headers and the length of physical hop count between neighbor cluster headers. According to [7], the logical routing path length on the overlay network is $O(sqrt(N/k))$. Consequently, the total number of physical hop count in our protocol is $O(sqrt(N/k))^{*} O(2k) = O(sqrt(N))$. The number of messages for lookup linearly increases in proportion to the total number of physical hop count. Thus, our protocol generates lookup messages up to

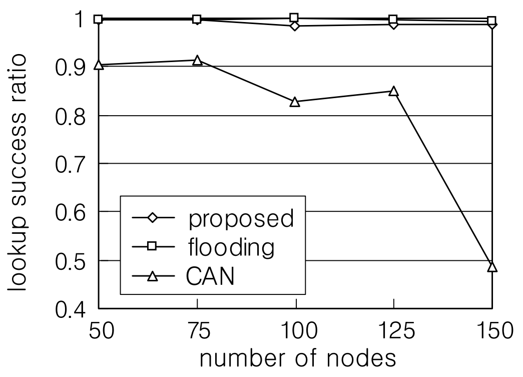

Fig. 6. Success ratio vs. number of nodes.

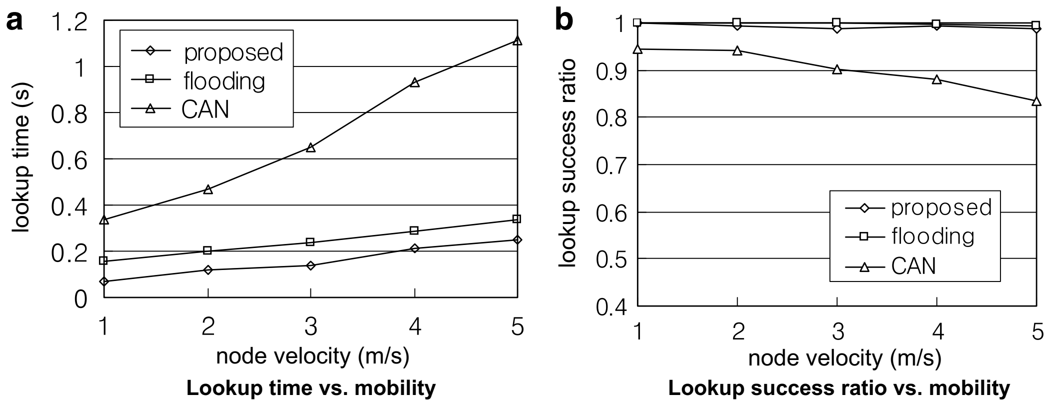

Fig. 7. Impact of mobility. (a) Lookup time vs. mobility (b) lookup success ratio vs. mobility.

$O(sqrt(N))$ and scales by $O(sqrt(N))$. In addition, the lookup time linearly increases in proportion to the generated lookup messages. Therefore, our protocol also has $O(sqrt(N))$ lookup time. As a result, we can say that our protocol is more scalable than the flooding-based protocol. As shown in the message overhead result, the CAN-based protocol generates fewer logical messages than our protocol. This is because our protocol generates control messages for clustering. However, in the view of the physical message overheads, the CAN-based protocol may generate large number of physical messages. This is because the CAN-based protocol cannot reflect dynamically changing physical topology and the average hop stretch between logical hops is generally larger than our protocol. Fig. 4b shows message overhead of our protocol in detail. In this figure, we observe that the growth rate of message overhead decreases as the number of nodes increases. This is because the number of messages generated for service discovery increases by $O(sqrt(N))$ although control messages increase almost linearly.

As shown in Fig. 5a, the CAN-based protocol takes much longer lookup time than the flooding-based protocol. In the CAN-based protocol, the hop stretch significantly increases because it does not consider any physical topology. Therefore, the CAN-based protocol is not suitable for large-scale wireless networks. At a glance, in Fig. 5a, our protocol and the flooding-based protocol seem to have almost the same performance. But, in Fig. 5b, we can find that our protocol shows shorter lookup time than that of flooding protocol. As the number of nodes increases, the lookup time of the flooding-based protocol increases a little until the saturation state ($N = 125$). But, after the saturation state is reached, the lookup time increases significantly. In our protocol, however, the lookup time increases almost linearly. Therefore, we can conclude that our protocol is scalable.

As shown in Fig. 6, the CAN-based protocol fails to lookup desired services frequently because the increased hop stretch generates too much traffic which results in high packet loss. In the flooding-based protocol, a success ratio is almost 100%. This is because nodes can receive messages from other nodes with high probability when messages are dropped. The success ratio of our protocol is comparable to that of the flooding-based protocol since our success ratio is over 98%.

4.2. Mobility

We now consider a network of 100 nodes over 1400 m × 1400 m area. We set the velocity of nodes from 1 to 5 m/s in order to evaluate the impact of node mobility.

The underlying routing protocol in our simulation is DSR [14]. The DSR takes more routing time as mobility increases. Fig. 7a shows that the lookup time in all protocols increases in proportion as mobility increases. DSR routes one message per lookup in the flooding-based protocol. However, it routes as many messages as the number of logical hops per lookup in the CAN-based protocol. Thus, the growth rate of the CAN-based protocol is higher than that of the flooding-based protocol. If our overlay networks cannot reflect a physical topology when mobility is high, the lookup time of our protocol will be close to that of the CAN-based protocol. However, as shown in Fig. 7a, the growth rate of our protocol is almost the same as that of the flooding-based protocol. Therefore, we can infer that our overlay networks reflect a physical topology very well regardless of node mobility. Fig. 7b shows the success ratio according to the mobility. The result is almost the same as Fig. 6. Consequently, we can conclude that the success ratio of our protocol is not sensitively affected by the node mobility.

5. Conclusions

In ubiquitous computing environments, numerous computing devices will be widely distributed and connected by large-scale wireless sensor networks. But, existing service discovery protocols are inappropriate for such environments since they are not scalable.

In this paper, we have proposed the DHT-based service discovery protocol including the mechanism that constructs topology-aware overlay networks in large-scale wireless sensor networks. Our protocol is scalable since it does not require a lookup server and does not rely on multicast or flooding. Through simulations, we have shown that our protocol is scalable and outperforms traditional flooding-based protocols in terms of message overhead and lookup time.

References

[1] K. Arnold, B. Osullivan, R.W. Scheifler, J. Waldo, A. Wollrath, The Jini Specification (The Jini Technology), Addison-Wesley, Reading, MA, 1999.
[2] Universal Description Discovery and Integration Platform. Available from: <http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf>, September 2000.
[3] The Salutation Consortium Inc., Salutation Architecture Specification Part 1, Version 2.1 Edition. Available from: <http://www.salutation.org>, 1999.
[4] E. Guttman, C. Perkins, J. Veizades, RFC 2165: Service Location Protocol, June, 1997.
[5] R. John, UPnP, Jini and Salutaion - A look at some popular coordination frameworks for future network devices, technical report, California Software Labs. Available from: <http://www.cswl.com/whiteppr/tech/upnp.html>, 1999.
[6] S. Helal, N. Desai, C. Lee, Konark-A Service Discovery and Delivery Protocol for Ad-Hoc Networks, in: Proc. Third IEEE Conf. Wireless Comm. Networks (WCNC), March, 2003, pp. 2107–2113.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proc. ACM SIGCOMM 2001 Conf., San Diego, CA, USA, 2001, pp. 161–172.
[8] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, in: Proc. ACM SIGCOMM Conf. 2001, San Diego, CA, USA, 2001, pp. 149–160.
[9] F. Sailhan, V. Issarny, Scalable Service Discovery for MANET, in: Proc. 3rd IEEE Con. Pervasive Computing and Communications (PERCOM), Washington, DC, USA, 2005, pp. 235–244.
[10] D. Chakraborty, A. Joshi, Y. Yesha, T. Finin, Toward distributed service discovery in pervasive computing environments, IEEE Transactions on Mobile Computing 5 (2) (2006) 97–112.
[11] C. Cramer, T. Fuhrmann, Proximity Neighbor Selection for a DHT in Wireless Multi-Hop Networks, in: Proc. 5th IEEE Con. Peer-to-Peer Computing (P2P), New Orleans, LA, USA, March 2003, pp. 1143–1148.
[12] R. Winter, T. Zahn, J. Schiller, DynaMO: a topoloy-aware P2P overlay network for dynamic, mobile ad-hoc environments, Kluwer Telecommunication System Journal 27 (2–4) (2004) 321–345.
[13] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, Topologically-aware overlay construction and server selection, in: Proc. 21th IEEE Conf. INFOCOM, New York, NY, June 2002, pp. 1190–1199.
[14] D.B. Johnson, D.A. Maltz, J. Broch, DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks, in: C.E. Perkins (Ed.), Ad Hoc Networking, Addison-Wesley, Reading, MA, 2001, pp. 139–172.
[15] The network simulator NS-2, Available from: <http://www.isi.edu/nsnam/ns>.