

Energy-Efficient Cluster-based Service Discovery for Ubiquitous Computing

Gregor Schiele, Christian Becker and Kurt Rothermel
Institute for Parallel and Distributed Systems (IPVS)
Universität Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
{gregor.schiele, christian.becker, kurt.rothermel}@informatik.uni-stuttgart.de

Abstract:

Service discovery in Ubiquitous Computing is a task that has to be done frequently due to dynamically changing environments. The limited battery power of mobile devices requires us to optimize frequent and energy costly tasks, especially the ones incurring in communication activities. In this paper we present a novel service discovery algorithm based on node clustering. Nodes within a cluster may sleep to save energy when idle. A cluster head node is always active and answers discovery requests on behalf of other nodes to achieve low discovery latencies. Simulation experiments show energy savings of up to 66% compared to an approach where all nodes are permanently active while the discovery latencies were not increased.

1 Introduction

In peer-based Ubiquitous Computing (UC) systems [3], a multitude of mobile, specialized devices (e.g. sensors or smart pens) are interconnected dynamically in a mobile ad hoc network (MANET) using wireless communication technology, e.g. IEEE 802.11 [9]. No central infrastructure services are presumed. Instead, applications make use of the functionality of devices available in their current environment. To allow this, a prompt and efficient dynamic service discovery system (SDS) is required. Service discovery is conducted frequently, both at the start of an application and whenever the environment changes, e.g. to detect a new and better service that has just become available. Because many of the participating devices are battery powered, energy-efficiency is a must for such a frequently performed task.

With one exception [12], current service discovery protocols (e.g. [14][10][1] [7]) do not address energy-efficiency. In this paper, we propose a new approach for an energy-efficient SDS that is specifically suitable for peer-based UC. Our approach is based on the observation that, although devices are mobile, we can often identify sub-groups of devices with similar mobility patterns in typical UC scenarios, e.g. devices carried by a user in its personal area network (PAN) or devices of two users in a meeting room. This effect – named *group mobility* – has been discussed by a number of authors (e.g. [8][15]). To exploit group mobility for service discovery, we propose to form clusters out of nodes with similar mobility patterns. For each cluster, one designated node stays awake and answers discovery requests. All other nodes in the cluster may sleep in order to save energy. This provides small discovery latencies and allows for high energy savings. Similar approaches can be found in other contexts, like energy-efficient routing in MANETs (e.g. [17][16][4]). However, node clustering for energy-efficient service

discovery has – to the best of our knowledge – not been proposed before.

The remainder of this paper is structured as follows. First, we define our system model and derive requirements. After that, we discuss related work, present our initial approach and evaluate it briefly. We conclude the paper with a summary and a discussion of our future work.

2 System Model

Our system comprises a set of networked devices, or *nodes*. All nodes are integrated into a wireless MANET with multi-hop routing ability. We assume communication technology with symmetric communication ranges and bandwidth, omnidirectional antennas and the ability to send acknowledged multi-hop unicast messages and unacknowledged one-hop local broadcast and multi-hop multicast messages. The device mobility may lead to frequent and unpredictable network partitions.

Nodes are typically resource-constrained and possibly embedded in everyday objects. They can be stationary, e.g. sensors or info stations, or mobile, such as PDAs. Nodes are powered by a battery with limited capacity. They offer two modes of operation: *sleep* and *awake*. While in sleep mode, we assume that nodes cannot communicate or perform computations. We also assume that, the state of a sleeping node is preserved, i.e. the node can continue its operation after wakeup. Mode transitions demand a certain amount of energy and induce a transition latency. Transition from sleep to awake mode is initiated by a node-internal timer, e.g. a watchdog timer. We assume no special hardware for waking up sleeping nodes remotely, e.g. by sending a signal [13]. Node clocks do not have to be synchronized. However, we assume a bounded clock drift to be able to specify a node's remaining sleep time towards other nodes.

Nodes offer services to local or remote clients like a navigation application or a room control application. Services encapsulate a distinct functionality and offer a well-defined interface to access it. Example services are a graphical output service or a data storage service.

3 Requirements

Based on the system model, we can derive three specific requirements for energy efficient service discovery in ubiquitous computing environments:

A) Energy-efficiency: In order to support battery-operated devices, a SDS must be energy-efficient. Energy-efficiency is characterised by the amount of time that idle nodes spend in sleep mode. The ideal case is that all nodes spend all idle time sleeping, maximizing the amount of energy saved

compared with staying awake while being idle. A variety of analyses of wireless communication technology have shown that the energy cost for sending, receiving, and idle mode dominate the cost for the sleep mode by an order of magnitude [13][4][5].

B) Discovery latency: Applications in UC aim for prompt discovery of new devices and their services, to plan and prepare for their usage, e.g. by extracting the state of a used service to switch to a new one. Therefore, a SDS should try to minimise the discovery latency for new devices and services.

C) Decentralised operation: To be usable in the highly dynamic networks described in Chapter 2, a SDS must work despite frequent and unpredictable network partitions. Therefore, it cannot rely on centralised components and must operate in a completely decentralised manner.

Requirements (A) and (B) are in conflict and a concrete SDS has to provide a trade-off between them.

4 Related Work

Although a number of service discovery approaches have been proposed, none of them is able to fulfil all of the requirements stated above.

Existing Service Discovery Approaches

There are two main classes of service discovery approaches: *mediator-based discovery approaches* and *direct or peer-based discovery approaches*.

In **mediator-based approaches**, service discovery is performed using one or multiple mediators, called lookup services (LUS). Services are registered at the LUS. To discover a service, a client requests a desired service at a LUS. Alternatively, the LUS can proactively push service descriptions to clients. Existing mediator-based approaches like Jini [14] or the Intentional Naming System [1] cannot provide infrastructure-less operation and do not support sleeping service provider nodes. The Service Location Protocol [7] allows optional infrastructure-less discovery in case no LUS is found but does not address energy-efficiency.

Direct or peer-based approaches like Universal Plug and Play (UPnP) [10] or DEAPSpace [12] do not rely on the presence of a mediator. In UPnP, to find a service, a client multicasts a discovery request. Nodes offering a suitable service reply with a service description. Alternatively, service providers can proactively multicast descriptions of their services to all potential clients. Communication costs are high due to repeated multicasting – particularly in multi-hop environments. In addition, to receive requests or announcements, nodes must stay awake permanently.

IBM's DEAPSpace project [12] provides a service discovery protocol for single-hop MANETs that allows nodes to sleep temporarily. Each node maintains a global view of all services offered in its one-hop environment. It announces this view regularly to all neighbours. If a node receives an announcement that includes all its services, the node omits its own announcement for some time; if one or more of its services are missing, it sends the announcement. Sending announcements is synchronized to a common time window for all nodes, allowing them to sleep in between. Still, to enable new nodes to join in, nodes have to stay awake for more than 50% of their time. We want to allow much longer sleep periods. Also, while DEAPSpace provides a simple and

efficient way for service discovery in single-hop environments, a global view of services is unfeasible for multi-hop environments as they are addressed in our work.

Energy-efficient Communication Protocols

Our approach is based on clustering nodes, electing a node to perform service discovery for the cluster and putting the other nodes into sleep mode whenever possible. Related approaches have been proposed for energy-efficient communication, both on the MAC (e.g. [9][18]) and the routing layer (e.g. [17][16][4]). Here, the network card is frequently switched between sleep and idle mode to save energy at the cost of a higher communication delay. As an example, the power save mode (PSM) of IEEE 802.11 switches modes up to 10 times per second. We aim at much higher sleep durations of multiple seconds or more.

Approaches that work on the routing layer (e.g. [17][16][4]) use dynamic node clustering to create a routing backbone. All nodes in the backbone have to stay awake to forward messages, while all other nodes can sleep. If a backbone node receives a message for a sleeping node in its neighbourhood, it buffers the message until the node awakes. In contrast to this, our approach allows to answer messages to find a service instantly.

Our energy-efficient service discovery builds upon these approaches to provide energy-efficient communication. Using clustering on the application layer allows us to incorporate knowledge about service usage that is not available on the MAC or routing layer.

5 Our Approach: SANDMAN

The overall objective of our approach, called *service awareness and discovery for mobile ad hoc networks* (SANDMAN) is to provide a service discovery system that fulfils the requirements given in Chapter 3: (A) energy-efficiency by letting nodes sleep, (B) minimised discovery latency and (C) decentralised operation.

Overview

The overall approach of SANDMAN is to organise the MANET dynamically into node clusters. Each cluster consists of one cluster head (CH) and an arbitrary number of clustered nodes (CNs). Both, CH and CNs can offer and use services. In each cluster, the CH acts as a representative for its CNs in terms of service discovery.

CNs periodically sleep, if they are idle, in order to save energy. After waking up, a CN waits for incoming client requests for the duration of a timeout interval. If no requests are received, it informs the CH and returns to its sleep mode. This way, CNs that are not used continuously can save a substantial amount of energy (requirement A).

To provide small discovery latencies (requirement B) despite the sleeping CNs, the CH stays awake all the time. It acts as a local LUS on behalf of services running on CNs in its cluster. Therefore, the CH can instantly answer any client's discovery request with a suitable list of service descriptions and wake-up times of the respective CNs. Letting the CH stay awake constantly is the trade-off taken in SANDMAN between energy-saving and discovery latencies. The CH is re-elected periodically to avoid draining a single node's battery.

Note that after a client has discovered a service, it may not be usable instantly if the CN providing it is currently sleeping. Therefore, depending on the length of a node's sleep period,

clients using this node's services will experience corresponding interaction latencies. We discuss interaction latencies in more detail in the evaluation section.

Clusters are formed according to connectivity and node mobility. Connected nodes are clustered, if they have similar mobility patterns, i.e. are members of the same mobility group. This leads to much more stable clusters and allows nodes to sleep longer. Typical durations of such clusters are expected to be in the range of minutes to hours. Examples for clusters are devices carried by one user and forming a PAN or devices of multiple users sitting in a meeting room together. Although the whole system operates decentralized (requirement C), a smart environment could also be integrated in a cluster as a special device with no resource limitation and offering a multitude of services.

Putting nodes into sleep mode on the application layer provides us with additional knowledge about the types and characteristics of the communicating services and clients. As an example, a node's sleep times can be chosen according to the services offered by it, e.g. depending on the number of similar services in the vicinity or the maximum acceptable usage delay for its services. This can lead to longer sleep times without inducing unacceptable latencies. If the sleep times are long enough to justify the additional overhead, we can even shutdown devices completely, saving even more energy.

In the remainder of this section, we derive an initial simple realisation of the approach presented above. We describe the interaction between services and CH, clients and CH for service lookup, and briefly discuss cluster management.

Service/Cluster Head Interaction

Figure 2 depicts the overall interaction in SANDMAN. We assume that an appropriate CH has already been selected.

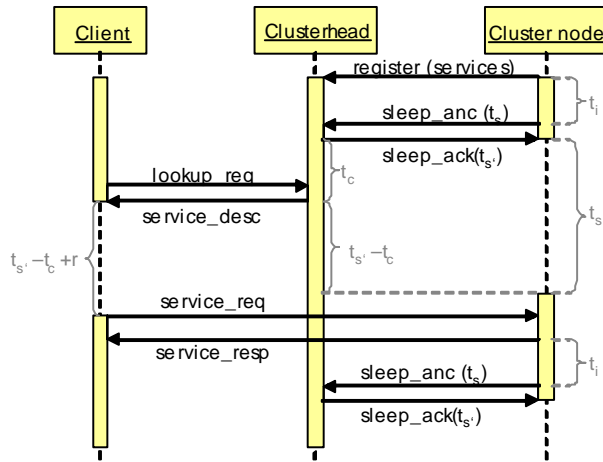


Figure 2: SANDMAN overview

When a node enters a cluster, i.e. when it becomes a CN, it registers its services at the CH (*register*). After that, the CN starts an idle timer t_i . Each time a client uses one of its services, the CN restarts the idle timer. If a timeout occurs, i.e. the CN has not been used by any client for t_i , the CN sends a message (*sleep_anc*) to the CH. This message tells the CH, that the CN wants to enter its sleep mode for the duration t_s (*sleeping time*). The CH acknowledges this message (*sleep_ack*) and sends back a sleeping time t_s' . This can be either the unchanged t_s or a modified value. After

receiving the acknowledgement from the CH, the CN enters its sleep mode for the duration t_s' . When it awakes again, it restarts its idle timer t_i and starts listening for client requests again.

This simple protocol allows us to use individual sleep cycles per CN, instead of e.g. one common cycle per cluster. The additional flexibility can be used, e.g. to evenly distribute CN awake times or choose service specific sleep times.

Client/Cluster Head Interaction

If a client searches a service, it sends a lookup request (*lookup_req*) to one or more CHs. These can be discovered by single or multi hop broadcasts on the underlying network, e.g. using a flooding technique. Upon reception of a *lookup_req*, each CH creates a list of service descriptions (*service_desc*) for all suitable services in its cluster. Each description in the list includes the delay until the service can be contacted by the client ($t_s - t_c$). This list is sent back to the client. Thus, clients can select the service suited best for their goals, e.g. the service that awakes first. After selecting a service, the client waits for the specified time to contact the service. During this time, a client can sleep in order to save energy. Note that clients can use a randomized backoff delay (r) before contacting the service to avoid collisions with other clients.

Cluster Management

So far, we have assumed that a cluster has been established and a CH has been elected. SANDMAN is independent of the specific cluster management protocol used to achieve this. We are currently analysing existing approaches (e.g. [6][16][2]) and their impact on the performance of SANDMAN.

6 Performance Evaluation

SANDMAN is designed to preserve energy, offer a low discovery latency, and – as a trade-off – accept longer service interaction latencies. In order to evaluate SANDMAN, we conducted a number of simulation experiments using the network simulator ns-2 [11]. The presented experiments are designed to provide an initial assessment. Our goal is to measure the energy consumption and the discovery and interaction latencies. We expect the chosen parameter values to result in high energy savings and are interested in the resulting latencies.

In our experiments, we omit cluster management. The experiments were conducted using a fixed cluster with one node acting as CH and the others as CNs, each offering one service. We varied the number of CNs between 1, 2 and 4 and the number of clients between 1 and 30. All experiments were repeated 20 times.

Clients repeatedly discover a randomly selected service, wait until it is awake and perform a single request/response interaction to use it. Each client then waits for a time randomly chosen from a time interval $[0s; 8s]$. After that, the client starts discovering a service again.

For simplicity, we omit state transition costs and delays and measure energy consumption for the network card only. We use values for an IEEE 802.11b network card, the ORiNOCO PC GOLD card, taken from [13]. Further parameter values for the simulation are shown in Table 1.

In order to compare our approach with non-cluster based approaches we simulated a peer-based approach resembling UPnP [10], without energy saving (called *P2P* in the

following). All nodes stay constantly awake. Clients broadcast service requests, suitable services answer and can be used instantly. Because the results for different number of nodes offering services are nearly identical for P2P, we only present the results for 2 nodes.

idle timeout t_i	1 s
sleep time t_s	9 s
idle consumption P_{idle}	805 mW
sleep consumption P_{sleep}	60 mW
send consumption P_{send}	1400 mW
receive consumption $P_{receive}$	950 mW
bandwidth (unicast)	11 Mbps
bandwidth (broadcast)	1 Mbps
lookup_req message length	1024 bytes
service_desc message length	5000 bytes
service_req message length	20000 bytes
service_resp message length	20000 bytes
sleep_ack message length	100 bytes
randomized backoff delay r	0 s

Table 1: Simulation parameter values

The average power consumptions of the clustered nodes (CNs and CH) in SANDMAN are shown in Figure 2. As can be seen, even for only one CN, SANDMAN is able to save a considerable amount of power for few clients, up to 40 % for one client. This is because the CN is able to sleep about 89% of the simulation time in this case¹. As nodes in P2P do not sleep, their power consumption is always above the idle consumption, even for cases with very little work.

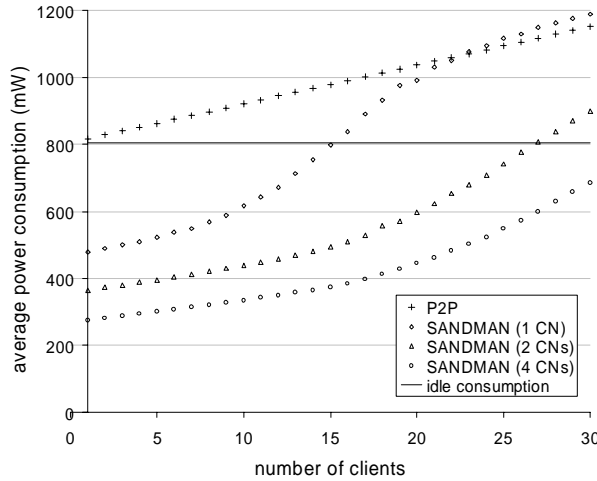


Figure 2: Average power consumption

Above 22 clients, SANDMAN has a slightly higher power consumption than P2P. This is because now the CN is awake most of the time in order to handle client requests (90% of the time for 22 clients, 98% for 30). For larger clusters with 2 or 4 CNs, the average power consumption when using SANDMAN is even lower. This is because the overhead induced by the CH is distributed among more CNs and because client requests are distributed among all CNs due to our workload model.

As already discussed, sleeping service providers introduce high interaction delays. Figure 3 shows the interaction delays for SANDMAN (with 1, 2 and 4 CNs) and P2P in our experiment.

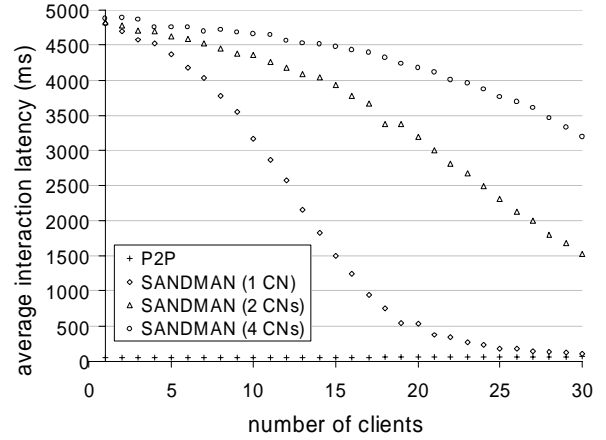


Figure 3: Average interaction latency

P2P adds no additional delay over communication and therefore has a more or less constant delay. As expected, delay in SANDMAN is dominated by the time clients wait for sleeping service providers. For few clients and a therefore frequently sleeping service provider, clients have to wait before each interaction. For more clients, the service provider stays awake longer and the probability to use it while it is still awake increases. For a cluster size of one CN, at about 25 clients the interaction delay of SANDMAN starts resembling that of P2P. For larger clusters, this point is not reached in the given simulations, again because client requests are distributed amongst CNs. The discovery delay is the same in all simulation scenarios, since the CH is always awake and answers lookup requests immediately.

Because of the sleep durations of the CNs and the chosen workload model, SANDMAN processes fewer interactions, e.g. only 45% for the scenario with one CN and one client, than P2P. Note that the workload model chosen in our evaluation is a worst-case scenario in terms of interaction delay, as each interaction is preceded with a discovery lookup and followed by a wait time long enough for the service provider to enter its sleep mode if not massively used by other clients. In most cases, we expect clients to exchange multiple messages with a service in fast succession (i.e. with less delay than t_i). This would lead to much lower interaction latencies even for few clients. Some initial measurements for 10 and 20 interactions per discovery showed interaction latencies of less than 500 ms, even for only one client. Thus, the experiments shown here give an upper bound for the interaction latency.

7 Conclusion and Future Work

We have presented SANDMAN, a protocol for energy-efficient service discovery in UC environments. SANDMAN uses a node clustering approach to identify a number of nodes – the cluster heads – that stay awake permanently and answer discovery requests on behalf of the nodes in their clusters. This allows the nodes in the cluster to maximize their sleep times. The discovery delays are unaffected, as the CH can answer every discovery request instantly. SANDMAN is

¹ For $t_s = 9s$ and $t_i = 1s$ the theoretical maximum is 90%.

readily applicable with today's hardware. Our simulations show high energy savings for scenarios with large idle times. Still, our work with SANDMAN is at its beginning.

A more thorough evaluation of SANDMAN, including transition costs and latencies as well as cluster management is required.

We are going to investigate further improvements of SANDMAN with respect to the scheduling of client requests and using the CH as proxy for client/service interaction.

The scheduling of the sleep times t_s and the idle timeout t_i is important with respect to the energy consumption and latency imposed to clients until their request is served. Since minimizing energy consumption and minimizing the interaction latency are conflicting goals, careful adjustment of t_i and t_s are required. The current implementation of SANDMAN uses a simple approach, in which t_i and t_s are statically chosen by the service developer. In the future, we plan to add strategies to dynamically adapt these parameters, e.g. based on cluster stability or service popularity.

Currently, CNs detect that they can go to sleep by waiting for client requests for the idle timeout t_i . Therefore, each CN has to stay awake for at least t_i in each cycle, even if no client requests its services. If the CH mediates the requests from clients, t_i can be interleaved with the CNs sleeping time and unused CNs can be send back to sleep mode by the CH immediately. In addition, the CH can provide means for load balancing within the cluster.

Forcing a single node to stay in the role of a CH would drain its battery quickly. In order to improve fairness the CH may be re-elected regularly. We are currently working on different enhancements of our protocol, e.g. to enable a hand-over of CNs between CHs.

Further, we want to examine data management algorithms for the cooperation between multiple CHs, e.g. using hierarchical approaches, caching and replication mechanisms.

Finally, we are going to analyse the dependencies between our approach and cluster-based approaches on lower layers, e.g. cluster-based routing in mobile ad hoc networks, to further reduce communication-based energy-consumption.

8 Acknowledgements

The authors would like to thank Karsten Angstmann, Joerg Haehner, Arno Wacker and Marcus Handte for their valuable comments.

9 References

- [1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley: *The design and implementation of an intentional naming system*. In Proc. of the 17th ACM Symposium on Operating Systems Principles (SOSP '99), Kiawah Island, SC, USA, December 1999.
- [2] P. Basu, N. Khan, T. D. C. Little: *A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks*. In Proceedings of the 21st Int'l Conference on Distributed Computing Systems Workshops (ICDCSW '01), Phoenix (Mesa), AZ, USA, April 2001.
- [3] C. Becker and G. Schiele: *Middleware and Application Adaptation Requirements and their Support in Pervasive Computing*. In Proc. of the 3rd Int'l Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES) at ICDCS 2003, Providence, RI, USA, May 2003.

- [4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris: *Span: an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks*. ACM Wireless Networks Journal, vol. 8, no. 5, September 2002.
- [5] L. M. Feeney, M. Nilsson: *Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment*. In Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Anchorage, AK, USA, April 2001.
- [6] M. Gerla and J.T.-C. Tsai: *Multicluster, mobile, multimedia radio network*. ACM/Baltzer Journal of Wireless Networks, vol. 1, no. 3, 1995.
- [7] E. Guttman, C. Perkins, J. Veizades, and M. Day: *Service Location Protocol*, Version 2. IETF, RFC2608, June 1999.
- [8] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang: *A Group Mobility Model for Ad Hoc Wireless Networks*. In Proc. of the 2nd ACM Int'l Workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWiM'99), Seattle, WA, USA, August 1999.
- [9] IEEE 802.11 Standard: *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.
- [10] B. A. Miller, T. Nixon, C. Tai, and M. D. Wood: *Home Networking with Universal Plug and Play*. IEEE Communications Magazine, vol. 39, no. 12, December 2001.
- [11] *The Network Simulator ns-2 Homepage*. <http://www.isi.edu/nsnam/ns/>
- [12] M. Nidd: *Service Discovery in DEAPspace*. IEEE Personal Communications, vol. 8, no. 4, August 2001.
- [13] E. Shih, P. Bahl, and M. Sinclair: *Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices*. In Proc. of the 7th ACM SIGMOBILE Annual Int'l Conference on Mobile Computing and Networking (MobiCom), Atlanta, GA, USA, September 2002.
- [14] Sun Microsystems: *Jini™ Technology Core Platform Specification, Version 1.2*. December 2001.
- [15] K. H. Wang and B. Li: *Group Mobility and Partition Prediction in Wireless Ad-Hoc Networks*. In Proc. of the IEEE Int'l Conference on Communications (ICC), New York, NY, USA, April 2002.
- [16] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin: *Topology Control Protocols to Conserve Energy in Wireless Ad Hoc Networks*. Technical Report 6, University of California, Los Angeles, Center for Embedded Networked Computing, January 2003.
- [17] Y. Xu, J. Heidemann, D. Estrin: *Geography-informed energy conservation for Ad Hoc routing*. In Proc. of the 7th ACM SIGMOBILE Annual Int'l Conference on Mobile Computing and Networking (MobiCom), Rome, Italy, July 2001.
- [18] W. Ye, J. Heidemann, and D. Estrin: *An Energy-Efficient MAC protocol for Wireless Sensor Networks*. In Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY, USA, June 2002.