uCast: Unified Connectionless Multicast for Energy Efficient Content Distribution in Sensor Networks

Qing Cao, *Student Member*, *IEEE*, Tian He, *Member*, *IEEE*, and Tarek Abdelzaher, *Member*, *IEEE*

Abstract—In this paper, we present uCast, a novel multicast protocol for energy efficient content distribution in sensor networks. We design uCast to support a large number of multicast sessions, especially when the number of destinations in a session is small. In uCast, we do not keep any state information relevant to ongoing multicast deliveries at intermediate nodes. Rather, we directly encode the multicast information in the packet headers and parse these headers at intermediate nodes using a scoreboard algorithm proposed in this paper. We demonstrate that 1) uCast is powerful enough to support multiple addressing and unicast routing schemes and 2) uCast is robust, efficient, and scalable in the face of changes in network topology, such as those introduced by energy conservation protocols. We systematically evaluate the performance of uCast through simulations, compare it with other state-of-the-art protocols, and collect preliminary data from a running system based on the Berkeley motes platform.

Index Terms—Sensor networks, multicast, content distribution.

1 Introduction

Recent work has articulated the unique challenges of sensor networks that stem from their resource limitations. In this paper, we study the implications of such limitations on multicast protocols. In particular, we address the problem of designing protocols to support a large number of small multicast groups, where the number of destinations in a single session is limited.

There are many applications of small-group multicast in sensor networks. For example, in a typical directory service, such as the protocol described in [16], each node periodically updates a small set of other nodes (named *directory servers* in [16]) with its location. Therefore, one node needs to multicast information to several destination nodes, which form a small group. Furthermore, when multiple nodes use the directory service, they will generate many small-group multicast sessions. Another common example involves data-centric storage (DCS) [21], [24]. One key component of some DCS protocols is the use of Geographic Hash Tables (GHT). GHT hashes keys, usually the names of data or events, into geographic coordinates. It then stores the values at the node that is geographically nearest to the hash value of the key. Usually, the data storage protocol suggests that the key-value pairs should be stored at multiple locations for robustness. Therefore, such protocols naturally require a small-group multicast session for each storage operation, and many multicast sessions for storing a large amount of data.

Providing a small-group multicast service in sensor networks is complicated by several unique challenges. One challenge is that sensor nodes are extremely energy-constrained. Consequently, sensor networks generally employ energy conservation protocols, which usually allow individual nodes to switch between sleep and wake states. Therefore, the topology of sensor networks changes dynamically at a high rate, which poses unique requirements on the multicast service. Any protocol that relies on certain multicast structures to keep state information, such as multicast tree routing tables, must adapt these structures to topology changes. This performance issue has been largely overlooked in the design of current multicast protocols.

Another challenge in multicast design is that there are many unicast routing protocols to interface with in sensor networks. In fact, there is no consensus on which one is the best. The choice of unicast protocol usually depends on the particular application. For example, when geographic location information is readily available for each node, several well-known routing protocols that take advantage of geographic information are appropriate [4], [12], [14]. On the other hand, when there is no geographic information available, protocols based on certain topology encodings are preferred [5], [20], [18]. Because of the wide range of unicast choices, it is undesirable to design multicast protocols that make assumptions regarding the particular unicast service since that will limit the applicability of the multicast. On the other hand, it is also not useful for the multicast to provide a routing service from scratch since doing so will lead to considerable functional overlap with unicast protocols.

To address these two challenges, we present a multicast protocol that is both general (supports multiple unicast protocols by using a *unified* interface) and robust (tolerant to

- *Q. Cao and T. Abdelzaher are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, IL 61801. E-mail: {qcao2, zaher}@cs.uiuc.edu.*
- *T. He is with the Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455. E-mail: tianhe@cs.umn.edu.*

Manuscript received 6 Apr. 2005; revised 9 Oct. 2005; accepted 3 Jan. 2006; published online 27 Dec. 2006.
Recommended for acceptance by G. Lee.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0227-0405.

topological changes by providing a *connectionless* service). We call this protocol **unified connectionless multicast**, or uCast. To the best of our knowledge, uCast is the first protocol specifically optimized for small-group multicast sessions in sensor networks. We now give a more detailed explanation of the two features of uCast.

The first feature is that uCast can support multiple unicast routing protocols through a unified interface. In this sense, uCast is a modular extension to the underlying unicast layer. In fact, it can extend any unicast routing protocol as long as this unicast can export a common comparison interface, which allows a comparison operation between two next-hop nodes for the same destination to determine which one is better (in terms of some notion of *cost*). We implemented uCast on top of three unicast routing protocols with different addressing schemes to prove our point.

The second feature is that uCast is tolerant to topology changes caused by energy saving protocols. To achieve this, uCast does not keep any multicast-specific state at intermediate nodes. Instead, uCast dynamically decides the multicast delivery path at each intermediate node based only on local topology information and the comparison interface as discussed earlier. Since local information is much easier to reconstruct upon topological changes than a superimposed global multicast overlay, uCast is much more adaptable to unpredictable changes in network connectivity than previous multicast protocols.

The rest of the paper is organized as follows: We outline related work in Section 2. We discuss the details of uCast in Section 3. In Sections 4 and 5, we report simulations-based and real platform-based results. At last, we provide further discussions and conclusions in Section 6.

2 Related Work

2.1 Multicast

Multicast is a classical topic in networking. Interestingly, we find only a few multicast protocols designed for sensor networks. Multicast protocols developed for ad-hoc networks and for the Internet cannot be easily applied in the sensor networks domain. In the following, we survey three different types of multicast, namely, multicast protocols for sensor networks, ad-hoc networks, and the Internet.

One category of sensor network multicast is called geocast. It considers the scenario where multicast destinations are located within a bounded geographical area. Representative work includes [13], [17]. Another multicast category is called spatiotemporal multicast or mobicast [10]. Mobicast features a moving zone of multicast destinations. The goal is to deliver packets just in time to this zone for tracking purposes. Another category [1] studies multicast for data caching and placement. It focuses on using multicast trees for asynchronously updated data deliveries. Yet another is called TTDD [30], which is optimized for mobile sinks. It uses a grid structure, coupled with localized flooding to track mobile sinks. These protocols do not consider the effect of topology changes introduced by energy conservation protocols nor are they designed to handle the small-group multicast scenarios. Further, none of these protocols takes into account the compatibility issue with unicast protocols. Therefore, they are usually implemented in isolation from the unicast routing protocols that are already available and often provide unicast as a special case [13], [17]. This redundancy is not desirable. Since the memory size of current sensor network nodes is extremely limited ($4K$ bytes on Mica2/MicaZ), it is not useful to have functional redundancy between different routing services. From this viewpoint, these previous multicast protocols are different from uCast in their lack of compatibility with multiple unicast routing protocols. Therefore, we do not compare uCast with them in this paper.

Many multicast protocols are developed for ad hoc networks. Representative approaches include multicast-tree-based (Multicast AODV [22]), mesh-based (CAMP [7]), and group-based (ODMRP [15]) protocols. However, these protocols cannot be easily applied to sensor networks because they all rely on preestablished overlays. These overlays are associated with considerable signaling costs. Therefore, they are usually too expensive to reconstruct in the face of frequent topology changes, such as those introduced by energy conservation protocols in sensor networks. Further, since they are usually designed for mobile nodes, such as laptops, that are much more powerful than sensor nodes, they are usually too heavy-weight to implement in sensor networks.

At last, we also find many multicast protocols for IP networks in Internet literature. Representative protocols include IGMP [6], Xcast [3], [2], [25], and DVMRP [23]. Among these protocols, Xcast [3], [2], [25] is the most relevant to our work in that, similarly to ours, it encodes the destination list into packet headers. Our work is different from Xcast in two aspects. First, Xcast relies on routing tables at intermediate hops to decide the packet flow. In contrast, we do not assume any particular routing structure, such as a routing table. Second, Xcast can only work with a single unicast routing protocol. Therefore, if the underlying routing protocol modifies the structure of the routing table, Xcast has to be modified as well. Thus, it is impractical to build a multicast layer for wireless sensor networks using Xcast. We overcome this problem by designing uCast on top of the common comparison interface exported by any underlying unicast layer. This design choice essentially decouples uCast from the underlying unicast routing details and leads to a generalized and flexible service that is significantly different from Xcast.

Based on this survey, we consider uCast as a necessary complement to previous protocols. Primarily, our work is targeted at the small group multicast scenarios. Conceptually, the application domain of uCast is shown in Fig. 1.[1]

As shown in Fig. 1, as the number of members for a particular multicast session increases or the traffic per session increases, the average cost per member decreases for connection-based protocols because the signaling cost becomes less significant. This implies that connection-based protocols are more suitable for long-term large-scale multicast. However, when the number of members is small and the traffic is low, the corresponding application domain can be characterized by spontaneous, short-term content delivery

1. The curve shown is only conceptual and helps the understanding of the application domain of uCast. It does not reflect any quantitative results.

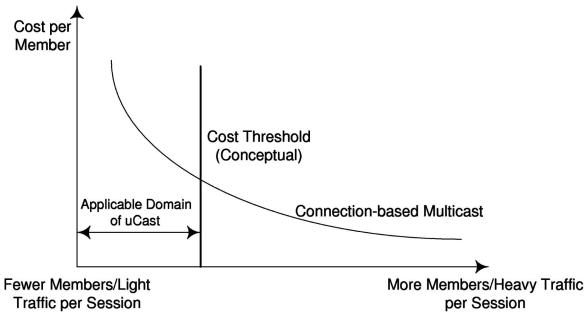


Fig. 1. Application domain of uCast.

requests within small groups. The expected cost per member will increase due to the signaling cost of connection-based multicast. When the cost per member is higher than a certain threshold, uCast becomes more efficient and preferable.

2.2 Unicast

We briefly survey different unicast protocols in this section. Later, we shall explain how uCast extends each of them. We discuss unicast protocols according to their *addressing schemes* (i.e., models for addressing individual nodes in the routing structure). We classify current addressing schemes in sensor networks into three broad domains: identifier-based, geographical location-based, and network encoding-based.

Identifier-based addressing is inherited from general ad hoc networks. In this scheme, nodes are addressed by their identifiers. Representative routing protocols of this type include those designed for ad hoc networks, such as DSR [11] and AODV [22]. One commonality of this type is that since identifiers essentially provide no information regarding topology, protocols in this type often require a flooding stage to find routes. Another commonality is that nodes need to keep routing tables. The next hop for data delivery is usually decided by a table look-up operation.

Geographical location-based addressing is another scheme primarily used for sensor networks. In this scheme, individual nodes are assumed to be location aware either through GPS or through some localization algorithm [8]. Geographical locations can be directly used for routing purposes since they approximate the relative topology of sensor network nodes. Representative protocols include GFG [4], GPSR [12], GEDIR [26], LAR [14], etc. These protocols no longer need flooding to find routes. Usually only local neighborhood information is needed to make routing decisions.

Several recently proposed protocols fall into a new (third) category we call network-encoding-based addressing. The key idea is to encode topology information into node identifiers. Such identifiers can be directly used for routing, thereby avoiding the expensive flooding process. Several network encoding schemes have been proposed, such as virtual location-based geographical routing [20], relative logical coordinate-based routing (LCR) [5], and graph embedding-based routing (GEM) [18]. Since protocols based on network encodings do not require physical location information, they are good complements for the first two types of protocols.

3 Unified Connectionless Multicast (uCast)

We first present our assumptions. As discussed earlier, we design uCast to support multiple unicast routing protocols. Therefore, we only make minimal assumptions regarding the unicast routing layer. Specifically, we do not assume any distance information or particular configuration of the routing table. On the other hand, in order to avoid functional overlap with the unicast layer, we introduce an interface that we expect the underlying unicast to export. We demonstrate that this is feasible in practice and requires minimal or no changes to the existing unicast routing protocols. As examples, we implement such an interface for three different unicast routing protocols in Section 5.

The interface is defined as a pairwise comparison in the following manner:

```
Function: Compare(NODE N1,NODE N2,NODE DESC)
Return Type: NODE (N1 or N2)
```

In this interface, N1 and N2 are candidate nodes that lead to node DESC. The interface compares these two nodes and returns the better candidate. In the following, we say the returned node is "closer" to the destination. Of course, "closer" is used only metaphorically. We do not make any assumptions on semantics of distance or on the way the comparison interface is implemented in the unicast layer.

3.1 Design

We now describe the design of uCast. The core of the protocol is the *scoreboard algorithm*, which is executed at each intermediate node along the content delivery path. The algorithm takes the list of destinations and all the neighbors of the current node as input and outputs the multicast task allocation, which is the list of next hop nodes that should receive and forward the multicast packet. Using this output, the current node generates one or more packets as required and forwards these packets to the next-hop neighbors. This process continues until all destinations receive the multicast.

So, how does the algorithm work internally? The detailed pseudocode of the scoreboard algorithm is shown in Fig. 2. As the first step, the algorithm considers the destinations one by one. For each destination, it applies the comparison interface to determine which neighbors are "closer" than the current node. Those neighbors are said to *cover* this destination and get one score point. When all destinations are considered, the neighbor node that has the highest score is chosen as a forwarding candidate. When multiple neighbor nodes share the same score, the algorithm breaks the tie either by randomly choosing one node or by using node ID. Next, the algorithm records and removes the neighbor with the highest score, as well as those destinations that have been covered by this node, from the next round of comparisons. This comparison-select-removal process continues until all destinations are covered. At this point, the preliminary neighbor selection is complete. The result set of candidate neighbors is called the *forwarding candidate set*.

Next, the algorithm begins to further optimize the candidate set. For each destination, it compares every pair of nodes in the forwarding candidate set to determine the closest node. This node is assigned the corresponding

uCast Scoreboard Algorithm

Comment: We assume the compare interface is Compare(Node1,Node2,Destination). Each neighbor node has a state of being selected or unselected.

Inputs: Destination node set DS, neighbor node set NS, current node S.
Outputs: The selected neighbor set SN. For each node in SN, the algorithm outputs a subset of DS, called SD, that forms its task.

Main Algorithm:

0 Initialization of data structures.
1 For each neighbor node in NS, set it to be unselected.

Comment: First consider three special cases in steps 2,3 and 4.
2 For each node in NS, if it is in DS, set it to be selected. Remove this node from DS and insert it into Covered Set (CS) (destination nodes in CS are assumed to be covered by a certain neighbor node).
3 For each node in DS, if there is only one neighbor in NS that is closer than S according to the Compare Interface, remove it from DS and insert it into CS. Set the status of the corresponding neighbor in NS to be selected.
4 For each node in DS, if there is no neighbor in NS that is closer to it than S using the Compare Interface, insert it into LocalMaximum Set (LS). Remove it from DS.

5 For each selected neighbor, find all destinations for which it is closer compared to S. Insert these destinations into CS remove from DS.

While *(DS is not empty)*

6 For each node in DS, find all nodes in NS that are unselected. Set each node with a score of 0. Assign one node one more score if this node is closer than S to a particular destination in DS based on the Compare Interface.
7 Select the highest score among unselected nodes in NS. In case of tie, randomly select one or break the tie using node IDs. Suppose the node selected is K and set it to be selected.
8 Among nodes in DS, find those nodes for which K is closer than the current node S and insert them into CS. Remove them from DS.

Comment: DS is empty after the loop.
Comment: Following is the optimization stage.
9 Insert all selected nodes in NS into set SN.
10 For each destination in Covered Set, choose among nodes in SN the best node (snode) using the Compare Interface. Add this destination to the corresponding SD for snode.
11 Remove those nodes in SN with an empty SD. For remaining nodes, form individual delivery tasks based on its SD.
12 If LS is not empty, switch to the underlying unicast protocol and use the corresponding local maximum handling approach to deliver packets to destinations in LS.

Fig. 2. The scoreboard algorithm.

destination node. Note that this node may not be the one that initially *covered* the destination. After this step, some nodes in the forwarding set may not be assigned any destination. They are removed from the forwarding candidate set. The remaining nodes form the optimized candidate set, and each node in this set gets a list of assigned destinations. The resulting set and the destination assignment constitute the final output of the algorithm.

3.2 Design Discussion

We now discuss several trade-offs in the algorithm. First, we discuss the performance implications of the scoreboard algorithm. Since it is greedy by nature, it remains unclear how close to optimal it is. We provide an analysis on this topic. Second, uCast uses packet headers to enumerate destinations. Therefore, there is a limit on the maximum number of destinations that one packet can address. We describe several possible solutions to this problem and discuss their effects on our protocol.

3.2.1 Analysis on the Greedy Neighbor Selection

In this section, we show by simulations that our scoreboard algorithm is very efficient at minimizing the number of branches in the multicast tree, hence reducing its cost. Recall that we always select the node with the highest score in the neighbor table until all destinations are covered. We now show that this approach is approximately as good as finding a minimal cover of destinations at each hop. Since choosing the minimal cover is the well-known set cover (SC) problem which is NP-Complete; solutions to it do not scale with the neighbor table size. General greedy selection approaches for SC problems guarantee an approximation ratio of $1 + ln(maximal\ subset\ size)$ [19] (here, approximation ratio refers to the ratio between the size of the subset selected by the greedy algorithm to the size of the subset selected by the locally optimal minimal cover algorithm). In practice, we show that the scoreboard algorithm is much closer to the optimal case than what is guaranteed by the general approximation bound.

Note, however, that although we use the minimal cover technique as the comparison baseline, this technique is not globally optimal. In fact, finding the globally optimal tree is another NP-complete problem, namely, the Steiner tree generation in graph theory. Because of the large number of nodes, the globally optimal tree structure cannot be generated in a reasonable period of time. There are, of course, various heuristic techniques to construct approximate Steiner trees. However, constructing these trees is not practical in real implementations either because this process requires global topology information. In real sensor networks, each node only has local topology information. Therefore, we compare our scoreboard algorithm with the minimal set cover algorithm because both of them only require local topology information.

In simulations, we deploy nodes with a communication range of $50\ m$ in a region of $500\ m \times 500\ m$. We place the source node at (250, 250) and multicast packets to six nodes located at the boundary of the region within a maximum angle of 60 degrees. The packets need to be relayed at least six hops, thereby ensuring that different neighbor selection approaches will have an effect. The density of the network increases from 18 to 26 nodes per communication range. We deliberately choose a relatively high density so that the size of neighbor tables is large, thereby emphasizing the effects of different neighbor selection strategies. Each scenario is tested for 100 rounds. We ensure that exactly the same topology is replayed for the minimal cover selection and the greedy selection (the scoreboard algorithm), respectively. The results are shown in Fig. 3.

In this figure, we use the average number of packets sent in one round, plotted on the Y axis, to compare the performances of different neighbor selection strategies. We observe that the difference between the minimal cover and the scoreboard algorithms can be neglected. In fact, since

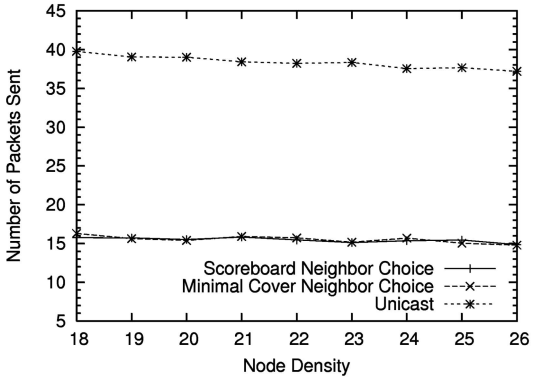


Fig. 3. Algorithm optimality analysis.

the minimal cover neighbor selection at each hop is not globally optimal, we find that the scoreboard algorithm sometimes performs better (globally) that the local optimum. Therefore, we conclude through these simulations that the scoreboard algorithm is at least as good as the minimal cover neighbor selection strategy.

A further implication of this neighbor selection strategy is that, when the destinations are clustered, usually only one node or two will be selected as the next hop, since they are expected to get the most scores. Therefore, uCast has the tendency to minimize branches and reduce potential for congestion. Moreover, since uCast does not incur state reconstruction overhead when topology changes, it further decreases network load. Hence, uCast reduces potential for network congestion (although it does not explicitly perform any type of congestion control).

3.2.2 Discussion on the Effects of Destination Encoding

In our design, we list all multicast destinations in packet headers. This design choice poses a limit on the maximum number of destinations a single packet can address. In this section, we discuss three possible trade-offs to mitigate the scalability problem introduced by this design choice.

First, as the ratio on sensor nodes becomes more powerful (for example, from CC1000 on Mica nodes to CC2420 on MicaZ nodes), it is likely that nodes will send longer packets in the next-generation sensor networks. Further, new sensors such as video cameras naturally require long packets to transmit images. In such cases, it will not be a problem to encode all destinations into the packet headers. Second, instead of enumerating all destinations, we can compress the destination list before storing it. This approach exchanges computation time for storage space. In the case where the space limit is severe, the designer may switch to this approach for better performance. At last, nodes can employ in-network aggregation techniques to further reduce the effects of destination enumeration. More specifically, after one node sends out a packet containing all destinations, it can follow up with a train of pure data packets which do not contain any destination information. To achieve this, certain synchronization and retransmission mechanisms may be employed to guarantee correctness. This train of packets that share the same destination list can be viewed as a single large packet at the receiver side.

Having said that, we emphasize that uCast is designed for small-group multicast. We expect the number of destinations to be small. Therefore, encoding all destinations into packet headers will not be a problem.

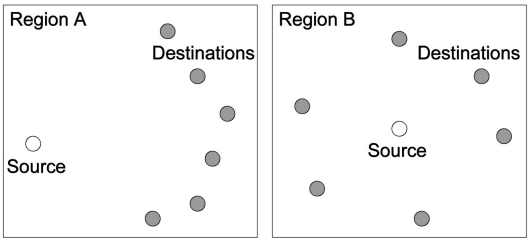


Fig. 4. Different deployment impact.

4 Performance Evaluation of uCast

We now present the performance evaluation of uCast. We are primarily interested in three aspects: the energy efficiency of uCast, its interaction with energy conservation protocols, and its integration with different unicast routing protocols. We observe that the performance of uCast is considerably affected by the positions of the destination nodes (how clustered the destinations are and how far away they are from the source node). Therefore, we first present a parameterized destination placement model to control the above attributes. We then evaluate the performance of uCast using this model.

To demonstrate the performance advantage of uCast, we compare it to connection-based protocols. The baselines include Shortest Path Tree (SPT), Greedy Incremental Tree (GIT), and plain unicast. In SPT, we assume that the source node sends packets along the shortest paths to all destinations and aggregates common paths to form a tree structure. We select SPT because it is the backbone tree structure used in several representative connection-based multicast protocols [22]. GIT is another selected baseline. The construction process of GIT is centralized and requires full knowledge of the topology. It proceeds as follows: First, we connect the source node with the nearest destination via a shortest path. This path forms a partially completed tree structure. Then, we find the nearest destination node to the existing tree and connect this node to the closest node in the structure. We iteratively find the next nearest node in the remaining destinations and connect it until all destinations are connected. Clearly, each step requires global topology information, and the construction process is quite computationally intensive. Therefore, GIT is not applicable for sensor networks. However, previous literature has pointed out that a GIT tree is usually very compact, implying that if we deliver packets along such a tree, we may distribute data in fewer hops compared to other tree structures. Therefore, we use the GIT structure as a best-case baseline for comparison.

4.1 The Destination Placement Model

We describe the destination placement model in this section. We first give an intuitive explanation on why this model is important for the performance evaluation of uCast. Consider the two scenarios in Fig. 4. Intuitively, using multicast in region A saves more energy than in region B compared with using unicast because the destinations are more clustered in region A. Our model is designed to characterize such differences. It presents four parameters of

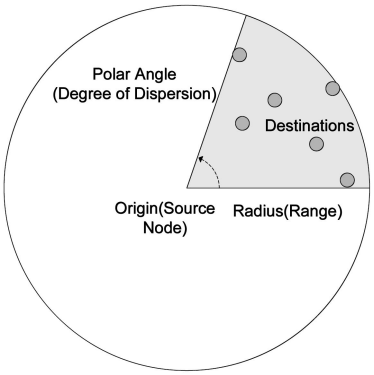


Fig. 5. Destination placement model.

destination placement that have effects on the performance of multicast. These parameters model a minimal pie-shaped region that contains the destinations and the source, as shown in Fig. 5. The parameters are the angle of dispersion (AOD); the radius, which corresponds to the farthest distance that one destination can be positioned from the source node, the density, i.e., the number of nodes within a communication range, and the number of destination nodes. We note that if we set AOD as 2π and the range large enough, our model defaults to a random placement model. In the following simulations, once the polar angle is set, the distances of nodes from the source conform to a uniform distribution.

Unless otherwise stated, the default parameters are as follows: The communication range is $50\,m$, the area is $500\,m \times 500\,m$, the density is 20 nodes per communication range, AOD is 90 degrees, the number of destinations is 10, and the radius of the pie shaped area is $250\,m$. A total of 636 nodes are deployed by default. The data rate is 6 packets per minute, except in Section 4.3, where multiple data rates are tested. In Section 4.2, we simulated for 100 packets (about 16 minutes). In Section 4.3, we simulated for 120 minutes. We selected different time lengths because the evaluation purposes are different. We assume that each node has the same transmission power level. The simulations are done in the Glomosim [28] environment.

4.2 Energy Efficiency

In this section, we compare the energy efficiency aspect of uCast with other multicast protocols. To accurately estimate energy consumption, we use the parameters of MicaZ nodes (one of the most advanced sensor network nodes currently available) in energy consumption simulations. More specifically, energy is consumed on both sending and receiving packets. According to the data sheet of the CC2420 radio on MicaZ [27], sending and receiving have current levels of $17.4\,mA$ and $18.8\,mA$, respectively. The voltage supply is assumed to be $3\,V$, and the data rate is $250\,kbps$. Packets are assumed to have a payload of 20 bytes, and each destination requires 4 bytes in the header. We do not consider the signaling cost of connection-based protocols since the impact of this cost depends on how the specific protocol is implemented and how frequently the topology changes. The key metric we use is the total energy consumption, in joules, for sending 100 packets to all destinations from the source.

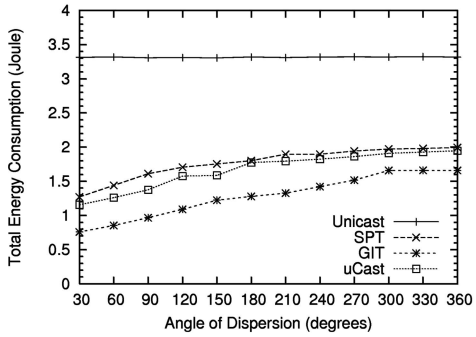


Fig. 6. Impact of AOD on energy consumption.

We begin with a static network topology. Observe that this is not the scenario uCast is optimized for. The main advantage of uCast lies in its robustness to topological dynamics. Hence, our objective from using a static topology is to show that we do not degrade the performance by removing multicast state when the network is static. Later, we shall present the key advantages of uCast by considering topology changes.

In the following simulations, uCast is integrated with geographical forwarding, a commonly employed unicast protocol in sensor networks. The common comparison interface is implemented by returning the node that is geographically nearer to the destination. When a local minimum is reached, uCast leverages the GPSR [12] traversing technique to handle nodes in the *LocalMinimum* set. Since there are no state transitions in this experiment, no routing layer route repairs are needed.

Figs. 6, 7, 8, and 9 show the impact of the four destination placement parameters on multicast performance. Based on these results, we have several observations. First, observe that uCast performs better than SPT in these figures, except Fig. 9, where the traversing technique of GPSR significantly increases the path length. Also observe that, as we expected, GIT performs better than uCast. We note that the prohibitive construction cost of GIT makes it unsuitable for sensor networks and, hence, it is not a contender in practice.

Fig. 9 is especially interesting. In this case, both uCast and unicast increasingly turn to the GPSR traversing technique to deliver packets around voids, which degrades their performance. Considering that practical sensor networks are usually deployed with a sufficiently high density to ensure coverage, topology voids are not common. Furthermore, the designer may decide to incorporate

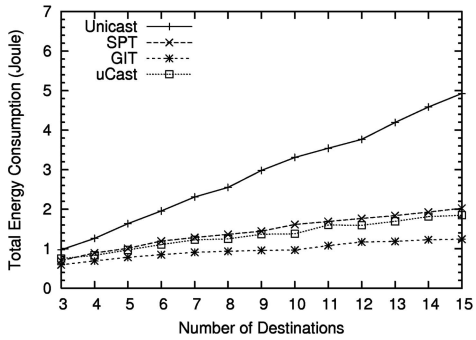


Fig. 7. Impact of number of destinations on energy consumption.

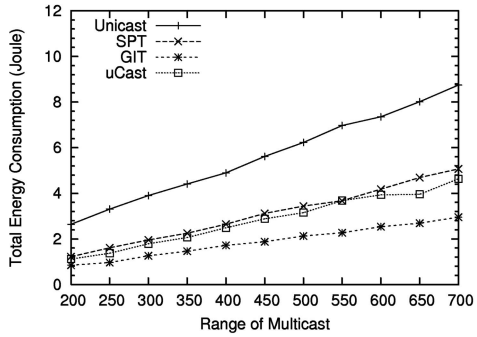


Fig. 8. Impact of range on energy consumption.

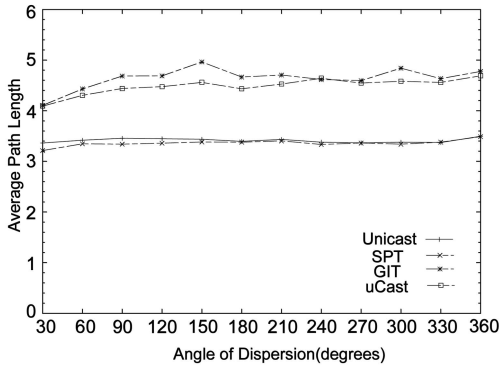


Fig. 10. Impact of AOD on path length.

adaptive features into multicast, where the applications have the option of switching from uCast to SPT when the density becomes extremely low. Therefore, we conclude that our stateless multicast generally does not incur a performance penalty compared to stateful approaches even when the network is static.

Figs. 10 and 11 show the comparison results of the average path length. Due to the effect of path aggregation, we observe that uCast and GIT deliver packets along longer routes compared with SPT and unicast. This is intuitive since SPT and unicast typically find near-optimal paths. The increase in the path length means that uCast may have a slightly higher end-to-end delay. Since the main constraint in sensor networks is the limited energy supply, we believe that increasing path lengths to save total energy consumption is an acceptable compromise. An operator would welcome a slightly longer latency for each packet in exchange for a significantly extended network lifetime.

4.3 Impact of Topological Changes

In this section, we evaluate uCast in the presence of topological changes. Such changes are introduced by energy saving protocols that turn nodes into and out of sleep states. We expect that, in this case, the advantages of uCast should dominate.

We use three parameters of energy conserving protocols to evaluate the multicast performance:

- Toggle Cycle: Toggle Cycle is the time interval between consecutive transitions into the sleep state by individual nodes. This parameter reflects the frequency at which the state information kept by intermediate nodes becomes invalid. As the frequency goes higher, the performance of state-based multicast protocols should drop accordingly.
- Scale: Scale refers to the size of the multicast area. As the size scales up, the impact of topological changes becomes more significant and the reconstruction cost goes higher. As a result, we expect that the performance of state-based multicast protocols will drop with a larger scale.
- Packet Delivery Rate: Another parameter that we change is the packet delivery rate. We use two such rates in our experiments, 6 packets per minute and 12 packets per minute, respectively. Observe that these are source packets. If a multicast is sent to 10 destinations and there are four hops on the way to each, up to 480 packets are generated per minute in the network, which is acceptable for sensor network applications. We do not choose higher rates because we observed a higher level of radio congestion, which would typically be avoided in a practical scenario.

In the simulation setting, we place the source node at (0, 0) and let it periodically deliver packets to 10 destinations with an AOD of 90 degrees. The total simulated time is 120 minutes. Other settings are left at the default.

The energy conservation model we use is random sleep scheduling. For example, in Fig. 12, 10 percent sleep scheduling with a 10 seconds toggle period means that one node sleeps for one second in every 10 seconds. Each node has the same toggle period. We assume that there is no coordination between nodes since this is the model that can

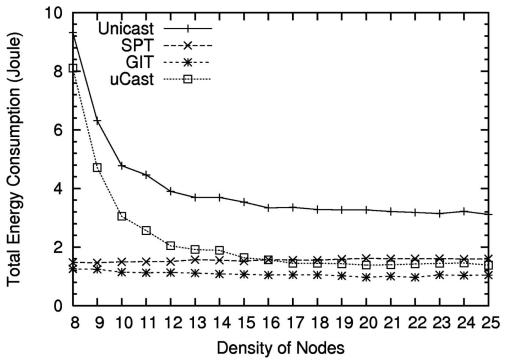


Fig. 9. Impact of density on energy consumption.

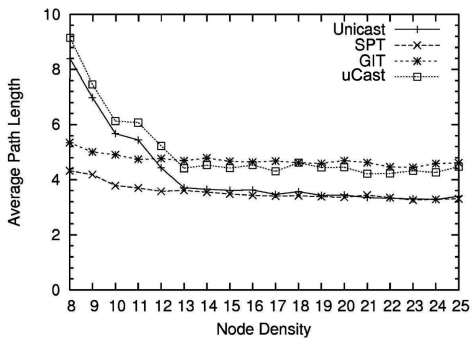


Fig. 11. Impact of density on path length.

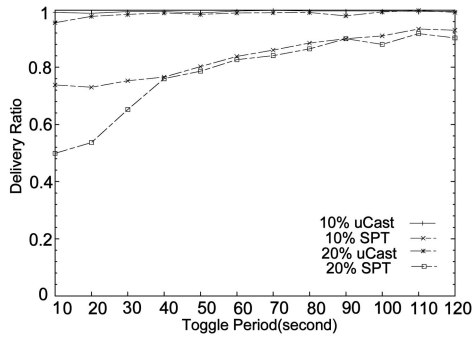


Fig. 12. Impact of toggle period on delivery ratio.

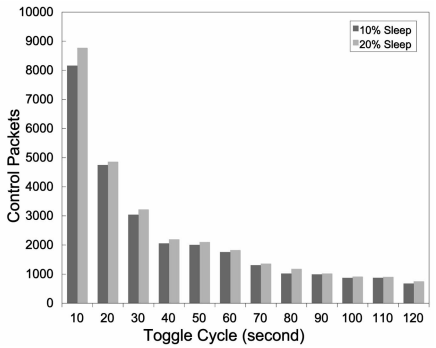


Fig. 14. Control packets of SPT multicast with range of 250 m.

be most easily implemented in sensor networks. It is also the foundation of a variety of other more complex sleep scheduling protocols [9], [29].

We compare uCast with SPT in this section. We do not include GIT because it is computed in a centralized manner and it has a prohibitively high computational cost in the presence of topological changes.

Figs. 12 and 13 show the performance evaluation results. These two experiments are carried out for a data rate of 6 packets per minute. The comparison results demonstrate the superiority of stateless multicast in the presence of node state transitions. More specifically, we have the following observations.

First, as the toggle periods become shorter, we observe that the delivery ratio for SPT multicast degrades considerably. For example, when nodes use a toggle cycle of 10 seconds and sleep 20 percent of the time, only around half of all packets successfully arrive at the destinations using the SPT tree for multicast. On the other hand, we observe that uCast achieves a delivery ratio of around 96 percent, enough for common multicast purposes. We attribute the superior performance of uCast to its statelessness.

Second, we observe from Fig. 13 that connection-based multicast are less scalable compared with uCast. This is quite intuitive in that, as the multicast range scales up, it is more likely for one node on the tree to a enter sleeping state for energy conservation purposes. Therefore, there is a higher probability for a packet delivery session to encounter a state loss.

One tentative solution to fix the state loss problem for state-based protocols is to let the last node that has successfully received the packet locally reconstruct the SPT, once it detects that the next hop has entered a sleeping mode. This approach will guarantee that the SPT achieves a 100 percent delivery ratio. However, this approach is quite expensive. We implemented this tentative patch for SPT and recorded how many control packets are sent out to reconstruct the SPT structure. The results are plotted for two different multicast ranges, $250\,m$ and $500\,m$, respectively, as shown in Fig. 14 and 15.

Figs. 14 and 15 demonstrate that, even with only 100 packets sent from the source, there are usually thousands of control packets required to locally rebuild the tree. The reason is that, when a state transition occurs for a node that was initially in the SPT tree structure, it can no longer forward packets from its upstream nodes. Therefore, the upstream node must initiate a flooding process to try to locate the next downstream node available. In our simulation, we find that this upstream nodes usually needs to flood packets to two-hop neighbors, while, in rare cases, three-hop neighbors are needed. Therefore, even if the tree structure is only partially broken, the flooding process generates a considerable amount of traffic. Of course, other modification possibilities also exist, such as enforcing that nodes should not go to sleep when they are in multicast sessions. However, doing so incurs nontrivial reductions in energy savings. On the other hand, uCast has a significantly smaller overhead because it does not need any control packets to handle individual node state transitions. We acknowledge that uCast does have additional overhead in the form of destination lists in the packet headers. This overhead, however, is usually quite small when only a few destinations need to be enumerated. We

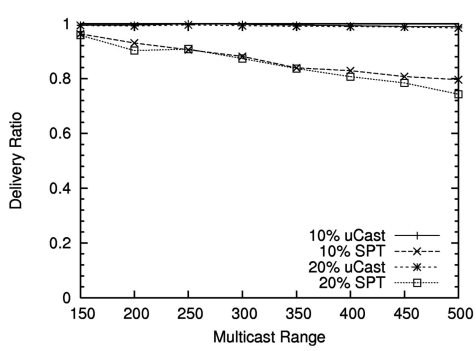


Fig. 13. Impact of scale on delivery ratio.

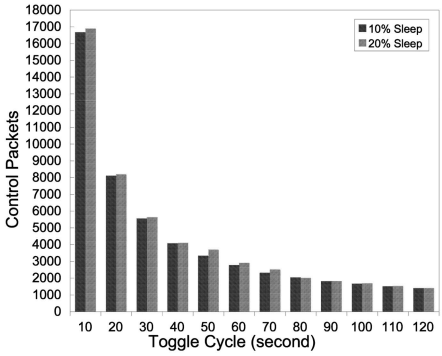


Fig. 15. Control packets of SPT multicast with range of 500 m.

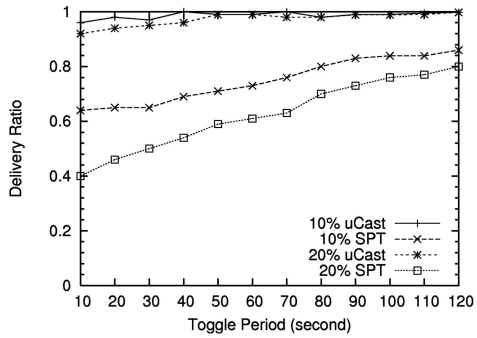


Fig. 16. Impact of toggle period on delivery ratio with a higher data rate.

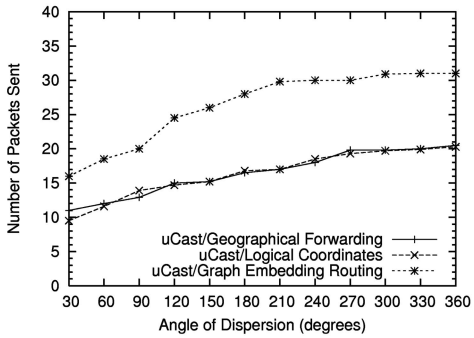


Fig. 17. Impact of addressing schemes on traffic.

shall consider this overhead in the experiments based on a realistic testbed in Section 5 and show that it does not have a significant effect on the efficiency of uCast.

Fig. 16 studies the effect of the increased data rate in which we change the data rate to 12 packets per minute. We can observe a slight decrease in the delivery ratio, compared with Fig. 12. As expected, the advantages of uCast still dominate.

4.4 Integration of uCast with Unicast Protocols

Another goal of uCast is to interface with different unicast protocols. We implemented uCast on top of three unicast protocols: geographical forwarding, logical coordinate-based routing, and graph embedding-based routing. In each of them, we made no changes to the existing unicast protocols other than extending them to provide the common comparison interface. In this section, we first describe how we implemented the common comparison interface, followed by performance comparisons based on simulations.

For the geographic forwarding routing protocol, we implemented the comparison interface based on physical distance comparisons. More specifically, the interface returns the node that is nearer to the destination. The second routing protocol we use is the logical coordinate based routing protocol (LCR) [5]. LCR uses hop counts to a few landmarks from each node as its logical coordinate vector. Based on these vectors, LCR also provides a definition of logical distances. In the comparison interface, we simply compare the logical distances from nodes $N1$ and $N2$ to node $DEST$, and the node with the smaller distance is returned by the interface.

The way we implemented the compare interface in Graph Embedding-based Routing (GEM) [18] is a little more complex. In GEM, one node is chosen as the root. GEM then constructs a tree structure and assigns a (level, angle) combination to each node based on its topological position. The assigned combination forms a unique identifier for each node. GEM then delivers packets using this tree structure based on considerations of both the level and the angle of each node. Interestingly, GEM has no definition of distance. Therefore, we used both the level and the angle information to implement the comparison interface. More specifically, when comparing two nodes, we followed the same procedure as the routing process in GEM: If one node is the parent or the offspring of the destination node in the tree structure and if the other node is not, then the parent/offspring node is returned by the interface; if both nodes are parent/offspring nodes, then the node with a level nearer to the destination is returned; if both nodes are not parent/offspring nodes, then the node with a nearer angle range is returned. Theoretically, this approach guarantees 100 percent delivery ratio if all nodes in the same level are perfectly aligned.

Fig. 17 shows the performance evaluation results of running uCast on the three aforementioned unicast protocols. We observe that both geographic forwarding-based and logical coordinates-based routing appear quite similar in their performances. However, uCast based on GEM shows quite different performance characteristics. We attribute such differences to the more convoluted delivery paths in GEM, which increase path lengths considerably. Another way to explain the differences is that both logical coordinates and physical coordinates are based on Cartesian-like coordinate frameworks, which are considerably different from GEM, whose identifiers are more like polar coordinates.

We did not implement uCast on identifer-based unicast routing protocols like DSR. In such protocols, a look-up operation is used to return the next node on the path to the destinations. The implementation of the compare interface is therefore very straightforward: It simply returns the next hop node for a given destination from the look-up table and this node will always percolate to the top and be chosen as the best candidate node.

5 Implementation on Sensor Platform

To investigate the performance of the uCast protocol in a running system, we implemented it on the MICA2 platform. The code size is 992 bytes. As shown in Fig. 18, we bridge the uCast protocol with the underlying unicast routing protocol (Geographic Forwarding) using the uCast2uniCast interface (the NesC definition of this interface is shown in Fig. 19). In this interface definition, the $compare()$ command is the mandatory part of the interface for node comparison. The $getNeighborTable()$ command is optionally provided.

In the experiment, we used a testbed of 25 MICA2 motes (5 by 5). Fig. 20 shows the experimental setting and data delivery traces. We conducted three sets of experiments, with 3 or 5 destination nodes selected in each set. We plot the multicast traces and unicast traces in this figure. All data are gathered from real tests. For multicast traces, we use *forking points* to represent the positions where data are sent to multiple receivers.

We now compare the energy consumption of uCast and unicast. We use the parameters of MICA2 motes: The sending current is $21.5\ mA$, the receive current is $7.4\ mA$,

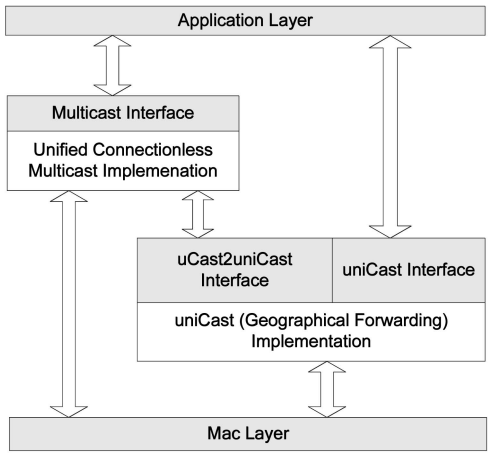


Fig. 18. Implementation of uCast protocol.

```
interface uCast2uniCast
{
  /* For comparing distances of two neighboring nodes */
  command result_t compare(uint16_t ID1, uint16_t ID2, uint16_t destination);
  /* Optional: get neighbor information */
  command result_t getNeighborTable(uint16_t *IDs, uint8_t *count, uint8_t  MAX_COUNT);
}
```

Fig. 19. uCast interface in NesC definition.

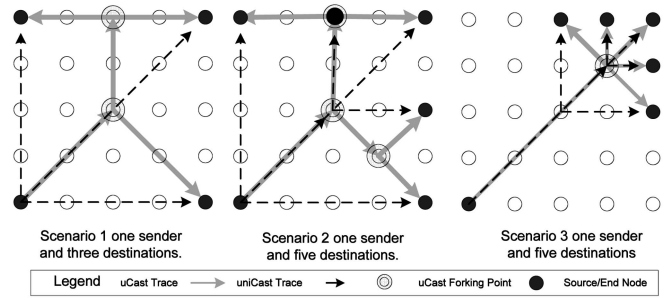


Fig. 20. The prototype experiment traces.

the bandwidth is $19.2\ Kbps$, each node has a $3\ V$ supply, and each packet contains a 12 byte payload. We then calculate energy consumption of different data distribution approaches using these parameters and plot the results in Fig. 21. Observe that, in these three settings, uCast significantly decreases energy consumption compared to unicast. Furthermore, if we compare Scenario 1 and Scenario 2, we notice that, as the number of destinations increases, uCast saves more energy. The same observation also holds when the destinations become more clustered, as from Scenario 2 to Scenario 3. These observations are consistent with our analysis in Section 4.

At last, we study a more generalized scenario. In this experiment, the source node 0 delivers data packets periodically to three randomly selected destinations. We compare the total number of packet transmissions in uCast to unicast. Again, we use geographic forwarding as the basic unicast routing protocol. A total of 300 packets are delivered and the recorded data load for each node is plotted in Fig. 22. We observe a considerably reduced data load for uCast. In fact, uCast reduces the total number of data transmissions by 45.7 percent compared to unicast. Therefore, we conclude that uCast exhibits a very satisfactory energy efficiency for content delivery in realistic experiments.

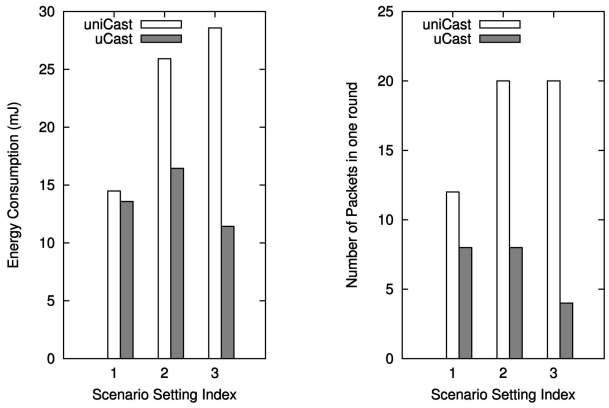


Fig. 21. The performance comparison.

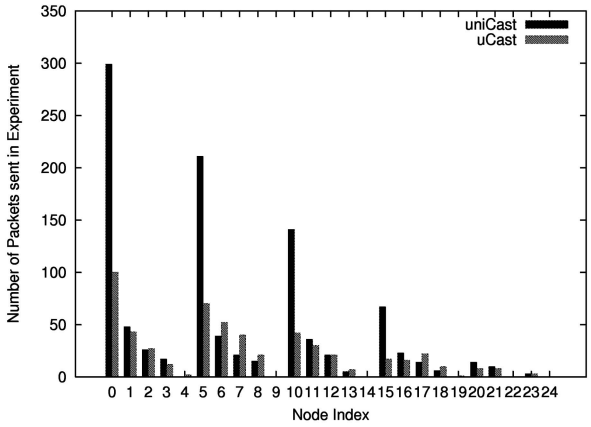


Fig. 22. Experimental comparison between uCast and uniCast.

6 Conclusions

In this paper, we presented uCast, a unified connectionless multicast protocol for sensor networks. The design of uCast is motivated by the problem that state-based protocols cannot adapt efficiently to the network dynamics introduced by energy conservation protocols. We designed and implemented uCast on top of three different unicast routing protocols to show that it is generic. Several conclusions are drawn from our evaluation and comparisons. First, uCast is generally as efficient as connection-based multicast protocols, even when the network is static. Second, the connectionless nature of uCast makes it more robust in the face of network dynamics. Finally, uCast can be easily implemented on different unicast routing protocols. The implementation of uCast on a real testbed also supports our conclusions.

Acknowledgments

The work reported in this paper is funded in part by US National Science Foundation grants EHS-0208769 and EHS-0509233.

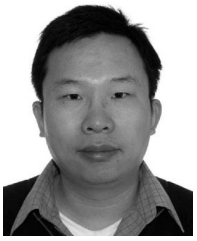
References

[1] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher, "Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks," *Proc. ACM MobiSys*, 2003.

[2] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, and E. Muramoto, Internet Draft, draft-ooms-xcast-basic-spec-09.txt, 2005.

[3] R. Boivie, N. Feldman, and C. Metz, "On the Wire—Small Group Multicast: A New Solution for Multicasting on the Internet," *IEEE Internet Computing,* vol. 4, pp. 75-79, 2000.
[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *ACM/Baltzer Wireless Networks,* 2001.
[5] Q. Cao and T. Abdelzaher, "A Scalable Logical Coordinates Framework for Routing in Wireless Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS),* 2004.
[6] W. Fenner, "Internet Group Management Protocol, Version 2," *IETF RFC 2236,* Nov. 1997.
[7] J. Garcia-Luna-Aceves and E.L. Madruga, "The Core-Assisted Mesh Protocol," *IEEE J. Selected Areas in Comm.,* special issue on ad hoc networks, Aug. 1998.
[8] T. He, C.D. Huang, B. Blum, J.A. Stankovic, and T. Abdelzaher, "Range-Free Localization and Its Impact on Large Scale Sensor Networks," *Proc. ACM MobiCom,* Sept. 2003.
[9] T. He, S. Krishnamurthy, L. Luo, T. Yan, B. Krogh, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J.A. Stankovic, T.F. Abdelzaher, and J. Hui, "Vigilnet: An Integrated Sensor Network System for Energy-Efficient Surveillance," *ACM Trans. Sensor Networks,* 2006.
[10] Q.F. Huang, C.Y. Lu, and C.C. Roman, "Spatio-Temporal Multicast in Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2003.
[11] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing,* chapter 5, pp. 195-206, 1996.
[12] B. Karp and H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom,* Aug. 2000.
[13] Y. Ko and N. Vaidya, "Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms," *Proc. IEEE Workshop Mobile Computing Systems and Applications (WMCSA),* Feb. 1999.
[14] Y.B. Ko and N.H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Proc. ACM MobiCom,* 1998.
[15] S.J. Lee, M. Gerla, and C.C. Chiang, "On-Demand Multicast Routing Protocol," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '99),* Sept. 1999.
[16] J.Y. Li, J. Jonnotti, D.D. Couto, D.R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," *Proc. ACM MobiCom,* 2000.
[17] J.C. Navas and T. Imielinski, "Geocast, Geographic Addressing and Routing," *Proc. ACM MobiCom,* 1997.
[18] J. Newsome and D. Song, "Gem: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2003.
[19] V.T. Paschos, "A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems," *ACM Computing Surveys,* June 1997.
[20] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *Proc. ACM MobiCom,* 2003.
[21] S. Ratnasamy et al., "Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table," *Mobile Networks and Applications (MONET),* special issue on wireless sensor networks, 2003.
[22] E. Royer and C.E. Perkins, "Multicast Operation of Ad-Hoc, On-Demand Distance Vector Routing Protocol," *Proc. ACM MobiCom,* 1999.
[23] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended Lans," *ACM Trans. Computer Systems,* 1990.
[24] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-Centric Storage in Sensornets," *Proc. ACM SIGCOMM HotNets,* 2002.
[25] M. Shin, K.S. Park, Y.J. Kim, and S.H. Kim, "Explicit Multicast Extension (xcast+) for Efficient Multicast Packet Delivery," *ETRI J.,* vol. 23, no. 4, pp. 202-204, 2001.
[26] I. Stojmenovic and X. Lin, "Gedir: Loop-Free Location Based Routing in Wireless Networks," *Proc. Int'l Conf. Parallel and Distributed Computing and Systems,* Nov. 1999.
[27] The CC2420 Datasheet from Chipcon Company, http://www.chipcon.com, 2003.
[28] The Glomosim Project, 2006, http://pcl.cs.ucla.edu/projects/glomosim/.
[29] T. Yan, T. He, and J. Stankovic, "Differentiated Surveillance for Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2003.
[30] F. Ye, H.Y. Luo, J. Cheng, S.W. Lu, and L.X. Zhang, "A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks," *Proc. ACM MobiCom,* 2002.

Qing Cao received the MS degree from the University of Virginia in 2004 and the BS degree from Fudan University, Shanghai, China, in 2002, both in computer science. He is currently a PhD student in the Department of Computer Science at the University of Illinois, Urbana-Champaign. His research interests include wireless sensor networks, real-time, embedded systems, and networking systems. He is a student member of the IEEE.



Tian He received the BS degree from the Nanjing University of Science and Technology, Nanjing, China, in 1996, the MS degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2000, and the PhD degree under Professor John A. Stankovic from the University of Virginia in 2004. He is currently an assistant professor in the Department of Computer Science and Engineering at University of Minnesota-Twin Cities. Dr. He is the author or coauthor of more than 30 refereed publications in international conferences and journals. His research interests include wireless sensor networks, real-time embedded systems, and distributed systems. He is a member of the ACM and the IEEE.

Tarek Abdelzaher received the BSc and MSc degrees in electrical and computer engineering from Ain Shams University, Cairo, Egypt, in 1990 and 1994, respectively. He received the PhD degree from the University of Michigan in 1999 on quality of service adaptation in real-time systems. He was an assistant professor at the University of Virginia until his promotion with tenure in 2005, where he founded the Software Predictability Group. He is currently an associate professor in the Department of Computer Science, the University of Illinois at Urbana-Champaign. He has authored/coauthored three book chapters and more than 60 refereed publications in leading conferences and journals in several fields including real-time computing, distributed systems, sensor networks, and control. He is an associate editor of the *IEEE Transactions on Mobile Computing*, the *Journal of Real-Time Systems*, the *International Journal of Embedded Systems*, and the *Ad Hoc Networks Journal*, as well as the editor of the *ACM SIGBED Review*. He was a guest editor for the *Journal of Computer Communications* and the *Journal of Real-Time Systems*, and is a coeditor of *IEEE Distributed Systems Online*. He has served on numerous technical program committees in real-time computing, networking, quality of service, distributed systems, sensor networks, multimedia, and mobile computing, among others. He also held several conference organization positions including program chair of RTAS '04, demo chair of MobiSys '05, poster chair of ICDCS '03, sensor networks vice chair of RTSS '05, and general chair of RTAS '05. His research interests lie broadly in understanding and controlling the temporal properties of software systems in the face of increasing complexity, distribution, and degree of embedding in an external physical environment. He is a member of the IEEE and the ACM.