AN INTEGRATED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS

Yong Liu[†], Xuhui Hu[†], Taekyoung Kwon[‡], Chunhui Zhu[†], Jianliang Zheng[†], Myung J. Lee[†]

[†]Graduate School and University Center
City University of New York, City College
New York, NY

[‡]School of Computer Science and Engineering
Seoul National University
Seoul, Korea

ABSTRACT

Since the size of on-chip memory greatly affects the chip cost, sensors targeting at tiny size and low cost cannot afford large memory module. How to route packets efficiently in networks containing memory constraint nodes is an interesting and challenging research topic. In this paper, we propose an integrated routing protocol for large-scale sensor networks consisting of both memory-constraint nodes and memory-sufficient nodes. In our design, a spanning-tree is first built to ensure that every node pair is connected through a tree route. Source nodes can directly use tree routes to deliver short or urgent messages; they can also choose to discover shortcuts through a reactive routing process. Intermediate nodes, depending on their memory conditions, either create routing entries to guide data forwarding, or simply relay all packets following tree routes. Simulations verify the feasibility of our protocol and show it a promising scheme empowering highly flexible network designs.

I. INTRODUCTION

Rapid advances in VLSI technology have made it possible to integrate all modules required by a wireless sensor into one or a few chips. As chip memories occupy silicon areas multiple times larger than processor cores do, it is important to take the size of sensor memories into account when developing networking protocols for wireless sensor networks.

L. Hester *et al*. [1] proposed to organize isolated wireless sensors as a spanning-tree. One of the important features of the tree-type network is its ability to do self-routing. Nodes in a spanning-tree network can have their addresses assigned in conformity to the tree architecture. Therefore, they can easily route a packet based on the packet's destination address and their children's addresses. No storage of routing tables is necessary. Clearly, the self-routing feature is desired by nodes with very limited memory capacities.

Spanning-tree formation algorithm like [1] establishes the shortest or near-shortest routes between the root node and its descendants. However, most of the other tree routes, especially the tree routes between leaf nodes, are far from optimal. Relying on tree routes to carry all communication traffic will result in significant transmission overhead and uneven distribution of traffic loads.

Proactive and reactive routing protocols [2]-[4] proposed for wireless ad hoc networks aim at the construction of optimal routes between source-destination pairs. Proactive routing protocols like DSDV [3] enable instant message deliveries. As the tradeoff, every node has to create and maintain a large routing table with routing entries destined for all other nodes. Reactive routing protocols like AODV [4] permit nodes to store only the routing entries for active routes. However, on-demand route discoveries incur extra control overhead and delays to message transmissions.

Clearly, the design of routing protocols for wireless sensor networks faces a tradeoff among sensor memory capacity, network latency and communication overhead. In view of device cost, sensors are typically equipped with low-capacity memory modules. Proactive routing protocols, therefore, become too "luxurious" to most sensors in a large network. Some sensor devices, targeting at extremely low costs, even do not have enough memory spaces to keep the routing entries for a few active source-destination pairs. This type of sensors is hereafter referred to as Memory-Constraint node (MCN). In contrast, Memory-Sufficient node (MSN) represents sensors and networking devices that have large enough memory to conduct at least the reactive routing process. For MCNs, the spanning-tree self-routing approach is the preferred choice. MSNs can support both the tree-based routing and the reactive routing – the former is suitable to transmit short, sporadic messages with stringent delay requirements, while the latter is more efficient in continuous data exchanges or deliveries of long messages.

In order to support both MCNs and MSNs and at the same time combine the merits of the tree-based routing and the

Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

reactive routing, we design an integrated routing protocol for efficient data transmissions in low-cost, large-scale sensor networks. Our protocol first organizes all sensor nodes as a spanning-tree network, so that every node automatically has the ability to do tree routing. Active sources or source agents, based on the types of traffic requests, determine whether to use existing tree routes or start up an on-demand routing process to discover possible shortcuts. The on-demand routing process creates temporary routing entries at MSNs to guide data transmissions through the newly discovered shortcuts, while MCNs relay all packets conforming to the tree-based routing algorithm.

The rest of the paper is organized as follows. Section 2 gives a brief overview of two routing protocols that represent the tree-based routing and the reactive routing, respectively. Section 3 describes two integrated routing approaches in details. Section 4 compares the performances of different approaches via simulations. Finally, section 5 concludes the paper.

II. OVERVIEW OF TREE ROUTING AND AODV

Before going into the details of the integrated routing, we first introduce two existing routing protocols: the tree-based routing protocol and AODV.

A. Tree-based Routing

Network formation and node addressing are essential to a tree-type network with self-routing capability. In [1], L. Hester *et al*. presents a detailed spanning-tree formation method. In this method, the root initiates tree formation by sending beacons to its neighbors and inviting them to join the tree. The neighbors successfully connecting to the root become the children of the root and also the members of the tree. Once a node joins the tree, it begins to send beacons to recruit new members as its children. When a node receives several beacons from its neighbors, it selects the neighbor with the shortest hop-count to the root as its parent.

In order to enable the self-routing ability, each tree member has to obtain a logic address corresponding to its position in the tree. For efficient usage of the address space, it is beneficial for different levels of tree members to collect the numbers of their descendants and report to their parents. Once the root receives the reports from its children, it initiates the addressing process by dividing the overall address space among its children. Children with more descendants are assigned with bigger address blocks. The children of the root further divide the address blocks among their own children. Similarly, the grandchildren with more descendants are allotted bigger address sub-blocks. These address assignment is continued until all leave nodes receive their addresses.

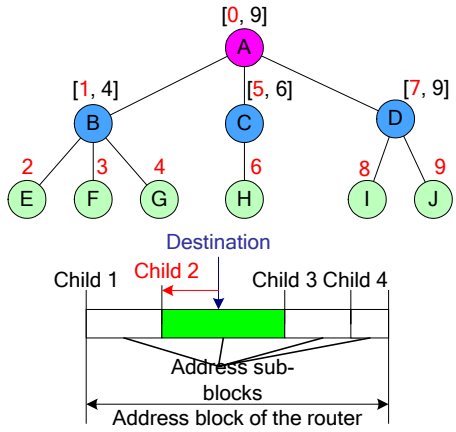

Figure 1. Tree addressing and routing

Figure 1 shows an example of tree addressing. The root A has an address of 0. After the tree is successfully formed, the root A collects the descendant numbers of all its children. Based on these descendant numbers, A assigns 4 addresses to child B, 2 addresses to child C and 3 addresses to child D. B, C and D pick up the first addresses in their address blocks and assign the remaining addresses in the address blocks to their children. This addressing method facilitates the self-routing process. For instance, node H relays a packet destined for node F. As H does not have any child, it forwards the packet to its parent C. C finds the destination F has an address out of its address block. So it forwards the packet to its parent A. As F's address 3 falls into A's address block, and A further finds address 3 belongs to the address sub-block of its first child B, A forwards the packet to B, and B forwards it to F. Once the tree is formed and the addresses are allotted, packets can be quickly relayed from sources to destinations without additional route discovery efforts. More importantly, nodes do not burden their memories to maintain routing entries. Nevertheless, the route optimization is penalized.

B. AODV

AODV is one of the most famous reactive routing protocols. In AODV, a source that intends to reach a distant destination floods the whole network with a route request (RREQ) packet to search for all possible routes leading to the destination. Upon receiving the RREQ, each intermediate node creates a reverse routing entry for the source if it does not have a fresh one. The intermediate node also checks whether it has an existing entry for the destination. If it has, a route reply (RREP) packet is generated and unicast back to the source along the reverse RREQ route. Otherwise, it rebroadcasts the first received RREQ and suppresses the duplicated ones. When the destination receives the first RREQ or a RREQ coming from a shorter route, it sends a RREP back to the source. The nodes along the newly discovered routes create forward routing entries

for the destination when receiving the RREPs. Source and destination sequence numbers are included in the control packets and routing entries to prevent loop problems. When a route entry is not used for a long time, it is deleted from the routing table to leave space to active entries.

AODV requires all nodes to reserve big enough memory spaces to store possible routing entries for active sources and destinations. Network-wide route explorations facilitate the discoveries of optimal or near-optimal routes. Nevertheless, they introduce large control overhead, too. As most routes are formed on demand, network latency is quite high.

For simplicity, we employ a "light-weight" version of AODV – AODVjr [5] in our integrated protocol.

III. INTEGRATED ROUTING PROTOCOL

The objective of the integrated routing protocol is two-fold: (i) to provide a routing solution that combines the virtues of both the tree-based routing and AODVjr; (ii) to support sensor networks with MCNs and MSNs mixed in any ratio.

The integrated routing protocol has two phases: tree formation and data routing. The tree formation phase is initiated at the network startup stage, and ends with a completely established spanning-tree topology. The data routing phase starts when both a source and its desired destination join the tree. The source can choose to rely only on the tree route to transmit current messages, or discover a better route through a reactive routing process. The selection of routing modes is based on such factors as delay bounds, message lengths and traffic types etc. The routing layer of the source has to seek cooperation from its upper layers to make a proper choice. Even if the source decides to discover a new route, it can use the tree route during the new route discovery and formation periods. As soon as the new route is ready, the source can switch from the tree route to the new route to continue message transmissions. We call the new route discovered by the reactive routing process as ReActive Route (RAR).

MCNs and MSNs should behave differently in the reactive routing process since MCNs do not have enough memories to store big routing tables. In the remaining of this section, we address on-demand route formations in networks containing both MCNs and MSNs. Two approaches are proposed under different assumptions and requirements.

A. Slim MCN approach

In this approach, we assume that MCNs cannot store any routing entry, and the tree routing is the only way for them to relay packets. Therefore, if a MCN appears in one RAR, its next-hop neighbor (toward the destination) must belong to the tree route from the MCN to the destination. If the same RAR is also used to carry reverse traffic from the destination to the source, the MCN's previous-hop neighbor (toward the source) should lie in the tree route from the MCN to the source. For example, in Figure 2, source S builds a RAR to reach destination D. If the RAR passes through node K, it must pass though node N and H as well, since K shall forward all data packets destined for D to H, and forward all data packets destined for S to N.

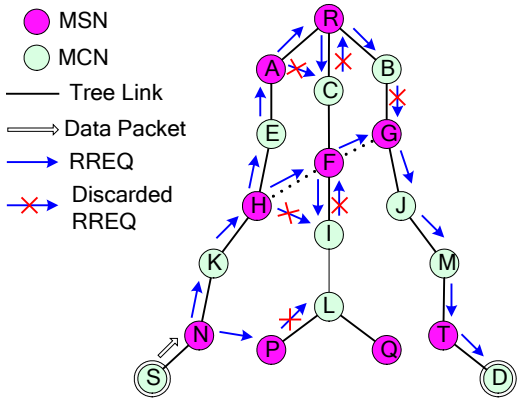

Figure 2. Slim MCN approach

1) Handling of data packets

We add a tree-route flag in each data packet. If the tree-route flag is set or the packet destination is a descendant of the routing nodes, both MCNs and MSNs should relay the packet by the tree routing. Otherwise, MCNs still use the tree routing, but MSNs should route the packet by consulting their routing tables.

If a MSN source decides to use a RAR to deliver a message, but cannot find any available routing information, it initiates the route discovery process by broadcasting a RREQ. At the same time, it sets the tree-route flags in the data packets received from the upper layer, and sends them to the destination through the tree route.

A MCN source cannot initiate the route discovery process because it is incapable of storing the routing entry for the destination; therefore, it sends them to the destination through the tree route. Before doing that, it resets the tree-route flags in data packets to show its desire for a RAR.

When an intermediate MSN receives a data packet with the tree-route flag unset, it first checks the routing table to see whether it has an existing routing entry for the packet destination. If it has, the packet is forwarded according to the entry. Otherwise, it knows the packet must come from some MCN source. In this case, it serves as the "agent" of the MCN source by initiating the route discovery process. At the same time, it sets the tree-route flag in the data packet and sends it to the destination through the tree route. In the example of Figure 2, Source S, as a MCN,

cannot initiate the route discovery process. It forwards data packets with the tree-route flags unset to node N. N, as a MSN, is able to serve as the "agent" of S and initiate the route discovery process. Before discovering the new RAR, N sets the tree-route flags in the newly received data packets and forwards them by the tree routing to K.

2) Handling of RREQ

Intermediate MSNs rebroadcast the first received RREQs as well as the duplicated ones propagating from better routes. They also create or update the routing entries for the source to enable the reverse routes. If the destination is a MSN, it sends RREPs back to confirm the newly discovered routes.

In order to save control traffic, an intermediate MSN, when finding the destination is its descendant, unicasts the first received RREQs to the destination along the tree route, since the tree route between it and the destination is the shortest route.

When a MCN receives a RREQ, it accepts the RREQ only if the RREQ sender belongs to the tree route between it and the RREQ originator. In all other cases, the received RREQ is discarded. Intermediate MCNs unicast the accepted RREQs along the tree route to the destination. If the destination is a MCN, it sends a RREP back for each accepted RREQ.

In the example of Figure 2, the RREQ from N is acceptable to K since N is in the tree route from K to N. K unicasts the RREQ to H along the tree route. Similarly, J and M can accept the RREQs from G and J. They forward the RREQs to M and T respectively. Node L, however, cannot accept the RREQ relayed by P because P does not belong to the tree route from L to N. Finally a new route S-N-K-H-F-G-J-M-T-D is discovered.

3) Handling of RREP

Intermediate MSNs forward newly received RREPs by consulting reverse routing entries, while MCNs relay RREPs by the tree routing. Upon receiving multiple RREPs, the MSN source or source agent selects the best route and begins to transmit data packets with the tree-route flag unset along the new RAR.

The slim MCN approach is very simple, and as its name shows, it does not add any memory burden to MCNs in the discovery and formation of RARs. However, as cross branch shortcuts can only be established along all MSN nodes, its improvement of route optimization is very limited, especially when big portions of nodes are MCNs.

B. Enhanced MCN approach

In order to make the reactive routing process discover better routes, we propose to relax the memory restrictions at the MCNs and permit MCN endpoints (sources / destinations) to maintain unilateral routing entries. This is a reasonable requirement since MCN endpoints should “pay” more for traffic originate from them or destined for them. As a MCN endpoint is not expected to originate/terminate multiple traffic flows simultaneously, it only needs to reserve a very small memory space with the capacity to store a couple of routing entries.

Let’s go back to the example of Figure 2, and aim at a shorter route S-N-P-L-Q-T-D. The slim MCN approach cannot establish such a route because node L is not able to record the next-hop node Q. One way to inform L about Q is to put Q’s address in data packets. In this approach, we add a field called “next-MSN address” in each data packet to do this job. This new field is first filled by sources and updated by intermediate MSNs. In the route discovery stage, sources and intermediate MSNs are required to record the addresses of the next-MSNs. In other words, even a MCN should create a routing entry when it is the source of a RAR. The destination of a RAR, as the originator of the reverse traffic, is also demanded to maintain a routing entry to store the previous-MSN address.

To assist the new approach, a “tree-link” flag is added to each routing entry. The “next-hop address” field in the routing entry is also renamed to “next MSN address”.

1) Handling of data packets

As MCN sources are able to create routing entries now, they should have the same functions as the MSN sources. In the following, we assume sources have decided to create or employ RARs to deliver messages.

When a source does not have an entry for the desired destination, it initiates the route discovery process by broadcasting a RREQ. At the same time, it forwards the currently generated data through the tree route.

If a source or an intermediate MSN has an existing routing entry, it first copies the next-MSN address from its entry to the “next-MSN address” field of the received data packets. Then, it checks the tree-link flag in the entry. If the flag is set, the data packet is relayed along the tree route toward the next-MSN; otherwise, the packet is sent to the next-MSN directly.

An intermediate MCN relays every data packet along the tree routes toward the next-MSN, whose address is embedded in the data packet.

2) Handling of RREQ

We add three new fields to the original RREQ messages:

Tree-link flag – indicates whether a RREQ incoming link should be a tree-link or not.

Previous-MSN address – the address of the previous-MSN in the reverse route.

Anti-loop address – The address of the node that forwards the RREQ to the most recent RREQ sender. This field is designed to prevent the RREQ loop between two adjacent MCNs, which do not maintain RREQ tables.

The RREQ broadcast from a source has the source address as the "previous-MSN address" as well as the "anti-loop address". The tree-link flag is not set.

When an intermediate MCN receives a RREQ, it first checks the "anti-loop address". If the address is the same as its own address, the RREQ is discarded as a loop-back packet. Otherwise, it sees whether the RREQ sender is its parent or child, if not, the RREQ is discarded. For those RREQs that pass the aforementioned checks, the MCN sets their tree-link flags, updates the "anti-loop addresses" with the addresses of the latest RREQ senders, and broadcasts them out.

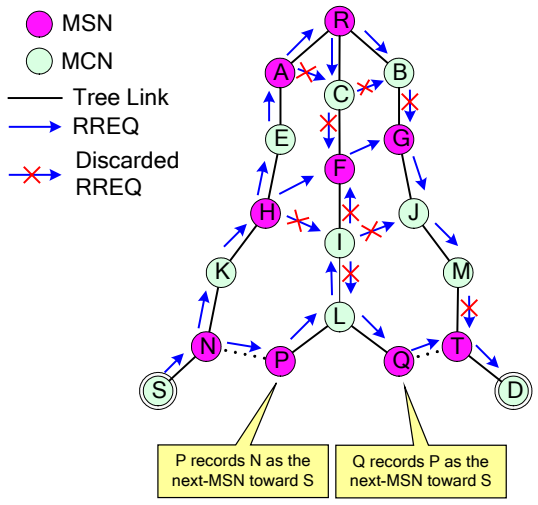

Figure 3. Enhanced MCN approach – RREQ handling

A MSN only processes the first received RREQ or a duplicated one coming from a better route. When a MSN receives a RREQ that satisfies above conditions, it checks the tree-link flag in the packet as well as the packet incoming-link, if the flag is set but the RREQ is not from a tree-link, it discards the packet silently; in all other cases, the packet is accepted. Once a MSN decides to accept a newly received RREQ, it creates a reverse routing entry for the RREQ originator or updates an existing one. Both the previous-MSN address and the tree-link flag have to be copied to the corresponding fields in the entry. If the MSN is not the destination, it resets the tree-link flag in the RREQ, updates the previous-MSN address with its own address, and rebroadcasts the packet out. If the MSN is the destination, it discards the RREQ and unicasts a RREP back to the source. MCN destinations handle RREQs the same way as MSN destinations.

In the example of Figure 3, as P is the child of L, L can now accept the RREQ broadcast from P. L rebroadcasts the RREQ with the tree-link flag set. Node Q, as another child of L, can accept the RREQ from L and rebroadcasts it to T. Finally, the RREQ reaches D. Therefore, a new route S-N-P-L-Q-T-D is discovered.

3) Handling of RREP

We also add three new fields to the RREP messages:

Tree-link flag – indicates whether a RREP incoming link is a tree-link or not.

Previous-MSN address – The address of the previous-MSN in the reverse route.

Next-MSN address – The address of the next-MSN in the forward route.

A RREP just leaving a destination has the destination address as the "next-MSN address"; the "previous-MSN address" is copied from the reverse routing entry for the source; and the tree-link flag is left unset.

When an intermediate MCN receives a RREP, it sets the tree-link flag in the packet and forwards the packet along the tree route toward the previous-MSN, whose address is contained in the RREP.

A MSN has to create a forward routing entry or update an existing one upon receiving a RREP. Both the next-MSN address and the tree-link flag have to be copied from the RREP to the newly created or updated entry. If the MSN is not the source, it updates the next-MSN address in the RREP with its own address, and resets the tree-link flag in the packet. In order to forward a RREP back correctly, the MSN has to check the tree-link flag in the reverse routing entry. If the flag is set, the RREP is relayed along the tree route toward the previous-MSN. Otherwise, it is transmitted to the previous-MSN directly. Either a MSN source or a MCN source is able to select the best route according to the route costs in the received RREPs.

In the example of Figure 4, the RREP creates a forward entry at Q with T as the next MSN and the tree link flag unset. P creates a forward routing entry with Q as the next MSN and the tree link flag set. When receiving a data packet destined for D, P checks the tree-link flag in the forward routing entry. Since the flag is set, P copies Q's

address to the next-MSN address field of the data packet and forwards the packet along the tree route to L. From the data packet, L knows Q is the next MSN. So it forwards the packet along the tree route to Q. Q finds the tree-link flag in its forward entry is unset, so it directly forwards the packet to the next-MSN T.

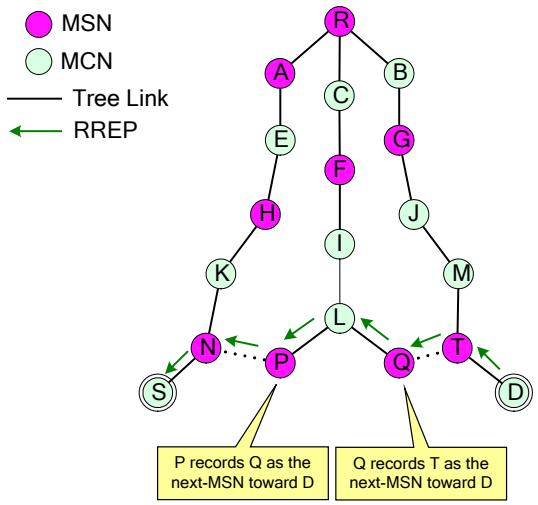

Figure 4. Enhanced MCN approach – RREP handling

The enhanced MCN approach improves route optimization, however, it burdens MCN endpoints and adds on overhead to control and data packets.

IV. SIMULATION RESULTS

We simulate the integrated routing protocols in ns2 and compare their performances with those of AODV and the tree-based routing.

In the simulations, we investigate the route optimization, control overhead and message transmission delays of all the routing protocols in a network composed of 100 nodes. The nodes are uniformly distributed in the network and have the same transmission range of 11m. The network channel capacity is 250 kb/s. We build a spanning-tree to cover the whole network. The simulation results given in the following are the average values obtained from the tests over 20 source-destination pairs.

A. Route Optimization

Figure 5 shows the average lengths of the routes established by using different approaches. Apparently, AODV achieves the best route optimization, however, it can only be applied to a network without any MCN. All other schemes can be used in networks with various MSN/MCN ratios. When the MSN ratio increases, the integrated routing protocols are able to discover shorter routes because more nodes participate in the RAR constructions. Tree routes are always the longest routes. Their lengths are not affected by MSN ratios. Between the two integrated routing schemes, the enhanced MCN approach has much better performance than the slim MCN approach especially when the MSN ratio is high.

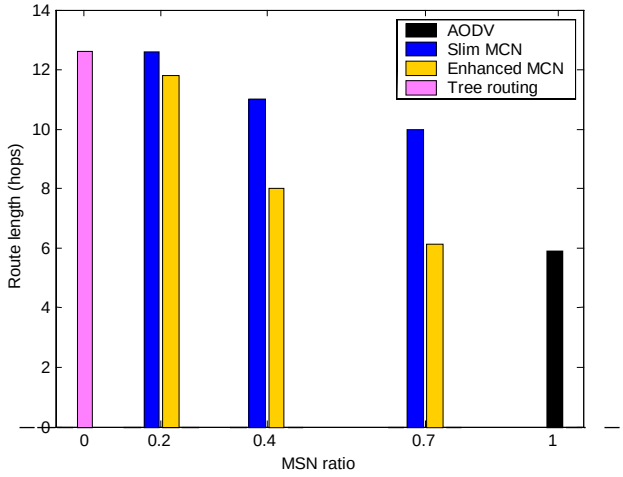

Figure 5. Route Optimization

B. Control Overhead

In Figure 6, the reactive routing overheads of different schemes are presented. The slim MCN approach is shown to generate very low control traffic, while the enhanced MCN approach suffers from large routing overhead. Since MSNs can nicely control RREQ traffic by virtue of RREQ tables and routing tables, how to process RREQs in MCNs is the key to determine overall control overhead. In the slim MCN approach, MCNs have a low RREQ acceptance rate. Besides, the accepted RREQs are unicast by the MCNs to the next-hop neighbors along tree routes. Many nodes are thus screened from the RREQ spread. The enhanced MCN approach increases the RREQ acceptance rate at MCNs, and also, the accepted RREQs are rebroadcast from MCNs, instead of being unicast to some particular neighbors. As MCNs do not have RREQ tables and routing tables to filter duplicated RREQs, many MCNs rebroadcasts a same RREQ for multiple times, which engenders big control traffic.

Unlike the slim MCN approach, the control overhead of the enhanced MCN approach is not simply proportional to the MSN ratio. For instance, when the MSN ratio climbs, more nodes participate in the route discovery process, which increases the routing overhead, however, as the MCN ratio goes down, repeated RREQ relays from MCNs are decreased, so the overhead increase is counteracted.

As the tree routing does not generate any reactive routing overhead, when messages sizes are small, it is more efficient to utilize existing tree routes instead of burdening the whole network with the route discovery activities.

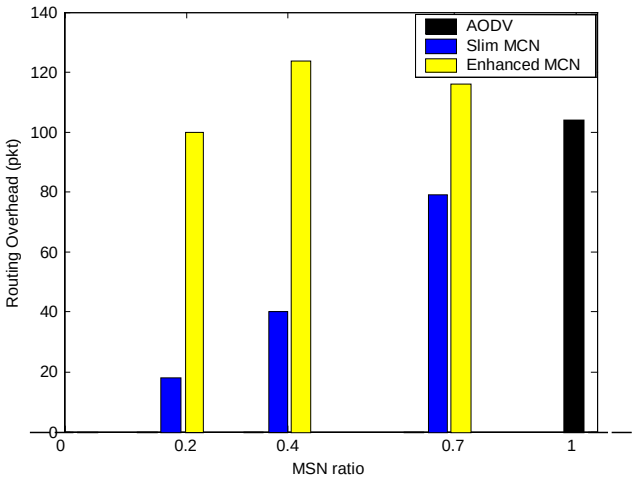

Figure 6. Reactive routing overheads

Table 1. Message transmission delays (second)

Message size (pkt)	Tree Routing	Slim MCN	Enhanced MCN	AODV
1	0.080	0.081	0.082	0.137
16	0.465	0.466	0.463	0.451
50	1.319	1.279	1.256	1.170

C. Message Transmission Delay

Table 1 lists the message transmission delays of different approaches. Since the integrated routing approaches permit sources to forward data packets along tree routes to destinations during the RAR formation periods, the first data packet is shown to quickly propagate from the source to the destination in both approaches. As the integrated routing schemes conduct on-demand route discoveries together with data forwarding, RREQ and RREP transmissions may interfere with the data relays. Table 1 shows that the integrated routing approaches have their first-packet-delays a little larger than that of the tree routing. However, this start-up delay is quickly compensated when new RARs are established. After continuous transmissions of 50 packets, the integrated routing protocols demonstrate their advantages by achieving much shorter message transmission delays than the tree routing. In AODV, the transmission of the first data packet is deferred until a route is completely established. So among all schemes, it incurs the largest first-packet-delay.

The simulation results of message transmission delays suggest again that for short messages, the existing tree routes are better choices.

V. CONCLUSIONS

In this paper, we propose to integrate the tree-based self-routing mechanism with the AODV reactive routing scheme. In our designs, a spanning-tree topology is first formed to interconnect all nodes of a sensor network. Any source within the network has the freedom to utilize existing tree routes or form RARs to reach desired destinations. In this way, the integrated routing can achieve the advantages of both the tree routing and the reactive routing. We develop two protocols to perform on-demand routing in a spanning-tree network composed of both MCNs and MSNs. Our approaches are shown to be good compromises between the tree-based routing and AODV. When applied to all MSN networks, they can achieve the same good performance as AODV. While in all MCN networks, they can tolerate extremely limited memory supplies. More importantly, they enable highly flexible system designs by supporting networks with MSNs and MCNs mixed in any manner.

REFERENCES

[1] L. Hester, Y. Huang, A. Allen, O. Andric, P. Chen, "neuRFon^{TM} Netform: A Self-Organizing Wireless Sensor Network," Proceedings of the 11th IEEE ICCCN Conference, Miami, Florida, Oct. 2002.

[2] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," IEEE Personal Commun., vol. 6, no. 2, Apr. 1999.

[3] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in Proc. ACM SIGCOMM'94, London, U.K., Sept. 1994, pp. 234-244.

[4] C. Perkins, E. Belding-Royer, S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," IETF RFC 3561, Jul. 2003.

[5] I. Chakeres and L. Klein-Berndt, AODVjr, AODV Simplified, ACM SIGMOBILE Mobile Computing and Communications Review, July 2002.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, Vol. 40, No. 8, Aug. 2002.