제27회 한국정보처리학회 춘계학술발표대회 논문집 제14권 제1호 (2007. 5)

센서 네트워크의 실시간 지리 정보 라우팅

카오 민 트랑, 공형윤, 황윤경
울산대학교 전기전자정보시스템공학부
e-mail : {trangcm, hkong, pipilullu}@mail.ulsan.ac.kr

Real time geographic routing in sensor networks

Cao Minh Trang, Hyung Yun Kong, Yun-Kyeong Hwang
Dept. of Electrical Engineering, Ulsan University

Abstract

In this paper we study the problem of real time support in wireless sensor networks and propose a Real time Geographic Routing Protocol (ReGeo) which routes a packet towards the destination based on a compromise between distance and queue count to reduce traffic concentration wherever it has been determined to be too high and uses Gradient Table to store the route satisfying the delay constraints. We describe our prototype implementation of ReGeo Routing in TOSSIM – a TinyOS mote simulator. The simulation results show that the proposed routing protocol not only increases the packet delivery ratio but also keeps overall End to End Delay under a bounded value.

1. Introduction

Advances in micromechanical and computer engineering technology facilitate the development of low cost, low power, multifunctional sensor devices. It is feasible to deploy these small sensor nodes in large numbers, and without preexisting infrastructure. Therefore sensor networks, with their flexible and scalable nature, have great potential for a variety of applications such as battlefield surveillance, monitoring wildlife habitats, tracking vehicles, health monitoring, etc.

Since Wireless Sensor Networks (WSNs) deal with real world, the timeliness requirement is now becoming an important issue for sensor network. In fact, sensor devices are intended to capture data from the physical environment, process them and send them to the base station in order to interpret the data, and consequently take the convenient action. Hence, a sensory data that reaches the base station out of a given delay may have negative effect on the quality of some sensor applications. A lot of real-time sensor applications are being defined. In military or environmental surveillance systems, the sensory data must reach the control monitor in a limited time interval to ensure a fluid tracking in the observed region. Also, for medical care applications, sensor devices are expected to capture real-time vital signs from a large number of patients. In emergency situation, critical sensory data must be displayed at the doctor's control monitor in a limited time so that to take the correct action.

In this paper we develop a real time geographic routing protocol based on Depth First Search (DFS) algorithm for sensor networks. We have implemented our proposal by using TOSSIM, a TinyOS simulator that compiles directly from TinyOS code. This paper is organized as follows. The detail of our protocol is described in Section 2. Experiment results are given in section 3. Section 4 concludes the paper.

2. Regeo protocol

2.1 Overview and Assumption

We now describe the detail of ReGeo. Our protocol has been broken into several components, show in Figure 1.

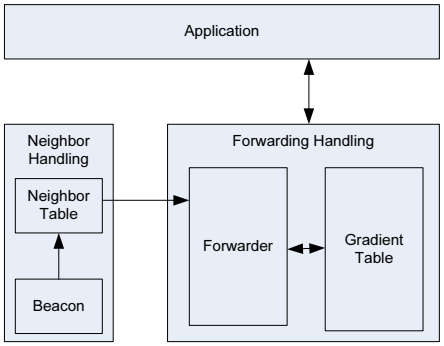

Fig. 1 ReGeo Components.

Packet received from Application Layer will be passed to Forwarding Handling component where it is processed. The Forwarder module in Forwarding Handling

component is responsible for choosing the next node from its Neighbor Table to forward the packet. Each time the next node is known, sensor node will set up appropriate gradient in Gradient Table which is used in future.

Neighbor Handling component stores the information of one hop neighbors in Neighbor Table which is updated by beacons (e.g. HELLO message).

Before we discuss the operation of each component in detail, we state the assumptions:

- The link is bidirectional, i.e. if node A is a neighbor of node B then node B is A's neighbor.
- The time is synchronized on all nodes in networks.
- Each node knows its own location and base station's location.

2.2 Neighbor Handling

Due to the routing decision based on the information of one hop neighbors, it is very important to guarantee that this information is correct and up to date. Each sensor node broadcasts the HELLO message to its neighbors periodically. To avoid collision of the HELLO messages from two neighbors, the sensor node adds a random delay (within the selected range) before broadcasting. The HELLO message includes the identifier of node (ID), location of node and a sequence number. The purpose of the sequence number is to ensure freshness of an HELLO message. Upon receiving HELLO message from node B, node A checks to see if node B is in its neighbor table. If so, it will update new lifetime for B's entry. If not, A will add a new entry for B.

The Neighbor Table includes following fields:

- Neighbor address (NodeID) - This is the identifier of neighbor node.
- Lifetime - This is the interval to wait without receiving anything from a neighbor before considering the link unavailable.
- Queue count - This is the number of packets waiting in a queue to be sent. If this value is zero, there are no packets need to be sent to this neighbor in the queue.
- Sequence Number (SeqNo) - This is the number of the last packet received from that neighbor. This field is used to acknowledge a transmission of a neighbor and to identify packets that are out of sequence.
- Location – geographic position of the node.

2.3 Forwarding Handling

We designed and implemented our protocol based on the Depth First Search (DFS) routing algorithm. In DFS, each node puts its name and address on the route discovery packet p. Then it forwards p to a neighbor who has not seen p before. This neighbor is one of all the neighbors which minimize d(S,y)+d(y,D), where d(x,y) is Euclidean distance between nodes x and y. If a node has no possibilities to forward the packet, it removes its name and address from the packet and returns the packet to the node from which it originally received it. Route discovery packets are kept for some time. If a node receives twice the same packet, it refuses it [6].

Figure 2 illustrates the operation of DFS. Let S be a sensor node generating time sensitive data to be sent to the sink D. Node S checks its Neighbor Table and finds that node A has smallest distance to D. Therefore node S forwards packet to node A. Similarly, the packet is forwarded to node B, C, E. At node E, there is no neighbor which satisfies the forwarding condition of DFS algorithm. Node E will send packet back to node C and node C send to node B. Node B forwards the packet to the next choice H in its sorted list of neighbors, if such a neighbor exists. Finally node H forwards packet to node D.

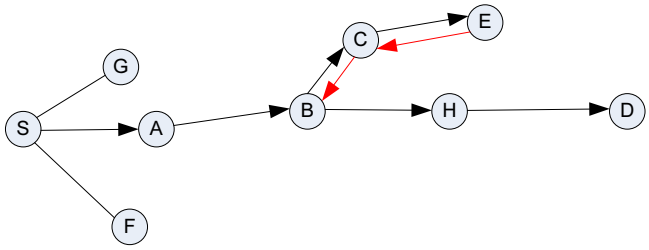

Fig. 2 An example of DFS routing

It is easy to see that DFS does not support QoS. First, the link between sensor node and the neighbors closest to the destination may have high delay. Second, those neighbors can attract much traffic. It results in the increasing of delay and collision probability. Third, in some cases the packet has to go through many unnecessary nodes. It means the end to end delay may be very high. To overcome these problems, we define a cost function as follows:

$$c(A_i, D) = r * \frac{d(A_i, D)}{Max\{d_i\}} + (1-r) * \frac{qc_i}{Max\{qc_i\}}$$

Where

- $c(A_i, D)$: the cost value of neighbor A_i of node A to forward to destination D (in case of broadcasting to region R, D is center of R).
- r: a weight that is generated randomly
- $d(A_i, D)$: the Euclidean distance from A_i to D.
- $Max\{d_i\}$: the largest Euclidean distance among all neighbors of A to D.
- qc_i: the number of messages is waiting in queue to be sent to node A_i.
- $Max\{qc_i\}$: the highest queue count (qc_i) of all neighbors.

The cost function is a compromise between queue count and the distance to destination. It helps to reduce the traffic concentration on nodes closest to destination.

In order to guarantee the end to end delay, we add a delay bound to packet header. Each time node detects the invalidation of delay bound in received packet it will send packet back to the neighbor which sent the packet. The reason we don't include the single hop delay in cost function is that it is difficult to measure this delay exactly.

The use of routing table in sensor network is infeasible since WSN can have a large number of sensor nodes. But it is inefficient if we have to discover the route each time sending packet. Especially, the route can include many unnecessary nodes (e.g C, E as shown in figure 2). To solve this problem we create a Gradient Table with following fields:

- Next Hop – This is the next hop to forward
- Location – The location of destination

- Radius – The radius of region R (equal to 0 for unicast communication)
- Expire time

We now describe the process of forwarding a packet to base station D.

ReGeo packet header includes a flag field indicating whether the packet is in outside forward mode, outside backward mode or inside broadcast mode. Upon receiving a packet P, node A checks its gradient table. If a matching gradient is found out and the queue count corresponding to this node indicated in gradient is not too high, A will forward P to that node. If such a gradient does not exist, packet P will be handled according to one of following cases:

1. **P is an outside forward packet.** A inspects the delay bound field to determine whether or not the deadline of P has expired. If delay bound is negative, P will be updated to backward side mode and immediately returned to node which sent P (i.e. node S). A also checks whether it is destination. If yes, A transmits P to higher layer. If not, Forwarder module memorizes the neighbor that forwarded P in its local cache. We choose node A_i which has smallest $c(A_i, D)$ and A_i is different from S, is the next hop for forwarding. If node A_i exists, A set up a gradient to A_i in Gradient Table, update delay bound field in P and then forward P to A_i. If not, node A changes the flag field in P's header to outside backward and returns P to S.
   Delay Bound = Delay Bound - (Current Time – Sent Time)
2. **P is an outside backward packet.** First it restores delay bound from cache. Then it selects the next choice B in the sorted list of its neighbors according to $c(A_i, D)$, if such a neighbor exists. Assume that A has the set $\{B_1, B_2, \ldots, B_i, \ldots, B_k\}$ is the sorted list neighbors where B_i is the node from which A received P. If $k > i$, the next hop for forwarding will be B_{i+1}. In that case, the flag field will be changed to outside forward. A also updates the gradient to node B_{i+1}. If $k = i$, it will return P to the appropriate node. There are two cases. If node A has not seen P before it will return P to the node which sent P to A. Otherwise it returns P to the neighbor which is memorized above. It will also delete the gradient with respect to destination in Gradient Table. To prevent loop, node will drop packet if it is source node. There is not any route to destination in this case. To eliminate the duplicate packets, sensor nodes only need check their cache when they receive outside forward packets and drop the redundant packets.

Next Hop	Location	Radius	Expire
B	(x_D, y_D)	0	100s

Table 1 Gradient Table at node A

Next Hop	Location	Radius	Expire
H	(x_D, y_D)	0	100s

Table 2 Gradient Table at node B

3. Experimental Results

We have implemented the algorithm in TOSSIM. TOSSIM is a discrete event simulator for TinyOS sensor networks. In order to generate network traffic we created a simple temperature monitoring application. Since every node can communicate with each other by default in TOSSIM, we defined a loss topology manually. All links were defined as having a 0.1% error rate. We can guarantee the bidirection neighbor relationship by using ACK. According to articles on the TinyOS mailing lists ACKs are on in TOSSIM by default. Since we do not consider mobility, the node's coordinates are static. The links between pairs of nodes that can communicate with each other are shown in Fig. 3.

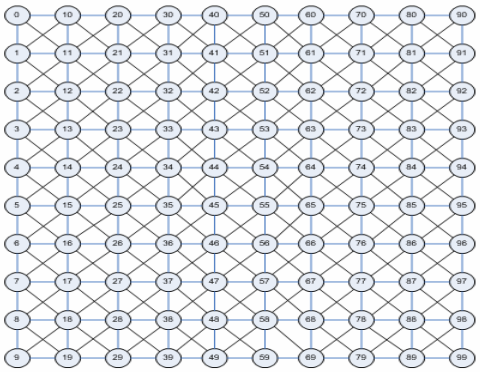

Fig. 3 Network topology

We used the following metrics to evaluate the performance of our protocol.

- **Packet Delivery Ratio**: is defined as ratio of the total over all nodes of the number of data packets received, divided by the total number of data packets sent from the sources.
- **Routing Overhead**: the number of routing packets transmitted per data packet delivered at the destination.
- **End to end delay**: the time taken for a packet to be transmitted across a network from source to destination.

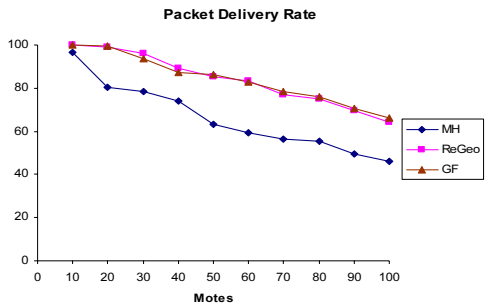

Fig. 4 Packet Delivery Rate vs. Number of Motes

In our evaluation, we compare the performance of our protocol with the existing adhoc multihop (MH) routing protocol in TinyOS and Geographic Forwarding (GF) protocol. MH protocol uses a shortest path first algorithm with a single destination node (the root) and active two way link estimation. The packet delivery ratio of ReGeo is always higher than MH even with the large number of motes (Fig. 4) while the routing overhead of ReGeo is smaller than MH

(Fig .5). Figure 4 shows that ReGeo has approximately 100% delivery ratio when the network has smaller than 20 nodes. It is clear that the number of nodes in a network increase the transmitting successful rate also decreases due to the high collision probability in large networks.

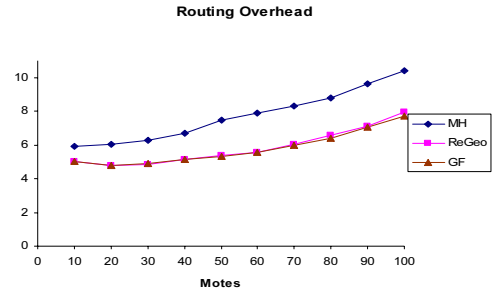

Fig. 5 Routing Overhead vs. Number of Motes

From Fig. 6, we can see that ReGeo has higher End to End Delay than MH. The reason is our protocol doesn't find the shortest route to forward. It chooses the route which has high successful rate and guarantees the total transmit time in the given bound. MH has low end to end delay since it discovered route before sensor node start sending data and a lot of the packets which sent by sensor nodes far from base station are lost in large networks. Therefore, although the End to End Delay of ReGeo may be higher than MH the overall performance of ReGeo is still better.

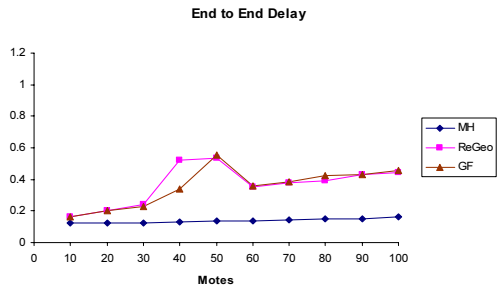

Fig. 6 End to End delay vs. Number of Motes

Because the void problem does not occur in the simulated topology (Fig. 3) the performance of ReGeo and GF in the above figures is almost similar. In order to evaluate the performance of ReGeo in handling void problem of geographic protocol, we changed the loss topology file so that the link between node 1, 11 and node 2 is broken. Looking at the simulation results for a network of 20 nodes it is clear that the ReGeo protocol is the best protocol in terms of successful transmitting rate. ReGeo has 99% packets sent were delivered while GF just has 21%. The reason is that the packets go through node 2 in GF protocol cannot choose next hop for forwarding since there is no closer node to base station (node 0). Hence these packets are dropped. In the case of ReGeo, node 2 can choose node 12 as next hop and continue forwarding process until it reaches to node 0.

4. Conclusions

We have proposed a DFS based routing protocol that provides QoS for sensor networks. The results of our implementation in TOSSIM show that the Packet Delivery Ratio of our routing protocol is improved even with large networks. The route discovered in our protocol may not be the shortest route but it has high successful transmitting rate and the E2E Delay smaller than the required delay. If the route stored in Gradient Table is no longer valid (e.g. delay exceeds the given bound) node will immediately find other node to forward. Therefore, our protocol not only solves successfully the void problem of geographic algorithms but also adapts to the change of topology. Due to the limited available data the evaluation of the protocols was not carried out as well as it might have been. We are working to implement our protocol in Mica Mote to study its performance in a real world.

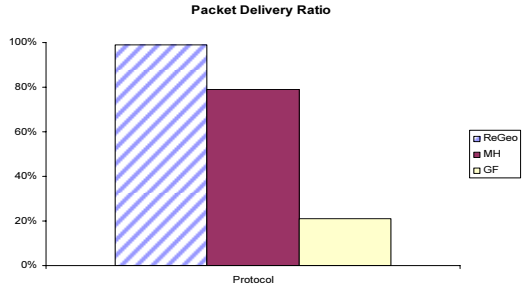

Fig. 7 Packet Delivery Ratio in case of void problem at node 2

Acknowledgement

This research was supported by the Ministry of Information and Communication(MIC), Korea, under the Information Technology Research Center(ITRC) support program supervised by the Institute Technology Assessment(IITA).

Reference

[1] Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks", IEEE ICDCS, May 2003.

[2] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks", RTAS, Jun 2002.

[3] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, J.C. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks", Proceedings of the IEEE, vol. 91, num 7, Jul 2003.

[4] B. Karp and H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," MobiCom 2000, Aug. 2000.

[5] C. Intanagonwiwat, RC. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," 6th Conf. on Mobile Computing and Networking, Aug. 2000, Boston, MA, USA.

[6] I. Stojmenovic, M. Russell, and B. Vukojevic, "Depth first search and location based localized routing and QoS routing in wireless networks", IEEE International Conference on Parallel Processing, Aug. 2000, Toronto.

[7] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," SenSys'03, Los Angeles, CA, Nov. 2003.