Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks

Tsu-Wei Chen and Mario Gerla
Computer Science Department
University of California, Los Angeles
{tsuwei,gerla}@cs.ucla.edu

Abstract

In an ad-hoc environment with no wired communication infrastructure, it is necessary that mobile hosts operate as routers in order to maintain the information about connectivity. However, with the presence of high mobility and low signal/interference ratio (SIR), traditional routing schemes for wired networks are not appropriate, as they either lack the ability to quickly reflect the changing topology, or may cause excessive overhead, which degrades network performance. Considering these restrictions, we propose a new scheme especially designed for routing in an ad-hoc wireless environments. We call this scheme "Global State Routing" (GSR), where nodes exchange vectors of link states among their neighbors during routing information exchange. Based on the link state vectors, nodes maintain a global knowledge of the network topology and optimize their routing decisions locally. The performance of the algorithm, studied in this paper through a series of simulations, reveals that this scheme provides a better solution than existing approaches in a truly mobile, ad-hoc environment.

I. Introduction

In an ad hoc wireless network where wired infrastructures are not feasible, mobility and bandwidth are two key elements presenting research challenges. Mobility causes the life time of a connection between two hosts to vary greatly; and limited bandwidth makes a network to be easily congested by control signalling. Routing schemes developed for wired networks seldom consider restrictions of this type. Instead, they assume that the network is mostly stable and the overhead for routing messages is negligible. Considering the difference between wireless and wireline network, we believe it is necessary to develop a wireless routing protocol that reacts quickly to changes of network topology but consumes only a reasonable amount of bandwidth for the control traffic.

In this paper, we propose a new routing scheme for ad-hoc wireless networks. It is MAC (medium access control) layer efficient because the overhead of control message is kept low. It still provides accurate solutions for finding optimal paths. The rest of this paper is organized as follow. In section II, we survey the existing wireless routing protocols. Then we propose a new routing scheme in section III. Section IV presents the complexity analysis of this scheme and compares it with others. To verify the effectiveness, we simulate a mobile environment and the report the performance results in section V. Lastly, we conclude our work and propose issues for future research in section VI.

II. Previous Work

Several routing schemes based on DBF (distributed Bellman-Ford) [1] or LS (link state) [2] have been proposed in the past for both wireline or wireless networks. The advantages of DBF are its simplicity and computation efficiency due to its distributed characteristic. However, the slow convergence and the tendency of creating routing loops make DBF not suitable for a wireless network with high mobility. Though approaches were proposed in [3, 4, 5, 6] to solve looping problem, none of them overcome the problem of slow convergence.

In part for these reasons, LS is preferred and used in many modern networks like Internet [7] or ATM [8]. In LS, a global network topology is maintained in all routers, and any link change will be updated by flooding immediately. As a result, the time required for a router to converge to the new topology is shorter than in DBF. Since global topology is maintained, preventing routing loop is also easier. Unfortunately, as LS relies on flooding to disseminate the update information, excessive control overhead may be generated, especially when the mobility is high. In addition, because of the large amount of small packets, flooding is also inefficient for the radio MAC layer.

A third routing scheme proposed recently for ad-hoc wireless network is called "on-demand" routing. Namely, the route between two nodes is computed when there is a need. Most on-demand routing are based on a query/response approach [9, 10, 11]. Since flooding is used for query packet dissemination and route maintenance, on-demand routing tends to become inefficient when traffic load and mobility increase.

III. The Global State Routing

Our goal is to design a routing scheme that is MAC efficient in ad-hoc wireless radio networks. That is, the control packet size should be able to achieve optimized MAC throughput, and the number of control packet should be controllable. We prefer to maintain the knowledge of full network topology as in link state routing, but wish to avoid the inefficient flooding mechanism. Therefore, we develop our scheme based on LS, which has the advantage of routing accuracy, and we adopt the dissemination method used in DBF, which has the advantage of no flooding. This scheme is called "Global State Routing" (GSR), and more detailed description is given below.

A. Network Model

The ad-hoc wireless network is modeled as an undirected graph $G = (V, E)$, where V is a set of $|V|$ nodes and E is a set of $|E|$ undirected links connecting nodes in V. Each node has a unique identifier and represents a mobile host with a wireless communication device with transmission range R, and an infinity storage space. Nodes may move around and change their speed and direction independently. An undirected link (i, j) connecting two nodes i and j is formed when the distance between i and j become less than or equal to R. Link (i, j) is removed from E when node i and j move apart, and out of their transmission ranges.

For each node i, one list and three tables are maintained. They are: a neighbor list A_i, a topology table TT_i, a next hop table *NEXT*$_i$ and a distance table D_i. A_i is defined as a set of nodes that are adjacent to node i. Each destination j has an entry in table TT_i which contains two parts: $TT_i.LS(j)$ and $TT_i.SEQ(j)$. $TT_i.LS(j)$ denotes the link state information reported by node j, and $TT_i.SEQ(j)$ denotes the timestamp indicating the time node j has generated this link state information. Similar, for every destination j, *NEXT*$_i(j)$ denotes the next hop to forward packets destined to j on the shortest path, while $D_i(j)$ denotes the distances of the shortest path from i to j.

Additionally, a weight function, *weight*: $E \rightarrow Z_0^+$, is used to compute the distance of a link. Since min-hop shortest path is the only objective in this paper, this weight function simply returns 1 if two nodes have direct connection, otherwise, it returns ∞. This weight function may also be replaced with other functions for routing with different metrics. For instance, a bandwidth function can be used to realize a QoS routing.

B. Algorithm

The details of GSR protocol are listed in Fig. 1. At the beginning, each node i starts with an empty neighbor list A_i, and an empty topology table TT_i. After node i initializes its local variables with proper values as described in procedure *NodeInit(i)*, it learns about its neighbors by examining the sender field of each packet in its inbound queue, *PktQueue*. That is, assuming that all nodes can be heard by i are i's neighbors, node i adds all routing packet senders to its neighbor list, A_i.

Node i then invokes *PktProcess(i)* to process the received routing messages, which contain link state information broadcasted by it neighbors. *PktProcess(i)* makes sure that only the most up to date link state information is used to compute the best route by comparing the embedded sequence number, $pkt.SEQ(j)$, with the ones stored in node i's local storage, for each destination j. If any entry in the incoming message has a newer sequence number regarding destination j, $TT_i.LS(j)$ will be replaced by $pkt.LS(j)$, and $TT_i.SEQ(j)$ will be replaced by $pkt.SEQ(j)$.

After the routing messages are examined, node i rebuilds the routing table based on the newly computed topology table and then broadcasts the new information to its neighbors. Such process is periodically repeated.

C. Information Dissemination

The key difference between our GSR and traditional LS is the way routing information is disseminated. In LS, link state packets are generated and flooded into the network whenever a node detects topology changes. GSR doesn't flood the link state packets. Instead, nodes in GSR maintain the link state table based on the up to date information received from neighboring nodes, and periodically exchange it with their local neighbors only. Information is disseminated as the link state with larger sequence numbers replaces the one with smaller sequence numbers. In this respect, it is similar to DBF (or more precisely, the DSDV [4]) where the value of distances is replaced according to the time stamp of sequence number.

D. Shortest Path Computation

FindSP(i) creates a shortest path tree rooted at i. In principle, any existing shortest path algorithm can be used to create the tree. In this paper, however, the procedure listed in Fig. 1 is based on the Dijkstra's algorithm [12] with modifications so that the next hop table (*NEXT*$_i$) and the distance tables(D_i) are computed in parallel with the tree reconstruction.

At node i, *FindSP(i)* initiates with $P = \{i\}$, then it iterates until $P = V$. In each iteration, it searches for a node j such that node j minimizes the value of $(D_i(k) + weight(k, j))$, for all j and k, where $j \in V - P$, $k \in A_i$ and *weight(k,j)* $\neq \infty$. Once node j is found, P is augmented with j, $D(j)$ is assigned to $D(k) + weight(k, j)$ and *NEXT*$_i(j)$ is assigned to $next_i(k)$. That is, as the shortest path from i to j has to go through k, the successor for i to j is the same successor for i to k.

IV. Complexity

In this section, we analyze the complexity of the GSR scheme and compare it with two other routing schemes: DBF and LS. The complexity is studied under five aspects:

1. Computation Complexity (CC): the number of computation steps for a node to perform a routing computation after an update message is received;
2. Memory Complexity (MC): the memory space required to store the routing information;
3. Data Complexity (DC): the aggregate size of control packets exchanged by a node in each time slot;
4. Packet Complexity (PC): the average number of routing packets exchanged by a node in each time slot;
5. Convergence Time (CT): the times requires to detect a link change.

Protocol	CC	MC	DC	PC	CT
GSR	$O(N^2)$	$O(N \cdot d)$	$O(\|E\|)/I$	$O(1)$	$O(D \cdot I)$
LS	$O(N^2)$	$O(N^2)$	$O(\|E\|)/I$	$O(N)$	$O(D)$
DBF	$O(N)$	$O(N)$	$O(N)/I$	$O(1)$	$O(N \cdot I)$

Table 1. Complexity Comparison

Table 1 shows the results of our comparison. In the table, N denotes the number of nodes in network ($|V|$), D denotes the maximum hop distance, the diameter, in the network, d and I denote the degree of node connectivity and the routing update interval, respectively.

GSR and LS have same memory complexity and computation complexity as both maintain the topology for the whole network and use Dijkstra's algorithm to compute shortest path routes. Dijkstra's algorithm requires typically $O(N^2)$ steps to compute the shortest paths from one source to all destinations, although it is possible to reduce it to $O(NlogN)$ [12]. $O(N^2)$ memory space is required to store the network topology represented by a connection matrix. As for DBF, it has complexity of $O(N)$ for computing and memory, as it only keeps the distance information for each destination, and computes shortest paths in a distributed fashion.

For the data complexity, in GSR each node broadcasts information for $N \times d$ links on average, and the complexity is divided by I, the update interval. LS, on the other hand, has similar accumulated data size for each link update, but its update interval I may become extremely small when mobility increase. This issue, to be addressed shortly, was verified through simulation.

In addition, as LS transmits one short packet for each link update, its packet complexity can be as high as $O(N)$ when the mobility is high. On the other hand, both GSR and DBF transmit a fixed number of update tables using longer packets to optimize the MAC throughput.

Lastly, the convergence time for GSR is also superior than that for DBF. In fact, if shorter update interval is used, GSR can converge as fast as LS.

V. Simulation

Unlike in [5, 4, 9, 10], where wireless network is simulated by static network with higher link failure rate, we used a truly mobile environment in our simulator to determine the connectivity among mobile hosts. The simulation is programmed in C++ to simulate an environment of 500×500 unit^2. Arbitrary numbers of nodes, representing the mobile hosts, move independently on their own trajectories within this virtual space. The maximum moving speed and the number of nodes are given at run time.

Additional assumptions used in our simulations includes:
(1) no node failure during simulation;
(2) node number is always constant in the run time of simulation;
(3) a time slotted system;
(4) radio transmission range is fixed at R, which is specified at the beginning of the simulation;
(5) two nodes can hear each other if they are within the transmission range, that is, open space channel model is used.

Three routing schemes: DBF, LS and GSR are used exclusively in the simulation. The DBF and LS are based on the schemes described in [13]. Both DBF and GSR can be executed with a routing update interval (I) specified at run time. By default, I is set to 3 (one update per three time slots), while in LS, nodes flood link state packets whenever they detect changes in their local connectivities. Also, the number of nodes in our simulation is set to 60.

A. Performance Measurements

Two metrics are used to evaluate the routing performances: routing inaccuracy and control overhead. Using them, we examine the impact to the performances of different mobility, update interval and radio transmission range.

A.1. Routing Inaccuracy Routing inaccuracy is checked by comparing the next hop table of each node with the tables generated by an off-line algorithm. This off-line algorithm has knowledges of the exact network topology to compute the optimal solution for each node at each time slot. For a destination which is still far away, an incorrect value in the next hop table is less critical than nodes that are close by. Considering this, we define the routing inaccuracy for node i, A_i, as:

$$A_i = \frac{1}{D} \sum_{next_i(k) \neq next^i_M(k)} (D - hop_i(k) + 1)$$

then the overall routing inaccuracy is computed by averaging A_i, for all $i \in N$, where N, $next_i()$, $hop_i()$, D are defined in section III, and $next^i_M()$ is the next hop table computed by the off-line algorithm.

A.2. Control Overhead The control overhead is evaluated by examining the average number of routing control packets exchanged on each link. The reason for using the number of control packets instead of the total control bits exchanged is due to the characteristic of radio devices and MAC layer protocol. It is known that a radio device spends considerable time to switch from receive to transmit mode. This typically exceeds the time used for sending a small packet. If spread spectrum is used, the acquisition time may become even more significant.

For LS, we account for each link state packet that is generated by a node either because it detects a topology change, or it receives one from its neighbors and forwards it by flooding. Each of packet requires a transition for radio device from receiving mode to transmitting mode. For DBF type algorithm, a routing table update is counted as one packet. This is under the assumption that the routing table can be transmitted in a fixed number of consecutive MAC layer frames (without transmit/receive switching.)

B. Simulation Results

In addition to routing accuracy and control overhead, we also examine the impact to performance due to changes in mobility, update interval and radio transmission range. These results are summarized below.

B.1. Routing Inaccuracy Fig. 2 shows the inaccuracy of different routing schemes at different node speeds. LS performs best at all speed ranges, since it reacts the fastest to the topology changes. GSR performs less accurately than LS since it updates the routing information only every three time slots. However, GSR still performs better than DBF.

B.2. Control Overhead As shown in Fig. 3, both DBF and GSR algorithms have a flat distribution of packet overhead, which means the overhead of both cases remains constant regardless of mobility. This is because nodes in both schemes exchange routing information periodically with only their adjacent neighbors. On the other hand, with LS schemes, the overhead is much worse than DBF and GSR which means more packets are generated. The figures also show that as degree of mobility becomes higher, the overhead for LS increases. This validates that LS is not suitable for high mobility environment.

B.3. Mobility Impact As Fig. 3 shows, the control overhead for LS increases rapidly as nodes move at higher speeds. An unmanageable flood of packets overwhelms the radio channel and dominates packet queue in each node. On the other hand, mobility has no effect on control overhead for DBF and GSR. This is reasonable because in LS, routing updates are event driven: a node sends a link state packet into the network whenever changes in its neighborhood are detected. And a large amount of these link state packets will then be generated due to the flooding mechanism.

The impact of mobility to routing inaccuracy is, however, independent of the impact to control overhead. Overall, higher mobility causes higher inaccuracy for all three schemes. LS performs the best in every mobility value, as indicated by Fig. 2. LS sustains inaccuracy equal to or lower than 15% even at a node speed of 160 units per time slot, while DBF provides poorly acceptable routing solutions. Our GSR performs between DBF and LS; in low speed range, GSR is as accurate as LS; while in high speed range, GSR becomes worse but is still better than DBF.

B.4. Update Interval Update interval also plays an important role for the routing overhead and inaccuracy. As we showed earlier, LS achieves higher routing accuracy because update packets are sent out immediately whenever a node detects topology change. Thus the delay is bounded by the minimum time resolution used by the system. Fig. 4 shows that GSR inaccuracy is degraded or improved by adjusting the routing interval up or down. The same holds for DBF as shown in Fig. 5, except that the improvement is not very significant as the accuracy is already poor even in the low mobility conditions. As expected, we note that more improvements can be observed when mobility is high.

B.5. Radio Transmission Range The range of radio transmission determines the degree of node connectivity. As shown in Fig. 6, the larger the transmission range the larger the connectivity degree and the larger the control packet size for GSR and LS.

Fig. 7 shows the decrease in routing error rate as transmission range increases. Larger transmission range means more nodes can be reached in one hop without requiring routing decision. However, as indicated in [14], the spatial reuse is less efficient when transmission range is large. It is interesting to note that, the worst case doesn't happens when $R = 80$, which is the smallest range in our simulation. Instead, it happens at about $R = 150$, regardless of mobility. It is obvious that as transmission range increases, the hop distance between any two nodes also decreases, so less routing error may be formed. On the other hand, as transmission range decreases, the graph will become disjoint. To the extreme, when the transmission range equals zero, no two nodes can talk to each other, and the next hop entries will becomes all empty.

VI. Conclusion

In this paper, we introduce a new routing scheme, the Global State Routing, to provide an efficient routing solution for wireless, mobile networks. The routing accuracy of GSR is comparable to an ideal LS scheme and thus superior to the traditional DBF, although it doesn't require individual link state broadcasting which may cause serious consumption of wireless bandwidth. As a result, GSR is more desirable for a mobile environment where mobility is high and bandwidth is relatively low.

Our future plan includes the evaluation of MAC layer impact to the routing efficiency. This requires enhancements to our simulators with different MAC layers such as MACA, MACA/PR and Cluster/TDMA [14]. QoS routing for multimedia support in wireless environment is also considered to be embedded in GSR since it provides higher routing accuracy without sacrificing bandwidth. Besides simulation, implementing GSR in an IP wireless testbed is also in progress.

References

[1] D. Bertsekas and R. Gallager, *Routing in Data Networks*, chapter 5, Prentice Hall, second edition, 1987.

[2] J.M. McQuillan et al., “The new routing algorithm for the ARPANET,” *IEEE Transaction of Communications*, vol. 28, no. 5, pp. 711,719, May 1980.

[3] S. Murthy and J.J. Garcia-Luna-Aceves, “A routing protocol for packet radio networks,” in *Proc. IEEE Mobicom*, Nov. 1995, pp. 86–95.

[4] C.E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers,” in *ACM SIGCOMM'94*, 1994, pp. 234–244.

[5] S. Murthy and J.J. Garcia-Luna-Aceves, “A routing protocol for packet radio networks,” in *Proc. IEEE Mobicom*, Nov. 1995, pp. 86–95.

[6] C. Hedrick, “Routing Information Protocol,” in *IETF RFC 1058*, 1988.

[7] J. Moy, “OSPF Version 2,” in *IETF RFC 1583*, 1994.

[8] The ATM Forum, “Private Network-Network Interface Specification v1.0,” 1996.

[9] M. S. Corson and A. Ephremides, “A destributed routing algorithm for mobile wireless networks,” *ACM-Baltzer Journal of Wireless Networks*, vol. 1, pp. 61–81, Jan. 1995.

[10] V. D. Park and M. S. Corson, “A highly adaptive destributed routing algorithm for mobile wireless networks,” *IEEE Infocom*, 1997.

[11] C.-K. Toh, “A novel distributed routing protocol to support ad-hoc mobile computing,” in *IEEE IPCCC*, 1996.

[12] R. Sedgewick, *Weighted Graphs*, chapter 31, Addision-Wesley, 1983.

[13] Andrew S. Tanenbaum, *Computer Networks, Third Edition*, Prentice Hall, 1996.

[14] M. Gerla and J. T. Tsai, “Multicluster, mobile, multimedia radio network,” *ACM-Baltzer Journal of Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

```
proc $NodeInit$... 
```