Fisheye State Routing in Mobile Ad Hoc Networks *

Guangyu Pei
Computer Science Department
University of California, Los Angeles
405 Hilgard Avenue
Los Angeles, CA 90095
email: pei@cs.ucla.edu

Mario Gerla
Computer Science Department
University of California, Los Angeles
405 Hilgard Avenue
Los Angeles, CA 90095
email: gerla@cs.ucla.edu

Tsu-Wei Chen
Bell Laboratories
Lucent Technologies
600 Mountain Avenue
Murray Hill, NJ 07974
email: tsuwei@research.bell-labs.com

Abstract

In this paper, we present a novel routing protocol for wireless ad hoc networks – Fisheye State Routing (FSR). FSR introduces the notion of multi-level fisheye scope to reduce routing update overhead in large networks. Nodes exchange link state entries with their neighbors with a frequency which depends on distance to destination. From link state entries, nodes construct the topology map of the entire network and compute optimal routes. Simulation experiments show that FSR is simple, efficient and scalable routing solution in a mobile, ad hoc environment.

1 Introduction

As the wireless and embedded computing technologies continue to advance, increasing numbers of small size and high performance computing and communication devices will be capable of tetherless communications and ad hoc wireless networking. An ad hoc wireless network is a self-organizing and self-configuring network with the capability of rapid deployment in response to application needs. An important characteristic which sets ad hoc networks apart from cellular networks is the fact that they do not rely on a fixed infrastructure. Ad hoc networks are very attractive for tactical communication in military and law enforcement. They are also expected to play an important role in civilian forums such as convention centers, conferences, and electronic classrooms. Mobility, potentially very large number of mobile nodes, and limited resources (e.g., bandwidth and power) make routing in ad hoc networks extremely challenging. The routing protocols for ad hoc wireless networks have to adapt quickly to the frequent and unpredictable changes of topology and must be parsimonious of communications and processing resources.

*This work was supported in part by NSF under contract ANI-9814675, in part by DARPA under contract DAAB07-97-C-D321 and in part by Intel.

In this paper, we introduce a new routing scheme for ad hoc wireless networks. It is a link state based routing protocol which is adapted to the wireless ad hoc environment. The rest of the paper is organized as follows. In section 2, we survey the existing wireless routing schemes. We describe the Fisheye State Routing (FSR) in section 3. Section 4 presents the performance results and we conclude our paper in section 5.

2 Brief Review of Routing Protocols

Existing wireless routing schemes can be classified into three categories according to their design philosophy: (a) distance vector based; (b) link state based; (c) on demand. Historically, the first type of routing scheme used in early packet networks such as the ARPANET was the distance vector type. The main advantages of the distance vector approach are simplicity and computation efficiency. However, this approach suffers from slow convergence and tendency of creating routing loops. While several approaches were proposed which solve the looping problem [17, 15], none of them overcome the problem of slow convergence. The solutions to both convergence and looping come in the form of the Link State (LS) approach. LS is the preferred scheme for wired nets (e.g., Internet [14] or ATM [1]). In Link State, global network topology information is maintained in all routers by the periodic flooding of link state updates by each node. Any link change triggers an immediate update. As a result, the time required for a router to converge to the new topology is much less than in the distance vector approach. Due to global topology knowledge, preventing routing loop is also easier. Unfortunately, as Link State relies on flooding to disseminate the update information, excessive control overhead may be generated, especially when mobility is high and frequent updates are triggered. In addition, the small update packets make for inefficient use of the wireless MAC layer. When wireless, ad hoc network size and mobility increase (beyond certain thresholds), current proactive routing schemes (i.e., the dis-

tance vector and link state) become infeasible since they will consume a large part of network capacity and node processing power to transmit update control messages just to keep up with the topology changes.

The most recent addition to the family are the on demand routing schemes. These have been specifically introduced in order to overcome some limitations of the proactive protocols in mobile environments. Examples include AODV [18], TORA [16], DSR [6], ABR [20]. The basic idea behind these reactive protocols is that a node discovers a route in an "on demand" fashion, namely, it computes a route only when needed. In on demand schemes, query/response packets are used to discover (possible more than) one route to a given destination. These control packets are usually smaller than the control packets used for routing table updates in proactive schemes, thus causing less overhead. However, since a route has to be entirely discovered prior to the actual data packet transmission, the initial search latency may degrade the performance of interactive applications (e.g., distributed database queries). Moreover, it is impossible to know in advance the quality of the path (e.g., bandwidth, delay etc) prior to call setup. Such a priori knowledge is very desirable in multimedia applications, since it enables more effective call acceptance control. If the route breaks down because of mobility, a packet may need multiple route discoveries on the way to destination. Route caching becomes ineffective in high mobility. Since flooding is used for query dissemination and route maintenance, on demand routing tends to become inefficient when traffic load and mobility are high and network size grows large [10].

A recent proposal which combines on demand routing and conventional routing is Zone Routing Protocol (ZRP) [7, 8]. For routing operation inside a local zone, an arbitrary proactive routing scheme (e.g., distance vector) can be applied. For interzone routing, on demand routing is used. The advantage of zone routing is its scalability, as "global" routing table overhead is limited by zone size. Yet, the benefits of global routing are preserved within each zone. The performance of ZRP is dependent on a key parameter: the *zone radius*. The choice of radius is determined by network characteristics (e.g, node density, relative node velocity etc.) [8], which dynamically change in ad hoc networks. Moreover, the interzone route discovery packets may loop back into zones already queried. This must be avoided to prevent overhead which can be potentially worse than for flooding based queries [8].

With the availability of GPS [11] technology, any of the previous routing protocols can be assisted by GPS location information. For example, LAR [13] is an on demand protocol similar to DSR but it restricts control packet flooding by using location information. DREAM [3] is a location based proactive scheme. Each node in the network periodically exchanges control packets to inform all the other nodes of its location. Each control packet is assigned a *life time* based on the geographical distance from the sender. DREAM sends *short lived* packet more frequently than *long lived* packets due to the so called *distance effect*, i.e., the farther two nodes separate, the slower they seem to be moving with respect to each other. The data packet is broadcast to the nodes in the direction of the destination using only location information stored at the sender.

3 Fisheye State Routing (FSR)

3.1 Topology Representation in FSR

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.

In the FSR routing implementation, 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. $TT_i.SEQ(j)$ denotes the time stamp 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 distance 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.

3.2 Description of FSR protocol

FSR is an implicit hierarchical routing protocol. It uses the "fisheye" technique proposed by Kleinrock and Stevens [12], where the technique was used to reduce the size of information required to represent graphical data. The

eye of a fish captures with high detail the pixels near the focal point. The detail decreases as the distance from the focal point increases. In routing, the fisheye approach translates to maintaining accurate distance and path quality information about the immediate neighborhood of a node, with progressively less detail as the distance increases.

FSR is functionally similar to LS Routing in that it maintains a topology map at each node. The key difference is the way in which routing information is disseminated. In LS, link state packets are generated and flooded into the network whenever a node detects a topology change. In FSR, link state packets are not flooded. Instead, nodes maintain a link state table based on the up-to-date information received from neighboring nodes, and periodically exchange it with their local neighbors only (no flooding). Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers. The FSR periodic table exchange resembles the vector exchange in Distributed Bellman-Ford (DBF) (or more precisely, DSDV [17]) where the distances are updated according to the time stamp or sequence number assigned by the node originating the update. However, in FSR link states rather than distance vectors are propagated. Moreover, like in LS, a full topology map is kept at each node and shortest paths are computed using this map.

In a wireless environment, a radio link between mobile nodes may experience frequent disconnects and reconnects. The LS protocol releases a link state update for each such change, which floods the network and causes excessive overhead. FSR avoids this problem by using periodic, instead of event driven, exchange of the topology map, greatly reducing the control message overhead.

When network size grows large, the update message could consume considerable amount of bandwidth, which depends on the update period. In order to reduce the size of update messages without seriously affecting routing accuracy, FSR uses the Fisheye technique. Fig. 1 illustrates the application of fisheye in a mobile, wireless network. The circles with different shades of grey define the fisheye scopes with respect to the center node (node 11). The scope is defined as the set of nodes that can be reached within a given number of hops. In our case, three scopes are shown for 1, 2 and > 2 hops respectively. Nodes are color coded as black, grey and white accordingly. The number of levels and the radius of each scope will depend on the size of the network.

The reduction of routing update overhead is obtained by using different exchange periods for different entries in routing table. More precisely, entries corresponding to nodes within the smaller scope are propagated to the neighbors with the highest frequency. Referring to Fig. 2, entries in bold are exchanged most frequently. The rest of the entries are sent out at a lower frequency. As a result,

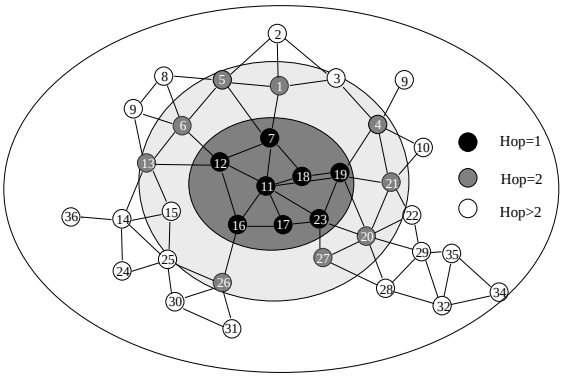

Figure 1. Scope of fisheye

a considerable fraction of link state entries are suppressed in a typical update, thus reducing the message size. This strategy produces timely updates from near stations, but creates large latencies from stations afar. However the imprecise knowledge of the best path to a distant destination is compensated by the fact that the route becomes progressively more accurate as the packet gets closer to destination. As the network size grows large, a "graded" frequency update plan must be used across multiple scopes to keep the overhead low.

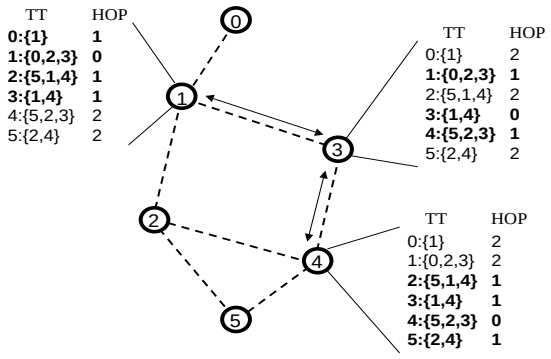

Figure 2. Message reduction using fisheye

The FSR concept originates from Global State Routing (GSR) [5]. GSR can be viewed as a special case of FSR, in which there is only one fisheye scope level and the radius is ∞. As a result, the entire topology table is exchanged among neighbors. Clearly, this consumes a considerable amount of bandwidth when network size becomes large. Through updating link state information with different frequencies depending on the scope distance, FSR scales well

```
proc Node(i) ≡
  NodeInit(i);
  while TRUE do
      if PktQueue ≠ φ   !! packet received
        foreach pkt ∈ PktQueue do
          A_i ← A_i ∪ {pkt.source}
          PktProcess(i, pkt)
        od;
      fi
      CheckNeighbors(i);
      TT_i.LS(i) ← A_i;
      FindSP(i);
      RoutingUpdate(i);
  od
.
proc NodeInit(i) ≡
  foreach j ∈ V
    do
    A_i(j) ← φ;
    D_i(j) ← ∞;
    NEXT_i(j) ← −1;
    SEQ_i(j) ← −1;
  od
  A_i ← A_i ∪ {x | link(i, x)   exists};
  TT_i.LS(i) ← A_i;
  D_i(i) ← 0;
  NEXT_i(i) ← i;
  t_i ← 0;
  SEQ_i(i) ← t_i;
.
proc RoutingUpdate(i) ≡
  t_i ← t_i + 1;
  TT_i.SEQ(i) ← t_i;
  TT_i.LS(i) ← φ;
  foreach x ∈ A_i
    do
    TT_i.LS(i) ← TT_i.LS(i) ∪ {x};
  od
  message.Senderid ← i;
  foreach x ∈ N do
    for ScopeLevel l := 1 to L do
      if ((clock() mod UpdateInterval_l = 0)
        ∧ (D_i(x) ∈ FisheyeScope_l))
        then message.TT ←
            message.TT ∪ {TT_i.LS(x)};
      fi
    od
  od
  broadcast(j, message)   to all   j ∈ A_i;
.
proc FindSP(i) ≡
  Dijkstra's shortest-path algorithm
  P ← {i};
  D_i(i) ← 0;
  foreach x ∈ {j | (j ∈ V) ∧ (j ≠ i)} do
    if x ∈ TT_i.LS(i)
      then D_i(x) ← weight(i, x);
           NEXT_i(k) ← k;
      else D_i(x) ← ∞; NEXT_i(k) ← −1;
    fi
  od
  while P ≠ V do
      foreach k ∈ V − P, l ∈ P do
        Find   (l, k)   such that
        weight(l, k) = min{D_i(l) + weight(l, k)};
      od
      P ← P ∪ {k};
      D_i(k) ← D_i(l) + weight(l, k);
      NEXT_i(k) ← NEXT_i(l);
  od
.
proc PktProcess(i, pkt) ≡
  source ← pkt.source;
  TT_i.LS(j) ← TT_i.LS(j) ∪ {source};
  foreach j ∈ V
    do
    if (j ≠ i) ∧ (pkt.SEQ(j) > TT_i.SEQ(j))
      then begin
          TT_i.SEQ(j) ← pkt.SEQ(j);
          TT_i.LS(j) ← pkt.LS(j);
        end
    fi
  od
.
proc CheckNeighbors(i) ≡
  foreach j ∈ A_i do
    if weight(i, j) = ∞
      A_i = A_i − {j};
    fi
  od
.
```

Figure 3. The FSR Protocol

to large network size and keeps overhead low without compromising route computation accuracy when the destination is near. By retaining a routing entry for each destination, FSR avoids the extra work of "finding" the destination (as in on-demand routing) and thus maintains low single packet transmission latency. As mobility increases, routes to remote destinations become less accurate. However, when a packet approaches its destination, it finds increasingly accurate routing instructions as it enters sectors with a higher refresh rate.

3.3 Pseudo code for FSR

Fig. 3 provides FSR pseudo code. Initially, each node 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 ID of each received packets. 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,pkt)* to process the received routing messages, which contain link state information broadcasted by its neighbors. *PktProcess(i,pkt)* 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. Node i then exchanges the latest link state information with its neighbors. Procedure *RoutingUpdate(i)* scans through the topology table according to the distance $D_i(x)$ between i and x. If $D_i(x)$ is within the range of fisheye scope level l, $TT_i.LS(x)$ will be included in the update message. Note that the update interval for entries which belong to fisheye scope level l is $UpdateInterval_l$. FSR uses different exchange intervals for different entries in the table. To be precise, entries corresponding to nodes that are nearby (within a predefined *scope*) are propagated to the

neighbors more frequently than entries of nodes that are far away.

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. 3 is based on the Dijkstra's algorithm [19] 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.

3.4 Complexity

In this section, we analyze the complexity of the FSR scheme and compare it with that of other three routing schemes: GSR, DBF and LS. Note that the GSR is a the special case of FSR, where we have only one scope and the radius is the the diameter of the network.

The complexity is studied under five aspects:

- **Computation Complexity (CC):** the number of computation steps for a node to perform a routing computation after an update message is received;
- **Memory Complexity (MC):** the memory space required to store the routing information;
- **Line Overhead (LO):** the aggregate volume of control bytes exchanged by a node per unit time.
- **Control Packet(CO):** the average number of routing packets exchanged by a node per unit time.
- **Convergence Time (CT):** the time required to detect a link change.

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. n_i is the average number of nodes in scope i, where $i = 1 \ldots L$ (L is the number of scope levels used in FSR). w_i is the damping factor of update frequency for level i.

FSR 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 $O(NlogN)$ [19]. $O(N^2)$ 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 line overhead, in FSR each node broadcasts information for $n_i \times d$ links on average for nodes in the scope of level i with update interval $w_i I$. So the total line overhead LO for FSR will be $O(\sum_{i=1}^{L} \frac{n_i d}{w_i})/I$. In the case of GSR, w_i is equal to 1 for $i = 1 \ldots L$. So $(\sum_{i=1}^{L} \frac{n_i d}{w_i})/I = (\sum_{i=1}^{L} n_i)d/I = N \cdot d/I$. 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 increases.

In addition, as LS transmits one short packet for each link update, its control packet complexity CO can be as high as $O(N)$ when the mobility is high. On the other hand, both FSR and DBF transmit a fixed number of update tables using longer packets to optimize the MAC throughput. Thus, CO $= O(1)$.

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

4 Performance Evaluation

4.1 Simulation model and methodology

We implement our routing scheme in a multihop, mobile wireless network simulator using the parallel discrete-event simulation language PARSEC [2]. In most of experiments unless specified, the network consists of 100 nodes roaming randomly in a 1000x1000 meter square. The roaming area for network sizes 50, 200, 400 and 1000 is 700x700, 1500x1500, 2000x2000 and 3000x3000 meter square respectively. The radio transmission range is 120 meters and channel capacity is 2 Mbits/sec. A free space propagation channel model is assumed. We use IEEE 802.11 MAC protocol with Distributed Coordination Function (DCF) [9] as the MAC layer in our experiments. The random waypoint model [4, 6] was used in the simulation runs. In this model, a node selects a destination randomly within the roaming area and moves towards that destination at a predefined speed. Once the node arrives at the destination, it pauses at the current position for 5 seconds. The node then selects another destination randomly and moves towards it, pausing

Protocol	CC	MC	LO	CO	CT
FSR	$O(N^2)$	$O(N^2)$	$O(\sum_{i=1}^{L} \frac{n_i d}{w_i})/I$	$O(1)$	$O(D \cdot I)$
GSR	$O(N^2)$	$O(N^2)$	$O(N \cdot d)/I$	$O(1)$	$O(D \cdot I)$
LS	$O(N^2)$	$O(N^2)$	$O(N \cdot d)/I$	$O(N)$	$O(D)$
DBF	$O(N)$	$O(N)$	$O(N)/I$	$O(1)$	$O(N \cdot I)$

Table 1. Complexity Comparison

there for 5 seconds, and so on. Note that the pause time is not considered in computation of node speed. Each simulation executed for 600 seconds of simulation time. Multiple runs with different seed numbers were conducted for each scenario and measurements were averaged over those runs.

4.2 Performance Measurements

The performance measures monitored in our study are: (a) weighted routing inaccuracy; (b) control overhead (O/H); and (c) packet delivery ratio. The variables are: mobility, number of nodes, update interval, and; fisheye scope radius. For FSR, we use 2-level fisheye scoping in our experiments.

4.2.1 Weighted 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 knowledge of the instantaneous network topology and computes optimal routes. For a destination which is far away, an incorrect value in the next hop table is less critical than for a destination close by. Considering this, we define the weighted routing inaccuracy for node i, A_i, as:

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

where D is the diameter of the network. $next^i_M(k)$ and $hop^i_M(k)$ is the next hop and hop distance to destination k calculated by the off-line algorithm respectively. Finally, the overall routing inaccuracy is computed by averaging A_i, for all $i \in N$.

Fig. 4 shows the inaccuracy of different routing schemes at different average speeds. LS performs best at all speed ranges, since it reacts the fastest to topology changes. The inaccuracy for FSR decreases with the increases of scope radius. However, FSR still performs better than DBF even with radius $R = 1$ due to the fact nodes are exchanging link state information among each other and thus keep track of the entire network topology.

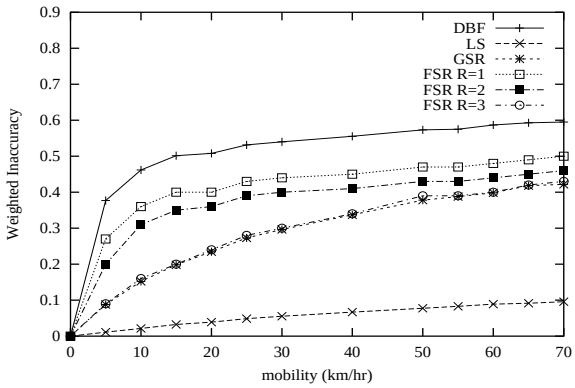

Figure 4. Inaccuracy vs Mobility

The route update interval also has impact on routing inaccuracy. As shown in Fig. 5 (scope radius is 2), FSR accuracy degrades as the update interval increases. Intuitively one would expect that the higher the speed, the smaller the interval to achieve a given accuracy. This is true for low speeds. Above a certain speed threshold (say, 20 km/hr in Fig. 5) the accuracy is relatively insensitive to speed.

Fig. 6 reports the inaccuracy of FSR with different scopes v.s. network size. The mobility is 50 km/hr. For large network size, increasing the radius will improve the accuracy but not very dramatically. This is because in a mobility environment, a change on a link far away from the source does not necessarily cause a change in the routing table at the source. Moreover, the fact that the route error is weighted by distance obviously reduces the sensitivity to network size. Thus, receiving updates about far away nodes at low frequency will not significantly affect the routing accuracy. On the other hand, using larger radii will cause more control O/H as shown in next section.

4.2.2 Control Overhead

Fig. 7 plots link control O/H as a function of network size. Since GSR exchanges full size link state vectors, the control O/H grows linearly with the network size. In contrast, the control O/H is greatly reduced in FSR. Note that the

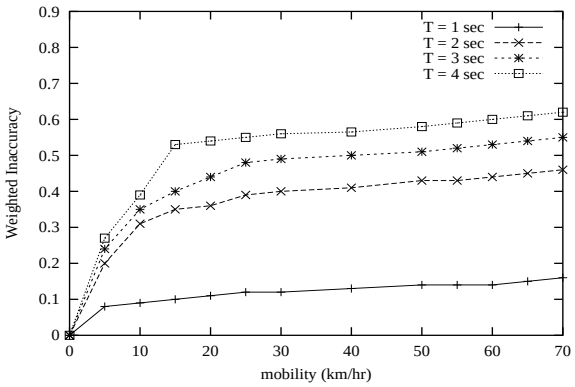

Figure 5. Inaccuracy vs Update Interval

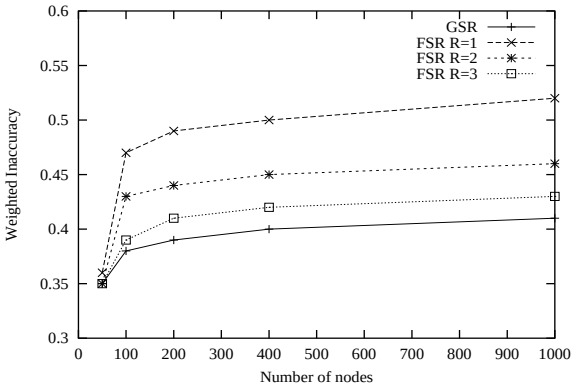

Figure 6. Inaccuracy vs network size

average number of neighboring nodes is independent from network size since node density is kept constant. The reason why FSR reduces O/H is that only a fraction of the entries are updated each time. In a two-level fisheye hierarchy, the smaller radius, the smaller fraction of entries updated in the "fast" interval, and the lower the control O/H. The tradeoff between routing accuracy and control O/H must be taken into account when choosing the scope radii of the fisheye solution.

4.2.3 Packet Delivery Ratio

The packet delivery ratio is the ratio of data packets delivered to the destinations versus data packets originated by the sources. This number presents the routing effectiveness of a protocol. Fig. 8 shows the packet delivery ratio as function of node mobility. As node mobility increases, the performance of the Link State is dramatically degraded due to flooding O/H. While the routing tables are maintained accurate, there is not much bandwidth left for data traffic. GSR faces better than LS; in fact, it performs as well as FSR for small to moderate size networks. However, performance

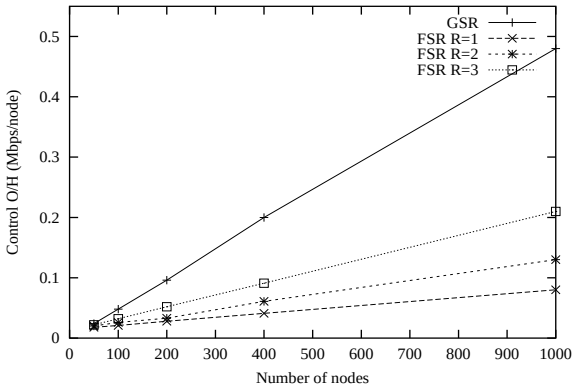

Figure 7. Control O/H vs network size

rapidly degrades for size larger than 100 nodes, as shown in Fig. 9. When network size grows large, routing O/H will cause considerable performance degradation of the GSR. The advantage of FSR is clearly shown as FSR maintains high delivery ratio across different network sizes.

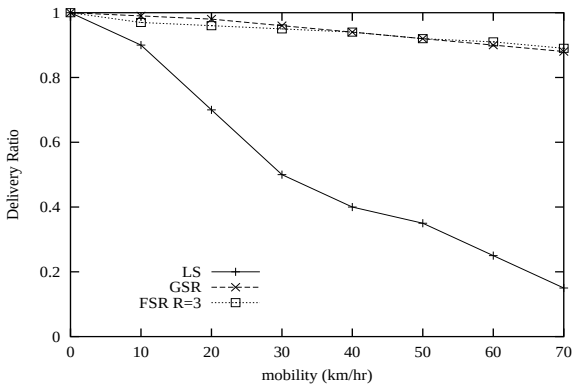

Figure 8. Delivery ratio vs mobility (for 100 nodes)

5 Conclusions

In this paper, we present a new routing scheme, Fisheye State Routing, which provides an efficient, scalable solution for wireless, mobile ad hoc networks. The routing accuracy of FSR is comparable with an ideal LS scheme and the routing overhead is kept low. As a results, FSR is more desirable for large mobile networks where mobility is high and the bandwidth is low. By choosing proper number of scope levels and radius size, FSR proves to be a flexible solution to the challenge of maintaining accurate routes in ad hoc networks.

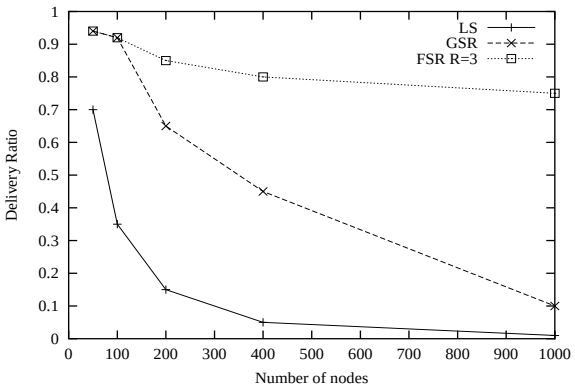

Figure 9. Delivery ratio vs network size (speed = 50 km/hr)

References

[1] The ATM Forum, "Private Network-Network Interface Specification v1.0," 1996.

[2] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H.Y. Song, "PARSEC: A Parallel Simulation Environment for Complex Systems," *IEEE Computer*, vol. 31, no. 10, Oct. 1998, pp.77-85.

[3] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," In *Proceedings of ACM/IEEE MOBICOM'98*, Dallas, TX, Oct. 1998, pp. 76-84.

[4] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," In *Proceedings of ACM/IEEE MOBICOM'98*, Dallas, TX, Oct. 1998, pp. 85-97.

[5] T.-W. Chen and M. Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks," In *Proceedings of IEEE ICC'98*, Atlanta, GA, Jun. 1998, pp. 171-175.

[6] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," In *Mobile Computing*, edited by T. Imielinski and H. Korth, Chapter 5, Kluwer Publishing Company, 1996, pp. 153-181.

[7] Z.J. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," In *Proceedings of IEEE ICUPC'97*, San Diego, CA, Oct. 1997, pp. 562-566.

[8] Z.J. Haas and M. R. Pearlman "Determining the Optimal Configuration for the Zone Routing Protocol," In *IEEE Journal on Selected Areas in Communications*, Aug. 1999, pp. 1395-1414.

[9] IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, NY, 1997.

[10] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable Routing Strategies for Ad-hoc Wireless Networks," In *IEEE Journal on Selected Areas in Communications*, Aug. 1999, pp. 1369-1379.

[11] E.D. Kaplan (Editor), *Understanding the GPS: Principles and Applications*, Artech House, Boston, MA, Feb. 1996.

[12] L. Kleinrock and K. Stevens, "Fisheye: A Lenslike Computer Display Transformation," Technical report, UCLA, Computer Science Department, 1971.

[13] Y.-B. Ko and N.H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," In *Proceedings of ACM/IEEE MOBICOM'98*, Dallas, TX, Oct. 1998, pp. 66-75.

[14] J. Moy, "OSPF Version 2," In *IETF RFC 1583*, 1994.

[15] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM/Baltzer Mobile Networks and Applications*, vol. 1, no. 2, Oct. 1996, pp. 183-197.

[16] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, Apr. 1997, pp. 1405-1413.

[17] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," In *Proceedings of ACM SIGCOMM'94*, London, UK, Sep. 1994, pp. 234-244.

[18] C.E. Perkins and E.M. Royer, "Ad-Hoc On-Demand Distance Vector Routing," In *Proceedings of IEEE WMCSA'99*, New Orleans, LA, Feb. 1999, pp. 90-100.

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

[20] C.-K. Toh, "Associativity-Based Routing For Ad Hoc Mobile Networks," *Wireless Personal Communications Journal*, Special Issue on Mobile Networking and Computing Systems, Kluwer Academic Publishers, vol. 4, no. 2, Mar. 1997, pp. 103-139.