UC Santa Cruz

UC Santa Cruz Previously Published Works

Title

Automatic Routing Using Multiple Prefix Labels

Permalink

https://escholarship.org/uc/item/1vj6t5bm

Authors

Garcia-Luna-Aceves, J.J.
Ghosh, R.

Publication Date

2012-12-03

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Automatic Routing Using Multiple Prefix Labels

Rumi Ghosh
Computer Engineering Department
University of California
1156 High Street
Santa Cruz, CA 95064, USA
Email: rumi@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
Computer Engineering Department
University of California
1156 High Street
Santa Cruz, CA 95064, USA
Email: jj@soe.ucsc.edu

***Abstract*— We present Multi-label Automatic Routing (MAR), the first compact routing protocol that attains a low path stretch (ratio of selected path length to the optimal path length) while maintaining a low routing state for mobile networks. MAR is also resilient to node movements in the network. In MAR, nodes assign themselves labels based on their location in the network through a distributed algorithm. Distributed Hash Tables (DHTs) for the node to label mappings are established in some anchor nodes. Once the labels are established, the routing is automatic based on the positional labels of the nodes and DHT lookups. This eliminates flooding completely. MAR does not need destinations-based routing tabels. Unlike traditional routing protocols. Hence, MAR has a small routing state. With the use of multiple labels per node, the average path length is close to the shortest path and there are multiple paths between source and destination nodes. In Qualnet simulations MAR shows a path stretch close to or better than traditional table-driven and on-demand protocols like OLSR and AODV. Simulation results also show shorter end-to-end delays due to the automatic routing. The delivery ratio of MAR is comparable to these traditional protocols but with a significantly lower network overhead.**

I. Introduction

Traditional approaches for routing in MANETs use node identifiers such as MAC and IP addresses that are not dependent on the location of nodes in the network. This results in the need to flood the network with control messages to find routes to intended destination. The flooding can be link-state (e.g. OLSR [1]) or distances to destination (e.g. DSDV [2]) originated at the destination node by proactive protocols. On-demand protocols such as AODV [4] and DSR [3] incur source-based flooding of routr requests. The flooding of control packets does not scale for large networks or networks with a large number of off-on flows, which is very typical of ad-hoc networks.

The Automatic Incremental Routing (AIR) [5] protocol eliminates the need of flooding of control packets for route discovery by assigning location based labels to all nodes. The drawback of the AIR is that the path length from a source to a destination can be much longer than shortest paths. Also, AIR does not provide multiple paths to destinations which are essential for QoS implementation.

In Section III we present Multi-label Automated Routing (MAR) which is a new compact routing protocol for mobile ad-hoc networks with a low path stretch and multiple paths to each destination. The main contribution of our work is to introduce the use of multiple prefix labels at each node with one label assigned from each one-hop neighbor. This provides multiple paths and reduces the stretch of the path. It also facilitates local repair for node and link failures. MAR routes packets from source to destination using two distributed mechanisms, assignment of routing labels to nodes and dynamic mapping of node ids (IP, MAC address, URL or any other id) to routing labels. Addition of new nodes, node mobility and link or node failures have limited impact on other node labels. Our simulations comparing MAR with OLSR and AODV show that the stretch attained in MAR through the multiple labels is as low as the shortest paths computed by these protocols. The routing state of MAR is far less than these protocols and hence MAR is more scalable.

II. Related Work

The scalability of a network is affected by the size of routing tables at each node. Several compact routing schemes have been proposed where the size of the routing table grows sub-linearly with the number of nodes in the network. However, these protocols usually achieve their scalability at the expense of using paths that can be much longer than the shortest path. Compact routing schemes [6] generate sub-linear routing table sizes, constant stretch and poly-logarithmic labels in general graphs. However, it is hard to have a distributed implementation of these algorithms.

Tribe [8] uses a depth-first approach and partitions the address space into control regions based on intervals of addresses. However, Tribe incurs a lot of re-labeling of nodes for node mobility. DART [7] uses prefix labels to generate clusters of nodes based on prefix address trees. It has the same node-to-cluster affiliation problem of hierarchical routing and hence involves a lot of re-labeling of nodes for node or link failures.

Small State and Small Stretch (S4) [9]is a compact routing protocol for large scale sensor networks that achieves worst-case stretch of 3 and average case stretch of 1 with very little routing state at each node and high failure resilience. S4 maintains shortest paths for nodes inside the cluster for each source. Every node maintains shortest hop count and next hop for each of the many beacon nodes. For destinations outside the cluster, the source routes the packet to the beacon closest to the destination. The beacon routes the packet to the

destination. S4 has a resilient failure recovery by assigning priorities to nodes based on distances from destinations. The key limitation of S4 is that its signaling is not well suited for mobile networks.

III. Protocol Description

MAR is a distributed compact routing scheme. Every node in MAR routes packets based on the routing labels assigned to it and information of its immediate neighborhood. Thus the routing is both *automatic* and *incremental* and does not need elaborate routing tables at each node. In MAR the storage and communication complexities grow sub-linearly with the number of nodes or links in the network.

A. Information Stored and Messages Exchanged

Each node in MAR maintains the two-hop neighborhood information. Each one hop neighbor entry stores the neighbor's identifier, the prefix labels of the neighbor and the newest local sequence number heard from the neighbor. For the two hop neighbors the information stored is the same as one hop neighbors except only the basic prefix tree label is exchanged and stored to reduce control overhead. The anchor node of a node maintains the mapping of the node identifier to the prefix labels of the node.

MAR uses *Hello*, *Anchor Update* and *Anchor Reply* messages. The message *Anchor Request* is implicit in the first data packet sent to an anchor.

The Hello packet is a neighbor to neighbor broadcast message. The Hello messages originate in the root with a new sequence number periodically and propagate in a breadth-first manner. The Hello message carries the *root id*, *root sequence number*, *node id*, *node sequence number*, *node's prefix labels*, *list of one and two hop neighbor id and labels*.

The Anchor Request is sent to the anchor of the destination for lookup of the destination's prefix labels. The source of a flow embeds the Anchor Request message in the first data packet and sends it to the anchor. The anchor replies to the source with a unicast Anchor Reply which has the mapping of the destination node id to the prefix labels. Then the anchor forwards the data to the destination. If any node other than the anchor that is forwarding the Anchor Request, has the requested anchor mapping in its cache, it would send an Anchor Reply directly to the source.

All nodes send periodic Anchor Update messages to their anchors with their prefix labels to refresh the mapping of the node id to the prefix labels.

B. Distributed Root Election

The root election is done distributively in the network. The network has only one root node. When a node comes up, it assigns itself as the root. It communicates its root id with the neighbors and selects a lower root if available from the Hellos it receives. Eventually, the node with the lowest id is elected the root.

C. Basic Prefix Tree Labels

A node in MAR has two unique identifiers. The first identifier is location independent node identifier and the second one is the topology dependent label. As the node moves the first identifier stays the same but the second one gets re-assigned.

From the labeling point of view the network is visualized as a k-ary Labeled Directed Acyclic Graph (LDAG), where k is the degree of the LDAG. Each node in the LDAG is labelled in a breadth-first manner starting from the root. The links between nodes represent neighbor relationships in the actual topology.

If Σ is the finite set of symbols, then the prefix label of a node, l, is a string with symbols from Σ such that $|l| \geq 1$. The root node has the smallest label. Once a node has a prefix label, it assigns a unique suffix s_i to each of its children, i. The child then assigns itself the label $l \odot s_i$, where $\odot$ is the concatenation operator.

The neighbor-to-neighbor Hello message is used to assign labels to a subset of nodes in the network relative to their position from the root node. This generates the LDAG from the elected root node. The LDAG formed by assigning the prefix labels with respect to the root node is called the Basic Prefix Tree (BPT). A source node *S* can reach a destination node *D* by traversing the BPT. The BPT traversal is based on *Maximum Prefix Match* with the neighbor node's label. This results in traversing up the tree from *S* to a common ancestor and then traversing down to *D*. Sometimes the common ancestor might be the root node.

The first label assigned to a node is the Basic Prefix Tree label. The BPT label is label that is an ordered extension of the smallest label of all parents along the BPT. If a label assigned to the node later is found to be smaller in length than the BPT label, the node assigns that as the new BPT label. The parent child relationship of neighbor nodes is defined with respect to the BPT label.

D. Assigning Multiple Labels

A node other than the root can also acquire additional labels from nodes other than its parent in the BPT. A node can accept an additional label from each of its one hop neighbors as long as the neighbor is not in its sub-tree already with respect to that label. Each neighbor node advertises all its labels in the Hello message. A node accepts only one label from a neighbor from all of ones advertised. The node selects the label that is most disjoint from its existing labels as the new label from the neighbor. This ensures the node has the maximum number of node disjoint paths available to reach the destination. Every time the neighbor advertises a new label set, the node tests the disjointness of each of these with the node's current set of labels and accepts the most disjoint and smallest one.

A node can also get a label from any of its children in the LDAG. The node chooses the label that is maximally disjoint from its current set of labels. This automatically ensures that there would not be any loop in labeling. The additional labels from parents and children ensure a node gets as close to shortest path routing as possible.

The advantages of multiple labels is having multiple paths from a node to all other nodes in the network. This ensures shorter automatic paths. It also ensures less traffic through the root node and nodes higher up in the LDAG and less congestion around these nodes. The multiple paths allow for higher fault tolerance. They are also useful in implementing QoS routing which is essential for actual deployments.

E. Publish and Subscribe Operation

A distributed hash table (DHT) is maintained in the network to store the node id to prefix labels mappings. As the name implies, this node lookup table is distributed in all the anchor nodes of the network. A globally known hash function takes the node id as an argument and returns the anchor label. The node publishes itself to the anchor node by piggybacking an Anchor Update with the Hello. As the Anchor Update travels through the network towards the anchor node, the nodes that do not match the anchor's label just forward the update. Only the node that matches the anchor label stores the mapping and becomes the anchor node. The *Publish* operation like the *Hello* takes place on the expiry of a timer.

The *Subscribe* is best-effort. A node subscribes to a destination if it has data to send to that destination. The source hashes the node id of the destination using the globally known hash function to get the anchor label. Then it sends the first data packet to the anchor label. The anchor on receiving the data forwards it to the destination and sends a *Anchor Reply* to the source with the destination's labels. The source sends the rest of the data directly to the destination's labels.

F. Routing

In MAR the packets are routed using the two-hop neighborhood information and destination node's labels. The node finds the longest prefix match from the destination labels and the labels of the one-hop neighbors. Two-hop neighbors are considered only if no next hop is found from the one-hop neighbors.

The use of multiple labels enables the nodes to find multiple paths of same or different length to the destination. While the path of the maximum matching prefix is mostly chosen to have shorter length paths, sometimes longer paths are chosen to avoid congestion around the root. For prefix matches of same length, a next hop is randomly chosen from the matches.

G. Network Dynamics

There are several features in MAR which allow the nodes to find their new labels quickly in case of node movement and node or link failure.

When a new node joins an existing MAR network, it assumes it is the root. If it finds a better root by Hello exchanges with its neighbors, it assigns that node as root and re-labels itself. Then it selects the smallest label offered by its neighbors as its BPT label and stores the additional labels from other neighbors as alternate paths.

If a node n moves and attaches itself at some other point in the network, the node n follows the join procedure. If the node n was a leaf node earlier, none of the other nodes at the old point of contact need to be relabelled. However, if it was an internal node, the nodes in the previous sub-tree have to remove the old label and select the lowest label as the new BPT label. If a parent node moves or goes down, the child node detects it by loss of a few successive Hellos. The old label from the parent is deleted. If the parent is the BPT parent, the child sets its BPT label to the next smallest prefix label. The child node sends all the new labels in the next periodic Anchor Update. This Anchor Update, piggybacked on a Hello, has an incremented local sequence number. The neighbor accepts the Hello only if it has a higher root sequence number or local sequence number.

The caches for DHT entries are flushed on a periodic basis. This timer is several times the Anchor Update period. A neighbor entry is deleted if not heard from the neighbor in a few Hello periods. In case of a deleted neighbor, the corresponding node label is deleted. If the deleted label was the BPT label, a BPT label is assigned .

Once the destination receives the first data packet from the source, it updates the source with its labels periodically in an Anchor Update message. All the intermediate relay nodes between the source and destination nodes that forward this message also store the destination label mapping in their cache. Hence, the source does not have to do a DHT lookup at the anchor if the destination moves. Also any other node that stores the destination label mapping may use it without an Anchor Request, if it needs to send data to the destination.

There is a hysteresis zone of nodes around each anchor node that also maintain the DHT mappings of the anchor. In case the anchor node moves, these nodes still continue to have the DHT mapping for the destination node till the mapping is flushed on a periodic timeout. By that time the destination will find a new anchor node to store its DHT mapping. This is to ensure the DHT requests by the source do not fail in the transition time when the new anchor takes over due to movement of the old anchor node.

In case of a link failure, a node does a local repair through its alternate multiple paths. If the node finds an alternate path, it retransmits the packet. The link failure is detected at the MAC.

IV. EXAMPLE

Figure 1(a) shows an ad hoc network of 14 nodes. Node *a* has been elected the root as per the root election algorithm and is assigned the label "0". Figure 1(a) shows the Basic Prefix Tree labels with respect to the root *a* in red. A node selects the lexicographically smallest label advertised by its neighbors and adds a unique prefix to it to generate its BPT label. This neighbor is called a parent of the node. The basic prefix LDAG edge shown in solid black, points from the parent to the child. Once the basic prefix LDAG has been generated, all neighbors of the node having smaller labels than the node are parents of the node. All other neighbors are defined to be children of the node. For example, node *e* is the basic prefix parent of node *l* and node *k* is another parent. Node *j* is a child of node *e*.

Once the LDAG gets established, a node can get additional labels from other parents. In figure 1(b) these labels have been shown in blue and the edges are shown as dotted lines. Node l gets the additional label "331" by appending suffix "1" to the label of k. The additional label is selected so that it is maximally disjoint from the current labels and is the smallest possible. In the case of l, The label "33" of k is more lexicographically different from its current label "211" than the other label "23" of k. Similarly, h gets the additional label "42" from d.

Ideally each node should have one label per root node neighbor as that would provide a node with many alternate paths to every other node. This would be helpful in reducing the stretch of the path to make it closer to the shortest path. Sometimes a node might get additional labels from its children if it does not have enough labels from its parents alone. The rules of additional labels from children is same as that from parents, the new label has to be maximally disjoint from the existing labels and smallest possible. Figure 1(c) shows the additional labels nodes could get from their children in green. Node x is a neighbor of the root node and has a label from three other neighbors of the root node, starting with "2", "3" and "4". All three of these labels have been generated from its children. Sometimes a node might refuse labels from its children because it does not benefit from them. For example b does not accept labels from children f and g because b already has labels starting with "3" and "4". A node does not benefit from redundant labels because they unnecessarily increase the size of the Hello message. In our design a node selects only one label from each neighbor.

The root node already knows the labels of its one-hop and two-hop neighbors from the Hello messages sent by neighbors of the root. So the root does not get any additional labels from any neighbor.

The advantage of prefix diversity is the nodes can select paths that are closer in length to the shortest path. For example in figure 1(c) the path from h to g through the basic prefix tree is $h \rightarrow x \rightarrow a \rightarrow b \rightarrow g$. When used multiple labels, the path reduces to $h \rightarrow d \rightarrow i \rightarrow g$, which is the shortest path. Another inherent advantage is, routes through the root can be avoided by choosing next-hops that are not on the basic prefix LDAG. For example, to reach f from d, the path $d \rightarrow i \rightarrow g \rightarrow f$ can be chosen instead of $d \rightarrow a \rightarrow b \rightarrow f$. This prevents congestion around the root.

V. Performance Comparison

We have compared MAR with traditional protocols like OLSR and AODV and with the compact routing protocol AIR. The Qualnet simulator version 5.0 has been used for the simulations. We have IEEE 802.11 DCF as the MAC protocol at 2 Mbps bandwidth. Random way point mobility with speed of 10 m/s has been used for mobile scenarios. 25, 50, 75, 150 and 200 nodes have been simulated in different scenarios.

The metrics used are delivery ratio, end to end delay, path length and network load. The delivery ratio is the ratio of number of CBR packets received by the destinations to the number of CBR packets sent by the sources. End to end delay is the one-way delay between the time a CBR source sending the packet and the destination receiving it. Path length is the average number of hops traversed by each data packet. Network load is the control overhead per node.

TABLE I
Performance comparison

Metric	MAR	OLSR	AIR	AODV
Path Length	2.123396	3.194472	1.557749	2.605133
	1.022468	1.043889	1.011155	1.032103
	3.224324	5.345054	2.104344	4.178164
Delivery Ratio	0.773939	0.624877	0.229023	0.834156
Network Load	756.44	2994.28	1514.68	149.08
Delay	0.019243	0.020895	0.017588	0.021600

Table I shows the delay, delivery ratio, network load and path length for 25 nodes and 2 CBR flows in a network of size 900m x 900m. Each flow generates 256 byte packets at 5 packets/sec. The 95% confidence interval has been shown for path lengths for the data packets. The results clearly demonstrate how MAR finds shortest paths to destinations while maintaining lower delay and network load compared to OLSR and AODV. AIR shows a lower value of path length than MAR. But in AIR most packets through the longer paths get dropped due to overload near the nodes higher up the prefix tree. This is evident from the very low delivery ratio.

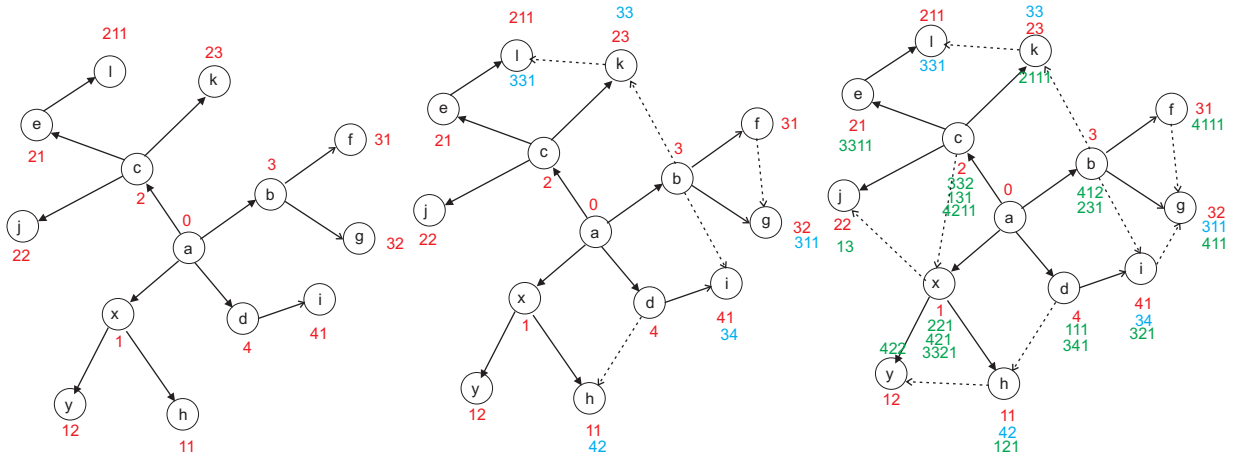
In the scenarios below the simulations have been run for multiple random node placements by altering the seed value of the simulator. The mean performance of several runs with 95% confidence interval has been reported.

A. Static configuration with increasing number of nodes and exponential flows

The CBR flows have exponential arrival times. The mean inter-arrival time for flows is 10 sec and mean flow duration is 200 sec which one-third of the simulation duration of 600 sec. At any instant we have about 20 CBR flows. Each CBR flow generates 256 byte packets at 5 packets/sec. The network size is varied from 25 to 200 nodes while the concurrent load is kept more or less constant.

Figures 2(a)- 2(d) show the results for delay, delivery ratio, path length and network load.

The delay of MAR lower than AODV and OLSR at almost all node densities. At 150 nodes the delivery ratio of AODV falls sharply due to increased contention as is evident from the network load graph. The delay of MAR is a little higher than AIR because of the extra overhead of additional labels in the messages. AIR has a slightly higher network load than MAR. But for higher network sizes, it generates a lot of congestion around the nodes higher up in the LDAG and drops these packets. MAR on the other hand, uses a lot of short cut paths using the multiple labels and does not have the same problem as AIR. So it delivers far more packets than AIR. For 200 nodes the delivery ratio of MAR is almost similar to OLSR, slightly higher than AODV and almost three



(a) Basic Prefix Labels (b) Labels from other parents (c) Labels from other children

Fig. 1. Example showing set up of labels in MAR

times of AIR. The network load of MAR is lower than OLSR for all network sizes. For lower network density AODV has a low network load because it discovers routes on demand. AIR and MAR on the other hand have to maintain the DHT mappings and exchange the Hellos even if there is no traffic. After 100 nodes, the network load of AODV is more than AIR and MAR as AODV interprets congestion as link failures and has to rediscover routes. The network load for AIR and MAR do not increase much with node density unlike AODV and OLSR. The path length for MAR is the lowest due to availability of multiple paths. Here we don't see a higher path length for AIR because it drops most of the packets that follow the longer paths due to congestion around the nodes higher up in the LDAG. This can also be observed in the delivery ratio graph. In summary, MAR performs as good as or better than traditional protocols while incurring a fraction of the network load and maintaining a much smaller routing state at each node.

B. Mobile configuration with increasing mobility and off-on flows

In this test we used a lot of on-off flows as that is a typical scenario for MANETs. Each node has a round-robin schedule of flows to every other node. From node n_1, the flows look like: off, on to n_2, off on to n_3, off, ..., off, on to n_N, with N being the number of nodes in the network. When the on period starts, the node finds a route and then it sends packets at a constant packet rate. Then the flow stays off for some time and starts a flow to a different node. In a network of N nodes, the number of concurrent flows is N and the total number of flows is N^2. Each flow is between a different pair of source and destination nodes.

The number of nodes is constant at 50. Each source sends 512 byte packets for 50 secs. The rate is 4 packets/sec. The nodes move at 10 m/s rate. The pause time has been varied from 0 (always mobile) to 300s. The high mobility is chosen to exercise the signaling due to frequent route breaks.

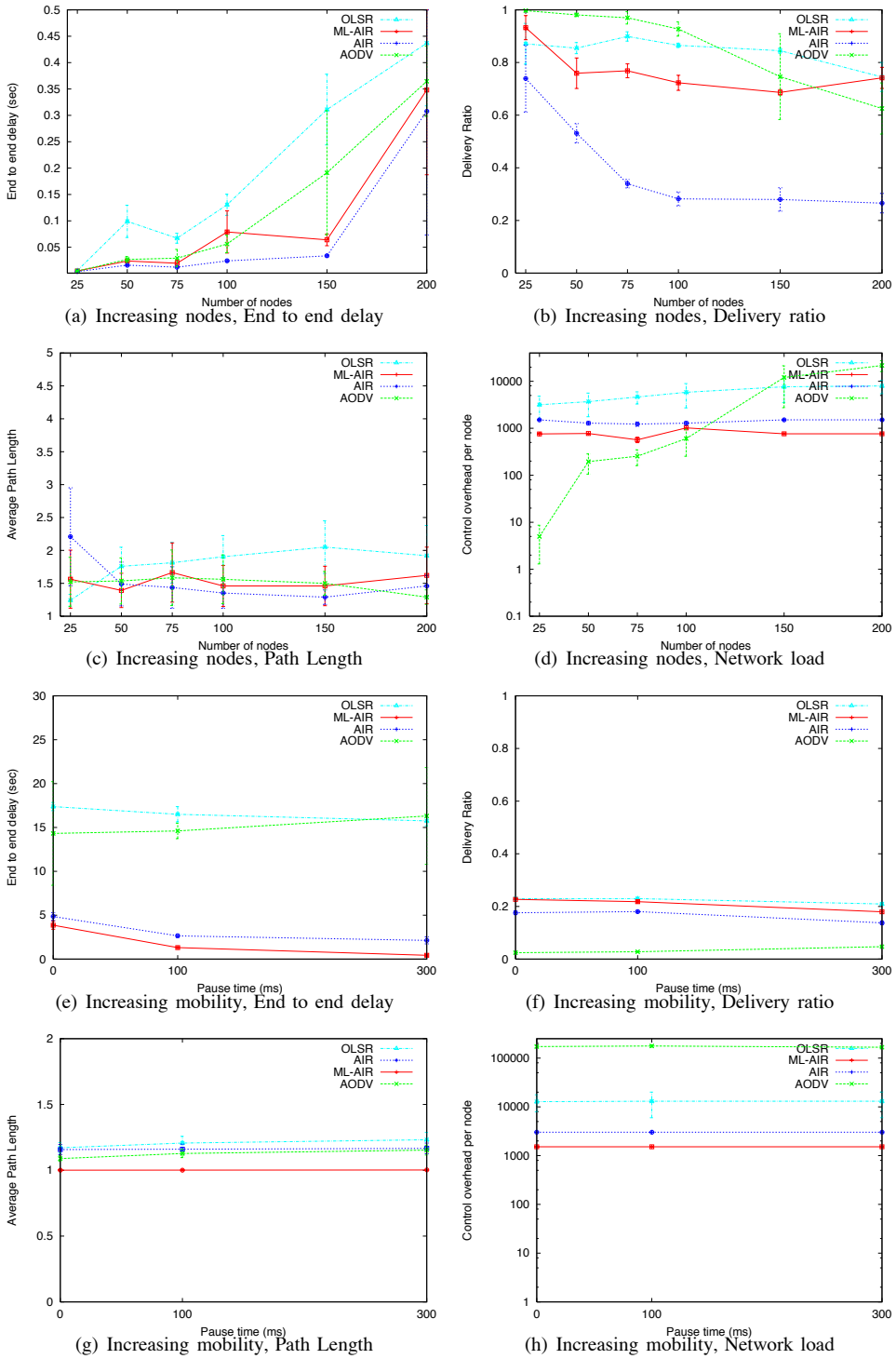
Figures 2(e)- 2(h) show the results for delay, delivery ratio, path length and network load.

For this scenario MAR performs far better than the traditional protocols and the other compact routing protocol AIR. The delay of MAR is the lowest, even lower than AIR. For 300s pause time, the delay for OLSR and AODV is roughly 15 times that of MAR. The delay of MAR is one fourth of AIR at 300s pause time. All the protocols have low delivery ratio due to the high data traffic. The delivery ratio of MAR is the same as OLSR and is about 10 times higher than AODV. It is also slightly higher than AIR. MAR has the lowest network load. It is almost 1/100th of AODV, 1/10th of OLSR and about half of that of AIR. The path length of MAR is also the lowest. The path length of all protocols in this scenario is low due to high network density. We have shown in Table I earlier that MAR delivers packets beyond the two hop neighborhood successfully.

VI. Conclusion

We presented MAR which is the first compact routing protocol that attains a path stretch close to shortest path routing and works in mobility scenarios. In MAR each node gets basic prefix label from a distributively elected root node. The basic prefix label is based on the location of the node with respect to the root in the network. Then each node communicates its label to its neighbors and assigns itself additional labels. The use of additional labels allows the node to have one path to each of its neighbors. So the node can find the shortest path to the destination. The multiple paths makes it resilient to node and link failures and would also be useful in a future QoS implementation.

We have run Qualnet simulations for static and mobile scenarios. The simulations show that MAR is has a much lower control overhead than AODV, OLSR and is scalable. The end to end delay, delivery ratio and path stretch are comparable or better than these protocols.



(a) Increasing nodes, End to end delay

(b) Increasing nodes, Delivery ratio

(c) Increasing nodes, Path Length

(d) Increasing nodes, Network load

(e) Increasing mobility, End to end delay

(f) Increasing mobility, Delivery ratio

(g) Increasing mobility, Path Length

(h) Increasing mobility, Network load

Fig. 2. Performance of MAR with increasing number of nodes and mobility

REFERENCES

[1] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626 (Experimental), Oct. 2003.

[2] C. E. Perkins, P. Bhagwat, *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers*, SIGCOMM 1994, pp 234-244.

[3] D. B. Johnson and D. A. Maltz, *Dynamic source routing in adhoc networks*, Mobile Computing, Kluwer Academic Publsihers, 1996, vol. 353.

[4] C. Perkins, E. M. Royer, *Ad-hoc on-demand distance vector routing*, Proc. of the Second IEEE Workshop on Mob. Comp. Syst. and App., 1999. WMCSA 99, pages 90-100, Feb 1999.

[5] J. J. Garcia-Luna-Aceves, and D. Sampath, *Scalable Integrated Routing Using Prefix Labels and Distributed Hash Tables for MANETs*, Proc. IEEE MASS 2009: The 6th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, October 12 - 15, 2009, Macau SAR, P.R.C.

[6] M. Thorup, U. Zwick, *Compact routing schemes*, SPAA 01: Proceedings of the thirteenth annula ACM symposium on Parallel algorithms and architectures, pages 1-10, New York, NY, USA, 2001.

[7] J. Eriksson, M. Faloutsos and S. V. Krishnamurthy, *Dart: dynamic address routing for scalable ad hoc and mesh networks*, IEEE/ACM Trans. Netw., 15(1):119132, 2007.

[8] A. C. Viana, M. D. de Amorim, S. Fdida and J. F. de Rezende. *An underlay strategy for indirect routing*, Wirel. Netw., 10(6):747758, 2004.

[9] Y. Mao, F. Wang, L. Qiu, S. Lam and J. Smith, *S4: Small State and Small Stretch Compact Routing Protocol for Large Static Wireless Networks*, IEEE/ACM Trans. on Networking, Vol. 18, No. 3, June 2010.

[10] L. Kleinrock and F. Kamoun, *Hierarchical Routing for Large Networks, Performance Evaluation and Optimization*, Computer Networks, Vol. 1, No. 3, pp. 155-174, January 1977.

[11] Qualnet 5.0, Scalable Network Technologies, http://scalablenetworks.com, 2004.