PROSE: Scalable Routing in MANETs Using Prefix Labels and Distributed Hashing

Dhananjay Sampath
Computer Engineering Department
University of California, Santa Cruz
1156 High Street, Santa Cruz, CA 95064
e-mail: dsampath@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
Palo Alto Research Center (PARC)
3333 Coyote Hill Road
Palo Alto, CA 94304
e-mail: jj@parc.com

***Abstract*—We introduce the Prefix Routing Over Set Elements (PROSE) protocol for scalable routing in MANETs based on the combined use of prefix labels and distributed hashing. In PROSE, nodes use neighbor-to-neighbor signaling to label themselves with prefix labels that provide implicit routing from any node to any network destination, and nodes implement a distributed hash table to store the mappings between node identifiers (e.g., a MAC or IP address) and their prefix labels and allow destinations to publish their existence and sources to subscribe to their intended destinations. We show that PROSE provides correct routing based on prefix labels and that its signaling overhead grows sub-linearly with the network size. We present simulation and testbed results that illustrate the benefits of PROSE compared to traditional MANET routing protocols, and show that the stretch of the prefix-based routes compared to shortest-paths is smaller than three.**

I. Introduction

In spite of the many recent advances in hardware and software, providing seamless connectivity and information to people and devices on the move remains a big challenge. We argue that, at least in part, this is due to the scaling limitations of today's routing protocols for mobile ad hoc networks (MANET).

A close look at the MANET routing protocols in use today reveals that the node addresses used in the forwarding of data packets (e.g., MAC addresses or IP addresses) are really names, in that they are constant and assigned to nodes independently of the relative location of the nodes in the MANET. Consequently, sources do not have to use a mapping from the name of a destination to its location. However, since names are used directly in routing tables, these tables must be updated as nodes change their relative locations in the network. Furthermore, since a node address has no correlation to its relative location in a MANET, the only way to find and establish a route to a destination is by some form of destination-driven or source-driven flooding. Destination-driven flooding is used in proactive routing protocols (e.g., OLSR [5]) to disseminate either distances to destinations or link-state information to all nodes, so that nodes can maintain a routing entry to each destination. Source-driven flooding is used in on-demand routing protocols (e.g., AODV [18], DSR [11]), in which a source floods a route request (RREQ) that is answered by the destination or nodes with active routes to it. Clearly, neither approach is efficient for very large or very dynamic MANETs, because the signaling overhead incurred in these approaches grows at least linearly with the number of destinations or links.

The main contribution of this paper is the introduction of the Prefix Routing Over Set Elements (PROSE) protocol. PROSE uses a self-organizing prefix labeling of nodes and the distributed hashing of node identifiers to prefix labels to integrate naming, addressing and routing efficiently in a MANET.

Section II presents a summary of the many schemes proposed in the past to make routing in wireless networks more efficient. These prior works organize a network hierarchically, limit the rate or distance used to disseminate updates, establish virtual overlays, use geographical coordinates, or use virtual coordinates for routing. Interestingly, no prior proposals exist that combine prefix labels with distributed hashing effectively for routing in MANETs.

Sections III to VI describe PROSE and its mechanisms. Nodes use hop-by-hop signaling to assign themselves prefix labels denoting their location relative to an elected root node according to a breadth-first coverage of the network. The prefix labels assigned to nodes implicitly define at least one route between any two nodes. To allow sources to route packets to destinations, a distributed hash table (DHT) is built and maintained in the network to store the mapping between the unique identifier of each destination (e.g., a MAC or IP address) and the prefix label assigned to the node. Each destination uses a common hashing function to map its unique node identifier to a public prefix label, and forwards its own mapping towards that public prefix label. A node with the closest match to the public prefix label becomes the *anchor* for the destination node. A source node uses the same common hashing function to send a request to the anchor of a destination asking for the actual prefix label of the destination.

The key difference between PROSE and all prior approaches based on clustering or prefix addresses is that prefixes in PROSE are assigned using a self-organizing breadth-first search of the nodes, rather than on the grouping of nodes into sets or clusters controlled by special nodes or landmarks. Section VII analyzes the resulting signaling overhead incurred by PROSE and shows that it grows sub-linearly with the number of network nodes. Sections VIII and IX present simulations and testbed results illustrating the performance advantages of PROSE compared to traditional routing in MANETs in networks ranging from as few as 10 to as many as 500 nodes. Our results clearly show that incurring limited overhead in establishing routes that may be longer than the shortest paths is a much more attractive approach that attempting to establish shortest paths in MANETs at the expense of large amounts of signaling.

II. Related Work

Many approaches have been proposed to reduce the amount of signaling overhead incurred in routing compared to the traditional proactive and on-demand routing schemes, and we only cite a few examples for each major category due to space limitations.

Hierarchical routing schemes reduce signaling overhead by organizing nodes into clusters (e.g., [13], [21]). The key limitation with clustering (or hierarchical routing) schemes is that the affiliation of nodes to clusters is easily broken when nodes move and re-establishing such affiliations involves flooding that may become too costly. Related approaches attempt to damp down the propagation of signaling messages as they travel away from their originating points without creating clusters (e.g., HSLS [19] and FSR [16]). The limitation of these approaches is that nodes must still update information about all destinations. An approach to reducing the amount of information communicated among nodes is to hash the node identifiers of destinations into Bloom filters, which are then used in routing updates (e.g., [1], [2]). The key limitation of this approach is the existence of false positives, which forces nodes to verify whether a route to a destination through a given neighbor is true.

An alternative approach consists of establishing a DHT over a virtual topology defined on top of the physical network. Examples of this approach are Kademlia [15], Tapestry [23], and VRR [3]. The advantage of this approach is that the DHT size grows only logarithmically with the number of intended destinations reached over the virtual topology. The key limitation with approaches based on overlays is that a virtual link corresponds to a multi-hop path in the physical network topology. Accordingly, signaling overhead must be incurred to maintain such links, which becomes excessive in large MANETs.

Many geographical routing schemes have been proposed (e.g., GPSR [12], XYLS [6], GLS [14]) using geographical coordinates for routing, rather than node identifiers. The key limitation of these schemes is that each node must know its own geographical location, which in turn requires a positioning system (e.g., GPS) to inform each node of its location.

Several proposals try to attain the scaling properties of geographical routing schemes, but without the need for an external position information service like GPS, by using virtual coordinates. A number of virtual-coordinate schemes use the distances that a node has to a set of reference nodes (beacons) as the location of the node, rather than its geographical location. Examples of this approach are Beacon Vector Routing (BVR) [8] and LCR [4] . The main limitation of this type of virtual coordinates is that multiple nodes may be assigned the same virtual coordinates, and there is no inherent uniqueness to a specific vector of distances to beacons. This results in either incorrect routing or the use of additional signaling (including flooding) aimed at resolving false positives.

Many compact routing schemes have been proposed based on the notion of interval routing, in which a depth-first search approach is used to label nodes with intervals of identifiers in a way that the route between any two nodes is implicit from their intervals. Tribe [22] is an example of this approach. Tribe partitions a finite address space into ”control regions” corresponding to finite continuous intervals of addresses. The key limitation of all schemes based on interval routing is that they are not applicable to MANETs, because the addition or deletion of any node or link requires the relabeling of large portions of the network.

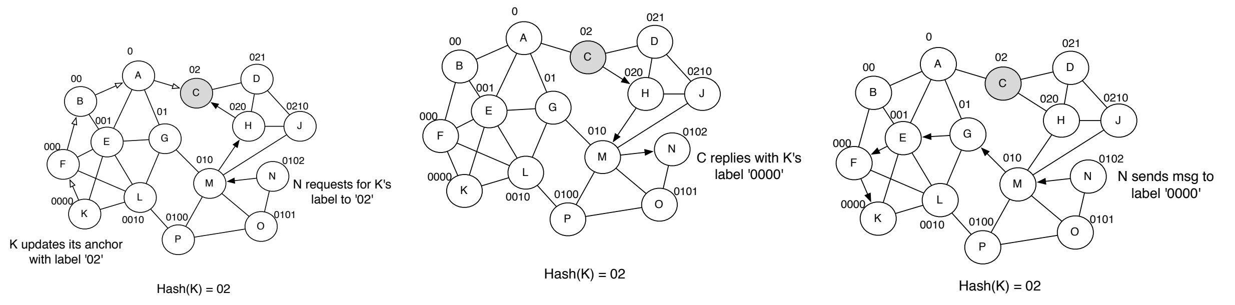
The clear alternative to a depth-first assignment of labels is to implement a breadth-first strategy. DART [7] is a recent example of this type of routing. In DART, nodes are assigned ”routing addresses” corresponding to leaves of a prefix address tree, such that the leaf nodes of every subtree in the prefix address tree form a connected subgraph in the network topology. Hence, DART amounts to establishing clusters of nodes based on prefix address trees. The key limitation of this approach is that substantial relabeling of nodes must occur when nodes move or links and nodes fail, which is the same node-to-cluster affiliation problem present in hierarchical routing.

III. PROSE Operation

PROSE routes packets from sources to destinations by the assignment of prefix labels to nodes, and the dynamic mapping of unique *node identifiers* (NID)(e.g., MAC or IP addresses), to prefix labels.

PROSE builds and maintains a labeled directed acyclic graph (LDAG) rooted at a distributively elected node called the root node. The election of a root node is similar to the election of a root in the distributed spanning tree algorithm. Neighbor-to-neighbor signaling packets propagate from the root node throughout the network on a breadth-first search manner. This results in each node receiving a prefix label from the same elected root. A node announces to its neighbors its own prefix label and the prefix labels it assigns to its children in the LDAG, and stores the prefix labels it hears from all its neighbors. In this paper, we assume that a node advertises to its neighbors only the smallest prefix label that is assigned by its parents. Figure 1 shows an example MANET in which node A is the elected root.

Clearly, a source must know the prefix label of its intended destination for prefix routing to be effective. However, storing all the mappings of NIDs to prefix labels at one (e.g., the core) or just a few nodes would lead to bottlenecks and single-points of failure. Accordingly, nodes build and maintain a distributed hash table (DHT) to store the mappings throughout


K updates its anchor with label '02'
Hash(K) = 02
N requests for K's label to '02'
C replies with K's label '0000'
Hash(K) = 02
N sends msg to label '0000'
Hash(K) = 02

(a) (b) (c)

Fig. 1. PROSE operation: Node A is the root; node C is the anchor of node K; node N requests the prefix label for node K and then routes directly to it.

the network, and carry out publish-subscribe operations using soft-state signaling that uses prefix labels to route packets to specific nodes.

Each destination uses a consistent hashing function that takes as input its NID and returns its public prefix label, which states the location where its mapping should be stored. The node with a prefix label closest to the public prefix label becomes the *anchor* for the destination. A destination then publishes its presence in the network by sending its own mapping (i.e., NID and prefix label) to its anchor (see Figure 1(a)). To find a destination and subscribe to it, a source uses the same consistent hashing function with the destination NID as input. The source then sends a request towards the resulting anchoring prefix label. The anchor of the destination replies to the source with the prefix label of the destination, or forwards the request to the destination. The destination, then replies directly to the source. The example shown in Figure 1 assumes that the anchor replies to the source. Once the source and destination have each other's prefix labels, they can communicate directly (see Figure 1(c)). From the figure, it is clear that PROSE supports multipath prefix routing, i.e, multiple paths may be used from source to destination (e.g., nodes A, B, E and G have multiple prefix paths to node K in Figure 1(c)).

The LDAG serves as a *'churn-tolerant'* resilient structure that defaults to a node with the closest prefix label when a request for an exact match fails. The routing algorithm that routes over the LDAG follows a greedy strategy. When it encounters a local minima, it chooses the next hop with the lowest NID. Given that nodes have prefix labels, the choice of NID does not affect the path to the destination. A node uses the *maximum matching prefix* logic of the routing algorithm to choose its next hop.

IV. Routing over Prefix Labels

A. Structure of Prefix Labels

Let Σ be the alphabet containing a finite number of symbols and Σ^* be the set of all strings over Σ such that $|\Sigma| \geq 2$. Every node labels its links to each of its neighbors with a letter w from Σ. If w_i represents the letter assigned to the i^{th} link of any node then, $w_i \mapsto \{w_i \in \Sigma \mid w_i \neq w_{i+1} \forall i \leq d-1\}$, where d is the degree of the node. Hence, a unique letter is assigned to each link connecting any node to its neighbors. The labeling logic labels each node in the LDAG in a *breadth first* fashion. Since the LDAG can be organized as a k-ary tree, with k being the degree of the LDAG, each child (up to k), is assigned a prefix label Λ as defined below:

Prefix Label: A prefix label Λ for node y is a word in Σ^* such that $\Lambda = \Lambda_{parent} \odot l'$, where Λ_{parent} is the prefix label obtained from the parent and $\odot$ is a concatenation operator where Λ_{parent} is concatenated with a unique suffix over k different choices from Σ to form Λ

It follows from the above definition that the prefix label of a node Λ, uniquely identifies the node in a given LDAG. The prefix labels of nodes define a *predecessor* relation in the LDAG, represented by a $\hookleftarrow$, such that for any two nodes s and d in the LDAG:

1) $s \hookleftarrow d$: s *precedes* d and hence d can be reached by traversing the subtrees of s.
2) $d \hookleftarrow s$: d *precedes* s and hence d can be reached upstream from s by traversing s's ancestors.
3) $d \approx s$: d and s are peers provided that $|\Lambda_d| = |\Lambda_s|$ and both share a common ancestor. In this case, d can be reached by traversing up to the common ancestor until $r \hookleftarrow s$ holds, where r is the common ancestor, and then down to the closest subtree, while $r \hookleftarrow d$ holds and d is reached. Therefore, $d \approx s \implies r \hookleftarrow s \implies r \hookleftarrow d$.
4) $dr \approx s$: This is a special case of the above, where $|\Lambda_d| \neq |\Lambda_s|$ and the only common ancestor is the root node.

Nodes build and maintain an LDAG rooted at an elected root node using *hello* messages exchanged among one-hop neighbors. Each hello message specifies the NID of the root, a monotonically increasing sequence number assigned by the root node, the prefix label and NID of the sending node, and a list of tuples with the assignment of prefix labels to NIDs. The root election algorithm chooses the node with the largest one-hop coverage and breaks any tie with the lowest root NID. As the network begins to self organize with prefix labels, the labeling process can produce multiple LDAGs. When a node is at the interface of two such LDAGs, the label of each of them is compared and the lexicographically larger label is chosen as the dominant one.

Consider the example in Figure 1. The prefix labels for nodes are assigned over the alphabet $\Sigma = 0\ , 1, 2$. The root of the tree is assigned a letter from Σ and the nodes attached to the root are labeled 00 and 01 successively. At the next level, each node is assigned a unique prefix label over the link combined with the prefix of its parent. Nodes E and M are labeled 001 and 010, respectively.

B. Routing Procedure

To route to a destination d, node s chooses the link to any of its two-hop neighbors that offers the maximum length of prefix label that matches with the prefix label of the destination. This is simply *maximum matching prefix* logic, which selects the next hop using a greedy strategy that considers the two-hop neighborhood of a node, and can find shorter paths than the traditional prefix-tree routing by leveraging the richer path diversity of an LDAG compared to a prefix tree. For instance if there exists a label in the two-hop neighborhood that is lexicographically closer to the destination, the next hop is chosen such that the packet is forwarded to that node instead of routing via the prefix-tree parent. We ensure that this greedy strategy does not encounter a local minima by randomly selecting a next hop when all nodes offer the same matching prefix.

From the *predecessor relation* induced by the prefix labels in the LDAG of a MANET, a given node selects a next hop to a prefix label according to the four possible cases allowed by the predecessor relation. The following lemma shows that PROSE converges to correct routes.

Lemma IV.1. *The prefix-based routes built using PROSE are correct in a connected network with no topology changes.*

Proof: By the definition of prefix labels, every node has a unique prefix label, and the LDAG is free of loops by construction. Without loss of generality, consider a source s and a destination d. Because they belong to the LDAG, they must satisfy one of the four cases of the predecessor relation, and each node chosen as next hop to d starting with s must also satisfy the same relation. Because the LDAG is finite and does not change, there must be at least one loop-free path from s to d in the LDAG. ■

However, the topology of a MANET changes constantly and nodes join or leave the network arbitrarily. This results in inconsistent labels at the point of churn. To preserve consistency, we enforce a *strict labeling* by ordering the prefix labels using sequence numbers.

Let L represent a tuple (S_n^a, Λ_n^a), where S_n^a denotes the sequence number that originates at a and is forwarded by node n, and Λ_n^a denotes the prefix label of the current node with respect to a. S is a monotonically increasing integer, while Λ is a word in Σ^*. We define an operator $\prec$ over the set of ordered-pair of identifiers L. If L_x^a, L_y^a are two such tuples then,

$$\{L_x^a \prec L_y^a \mid L_{x,y}^a \in \Sigma\} \tag{1}$$

$$if\,\{\,(S_x^a < S_y^a) \vee [(S_x^a = S_y^a) \wedge (\Lambda_x^k \succ \Lambda_y^j)]\,\}$$

To satisfy the claim that this tuple establishes an ordering among the nodes, we show that the operator $\prec$ is anti-reflexive and transitive over the set Σ^*.

Lemma IV.2. *The relation $\prec$ is a partial ordering of tuples over Σ^*.*

Proof: The first part of the lemma is intuitive, because two tuples are equal only if both the sequence number and the prefix label are the same. Hence, $L_x^a \prec L_y^a$ or $L_y^a \prec L_x^a$, only one of which can hold and is therefore anti-reflexive. To prove that transitivity holds, consider three labels and for convenience call them L_1, L_2, L_3 such that $L_1 \prec L_2$ and $L_2 \prec L_3$. From Equation 1, we see that, $S_1 < S_2 < S_3$ as the Ss are monotonically increasing. This implies that $S_1 < S_3$. It follows that, if $S_1 \not< S_3$ then $L_1 \prec L_3$ only if $S_1 = S_3$ and $\Lambda_1 < \Lambda_3$ (the comparison for Λ is lexicographic in nature). Because we know that Λ holds for transitivity, the relation $\prec$ is an ordering over Σ^* and establishes a successor-predecessor relation. ■

Having shown that the sequenced prefix labels preserve an ordering, we show in a more general theorem that the routing algorithm in PROSE converges to loop-free paths, even in the face of dynamics in the topology.

Theorem IV.3. *A prefix routing scheme with sequence numbers routes correctly*

Proof: From Lemma IV.1 we know that all nodes have a label and that each node shares a relation with all its one-hop nodes. From Lemma IV.2 we have that every labeled node is ordered with respect to the lexicographic length of the prefix label and a sequence number. Therefore, if the routing algorithm routes between two nodes, the greedy strategy is guaranteed to improve some metric (hop count, distance, load, etc.) towards a destination at each hop. Hence, a prefix routing scheme with sequence numbers routes correctly, even in the face of changing topologies. ■

V. Building a DHT

As we stated before, to avoid bottlenecks and single-points of failure, the mappings of NIDs to prefix labels should be disseminated throughout the network. PROSE uses a consistent hashing function to accomplish this objective. Each destination publishes its existence by sending a *publish request* to the node whose prefix label best matches the result obtained by hashing the NID of the destination, which we call the anchor of the destination. A publish request is a tuple consisting of the NID and prefix label of a node. A source wanting to communicate with a destination sends a *subscription request* to the node whose prefix label best matches the result obtained by hashing the NID of the intended destination (i.e., the anchor of the destination). A subscription request consists of the NID and prefix label of the source and the NID of the intended destination. The anchor can either reply to the request from the source or forward the request to the destination, depending on the nature of the source-destination exchange. In this paper, we assume that the anchor replies to the source of a mapping request.

PROSE uses soft-state signaling in the dissemination of publish and subscription requests, in that sources and destinations are responsible for the success of their requests, and anchors service them on a best-effort basis. To save bandwidth, requests to different anchors are aggregated in the hello messages exchanged among nodes. To cope with the changes in the topology of the network, destinations update their anchors periodically, and events that change the prefix label of a node trigger an update. An update refreshes the mapping stored at anchors. Additionally timers at source nodes and anchors are maintained to refresh these mappings and ensure that any stale information is removed.

VI. Adapting to Network Dynamics

The prefix labels assigned to nodes are updated as the topology changes.We discuss how PROSE adapts to these changes by describing the actions taken in the event that nodes and anchors are relabeled.

A. Node Relabeling

Nodes acquire a new label when they reposition themselves in the network. Depending on the type of node, the network organizes itself differently. Further, the relabeling event is atomized into a *join* or a *leave* event. A *join* event is handled similar to the initial labeling process, as described before, irrespective of the type of node. Each node acquires a label from the node that is its immediate predecessor at the point of join. On the other hand, a *leave* event is handled differently for different type of nodes. If the node that leaves is a *leaf node*, then no further nodes are affected by this event. However, if the departing node is an internal node, then the adaptation is more involved. This is primarily owing to the construction of the LDAG, wherein the labels of the subtree are derived from prefixes of some ancestor. However, while the prefixes might change owing to repositioning, the suffixes remain unique in the entire subtree.

To further understand how an internal node *leave* is handled, consider three nodes x, y and z as shown in Figure. 2. These three nodes are peers and have the same parent p and the whole LDAG is rooted at some node r. Let p's most recent label be $k\alpha p$ and its children's be $k\alpha px$, $k\alpha py$, $k\alpha pz$. Also assume that there is some path between the three subtrees such that a node in each subtree can reach an arbitrary node in its peer-subtree in addition to the path through its parent. Now if node p were to fail or leave, the following steps are initiated to ensure that rest of the network remains largely unaffected:

- Each child - $k\alpha px$, $k\alpha py$, $k\alpha pz$ initializes a routine to discover its peers (of which it was entirely aware) using routes other than the failed one.
- As it discovers paths, the peer with lowest global identifier, say $k\alpha px$ elects itself as a *secondary-root* and attaches the prefix label of its lost parent as its own prefix.
- Once a *secondary-root* is setup, the remaining peers continue using their unchanged prefix labels but route to other peer-subtrees using alternative routes.
- If a node were to join any of the peers, say $k\alpha px$ and has shorter paths to all of its peers then the secondary-root relinquishes its position and a new *root* is reinstated. The new *root* node sends an update with a larger sequence number to ensure that its shorter paths are not a result of a loop in the network.

B. Anchor Relabeling

Assume that some node with label $k\alpha x$ is the designated anchor for node y in the network. Node y proactively updates its anchor with its current label. If node $k\alpha x$ fails or moves to another location in the network, or if no node with such a label exists, then a node in some subtree with the maximum matching prefix is designated as the *secondary-anchor*. Note that the secondary-anchor is logically in the path towards the anchor. To exemplify, if a node with label $k\alpha x$ were to go down and the prefix of this label were $k\alpha$, then the last node actively engaged in providing directory service is in the subtree α. In the worst-case scenario, the scheme backs up to the root node, because no other node has a closer prefix to the destination currently being sought. If another node joins as a child to the same parent and acquires a label that matches more closely to the anchor, proactive updates from the node automatically expire the parent node as an anchor and route to the new anchor.

In the case of transient topologies, resetting the prefix labels of an entire subtree can cost as much as the number of nodes in that subtree. In the event that a child node is unable to reach any of its peers, or becomes completely disconnected, it initiates a relabeling process. The relabeling begins either with the prefix of a known subtree or with a new LDAG rooted at the disconnected peer. This is determined by setting a *hold-timer*. The hold-timer waits for a certain amount of time to allow for the link to reconnect. When the hold-timer expires it assumes disconnection and begins the label reset process.

C. Adaptive Update Timers

It is critical to clamp the updates and aggregate event messages to reduce the amount of propagation. Here we describe an intelligent timer that determines if an update needs to be sent within a certain *hold-down time* and this decision is made based on the topological and prefix label changes that occur within the neighborhood of the node. Each node is bootstrapped with a hold-down time (δ). When the δ timer expires, a node transmits a *hello* message with the corresponding payload. Within δ if a node detects topological changes, it aggregates these changes and determines if number of changes crosses a certain threshold (T_δ). The threshold is in turn determined by the number of changes in the two-hop neighborhood. Another threshold is defined for the number of mappings and upon detecting changes in mappings greater than the mapping-threshold (T_m), an update with the new mappings is sent. If P_δ is defined as the probability of the

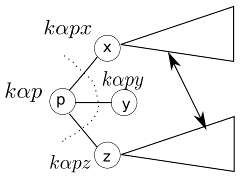


Fig. 2. Resilience of prefix labels

topological threshold being reached and P_m be defined as the probability of the mapping threshold being reached, then the overall timer function can be characterized as

$$\Delta = \delta - \{P_\delta \cdot \delta_{T_\delta} + P_m \cdot \delta_{T_m}\}$$

VII. Complexity Analysis

For the purposes of our analysis, we model a MANET network as a graph $G(V, E)$ where V is the set of vertices representing the network nodes and E is the set of edges that correspond to the physical link between the nodes. Links are assumed to be bi-directional and link costs are non-negative. We further make the assumption that G is a random geometric graph (RGG) [17] such that the nodes are placed uniformly at random and independently of each other. Two nodes are connected if and only if they are within a radius of r of each other. The radius corresponds to the radio range of the actual physcial nodes. Nodes are allowed to join or leave the network arbitrarily. The radius of the threshold value of distances in the RGG is set to $R_c = \sqrt{((ln(n)) + 0(1)/(pi * n))}$ [9] so that the nodes are connected. This gives us the property [10] that the diameter of the network is given by $\sqrt{(ln(n)/n)}$. Also, in a RGG we can approximate the number of one-hop neighbors of any given node as $ln(n)$, where n in all of the above expressions is the number of nodes in the network.

To understand the overhead complexity incurred in PROSE, we begin by identifying the signaling overhead contribution of each mechanism used in the protocol. We compute the signaling overhead in terms of messages transmitted over the entire network.

The *hello* messages used in PROSE are sent periodically and on an event-driven basis among one-hop neighbors. Hence, we compute the resulting overhead as the sum of two separate terms. From our model, the number of one-hop neighbors can be represented as $ln(n)$ and therefore the cost of this phase, network-wide is $C_{p1} = (f_m(t - t_e) + f_m(t_e))(n * ln(n) * \delta_h)$, where n is the number of nodes in the network, δ_h is size of the *hello* message exchanged and f_m is a function of the chosen mobility model. This function f_m describes the number of events that trigger changes in the topology. The first term represents the frequency of the periodic update triggered. Note that each time an event based update is sent, the periodic timers are reset. The second component purely captures the event-driven updates.

Each destination node in the network sends a message to its anchor node and refreshes the stored mapping. The periodic timer for this update is much larger than the timer in the neighbor-to-neighbor message exchange. This component, however, has two distinct cases. In the worst-case scenario when the anchor is situated at the diametrically opposite end of the network, the message travels across. Hence, it is a function of the diameter of the network. A more common case, however, is the average path length that the message has to travel. Due to the prefix labels, this path length is lesser. In the worst case, the cost of this part of PROSE is:

$$C_{p2} = (n/\gamma)(f_m(t' - t'_e) + f_m(t'_e))(\sqrt{n/ln(n)})$$

where γ reduces the number of nodes to ones in the common subtree. γ represents the factor of nodes within the common matching subtree. The expression in the average case is $C_{p2} = (n/\gamma)(f_m(t' - t'_e) + f_m(t'_e))ln_d(n)$, where d is the average degree of the network.

The third component of signaling overhead in PROSE corresponds to the subscription requests, which involves an exchange that increases constantly based on the traffic sources and in any case increases the above expression by a small constant factor $(\gamma + k)$.

An additional component of overhead is incurred when a label reset occurs owing to disconnected components. If the rate of resets in the network is defined as η then this component becomes $C_{reset} = (n\eta/\gamma)(\sqrt{n/ln(n)})$, and in the worst case and in the average case $C_{reset} = (n\eta/\gamma)(ln_d(n))$.

To see how PROSE saves in terms of signaling overhead compared o traditional routing schemes, let us now compare these expressions with that of a link-state protocol such as OLSR, and an on-demand routing protocol like AODV. The first component of the total cost in the case of link state grows much faster than in PROSE, because each link-event is propagated n^2 instead of $n*ln(n)$ times. Though in a link-state protocol the second component C_{p2} is non-existent. However, this trader-off is much costlier for link-state updating, because the second phase of PROSE only increases the signaling overhead by a factor of $O(n)$. In the case of on-demand routing protocols, constantly moving nodes cause the second term of the total cost to shoot up. On-demand protocols have to flood the entire network in the worst case and do so for each event. This means that the flooding reaches $\pi * (n/ln(n))$ nodes instead of $\sqrt{n/ln(n)}$ nodes. This makes C_{p2} a $O(n^2)$ term as well.

Figure 3 shows how slowly the signaling complexity grows in PROSE as the number of network nodes increases, and that it clearly outperforms the complexity of traditional link-state or reactive protocols.

VIII. Performance Analysis

To analyze the performance of PROSE, we modeled it in the Qualnet simulator [20]. Qualnet is a discrete event simulator that allows high fidelity performance evaluations of network models.

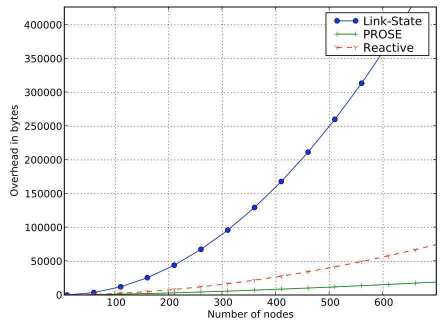


Fig. 3. Asymptotic growth of overhead for different protocols.

A. Setup and Benchmarks

Each of the experiments were instrumented with a network of nodes over a terrain 600m long and 900m wide. Nodes were placed randomly within a unit block of 5% the dimensions of the terrain and each of these unit blocks were distributed uniformly. We chose the random-waypoint mobility model in which the way points are distributed over some convex region within the terrain. The velocity distribution was varied from 1 to 10 m/s uniformly and each node paused for a specific time interval selected from a exponential distribution of mean 30ms.

The radios were instrumented with a CSMA medium access control and the data rate, sensitivity and power were set to correspond to the chosen terrain in such a way that under a static scenario, the entire network remained connected with minimal interference. No path loss was simulated in the interest of understanding the protocol behavior without being mitigated by losses.

We simulated up to 500 nodes in these networks and in each of these runs, the number of active flows was set to 250 flows. The interval of each of these simulations were distributed exponentially with mean equal to 1/20th of the duration of simulation. Each protocol was also run at separate pause times from static to continuously mobile scenarios. The performance of all protocols was observed for different mobility rates with increasing pause times from 0 to the duration of the simulation. The duration of the simulation was set as 300s. To remove any topological artifact (owing to node placement strategy) we ran the simulations over 10 random seeds and normalized the results.

We compared PROSE with OLSR and AODV, given that these are most widely used proactive and on-demand routing protocols for MANETs, respectively. These protocols demonstrate opposing design choices and help to understand how PROSE contributes to reducing stress on the network with the help of its anchors and prefix labels.

B. Scenarios

In each of our experiments, we use the following metrics in our comparison: (a) *control packets*, which measures the total control overhead packets generated for the given flow in bytes; (b) *latency*, which is defined as the delay measured per flow from one end to the other; and (c) *packet delivery ratio*, which is an indicator of successful delivery, defined as the number of data packets delivered per total number of messages sent.

1) Mobility Scenario: Our first scenario demonstrates the robustness of PROSE in the face of mobility by comparing the different protocols under pause times increasing from 0 to 300 seconds in a MANET of 500 nodes. Figure VIII-B4 shows the results for this scenario. We notice that under smaller pause times, when nodes move constantly, all protocols suffer from observably low delivery ratio. OLSR suffers the most, because LSUs are propagated to the entire network after any topology change. AODV manages to catch up faster as the rate of mobility goes down. Since the network is flooded for discovery of a destination each time, AODV leverages the unexpired entries in its routing table cache to respond to nodes that are not moving as fast.

PROSE utilizes anchors very effectively. It achieves almost 15% higher delivery ratio than either OLSR or AODV and yet has 50% less overhead than the two. The higher delivery ratio is explained by fewer control messages transmitted, resulting in lesser congestion across the network. The end-to-end latency of PROSE is also low under high mobility because fewer packets are retransmitted. While it is hard to outperform flooding in terms of latencies, flooding under overloaded conditions results in severe performance degradation and this is where PROSE shows better end-to-end latencies. The performance of AODV and OLSR come closer to that of PROSE only in relatively static scenarios.

2) Scalability Scenario: In our second scenario, we show the performance of PROSE, OLSR and AODV as the number of nodes varies from 10 to 250. Nodes are mobile at a constant rate for all experiments with a pause time of 15m/s. Figure VIII-B4 shows the results for delivery ratio, end-to-end delay, and control overhead.

PROSE clearly outperforms all the other protocols by delivering almost 15%-20% more. As the number of nodes increases, OLSR and AODV scale poorly, while PROSE signaling overhead increases at a far smaller rate. Figure VIII-B4 shows that the number of messages for 100 to 250 nodes is almost four times that of PROSE. Message aggregation plays a crucial role here, as the compact representation of the mappings keep the size of the messages small and helps to aggressively aggregate information in PROSE. As a result, even with hold-down timers, the end-to-end latency in PROSE is 5 seconds lower than in AODV and OLSR because there are fewer collisions.

3) Churn Scenario: In our third scenario, the network is static at the start of the simulation and all protocols converge to stable routing tables. A single node is then moved from one end of the simulation space to a diametrically opposite location. We then measure the amount of signaling overhead incurred. Figure 6(a) shows the overhead incurred in the three protocols in networks of 10 to 500 nodes after a single node disruption. In the interval of 10-50 nodes, we see a huge number of messages being exchanged. PROSE, on the other hand incurs a 6 times reduction in the same interval. PROSE clearly incurs much smaller signaling overhead than the other two protocols.

4) Path Stretch and Label Resets: Using the traces obtained from the second scenario, we computed the path stretch (ratio of prefix paths over the corresponding shortest paths) for PROSE, and the results are shown in Figure 6(c) where stretch is plotted against the node degree. The results are very promising, because the stretch incurred in PROSE ranges from less than 3 for node degrees of 3 to less than 2 when the node degree is 8.

While most MANET protocols achieve shortest paths, they fail to account for the congestion that results from discovering these paths. In PROSE, we incur a small path stretch of under 3; however, because of the large reduction in signaling overhead attained with PROSE, it results in significant performance gains.

Figure 6(b) shows how the prefix label resets are reduced as the node degree increases. This is an indicator of the fault tolerant logic at work. Each time a disconnection occurs, nodes find paths to the peers of the nodes and prevent a label reset by electing *secondary-roots*.

IX. Implementation

We designed a prototype of PROSE to demonstrate some of its characteristics using a testbed. We wrote PROSE in Python, because it enabled rapid prototyping. The testbed consisted of 10 nodes and each node in the network was equipped with a 802.11b/g radio on a Mini-ITX board running Debian. Each node establishes a UDP connection with the neighboring nodes to exchange messages. We built our own version of flow control without explicit *acks* to leverage the use of the *hello* messages. The testbed nodes were deployed within a building as shown in the figure IX and were placed such that any source-destination pair was connected over multiple hops. We evaluated the behavior of PROSE under different scenarios. As a benchmark, we used the OLSR protocol from [5] for our comparisons.

Due to space limitations, we briefly discuss a single scenario to show a proof of concept and also demonstrate some crucial properties of PROSE. To characterize the effects of mobility, we studied the scenario where a node is moved to a different part of the network after 10 minutes of static placement, and determined the overhead incurred by PROSE and the OLSR protocol. Table I shows the performance results for the two protocols. Note that while there is a slight improvement in the delivery ratio in PROSE, the number of control messages transmitted when a single node moved is almost four times less in PROSE than in OLSR.

X. Conclusion

We presented PROSE, an approach for scalable routing in MANETs using the integration of prefix labels and distributed hashing in a way that differs from all prior schemes proposed in the past based over DHTs and virtual coordinates. PROSE scales very well with the number of nodes and in contrast to several prior compact routing schemes it is applicable to MANETs. The performance results obtained by simulation and in testbed experiments are very promising. On the one hand, PROSE incurs very small signaling overhead and a stretch factor (over the shortest paths that could be attained) below three, which makes it very attractive for large MANETs. On the other hand, PROSE provides the same or better delivery ratios than traditional routing protocols and much better end-to-end delays.

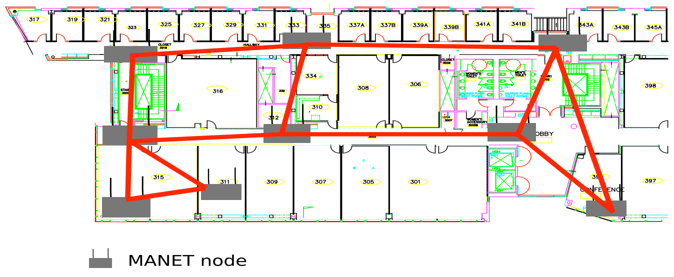


Fig. 7. Testbed deployment

	Delivery Raio	Overhead single link	Overhead Total
OLSR	95.224%	63 pkts	5400 pkts
PROSE	97.117%	16 pkts	3100 pkts

TABLE I
Implementation results

References

[1] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. Scalable bloom filters. *Inf. Process. Lett.*, 101(6):255–261, 2007.

[2] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey, 2002.

[3] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: network routing inspired by dhts. *SIGCOMM Comput. Commun. Rev.*, 36(4):351–362, 2006.

[4] Qing Cao and Tarek Abdelzaher. Scalable logical coordinates framework for routing in wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(4):557–593, 2006.

[5] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). In *RFC Editor*, United States, 2003.

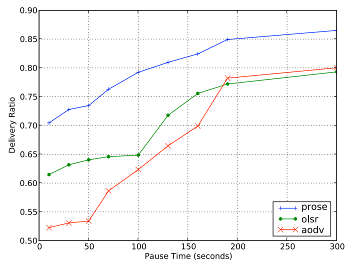
[6] S.M. Das, H. Pucha, and Y.C. Hu. Performance comparison of scalable location services for geographic ad hoc routing. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2:1228–1239 vol. 2, March 2005.

[7] Jakob Eriksson, Michalis Faloutsos, and Srikanth V. Krishnamurthy. Dart: dynamic address routing for scalable ad hoc and mesh networks. *IEEE/ACM Trans. Netw.*, 15(1):119–132, 2007.

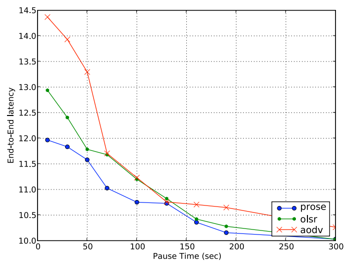
[8] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: scalable point-to-point routing in wireless sensornets. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 329–342, Berkeley, CA, USA, 2005. USENIX Association.

[9] P. Gupta and P.R. Kumar. Critical power for asymptotic connectivity. *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, 1:1106–1110 vol.1, 1998.

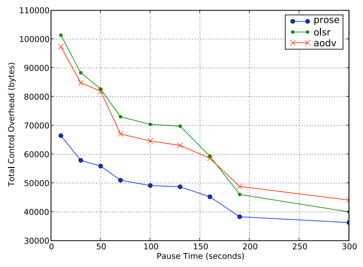
[10] Xingde Jia. Wireless networks and random geometric graphs. *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on*, pages 575–579, May 2004.



(a) Delivery ratio

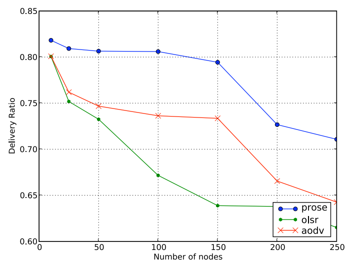


(b) End-to-End latency

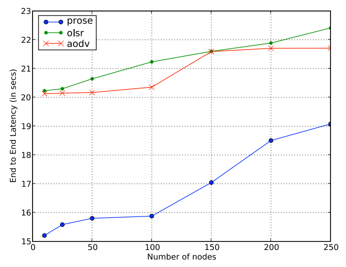


(c) Control Overhead

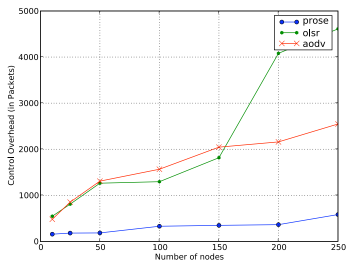
Fig. 4. Protocol performance for different pause times.



(a) Delivery ratio

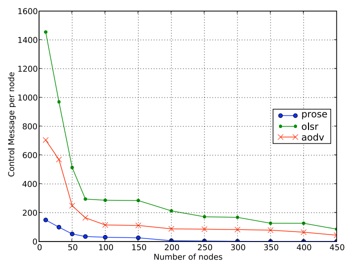


(b) End-to-End Latency

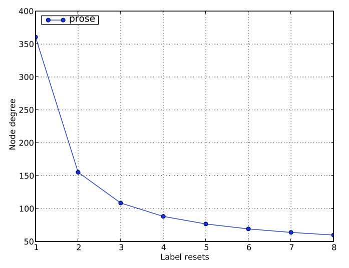


(c) Control Overhead

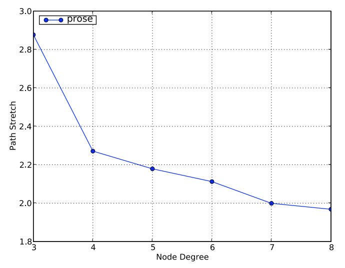
Fig. 5. Protocol performance for different numbers of nodes



(a) Number of message seen at each node after a single disruption



(b) Decreasing label resets with increasing node degree



(c) Path stretch comparison

Fig. 6. Performance of PROSE in terms of messages after a single event, different node degrees, and path stretch.

[11] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[12] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM.

[13] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks performance evaluation and optimization. *Computer Networks (1976)*, 1(3):155 – 155, 1977.

[14] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130, New York, NY, USA, 2000. ACM.

[15] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *1st International Peer-to-Peer Symposium (IPTPS 2002)*, pages 53–65, 2002.

[16] Guangyu Pei, M. Gerla, and Tsu-Wei Chen. Fisheye state routing: a routing scheme for ad hoc wireless networks. *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, 1:70–74 vol.1, 2000.

[17] Mathew Penrose. *Random Geometric Graphs (Oxford Studies in Probability)*. Oxford University Press, USA, July 2003.

[18] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. In *RFC Editor*, United States, 2003.

[19] César A. Santiváñez, Ram Ramanathan, and Ioannis Stavrakakis. Making link-state routing scale for ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 22–32, New York, NY, USA, 2001. ACM.

[20] Scalable N. Technologies. Qualnet. http://www.scalable-networks.com/.

[21] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. *SIGCOMM Comput. Commun. Rev.*, 18(4):35–42, 1988.

[22] Aline Carneiro Viana, Marcelo Dias de Amorim, Serge Fdida, and José Ferreira de Rezende. An underlay strategy for indirect routing. *Wirel. Netw.*, 10(6):747–758, 2004.

[23] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, University of California at Berkeley, Berkeley, CA, USA, 2001.