Loop-Free Routing Using a Dense Label Set in Wireless Networks

Marc Mosko and J.J. Garcia-Luna-Aceves
Computer Engineering Dept.
Baskin School of Engineering
University of California
Santa Cruz, CA 95064
{mmosko, jj}@soe.ucsc.edu

***Abstract*— We present a new class of on-demand routing protocols called Split Label Routing (SLR). The protocols guarantee loop-freedom at every instant by ensuring that node labels are always in topological order, and thus induce a directed acyclic graph (DAG). The novel feature of SLR is that it uses a dense ordinal set with a strict partial order to label nodes. For any two labels there is always some label in between them. This allows SLR to "insert" a node in to an existing DAG, without the need to relabel predecessors. SLR inherently provides multiple paths to destinations. We present a practical, finitely dense implementation that uses a destination-controlled sequence number. The sequence number functions as a reset to node ordering when no more label splits are possible. The sequence number is changed only by the destination. Simulations show that our proposed protocol outperforms existing state-of-the-art on-demand routing protocols.**

I. Introduction

A wireless ad hoc network consists of nodes with radio network interfaces cooperatively relaying data without the aid of such fixed infrastructures as cell sites or base stations. Examples of ad hoc networks are laptops or PDAs with wireless interfaces in a meeting room, or emergency rescue workers rapidly establishing temporary networks. The routing problem in a wireless ad hoc network is to find multi-hop paths between sources and sinks of data. Because of mobility, unreliable channels, limited power, limited bandwidth, and channel contention, routing protocols designed for wired networks exhibit poor performance over ad hoc networks. We present a new class of on-demand protocols designed for wireless ad hoc networks that is loop-free at every instant.

In our discussion of routing, we use the terms *predecessor* and *successor* in the context of an underlying directed acyclic graph (DAG). At a node i, for destination j, the successors of node i for j are those intermediary nodes along the path from i to j, including j. When used in the singular, "the successor" of i to j means the adjacent successor of i to j. If using multiple paths per destination (called "multi-path"), "the successor" means collectively all such one-hop nodes. The predecessors of i for j are those nodes that have i on their successor paths to j. For loop-freedom, when a node picks a new successor for a destination, it must ensure that no predecessors are on that new successor path.

A class of on-demand routing protocols use "link reversal" algorithms that maintain a DAG by manipulating edges in a digraph. These protocols include GB [6], LMR [4] , and TORA [14]. GB and LMR operate by reversing the direction of certain links at each iteration of the algorithm. This is realized by associating an ordered pair (α_i, i) to each node i and defining a lexicographic total order on the label. The destination has the minimum label. If a node is a local minimum with respect to its neighbors, it does not have a path to the destination. Such a node increases its label, reversing some or all of its links, and the algorithm continues. The idea behind TORA is that a node that becomes a local minimum chooses a new label such that it becomes a global maximum.

Another class of on-demand loop-free protocols uses source routing. DSR [10] builds complete hop-by-hop routes at each source node. Packet paths are inherently loop-free. DSR works by broadcasting a route request over the network and recording the path of the packet. When a node with a path to the destination receives the request, it can send a reply along the reverse route. The reply contains the responding node's path and records its route back to the requesting node. Thus, the requesting node has the complete path.

A third class of on-demand loop-free protocols operate by maintaining node labels in a topological order. AODV [15], ROAM [17], and LDR [7] use such a technique. AODV maintains a sequence number and hop-count per destination at each node. AODV's use of sequence numbers is such that when a node looses its successor to a destination and increases the stored sequence number to prevent loops, it generally becomes a local maximum in the topological ordering. ROAM and LDR are based on DUAL [8].

The basis of DUAL is the concept of feasible distance (FD). Each node keeps a FD for each known destination. The FD tracks the minimum distance ever known to the destination, and is thus a non-increasing function over time. To prevent loops, a node may only use a successor whose reported distance is less than the stored FD. Because link costs are positive, it would be impossible for a predecessor to have a smaller distance than the stored FD. One problem is how to reset a node's FD to a larger value so it may forget about old paths and begin using a longer path, such as when a link fails. DUAL implements a diffusing computation [5] over reliable communications to break potential loops and reset predecessor FDs before a node may change successor to a longer path.

Notation	
General SLR	
$\mathcal{L}$	An ordinal set for labeling vertices.
$\mathcal{L}_i$	The label of node i for a given destination.
$\mathcal{M}_i$	The minimum label of node i's predecessors based on a request.
$\mathcal{G}_i$	The proposed new label for node i based on a routing event.
$\mathcal{S}_i$	At node i, the set of nodes used as successors to a given destination.
SRP Implementation	
sn_T^A	The sequence number of T as known at node A.
d_T^A	The measured distance from node A to T. If all link costs are 1, it is a hop count.
$\mathbf{F}_T^A$	The feasible distance pair (N, D) at node A for T. The proper fraction is ${}^N/_D$.
lc_B^A	The link cost from node A to neighbor B, assumed to be positive and equal to unity if using hop count metrics.
$\star$	An advertisement, for example $sn_T^\star$ is the sequence number in an advertisement for destination T.
$\#$	A solicitation, for example $sn_T^\#$ is the sequence number in a solicitation for destination T. Each issuer adds its own unique identifier $rreqid$.
$\mathbf{LF}_T$	The last-hop feasible distance for destination T. Contained in an advertisement $\mathbf{LF}_T^\star$ or in the advertisement portion of a RREQ.
ld_T	The last-hop measured distance for destination T. Contained in an advertisement $ld_T^\star$ or in the advertisement portion of a RREQ.
$rr_T^\#$	Reset required bit (T bit) for solicitation $\#$ for destination T. Indicates that an invariant ordering violation could occur and the path must be reset.
$\mathcal{O}_T^A$	The ordering of node A for destination T based on sequence number and feasible distance proper fraction; may also refer to an advertisement $\mathcal{O}_T^\star$ or a solicitation $\mathcal{O}_T^\#$.
$\mathcal{O}_\star^A$	The cached ordering for the # corresponding to $\star$, based on the source and $rreqid$.

This need for reliable communication over multiple hops makes DUAL impractical for wireless ad hoc networks. ROAM, an adaptation of DUAL to wireless networks, uses a feasible distance and diffusing computation, so it has a high overhead. LDR also uses a feasible distance, but instead of resetting all predecessors to maintain ordering, it uses a destination-controlled sequence number to denote fresher routes. In many cases, LDR can repair broken routes with localized recovery based on feasible distance ordering, but in some cases a route request and route reply travel over out-of-order nodes whose feasible distances cannot be put in-order. In such cases, LDR requires the route request to travel all the way to the destination, which may issue a route reply with a larger sequence number. The larger sequence number resets the feasible distances along the reply path to establish a new ordering.

The feasible distance establishes an ordering in the graph. Along a path $\{v_k, \ldots, v_0\}$ from node v_k to node v_0, the DUAL condition SNC [8, p. 132] maintains the invariant that $fd^{v_{i-1}} \leq d^{v_{i-1}} < fd^{v_i} \leq d^{v_i} < fd^{v_{i+1}}$. This reduces to an ordering of feasible distances. More generally – and departing from SNC – for the set of all nodes P that are adjacent predecessors of node v_i for the destination v_0, node v_i must satisfy $(\forall k \in P)\ (fd^{v_i} < fd^{v_k})$ and $fd^{v_{i-1}} < fd^{v_i}$. The node v_i, when choosing a new successor v_{i-1}, must maintain this *doubly bounded inequality*. It is possible for node v_i to set its feasible distance to any value in that bound without the possibility of creating loops or breaking the ordering that prevents future loops. In specific, a node may independently decrease its feasible distance to just above the maximum of all successors' reported feasible distance.

The present work generalizes the concept of feasible distance routing to use a sub-divisible feasible distance, such as a lexicographically sorted string or a subset of the real numbers. This allows nodes to stitch together feasible distance orderings that maintain the doubly bounded inequality using locally controlled information.

Section II presents the Split Label Routing (SLR) class of protocols. We show that SLR is loop-free at every instant and that it is satisfiable. Section III describes an implementation of SLR, called Split-label Routing Protocol (SRP) using a label set constructed from proper fractions and a sequence number. Section IV shows that SRP is an instance of SLR. The proofs show that it has correct operation even with a fixed-size label set. Section V presents simulation results showing that SRP out performs existing protocols in terms of delivery ratio, packet latency, and network load.

II. Split Label Routing

We first introduce the principles of Split Label Routing (SLR) assuming an unbounded label set. In such a set, there is no need for path resets, however the size of the labels becomes large. In the next section, we present a specific implementation called Split-label Routing Protocol (SRP), using a fixed label set that grows no faster than a real-time clock. Ordering in SLR is based on an ordinal set, not the hop count or measured distance to a destination. We assume that a routing protocol based on SLR computes a measured distance based on link costs and propagates that information as a QoS parameter with routing advertisements. A node may use the measured distance to choose between possible multi-paths along with any other QoS metrics. The procedures below compute the measured distance assuming symmetric link costs.

Our work is based on maintaining vertex labels in topological order. In SLR, the vertex label set $\mathcal{L}$ has several special properties, one of which is a strict linear order $(\mathcal{L}, <)$. Similar to natural numbers, two elements must satisfy exactly one of $a < b$, $a = b$, or $a > b$, and all $a, b \in \mathcal{L}$ are comparable. A directed graph is in topological order if and only if for every directed edge (i, j), the vertex labels satisfy $\mathcal{L}_j < \mathcal{L}_i$. It is well-known that a digraph is acyclic if and only if it has a topological order [1, p. 77]. Our definition of topological order is reversed from Ahuja [1], where it is defined as $\mathcal{L}_i < \mathcal{L}_j$. In SLR, the node with the minimum label is the destination; it is a vertex with zero out-degree in the digraph.

Let $\mathcal{L}$ be a dense, infinite ordinal set with a greatest element, and a strict ordering operator $<$. It is convenient if the set also has a smallest element, as that is a natural label for the destination of a DAG. Let each element $\varepsilon \in \mathcal{L}$, except the greatest, have a well-defined next-element ε^+, such that $\varepsilon < \varepsilon^+$. The greatest element is not the next-element of any element. We use the symbol ∞ to denote the greatest element. $\mathcal{L}$ is clearly sufficient to label any finite DAG in topological order, because it has at least as many elements as the natural numbers.

A simple example of such an ordinal set is the proper fractions with a least element ${}^{0}/_{1}$ and a maximum element ${}^{1}/_{1}$ [9, p. 35]. Because we will make extensive use of proper fractions in SRP, we review several of their properties. A proper fraction ${}^{m}/_{n}$ is made up of two positive integers m and n, where $m < n$. The range of proper fractions is the open interval $(0, 1)$. The inequality in Eq. 1 [9, p. 35] [11, p. 14] defines how we interpolate between two elements ${}^{m}/_{n} < {}^{p}/_{q}$. It is known as the *mediant*, which has the same numerical value as the mean numerator divided by the mean denominator $\frac{(m+n)/2}{(p+q)/2}$. Eq. 2 defines the next-element operator, which is equivalent to the mediant of ${}^{m}/_{n}$ and ${}^{1}/_{1}$.

$$\frac{m}{n} < \frac{m+p}{n+q} < \frac{p}{q} \tag{1}$$

$$\left(\frac{m}{n}\right)^{+} = \frac{m+1}{n+1} \tag{2}$$

When SLR initializes a graph, the destination, T, may have any label for itself, except the greatest, and all other nodes have the greatest label. The DAG to T is empty; no node has a successor path to T. The initial label for T is arbitrary and may be any label except the greatest. Whatever label T first issues for itself is *de facto* the minimum label.

Because any SLR-based routing protocol maintains a separate DAG per destination, we only consider the operation of such a protocol for one arbitrary destination. In an error-free DAG, only the destination has in-bound arcs and zero out-degree. Due to mobility or channel conditions, however, other nodes may temporarily have positive in-degree and zero out-degree. For each destination, a node i maintains its current label $\mathcal{L}_i$ and a table of successor labels. For each successor link (i, j) node i records the advertised label of j in $\mathcal{S}^i_j$. Node i may then compute the maximum successor label $\mathcal{S}^i_{max}$, which is a strict lower bound for i's own label. If the successor table is empty, $\mathcal{S}^i_{max}$ is the least element of $\mathcal{L}$.

We use a route error procedure similar to AODV, which we only outline here. If a node loses its last successor, it transmits a route error to any and all predecessors. If a node receives a data packet for a destination to which it has no successor, it unicasts a route error message to the last-hop of the data packet. Route error messages do not need to be reliable, because they are repeated for each such data packet.

In the following, we will assume that a request follows the path $\{v_k, \ldots, v_0\}$ in a route computation, where node v_k issues the request and node v_0 issues the reply. Node v_0 may be the destination itself or an intermediate node replying on behalf of the destination. At a node v_i, let $\mathcal{M}_i = \min\{v_k, \ldots, v_{i+1}\}$ be the minimum predecessor label, which is carried in the request. At node v_k, let $\mathcal{M}_k$ be ∞. This value is cached at node i. Manifestly, $\mathcal{M}_i \leq \mathcal{M}_{i+1}$.

When a node k requires a route to the destination, it places its current label in a request $\#$ that is flooded over the network.

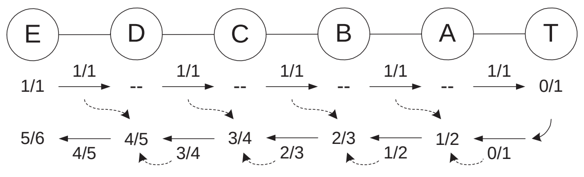

Fig. 1. Initial graph labeling

A flooding mechanism is described in Section III. We assume that a node only processes a given request once. As each node i relays the request, it caches the requested ordering $\mathcal{L}_{\#}$ as $\mathcal{M}_i$. It also caches the last-hop of the request, so a reply may follow the reverse path of the request. Node i places the minimum of $\mathcal{M}_i$, and its own label, $\mathcal{L}_i$, in the relayed request. When the destination, or some other node j with non-zero out-degree and label $\mathcal{L}_j < \mathcal{L}_{\#}$, receives the request, it may send an advertisement $\star$ along the reverse path of the request. Node j places its label $\mathcal{L}_j$ in the advertisement. Each node i along the reverse path creates a successor route to the destination and relays the advertisement. Node i will relabel itself, generally choosing the next-element $\mathcal{L}_\star{}^{+}$, so long as it maintains order (Definition 1). Otherwise, node i will split the ordering of $\mathcal{L}_\star$ and the cached $\mathcal{M}_i$. The advertisement progresses until it reaches k. If a node receives an infeasible advertisement ($\mathcal{L}_\star \not< \mathcal{L}_i$) but has positive out-degree, it may issue a new advertisement based on its current label.

Example 1: Consider the network shown in Fig. 1 which uses the proper fraction ordinal set. Node E issues a request for a route to destination T. Initially, T has the label ${}^{0}/_{1}$ and all other nodes are unlabeled, which is equivalent to having the ${}^{1}/_{1}$ label. Node E places its label in the request, which goes hop-by-hop carrying $\mathcal{L}_{\#} = {}^{1}/_{1}$. When node T receives the request, it issues a reply with the label $\mathcal{L}^T_\star = {}^{0}/_{1}$. When node A receives the reply, it splits $\mathcal{M}_A = {}^{1}/_{1}$ and $\mathcal{L}^A_\star$, taking on the new label $\mathcal{L}_A = {}^{1}/_{2}$. Node A issues a new advertisement with label $\mathcal{L}^A_\star = \mathcal{L}_A$. This process continues with each node splitting the reply label and the cached predecessor label. The final successor graph has the topological ordering $\frac{5}{6} \to \frac{4}{5} \to \frac{3}{4} \to \frac{2}{3} \to \frac{1}{2} \to \frac{0}{1}$. ■

In Definition 1, we state four inequalities that we show maintain a topological order and thus a DAG. An algorithm that chooses a new label $\mathcal{G}$ must be specific to the ordinal set $\mathcal{L}$, and is thus not part of the general SLR description. In Section III, Algorithm 1 satisfies Definition 1 for the proper fraction ordinal set.

Definition 1 (Maintain Order): For an advertisement $\star$ with terminus k, let a node i have a current label $\mathcal{L}_i$ and a cached ordering $\mathcal{M}_i$. If node i chooses new label $\mathcal{G}_i < \infty$ that satisfies Eqs. 3 — 6, the new label $\mathcal{G}_i$ is said to *maintain order* in the graph.

$$\mathcal{G}_i \leq \mathcal{L}_i \tag{3}$$

$$\mathcal{G}_i < \mathcal{M}_i \tag{4}$$

$$\mathcal{L}_\star < \mathcal{G}_i \tag{5}$$

$$\mathcal{S}^i_{max} < \mathcal{G}_i \tag{6}$$

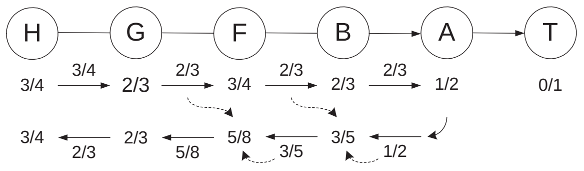

Fig. 2. Graph re-labeling

Eq. 3 ensures that the new label satisfies existing predecessor order. Eq. 4 ensures that the advertisement relayed by node i is feasible along the reverse path to node k, assuming a sufficiently stable network during the route calculation. Eq. 5 is similar to the feasibility condition SNC of DUAL and prevents successor loops. Hop-by-hop, as long as each node issuing $\star$ has a label less than the next node's label, it is impossible for that choice of successor to create a loop. Such an advertisement is called *feasible*. Eq. 6 states that if a node has existing successors, it must keep its label in-order with respect to them.

If all nodes executing SLR maintain order in their choices of labels based on advertisements, then labels are non-increasing with time. This is a direct result of Eq. 3.

It is possible for a node to receive a feasible advertisement (Eq. 5) that does not satisfy all four inequalities. Eqs. 3 – 5 have simultaneous solutions (see Theorem 4), but to satisfy all four a node may need to eliminate certain existing successors to reduce $\mathcal{S}^i_{max}$ such that it is no larger than $\mathcal{L}_\star$.

Example 2: To illustrate the re-labeling process, consider the network established in Fig. 1. At some later time, nodes F, G, and H appear, as shown in Fig. 2. Nodes F, G and H have empty successor sets, but they once knew a route to T, so they have node labels. Node H issues a request with label $\mathcal{L}^H_\# = {}^3/_4$. Node G caches $\mathcal{M}_G = {}^3/_4$, and issues a new request $\mathcal{L}^G_\# = \min\{\mathcal{L}_G, \mathcal{L}^H_\#\}$. Node F receives $\mathcal{L}^G_\# = {}^2/_3$, and caches it as $\mathcal{M}_F$. It relays $\mathcal{L}^F_\# = {}^2/_3$. Node B has a successor to T, but $\mathcal{L}_B \not< \mathcal{L}^F_\#$, so it cannot reply. It relays the request. Finally, node A may reply because $\mathcal{L}_A < \mathcal{L}^B_\#$ and A's successor set is not empty. It sends an advertisement with $\mathcal{L}^A_\star = \mathcal{L}_A$. As in the previous example, nodes B and F relabel themselves based on splitting the cached predecessor minimum and the advertised label. Nodes G and H, however, satisfy Eq. 4 with their current labels, so no change is necessary. All nodes in Fig. 2 now have a successor path to T and the topological order is $\frac{3}{4} \to \frac{2}{3} \to \frac{5}{8} \to \frac{3}{5} \to \frac{1}{2} \to \frac{0}{1}$. In truncated decimal, the labels are $(0.75, .66, .625, .6, .5, 0)$. ■

Theorem 1 (Predecessor Ordering): A node i choosing a new label $\mathcal{G}_i$ that maintains order preserves predecessor ordering. That is, in an existing DAG $D = (U, A)$, node i maintains $\mathcal{G}_i < \mathcal{L}_x$ for all nodes x where $(x, i) \in A$.

Proof: We show that if node i sends an advertisement at time t_0 to create a predecessor link (x, i) at time $t_2 > t_0$, node i may change its own label at any time $t_1 > t_0$ and maintain $\mathcal{L}_i(t_2) < \mathcal{L}_x(t_2)$, regardless of the sequencing of t_1 relative to t_2.

For a given predecessor node x of i, node i transmitted an advertisement that established the link (x, i) at time t_0, being the most recent advertisement from i to x. By Eq. 3, node i's label must be non-increasing with time, so the most recent advertisement is always no greater than an earlier advertisement. Once a predecessor x chooses node i as a successor, it can never decrease its own label to be less than node i's label at time t_0 by Eq. 6. Therefore, node x's label is bound from the bottom by $\mathcal{L}_i(t_0) < \mathcal{L}_x(t > t_0)$ so long as x maintains the successor link or until it receives a new advertisement from i.

At time $t_1 > t_0$, node i changes its label. By Eq. 3, the new label $\mathcal{G}_i(t_1) \le \mathcal{L}_i(t_0)$.

The time t_2 at which node x receives, processes, and creates the link (x, i) may be at any time after t_0 and may be after t_1. However, $\mathcal{L}_i(t_0) < \mathcal{L}_x(t_2)$ by the assumption that the advertisement i sent to create link (x, i) is feasible for x.

If $t_2 < t_1$, then $\mathcal{L}_i(t_2) \le \mathcal{L}_i(t_0)$, so $\mathcal{L}_i(t_2) < \mathcal{L}_x(t_2)$. If $t_2 \ge t_1$, then by Eq. 3 and the transitivity of the partial ordering $\mathcal{G}_i(t_1) \le \mathcal{L}_i(t_1) \le \mathcal{L}_i(t_0)$. At time t_1, node i adopts $\mathcal{L}_i(t_1) \leftarrow \mathcal{G}_i(t_1)$. At time t_2, $\mathcal{L}_i(t_2) \le \mathcal{L}_i(t_1)$, therefore $\mathcal{L}_i(t_2) < \mathcal{L}_x(t_2)$. ■

Theorem 2 (Successor Ordering): Without creating a loop, node i may accept an advertisement $\star$ with label $\mathcal{L}_\star$, so long as $\mathcal{L}_\star < \mathcal{L}_i$.

Proof: Let node j be the issuer of $\star$. At time t_0 it sets $\mathcal{L}_\star \leftarrow \mathcal{L}_j(t_0)$ and transmits $\star$. Node i receives $\star$ at time t_1. We must show that $\mathcal{L}_j(t_1) < \mathcal{L}_i(t_1)$, which maintains the topological order of the graph, and thus node i cannot be on node j's successor path. By assumption, $\mathcal{L}_\star < \mathcal{L}_i(t_1)$, so $\mathcal{L}_j(t_0) < \mathcal{L}_i(t_1)$. Because node labels are non-increasing with time, $\mathcal{L}_j(t_1) \le \mathcal{L}_j(t_0)$, so $\mathcal{L}_j(t_1) < \mathcal{L}_i(t_1)$. ■

Theorem 3 (Loop-freedom): If all nodes maintain order in the graph, SLR is loop-free at every instant.

Proof: By theorems 1 – 2, each node maintains both predecessor and successor ordering at all times. The node labels are therefore in a topological order, which induces a DAG. ■

The next theorem states that a solution to the path finding problem always exists in SLR protocols, so long as the network is stable during a route calculation. It is a general problem of routing protocols that if the underlying network is changing rapidly, convergence becomes difficult or impossible. Simulations show that SRP finds routes even with constant mobility.

Theorem 4 (Existence): Assuming no label changes apart from those caused by a request # and reply $\star$, node i may always find a label $\mathcal{G}$ that maintains order based on $\star$.

Proof: We do not consider Eq. 6, because a node may trivially satisfy it by dropping all existing successors and taking up only the path induced by $\star$.

To show that $\mathcal{G}$ simultaneously satisfies the other three inequalities, we proceed by induction. Let the request path be $\{v_k, \ldots, v_0\}$. Let node v_0 – which does not change its label – issue $\mathcal{L}^0_\star < \mathcal{M}_0$. For the base case, at node v_1, we have $\mathcal{L}^0_\star < \mathcal{M}_0$ implies $\mathcal{L}^0_\star < \mathcal{M}_1$ and $\mathcal{L}^0_\star < \mathcal{L}_1$. Therefore, if we can find a $\mathcal{G}_1$ that satisfies $\mathcal{L}^0_\star < \mathcal{G}_1 < \min\{\mathcal{M}_1, \mathcal{L}_1\}$, we will satisfy all three inequalities. Because the ordinal set is dense, such a label exists.

In the inductive step at some node i, we know that the advertisement issued by node $i-1$ satisfies $\mathcal{L}_{\star}^{i-1} < \mathcal{M}_{i-1}$. This implies $\mathcal{L}_{\star}^{i-1} < \mathcal{M}_i$ and $\mathcal{L}_{\star}^{i-1} < \mathcal{L}_i$. Therefore, $\mathcal{G}_i$ must satisfy $\mathcal{L}_{\star}^{i-1} < \mathcal{G}_i < \min\{\mathcal{M}_i, \mathcal{L}_i\}$. Because the ordinal set is dense, such a label exists. ■

III. Split-label Routing Protocol

The Split-label Routing Protocol (SRP) implements an ordering based on a sequence number sn and a feasible distance proper fraction ${}^{m}/_{n}$ constructed from the ordered pair $\mathbf{F} = (m, n)$. The composite label is denoted $\mathcal{O} = (sn, \mathbf{F})$. As noted above, the set of proper fractions is a dense ordinal set. We use Eq. 1 to split pairs of fractions and Eq. 2 to compute a next-element.

For a practical implementation, we use 32-bit unsigned integers for m and n, which will put an upper bound on the number of times we may interpolate between two fractions without reductions. One observes that the mediant of two proper fractions ${}^{m}/_{n}$ and ${}^{p}/_{q}$ involves the sum $n+q$, which is always greater than $m+p$. The least upper bound on the number of times we may do this in a 32-bit unsigned integer is found from the Fibonacci sequence to be 45 times. Thus, this scheme can mask at least 45 ordering violations along a path without requiring a sequence number increase to reset a path. The maximum number of hops is in the billions.

Similarly to LDR [7], we use a 64-bit time-stamp sequence number. This avoids reset on reboot and avoids wrap-around problems. It avoids wrap-around because we assume a node will not live longer than its real-time clock can count.

SRP is inherently multi-path. A node may choose to use one or more feasible successors, based on advertisements in the network. We do not specify a mechanism to choose good multi-paths or ensure that they are link or node disjoint. A simple implementation of SRP could use a single successor chosen from the min-hop set.

SRP uses a messaging procedure similar to AODV, but with extensive modifications to the packet fields. SRP uses the route request (RREQ), route reply (RREP), route error (RERR) and route acknowledgment (RACK) packets from AODV. The RERR is the same, and we do not discuss it. The RACK is modified to carry the src field and the newly introduced $rreqid$ field from the corresponding RREP packet, but otherwise its use is the same as in AODV. In the following, we only discuss the RREQ and RREP packets.

All multi-hop control packets include an Age field, similar to OSPF [13, pp. 79ff]. A node must increase the age for queuing time and estimated link transmission time. A node must drop any control packet with an age that equals or exceeds the constant $DELETE\_PERIOD$, which we take as 60 seconds. Under certain conditions, a node may forget about its current label for a destination after $DELETE\_PERIOD$, so it is vital that no packets remain in the network that references the forgotten label.

A RREQ has two parts. The solicitation piece is the tuple $\{src$, $rreqid$, dst, $dstseqno$, $\mathbf{F}$, d, *flags*$\}$. The advertisement piece is the tuple $\{src$, $srcseqno$, lfd, ld, *lifetime*, *flags*$\}$, where src and *flags* are shared between the two pieces. The fields src and dst are the unique node identifiers for the source of the RREQ and the sought destination, respectively. The $rreqid$ is a sequence number used to identify the RREQ. It controls flooding and prevents duplicates. d is the measured distance of the RREQ packet as it travels the network, and represents the cumulative traversed link costs. If the source has any information about the destination, it places the known sequence number in $dstseqno$ and stored feasible distance in $\mathbf{F}$. Otherwise, the source sets the flag U bit indicating it has no stored information about the destination. SLR introduces the N bit to indicate that a RREQ is no longer an advertisement for the source and that nodes receiving it cannot build a reverse path from it.

If a node transmitting a RREQ has an active route to the source, it may advertise the route in the RREQ. In this case, the last-hop feasible distance $\mathbf{LF}_{T}^{\#}$ and last-hop measured distance $ld_{T}^{\#}$ are set according to the rules below for advertisements. Note that d is not the same as ld, which measures the unicast distance to the source. The $srcseqno$ is the advertised source sequence number for the route. The *lifetime* is the maximum time a node may cache the advertised route to dst without using it.

A RREP packet is tuple $\{src$, $rreqid$, dst, $dstseqno$, $\mathbf{LF}$, ld, *lifetime*, *flags*$\}$, which is the same as the advertisement portion of a RREQ, except for a few field names. In a RREQ, the advertisement is for the field src while in a RREP, the advertisement is for the field dst, with a similar role reversal for the destination sequence number. The other fields are the same as in a RREQ.

When a node A creates a routing entry for a destination T with next-hop B based on advertisement $\star$, it stores B's ordering as $\mathcal{S}_{T,B}^{A} \leftarrow (sn_{T}^{\star}, \mathbf{F}_{T}^{\star})$. Node A maintains its own label for T in $\mathcal{O}_{T}^{A} \leftarrow (sn_{T}^{A}, \mathbf{F}_{T}^{A})$. SRP also tracks per successor the measured distance to a destination as the cumulative link cost. Because the measured distance is not used in the routing protocol for path computations, we do not discuss it further. Node A is free to use any successor contained in the successor table $\mathcal{S}_{T}^{A}$.

Definition 2 (Route Type): A given node I may have an *active* or *invalid* route for a destination T. The route is invalid if the set $\mathcal{S}_{T}^{I}$ is empty, otherwise it is active. As per AODV, routes time out if not used. They may also become invalid due to channel errors or RERR messages.

Definition 3 (Node State): At a given node I for destination T, node I may be *assigned* or *unassigned*. If I has an ordering $\mathcal{O}_{T}^{I}$, it is assigned. Otherwise, it is unassigned. A node must cache its ordering for each destination for at least $DELETE\_PERIOD$ seconds after the route becomes invalid, as per AODV.

Definition 4 (FD proper fraction ordering): The feasible distance proper fraction has a strict partial order $<$ defined in the normal sense for two fractions. Let $\mathbf{F}_{T}^{A} = (m, n)$ and let $\mathbf{F}_{T}^{B} = (p, q)$. The proposition $\mathbf{F}_{T}^{A} < \mathbf{F}_{T}^{B}$ is true if and only if $mq < np$. Let the notation $(0,1) = (0,1)$ and $(1,1) = (1,1)$.

Definition 5 (Ordering Criteria (OC)): The set

$\mathcal{O} = (sn, \mathbf{F})$ has a strict partial ordering $\succ$. For two instances $\mathcal{O}_T^A$ and $\mathcal{O}_T^B$, the proposition $\mathcal{O}_T^A \succ \mathcal{O}_T^B$ is true if and only if one of the following holds:

$$sn_T^A < sn_T^B \tag{7}$$
$$sn_T^A = sn_T^B \wedge \mathbf{F}_T^B < \mathbf{F}_T^A \tag{8}$$

An unassigned node may be thought to have the maximum ordering $(0, (1, 1))$. An ordering $(sn, (m, n))$ is called finite if ${}^m/_n < 1$. The minimum function $\min\{\mathcal{O}_T^A, \mathcal{O}_T^B\}$ returns $\mathcal{O}_T^B$ if $\mathcal{O}_T^A \succ \mathcal{O}_T^B$ or $\mathcal{O}_T^A$ otherwise.

$\mathcal{O}_T^A \succ \mathcal{O}_T^B$ reads as "B is a feasible in-order successor for A to destination T." The sequence number follows a reversed sense of increasing order than the feasible distance. A higher sequence number implies a fresher route to the destination and supersedes all routes with lower sequence number.

Definition 6 (Ordering Addition): For some proper fraction ${}^p/_q$ and finite ordering $\mathcal{O}_T^A = (sn_T^A, (m, n))$, the notation $\mathcal{O}_T^A + {}^p/_q$ is defined as $(sn_T^A, (m + p, n + q))$. Clearly, if ${}^m/_n < {}^p/_q$, then $\mathcal{O}_T^A + {}^p/_q \succ \mathcal{O}_T^A$.

Definition 7 (Node Initialization): When a node A initializes, it sets $\mathcal{O}_A^A \leftarrow (sn_A^A, (0, 1))$. sn_A^A is a new non-zero sequence number, as described above. For every other node B, A is considered to have $\mathcal{O}_B^A \leftarrow (0, (1, 1))$, but that value does not need to be stored.

When applied to a route advertisement, $\mathcal{O}_T^\star$ means the ordering $(sn_T^\star, \mathbf{LF}_T^\star)$. The $\mathbf{LF}$ is carried in all RREP packets and in the advertisement portion of RREQ packets. For a solicitation, $\mathcal{O}_T^\#$ means the ordering of the request $(dstseqno, \mathbf{F})$. If the U bit is set in #, then the solicitation is considered unassigned for T.

The destination T may respond to any solicitation for itself. Node T is always in-order for any other node because its stored sequence number can never be less than what is known in the network and its feasible distance fraction to itself is the minimum fraction. If T responds to a solicitation with the reset required bit set, it must ensure that the advertisement has a larger sequence number than requested. An intermediate node may send a route advertisement on behalf of T if it satisfies the Start Distance Condition.

Condition 1 (Start Distance Condition (SDC)): Node I may initiate an advertisement $\star$ for a solicitation $\#$ for destination T if I has an active route to T, and either of the following conditions is satisfied: $sn_T^I > sn_T^\#$ or $\mathcal{O}_T^\# \succ \mathcal{O}_T^I \wedge \neg rr_T^\#$

As per LDR [7], a node may be *active*, *passive*, or *engaged* for a routing computation identified by the pair (source, rreqid). The RREQ ID is a source-specific sequence number, used to control the flooding of a RREQ. When a node initiates a RREQ, it becomes active. When a node relays a RREQ, it becomes engaged. Only a passive node may be come active or engaged per (source, rreqid). When a node becomes engaged, it must cache the tuple $\{i, ID_i, \mathcal{O}_T^\#, lasthop\}$ so replies may follow the reverse path.

Procedure 1 (Initiate Solicitation): A node A that requires a route for destination T first checks to see if it is active for T. If it is, A should queue the packet that requires the route. If A is not active for T, it becomes active and increments its $rreqid$. Let ID_A be the incremented identifier. A issues a solicitation for T identified by (A, ID_A) and starts a timer with expiry $t = 2 \cdot ttl \cdot latency$, where ttl is the time-to-live of the broadcast flood and $latency$ is the estimated per-hop latency of the network. If the timer expires, A may retry the solicitation and increase the ttl based on network policies. If after the final attempt, A does not find a route to T, A should inform the packet origins of the failure and drop the queued packets.

If A is assigned for T, A should populate the sequence number and feasible distance fields of the solicitation. Otherwise, A sets the U bit to indicate these fields are unknown.

Procedure 2 (Relay Solicitation): A node B that receives a solicitation (A, ID_A) for destination T firsts checks to see if it is passive for (A, ID_A). If it is not passive, it silently ignores the solicitation. If it is passive, it becomes engaged. If B satisfies SDC, it may issue an advertisement for T. Otherwise, B relays the solicitation as constrained by the ttl. Let the last hop be node C (possibly equal to A) and let the new solicitation be denoted by $\ddagger$. Node B must cache the tuple $\{A, ID_A, \mathcal{O}_T^\#, C\}$ for a sufficient period of time such that all instances of (A, ID_A) have left the network and any advertisements in response to (A, ID_A) have had time to complete.

$$d_T^\ddagger \leftarrow d_S^\# + lc_C^B \tag{9}$$

$$\mathcal{O}_T^\ddagger \leftarrow \begin{cases} (0, (1, 1)) & \text{if } \# \text{ and } B \text{ unassigned} \\ \mathcal{O}_T^B & \text{if } sn_T^B > sn_T^\# \\ \min\{\mathcal{O}_T^B, \mathcal{O}_T^\#\} & \text{if } sn_T^B = sn_T^\# \\ \mathcal{O}_T^\# & \text{otherwise} \end{cases} \tag{10}$$

$$rr_T^\ddagger \leftarrow \begin{cases} 0 & \text{if } \# \text{ and } B \text{ unassigned} \\ 0 & \text{if } sn_T^B > sn_T^\# \\ 1 & \mathcal{O}_T^\# \not\succ \mathcal{O}_T^B,\ \mathbf{F} \text{ overflow} \\ rr_T^\# & \text{otherwise} \end{cases} \tag{11}$$

Eq. 10 ensures that the label of the relayed solicitation has the minimum label of B and $\#$. It corresponds to relaying the minimal label in SLR.

Eq. 11 controls the path-reset request mechanism. The first and second conditions set (or reset) the T bit to zero. If the request and relay node are unassigned, there is no need to request a path reset; any non-zero sequence number suffices. If the relay node's cached sequence number for T is greater than the requested sequence number, node B may reset the T bit because B has increased the requested sequence number by Eq. 10. Any advertisement sent in response to the solicitation functions as a path reset. The third condition demands a path reset if the relay node is out-of-order and the feasible distance fraction in $\#$ would overflow with another split. Let $\mathbf{F}_T^\star = (m, n)$ and let $\mathbf{F}_T^B = (p, q)$. If $n + q$ overflows a 32-bit unsigned number, then B must set the T bit . The sum $n+q$ is an estimate of the reply ordering. The fourth condition reflects that the relay node is in-order and can pass the requested T bit as is.

Algorithm 1:

```
NewOrder($\mathcal{O}_T^A$, $\mathcal{C}_\star^A$, $\mathcal{O}_T^\star$)
(1)    Let $\mathcal{C}_\star^A = (sn^C, (m,n))$ and $\mathcal{O}_T^\star = (sn_T^\star, (p,q))$
(2)    $\mathcal{G}_T^A \leftarrow (0,(1,1))$
(3)    if $sn_T^A < sn_T^\star$
(4)      if $sn^C < sn_T^\star$
(5)        $\mathcal{G}_T^A \leftarrow \mathcal{O}_T^\star + {}^1/_1$
(6)      else if $n+q$ does not overflow
(7)        $\mathcal{G}_T^A \leftarrow (sn_T^\star, (m+p, n+q))$
(8)    else if $sn_T^A = sn_T^\star$
(9)      if $\mathcal{C}_\star^A \succ \mathcal{O}_T^A$
(10)       $\mathcal{G}_T^A \leftarrow \mathcal{O}_T^A$
(11)     else if $n+q$ does not overflow
(12)       $\mathcal{G}_T^A \leftarrow (sn_T^\star, (m+p, n+q))$
(13)   Eliminate any $i \in \mathcal{S}_T^A$ where $\mathcal{G}_T^A \not\succ \mathcal{S}_{T,i}^A$.
(14)   return $\mathcal{G}_T^A$
```

A relay node B records the ordering of a solicitation. The cached ordering of $\#$ is denoted as $\mathcal{C}_\star^B$, where the advertisement $\star$ contains the $(source, rreqid)$ pair used to index in to the RREQ cache. This is equivalent to the minimum predecessor order $\mathcal{M}$ of SLR, but is indexed per solicitation.

As solicitations and advertisements progress through the network, it might happen that a relay node has lower ordering than is contained in the relayed packet. For advertisements, the relay node must discard the advertisement and issue a new advertisement, if possible. It may be that the relay node has a lower label, but an invalid route, in which case it cannot issue a new advertisement. For solicitations, the relay node strengthens the relayed packet, as per Procedure 2.

Procedure 3 (Set Route): When a node A receives a feasible advertisement $\star$ from B for destination T with ordering $\mathcal{O}_T^\star$, it must compute a new ordering $\mathcal{G}_T^A$ for itself by Algorithm 1. If $\mathcal{G}_T^A$ is finite, node A sets $\mathcal{O}_T^A \leftarrow \mathcal{G}_T^A$ and $d_T^A \leftarrow d_T^\star + lc_B^A$; otherwise A must drop the advertisement. If A accepts the route offered by B, it must cache the ordering in it successor table $\mathcal{S}_{T,B}^A \leftarrow \mathcal{O}_T^\star$ and compute $\mathcal{S}_{T,max}^A \leftarrow \max\{\mathcal{S}_T^A\}$.

For advertisements in a RREQ or Hello packet, which do not have a cached $\mathcal{C}_\star^A$, or if A is the terminus of a RREP advertisement, use $\mathcal{C}_\star^A \leftarrow (0,(1,1))$ in Algorithm 1.

If a node is the terminus of an advertisement and the denominator of the feasible distance fraction exceeds a certain threshold MAX_DENOM, the node should request a path reset. To request a reset, the node transmits a unicast RREQ along the forward path with the D bit set. This forces the RREQ to travel along the unicast path to the destination which issues a RREP with a larger sequence number. Each node along the RREP path may set its distance to $\mathcal{O}^\star + {}^1/_1$. The exact value of MAX_DENOM is not important, as long as it is large enough to not happen often and small enough to prevent overflow. We use a value of one billion.

If a node has an active route to the destination of an advertisement and is not itself the terminus of the advertisement, the node should issue a new advertisement for the route. If the node does not have an active route to the destination (because it could not update its routing table based on an infeasible advertisement), the node must not relay the advertisement. If the advertisement is a RREQ packet, the relay node will set the N bit to indicate the RREQ is no longer an advertisement for the source, but will still relay the packet per the Procedure 2. The N bit is not part of the current AODV specification. If the node replying to the RREQ does not have a reverse path, it will set the new corresponding N bit in the RREP indicating such. When the source receives a RREP with the N bit set, it may send a unicast RREQ probe along its forward path with the D bit set, which forces the RREQ to travel to the destination. The source should increase its sequence number to ensure that the reverse path is built. Nodes otherwise should not increase their sequence number when issuing a RREQ.

Procedure 4 (Relay Advertisement): If node A is not the terminus of the advertisement (e.g., the source address in a RREP), and it has an active route to destination T that is feasible for $\mathcal{C}_\star^A$, node A should issue a new advertisement for T upon receipt of an advertisement for the destination. Node A may create or update its own routing table by Procedure 3 upon receiving an advertisement, and uses its RREQ cache to ensure that it does not forward more than one reply per (source, rreqid) pair. Let the new advertisement be denoted by †, then $\mathcal{O}_T^\dagger \leftarrow \mathcal{O}_T^A$ and $d_T^\dagger \leftarrow d_T^A$.

IV. Analysis

We show that SRP is an instance of SLR. To do so, we must show that the ordinal set $\mathcal{O}$ meets the criteria of $\mathcal{L}$ and that the choice of new node labels by Algorithm 1 maintain order. Because the ordering $\mathcal{O}$ is finite in $\mathbf{F}$, it is possible that the implementation will not successfully terminate a route calculation. We show that when it fails to successfully terminate, it does so without creating loops.

We consider advertisements sent in response to solicitations. Advertisements sent in RREQ packets are loop-free because they must satisfy the same routing invariants as advertisements in RREPs, but they are not guaranteed to build paths over the entire network. Because they do not need to satisfy a specific request, nodes are free to ignore Eq. 4. This means that for a RREQ advertisement, a node keeps its existing label, or decreases it to the limits of Eqs. 5 and 6 as desired.

Theorem 5: The ordering $\mathcal{O}_T^A = (sn_T^A, \mathbf{F}_T^A)$ satisfies the conditions of $\mathcal{L}$.

Proof: The requirements for $\mathcal{L}$ is that it be dense, infinite, with a greatest element, a strict partial order $<$, and a next-element operator. $\mathcal{O}$ has a greatest element $(0,(1,1))$. It has a strict partial order $\succ$ (Definition 5). The next-element may be taken as $\mathcal{O} + {}^1/_1$.

To show that $\mathcal{O}$ is dense, consider two distinct orderings $\mathcal{O}^A = (sn^A, (m,n))$ and $\mathcal{O}^B = (sn^B, (p,q))$, and let $\mathcal{O}^A \succ \mathcal{O}^B$. We assume that the numerators and denominators of the proper fraction are not bounded by 32-bit precision. In cases where there is overflow, SRP either asks for a path reset or terminates without adding a successor path. We show by construction that there always exists a distinct ordering

$\mathcal{O}^C = (sn^C, (r,t))$ such that $\mathcal{O}^A \succ \mathcal{O}^C \succ \mathcal{O}^B$. If $\mathcal{O}^A$ and $\mathcal{O}^B$ have distinct sequence numbers, let $\mathcal{O}^C \leftarrow \mathcal{O}^B + {}^{1}\!/_{1}$. If $sn^A = sn^B$, then let $\mathcal{O}^C \leftarrow (sn^A, (m+p, n+q))$. ■

We now show that node labels chosen via Algorithm 1 either maintain order, as per Eqs. 3 – 6, or return an infeasible results which prevents a new link begin added to the successor graph. In both cases, the successor graph remains in topological order and loop-free.

Lemma 1: At a node A, for a finite choice of $\mathcal{G}_T^A$ based on an advertisement $\star$, it is always correct to use $sn_T^\star$.

Proof: As in Algorithm 1, let $\mathcal{C}_\star^A = (sn^C, (m,n))$ and let $\mathcal{O}_T^\star = (sn_T^\star, (p,q))$. Node A's ordering is $\mathcal{O}_T^A = (sn_T^A, (r,s))$. We must show that for a feasible advertisement $\star$ both $sn_T^A \leq sn_T^\star$ and $sn^C \leq sn_T^\star$.

Because the advertisement is feasible at A, $\mathcal{O}_T^A \succ \mathcal{O}_T^\star$, which implies $sn_T^A \leq sn_T^\star$.

As was shown in Theorem 4, a feasible advertisement will satisfy both the current node's label and its predecessor's label along the reverse path because the advertisement was based on the minimum label along the path. So, $sn^C \leq sn_T^\star$. ■

Theorem 6: In an ordered graph, a new ordering computed by Algorithm 1 at node A for destination T in response to an advertisement $\star$ either maintains order or returns the unordered result $(0, (1,1))$, which forces Procedure 3 to ignore $\star$.

Proof: From Lemma 1, we see there are two conditions for Algorithm 1, which we call Fact 1 and Fact 2: $\mathcal{O}_T^A \succ \mathcal{O}_T^\star$ (Fact 1) and $\mathcal{C}_\star^A \succ \mathcal{O}_T^\star$ (Fact 2). We show that in each of the five cases where the algorithm assigns $\mathcal{G}$, that assignment maintains order considering the conditions necessary for that assignment to be returned. All five assignments in Algorithm 1 explicitly satisfy Eq. 5. Line 13 satisfies Eq. 6. Therefore, we must show that in each case, $\mathcal{G}$ satisfies Eqs. 3 – 4.

Case I: Line 2. There are two conditions that return this value. If $sn_T^A > sn_T^\star$, the value is returned, but this contradicts the assumption that $\star$ is feasible at A, so this case never occurs. The second condition is if $sn^C = sn_T^\star$ and $n+q$ overflows. In such a case, we cannot compute a valid node label and must discard the advertisement. It is correct to return the infinite ordering $(0, (1,1))$.

Case II: Line 5. The precondition that $sn_T^A < sn_T^\star$ implies that any ordering $\mathcal{X} = (sn_T^\star, (x,y))$ is in-order for A, so in particular $\mathcal{O}_T^A \succ \mathcal{O}_T^\star + {}^{1}\!/_{1}$, which satisfies Eq. 3. The precondition $sn^C < sn_T^\star$ likewise implies that $\mathcal{C}_\star^A \succ \mathcal{O}_T^\star + {}^{1}\!/_{1}$, which satisfies Eq. 4.

Case III: Line 7. As in Case II, any $\mathcal{X} = (sn_T^\star, (x,y))$ satisfies Eq. 3. By Fact 2 and the precondition for this case that $sn^C = sn_T^\star$, we may find any $\mathcal{G}$ such that $\mathcal{C}_\star^A \succ \mathcal{G} \succ \mathcal{O}_T^\star$, where all three only vary in the feasible distance fraction. This further implies that ${}^{p}\!/_{q} < {}^{m}\!/_{n}$, so $\mathcal{G} \leftarrow \mathcal{O}_T^\star + {}^{m}\!/_{n}$ is such a choice. It maintains the ordering ${}^{m}\!/_{n} < \frac{m+p}{n+q} < {}^{p}\!/_{q}$ and satisfies Eq. 4.

Case IV: Line 10. By the precondition of this case that $\mathcal{C}_\star^A \succ \mathcal{O}_T^A$, the choice $\mathcal{G} \leftarrow \mathcal{O}_T^A$ trivially satisfies Eqs. 3 – 4.

Case V: Line 12. The preconditions of this case imply that $\mathcal{O}_T^A \succeq \mathcal{C}_T^A$, so any $\mathcal{G}$ that satisfies $\mathcal{C}_T^A \succ \mathcal{G} \succ \mathcal{O}_T^\star$ will satisfy Eqs. 3 – 4. The other precondition of this case that $sn^C = sn_T^\star$ makes the solution equivalent to case III. ■

Theorem 7: Solicitations and advertisements in SLR do not loop if there are no node failures.

Proof: For a given calculation (A, ID_A), a node may be passive, engaged, or active. A node enters any calculation at most once. Therefore, the propagation graph of the calculation forms a tree. By using the cached information at engaged nodes, advertisements for the calculation follow paths only in the calculation tree.

If a node unicasts a solicitation, it is guaranteed to not flow in a loop, even if the underlying routing table contains loops. This is because nodes enter the engaged or active states at most once per computation, regardless of the unicast or broadcast nature of the solicitation. Thus, the T bit does not affect the loop-freedom of control packets. ■

If a node fails, it is possible that RREQ and thus RREP packets could loop. This is because a relay node may forget that it is engaged for a computation and become engaged in the computation multiple times, but no more than once per failure. Because RREQ and RREP packets are subject to time-to-live, control packet loops caused by node failures are not permanent. RREP packets will never loop more than one hop, because at that hop the advertisement is infeasible and will be dropped. Such loops cannot create routing-table loops.

TABLE I
PERFORMANCE AVERAGE OVER ALL PAUSE TIMES

protocol	deliv. ratio	net load	latency (sec)
SRP	0.830 ± 0.010	0.905 ± 0.105	0.927 ± 0.084
LDR	0.766 ± 0.010	4.364 ± 0.212	1.172 ± 0.142
AODV	0.741 ± 0.042	4.996 ± 1.062	2.769 ± 0.416
DSR	0.500 ± 0.129	5.394 ± 2.447	5.725 ± 2.370
OLSR	0.710 ± 0.013	4.728 ± 0.198	0.781 ± 0.047

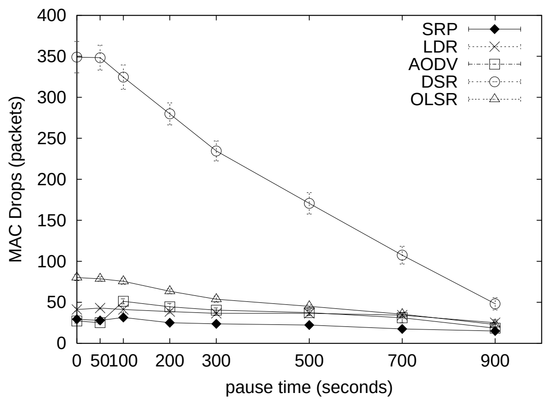

Fig. 3. Average MAC layer drops 100-nodes, 30-flow

V. SIMULATIONS

We present simulation results of uni-path SRP done in GloMoSim [2]. We compare the performance to AODV, DSR, LDR, and OLSR [3].

Like other ad hoc routing protocols, SRP uses several heuristics to improve performance in simulated network

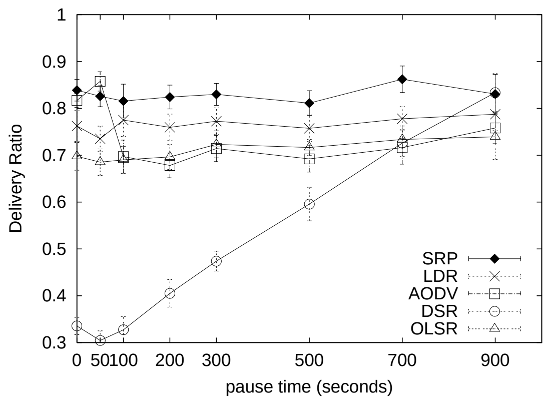

Fig. 4. Delivery ratio 100-nodes, 30-flows

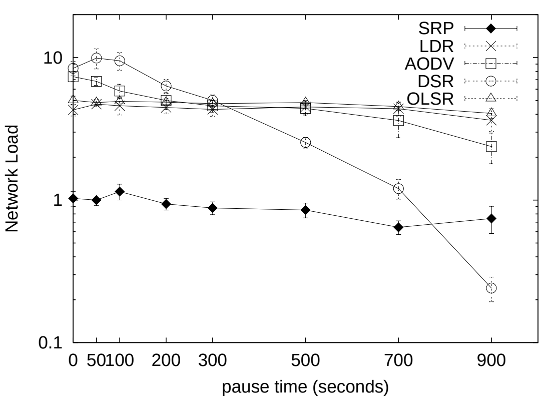

Fig. 5. Network Load 100-nodes, 30-flows

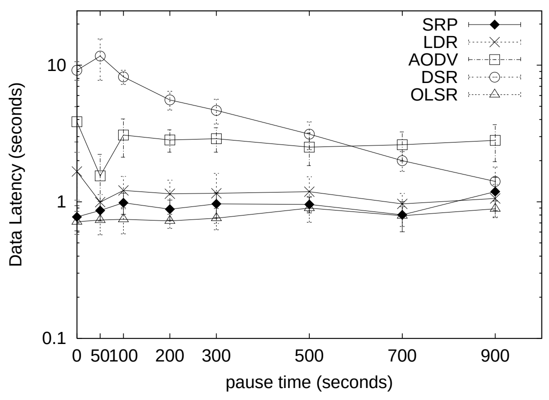

Fig. 6. Delivery ratio 100-nodes, 30-flow

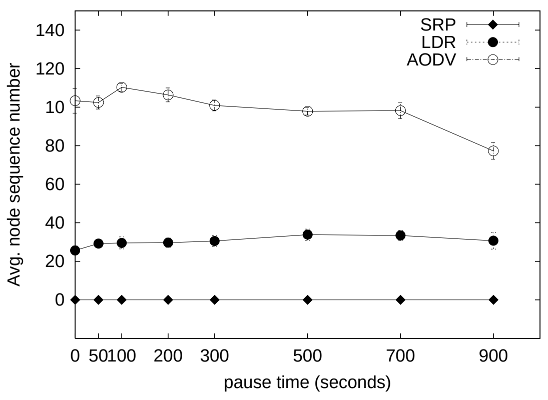

Fig. 7. Average Sequence Number (SRP is exactly 0)

topologies. SRP, along with AODV, DSR, and LDR, uses link-layer unicast loss detection, without hello packets. Our implementation uses a packet cache, similar to DSR. When the link layer reports a packet loss, the routing protocol will break that next hop and seek a new path, resending the dropped packet. DSR in simulation also uses a packet cache (salvaging). AODV uses local repair. We found that under high load, RREQ packets need to travel several hops before allowing a node to reply. This avoids "false positive" RREPs. When a node sends a RREQ, it lies about its ordering. If a node's true ordering is p/q, it sets the RREQ ordering to $(p-1)/(q-1)$. If $p = 1$, the node sets the RREQ ordering to $(p*k-1)/(q*k-1)$, where we used $k = 10000$ in simulation.

Our simulation parameters generally follow those in [16]. We use an 802.11 MAC layer on a 2200m x 600m terrain with 100-nodes and 30 CBR traffic flows. This is the highest traffic rate modelled in [16]. Each CBR packet is 512 bytes and the flows send 4pps, totaling 120pps, or just over 490 kbps network-wide. The channel is 2 Mbps. Each flow lasts for a mean of 60 seconds taken from an exponential variate. At the beginning of each flow, a random source and sink is chosen, and the simulation maintains 30 simultaneous flows. To model mobility, nodes move between 0 m/s and 20 m/s in a random-waypoint pattern with 8 pause times. A pause time of 900s represents no mobility and a pause time of 0s represents constant mobility.

Each data point represents the average of 10 trials over different topologies, traffic endpoints, and random number seeds. For each of the 10 trials, we fix the topology and traffic pattern using off-line generated mobility and packet generation scripts. This means that when we compare, for instance, AODV and SRP in a given trial, they both have the same node mobility and traffic demands. Performance differences should be due entirely to how the routing protocol creates overhead and regulates data traffic.

We present three metrics. The delivery ratio is the total number of CBR packets received divided by the total number of CBR packets transmitted. The network load is the total number of control packets sent divided by the number of CBR packets received. The latency is the mean end-to-end life time in seconds per CBR packet in the network. Each data point represents the average of 10 trials. Below, when we say two measurements are identical, we mean they are statistically identical and have overlapping 95% confidence intervals. Likewise, when we say something is better or worse, we mean it is so with disjoint 95% confidence intervals. In the figures, vertical bars show the 95% confidence interval. In Table I, we show the 95% confidence interval of the averages over all pause times.

Fig 4 shows the delivery ratio of each protocol. AODV and OLSR average around a 73% delivery ratio at this offered load. LDR averages around 77%. It is statistically identical to OLSR at low mobility and slightly better at high mobility. SRP has a higher delivery ratio that the other protocols at almost all times. Looking at Table I, we see that overall, SRP has an 8% ($\frac{.83-.77}{.77}$) higher delivery ratio than LDR, a 12% higher ratio than AODV, and a 17% higher ratio than OLSR.

In our simulations, DSR exhibits poor performance with node mobility. At lower loads than 100-nodes, 30-flows, the performance of DSR is better and generally comparable with OLSR or AODV. However, at this high load, DSR suffers a deep performance drop with mobility. Fig. 3 shows the number of MAC layer drops per node. We see that DSR has a very high MAC layer drop rate, and that it is inversely proportional to the delivery rate. We are not sure why this happens, but the effect is seen in both GloMoSim and Qualnet [18].

Fig. 5 shows the network load. In this semi-log graph, SRP has a much lower load than the other protocols. SRP has 0.2 ($\frac{0.9}{4.4}$) the load of LDR, 0.19 the load of OLSR, and 0.18 the load of AODV. A savings of over 80%.

Fig. 6 shows the packet latencies. OLSR is a pro-active protocol, but is not loop-free at every instant. As a proactive protocol, it sends route advertisements according to a schedule. This leads to high overhead, but generally very low latency because all nodes have routes to all destinations. From the figure, we see that OLSR and SRP have identical latencies, but the average in Table I gives OLSR a slight statistical advantage over SRP. SRP is better than AODV and LDR, according to the figure and the table.

Fig. 7 plots the average node sequence number for SRP, LDR, and AODV. LDR and AODV begin with a sequence number of zero, while SRP begins with a sequence number of one. For the purpose of this graph, we have subtracted one from SRP so all protocols have a base of zero. As expected, AODV has the highest node sequence number because that is the only means for the protocol to prevent loops. LDR has a much lower sequence number, because it can often repair broken paths by only using feasible distance ordering. SRP has identically zero sequence number. In the 80 simulations shown in the graphs (8 pause times, 10 trials each), SRP never needed to increment the sequence number to repair a path. The maximum denominator stayed under 840 million. In general, however, we would expect over time the need to reset a path due to 32-bit overflow.

VI. Conclusion

We presented a new approach to loop-free routing called Split Label Routing (SLR) and a specific protocol for on-demand routing in wireless networks called Split-label Routing Protocol (SRP). Routing protocols based on SLR maintain node labels in a topological order, which induces a directed acyclic graph per in-use destination. SLR generalizes the concept of a feasible distance from DUAL [8] and does not require a diffusing computation. The novel feature of SLR is that it uses a dense ordinal set, so it may insert new nodes and relabel existing paths without needing to relabel existing predecessors. SRP is an instance of the SLR class, and uses a ordinal set comprised of a sequence number and a feasible distance proper fraction ${}^{m}/_{n}$, where m and n are positive integers, with $m < n$, including a zero element and one element.

Results from simulation experiments illustrate that SRP outperforms other state-of-the-art protocols at high load. SRP has better delivery ratio and much lower network load than other protocols. Its packet latency is almost as good as OLSR, a pro-active routing protocol for wireless networks.

Our description of SRP does not incorporate fraction reductions. We would like to find a method to interpolate relatively prime proper fractions that yields a relatively prime proper fraction. Our current research is developing methods based on walking a Farey tree [12]. Another open area is how to choose good multipaths to maximize link or vertex disjointness.

Acknolwdgement

This work was supported in part by the U.S. Air Force/OSR under grant No. F49620-00-1-0330, and by the Baskin Chair of Computer Engineering at the University of California at Santa Cruz.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993. Includes bibliographical references (p. 821-839) and index. English.

[2] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A scalable network simulation environment. Technical Report 990027, UCLA Computer Science Department, 1999.

[3] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. IETF Internet draft, draft-ietf-manet-olsr-06.txt, Sep 2001.

[4] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *Wireless Networks*, 1:61 – 81, 1995.

[5] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, Aug. 1980.

[6] E. M. Gafni and D. P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Trans. Comm.*, COM-29(1):11–18, Jan. 1981.

[7] J. J. Garcia-Luna-Aceves, M. Mosko, and C. Perkins. A new approach to on-demand loop free routing in ad hoc networks. In *PODC 2003*, pages 53 – 62, July 2003.

[8] J. J. Garcia-Lunes-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking*, 1(1):130–41, Feb. 1993.

[9] E. V. Huntington. *The Continuum and other Types of Serial Order*. Harvard University Press, Cambridge, MA, second edition, 1942.

[10] D. Johnson, D. Maltz, Y.-C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (DSR). IETF Internet draft, draft-ietf-manet-dsr-07.txt, Feb 2002.

[11] A. Y. Khinchin. *Continued Fractions*. Dover Publications, Mineola, NY, third edition, 1997.

[12] D. W. Matula and L. D. McFearin. A $p \times p$ bit fractional model of binary floating point divison and extremal rouding cases. *Theoretical Computer Science*, 291(2):159 – 82, Jan. 2003.

[13] J. T. Moy. *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley, Reading, MA, USA, 1998.

[14] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, pages 1405–13 vol.3, Apr. 1997.

[15] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. IETF Internet draft, draft-ietf-manet-aodv-10.txt, Mar 2002.

[16] C. Perkins, E. Royer, S. Das, and M. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 8(1):16 – 28, Feb 2001.

[17] J. Raju and J. J. Garcia-Luna-Aceves. A new approach to on-demand loop-free multipath routing. In *IC3N'99*, pages 522–7. IEEE, Oct. 1999.

[18] Scalable Network Technologies. Qualnet simulator. http://www.scalable-networks.com, 2001.