

Using Labeled Paths for Loop-free On-Demand Routing in Ad Hoc Networks *

Hari Rangarajan
Computer Engineering Department
University of California
Santa Cruz, CA 95064
hari@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
Computer Engineering Department
University of California
Santa Cruz, CA 95064
jj@soe.ucsc.edu

ABSTRACT

We present the *Feasible Label Routing (FLR)* protocol for mobile ad hoc networks, which uses path information to establish routes to destinations on demand. FLR enables loop-free incremental (hop-by-hop) routing of data packets using only the addresses of their destinations. Like the dynamic source routing (DSR) protocol, FLR avoids the need for any time-stamps or sequence numbers by the use of path vectors exchanged when routes are established or repaired. Instantaneous loop freedom is attained by using path information for a destination as labels with which routers are ordered lexicographically with respect to the destination, i.e., FLR ensures that the labels of routers for a given destination become “smaller” the closer they are to the destination. Simulation experiments in Qualnet show that the performance of FLR is far better than the performance of the ad-hoc on-demand distance vector (AODV) protocol, the dynamic source routing (DSR) protocol, and the optimized link state routing (OLSR) protocol, in terms of the packet delivery ratio and average delivery latencies achieved, as well as the overhead incurred in the network.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Wireless communication—*ad hoc networks*; C.2.2 [Network Protocols]: Routing protocols—*On demand*

General Terms

Algorithms.

Keywords

routing, loop-free, ad hoc network, routing invariant, path information.

*This work was supported in part by the US Air Force/OSR under Grant No. F49620-00-1-0330, and by the Baskin Chair of Computer Engineering at UCSC.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiHoc'04, May 24–26, 2004, Roppongi, Japan.

Copyright 2004 ACM 1-58113-849-0/04/0005 ...\$5.00.

1. INTRODUCTION

Many routing protocols have been proposed for mobile ad hoc networks (MANET). Pro-active protocols maintain routes to every possible destination in the network. These protocols include the Optimized Link State Routing (OLSR) protocol [1], the Destination Sequenced Distance Vector Routing (DSDV) protocol [10], Source Tree Adaptive Routing (STAR) [5], and Topology Broadcast Reverse Path Forwarding (TBRPF) [8]. On-demand protocols establish routes only to those destinations for which there is traffic, and ensure loop freedom at every instant to minimize control overhead. Examples of these protocols are the Ad hoc On-demand Distance Vector (AODV) [11], Dynamic Source Routing (DSR) [7], and the Temporally Ordered Routing Algorithm (TORA) [9]. All on-demand protocols attempt to maintain loop-free routing tables, because routing loops could lead to excessive signaling overhead trying to repair loop-free routes. The three basic approaches used in on-demand protocols to eliminate loops are nodal synchronization, destination sequence numbers, and source routing.

The temporally-ordered routing algorithm (TORA) uses a link-reversal algorithm [2] to maintain one or more loop-free paths created with a query-reply process similar to that used in DSR and AODV. However, TORA and other protocols using nodal synchronization over multiple hops [13] incur excessive overhead.

AODV uses per-destination sequence numbers to maintain loop freedom. The sequence number carried in a route request ensures that it elicits only fresher route replies with an equal or higher sequence number. After a link failure, a node increases its sequence number for a destination and invalidates the route. Error updates can be sent unreliably, because increasing the sequence number invalidates the route entry of all upstream nodes. The key limitation with this is that it prevents responses from nodes that are closer to the destination but have an older sequence number, even if they have a valid loop-free path to the destination. Consequently, the likelihood that the destination itself must resolve a route request is high, because the destination is the only node that can increase its own sequence number.

The Labeled Distance Routing (LDR) protocol [4] was recently proposed to improve on the way in which AODV uses sequence numbers. LDR is an on-demand routing protocol based on a dual invariant consisting of destination sequence numbers and feasible distances [3]. A feasible distance in LDR roughly corresponds to the smallest distance to a destination attained by a node for its current sequence num-

ber for the destination. The destination sequence number is used to “reset” the distance to a destination, i.e., to allow a node to accept a next hop that reported a distance larger than the node’s feasible distance. Using feasible distances in LDR makes it more likely for nodes other than the destination to resolve route requests, which improves the performance significantly [4].

DSR is a well-known example of using path information on demand to avoid looping in MANETs. DSR enforces loop-free routes to a destination by carrying the path traversed in the route request and the reverse path is then used to source route data packets to the destination. After a link failure, reliable error updates are sent to the source, so that a new route can be searched. The limitation with this approach is that data-packet headers must specify source routes to avoid loops, which incurs additional processing overhead for each data packet. Packet salvaging can be used in DSR to recover from link failures locally by re-routing data packets along an alternative source route; however, this approach may result in the formation of temporary loops, and requires a mechanism to detect data packets flowing over those loops. An implicit source routing approach [6] has also been proposed to reduce the extra overhead of source routes in data packets by carrying flow Id’s instead of the entire source route. However, this scheme still requires additional per-packet processing of IP option headers, and loops formed are detected by using the time-to-live (ttl) of the data packets. Although several variants on the use of path information in on-demand routing protocols have been proposed (e.g., [14, 15, 16]), they require packet headers to specify source routes or the path traversed to avoid looping.

Using path information in an on-demand routing protocol offers an alternative to using destination sequence numbers that must be frequently “reset” by the destinations, as well as to synchronizing the routing-table update steps of several nodes, which has been shown to incur even more overhead than resetting destination sequence numbers. However, none of the on-demand routing protocols based on path information proposed to date are able to support instantaneous loop-free routing when data packets are forwarded incrementally (i.e., hop by hop) using only the addresses of their destinations. This constitutes the motivation for the work presented in this paper, which provides the following key contributions:

- Extending the notion of “feasible distances” (used in the past in such protocols as the diffusing update algorithm (DUAL) [3] and more recently the labeled distance routing (LDR) protocol [4]) to the concept of *feasible labels*, such that nodes are lexicographically ordered for a given destination according to their paths to the destination.
- Introducing and proving sufficient conditions for loop freedom based on feasible labels corresponding to path information to destinations, which can be applied to other routing protocols in the future.
- Presenting the Feasible Label Routing (FLR) protocol, which is the first on-demand routing protocol that uses path information only in its signaling, while supporting loop-free incremental forwarding of data packets when the header of each data packet simply states the address of the intended destination.

In a nutshell, just as AODV orders nodes according to increasing sequence numbers to a destination, and DUAL orders nodes according to decreasing feasible distances to a destination, FLR orders nodes according to labels that become lexicographically smaller as the destination is approached. FLR achieves this using route requests, route replies, and route errors that carry path information and are similar to the messaging structure of other on-demand routing protocols.

In FLR, each node maintains a label for a destination for which it needs to forward traffic, where a label is simply a path to a destination. The *feasible label* of a node for a destination is simply the lexicographically smallest path to the destination attained by the node since the last time it had to send route errors to its neighbors. A node can accept a route reply for a destination generated by the destination or any intermediate node if the label to the destination advertised in the reply is “smaller” than the feasible label at the node that issued the route request. Hence, to realize instantaneous loop freedom in FLR, route requests carry the minimum feasible label that must be satisfied by the node issuing a route reply, and a route reply carries the *current label* of the node forwarding the reply.

Section 2 presents sufficient conditions for loop-free routing using feasible labels. Section 3 describes FLR and several optimizations. Section 4 summarizes how some of our results could be used to enhance AODV and DSR. Section 5 shows an example of FLR in operation. Section 6 analyzes the instantaneous loop freedom of FLR. Section 7 compares the performance of FLR against AODV and DSR, which are two popular on-demand protocols, and OLSR, which is a popular pro-active protocol. The results show that FLR attains far better performance than DSR, AODV, and OLSR, in terms of end-to-end delays, packet-delivery ratios, and control overhead. Section 8 provides our concluding remarks.

2. SUFFICIENT CONDITIONS FOR LOOP FREEDOM USING LABELS

We specify and prove new sufficient conditions for loop-free routing that are applicable to on-demand and pro-active routing protocols based on path information. These conditions extend the notion of ordering distances to a destination, such that nodes closer to the destination have shorter distances to it, into the notion of ordering the paths to a destination lexicographically, so that nodes closer to the destination have labels representing their paths that are lexicographically smaller.

Table 1 summarizes the terminology used to describe the sufficient conditions for loop-free routing, as well as to specify FLR in the next section. The values of labels, link costs, or routing-table entries are functions of time, but the time for which the values of such functions apply are specified only when needed.

A *label* L of size n , $n \geq 0$, is an ordered sequence of unique elements $\{e_1(L), e_2(L), \dots, e_{n-1}(L), e_n(L)\}$, where each element $e_i(L) \in L$ consists of a valid node identifier (denoted by $e_i^{id}(L)$) and the cost of the link from $e_i(L)$ to $e_{i+1}(L)$ (denoted by $e_i^c(L)$). For the last element of L , $e_n^c(L) = 0$.

The label of node B for destination D is denoted L_D^B , and L_{DB}^A denotes the label for destination D reported by node B and stored by node A . In a routing protocol based on path information, when node A chooses node B as its successor

for destination D , the path from A to D consists of the concatenation of the link from A to B with the path from B to D . Hence, if we denote the concatenation of two labels L_1 and L_2 by $L_1 \oplus L_2$, if node A chooses node B as its successor to destination D , its label equals $L_D^A = (A, c_B^A) \oplus L_{DB}^A$, where c_B^A is the cost of the link from A to B .

The weight of a label L of size $n \geq 0$ is denoted by the function $W(L)$, where $0 \leq W(L) \leq \infty$, and is defined by

$$W(L) = \begin{cases} \infty & \text{if } n = 0 \\ \sum_{i=1}^n e_i^c(L) & \text{otherwise} \end{cases} \quad (1)$$

Relational operators on two labels L_1 and L_2 of sizes n_1 and n_2 , respectively, are defined as follows:

$$L_1 = L_2 \text{ if } \begin{cases} W(L_1) = W(L_2) \wedge \\ \{ n_1 = n_2 \wedge \\ (e_j^{id}(L_1) = e_j^{id}(L_2) \forall j \mid 1 \leq j \leq n_1) \} \end{cases} \quad (2)$$

$$L_1 > L_2 \text{ if } \begin{cases} W(L_1) > W(L_2) \vee \\ (W(L_1) = W(L_2) \wedge n_1 > n_2) \vee \\ \{ (W(L_1) = W(L_2) \wedge n_1 = n_2) \wedge \\ (\exists j \mid \{ 0 < j \leq n_1 \wedge (e_j^{id}(L_1) > e_j^{id}(L_2)) \wedge \\ (e_h^{id}(L_1) = e_h^{id}(L_2) \forall h \mid j + 1 \leq h \leq n_1) \}) \} \end{cases}$$

$$L_1 < L_2 \text{ if } \begin{cases} W(L_1) < W(L_2) \vee \\ \{ W(L_1) = W(L_2) \wedge n_1 < n_2 \} \vee \\ \{ (W(L_1) = W(L_2)) \wedge (n_1 = n_2) \wedge \\ (\exists j \mid \{ 0 < j \leq n_1 \wedge (e_j^{id}(L_1) < e_j^{id}(L_2)) \wedge \\ (e_h^{id}(L_1) = e_h^{id}(L_2) \forall h \mid j + 1 \leq h \leq n_1) \}) \} \end{cases}$$

The following sufficient conditions for loop-free routing using labels extend the conditions introduced for DUAL, which were based on distances to destinations [3]. The new conditions assume that (a) nodes communicate to neighbors their labels to destinations pro-actively or on demand, and (b) each node maintains a routing table specifying the next hop(s) to some or all destinations in the network, and the labels notified by other nodes for some or all destinations.

The label reported by node B for destination D and stored at node A is denoted by L_{DB}^A , and L_D^{*A} and L_{DB}^{*A} denote the minimum values attained by L_D^A and L_{DB}^A , respectively.

Feasible Label Condition (FLC): Node A can make node B its successor for destination D after processing an input event if $L_{DB}^A < L_D^{*A}$. If no neighbor exists with a smaller reported label than L_D^{*A} , then node A must keep its current successor if it has any.

Extended Label Condition (ELC): Node A can make node B its successor for destination D after processing an input event if $(A, c_B^A) \oplus L_{DB}^A < L_D^{*A}$, where (A, c_B^A) states the identifier of A and the cost of the link from A to B . If no neighbor exists with a smaller reported label than L_D^{*A} , then node A must keep its current successor if it has any.

Next-hop Label Condition (NLC): Node A can make node B its successor for destination D after processing an input event if $L_{DB}^A < L_{DS}^{*A}$, where S is node A 's current successor to destination D . If no neighbor exists with a smaller reported label than L_{DS}^{*A} , then node A must keep its current successor if it has any.

Table 1: Terminology

Notation	Description
L_D^A	The stored label for destination D at node A .
s_D^A	The successor for destination D at node A .
L_∞^A	An invalid route or a label of infinite cost.
FL_D^A	The smallest label assigned by node A for D since A sent its last route error for D .
PS_D^A	The set of neighbors of node A to whom node A has sent RREPs for D .
c_B^A	The cost of the link from node A to neighbor B .
rep	Superscript used for variables in a route reply.
req	Superscript used for variables in a route request.
rt_D^A	The state of the routing-table entry for D at node A . Either null (ϕ), valid, or invalid.

THEOREM 1. *Using FLC whenever nodes choose their successors to destination D is sufficient to ensure that no routing-table loops are created for destination D .*

PROOF. FLC is equivalent to SNC from DUAL, which is shown to be loop-free [see [3], Theorem 1, pp. 132ff]. FLC uses feasible labels, which simply extrapolate the ordering properties of feasible distances used in DUAL by using the relational operators on labels stated in Eq. 2. \square

As Theorem 1 shows, using FLC guarantees that routing-table loops are not created, and the other conditions can also be shown to be sufficient to ensure loop-free routing using labels. However, nodes must keep choosing as successors to destinations those neighbors that offer labels that are always “smaller” than the smallest labels they have attained. Consequently, even if there are physical paths from node A to destination D , it is possible for node A to be unable to pick any neighbor as successor to D if FL_D^A is not larger than any label reported by a neighbor of node A .

In practice, additional mechanisms are needed together with one of the sufficient conditions for loop freedom stated above to allow nodes to increase their feasible labels to destinations safely (without causing loops). One approach to allowing a node to safely increase its feasible label for a destination would be the use of diffusing computations as in DUAL or ROAM [13]. However, a diffusing computation requires the origin of the computation to coordinate the updating of its successor for a given destination with nodes many hops away whose paths to the destination include the origin of the computation. The next section introduces FLR, which implements much more efficient mechanisms for allowing nodes to increase their feasible labels to destinations, without creating routing-table loops.

3. FEASIBLE LABEL ROUTING PROTOCOL (FLR)

3.1 Principles of Operation

FLR uses route request (RREQ), route reply (RREP), and route error (RERR) messages similar to that of other on-demand routing protocols. The routing-table entry at node A for destination D includes the current label (L_D^A), the feasible label (FL_D^A), the successor (s_D^A), and the predecessor set for D (PS_D^A). Labels are as defined in Section 2, and FL_D^A is the lexicographically smallest label that node A has obtained for D since it sent its last RERR for D . PS_D^A

is the set of predecessors for destination D , which are those neighbors of node A to whom node A has sent RREPs for D . The superscripts req and rep are used for variables included in a RREQ and RREP, respectively.

FLR is based on the following four rules (called conditions), which apply for a given destination D independently of other destinations. These rules are used to implement FLC in an on-demand routing context and to permit nodes to increase their feasible labels when needed, without causing routing-table loops. The rules make use of the minimum feasible label of any of the nodes that relayed or originated a RREQ for destination D (denoted by MFL_D^{req}), the path traversed by the RREQ (denoted $path_D^{req}$), and the current label of the node that transmits a RREP for D (denoted by L_D^{rep}).

ALC: (Accept Label Condition). When node A receives a RREP from node B for destination D , then

If No Local Repair Used:

Node A sets $s_D^A \leftarrow B$ if $L_D^A = L^\infty$ or ($L_D^A \neq L^\infty$ and $(A, c_B^A) \oplus L_D^{rep} < L_D^A$ and $L_D^{rep} < FL_D^A$).

If Local Repair Used:

Node A sets $s_D^A \leftarrow B$ if $((A, c_B^A) \oplus L_D^{rep} < L_D^A$ and $L_D^{rep} < FL_D^A$). Node A sends a RERR reliably to its neighbors in PS_D^A and then sets $s_D^A \leftarrow B$ and $PS_D^A \leftarrow \phi$ if $((A, c_B^A) \oplus L_D^{rep} < L_D^A$ and $L_D^{rep} \not< FL_D^A$).

SLC: (Start Label Condition). Node I can issue a RREP responding to a RREQ for destination D if I has an active route to D and $L_D^I < MFL_D^{req}$.

MLC: (Minimum Label Condition). If node A relays a RREP for destination D , it sets $L_D^{rep} = L_D^A$. Node A relays a RREQ for destination D only if $A \notin path_D^{req}$ and sets $MFL_D^{req} = \min\{MFL_D^{req}, FL_D^A\}$.

RLC: (Reset Label Condition). If node A must change s_D^A , then it sets $L_D^A \leftarrow L^\infty$ and

If No Local Repair Used: Node A sends a RERR reliably to its neighbors in PS_D^A before setting $PS_D^A \leftarrow \phi$ and sending a RREQ for D with $MFL_D^{req} = FL_D^A$.

If Local Repair Used:

Node A sends a RREQ containing $MFL_D^{req} = FL_D^A$.

When no local repairs are used, node A first sends a RERR to block those neighbors using A as next hop to D with a reliable RERR, and then it attempts to obtain a new next hop to D with a RREQ (RLC). A RREQ traverses loop free paths and carries the minimum feasible label of any of its relays (MLC), and only a node with a label strictly smaller than the minimum feasible label of the RREQ can create a RREP (SLC). If local repairs are used, node A attempts to obtain a new successor with a label smaller than its own feasible label, and blocks those neighbors using A as next hop to D with a reliable RERR if no such neighbor is found and its feasible label has to be changed.

3.2 Information Stored and Exchanged

The routing information used by nodes running FLR is maintained in the routing table. The routing-table entry for a destination D at a given node A specifies: (a) The successor to D (s_D^A), (b) the predecessor set for D (PS_D^A), (c) the current path label (L_D^A), and (d) the feasible label for D (FL_D^A).

A RREQ consists of the tuple $\{dst, src, reqid, MFL_{dst}^{req}, path_{dst}^{req}\}$, where src is the identifier of the source of the

RREQ seeking a path to the destination dst . The $reqid$ is an identifier assigned by src to the RREQ, such that the pair $(src, reqid)$ is unique. The $path_{dst}^{req}$ field is a list of $\{id, c\}$ pairs specifying the nodes (id) that were traversed by the RREQ and the associated link cost c of each hop.

A RREP consists of the tuple $\{dst, src, L_{dst}^{rep}, ttl, path_{dst}^{rep}\}$. The field src specifies the origin of the RREQ that caused the RREP. The field ttl is the time remaining for the route to dst at the node transmitting the RREP. The label L_{dst}^{rep} is the current label of the node transmitting the RREP. The field $path_D^{rep}$ is a list of $\{id, c\}$ pairs specifying the nodes (id) that were traversed by the RREQ that originated the RREP and the associated link cost c of each link.

A RERR message consists of the tuple $\{orig, reset\}$. The field $orig$ is the node generating the route error message. The field $reset$ is the list of destinations for which the origin of the RERR must reset its feasible label, and informs the recipients of the RERR that its origin needs a new route for each destination listed in $reset$.

3.3 Basic Route Maintenance

3.3.1 Initiating a RREQ

Node A is said to be *active* for the route computation for destination D (i.e., the RREQ) that is uniquely identified by the pair (A, ID_A) . A node can be the origin of at most one RREQ for the same destination at any given time. The RREQ (A, ID_A) terminates when either node A attains a feasible label for destination D (by receiving a RREP for its RREQ or otherwise) or the timer for its RREQ expires.

A node A that requires a route for destination D buffers the data packets if it is active for destination D . Otherwise, it issues a RREQ $\{D, A, reqid = ID_A, MFL_{dst}^{req}, path_D^{req}\}$ by setting $ID_A \leftarrow \text{incremented request counter}$; $reqid \leftarrow ID_A$; $MFL_D^{req} \leftarrow FL_D^A$; $path \leftarrow \phi$; and $RREQ\ timer \leftarrow (2.ttl.latency)$, where ttl is the time-to-live of the broadcast flood and latency is the estimated per-hop latency of the network.

If node A receives no RREP for destination D after the expiry of its timer for RREQ (A, ID_A) , it retries a new RREQ with an increased ttl . If after a number of attempts node A does not receive a RREP, a failure is reported to the upper layer. The number of hops that a RREQ can traverse is controlled externally from the RREQ by means of the TTL field of the IP packet in which a RREQ is encapsulated, or by other means.

3.3.2 Relaying RREQs

A node relaying a RREQ originated by another node is said to be *engaged* in the RREQ. A node may be engaged in multiple RREQs for the same destination, and a node that engaged in a RREQ maintains no explicit state for it.

When node B receives RREQ $\{D, A, reqid = ID_A, MFL_{dst}^{req}, path_D^{req}\}$, it first determines its own status for (A, ID_A) . If B is active (i.e., $B = A$) or engaged (i.e., B is listed in $path_D^{req}$) in the computation (A, ID_A) , it silently drops the RREQ. Otherwise, node B is said to be passive. If this is the case and SLC is satisfied (i.e., $L_D^B < MFL_D^{req}$), then node B issues a RREP (Section 3.3.4). Else, if SLC is not satisfied, node B relays the RREQ with $MFL_D^{req} \leftarrow \min\{FL_D^B, MFL_D^{req}\}$ and $path_D^{req} \leftarrow (B, c_D^B) \oplus path_D^{req}$.

3.3.3 Route Failures

Node A sets $PS_D^A \leftarrow \phi$, $s_D^A \leftarrow nil$, $L_D^A \leftarrow L^\infty$ if no data packets have been forwarded using this route entry for *active_route_time* seconds.

Node A carries out the following steps if its predecessor set (PS_D^A) is not empty and either node A receives a link-level notification that its link to s_D^A has failed, or node A receives a RERR from s_D^A :

No Local Repair Used: Node A sends a RERR reliably to all the nodes in PS_D^A , sets $PS_D^A \leftarrow \phi$, and after taking those steps it sends a RREQ for D (Section 3.3.1) if node A is a source of data packets for D .

Local Repair Used: Node A originates a RREQ for D (Section 3.3.1) and waits for a RREP. If node A receives a RREP for D , then it proceeds as stated in Section 3.3.5.

3.3.4 Initiating and Processing RREPs

When node I processes RREQ $\{D, A, reqid = ID_A, MFL_D^{req}, path_D^{req}\}$ and SLC is satisfied (i.e., $L_D^I < MFL_D^{req}$), it issues a RREP $\{D, src, L_D^{rep}, ttl, path_D^{rep}\}$ where $L_D^{rep} \leftarrow L_D^I$. The destination or the intermediate node initiating the reply sets $path_D^{rep}$ to the reverse path $path_D^{req}$ traversed by the RREQ.

If node A receives RREP $\{D, src = S, L_D^{rep}, ttl, path_D^{rep}\}$, it determines if it is the source S of the RREQ that caused the RREP. If so, it proceeds as Section 3.3.5 describes. If $A \neq S$, then it updates its routing table as Section 3.3.5 states, and forwards the RREP along $path_D^{rep}$ setting $L_D^{rep} \leftarrow L_D^A$. Node A also adds the next hop of the RREP, B , to PS_D^A .

3.3.5 Adding and Updating Routes

By definition, if node A has no routing-table entry for destination D , its label and feasible label for D are assumed to have infinite cost. If link (A, B) changes its cost c_B^A , then for each destination D for which $s_D^A = B$, node A updates c_B^A in L_D^A and $FL_D^A \leftarrow \min\{FL_D^A, L_D^A\}$.

When Node A receives RREP $\{D, S, L_D^{rep}, ttl, path_D^{rep}\}$ from neighbor B , then it sets a temporary label $TL_D^A \leftarrow (A, c_B^A) \oplus L_D^{rep}$ and carries out the following steps:

No Local Repair Used: If $L_D^A = L^\infty$ or $L_D^A \neq L^\infty \wedge TL_D^A < L_D^A \wedge L_{rep}^A < FL_D^A$, then node A sets $s_D^A \leftarrow B$, $L_D^A \leftarrow TL_D^A$, and $FL_D^A \leftarrow \min\{FL_D^A, TL_D^A\}$.

Local Repair Used:

1. If $L_D^{rep} < FL_D^A$ and $L_D^A > TL_D^A$, then node A sets $L_D^A \leftarrow TL_D^A$, $FL_D^A \leftarrow \min\{FL_D^A, TL_D^A\}$, and $s_D^A \leftarrow B$.
2. If $L_D^{rep} \not< FL_D^A$ and $L_D^A = L^\infty$, then node A sends a RERR reliably to all the nodes in PS_D^A , and then sets $PS_D^A \leftarrow \phi$, $s_D^A \leftarrow B$, $L_D^A \leftarrow TL_D^A$, and $FL_D^A \leftarrow \min\{FL_D^A, TL_D^A\}$.

3.4 Route Maintenance Optimizations

Several optimizations can be added to FLR's basic operation, in terms of how cached labels and path information included in RREQs are used.

3.4.1 Inferring Paths to Relays

The labels already stored in the routing tables for some destinations can be used to infer paths to other destinations that appear as relays in such labels. Because FLR assumes that data packets are forwarded incrementally (hop by hop) using only the addresses of the intended destinations, a node

can forward a data packet for a destination to a neighbor only if that neighbor has a valid route to that destination, for otherwise the packet would simply elicit a RERR.

FLR can be extended with Source Routed Requests (SRREQ) that carry the path along which the request is forwarded before it is answered by a relay or the destination. The path to a destination is determined by executing a path selection algorithm (i.e., Dijkstra) over the current set of all active labels (paths) in the routing table. SRREQs over calculated feasible paths avoid flooding the network and exploit the path information already cached at nodes.

3.4.2 Multipath Routing

The use of labels is conducive to allowing multipath routing. Two approaches can be followed for multipath routing using FLR, and both require a node to store the label reported by each of its neighbors for a given destination. In FLR, node A should store the label for destination D reported by its neighbor B (denoted by L_{DB}^A) only if $L_{DB}^A < FL_D^A$.

In one approach, a node with multiple paths to a destination uses all of its available paths balancing the traffic load among them. The advantage of this approach is that it can be used to reduce packet-delivery delays (e.g., [17]). In another approach, a node uses only one path for a destination and uses other paths when the currently used path is lost. Because paths not being used may no longer be active in one or more of the relays along the path, a node that needs to use an alternate path must send a SRREQ in much the same way as when a node infers paths from cached information. The SRREQ is answered by the first relay with an active route to the destination.

3.4.3 Using Reverse Paths

A RREQ initiated by a node can be used by nodes receiving the request to update their routing tables with a route to the source of the RREQ in the reverse direction. However, because of the necessity for accurate predecessor information, the route cannot be used for data packet forwarding. A route entry created from the reverse route in a RREQ is valid for the reverse-route ttl, and a SRREQ is used to validate the path before data packets are forwarded. This allows the predecessor set to be built at nodes along the path when the RREPs to the SRREQs are received.

3.4.4 Keeping Labels Unique

We note that the basic operation of FLR allows a node A to accept a RREP from a neighbor B in which $A \in L_D^{rep}$. As an example of this case, consider node A with $L_D^A \ni \{X \oplus \dots \oplus D\}$ and assume that node X changes its label and feasible label to D to $FL_D^X = L_D^X > L_D^A$ after blocking its predecessors for D . If node X needs to find a new path to D and sends a RREQ, node A may be able to reply to X 's RREQ if L_D^A is smaller than the minimum feasible label of the RREQ created by X . The reply from A would provide X with a label $L_D^x \ni \{X \oplus \dots \oplus D\}$. Obviously, node X should not provide A with a label that already includes A , because that indicates to X that it has outdated path information.

Hence, in general, if node A has an active route to destination D and label L_D^A , it should not reply to any RREQ if $\exists j | 0 < j \leq |path^{req}| \wedge (e_j^{id}(path^{req}) \in L_D^A \vee src^{req} \in L_D^A)$, where $|path^{req}|$ is the number of elements in the label.

3.4.5 Using Destination-Sequenced Labels

We define a *destination-sequenced label* DL as the union of a label L and a sequence number assigned to L by the last element of the label, denoted by $SN(L)$. The relational operators for two labels defined in Eq 2 can be extended to destination-sequenced labels. Let $DL_1 = L_1 \cup SN(L_1)$ and $DL_2 = L_2 \cup SN(L_2)$, then

$$DL_1 = DL_2 \text{ if } \{ SN(L_1) = SN(L_2) \wedge L_1 = L_2 \} \quad (3)$$

$$DL_1 > DL_2 \text{ if } \begin{cases} SN(L_1) < SN(L_2) \vee \\ \{ SN(L_1) = SN(L_2) \wedge L_1 > L_2 \} \end{cases}$$

$$DL_1 < DL_2 \text{ if } \begin{cases} SN(L_1) > SN(L_2) \vee \\ \{ SN(L_1) = SN(L_2) \wedge L_1 < L_2 \} \end{cases}$$

With the relational operators of Eq. 3, the same sufficient conditions for loop-free routing using labels introduced in Section 2 can be shown to also hold for destination-sequenced labels. Hence, FLR can be based on destination-sequenced labels and destination-sequenced feasible labels.

To take advantage of sequence numbers as part of labels, SLC must be modified to allow the destination to increase its own sequence number each time it answers a RREQ:

SLC-DL: (Start Label Condition for Destination-Sequenced Labels). Node I can issue a RREP responding to a RREQ for destination D if I has an active route to D and $DL_I^I < MFDL_D^{req}$. If $I = D$, then node I increments $SN(L_D^I)$ before it sends its RREP (which specifies $DL_D^{rep} = L_D^I$) over the reverse path traversed by the RREQ.

RLC can be modified into an RLC with destination sequenced labels (RLC-DL), so that when node A has to originate a RREQ it increments the sequence number of the minimum destination sequenced feasible label ($MFDL$) sent in the RREQ. Doing so amounts to a generalization AODV's sequence numbering approach. The advantage of modifying RLC this way is that, as we state in the next section, no packet filtering based on predecessors is needed to prevent data packets from looping when RERRs are sent unreliably. The disadvantage is that fewer intermediate nodes are able to satisfy SLC-DL and generate a RREP.

A limitation with any optimization of FLR based on destination sequenced labels is that sequence numbers must be handled carefully when nodes reboot, a network partitions, and sequence numbers wrap around. However, there are many heuristics that solve such problems in practice [4].

3.5 Forwarding Data Packets

Thus far we have stated that RERRs are sent reliably to the predecessors of a node. However, achieving reliable transmissions of control packets intended for multiple destinations is not practical in MANETs that rely on contention-based MAC protocols (e.g., IEEE 802.11 DCF). Hence, the operation of FLR needs to be modified slightly to accommodate an unreliable link layer.

A simple way to accommodate an unreliable MAC protocol in FLR consists of allowing nodes to broadcast RERRs unreliably, and requiring nodes to determine if a packet that needs to be forwarded was received from a predecessor or not. Because a node that sends a RERR for a destination empties its predecessor set for the destination, it follows that no packets can traverse loops if nodes forward packets only

when they are received from valid predecessors. Accordingly, when node A receives a data packet for destination D from neighbor B , node A forwards the packet to its own next hop for destination D if the node has an active route for D and $B \in PS_D^A$. Otherwise, node A drops the packet and sends a RERR for destination D to node B .

Packet filtering is not needed if destination-sequenced labels are used and nodes that originate RREQs increase the sequence number for the destination in the MFDL carried in the RREQ. This is because nodes that use a given node A in their paths to a destination are unable to satisfy SLC-DL unless they have a destination-sequenced label with a sequence number that is larger than the one in the RREQ.

4. APPLICATION TO DSR AND AODV

FLR can be simplified by using node labels directly instead of feasible labels. With such simplification, a node simply maintains its current label, next hop and predecessor set for a destination; a RREQ carries the smallest label of any of the nodes that relayed the RREQ (ML_D^{req}) instead of $MF_L_D^{req}$; and ALC, SLC, MLC and RLC are implemented using ML_D^{req} instead of $MF_L_D^{req}$. This simplification is practical when links have unit cost only.

The above simplification can be used in two ways related to DSR. One approach consists of using the simplified signaling of FLR, which essentially adds ML_D^{req} in RREQs to DSR's signaling and requires using ALC, SLC, MLC and RLC, together with source-routed data packets. The other approach consists of using FLR's simplified signaling together with the packet filtering described in Section 3.5, so that loop-free packet forwarding can be enforced using only the destinations of the data packets on a hop-by-hop basis.

A "path-oriented" version of AODV can be attained using FLR with: the above simplification, destination-sequenced labels, and RLC-DL. This application of FLR is of interest if data-packet filtering based on predecessor information cannot be implemented in a MANET in which RERRs are delivered unreliably. Although the use of destination-sequenced labels increases the likelihood of intermediate nodes being able to send RREPs to RREQs compared to AODV, this approach does not perform better than the basic FLR scheme. This is because RLC-DL reduces the likelihood of intermediate nodes being able to answer RREQs compared to RLC.

The performance results presented in Section 7 for networks in which all link costs are equal are indicative of how a simplified FLR approach would perform. Important reasons why FLR performs so much better than DSR and AODV is that nodes apply SLC to reply to RREQs and ALC to accept RREPs in FLR and not in DSR and AODV. In DSR, this results in source routes leading to relays that must drop packets, and in AODV this translates into forcing the destinations to reply to RREQs more often than in FLR.

5. FLR EXAMPLE

Figure 1(a) shows the directed acyclic successor graph for destination E in a six-node network. The feasible labels for destination E at each node are shown for time t_1 next to each node. Nodes A , B , D and F have active flows for node E . The label for destination E at the nodes is not shown separately, because the feasible labels are equal to the assigned labels.

Node C does not have an active route to node E and

therefore has an invalid label L^∞ . The labels marked in the figure are composed of tuples of $(id, cost)$. The predecessor sets maintained at nodes F and B are $PS_E^F = \{B\}$ and $PS_E^B = \{A\}$, respectively.

5.1 Update Activity

Link e_2 fails at time $t > t_1$, and node B broadcasts a RERR. This RERR may not be received by A due to the unreliability of the MAC layer, but node B removes A from PS_E^B . Node B now initiates a new computation (B, ID_B) searching for a route for E using a feasible label $\{(B, 1), (E, 0)\}$.

When node A receives B 's RREQ, it cannot satisfy SLC and must relay the RREQ. The RREQ relayed by A is dropped by B , because it cannot engage itself again in computation (B, ID_B) . B 's RREQ is relayed by node C , because node C does not have an active route and therefore SLC cannot be satisfied.

Node D generates a RREP when it receives the RREQ relayed by C , because it satisfies SLC. The RREP from D carries the label $L_E^{r_{DD}} = \{(D, 1), (E, 0)\}$. Node C can accept this RREP, and updates its routing table, setting $L_E^C = FL_E^C = \{(C, 5), (D, 1), (E, 0)\}$. Node C adds node B to its predecessor set PS_D^C for D and forwards the RREP carrying the label $\{(C, 5), (D, 1), (E, 0)\}$.

When node B receives the RREP from C , it updates its routing table and makes $s_E^B = C$. Given that B already sent a RERR for destination E to its predecessors, it can reset its feasible label. Accordingly, node B sets L_E^B and FL_E^B to $\{(B, 1), (C, 5), (D, 1), (E, 0)\}$. The feasible labels for destination E at time t_2 are shown in Fig.1(b).

If on the other hand, B was performing a local repair, then it would have sent a RERR to A only after processing the RREP from C . The RERR would not be sent if C had offered a label satisfying ALC with local repair, in which case A would not have received a RERR from B .

5.2 Loop Detection

To illustrate loop detection in FLR, let us assume that the RERR sent by B was not received by node A . Node A still considers node B as its successor for E according to its routing table, and the ordering of the feasible labels along the path to E has been violated due to the undelivered route error.

Assume that node C becomes a source for destination E time $t_2 + \epsilon$. Because node C has an active route to E , a new RREQ is not started, and if forwards data packets to its next hop D . Now, at time $t > t_2 + \epsilon$, link e_1 fails. Node C detects the link failure and notifies B through a broadcast RERR that is not received by B . Node B maintains C as its next hop to destination E , and node C has removed node B from PS_E^C .

Now node C starts a new route computation (C, ID_C) and issues a new RREQ with feasible label $\{(C, 5), (D, 1), (E, 0)\}$. Node A satisfies SLC with a lower cost label $\{(A, 1), (B, 1), (F, 1), (E, 0)\}$ and issues a RREP with its current label for E . Node C accepts A 's RREP because it satisfies ALC and makes A its successor for destination E , creating a routing-table loop among nodes A, B and C .

Fig. 1(c) shows the directed successor graph for destination E at time t_3 after node C updates its routing table. However, this loop cannot persist, because A is no longer in the predecessor list of B , which forces B to drop any data

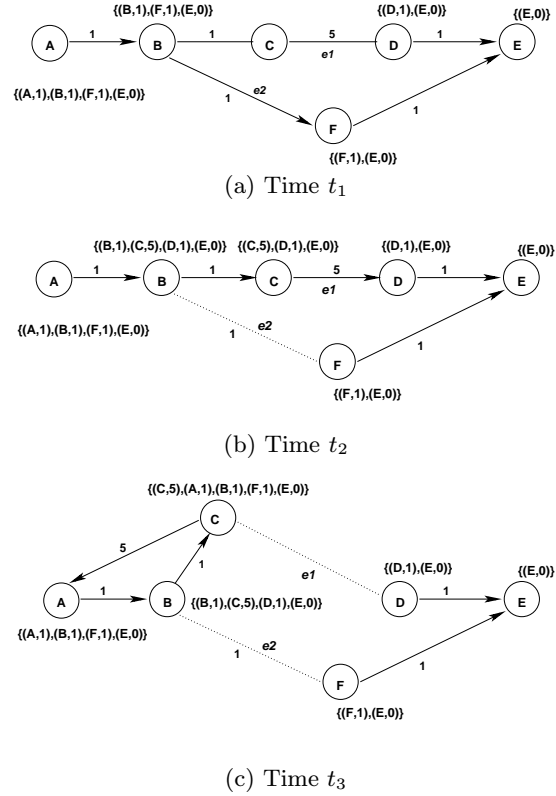


Figure 1: Illustration of FLR.

packet it receives from A and to send a RERR to A in each case. If A never sends a data packet, then the routing-table loop is broken when node A 's route entry expires. The same argument applies between nodes B and C .

6. ANALYSIS OF FLR

We prove that FLR as described in Sections 3.1 to 3.3 is loop-free at every instant if reliable RERRs can be delivered. We additionally show that, if RERRs must be sent unreliably, temporary routing-table loops caused by undelivered RERRs are broken within a finite time and data packets are never forwarded in a loop. We then demonstrate that a source requesting a route to a given destination must be able to establish a path if the network is connected and stable for a sufficiently long period of time after an arbitrary sequence of topology changes.

6.1 Loop-Freedom in FLR

THEOREM 2. *In FLR, RREQs and RREPs do not loop.*

PROOF. For a given route computation (A, ID_A) , a node may be passive, engaged, or active. A node can become active in a route computation at most once, because it maintains the identifiers it assigns to the RREQs it originates. A router can engage in a route computation only when the corresponding RREQ does not include the node in the path traversed by the RREQ. Hence, any RREQ can traverse only a directed acyclic graph (DAG), which may be a directed tree if no node relays the RREQ more than once, and any path traversed by a RREQ is free of loops.

Because RREPs are forwarded along the reverse path traversed by the corresponding RREQs, it follows that the RREPs must traverse loop-free paths. Furthermore, if the source route request (SRREQ) optimization is used, the SRREQ travels the loop-free path specified in the SRREQ by its source. \square

THEOREM 3. *FLR ensures that, if RERRs are sent reliably and path $P = \{n_k, \dots, n_1\}$ exists at some point in time as defined by the successor entries of the nodes along the path, it is true that $FL_{n_1}^{n_i} > FL_{n_1}^{n_{i-1}}$, for $i \in [2, k]$.*

PROOF. For path P to exist at a given time t , it must be true that all nodes in P have a valid successor. According to ALC, node n_i can make n_{i-1} its successor for destination n_1 if it received a RREP from n_{i-1} and either $L_{n_1}^{rep} < FL_{n_1}^{n_i}$ or n_i sent a RERR reliably to every node in its predecessor set, which forces all such nodes to stop using n_i as their successor for n_1 . Hence, if P exists at time t , every node n_i ($i \in [2, k]$) must have accepted a RREP from n_{i-1} with $L_{n_1}^{rep} < FL_{n_1}^{n_i}$. Because $FL_{n_1}^{n_{i-1}} \leq L_{n_1}^{n_{i-1}} = L_{n_1}^{rep}$, the theorem is true. \square

THEOREM 4. *FLR is loop-free at every instant if RERRs are delivered reliably.*

PROOF. Let node I initiate a RREP for destination D and let the RREP traverse the path $P = \{n_1, \dots, n_j\}$, where $n_1 = I$ (maybe the destination D) and $n_j = A$. Let the path from I to D be $Q = \{m_1, \dots, m_k\}$, where $m_1 = D$ and $m_k = I$ and Q can be null if $n_1 = D$.

For a loop to form, node A must be on the path Q and must change successors after processing the RREP generated by I . From Theorem 2, path P must be loop free. Thereby, if $n_1 = D$, node A cannot form a loop after processing I 's RREP. If $n_1 \neq D$ we must show that it is impossible for A to be on I 's successor graph.

Assume that, at time t_0 , path Q exists and is loop free and at time t_1 node A changes successors after processing I 's RREP. Node I sends its RREP along the loop-free path P to A , carrying $L_D^{rep} = L_D^I(t_0)$. Let node A be some node m_i , $1 < i < k$ and let m_{i+1} be its predecessor. From Theorem 3 we have that $FL_D^I(t_0) > FL_D^A(t_0)$. Node A 's routing-table entry for destination D can be in one of the following states.

Case (i): Node A has an invalidated route, which means A reliably notified its predecessor m_{i+1} , and the path Q no longer exists. Hence, node A cannot form a loop when it processes the RREP from I .

Case (ii): Node A has an active route. We know that $L_D^{rep} \geq L_D^I(t_0) \geq FL_D^I(t_0) > FL_D^A(t_0)$. At any time $t_0 \leq t \leq t_1$, node A 's feasible label FL_D^A could have only decreased if A lies on path Q within this time interval. Hence, $FL_D^A(t_1) \leq FL_D^A(t_0)$. Now $L_D^{rep} \geq FL_D^A(t_1)$, so node A will not process the RREP, because it does not satisfy ALC. If A increased its feasible label when it chose a new successor, then it must have reliably notified m_{i+1} with a RERR, which reduces to case (i). Therefore, no loop cannot form in this case.

Case (iii): Node A has an invalidated route and is performing a local repair operation. As discussed for Case (ii), ALC is not satisfied at A when I 's RREP is received. Although node A processes the RREP, it must send a RERR reliably to its predecessor set because $FL_D^A(t_1) \leq L_D^{rep}$. Hence, the path Q is no longer valid, because node m_{i+1} stops using node n_i as successor. Therefore, no loops can form in this case. \square

We now prove that, if an undelivered RERR causes a loop after nodes reset their feasible label according to RLC or ALC, then data packets are never forwarded along the loop and the routing-table loop is broken within finite time.

THEOREM 5. *FLR ensures that data packets never flow in loops and routing-table loops can exist only temporarily.*

PROOF. A routing-table loop can form only when the ordering criteria of Theorem 3 is violated. This means that along a path $P = \{n_k, \dots, n_1\}$ for destination n_1 , $\exists i, i \in [2, k-1]$, $FL_{n_1}^{n_{i+1}} < FL_{n_1}^{n_i}$. Node n_i or node n_{i+1} can never update its routing table to create this condition, unless node n_i sent a RERR as per ALC or RLC before increasing its feasible label $FL_{n_1}^{n_i}$.

For the ordering violation to occur, n_{i+1} must not have received the RERR. Before increasing the feasible label at n_i , it removes n_{i+1} from the predecessor list. With the ordering criteria violated, a loop can be formed at a later time if node n_i 's route request is replied to by a node upstream of n_{i+1} or node n_{i+1} itself. Assume such a loop is formed. If n_i receives a data packet from n_{i+1} , then node n_i must drop the data packet, and send a RERR for destination n_1 to n_{i+1} , because it received a data packet from a node that is not in the predecessor set for n_1 . If node n_{i+1} receives the RERR, it invalidates its route to n_1 and the loop is terminated. On the other hand, if n_{i+1} never sends a data packet, then the route entry expires and it no longer uses n_i as its next hop. Therefore, the routing-table loop terminates as both events are bounded by a finite-time duration, and data packets are never forwarded in loops even if the underlying routing table has loops. \square

6.2 Correct Termination in FLR

We now prove that any source is able to establish a route to a destination within finite time if there is a physical path between the source and the destination, and the network is stable and error-free after an arbitrary sequence of topology changes. We assume that reliable RERRs can be delivered in the network.

LEMMA 1. *If an intermediate node I initiates a RREP for destination D using SLC in response to a RREQ from A that traversed a path P , then the concatenation of path P and the successor path from I to D is loop-free.*

PROOF. Let the RREQ initiated by A traverse the path $P = \{n_1, \dots, n_{k-1}\}$ before it is received by I . From Theorem 2, it follows that path P is loop-free. Assume that node I has a path $Q = \{m_1, \dots, m_{k-1}\}$ to destination D , which is its current successor path for the active route. We have to show that any node $n \in P$ is not the same as any $m \in Q$ for the lemma to be true. If Q is empty, then the lemma trivially true. We consider the case where Q is not empty.

The proof is by contradiction. Due to the relaying procedure of RREQs, when the RREQ is received by node I , it carries the minimum of the feasible labels (MFL_D^{req}) of the nodes along the path P . Since I has an active route to D , by Theorem 3, we have along path Q from I to D , $FL_D^I > FL_D^{m_1} > FL_D^{m_2} > \dots > FL_D^{m_{k-1}} > FL_D^D$. At any instant, we have $L_D^I \geq FL_D^I$. For the RREQ to satisfy SLC at node I , we must have $L_D^I < MFL_D^{req}$. However, if any node $m \in Q$ is the same as any node $n \in P$, we will have $L_D^I > MFL_D^{req}$ and node I cannot initiate the RREP satisfying SLC. Therefore, the concatenation of path P and

successor path of I must represent a loop-free path to the destination. \square

THEOREM 6. *In a error-free stable connected network, FLR ensures that a node A starting a route computation (A, ID_A) for a destination D establishes a successor path to the destination within a finite time.*

PROOF. We consider the first RREP rp_A to reach A for the route computation (A, ID_A) . If multiple RREPs are received, then node A can switch to better routes (i.e., routes with smaller labels).

Let node A start a new route computation (A, ID_A) for destination D , by sending a RREQ, and let that RREQ traverse the path $P = \{n_1..n_{k-1}\}$ before arriving at node n_k (can be equal to D) which satisfies SLC. From lemma 1, if nodes switch along path P concatenated with the successor path from n_{k-1} , they have a loop-free valid successor path to the destination.

Let node n_k generate a RREP to the RREQ from source A . The proof must show that when nodes relay the reply along the reverse path, nodes which are engaged in the computation (A, ID_A) after processing the reply have a valid successor path to the destination.

We first prove that A establishes the successor path using the RREP relayed along path P when no node along the path P is affected by another route discovery event for D during the route computation (A, ID_A) . In this case, no node along P satisfies SLC with A 's feasible label, for otherwise that node would have responded to the RREQ instead of n_k . Each node $n \in P$ must be in one of three states: (i) n has no information about D , (ii) n 's information is invalid, or (iii) n has an active route but SLC could not be satisfied. In Case (i), node n may use any RREP sent by n_k . In Case (ii), node n has an invalid route, which means that n has notified any predecessors reliably through a RERR and node n can process the RREP and update its route entry for D . In Case (iii), node n can process the RREP and switch successors if the reply offers a smaller label; if node n does not switch successors, then it has a better route to D . In all three cases, node n will then forward a new route reply with its current label and forward it along the reverse path carried in the RREP. Node A will receive the RREP, because nodes in all states forward a RREP along the reverse path of P .

We now show that simultaneous route discovery events for destination D do not affect the route computation (A, ID_A) and nodes along path P relay a RREP to A . The three following cases represent the events that can interfere with the route computation of (A, ID_A) .

Case 1: During the computation period of (A, ID_A) , one or more nodes $n_i \in P$ are engaged for route computations (m_i, ID_{m_i}) , for destination D , where $m_i \notin P$. Denote by rp_A the RREP initiated by route computation (A, ID_A) and by rp_m the set of RREPs for (m_i, ID_{m_i}) . From the previous discussion, node n_i on processing rp_A or any rp_m , may update its routing table and forward a new RREP along the reverse path. If node n_i receives rp_A before any rp_m , the route computation (A, ID_A) has completed at node n_i . On the other hand, if rp_A was received after a $rp \in rp_m$, then this reduces to case (iii) above, when there are no other simultaneous route discovery events and node n_i may improve its route after processing rp_A . In either sequence of events, node n_i forwards a new RREP with its current label

along the reverse path to A , which ensures that the route computation (A, ID_A) is not affected by actions at n_i .

Case 2: Nodes $m_i \in P$ become active during the route computation period of (A, ID_A) . A node m_i becoming active for destination D is equivalent to the node being engaged in a new route computation for D . This reduces to Case 1 and the route computation for (m_i, ID_{m_i}) cannot interfere with that of (A, ID_A) .

Case 3: One or more nodes $m_i \in P$ are engaged and one or more nodes $m_j \in P$ are active for route computations to destination D . From Case 2, nodes that are active for a route computation to destination D reduce to the case of being engaged in the route computation. Hence, we can consider $m_j \in P$ to be engaged. So from Case 1, the route computations cannot interfere with that of (A, ID_A) .

Therefore, the nodes along path P engaged for the route computation (A, ID_A) are not affected by simultaneous route discovery events for destination D that can occur. Node A receives the RREP forwarded along the reverse path of P and establishes a successor path to the destination within a finite time given that each message is exchanged within a finite time. \square

7. PERFORMANCE COMPARISON

We present results for FLR over varying loads and mobility. The protocols used for our comparison with FLR are DSR, AODV, and OLSR, which are representative of the state of the art in routing protocols for MANETs. Simulations are run in Qualnet[18]. The parameters are set as in [12].

Simulations were performed on two scenarios, a 50-node network with terrain dimensions of 1500m x 300m, and a 100-node network with terrain dimensions of 2200m x 600m. Traffic loads were CBR sources with a data packet size of 512 bytes. Load was varied by using 10 flows (at 10 packets per second) and 30 flows (at 4 packets per second). The MAC layer used was IEEE 802.11 with a transmission range of 275m and throughput 2 Mbps. The simulated time is 900 seconds. Node velocity was set between 1 m/s and 20 m/s. Flows have an exponentially distributed length with a mean of 60 seconds. Each combination (number of nodes, traffic flows, scenario, routing protocol and pause time) was repeated for nine (9) trials using different random seeds.

We address four performance metrics. *Delivery ratio* is the ratio of the packets delivered per client/server CBR flow. *Latency* is the end to end delay measured for the data packets reaching the server from the client. The *network load* is the total number of control packets divided by the number of received data packets. *Data hops* is the number of hops traversed by each data packet (including initiating and forwarding) divided by the total number of received packets in the network. This metric takes into account packets dropped due to forwarding along incorrect paths, and provides a measure of the quality of the routes.

Tables 2 and 3 summarize the results of the different metrics by averaging over all pause times for the 50-node and 100-node networks. The columns show the mean value and 95% confidence interval. Two versions of FLR were simulated. One version is FLR without any of the optimizations discussed in Section 3.4 (FLR) and the other version (FLR-Opt) incorporates only the optimizations discussed in Section 3.4.1 and Section 3.4.3. As the tables show, the average performance of the FLR-Opt and FLR across the

Table 2: Performance average over all pause times for 50 nodes network

Protocol	Flows	Delivery Ratio	Latency (sec)	Net Load	Data Hops
FLR	10	0.9701±0.0108	0.0858±0.0303	0.1872±0.0557	2.5371±0.1833
FLR (Opt)	10	0.9705±0.0108	0.0886±0.0323	0.1888±0.0549	2.5228±0.1822
AODV	10	0.9545±0.0193	0.1151±0.0497	0.5980±0.2286	2.5531±0.1994
DSR	10	0.7929±0.0654	1.4612±0.5874	0.5373±0.1982	1.7331±0.2140
OLSR	10	0.8543±0.0396	0.3479±0.1390	0.8048±0.0883	2.3968±0.1740
FLR	30	0.9078±0.0284	0.4523±0.1718	0.7012±0.2190	2.5604±0.2086
FLR (Opt)	30	0.9108±0.0276	0.4494±0.1820	0.6835±0.2062	2.5539±0.2067
AODV	30	0.7647±0.0558	1.9168±0.6460	3.8584±1.1390	2.9681±0.3422
DSR	30	0.7378±0.0654	5.6598±1.5812	0.6013±0.1858	1.6920±0.2149
OLSR	30	0.8204±0.0381	1.3499±0.4458	0.6958±0.0744	2.3850±0.1720

Table 3: Performance average over all pause times for 100 nodes network

Protocol	Flows	Delivery Ratio	Latency (sec)	Net Load	Data Hops
FLR	10	0.8843±0.0282	0.2986±0.0942	1.0687±0.2895	3.9787±0.4724
FLR (Opt)	10	0.8863±0.0276	0.2996±0.0999	1.0639±0.2881	3.9506±0.4737
AODV	10	0.8359±0.0377	0.3536±0.0994	3.4476±0.9873	4.1541±0.5312
DSR	10	0.6774±0.0714	1.9072±0.4109	2.3397±0.8379	3.6690±0.5417
OLSR	10	0.6976±0.0547	2.1138±0.7194	5.7178±0.7198	3.9137±0.4811
FLR	30	0.7821±0.0363	0.9192±0.2126	2.9053±0.6039	4.0680±0.4520
FLR (Opt)	30	0.7863±0.0321	0.9044±0.1837	2.8382±0.5246	4.0883±0.4251
AODV	30	0.5619±0.0997	3.5194±1.3503	33.6577±35.7988	5.9871±0.8752
DSR	30	0.6266±0.0666	6.2102±1.3701	2.5235±0.7824	3.6898±0.4264
OLSR	30	0.6269±0.0461	6.4852±1.6194	5.3037±0.6408	4.0117±0.4518

difference metrics are statistically the same. In the rest of the section, therefore, we discuss the results with respect to the unoptimized version of FLR.

Figs. 6, 5, 8, and 9 show the delivery ratio for different pause times in the 50-node and 100-node scenario. Confidence intervals(95%) are shown with vertical bars in the graphs. FLR has a very consistent performance across all scenarios and outperforms other protocols in most cases. The exception occurs in scenarios with many flows and low mobility, where OLSR seems to do better.

The control overhead over different pause times for 10 flows and 30 flows in the 100-node scenario is shown in Fig. 2 and Fig.3, respectively. The end-to-end delay over different pause times for 10-flows and 30-flows in the 100-node scenario is shown in Fig. 4 and Fig.7, respectively.

The reason for AODV’s poor performance in the 100-node scenarios can be explained. RREQs in AODV are mostly answered by the destination, which is bad news in the 100-node network because the network diameter is large and RREQs are answered after traversing long paths to the destinations. The RREQs congest the MAC layer and cause packet collisions. In turn, packet losses necessitate more RREQs, resulting in more control overhead. The large confidence intervals of the control overhead in AODV indicate that some cases produce many more route search floods than others. On the other hand, RREQs in FLR are answered by intermediate nodes with high likelihood, resulting in less control overhead and variance, because RREQs do not flood the entire network.

In the scenarios with 10 flows and 50 nodes, FLR has a data delivery (0.9701 ± 0.0108) that is statistically equivalent to that of AODV (although the confidence intervals barely overlap). FLR has the lowest control overhead at 0.1872 ± 0.0557 and data delivery latency of 0.0858 ± 0.0303 secs. DSR only achieves a packet delivery of 0.7929 ± 0.0654.

In scenarios with 50 nodes and 30 flows, FLR has the highest data delivery (0.9078 ± 0.0284) followed by OLSR

at 0.8204 ± 0.0381, whereas AODV manages only 0.7647 ± 0.0558, followed by DSR with 0.7378 ± 0.0654. FLR has the lowest latency (0.4523 ± 0.1718 secs). The control overhead of FLR is 0.7012 ± 0.219, compared to AODV’s 3.8584 ± 1.1390. OLSR has a low control overhead, which we can expect due to network partitions that force the on-demand protocols to flood regularly, therefore causing additional overhead. DSR’s low control overhead is due to stale cached routes along which data packets are attempted to be routed. Route errors are received for the invalid source routes, and this long delay forces data packets to be dropped and the necessity for route searches is avoided. Routes are established after a long delay, which correlates with DSR having the worst latency (5.6598 ± 1.5812) and data delivery (0.7378 ± 0.0654) among all protocols.

In the 100-node configuration with 10 flows at 100 pps, FLR has the highest data delivery (0.8843 ± 0.0282), while AODV delivers 0.8359 ± 0.0377 and DSR manages a delivery of 0.6774 ± 0.0714. FLR exhibits the lowest control overhead (1.0687 ± 0.2895), whereas the latency of AODV and FLR are statistically equivalent. AODV shows huge performance problems in the 30-flow (120pps) scenario and has the lowest packet delivery (0.5619 ± 0.0997). In the high-mobility scenarios, we were not able to obtain tight bounds on performance and overall the net load of AODV.

In the 100-node, 30-flow scenario, FLR has the lowest control overhead (2.9053 ± 0.6039), latency (0.9192 ± 0.2126) and the highest delivery ratio (.7821 ± 0.0363).

The *data hops* metric provides a measure of the accuracy of the routes used for forwarding. FLR, AODV and OLSR have statistically equivalent data hops across all scenarios, except in the scenario with 30 flows and 100 nodes where AODV incurs max data hops, while DSR exhibits the lowest data hops throughout. The data hops metric reflects the number of hops traversed by each data packet whether or not it is delivered. In AODV, packets are dropped at intermediate nodes due to route failures, whereas in OLSR data

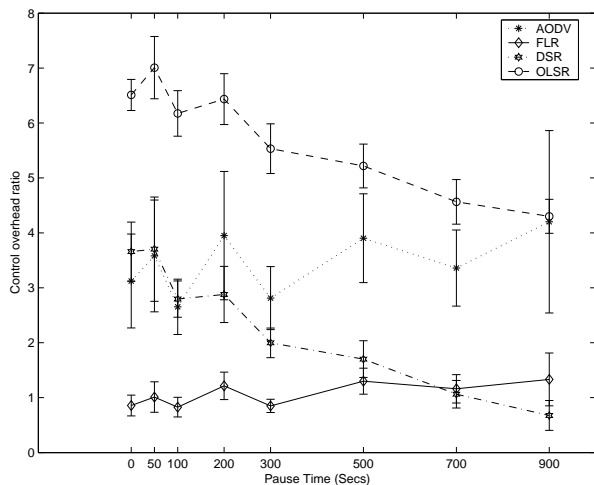


Figure 2: Control overhead ratio (100 nodes, 10 flows, 100 pps)

packets can temporarily traverse loops before being delivered or can get dropped due to lack of routes. The low data hops measure of DSR is due to stale source routes, which causes packets with invalid source routes to be dropped at the intermediate nodes. FLR's data hops in correlation with the packet delivery ratio shows the high accuracy of the active routes.

8. CONCLUSIONS

We extended sufficient conditions for loop-free routing previously stated for distances to labels that are ordered lexicographically. We introduced the feasible label routing (FLR) protocol as an illustration of how loop-free routing can be attained using path information on demand while allowing data-packet forwarding on the basis of the packet destinations only.

In FLR, nodes are ordered according to their labels to a destination to provide instantaneous loop-freedom. A node resets its label (path) for a destination when its route fails. Resetting labels requires a reliable route error message to predecessors. To cope with the case in which the MAC layer of a MANET does not support reliable transmissions efficiently, we proposed using a local data-packet filtering action to detect and break routing-table loops caused when RERRs are sent unreliably. Simulation results show that FLR significantly outperforms DSR, AODV, and OLSR.

9. REFERENCES

- [1] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," Request for Comments 3626, October 2003.
- [2] 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.
- [3] J. J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations," *Proc. IEEE/ACM Trans. Networking*, 1(1):130-41, Feb. 1993.
- [4] J. J. Garcia-Luna-Aceves, M. Mosko and C. Perkins, "A New Approach to On-Demand Loop-Free Routing in Ad Hoc Networks," *Proc. PODC 2003*, pp. 53-62, Boston, Massachusetts, July 13-16, 2003.

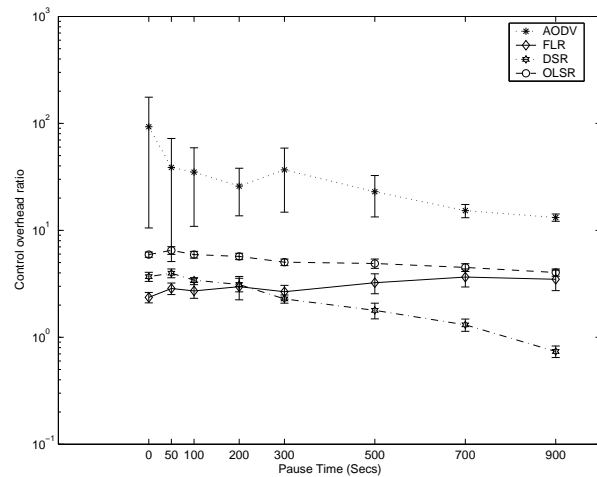


Figure 3: Control overhead ratio (100 nodes, 30-flow, 120pps)

- [5] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," *Proc. IEEE ICNP'99*, pp. 273-82, Toronto, Canada, October 31-November 3, 1999.
- [6] Y. C. Hu and D. B. Johnson. Implicit Source Routing in On-Demand Ad Hoc Network Routing. In *ACM MOBIHOC 2001*, pp. 1-10, Long Beach, CA, Oct. 4-5, 2001.
- [7] D. Johnson et al, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF Internet draft, draft-ietf-manet-dsr-09.txt, April 2003.
- [8] R. Ogier et al., "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)," Request for Comments 3684, February 2004.
- [9] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOM*, pp. 1405-13 vol.3, Apr. 1997.
- [10] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM 94*, 1994.
- [11] C. Perkins et al., "Ad hoc On-Demand Distance Vector (AODV) Routing," Request for Comments 3561, July 2003.
- [12] C. Perkins et al. "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, 8(1):16 - 28, Feb 2001.
- [13] J. Raju and J. J. Garcia-Luna-Aceves, "A New Approach to On-Demand Loop-Free Multipath Routing," *Proc. IEEE IC3N'99*, pp. 522-7, Boston, Massachusetts, Oct. 11-13, 1999.
- [14] S. Roy and J. J. Garcia-Luna-Aceves, "Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks," *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, April 22-26, 2001.
- [15] C. Sengul, "Local Route Recovery in Mobile Ad Hoc Networks," M.S. Thesis, Computer Science, Univ. of Illinois at Urbana-Champaign, 2003.
- [16] M. Spohn and J. J. Garcia-Luna-Aceves, "Neighborhood Aware Source Routing," *Proc. ACM MobiHoc 2001*, pp. 11-21, Long Beach, CA, Oct. 4-5, 2001.
- [17] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing," *Proc. ACM SIGCOMM '99*, Cambridge, Massachusetts, September 1-3, 1999.
- [18] Scalable Network Technologies. Qualnet 3.5.2.

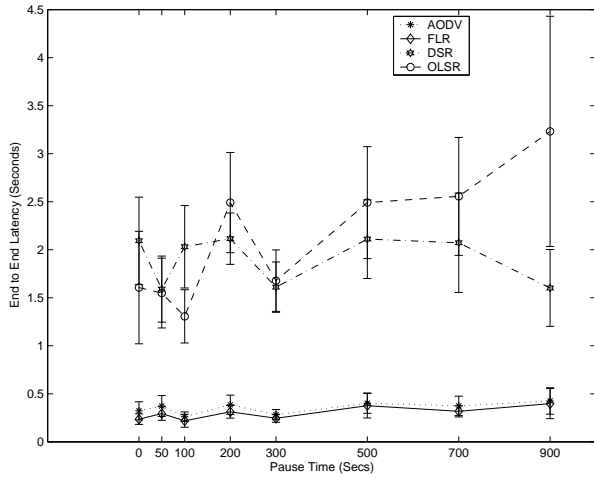


Figure 4: End-to-end latency (100 nodes, 10 flows, 100 pps)

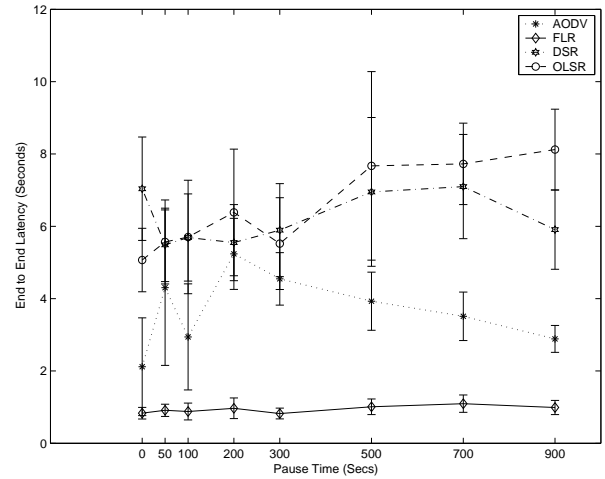


Figure 7: End-to-end latency (100 nodes, 30-flow, 120pps)

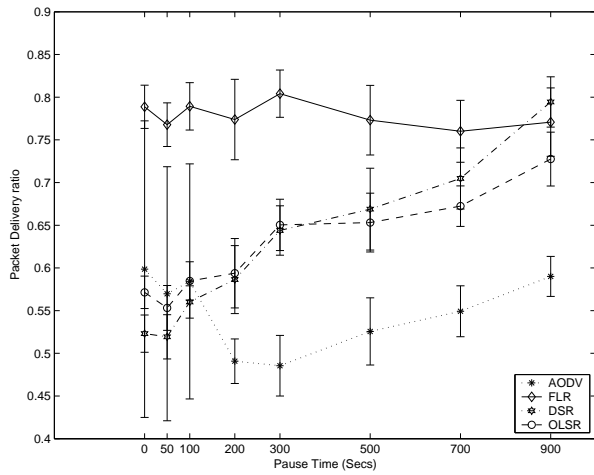


Figure 5: Delivery (100 nodes, 30 flows, 120 pps)

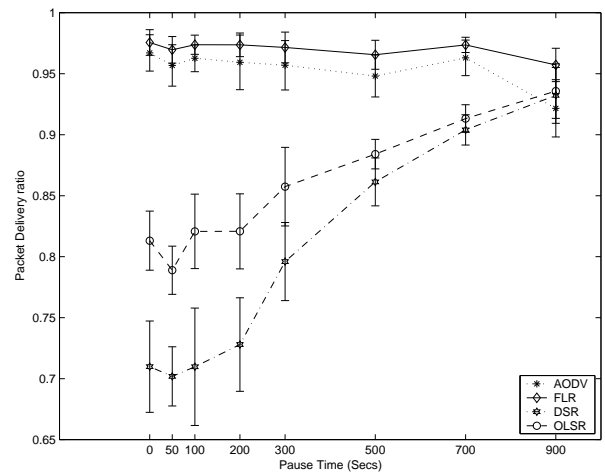


Figure 8: Delivery (50 nodes, 10 flows, 100 pps)

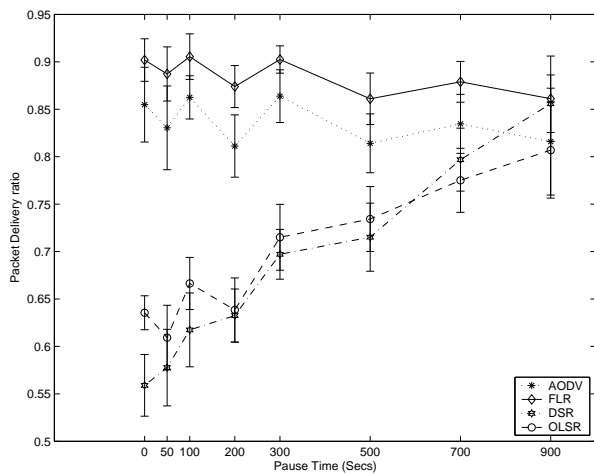


Figure 6: Delivery (100 nodes, 10 flows, 100 pps)

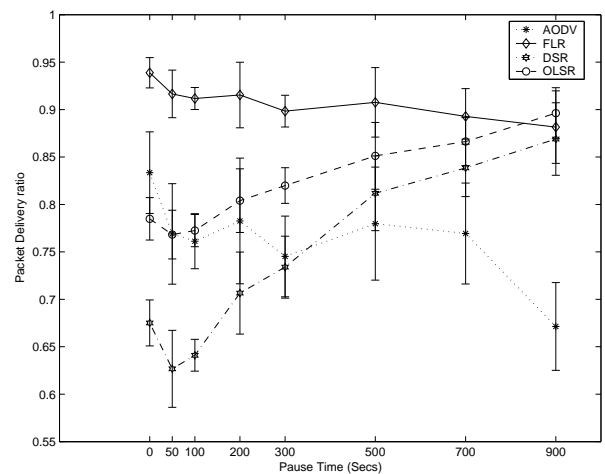


Figure 9: Delivery (50 nodes, 30 flows, 120pps)