Title

A New Approach to On-Demand Loop-Free Routing in Networks Using Sequence Numbers

Author

Garcia-Luna-Aceves, J.J.

Publication Date

2006-07-01

Peer reviewed

A New Approach to On-Demand Loop-Free Routing in Networks Using Sequence Numbers

J.J. Garcia-Luna-Aceves [b] Marc Mosko [b,1] Charles E. Perkins [a]

[a] *Communication Systems Laboratory, Nokia Research Center, 313 Fairchild Drive, Mountain View, CA 94043*

[b] *U.C. Santa Cruz, Jack Baskin School of Engineering, Santa Cruz, CA, 95062*

Abstract

A new protocol is presented for on-demand loop-free routing in ad hoc networks. The new protocol, called labeled distance routing (LDR) protocol, uses a distance invariant to establish an ordering criterion and per-destination sequence numbers to reset the invariant resulting in loop-freedom at every instant. The distance invariant allows nodes to change their next hops or distances to destinations without creating routing-table loops. The destination sequence number, which only the destination may increment, permits nodes to reset the values of their distance invariants. The performance of LDR is compared against the performance of three other protocols that are representative of the state-of-the-art, namely AODV, DSR and OLSR; LDR's performance is shown to be far better than the other three protocols.

Key words: Routing protocol, loop-free, algorithm analysis, ad hoc network

1 Introduction

An ad hoc wireless network is characterized by nodes functioning as routers, as well as sources and sinks of data traffic, with radio network interfaces and no fixed infrastructure to support communications. Wireless networks are usually

Email addresses: jj@soe.ucsc.edu (J.J. Garcia-Luna-Aceves), mmosko@soe.ucsc.edu (Marc Mosko), charliep@iprg.nokia.com (Charles E. Perkins).

[1] 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 in Computer Engineering.

limited bandwidth and battery power, so their routing protocols should have low control overhead. Reactive or on-demand routing protocols have been developed for this reason. In an on-demand routing protocol, a node only maintains routes for in-use destinations and does not pro-actively advertise routes. Rather, it queries for needed routes and offers routes in response to queries.

In our discussion of routing protocols, we will at times use the terms *predecessor* and *successor*. At some node i in the network graph, the successors of i for a destination D are those nodes along i's path to D. The immediate next-hop of i to destination D is referred to as "the successor" to D. The predecessors of i for destination D are those nodes that have i as one of their successors to D.

Many on-demand routing protocols have been proposed over the past few years, and all of them attempt to provide loop-free paths at every instant through various methods. Examples of such protocols include the Ad hoc On-demand Distance Vector (AODV) protocol [1], the Dynamic Source Routing (DSR) protocol [2,3], the Neighborhood-aware Source Routing (NSR) protocol [4], the Temporally-Ordered Routing Algorithm (TORA) protocol [5], and the Routing On-demand Acyclic Multipath (ROAM) protocol [6]. There are also many examples of pro-active routing protocols that attain loop-free routing [7] as well as pro-active routing protocols that tolerate temporary routing loops [8–10].

Two factors that decrease routing efficiency in ad hoc networks are routing loops and the maintenance of complete reachability information for all nodes. Routing loops increase packet-delivery latencies and reduce the number of packets delivered to the intended destinations. Furthermore, maintaining routing information for all destinations becomes less efficient than maintaining routing information on-demand at each node as the number of nodes in the network increases and the average number of destinations contacted by each source becomes a smaller and smaller portion of the total number of nodes. Accordingly, this paper addresses the problem of providing efficient on-demand loop-free routing in ad hoc networks. Prior work on on-demand loop-free routing has been based on the following approaches: (a) using source routes in data packets, (b) coordinating nodes by way of the directed graph implied by the next hop entry for a given destination at each node, and (c) using sequence numbers to establish an ordering among nodes.

DSR and NSR are examples of using source routing to avoid routing loops. In DSR, each route request records its traversed path, and the complete route between source and the requested destination is specified in the route reply sent back to the source by any node with a valid route the destination. The discovered route is stored in a route cache at the source and the relay nodes.

The header of every data packet specifies the source routes to their intended destinations. DSR incorporates a number of optimizations to shorten source routes, learn routes, and cache routes efficiently. NSR extends the source routing approach of DSR by having nodes communicate information regarding their two-hop neighborhood in route requests and route replies in addition to path information regarding specific in-use destinations.

TORA uses a link-reversal algorithm [11] to maintain loop-free multipaths that are created by a query-reply process similar to that used in DSR and AODV. TORA relies on synchronized clocks to create time stamps that maintain the relative ordering of events. The link-reversal algorithm is a form of synchronization among nodes spanning multiple hops.

The Diffusing Update Algorithm (DUAL) [7] attains loop freedom by means of nodal coordination. Based on routing advertisements, a node i may choose independently a successor j from its neighbors for a destination d as long as that choice cannot form a loop; such a choice is called feasible if it cannot form a loop. If the potential exists to form a loop, node i must coordinate with other nodes before changing its route to d. The feasibility of a change is calculated based on a feasibility condition. If the condition is satisfied, the change is feasible, and node i may proceed without coordinating with any other node. If the change is not feasible, node i starts a diffusing computation [12]. DUAL was developed for wire-line networks and the diffusing computation requires reliable communications between neighbors to enforce synchronization over potentially large segments of a network. The diffusing computation resets information in the network, ensuring that any change at i for d will not cause a loop. After the computation terminates, node i is free to make a change and then inform the network of that change.

ROAM extends DUAL to provide loop-free routing on demand. The distances to a given destination are used to establish ordering among nodes. ROAM uses a route request-reply process similar to that used in AODV, DSR and other on-demand protocols. However, a node can change its next hop to a destination without notifying its neighbors as long as it has a neighbor with a distance that is shorter than the node's own feasible distance value to the destination, where such a distance is the smallest value attained by the node's distance since it obtained a route to the destination after sending a route request. If such an invariant condition is not satisfied, the node must reliably send a route request to its neighbors, which serves the same purpose of DUAL's resets. After sending a route request, the node cannot select a new next hop to a destination until it receives route replies from all its neighbors.

ROAM and TORA require reliable exchanges among neighbors and coordination among nodes over multiple hops. Routes are locked down at a node until the portion of the distributed calculation in which it participates is com-

plete, which is signaled to the node when all its neighbors reply to its route request. This type of mechanisms incurs more control messages compared to AODV, DSR, and other on-demand protocols that work correctly even with unreliable transmissions of route requests and replies among neighbors.

AODV attains loop-free routing by using a sequence number for each destination as the means to establish an ordering invariant among nodes. AODV defines an active route as one that is fresh and likely to have a good successor path. Such a route was either recently learned through some advertisement or has been recently used without error. An inactive (or invalid) route is one that has expired its cache time without use or one for which the next hop is in an error state (e.g., lack of connectivity or successor route failure). The sequence numbers of active routes for a given destination are non-increasing moving away from the destination. When a node A needs to establish a route to a destination D, it broadcasts a route request to its neighbors. If A knows a route to D that becomes invalid, A increases the sequence number for destination D and includes it in the route request. A node receiving the request can send back a unicast route reply along its shortest path to node A only if it has an active route to D and the sequence number stored for D is no less than the sequence number in the route request. Otherwise, the recipient must forward the route request.

AODV ensures that no predecessors may reply to a route request from a node on their successor path by increasing the stored sequence number for a destination when a route breaks. Unfortunately, this also inhibits responses from successor nodes with smaller sequence numbers, even if they have a valid loop-free path to the destination. A key limitation to using only sequence numbers as the loop-free routing invariant is that it may inhibit responses both from predecessors – which is good – and from potential successors – which is inefficient. In many cases, this causes the destination to be the only node able to satisfy the request, because it alone can increase its own sequence number for a response.

We present the labeled distance routing protocol (LDR), which is an on-demand routing protocol that uses distance labels rather than sequence numbers, source routing, or inter-nodal coordination to ensure loop freedom at every instant. Section 2 describes LDR through two examples and presents the feasibility conditions and protocol procedures. Like any other on-demand routing protocol, LDR discovers routes to a destination through the network only when there are data for that destination.

Section 3 shows that LDR works correctly and is loop free at every instant. LDR uses a loop-free invariant for each destination similar to that first introduced in DUAL [7] and employs sequence numbers that can be incremented only by the destinations themselves to permit nodes to reset the values of their

Notation			
d_D^A	The measured distance from node A to D. If all link costs are 1, it is a hop count.	fd_D^A	The feasible distance from node A to D, being the minimum d_D^A for the current sn_D^A.
$\star$	An advertisement, for example $sn_D^\star$ is the sequence number in an advertisement for destination D.	sn_D^A	The sequence number of D as known at node A.
$\#$	A solicitation, for example $sn_D^\#$ is the sequence number in a solicitation for destination D. Each issuer adds its own unique identifier $rreqid$.	$rr_D^\#$	Reset required bit (T bit) for solicitation $\#$ for destination D. Indicates that an invariant ordering violation could occur and the path must be reset.

distance invariants. The combined use of a distance invariant and destination-controlled sequence numbers eliminates the need for inter-nodal coordination used in DUAL and other loop-free routing protocols, and enables more efficient responses to route requests compared to AODV, resulting in reduced network load.

Section 4 presents the results of simulation experiments comparing LDR with AODV, DSR, and OLSR. The other three protocols were used as points of comparison given that they are representatives of the state-of-the art in routing for ad hoc networks, and are being considered by the working group on mobile ad hoc networks (MANET) of the Internet Engineering Task Force (IETF). The simulation results clearly show that LDR attains higher packet delivery ratios, smaller packet latencies, and much lower signaling overhead than the other three protocols.

2 Labeled Distance Routing Protocol

LDR is a loop-free on-demand distance vector routing protocol for min-hop ad hoc wireless networks. Loop-freedom is maintained at all times through the use of a feasible distance invariant condition at each node and the use of a sequence number for path resets. This maintains a strict partial order in the network. LDR is based on AODV, and inherits AODV's distance vector nature. Nodes only exchange their estimates of shortest distances. LDR assumes all links have unit cost, and it finds min-hop paths through the network using available information. Because of packet loss, paths may be sub-optimal. Because of LDR's on-demand nature, existing paths may not necessarily be replaced by

a shorter path if one becomes available.

LDR uses a route request (RREQ), route reply (RREP), and route error (RERR) messaging structure that is based on that of AODV. We use the term *advertisement* to denote the portion of a packet that proffers reachability to a destination, and the term *solicitation* to denote the portion of a packet that requests information for a destination. The RREQ in LDR and other on-demand protocols constitute both an advertisement of a route to the node issuing the RREQ and a solicitation for a route to another node. In general, we will discuss advertisements and solicitations as separate entities apart from their concrete realizations in RREQs or RREPs. Table 1 summarizes the notation used to describe LDR in the rest of this paper.

The RREQ is the tuple $\{dst, sn_{dst}, rreqid, src, sn_{src}, fd, dist, flags\}$, where src is the identifier of the source of the RREQ seeking a path to the destination with identifier dst. The sequence numbers for the destination and source are sn_{dst} and sn_{src}, respectively. The $rreqid$ field is a source-specific unique identifier to control the flooding of the RREQ. The source's feasible distance is fd, and the measured distance of the path traversed by the RREQ is $dist$. Control bits are contained in $flags$.

The RREP is the tuple $\{dst, sn_{dst}, src, rreqid, dist, lifetime, flags\}$. The field $lifetime$ is the milli-seconds of time remaining for the route to dst and reflects the maximum time to cache the route if it is not used.

For a given destination D for which node A has a route, it maintains the sequence number originated by D (sn_D^A), its distance to D (d_D^A), its next hop to D, and its feasible distance to D (fd_D^A). The feasible distance fd_D^A is the minimum distance d_D^A ever known to D for the current sequence number sn_D^A.

2.1 Sufficient Conditions for Loop Freedom

The key to loop-freedom in previous works using feasible distances (DUAL [7], LPA [13], PDA [14]) is an invariant condition and some form of inter-nodal synchronization used when such a condition is not satisfied. DUAL states three invariant conditions, with the *source node condition* (SNC) being similar to our numbered distance condition, below. SNC is a minimum-cost condition that renders loop-free shortest paths. LPA uses a condition equivalent to SNC. PDA uses the *loop-free invariant condition* (LFI), which relaxes the shortest-path requirement of SNC to multiple loop-free successors. When nodes cannot satisfy the feasibility condition in these algorithms, they begin a co-ordinated computation to prevent loops while resetting their distances to higher values. In DUAL, a node enters an *active* state for a destination and requires that all up-stream nodes that potentially use it as a successor either accept the

proposed larger distance or change their successor. The node then resets the feasible distance to infinity and chooses a new successor. When a node using LPA or PDA wishes to change its successor and detects that doing so could cause a loop, it co-ordinates with all its immediate neighbors to ensure that none continue to use it as a successor to the destination. The reliable co-ordination with neighbors preempts any possible loops.

The key to loop-freedom in LDR is the dissemination of route requests over a tree, forcing route replies to follow paths in that tree, and enforcing a strict ordering of feasible distances along successor paths. The RREQ tree is formed by the conventional reverse-path flooding technique of AODV. The route reply paths are constructed by looking up the RREQ ID in the RREQ cache and sending route replies only along the reverse path of the flood. Strict ordering of feasible distances for a given destination is attained by ensuring that the following conditions are satisfied.

Numbered Distance Condition (NDC): Node A may accept a route advertisement from neighbor B for destination D and update its routing table independently of other nodes if A has no information about destination D or either one of the following two conditions is satisfied:

$$sn_D^B > sn_D^A \tag{1}$$

$$sn_D^B = sn_D^A \wedge d_D^B < fd_D^A. \tag{2}$$

Feasible Distance Condition (FDC): Node I must set the reset-required flag $rr_D^\# = 1$ in a relayed solicitation for destination D that it forwards if $sn_D^I = sn_D^\#$ and $fd_D^I \geq fd_D^\#$.

Start Distance Condition (SDC): Node I may initiate an advertisement for a solicitation from node A for destination D if I has an active route to D, and either of the following conditions is satisfied:

$$sn_D^I > sn_D^\# \tag{3}$$

$$sn_D^I = sn_D^\# \wedge d_D^I < fd_D^\# \wedge \neg rr_D^\#. \tag{4}$$

NDC is used to allow nodes to change successors without coordination among nodes. FDC is used to enforce ordering of the feasible distances of all the nodes along a path to a destination. SDC is used to allow a node that does not have a neighbor satisfying NDC to find a distant node that can provide a loop-free path to the destination. More specifically, a RREQ specifies the feasible distance of the node that originated the request and that nodes sequence number. Subject to TTL restrictions, other nodes relay the RREQ, until it reaches a node that satisfies SDC, and that node issues a RREP. Any node

along the path taken by the RREQ that does not satisfy FDC sets the T bit to prevent nodes closer to the destination and with a smaller feasible distance from sending a RREP. Requests with the T bit set require a path reset so a node with higher sequence number must reply.

Theorem 1 *Using NDC at node A to update successors for destination D independently of other nodes is sufficient to ensure that no loop is created.*

Proof: Let neighbor I send an advertisement to node A for destination D. If $sn_D^I > sn_D^A$, then node A cannot be on node I's path to destination D, in the absence of node failures. Otherwise it would have known of the higher sequence number. If the sequence numbers are the same, NDC is equivalent to SNC from DUAL, which is shown to be loop-free [7, Theorem 1, pp. 132ff]. ■

Proposition 1 *Using SDC at node I to initiate an advertisement for a solicitation from node A for destination D does not create loops.*

The proof of the above proposition is immediate from Theorem 1 and FDC. If a node I issues an advertisement that is not feasible, it would be rejected by node A according to NDC. Furthermore, a path from A to D can be discovered successfully only if each node in the path from A to D satisfy FDC.

Theorem 2 (Ordering Criteria) *NDC ensures that, along a successor path $P = \{n_k, \ldots, n_1\}$ from node n_k to node n_1, it is always true $(sn_{n_1}^{n_i} < sn_{n_1}^{n_{i-1}}) \vee (sn_{n_1}^{n_i} = sn_{n_1}^{n_{i-1}} \wedge fd_{n_1}^{n_i} > fd_{n_1}^{n_{i-1}})$ for all $i \in [k, 2]$.*

Proof: For any given sequence number for node n_1, $fd_{n_1}^{n_i} \leq d_{n_1}^{n_i}$. According to NDC, for equal sequence numbers, node n_i can use node n_{i-1} as a successor only if $d_{n_1}^{n_{i-1}} < fd_{n_1}^{n_i}$, which implies that $fd_{n_1}^{n_{i-1}} < fd_{n_1}^{n_i}$. At some later time, node n_{i-1} could change its successor, but that can only increase the sequence number or decrease the feasible distance, or remain the same. ■

Note that, because stable paths to destinations are preferable, if node A already has an active route to destination D, it should not change its successor after receiving an advertisement from node B if $sn_D^\star = sn_D^A$, unless its distance to D through node B can be reduced, i.e., $d_D^\star + 1 < d_D^A$.

2.2 Routing Process

This section describes how LDR discovers paths through a network using on-demand solicitations and advertisements. Section 2.2.1 describes the main problem in maintaining a topological order to motivate the route discovery and path reset mechanisms. Section 2.2.2 presents the procedures to initiate

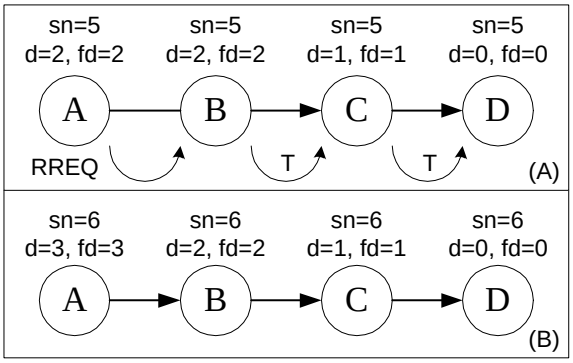

Fig. 1. T-bit for path reset

and relay a solicitation for a route. Section 2.2.3 presents the procedures for a node to update its routing table based on an advertisement, to initiate an advertisement, and to relay an advertisement.

2.2.1 Maintaining topological order with a sparse label set

LDR maintains loop-freedom at all times by using node labels that obey a strict partial order. These node labels come from the set of non-negative integers, which form a sparse label set. As a dynamic routing protocol, LDR must not only label the graph once, it must continually relabel portions of the graph to adapt to changing node positions and link conditions. The principle problem in maintaining node labels in topological order stems from the need to relabel nodes. Because the label set is sparse, it may be impossible to insert a node between existing nodes. Such an out-of-order condition requires some form of reset to relabel a subgraph. This section describes the out-of-order condition and the method LDR uses to relabel a subgraph.

LDR uses a successor-path reset. Previous work in feasible distance based protocols use a predecessor reset, which must include the entire predecessor subgraph. Because LDR uses a successor-path, the scope of the changes is bounded to a single set of successor nodes. The node causing the reset may detect termination of the relabel process by the reception of a feasible advertisement and does not need reliable communications or additional signaling for feedback.

Fig. 1(A) shows a portion of a network that has an out-of-order path. Node A issues a RREQ for destination D. The RREQ travels over node B, which is out-of-order with respect to A for destination D, because the sequence numbers are the same and B's feasible distance is no less than A's feasible distance (see Eq. 8 below). It is not possible to build a route $\{A, B, C\}$ with the metrics as they are. Therefore, node B sets the reset-required `T` bit and relays the RREQ. Node C is a feasible successor to A: it satisfies SDC for A and could reply with a loop-free path. Because the `T` bit is set, however, node

C must relay the RREQ because of an out-of-order condition along the reply path. When C relays the RREQ, it converts the broadcast packet to a unicast packet and sends it along C's unicast path to D. Node D, the destination, may increase its stored sequence number and send an advertisement that resets the path, as shown in Fig. 1(B).

If a node, such as C in Fig. 1, is feasible by SDC except for the reset-required test, then that node may relay the solicitation as a unicast and ensure the TTL is sufficient to reach the destination. The unicast path has two important properties: (a) it is loop-free with regard to all nodes on the solicitation reverse-path, and (b) the semi-reliable nature of unicast traffic ensures that the broadcast to unicast conversion cannot result in permanently undiscoverable nodes. Property (a) is a direct result of how nodes relay a solicitation (Procedure 2 below). If a node satisfies SDC except for the `T` bit test, then it is feasible at all reverse-path nodes and is thus loop-free with respect to them. Property (b) is a result of how nodes detect broken routes. Either through link-layer dropped packet detection or hello timeouts, a bad unicast path will be removed from the routing table. If the source of a solicitation is suitably persistent, a node with a feasible but bad route will eventually stop converting the broadcast to a unicast and enable node discovery.

Property (b) may appear undesirable; one would prefer to never have an unreachable subgraph when a physical path exists. From our simulations, however, it appears that such a condition is rare. The benefit of broadcast to unicast conversion greatly outweighs the rare instances of partition due to broadcast to unicast conversion. A bad unicast path will only result in a complete partition when the path is a network bridge and there is no alternate flooding path.

2.2.2 Route Discovery

A given node A enters into a route computation for destination D when it issues a solicitation for D with identifier ID_A (the *rreqid*). Such a node is called *active* for D in computation (A, ID_A). For a given destination, a node may have at most one active computation. The computation (A, ID_A) terminates when A receives any feasible advertisement for D or a timer expires. If A receives a feasible advertisement for A, the computation terminates in success, otherwise timer expiry indicates failure. The termination of computation (A, ID_A) is a local event at A and does not imply that all solicitations for (A, ID_A) are out of the network or that intermediate nodes participating in the computation have terminated their engagement.

If a node relays a solicitation, it participates in the computation and must cache certain data for a period of time. Such a node is said to be *engaged*

in (A, ID_A). A relay node must record the tuple $\{A,\ ID_A,\ lasthop\}$, where *lasthop* identifies the previous hop node participating in the computation. An engaged node terminates computation (A, ID_A) at the expiry of a timer. A node may be engaged in multiple computations, but may only enter the engaged state once per computation (A, ID_A). For the same computation (A, ID_A) the node A may not be both active and engaged (i.e., it may not relay its own solicitation). Note that a relay node has no state about the destination D of a computation and in fact may go active for D while being engaged in other computations for D. A node that is neither active or engaged in a computation (A, ID_A) is said to be *passive* for (A, ID_A). This is the default state of a node.

Procedure 1 (Initiate Solicitation) *A node* A *that requires a route for destination* D *first checks to see if it is active for* D. *If it is,* A *should queue the packet that requires the route. If* A *is not active for* D*, it becomes active and increments its rreqid. Let* ID_A *be the incremented identifier.* A *issues a solicitation for* D *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 the network policies. If after the final attempt,* A *does not find a route to* D*,* A *should inform the packet origins of the failure and drop the queued packets.*

Procedure 2 (Relay Solicitation) *A node* B *that receives a solicitation* (A, ID_A) *for destination* D *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* D*. Otherwise,* B *relays the solicitation. Let the last hop be node* C *(possibly equal to* A*) and let the new solicitation be denoted by* ‡*. Node* B *must cache the tuple* $\{A,\ ID_A,\ C\}$ *for a sufficient period of time such that all solicitation instances of* (A, ID_A) *have left the network and any advertisements in response to* (A, ID_A) *have had time to complete.*

$$d_S^{\ddagger} \leftarrow d_S^{\#} + 1 \tag{5}$$

$$sn_D^{\ddagger} \leftarrow \begin{cases} sn_D^B & \text{if } sn_D^B > sn_D^{\#} \\ sn_D^{\#} & \text{otherwise} \end{cases} \tag{6}$$

$$fd_D^{\ddagger} \leftarrow \begin{cases} fd_D^B & \text{if } sn_D^B > sn_D^{\#} \\ \min\{fd_D^B, fd_D^{\#}\} & \text{if } sn_D^B = sn_D^{\#} \\ fd_D^{\#} & \text{otherwise} \end{cases} \tag{7}$$

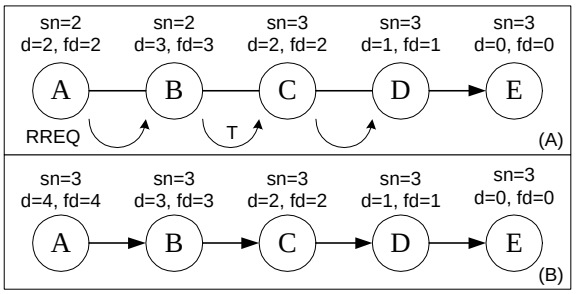

Fig. 2. Resetting the T-bit

$$rr_D^{\ddagger} \leftarrow \begin{cases} rr_D^{\#} & \text{if } (fd_D^B < fd_D^{\#}) \wedge (sn_D^B = sn_B^{\#}) \\ 0 & \text{if } sn_D^B > sn_D^{\#} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

Eq. 8 controls the path-reset request mechanism. The first condition reflects that B matches the ordering criteria, and therefore it relays the path-reset value that already exist. The second condition resets the T bit to zero when B's sequence number exceeds the requested sequence number. This is because B has increased the requested sequence number by Eq. 6, so that any advertisement sent in response to the solicitation functions as a path reset. The third condition sets the T bit when B violates the ordering criteria.

Fig. 2(A) shows a case where node C may reset the T bit in a RREQ. Node A issues a RREQ for E. When B relays the RREQ, it sets the T bit because it is out-of-order with respect to A. Node C does not have an active route to E, so it must relay the RREQ. Because C has a larger sequence number than requested by A and B, it resets the T bit . The RREQ emitted by C has a sequence number of 3. Node D satisfies SDC, so it sends a RREP to C, eventually building the network shown in Fig. 2(B).

The conditions NDC and SDC are based on a comparison of measured distance (d) to feasible distance (fd). The first condition of Eq. 8 compares feasible distance to feasible distance. NDC and SDC must compare distance to feasible distance to ensure an advertisement is loop-free. Eq. 8 does not apply to advertisements, but to solicitations. The idea is that any solicited advertisement must be feasible at both B and A. The case of interest is when the sequence numbers are all equal. Let us assume some node C sends B the advertisement for A, then it must be that $d_D^C < fd_D^B$ for it to be feasible at B. The new distance at B will be $d_D^B \leftarrow d_D^C + 1$. This implies that $d_D^C + 1 < fd_D^A$ for the advertisement to be feasible at A. Because we wish to use only local information in the reset-required decision at B, we take the upper-bound of $d_D^C + 1$, which reduces to $fd_D^B < fd_D^A$ in Eq. 8.

If the destination receives a solicitation with the `T` bit set, it must reset the path. If $sn_D^D > sn_D^{\#}$, it may use the current sequence number to reset the path. Otherwise, it must increment its stored sequence number before sending the advertisement. As with all solicitations, the destination will only send one advertisement per source and rreqid pair.

Eq. 7 ensures that a relayed solicitation elicits a feasible advertisement for the current node and all nodes on the reverse path. A node relaying a route request may need to modify the solicitation, such that the relaying node may use any solicited reply. In Fig. 2, node C receives a RREQ with sequence number 2 and feasible distance 3. A RREP for those metrics would not be feasible at node C, so it updates the solicitation to request sequence number 3 and feasible distance 2.

When a node relays a solicitation, it has to record the last hop of the solicitation, because it will use that last hop as part of a reverse path. When a node relays a RREP, it will look up the (originator, rreqid) pair and force the RREP to follow the RREQ reverse path, even if the node has an active route to the originator. This reverse path information may be kept in the RREQ cache. RREPs must follow the RREQ reverse path to ensure delivery. The process of strengthening the requested metrics in Procedure 2 is specific to each RREQ path. A RREP might not be feasible if it takes an alternate route, such as the unicast routing table path.

2.2.3 Route Advertisement

When a node receives a feasible advertisement, it must update its local routing table by Procedure 3. It may also need to relay the advertisement (or issue a new advertisement) base on Procedure 4.

Procedure 3 (Set Route) *When node* A *adds or updates a route to destination* D *via successor* B*, it updates its sequence number, distance, and feasible distance for destination* D.

$$sn_D^A \leftarrow sn_D^\star \tag{9}$$

$$d_D^A \leftarrow d_D^\star + 1 \tag{10}$$

$$fd_D^A \leftarrow \begin{cases} d_D^A & \text{if } sn_D^A < sn_D^\star \\ \min\{fd_D^A, d_D^A\} & \text{if } sn_D^A = sn_D^\star \end{cases} \tag{11}$$

Procedure 3 guarantees that the feasible distance for a given sequence number is a non-increasing function over time for a given node. The distance for a

given sequence number and feasible distance may fluctuate, but is never less than the feasible distance. This behavior prevents loops.

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 NDC), the node must not relay the advertisement.

It is possible for a RREQ to propagate through the network without creating the reverse path. If a node relays a RREQ without having an active reverse path to the RREQ origin, the relay node must set the (newly specified) `N` bit to indicate that the RREQ is no longer an advertisement for the RREQ origin. The `N` bit is not part of the AODV specification [1]. If the node replying to the RREQ does not have a reverse path, it sets the new corresponding `N` bit in the RREP indicating such. When the origin receives a RREP with the `N` bit set, it may send a unicast RREQ probe along its forward path with the `D` bit set. It 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 D, node A should issue a new advertisement for D 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 (originator, rreqid) pair. Let the new advertisement be denoted by $\dagger$, then $sn_D^\dagger \leftarrow sn_D^A$, $d_D^\dagger \leftarrow d_D^A$.*

As an advertisement $\star$ for destination D progresses through the network, it may happen that the advertisement is not feasible at a relay node R. This may happen when R learns of a route to D after it relayed the solicitation for D corresponding to $\star$. In such a case, the node R must discard the advertisement $\star$ and issue a new advertisement for D using its current metrics, if possible. If node R's route is active, R may issue a new advertisement that is better than $\star$, and is thus still feasible along the reverse path. If node R's route is invalid, R cannot accept $\star$ and cannot issue a new advertisement. This may cause a route discovery timeout.

2.3 Example

Figure 3 shows the directed acyclic successor graph for destination T in a six-node network. The numbers represent the stored distance and feasible distance

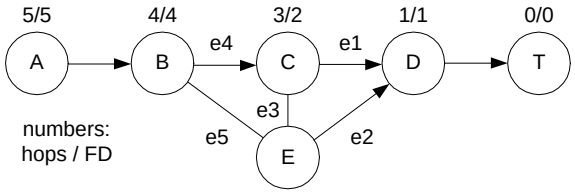

Fig. 3. Example using LDR

to node T. Let us assume that all nodes, except E, have the same sequence number for T. Initially, node E does not have a route to T and issues a RREQ. Nodes $\{B, C, D\}$ respond with RREPs. Let node C respond first. It happens to have a measured distance of 3 and a feasible distance of 2, which may occur due to mobility and changing successors. When node C issues a RREP, it sets the measured distance to 3. When node E receives this RREP, it sets its measured distance and feasible distance both to 4. Node B then issues a RREP with distance 4. Node E ignores that RREP, because it does not provide a shorter distance than E's current feasible distance. Node D issues a RREP with measured distance 1. When E receives that RREP, it updates both its feasible distance and measured distance to 2 and sets its successor to D.

If links e_2 and e_3 fail at some future time, node E issues a new RREQ with feasible distance 2. When node B receives the RREQ, consequently, it must forward the RREQ because it cannot create a RREP. Node B's measured distance of 4 does not satisfy the requesting feasible distance. Node B must also set the `T` bit to indicate the destination must reset the path. Node C must forward the RREQ because its measured distance of 3 is not sufficient either. Node D, finally, could issue a RREP because its measured distance satisfies the requesting feasible distance. However, the reset bit is set and C must unicast the RREQ to T, which would then issue a RREP with a larger sequence number and a distance of 0.

When D receives the path-reset RREP from T, it adopts the larger sequence number and keeps its distance and feasible distance set to "1/1", then relays the RREP to C. When node C receives the RREP from D, it adopts the new sequence number and resets its distance and feasible distance to "2/2", then relays the RREP to B. When node B receives the RREP, it adopts the larger sequence number and sets its distance and feasible distance to "3/3", then relays the RREP to E. Finally, node E receives the RREP, adopts the new sequence number, and sets its distance and feasible distance to "4/4".

3 Analysis

We first show that LDR is loop free at every instant. Then we demonstrate that a source that requests a route to a given destination is successful within a finite time, provided that there is a physical path between source and destination and the network is stable for a sufficiently long period of time after an arbitrary sequence of topology changes. Because all computations in LDR are bounded by finite timers, showing that LDR is live is trivial and is omitted for brevity.

LDR uses a sequence number consisting of a destination-specific time stamp taken from a node's real-time clock and an unsigned monotonically increasing counter. When the counter reaches its maximum value, the node places a new time stamp in its sequence number and resets the counter to zero. We assume that a node's real-time clock does not reset on reboot or adjust for daylight savings. This scheme is adopted because it does not require synchronized clocks or the explicit reset of sequence numbers throughout the network. When a node boots (or reboots), it must use a short reboot-hold, similar to AODV.

The reboot-hold in LDR is used to ensure that no node has a newly-booted node as a next-hop. Such a case could lead to a routing-table loop. AODV admits temporary routing-table loops in the reboot-hold period, because it allows a node to create routes based on received control packets [1, Sec 6.13]. LDR does not allow a node to create routes while in reboot-hold, but otherwise is the same as AODV's reboot hold [*ibid.*]. To minimize the length of the reboot hold period, a node should broadcast a RERR for the all-node address to its neighbors upon reboot to erase any routes through the node. This helps minimize extending the reboot-hold upon reception of data packets.

Theorem 3 *In the absence of node failures, solicitations and advertisements in LDR do not loop.*

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 (which they do not). 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 looses the RREQ cache. Without this information, the node may engage a second time when it reboots, assuming that happens fast enough

that the RREQ has not aged out of the network. A node that rapidly fails and reboots may become multiply engaged in a RREQ. For this to happen a RREQ must stay queued in the network longer than the reboot-hold period. If a node relays a RREQ multiple times, it could create a loop in the RREP path. Two things prevent the RREP loop from forming a routing table loop. NDC prevents any stable nodes from forming a loop based on repeated information. The reboot-hold feature prevents a reset node from participating in the routing protocol until all potential loops timeout from the network.

LDR treats each solicitation independently of each other by identifying each such computation by the identifier of its origin and a sequence number assigned by the origin. This independence enables LDR to guarantee successful termination of simultaneous calculations for the same destination by multiple active nodes.

Lemma 1 *If a node updates its routing table by Procedure 3 and relays advertisements by Procedure 4, then NDC ensures that the network maintains the ordering criteria during the creation of a successor path assuming no node along the path changes successor once on the path.*

Proof: By induction on the number of hops to the source of an advertisement. According to Procedure 4, the relaying of an advertisement is no different than initiating an advertisement. The relay node places its distance and sequence number in the packet before transmitting it. Consider the successor path from node n_k to node n_1 to be $P = \{n_k, \ldots, n_1\}$. Let the time $t^s_{n_i}$ be when node n_i chooses node n_{i-1} as its successor. Node n_1 initiates an advertisement $\star_1$ at time t_0 with $sn^{\star_1}_{n_1} = sn^{n_1}_{n_1}(t_0)$ and $d^{\star_1}_{n_1} = 0$. For node n_2 to choose node n_1 as successor, one of the following cases must be satisfied according to NDC.

Case I: Node n_2 has no information about node n_1. By Procedure 4, $sn^{n_2}_{n_1}(t^s_{n_2}) = sn^{\star_1}_{n_1}$ and $fd^{n_2}_{n_1}(t^s_{n_2}) > d^{\star_1}_{n_1}$. Because the sequence number is a non-decreasing function with time, $sn^{n_2}_{n_1}(t^s_{n_2}) \leq sn^{n_1}_{n_1}(t^s_{n_2})$. The sequence number $\star_1$ is fixed at time t_0 but the sequence number variable stored in $sn^{n_1}_{n_1}$ may increase. Node n_1 will never have a positive distance to itself and the feasible distance at node $n_i \neq n_1$ is always positive, so the ordering criteria is true in this case.

Case II: $sn^{\star_1}_{n_1} > sn^{n_2}_{n_1}(t^s_{n_2})$. Because the sequence number is non-decreasing, $sn^{n_1}_{n_1}(t^s_{n_2}) \geq sn^{n_1}_{n_1}(t_0)$. If $sn^{n_1}_{n_1}(t^s_{n_2}) > sn^{n_1}_{n_1}(t_0)$, the ordering is maintained. Otherwise, $sn^{n_1}_{n_1}(t^s_{n_2}) = sn^{n_1}_{n_1}(t_0)$. In this case by Procedure 3, $fd^{n_2}_{n_1}(t^s_{n_2}) > d^{\star_1}_{n_1}$ so the ordering is maintained.

Case III: $sn^{\star_1}_{n_1} = sn^{n_2}_{n_1}(t^s_{n_2}) \wedge d^{\star_1}_{n_1} < fd^{n_2}_{n_1}(t^s_{n_2})$. As per Case II, $sn^{n_1}_{n_1}(t^s_{n_2})$ could have increased, in which case the ordering criteria is true. If the sequence number did not change, then by Case I, $fd^{n_2}_{n_1}$ is always greater than $fd^{n_1}_{n_1}$.

By the inductive assumption, nodes $\{n_i, \ldots, n_1\}$ have a path that obeys the

ordering criteria. We show that when node n_{i+1} chooses n_i, it maintains the ordering criteria. At time $t^s_{n_i} < t < t^s_{n_{i+1}}$, node n_i emits the advertisement to n_{i+1}. Because n_i does not change successor during $(t^s_{n_i}, t^s_{n_{i+1}})$, $sn^{\star i}_{n_1} = sn^i_{n_1}(t^s_{n_i})$ and $d^{\star i}_{n_1} = d^i_{n_1}(t^s_{n_i})$. Following the three cases above, we only need to validate in Case I $fd^{i+1}_{n_1}(t^s_{n_{i+1}}) > fd^i_{n_1}(t^s_{n_{i+1}})$, as the other statements do not depend on the identity of node n_1.

Case I (revised): By Procedure 4, $sn^{n_{i+1}}_{n_1}(t^s_{n_{i+1}}) = sn^{\star i}_{n_1}$ and $fd^{n_{i+1}}_{n_1}(t^s_{n_{i+1}}) > d^{\star i}_{n_1}$. Because the sequence number is a non-decreasing function, $sn^{n_{i+1}}_{n_1}(t^s_{n_{i+1}}) \leq sn^{n_i}_{n_1}(t^s_{n_{i+1}})$. If the sequence number at node n_i increases (which does not happen by the premise of the lemma), then the ordering criteria is maintained regardless of $fd^{n_i}_{n_1}(t^s_{n_{i+1}})$. If the sequence number remains the same, then $fd^{n_i}_{n_1}(t^s_{n_{i+1}}) \leq fd^{n_i}_{n_1}(t) \leq fd^{\star i}_{n_1}$. Thus, the ordering is maintained. ■

Lemma 2 *Given an established path that obeys the ordering criteria, any change of successor along that path by NDC and Procedure 4 maintains the ordering.*

Proof: Let time $t^c_{n_i}$ be the time at which node n_i changes its successor off an established path. Taking the path in Lemma 1 $P = \{n_k, \ldots, n_1\}$ that obeys the ordering criteria, we show that if some node n_i changes its successor to n_1 to some other node m_j along the path $\{m_j, \ldots, m_1, n_1\}$, which is in order, that the ordering criteria is obeyed. That is, $sn^{n_i}_{n_1}(t^c_{n_i}) < sn^{m_j}_{n_1}(t^c_{n_i})$ or $sn^{n_i}_{n_1}(t^c_{n_i}) = sn^{m_j}_{n_1}(t^c_{n_i}) \wedge\ fd^{n_i}_{n_1}(t^c_{n_i}) > fd^{m_j}_{n_1}(t^c_{n_i})$. For the change to occur, node m_j must issue an advertisement $\star_{m_j}$ at time $t^s_{n_i} < t < t^c_{n_i}$. As per the discussion in Lemma 1, the invariants at m_j at time $t^c_{n_i}$ can be no weaker than at time t, so if $\star_{m_j}$ is feasible at n_i, Procedure 4 maintains the ordering criteria. What we must show is that Procedure 4 does not violate the ordering criteria for node n_{i+1}, which then by the assumption that P is ordered, is sufficient to show that the change does not violate the ordering criteria anywhere along the path.

If $\star_{m_j}$ is feasible at n_i at time $t^c_{n_i}$, then either $sn^{\star_{m_j}}_{n_1} > sn^{n_i}_{n_1}(t^c_{n_i})$ or $sn^{\star_{m_j}}_{n_1} = sn^{n_i}_{n_1}(t^c_{n_i}) \wedge d^{\star_{m_j}}_{n_1} < fd^{n_i}_{n_1}(t^c_{n_i})$. In the first case, Procedure 4 ensures that n_i's sequence number increases, which satisfies the ordering criteria at node n_{i+1}. In the second case, Procedure 4 decreases or leaves unchanged the feasible distance at node n_j, which also satisfies the ordering criteria at node n_{i+1} . ■

Theorem 4 *LDR is loop-free at every instant, as long as nodes update their routing tables according to NDC and Procedure 3, and relay messages by Procedures 2 and 4.*

Proof: Let node I be the node issuing the routing advertisement for destination D with terminus A and let the advertisement take the path $P = \{n_1, \ldots, n_j\}$, where $n_1 = I$ and $n_j = A$. I may be equivalent to D. Let the

path from I to D be $Q = \{m_1, \ldots, m_k\}$, where $m_1 = D$ and $m_k = I$ and it may be empty if $n_1 = D$.

For a loop to form, some node in P must be on the path Q. The path P is loop-free by Theorem 3. If $n_1 = D$, then the path Q is empty, so the theorem trivially holds. If $n_1 \neq D$, we show that it is impossible for a node in P to be on I's successor graph using a proof by contradiction.

Let node A be the first node in P also in Q. At time t_0, assume that path Q exists and is loop free. In the proof by contradiction, let node A be some node $m_i, 1 < i < k$. At time t_1, A chooses a successor path based on an advertisement from I. By the ordering criteria, $sn_D^A(t_0) > sn_D^I(t_0)$ or $sn_D^A(t_0) = sn_D^I(t_0)$ and $fd_D^A(t_0) < fd_D^I(t_0)$. At this time, node I sends an advertisement along the loop-free path P to A, thus $sn_D^\star = sn_D^I(t_0)$ and $d_D^\star = d_D^I(t_0)$.

At time t_1, node A receives the advertisement. The sequence number for a destination is a non-decreasing function with time, so $sn_D^A(t_1) \geq sn_D^A(t_0)$.

In the case where $sn_D^A(t_0) > sn_D^I(t_0)$, A cannot accept the advertisement because $sn_D^\star < sn_D^A(t_1)$.

In the case where $sn_D^A(t_0) = sn_D^I(t_0) = sn_D^A(t_1)$, we know that $d_D^\star \geq d_D^I(t_0) \geq fd_D^I(t_0) > fd_D^A(t_0)$. Because the feasible distance is a non-increasing function with time for the same sequence number, $fd_D^A(t_1) \leq fd_D^A(t_0)$, so $d_D^\star \geq fd_D^A(t_1)$. By NDC, node A cannot accept the advertisement.

In the case where $sn_D^A(t_0) = sn_D^I(t_0) < sn_D^A(t_1)$, we have $sn_D^\star < sn_D^A(t_1)$ and A cannot accept the advertisement. ■

To show that LDR successfully terminates route calculations, we consider several lemmas that cover different situations in route discovery. Theorem 5 combines the lemmas to show that LDR terminates a route discovery successfully in an error-free and stable connected network.

In our analysis of the route calculation process, we have to make certain assumptions about the network. No routing protocol can converge if there are certain errors or if the network topology changes too frequently. A route discovery process will also fail if no node exists that can satisfy the route, such as in a partitioned network. We will thus impose three conditions when we consider a node A initiating a route discovery: (1) There exists a node B, perhaps equal to D, such that node A and B are connected and B's route to D is feasible for every node along the path from A to B; (2) node A sends the solicitation with large enough time-to-live that it may reach a node capable of sending a feasible advertisement back to A; and (3) relay nodes follow Procedure 2.

Lemma 3 *If a single node A initiates a route discovery for destination D identified by (A, ID_A) in an error-free stable connected network, LDR guarantees that each solicitation for a route is answered with a feasible advertisement for the destination.*

Proof: We consider the case of the first advertisement $\star_A$ to reach A in response to (A, ID_A). If there are multiple advertisement, it does not affect the fact that A terminates successfully based on the first advertisement; A may improve its route.

Let node A send a solicitation for destination D, and let that solicitation traverse the path $P = \{n_1 \dots n_{k-1}\}$ before arriving at node n_k (possibly equal to D) which satisfies SDC. Node n_k issues an advertisement for D with terminus A. We show that the way in which nodes relay solicitations ensures that any solicited advertisement is usable by the relaying nodes. It is usable by A by virtue of satisfying SDC. By Procedure 2, we know any advertisement sent in response to (A, ID) will follow the reverse path of the solicitation, so it will follow the path P from n_k to A.

Let us first consider the case in which no node along the solicitation path P is affected by another route discovery event for D during the computation period. In this case, no node along P can satisfy A's invariants and has an active route; otherwise, such a node would have responded to the solicitation instead of node n_k. Each node $n \in P$ matches one of three cases: (i) n has no information about D, (ii) n's information is invalid, or (iii) no node n satisfies SDC for A.

In case (i), node n may use any advertisement sent by n_k. In case (iii), the advertisement sent by n_k will satisfy the invariants at n because it satisfies A.

For case (ii), we show by induction that the advertisement issued by n_k in response to node A's solicitation will satisfy all nodes along P if they followed Procedure 2. Let us first consider node n_1, being the immediate neighbor of node A. If $sn_D^{n_1} > sn_D^{\#}$, then node n_1 has an invalid route to destination D and placed its own sequence number and feasible distance in the advertisement. Because the sequence number increased, node A may use any solicited advertisement. If $sn_D^{n_1} = sn_D^{\#}$, then node n relayed the solicitation with the minimum feasible distance of nodes A and n_1, which ensures that nodes n_1 and A may use the solicited advertisement. Otherwise, node A had a higher sequence number than node n_1, and any solicited advertisement will be usable by node n_1.

If by the inductive assumption all nodes $A \dots n_{i-1}$ may use the advertisement, then we show that the actions at node n_i do not affect predecessors, and n_i will be satisfied. Node n_i may only increase the sequence number or, keeping the sequence number the same, decrease the feasible distance; n_i cannot invalidate

the usability of the advertisement for any predecessor node. Because node n_i may only improve or leave the same the requested sequence number and distance, it ensures that any solicited advertisement is usable by itself and the predecessor path. ■

Lemma 4 *Considering the case of Lemma 3, let there be one or more other nodes $m_i \notin P$ that go active for D during the calculation (A, ID_A). LDR guarantees that each solicitation for a route is answered with a feasible advertisement for the destination.*

Proof: By symmetry, if we show A's route calculation successfully terminates, the other calculations for set m successfully terminate. We thus restrict ourselves to considering A's calculation.

Because the relaying of a solicitation does not change the invariants stored at a node or the information cached for calculation (A, ID_A), we only need to consider the interaction of advertisements. Let $\star_j$ be the set of advertisements generated by the calculations (m_i, ID_{m_i}). For these advertisements to affect (A, ID_A) they must intersect $\{A, P\}$. If an advertisement intersects a node $n \in \{A, P\}$, it implies that n relayed a corresponding solicitation and by Lemma 3, the advertisement is feasible at n.

If one or more of $\star_j$ intersect A before $\star_A$, then A's calculation (A, ID_A) terminates in success. By Lemma 3, the first $\star_j$ to reach A is feasible at A.

Let some $\star \in \star_j$ intersect some node $n \in P$ before $\star_A$. By Lemma 3, $\star$ is feasible at n and n may relay it toward the appropriate m. The sequence number and feasible distance at n could only have remained the same or been improved by $\star$.

If they remained the same, $\star_A$ will be usable because there was no change. It could be that n changed from an inactive to an active route, but that does not affect the feasibility of $\star_A$ at n.

If at n the sequence number increased or distance decreased, then either (1) $\star_A$ is still feasible at n because some node in $\{A, \ldots, n\}$ requested a larger sequence or a lower distance, or (2) $\star_A$ is not feasible at n because n has a larger sequence number or lower distance. In case (1), $\star_A$ will improve the route at n and continue as per Lemma 3. In case (2) n will follow Procedure 4. It will discard $\star_A$ and issue a new advertisement $\dagger_A$ with the better route. ■

Lemma 5 *Considering the case of Lemma 3, let there be one or more other nodes $m_i \in P$ that go active for D during the calculation (A, ID_A). LDR guarantees that each solicitation for a route is answered with a feasible advertisement for the destination.*

Proof: A node $n \in P$ going active for destination D does not change the cached information from the engagement (A, ID_A) nor does it change the invariants at n. It follows from Lemma 4 that any advertisements sent in response to calculation (n, ID_n) cannot interfere with calculation (A, ID_A). ■

Theorem 5 *If node A initiates a route discovery for destination D identified by (A, ID_A) in an error-free stable connected network, LDR guarantees that A receives a feasible advertisement for D.*

Proof: Considering the set of possible events, we must have the situation described in Lemma 3, Lemma 4, Lemma 5 or both Lemmas 4 and 5. The first three situations follow from the Lemmas. The four situation, being nodes both on and off P go active for D during the calculation (A, ID_A), also follows from the Lemmas. No matter how many other nodes go active for D during A's calculation, the other calculations cannot affect the cached information at nodes engaged in (A, ID_A) and by Lemma 4, any advertisements generated by those other calculations must have invariants at least as strong as necessary to complete (A, ID_A) if they are to interfere with it. ■

4 Performance

We present results that show LDR out-performs other routing protocols over varying loads and mobility. Simulations are run in GlomoSim [15] for LDR, AODV, DSR, and OLSR. The simulations follow the parameters in [16]. Because of known packet jitter problems in the OLSR code from INRIA for Linux [17], we introduce a new FIFO jitter queue to OLSR. The FIFO jitter queue adds a uniformly chosen inter-packet jitter between 0 and 15ms and maintains FIFO packet order. The modified code performs substantially better than the base OLSR. We use the DSR implementation from GlomoSim, which implements DSR Draft 3.

To verify that DSR's poor performance in our simulations was not due to implementation bugs in GlomoSim, we also ran our simulations in Qualnet 3.5.2, which implements DSR draft 7, and observed similar results. Fig. 8 for Qualnet has the same mobility and traffic load patterns as Fig. 5 in GlomoSim. The performance of DSR is slightly better, but still shows the same downward trend with increasing mobility. The AODV implementation in Qualnet also showed poor performance in the 120 packet-per-second scenario. Although we only present one graph, DSR performance in other Qualnet runs mirrored GlomoSim over all tested scenarios.

We use the following optimizations to LDR:

Multiple RREPs: A node may relay multiple RREPs for the same (source, rreqid) pair as long as only RREPs with higher sequence numbers or shorter distances cross over time.

Request as error: If node A receives a solicitation for destination D from neighbor B and A has an active route for D with next hop B, it is likely that B no longer has a valid route to D. If $fd_D^{\#} > d_D^A - 1$, then B should have answered the query if it had an active route.

Reduced distance: Because of mobility and link failures, it is often desirable in an ad hoc network to use non-optimal bounds. A node may place in a RREQ an *answering* distance extension, which is any distance no greater than the node's feasible distance. Nodes use the answering distance to test SDC. We use a factor of 0.8, truncated to the lowest integer no less than 1.

Minimum lifetime: A node should not respond to a RREQ if the lifetime remaining in its active route is less than a threshold. We use 1/3 the $ACTIVE\_ROUTE\_TIMEOUT$, or 1 second using default values. If a node receives such a RREQ, it relays the RREQ.

Optimal TTL: The initial TTL of a RREQ should be set according to the known distance and RREQ feasible distance. Here, let FD be the value, possibly lowered by the *reduced distance* optimization, given in the RREQ. The initial TTL should be $TTL = D - FD + 1 + LOCAL\_ADD\_TTL$.

There are two main sets of simulations, one on a 50-node network over a 1500m x 300m terrain, and one on a 100-node network over a 2200m x 600m terrain with 10-flow and 30-flow traffic loads using 512 byte packets at 4 packets per second per flow. The simulations use the 802.11 MAC layer with a 275m transmission range. The simulations run for 900 seconds. Nodes move between 1 m/s and 20 m/s. Flows have a mean length of 100 seconds, chosen from an exponential variate. We repeat each configuration (nodes, number of sources, routing protocol, and pause time) for 10 trials using different random number seeds. The LDR results reflect using the suggested optimizations.

We present six metrics. A "transmitted" packet count includes all hop-wise transmissions. An "initiated" packet count only includes the first transmission of a packet. The *delivery ratio* is the fraction of CBR data packets received at destinations to data packets transmitted. The *network load* is the total number of control packets (RREQ, RREP, RERR, Hello, TC, etc.) transmitted divided by total number of received data packets. The *RREQ Load* is the total number of RREQs transmitted per received data packet. The *data latency* is the mean end-to-end latency of data packets. The *RREP Init* is the number of RREPs initiated per RREQ initiated. The *RREP Recv* is the number of hop-wise usable RREPs received per RREQ initiated. A RREP may be usable at multiple nodes along its path to the RREQ origin. In the graphs, the vertical

Table 1
Performance average over all pause times

flows	protocol	delivery ratio	Net load	latency (sec)
10	LDR	0.992 ± 0.003	0.464 ± 0.132	0.066 ± 0.034
10	AODV	0.962 ± 0.017	0.777 ± 0.203	0.731 ± 0.538
10	DSR	0.736 ± 0.087	8.337 ± 3.719	2.111 ± 0.831
10	OLSR	0.904 ± 0.026	5.773 ± 2.040	0.029 ± 0.004
30	LDR	0.857 ± 0.044	2.063 ± 0.759	0.656 ± 0.113
30	AODV	0.822 ± 0.028	2.353 ± 0.623	1.030 ± 0.105
30	DSR	0.603 ± 0.094	3.270 ± 1.625	3.837 ± 1.567
30	OLSR	0.799 ± 0.044	2.680 ± 1.009	0.489 ± 0.146

Table 2
RREQ and RREP statistics

flows	protocol	RREQ load	RREP Init	RREP Recv
10	LDR	0.410 ± 0.121	3.379 ± 0.129	1.335 ± 0.020
10	AODV	0.609 ± 0.136	5.641 ± 1.356	1.303 ± 0.006
10	DSR	0.043 ± 0.015	–	–
30	LDR	1.532 ± 0.556	7.183 ± 0.998	1.642 ± 0.105
30	AODV	1.746 ± 0.455	7.775 ± 0.774	1.533 ± 0.158
10	DSR	0.033 ± 0.011	–	–

error bars represent the 95% confidence interval for all measurements.

Table 1 summarizes the results by averaging over all pause times and both 50-node and 100-node scenarios for a given number of flows. The columns show the mean value and the 95% confidence interval range. In terms of delivery ratio, LDR has the highest delivery rate at 10 flows. At 30 flows, LDR and AODV are statistically equivalent. In terms of network load, LDR and AODV have the lowest load for 10 flows. At 30 flows, all protocols are statistically equivalent. In terms of latency, OLSR has the lowest latency with 10 flows. At 30 flows, LDR and OLSR have statistically equivalent latencies.

Table 2 examines the RREQ and RREP statistics. DSR has a very low number of RREQs issued per data packet received (RREQ Load). But it also has a very low delivery ratio. We did not collect statistics on *RREP Init* and *RREP Recv* for DSR. At 10 flows, LDR generates about 40% fewer RREPs per RREQ than AODV, but it receives more usable RREPs. Per RREQ packet initiated, LDR generates 3.379 RREP packets. Of those, only 1.335 are usable, which

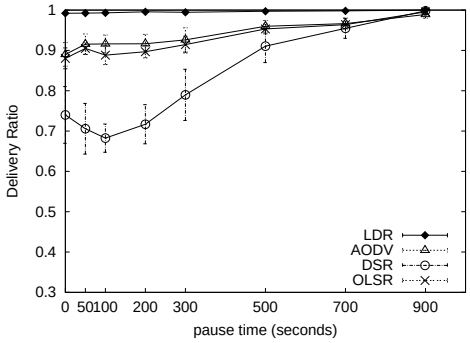

Fig. 4. Delivery 50-nodes, 10-flow

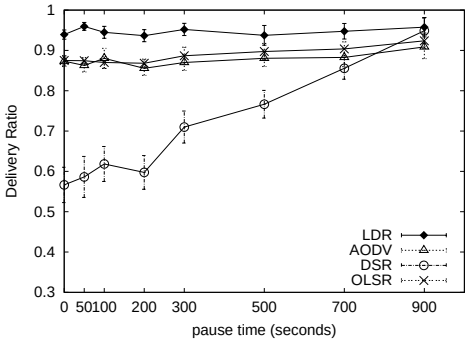

Fig. 5. Delivery 50-nodes, 30-flow

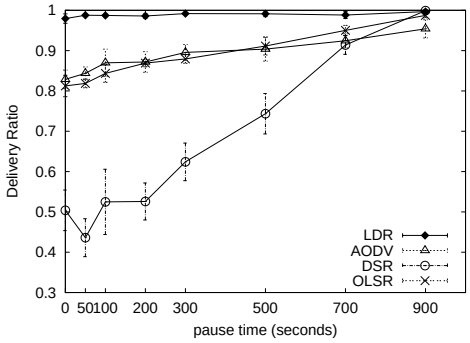

Fig. 6. Delivery 100-nodes, 10-flow

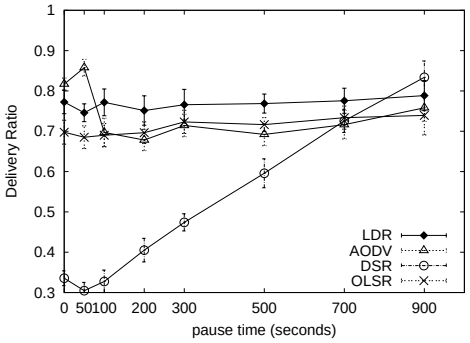

Fig. 7. Delivery 100-nodes, 30-flow

is a 40% efficiency rate. AODV generates 5.641 RREP packets per RREQ initiated, but of those only 1.303 are usable, which is a 23% efficiency rate. At 30 flows, LDR and AODV have statistically equivalent RREQ and RREP statistics, with about a 20% RREP efficiency rate.

Figs. 4, 5, 6, 7 show the delivery ratio. In practically all cases, LDR has a higher delivery ratio than the other protocols. The exception is at high load and high mobility, where AODV sometimes performs better, and at high load and low mobility, where DSR sometimes performs better. In the low-load, 40pps, scenarios, LDR maintains a very high delivery ratio at all mobility speeds. The minimum delivery ratio is 98.5% for the 200s pause time in Fig. 6 (100-node, 10-flow, 40pps). The average LDR delivery ratio for all pause times and all 10-flow scenarios is 0.992 ± 0.002. The next-best protocol in terms of delivery ratio is AODV, but in many cases it is almost identical to OLSR. For 30-flows, LDR, AODV and OLSR are statistically identical when averaged over all pause times for 50-node and 100-node scenarios.

Fig. 9 shows the mean destination sequence number for LDR and AODV for low load and high load. LDR with 10-flows has a maximum mean sequence number of 0.8 and with 30-flows has a maximum of 15.8. AODV, on the other hand, has a 10-flow maximum mean of 104 and a 30-flow maximum mean of 108.

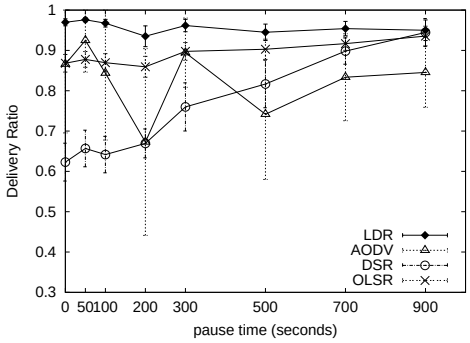

Fig. 8. Qualnet 50-node, 30-flow

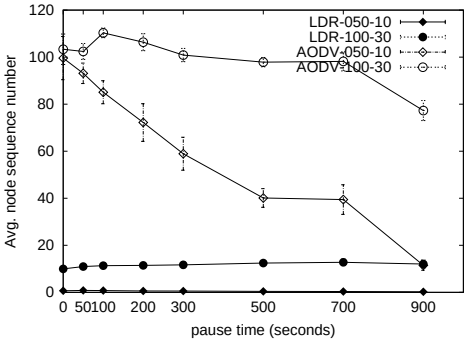

Fig. 9. Average sequence number

5 Conclusion

We have presented LDR, a novel on-demand loop-free routing protocol using a new distance label invariant based on DUAL and a messaging structure based on AODV. LDR removes the requirement of AODV for nodes to independently increase other node's sequence numbers, placing firm control of a sequence number with the owning node. LDR avoids loops by introducing a second routing invariant, the feasible distance. If the measured distance of an advertisement is less than a node's feasible distance, there can be no loop. Destination sequence numbers play the role of resets for feasible distances and allow nodes to increase their feasible distance along paths. LDR's reset mechanism is based on a node's successor path, so the scope of the reset is limited. Through simulation, we showed that LDR exhibits better packet delivery ratios than AODV, DSR, and OLSR, except at 3 out of 32 data points. At times, LDR's delivery ratio is significantly higher than other protocols and is never significantly worse. Our data also shows that LDR's network load is less than or comparable to the loads of other protocols. In terms of mean latency of data packets, LDR is second to OLSR at low load and comparable to OLSR at high load.

An important topic for future work is to extend LDR to networks with variable link costs. The main point of difficulty is Eq. 8. The reset-required feature detects when a feasible reply would be out-of-order for the requesting node. This means that Eq. 8 must take the worst case of the highest link cost to estimate the ordering of an advertisement. If a node has widely varying link costs, such a worst-case assumption could cause excessive reset-required conditions, which in turn would create excessive flooding.

References

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

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

[3] D. B. Johnson, D. A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Mobile Computing, Vol. 353, Kluwer Academic Publishers, 1996. URL `citeseer.nj.nec.com/johnson96dynamic.html`

[4] M. Spohn, J. J. Garcia-Luna-Aceves, Neighborhood aware source routing, in: MOBIHOC 2001, ACM SIGMOBILE, 2001, pp. 11–21.

[5] V. D. Park, M. S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in: IEEE INFOCOM, 1997, pp. 1405–13 vol.3.

[6] J. Raju, J. J. Garcia-Luna-Aceves, A new approach to on-demand loop-free multipath routing, in: IC3N'99, IEEE, 1999, pp. 522–7.

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

[8] J. J. Garcia-Luna-Aceves, M. Spohn, Source-tree routing in wireless networks, in: (ICNP'99), 1999, pp. 273–82.

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

[10] R. Ogier, F. Templin, B. Bellur, M. Lewis, Topology broadcast based on reverse-path forwarding (TBRPF), iETF Internet draft, draft-ietf-manet-tbrpf-06.txt (Nov 2002).

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

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

[13] J. Garcia-Luna-Aceves, S. Murthy, A path finding algorithm for loop-free rotuing, IEEE/ACM Trans. Networking.

[14] S. Vutukury, J. Garcia-Luna-Aceves, A simple approximation to minimum-delay routing, in: SIGCOMM 1999, 1999, pp. 227 – 238.

[15] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla, GloMoSim: A scalable network simulation environment, Tech. Rep. 990027, UCLA Computer Science Department (1999).

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

[17] A. Plakoo, A. Laouiti, INRIA OLSR draft 3 linux source code, http://menetou.inria.fr/olsr/#code, ported by Marco Spohn (2001).