Ad hoc routing with distributed ordered sequences

Marc Mosko*
* Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
Email: mmosko@parc.com

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

***Abstract*— We propose a new hop-by-hop routing protocol for ad hoc wireless networks that uses a novel sequence number scheme to ensure loop-freedom at all times. We use a single large per-destination label space to order nodes in a topological sort (directed acyclic graph). Nodes manipulate the label set in-network without needing destination-controlled resets, so path repair is localized. The label size is large enough that it should never be exhausted in the lifetime of any given network. Route request flooding is performed through a new method that exploits the inherent partial order of the network, so nodes can share RREQ floods. Whereas most previous route request pruning techniques create a request tree, the new technique creates a directed acyclic request graph. Simulation results compared to AODV, DSR and OLSR show that the new protocol has in most cases equivalent or better packet delivery ratio and latency, but with a fraction of the network load.**

I. Introduction

Wireless ad hoc computer networks are communications networks in which each node may be mobile and has at least one radio interface. There is no central infrastructure, such as cell towers, base stations, for access points. Examples of these networks include tactical military applications, commercial vehicle-to-vehicle systems such as DSRC [1], or emergency rescue impromptu networks. The Internet Engineering Task Force (IETF) studies such networks under the mobile ad hoc networks (MANET) working group. Three MANET routing protocols have request for comments (RFC) status and two have internet draft (ID) status. The three MANET RFC protocols are the Adhoc On-demand Distance Vector (AODV) routing protocol [2], the Optimized Link State Routing (OLSR) protocol [3], and the Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) [4]. The two Internet Drafts are the Dynamic Source Routing Protocol (DSR) [5], and the Dynamic MANET On-demand (DYMO) Routing [6]. TBRPF and OLSR are examples of link-state protocols, where nodes exchange topology information and execute a shortest path algorithm (e.g. Dijkstra's) using the topology information they maintain to find routing paths. Unfortunately, neither of these protocol are loop-free, which means that the routing tables at nodes may point in a directed cycle at times. AODV, DSR, DYMO operate as on-demand protocols, which means that they do not maintain routes for all destinations, only those for which there is traffic. Nodes discover paths in on-demand routing protocols through route request (RREQ) floods in the network and unicast route reply (RREP) advertisements.

The IETF on-demand protocols attempt to maintain loop-free operation through different techniques. DSR is a source-routing protocol, so each source node must maintain complete path information to each in-use destination. If there are path changes, then the protocol must either drop the traffic or use a recovery technique, which has been shown to be prone to looping. AODV uses distance labels (hop count) to order nodes along shortest paths. If a node needs to repair a path, it increments a destination sequence number and broadcasts a RREQ. By incrementing the sequence number, it prevents any predecessors (nodes that use the current node as a successor) from replying and maintains loop-freedom. DYMO also uses distance labels and sequence numbers to maintain loop-freedom. All RREQ broadcasts must be answered by the destination node, and the destination will increase a route sequence number if the requested sequence number is larger than the stored number, or the reply path length is longer than the requested path length.

Jaffe and Moss [7] made a key observation in the study of loop-free distance vector routing protocols. They noted that a node may independently add a new successor to a destination if the new distance does not exceed the current distance. If the distance increases, then the node must coordinate with other nodes through some mechanism. The coordination must ensure both that the new successor path is loop-free and that the new distance at the node is not out-of-order with respect to any predecessors. The conditions in DUAL [8] generalize the work of Jaffe and Moss and provide a reliable mechanism to reset the ordering information at predecessors through a diffusing computation [9]. Reliable diffusing computations, however, are impractical in a wireless ad hoc network due to their overhead and latency problems timing out non-existent links.

Loop-free ad hoc routing protocols address the reset condition several ways. One approach is to use source routing, such as in DSR and variations on it. Another approach consists of relying on reliable internodal coordination based on the values of distances to destinations reported by nodes (e.g., DUAL [8], LPA [10], ROAM [11]). Yet another approach, used in protocols similar to AODV, consists of having nodes use a distance label (D) and a sequence number (SN) to reset distance values. In this case, The ordered pair (SN, D) constitutes a lexicographic order. AODV manipulates the pair such that when there is a link break a node requests the next higher SN. This avoids loops, but it also invalidates many potential loop-free paths. The Labeled Distance Routing (LDR) protocol [12] also uses a distance label and sequence number, but it manipulates the pair such that only the destination can increase its own sequence number and it is much more likely than AODV to find a localized repair. Because LDR uses integer distance labels, there are some cases when a RREQ/RREP operation cannot put a new path in-order because the re-labeling cannot put a new node between two adjacent integer labels. In such cases, the RREQ will travel all the way to the destination, which may increase the sequence number and relabel an entire successor path. A third technique is to perform a topological sort of nodes using an abstract node label. Some examples of this sort are GB [13], LMR [14], and TORA [15]. These algorithms are sometimes called link-reversal algorithms because they operate iteratively but assigning ever-increasing node labels until there is a complete topological sort rooted at the destination. They have been shown to have oscillation and convergence problems. These protocols avoid a reset or overflow condition by assuming the node labels are large enough that they will not overflow for the lifetime of the network. The Split Label Routing Protocol (SLRP) [16] also uses an abstract node label, in this case a 64-bit fraction (32-bit numerator, 32-bit denominator). This arrangement allows SLRP to insert nodes in any existing path without extensive re-labeling, solving the integer problem noted in LDR. SLRP also uses a destination-controlled sequence number as a reset to protect against fraction overflows due to many path inserts. Hence, the ordering label is (SN, fraction), a 96-bit number. It may be possible to make an argument that SLRP could operate without sequence number resets with a large enough label (e.g. 64-bit numerator and denominator).

The focus of this paper is the introduction of a new graph re-labeling that enables localized route repairs whenever possible. In AODV, a RREQ must usually be answered by the destination itself, because the increased sequence number in the request invalidates many potential loop-free routes. DYMO, by construction, must get a RREQ to the destination for a RREP. DSR uses source routing, so while it has good performance with low mobility, the performance degrades as mobility increases. Section II presents the Distributed Ordered Sequences (DOS) routing protocol, which operates through a topological sort of abstract node labels. Unlike the fraction in SLRP, DOS uses a single 128-bit integer node label. This large size, as shown below, should outlast any ad hoc network and further study may show that a smaller 96-bit or 64-bit number is sufficient. It achieves loop-freedom by having nodes maintain a label that is always in-order with respect to a successor graph. DOS avoids node insertion problems by keeping a modest label spacing between adjacent nodes, when possible. Because these sequence numbers may change frequently, we allocate a large number of bits to the sequence number such that a node may perform a very large number of operations per second and not have the sequence number overflow for the life of a network. The basic mechanics of DOS is similar to other on-demand routing protocols, such as AODV or DSR. A node discovers new routes through a broadcast route request (RREQ) and receives replies via route replies (RREP). DOS, however, does not use destination controlled sequence numbers like AODV or source routing like DSR. Instead, as a RREQ propagates over the network, intermediate nodes adjust the requested label in the RREQ to ensure that any solicited RREP is usable along the entire RREQ reverse-path and at the intermediate node. We assume a route error (RERR) procedure similar to AODV, which we do not describe in this paper for brevity.

Section III analyzes DOS with proofs of correct operation. We show that DOS is loop-free at all times. We also show that DOS is lockout-free; that is, in a stable network any node originating a RREQ will receive a usable RREP and thus all nodes may make progress in finding routes. Section IV discusses our simulation implementation of DOS. Section V presents simulation results, which show that DOS performs as well as or better than other state-of-the-art ad hoc routing protocols, but with a fraction of the network load.

II. Routing Protocol

DOS is a hop-by-hop routing protocol that ensures loop-free paths by maintaining node labels in a strict topological order. We first describe the label set used by DOS, then present the three invariants of the routing protocol. The invariants ensure loop-freedom at all times. The protocol uses six procedures to implement message passing. DOS also uses a route error procedure similar to AODV, which we do not describe here. The six procedures are node initialization, initiate a route query, receive a route query, relay a route query, initiate a route reply, and receive a route reply. Relaying a route reply is equivalent to receiving and initiating a route reply. After the detailed description of the protocol, we present two examples of DOS. The first example shows operation on an un-labeled graph. The second example shows how DOS repairs a route over nodes with existing labels.

DOS is based on a label set, $\mathcal{G}$, which we call a Global Sequence Number (GSN) space. There exists a total order $(\mathcal{G}, \leq)$, and the implied operators $=$, $<$, and $>$. $\mathcal{G}$ is a well-ordered set [17, Section 2.2]. We implement the global sequence number as an unsigned 128-bit sequence number. We chose 128 bits to ensure that the number space should not be exhausted in the foreseeable lifetime of a network. Over 100 years, if there are 100 route computations per second, and DOS skips 100 sequence numbers per computation (to allow some slack as per Procedure 4 below), and this operation is done distributively by 100 nodes in parallel, we would use fewer than 52 bits.

Each node maintains three data structures per destination D, as shown in Table I. ao_D^A is the advertised ordering at node A for destination D. It is the minimum ordering node A ever advertised in a RREP packet for destination D. This is the key value that is ordered to guarantee loop-freedom. If a node A has never advertised a route to destination D, then ao_D^A is assumed to be the maximum value. S_D^A is the set of next-hop successors usable by node A towards destination

D. Each successor $x \in S^A_D$ is guaranteed to be loop-free, but not of minimum distance. Nodes may maintain distances or other metrics, but that is outside of the scope of the current work. $so^A_{D,x}$ is the stored ordering at node A advertised by neighbor x for destination D. When node A adds neighbor x to its successor list S^A_D, it must cache the ordering advertised by node x in $so^A_{D,x}$.

The ordering function $\mathrm{ord}(\cdot)$ returns the label of its argument. Its range is $\mathcal{G}$. For instance, when applied to a RREQ q, $\mathrm{ord}(q)$ means the requested ordering contained in q. When applied to a RREP r, $\mathrm{ord}(r)$ means the ordering advertised in r.

DOS routes towards nodes with lower labels, similar to how a hop count would work. To maintain the common definition of $\leq$, this means that a node initializes the GSN for a route to the maximum value and decreases it over time. At a node N, the advertised ordering for destination D is the minimum label ever transmitted in a route reply for D. A directed path $\{v_k, \ldots, v_0\}$ from node v_k to node v_0 implies that the advertised orderings of each node satisfy $ao^{v_k}_{v_0} > \cdots > ao^{v_0}_{v_0}$.

There are two boundary cases. Node i should have itself in its successor set $S^i_i = \{i\}$ and consider itself to have the minimum successor label for itself $so^i_{i,i} \leftarrow 0$. This allows node i to respond to any route request for i (see Eq. 7). The other boundary case is when a node i has no state for a destination d. In such a case, node i should consider itself to have $ao^i_d \leftarrow \infty$. In our implementation of the GSN, ∞ corresponds to a 128-bit binary string of '1'.

DOS is compatible with existing route request flooding procedures, such as the packet cache method of AODV or DSR. The purpose of a route request flooding procedure is to eliminate duplicate and cyclic RREQ packets as they flood the network. The packet cache methods of AODV and DSR accept one predecessor, namely the first predecessor to relay a RREQ, and form a reverse-path tree. Due to the way DOS relays route requests, there is an opportunity to use a different technique. One may accept a larger predecessor graph, similar to the method used by Lee and Gerla [18]. Lee and Gerla use the RREQ hop count to detect cyclic RREQs, but in DOS we use the requested ordering of a RREQ to detect cyclic RREQs. A node may accept and relay any RREQ with a unique RREQ ID. For duplicate RREQ IDs, a node will suppress relaying a RREQ, but will add the RREQ last-hop to the predecessor graph so long as the requested ordering is no less than what the node has already relayed.

Rule 1 (Non-increasing Advertisement Ordering (NIAO)): The advertised ordering for a route may never increase. This means that if a node does not have a feasible successor that satisfies the existing advertised label, it must discover a new successor with smaller ordering.

At a node A, for every destination d with non-empty successor set S^A_d,

$$ao^A_D(t_2 > t_1) \leq ao^A_D(t_1) \tag{1}$$

■

Symbol	Definition
S^A_D	Set of next-hops to destination D at node A
ao^A_D	The advertised ordering (gsn) for destination D
$so^A_{D,x}$	Ordering (gsn) advertised by next-hop x for D known at A
$\mathcal{G}$	A set of Global Sequence Numbers (GSNs) (integers)
$\mathrm{ord}(\cdot)$	A function mapping an object $\rightarrow \mathcal{G}$

TABLE I

NOTATION

Routes may timeout due to non-use. After a certain cache period, a node may erase the ordering history of a destination. The cache period is sufficiently long that all nodes that might have had active routes must have timed out, so no node in the network has an active route through a successor that erases its ordering history.

Rule 2 (Smaller Advertisement Condition (SAC)): A node A may accept an advertisement a for destination D if

$$\mathrm{ord}(a) < ao^A_D \tag{2}$$

■

Rule 3 (Internal Ordering Condition (IOC)): A node must always maintain its internal state such that its advertised label is in-order to all in-use successor labels. This implies that if a node wishes to transmit an advertisement for a route, it must ensure that all successors are in-order for the new advertised label. The node may need to drop certain successors that do not satisfy the ordering.

Let node A transmit an advertisement a for destination D, then:

$$\mathrm{ord}(a) > \max\{so^A_{D,x} | x \in S^A_D\} \tag{3}$$

■

Procedure 1 (Node Initialization): When a node boots, its successor table and link cost tables are empty. As a node discovers neighbors, it must negotiate and coordinate link costs between itself and neighbors through an external process. DOS does not use distance – and thus link costs – to maintain loop-free paths. It does propagate distances as a route metric to help nodes choose among multiple loop-free paths. When a node boots (or reboots), or when a node is subject to losses in the routing state for a given destination, it must use some method to avoid assuming an obsolete routing state (i.e., assume a new advertised ordering for the destination that increases with respect to a prior value of its advertised ordering) in order to ensure that loops do not form. The specific method may be based on real-time clocks, hold-down methods (e.g., those proposed in AODV [2, sec 6.13]), or internodal exchanges similar to those used for the reset of sequence numbers in topology-broadcast protocols.

■

Procedure 2 (Initiate RREQ): A node initiating a RREQ inserts its current advertised ordering in to the RREQ and broadcasts it with a TTL appropriate to the type of expanding ring search used in the network.

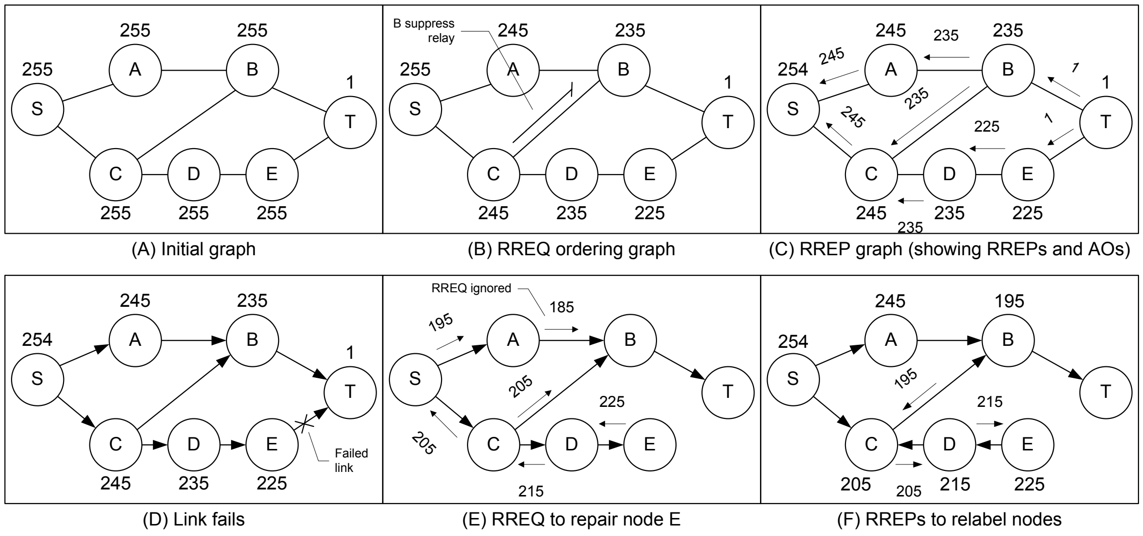

Fig. 1. Graph labeling examples

A node A originating a RREQ q for destination D sets

$$\mathrm{ord}(q) \leftarrow ao_D^A. \tag{4}$$

■

Procedure 3 (Receive RREQ): Let node A receive a RREQ q for destination D originated by node S from last-hop B. Node A ensures that the RREQ path is acyclic. Using traditional RREQ IDs, this means that the RREQ is unique. If the RREQ fails the test, node A silently drops the RREQ. Otherwise node A creates a cache entry and stores the tuple $\{S, D, \mathrm{ord}(q), B\}$. A node should maintain the cache information long enough for a RREQ flood to terminate.

If node A may send a RREP for q (Eq. 7), it should reply as per procedure 5. Otherwise, node A should relay the RREQ as per procedure 4, considering TTL limitations. Because the GSN is not source-specific, if a node A recently transmitted another route request q' that would satisfy q, then node A may suppress relaying q. A request q' satisfies q if and only if it satisfies Eqs. 5 – 6. ■

Procedure 4 (Relay RREQ): When a node A relays a RREQ for destination D, it must ensure that the ordering of the new RREQ, q', is sufficiently small that any solicited RREP may satisfy both node A and all predecessors. Therefore, node A must choose an ordering $\mathrm{ord}(q')$ such that:

$$\mathrm{ord}(q') < \mathrm{ord}(q) \tag{5}$$

$$\mathrm{ord}(q') \leq ao_D^A \tag{6}$$

A simple choice that satisfies these bounds is $\mathrm{ord}(q') \leftarrow \min\{\mathrm{ord}(q) - k, ao_D^A\}$. The constant $k > 0$ is a spacing interval between successive hops to allow some slack in the ordering of nodes. It allows a few new nodes to join paths without forcing a route request travel extra hops.

■

Procedure 5 (Initiate RREP): A node A receiving a RREQ q for destination D may send a RREP if

$$\exists\ g \in \mathcal{G},\ x \in S_D^A : so_{D,x}^A < g < \mathrm{ord}(q) \tag{7}$$

If such a g exists, node A must choose a specific ordering $g^\star$ such that $g^\star$ is the maximum ordering that satisfies Eq. 7 and NIAO. If using a k-skip spacing between nodes, $g^\star$ should include as much skip space, up to k, as possible while satisfying the bounds of Eq. 7. Upon choosing the ordering $g^\star$, node A sets

$$ao_D^A \leftarrow g^\star \tag{8}$$

and transmits the RREP to the RREQ last-hop. ■

A node answering a RREQ for itself could choose to advertise the 1 label. This choice is admissible by Eq. 7 and satisfies IOC (a choice of 0 violates both). By having a node advertise itself with the constant minimum possible label, it gives all one-hop neighbors maximum flexibility in choosing new labels and answering queries on behalf of the destination. All other nodes must choose the maximum $g^\star$ to preserve the sequence number space over time.

Procedure 6 (Receive RREP): Node A receiving RREP r from last-hop B for destination D may accept the RREP if it is feasible by SAC. If so, node A may add B to its successor set S_D^A and cache $so_{D,B}^A \leftarrow \mathrm{ord}(r)$. If node A is not the terminus of the RREP, node A may issue a new RREP for D as per procedure 5 and send it to any and all RREQ last-hops found from the RREQ cache that are satisfied by the new route. Any such satisfied cache entries should be marked "satisfied" or entirely removed to prevent future RREP generation. ■

The choice of how many RREPs to relay based on cached information can significantly affect protocol overhead. In our implementation, which has promiscuous overhearing of control packets, we use two rules: (a) Send at most one unicast

RREP per RREQ origin and per last-hop; and (b) choose (randomly) between last-hops based on minimum RREQ distance. Thus a relay node R, for a given RREQ origin, say A, where a RREQ was received from last-hops B and C, the relay node would send at most one RREP. It would choose between last-hops B and C based on the RREQ distance seen from each last-hop. If node R also satisfied another origin Z via last-hops B and X, it would choose the minimum distance last-hop, even though node B would be a minimum-overhead choice. When using promiscuous overhearing, we also use the rule that a node will not relay a RREP unless it is the unicast destination of the RREP.

Example 1 (Graph initialization): Fig. 1(A) shows an initial network for destination T. In this example, we assume an 8-bit sequence number with a k-skip value of 10. In Fig. 1(B), node S initiates a RREQ for T with a requested ordering of 255. Nodes A and C, not having a valid route, relay the request with a new requested ordering of 245. In the example, node B processes node A's copy of the RREQ before node C's copy. Node B relays the request from A, but also caches C as a RREQ predecessor with requested ordering 245. Once node T receives RREQs, it begins sending RREPs. In Fig. 1(C), node T always replies with an advertised ordering (AO) of 1. Nodes B and E, when they relay the replies, choose a maximum feasible ordering. In this case, it is 235 for node B and 225 for node E, which preserves the per-hop skip space. Node B sends a RREP to both nodes A and C because the new route satisfies both RREQs (one could use other multiple last-hop relaying rules like those described above). Nodes A and C would relay the RREPs to node S to complete the RREP operation. Later, when node C receives the RREP from node D, node C may use next-hops B and D paths as unequal cost multipath.

Example 2 (Graph re-labeling): In Fig. 1(D), node E looses its link to the destination T and initiates a RREQ flood. The flood progresses as shown in Fig. 1(E). Note that nodes relaying a RREQ do not break their successor links, so nodes C and D maintain their links through E until such time as E transmits a RERR messages (the use of RERRs is beyond the scope of this example). Node B receives two copies of the RREQ. The first copy has ordering 205 and the second copy has ordering 185. Node B is able to reply to node C from cache because $so^B_{T,T} = 1$. Node B chooses $g^\star = 195$ and sends a RREP to node C. Because node B responds from cache to answer the 205 query from C, it never relayed that query. This means that node B does not recognize the 185 as cyclic, so node B may choose to respond to it. Using the above rules for multiple RREPs, node B would choose not to reply to the copy from node A because the RREP path length is longer than the RREQ received from node C. Fig 1(F) shows the final successor graph with new node labels. Only nodes B, C, D and E had to change their labels to repair the route.

III. Analysis

The loop-free ordering of nodes is based on the advertised ordering, ao^A_D, of node A for destination D. We must show that the directed successor graph for D is acyclic at all times. The successor information $so^A_{D,x}$ plays a role in ensuring loop-freedom, but it is only a means to ensure that advertised orderings maintain order.

We prove several properties of DOS under the assumption that either nodes never forget their advertised ordering for a given destination, or implement a fail-safe method to ensure that no node increases its advertised ordering for a given destination after losing its routing state for the destination. A node executing DOS maintains at all times predecessor ordering. This means that any changes a node makes to its own advertised ordering never bring a predecessor out of order. We prove that a node executing DOS maintains successor ordering. This means that any changes a node makes to its successor graph or to its own advertised ordering maintain the global ordering of the network. Combining these properties with IOC is sufficient to show that DOS maintains the global ordering of the network at all times, and is thus loop-free at all times. We show that DOS is lockout-free; that is, given multiple concurrent route request and route reply operations for the same destination, every origin of a route request is guaranteed to find a route.

Theorem 1 (Predecessor Ordering): If every node obeys NIAO, SAC, and IOC, then a node i choosing a new advertised ordering $ao^i_D(t)$ at time t maintains ordering with all predecessors. That is, in an existing successor graph for destination D, node i maintains

$$ao^i_d(t) < \min\left\{ ao^j_d(t) \,\middle|\, \forall\, j : i \in S^j_d \right\} \tag{9}$$

Proof: The premise of "an existing successor graph" implies that at some time t_0 node i transmitted an advertisement $a(t_0)$ which built a predecessor link between some predecessor j and i. Let node j process $a(t_0)$ at time t_1 and create the edge (j, i). At some other time $t > t_0$, node i changes its advertised ordering. We show that node i maintains Eq. 9 by showing it maintains the bound for an arbitrary predecessor.

By premise of existing predecessor

$$i \in S^j_D \tag{10}$$

By SAC

$$i \in S^j_D \Rightarrow \operatorname{ord}\left(a(t_0)\right) < ao^j_D(t_1) \tag{11}$$

By NIAO, $\forall\, t < t_1$

$$ao^j_D(t_1) \le ao^j_D(t < t_1) \tag{12}$$

By Eqs. 11 and 12

$$\operatorname{ord}\left(a(t_0)\right) < ao^j_D(t < t_1) \tag{13}$$

By NIAO at time t

$$ao^i_D(t_0 < t) \le ao^i_D(t_0) \tag{14}$$

$$\Rightarrow ao^i_D(t_0 < t) \le \operatorname{ord}\left(a(t_0)\right) \tag{15}$$

By Eqs. 13, 15, and $t_1 > t_0$

$$ao^i_D(t_0 < t < t_1) < ao^j_D(t) \tag{16}$$

By IOC at j and Eq. 10

$$\text{ord}\left(a(t_0)\right) < ao^j_D(t_1 \le t) \tag{17}$$

By Eqs. 15, 17

$$ao^i_D(t_1 \le t) < ao^j_D(t) \tag{18}$$

By Eqs. 16, 18

$$ao^i_D(t_0 < t) < ao^j_D(t) \tag{19}$$

The result in Eq. 19 shows that any new advertised ordering of node i after time t_0 must be less than the ordering at any existing predecessor j. The key elements to the proof are that advertised orderings never increase (NIAO) and that once a predecessor accepts a successor, it must maintain its label greater than the successor's advertised ordering (IOC, Eq. 17). ■

Theorem 2 (Successor Ordering): Without creating a routing table loop, node i may accept advertisement a, if the advertisement satisfies SAC: $\text{ord}\left(a(t_0)\right) < ao^i_D$.

Proof: Let node j have a loop-free path to destination D and be the issuer of a. At time t_0, node j sets its advertised label to $ao^j_D(t_0)$ and $\text{ord}\left(a(t_0)\right) \leftarrow ao^j_D(t_0)$, then transmits a. At time t_1, node i processes the advertisement a. For node i to accept a, by SAC, $\text{ord}\left(a(t_0)\right) < ao^i_D(t_1)$, so $ao^j_D(t_0) < ao^i_D(t_1)$. By NIAO, $ao^j_D(t_1) \le ao^j_D(t_0)$, so $ao^j_D(t_1) < ao^i_D(t_1)$. ■

It may be the case that node i is unlabeled, in which case it is considered to have the ∞ label. If node i later in turn advertises the route to D, it must by IOC, ensure the new advertised label maintains order with $\text{ord}\left(a(t_0)\right)$, or it must drop the route through j.

Theorem 3 (Loop-free paths): As an advertisement propagates hop-by-hop, the resulting path is loop-free.

Proof: From Theorem 2, we know that each independent per-hop decision maintains local loop-freedom. We wish to show that over time and multiple hops, a constructed path is loop-free. We proceed by induction over the number of hops. Let the RREP path be from node v_0 to node v_k. Let node v_0 be the destination. Denote the time at which node v_i receives the route advertisement as t^{in}_i and the time at which node v_i advertises the route as time t^{out}_i.

By Theorem 2, we know in the base case, where v_1 relays the advertisement from node v_0, it holds that $ao^{v_0}_{v_0}(t^{\text{out}}_1) < ao^{v_1}_{v_0}(t^{\text{out}}_1)$. By strong induction, assume that node i has a loop-free path satisfying $ao^{v_0}_{v_0}(t^{\text{in}}_i) < \cdots < ao^{v_i}_{v_0}(t^{\text{in}}_i)$. By Theorem 2, we have $ao^{v_i}_{v_0}(t^{\text{in}}_{i+1}) < ao^{v_{i+1}}_{v_0}(t^{\text{in}}_{i+1})$. By NIAO and IOC, $ao^{v_0}_{v_0}(t^{\text{in}}_{i+1}) \le ao^{v_0}_{v_0}(t^{\text{in}}_i) < \cdots < ao^{v_i}_{v_0}(t^{\text{in}}_i) \le ao^{v_i}_{v_0}(t^{\text{in}}_{i+1})$. Therefore the final path is $ao^{v_0}_{v_0}(t^{\text{in}}_{i+1}) < \cdots < ao^{v_i}_{v_0}(t^{\text{in}}_{i+1}) < ao^{v_{i+1}}_{v_0}(t^{\text{in}}_{i+1})$. We maintain a strict topological order, so the graph is acyclic. ■

Theorem 4 (Loop-freedom): DOS is loop-free at all times.

Proof: By Theorem 3, all successor paths are in-order at all times. By Rule IOC, a node always maintains its own label in-order to successor order. By Theorem 1, a node always maintains predecessors in-order. Thus, the network graph is maintained in-order at all times, so the graph is acyclic. ■

In addition to correct (loop-free) operation, we show that DOS is lockout-free. As applied to routing, lockout-free means that given a connected lossless network, any given node will find a route (make progress) over any possible node labels regardless of how many other nodes are trying to find the same route. There are no "black holes". Possible node labels are those reachable through the application of DOS's routing rules. We first show that a single route request, route reply operation finds a route. We then show that DOS is lockout-free even when there are multiple concurrent request/reply computations for the same destination. We assume that the network is connected, lossless, and has stable links.

Theorem 5 (Proc. 5 correctness): A RREP generated by a node satisfying Proc. 5 is feasible for the RREQ last-hop.

Proof: Let node A receive from last-hop node B a RREQ q with ordering $\text{ord}(q)$. By Procedure 5, the advertised ordering of node A will be $ao^A_D \leftarrow g^\star$, which must be less than the requested ordering: $ao^A_D < \text{ord}(q)$. The reply is feasible by SAC. ■

Theorem 6 (Minimum label relaying): If a node relaying a RREQ requests an ordering that satisfies Eq. 5 – 6 in Procedure 4, then any solicited RREP is feasible for the relay node and the RREQ reverse path.

Proof: Consider a request traversing the path $\{v_k, \ldots, v_0\}$. As before, assume that a node without a label uses the ∞ label. Let node v_i issue a RREQ q_i with label $\text{ord}(q_i)$. Node v_{i-1} must provide a RREP r_{i-1} with label $\text{ord}(r_{i-1})$ such that $\text{ord}(r_{i-1}) < \text{ord}(q_i)$.

The proof is by induction on the hops the RREP travels from node v_0. The base case is to show that the RREP received by node v_1 is feasible at node v_2, establishing the first relay. We then show that a RREP received by node v_i is feasible at node v_{i+1}.

The proof that the RREP r_0 sent by node v_0 is feasible at node v_2 follows in Eqs. 20 – 25. Eq. 24 ties the RREP r_0 to r_1 because node v_1 would create its route entry based on r_0. The proof shows that the RREP r_1 is in-order with respect to the RREQ q_2 sent by node v_2. Therefore, the RREP r_1 is feasible at node v_2.

By Procedure 4

$$\text{ord}(q_1) < \text{ord}(q_2) \tag{20}$$

By Theorem 5

$$\text{ord}(r_0) < \text{ord}(q_1) \tag{21}$$

By Procedure 6

$$so^{v_1}_{v_0,v_0} \leftarrow \text{ord}(r_0) \tag{22}$$

Therefore

$$\text{ord}(r_0) < \text{ord}(q_1) < \text{ord}(q_2) \tag{23}$$

$$\Rightarrow \exists\, g, x : so^{v_1}_{v_0,x} < g < \text{ord}(q_2) \tag{24}$$

By Procedure 5

$$\text{ord}(r_1) < \text{ord}(q_2) \tag{25}$$

In the inductive step, we assume that node v_{i-1} sends a RREP r_{i-1} to node v_i and node v_i relays RREP r_i to node v_{i+1}. We must show that $\text{ord}(r_i) < \text{ord}(q_{i+1})$. The proof is identical to the case for node v_1 relaying r_0 to v_2. ■

Theorem 7 (Lockout-freedom): DOS is lockout-free in a connected lossless network for multiple concurrent route requests for the same destination. Every node that originates a route request will receive a feasible route reply.

Proof: For there to be lockout, the paths of two replies must cross at one or more intermediary nodes. For lockout to occur, the first reply to cross such an intermediary node must change the state at that node such that (1) it cannot accept the second reply and such that (2) the intermediary node cannot generate its own reply for the second origin. We proceed through a proof by contradiction, showing that conditions (1) and (2) cannot be simultaneously true.

Consider an intermediary node v that receives and relays query q_1 from predecessor w_1 and query q_2 from predecessor w_2. The predecessors node w_1 and node w_2 could be equivalent or distinct. The relayed queries are q_1' and q_2'. Without loss of generality, assume that the reply r_1 to q_1' arrives first at time t_1, followed by the reply r_2 to q_2' at time t_2.

For condition (1) to be true, we must have $ao_d^v(t_2) \leq \mathrm{ord}(r_2)$, so reply r_2 is not feasible at node v. However, by Theorem 6 $\mathrm{ord}(r_2) < \mathrm{ord}(q_2')$, so $ao_d^v(t_2) < \mathrm{ord}(q_2')$. By Procedure 4, $\mathrm{ord}(q_2') < \mathrm{ord}(q_2)$, so $ao_d^v(t_2) < \mathrm{ord}(q_2)$. This contradicts condition (2) because node v could generate a feasible reply to query q_2.

For condition (2) to be true $\mathrm{ord}(q_2) \leq ao_d^v(t_2)$. By Proc. 4, $\mathrm{ord}(q_2') < \mathrm{ord}(q_2)$, so $\mathrm{ord}(q_2') < ao_d^v(t_2)$. By Theorem 5, $\mathrm{ord}(r_2) < \mathrm{ord}(q_2')$, so $\mathrm{ord}(r_2) < ao_d^v(t_2)$. This contradicts condition (1), because reply r_2 would be feasible at node v. ■

In a network with failures, it is possible for conditions (1) and (2) of Theorem 7 to be simultaneously true. Node v could receive reply r_1 at time t_1, then lose the route, then receive r_2 a time t_2. At time t_2, reply r_2 might not be feasible at v, and v have no active route to the destination. It is a general problem of networks that if the network changes faster than the control messages, routing may not converge.

IV. DOS Implementation

In our implementation of DOS, we use several optimizations. Some of these optimizations are also found in the NS2 implementation of DSR and AODV. We use link-layer loss detection, so if a unicast packet is dropped by the MAC, the network layer may re-transmit the packet. The network layer may also manipulate the link-layer queue to remove or re-queue packets. At the link-layer, we queue at most one packet. All other queueing is done at the network layer in per-destination queues. Packets are classified by priority, which are, in order, ARP, DOS, CBR. ARP packets do not exist at the network layer, but the same priority scheme would apply to packets at the link layer if we queued more than one packet at that layer. Per-class, we permit up to 50 packets over all destinations (this is slightly less queuing capacity as found in the DSR and AODV implementations). The major advantage of this configuration is that the next-hop determination is deferred until just before packet transmission. In DSR and AODV implementations, the routing protocol makes a next-hop determination, then releases many packets to the link-layer without any assurance that the next-hop will be valid by the time the packet arrives at the air interface. We do not use "local repair". If an intermediate node has a foreign packet and no route to the destination, it will broadcast a RERR and drop the foreign packet. In the RREP process, a node will not add a successor to the routing table until it has a link-layer MAC address for the next-hop. If DOS does not see a MAC-layer ARP entry, it will send a unicast ECHO (new control packet) to the next hop, at no more than 1 echo per 3 seconds per next-hop. In the RREQ process, a node will use an initial TTL of 2, a re-try TTL of 6, and then up to three network-wide floods (i.e., TTL of 30). If a node fails the route discovery after three network-wide floods, the node will put a RREQ hold down in place to prevent initiating a RREQ for the failed destination for 3 seconds. The RREQ process is otherwise as described above. Nodes will cache a route for up to 10s without use before timing out the route. DOS allows control packet aggregation for packets destined to the same next-hop (or broadcast address). The implementation will scan the per-destination packet queues and aggregate any control packets for the same destination, up to the maximum UDP packet size. DOS, like DSR, uses promiscuous mode over-hearing of RREPs to build up larger route caches. Promiscuous mode is purely an optimization for building a route-cache and the protocol works correctly without promiscuous mode.

Algorithm 1:

```
PeriodicLinkQuality($N, w$)
(1)   $uses \leftarrow N.last\_uses + N.current\_uses$
(2)   $loss \leftarrow N.last\_loss + N.current\_loss$
(3)   $uses \leftarrow \max\{uses, loss\}$
(4)   if $uses > 0$
(5)     $newquality \leftarrow (uses - loss)/uses$
(6)   else
(7)     $newquality \leftarrow 1.0$
(8)   $quality \leftarrow w * newquality + (1 - w) * N.quality$
(9)   return $quality$
```

Algorithm 2:

```
InstantLinkQuality($N, w$)
(1)   $uses \leftarrow N.last\_uses + N.current\_uses$
(2)   $loss \leftarrow N.last\_loss + N.current\_loss$
(3)   $uses \leftarrow \max\{uses, loss\}$
(4)   if $uses > 1$
(5)     $quality \leftarrow w * N.quality + (1 - w) * (uses - loss)/uses$
(6)   else
(7)     $quality \leftarrow 1.0$
(8)   return $quality$
```

DOS uses link-quality measurements per next hop. The link quality measurement at the network layer is based on the number of packets forwarded to each next hop and the number of packet drops (after MAC retries) per next hop. Alg. 1 is executed periodically and tends to pull link quality up to 1.0 in the absence of errors. Alg. 2 is executed per packet loss and reduces link quality. The link-quality for node

N is measured as a moving average over 1-second buckets as per Alg. 1 with a weight of 0.75. This weights long-term link quality towards the historical value. We smooth the data over the current 1-second bucket and the previous 1-second bucket to reduce boundary effects where a packet is transmitted in one bucket and lost in the next bucket. Each link begins with a link quality of 1.0. Whenever there is a packet loss, as detected by the link-layer feedback, DOS computes an instantaneous link-quality as per Alg. 2 with a weight of 0.4. This weights the instantaneous link-quality towards the current value. The variables $last\_uses$ and $current\_uses$ are the number of packets forwarded to a given next-hop in the last (current) time bucket. The variables $last\_loss$ and $current\_loss$ are the number of packets dropped after 802.11 retries for a given next-hop in the last (current) time bucket. If the returned $quality$ from Alg. 2 is less than a global threshold LQ_THRESH, then the next-hop is considered down and removed from the forwarding table. LQ_THRESH begins at 0.85. As a node initiates more RREQs, the bound is lowered, allowing lower quality links. Over time and as there are more link-layer drops, the bound is raised, back towards the target 0.85 level. We impose a hard floor of 0.7 on LQ_THRESH.

DOS uses a link-quality weighted, minimum distance multipath. Over all multipaths of minimum distance, DOS will randomly distribute packets over next-hops in proportion to their link quality.

V. Simulation

We present random-waypoint simulations using the NS2 [19] v2.28 simulator. The simulations compare performance between DOS, DSR [5], AODV [2] and OLSR [3]. We used DSR and AODV with the default NS2 configuration. The OLSR implementation is from the INRIA NS v2.1b7 implementation of OLSRv3 [20], ported to NS v2.28.

The performance of DOS benefits from the implementation choices described in Sec. IV. In particular, not using a persistent local repair results in significantly higher delivery ratio because there is much lower network load. Using the described RREQ hold down significantly reduces the network load, but is not a major contributor to latency due to two reasons. Because there is no local repair, the only packets that are delayed due to pending RREQs are packets queued at the source node, so the statistical contribution of that delay is small compared to the total number of packets in flight. A RREQ operation is an amplification of a single data packet to many control packets, so by holding down a small number of sources that have persistent RREQ failures, there is a large savings in the amplified control packets. In a real-world application, hold down might not be desirable, but real-world applications should be responsive to ICMP destination unreachable packets, which are not used in simulation and do not throttle sources. The use of unicast ECHO packets to learn an ARP address before committing a data packet to the network is partially a NS2 hack to work around the 1-packet ARP queue, though real-world network stacks will also drop packets if there are too many outstanding ARP requests.

The channel is an 802.11 MAC at 914 MHz with 2 Mbps bandwidth. The 50-node simulations are over a 1500m x 300m rectangle and the 100-node simulations are over a 2200m x 600m rectangle. Mobility patterns come from the NS2 bundled "setdest" program. Traffic patterns are 512-byte CBR flows, generated by the NS2 bundled "cbrgen.tcl" program for 10-sources and 30-sources. Simulations last for 900s, and we use pause times of 0s (no pauses), 100s, 300s, 500s, 700s, and 900s (no mobility). For each configuration (number nodes, sources, pause time), we generated 10 trials with different random number seeds. The figures below show the mean performance and the 95% confidence interval, assuming a normal distribution. If two data points have overlapping confidence intervals, we say that the two points are statistically equivalent.

The metrics we use are delivery ratio, latency, network load, and loop ratio. The delivery ratio is the number of CBR packets received by the destination nodes divided by the number of CBR packets sent by the source nodes. The latency is the one-way end-to-end delay between when a source generates a CBR packet and the destination receives it. The network load is the total number of network-layer control packets (e.g. RREQ, RREP, RERR) divided by the total number of CBR packets received at the destinations. The loop ratio is the total number of duplicate hops divided by the total number of CBR packets sent by the sources.

The intuition behind the loop-ratio metric is that a loop ratio of 1.0 means that on average, each CBR packet loops through one duplicate node someplace along its path. A loop ratio of 0.0 means that a packet never traversed the same node twice. An obvious question is why do "loop-free" protocols have loops? It is caused by packet queueing. The term "loop-free" means that routing tables never point in a directed cycle at any given instant. During the lifetime of a packet, while queued at intermediate nodes, the routing topology may change and a packet may find itself re-visiting a node.

In summary, DOS and AODV have approximately the same delivery ratio and latency. At times, DOS has a better delivery ratio and at times, AODV has a better end-to-end latency. DOS has a significantly lower network load, on the order of 1/2 to 1/5 the load of AODV. DSR, in most cases, performs worse than AODV or DOS, except in low mobility cases. In terms of packet loops, DOS is about 1/2 to 1/10 the loop ratio of AODV, and about 1/1000 the loop ratio of DSR. Compared to OLSR, DOS has a better delivery ratio, especially with high mobility. DOS's network load is much lower than OLSR, at times by an order of magnitude. OLSR suffers a large loop ratio, similar to DSR. In a few cases, OLSR has a lower end-to-end latency, but it is not consistent. We have also repeated the simulations for DOS without multipath and without using DOS link-quality measurements. Removing these features does not change the overall picture, but it does narrow the difference in network load.

The delivery ratio is shown in Figs. 2, 3, 8, and 9. At low load (Fig. 2), DOS, DSR, and AODV, have equivalent delivery ratios. At mid load (Fig. 3 and 8), DOS and AODV have equivalent delivery ratio, but DSR is significantly lower except in low mobility (high pause time) cases. At high load (Fig. 9), DOS has the best delivery ratio at high mobility (0s and 100s pause time), but is otherwise tied with AODV. The OLSR delivery ratio is particularly bad in the 10-source scenarios, and is similar to AODV is the 30-source scenarios.

The network load is shown in Figs. 4, 5, 10, and 11. At low load (Fig. 4), DSR and AODV have equivalent network loads. DOS has a lower load at high mobility (300s and lower pause time), and is equivalent to DSR at lower mobility (500s and above pause time). At 50-node mid load (Fig. 5, 50-node 30 source), DOS has about 1/2 to 1/5 the load of DSR. DOS has consistently about 1/6 the load of AODV at all mobility. At 100-node mid load (Fig. 10, 100-node 10 source), DOS has about 1/2 to 1/10 the load of AODV, and about 1/10 to 1/40 the load of DSR. At high load (Fig. 11, 100-node 30 source), DOS has about 1/6 the load of AODV over all mobility ranges, and about 1/2 to 1/20 the load of DSR. As one would expect, OLSR has a fairly consistent and high network load, because it mostly depends on the number of nodes. The OLSR load can be up to 10x higher than DOS in the 10-source scenarios and about 1.5x to 2x higher in the 30-source scenarios.

The end-to-end latency is shown in Figs. 6, 7, 12, and 13. In the 10-source scenarios (Figs. 6 and 12), DOS and AODV have equivalent latency. With mobility, DOS is about 1/2 to 1/10 the latency of DSR. Without mobility (900s pause), DSR drops to statistically equivalent latency. In the 30-source scenarios (Figs. 7 and 13), DOS and AODV have equivalent latency with high mobility, but as the pause time increases DOS grows to about twice the latency of AODV. DOS is about 1/2 to 1/10 the latency of DSR over all mobility patterns. OLSR has a similar latency to DOS except in a handful of cases. In the 50-node, 10-source (Fig. 6) scenarios, OLSR has a significantly higher latency at high mobility. In the 100-node, 10-source (Fig. 12) scenarios, OLSR has a significantly lower latency with slow mobility.

The loop ratio is shown in Figs. 14, 15, 16, and 17. With a few exceptions, DSR is around 0.1 loop ratio. This means that on average one in ten packets loop through a single node once. In a couple cases, it is closer to 0.2 and in a couple cases it is down to around 0.05. In exactly one case (low load Fig. 14, 900s pause), DSR is around 10^{-4} (one packet in 10,000 loops through one node). AODV is generally around 10^{-3}, though in one case (low load Fig. 14, 900s pause), it drops to the order of 10^{-5}. DOS is generally between 10^{-5} and 10^{-4}, though in one case (low load Fig. 14, 900s pause), it is exactly zero. OLSR has a high loop ratio around 0.1, which is similar to DSR.

VI. Conclusion

We have presented the Distributed Ordered Sequences (DOS) routing protocol. The key feature of this on-demand ad hoc routing protocol is that it uses a global sequence number per destination that is a non-increasing number at each intermediary node. The route request and route reply processes manipulate the sequence number to maintain the advertised order of each node in a strict topological order, and thus ensure that DOS is loop-free at every instant.

Simulation results in NS2 show that in general DOS performs as well as or better than AODV, DSR and OLSR in terms of packet delivery ratio and data latency. DOS has a much lower network load than AODV, DSR or OLSR. It is usually at least 1/2 the load of those protocols, and at times may be up to 1/40 the load of DSR. We also measured the number of packet loops, which in protocols such as AODV and DOS are caused by packet queuing during link failures. We find that DOS has about 1/10 the number of loops of AODV and about 1/1000 the number of loops of DSR or OLSR. In DOS, approximately one packet it 10,000 loops through a single node once.

Future work includes investigating efficient schemes to ensure that no node increases its advertised ordering for a destination after losing its routing state. We also wish to look at models for distributing packets over an unequal multipath that considers link quality, path quality, and path distance. Another area is applying constraint-based routing, such as delay or path energy. We also think that work may be borrowed from dominating set broadcast distribution and applied to the RREP reverse path problem, where a node must send multiple RREPs to multiple last-hops to satisfy multiple outstanding RREQs.

References

[1] ASTM International, "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems – 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2003.

[2] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[3] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[4] R. Ogier, F. Templin, and M. Lewis, "RFC 3684: Topology dissemination based on reverse-path forwarding (TBRPF)," Feb. 2004.

[5] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, vol. 353.

[6] I. Chakeres, E. Belding-Royer, and C. Perkins, "Dynamic MANET On-demand (DYMO) Routing," June 2005. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-02.txt

[7] J. Jaffe and F. Moss, "A responsive distributed routing algorithm for computer networks," *IEEE Transactions on Communications*, vol. COM-30, no. 7, pp. 1758–62, July 1982.

[8] J. J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 130–41, Feb. 1993.

[9] E. W. Dijkstra and C. S. Scholten, "Termination detection for diffusing computations," *Information Processing Letters*, vol. 11, no. 1, pp. 1–4, Aug. 1980.

[10] J. J. Garcia-Luna-Aceves and S. Murthy, "A path finding algorithm for loop-free routing," *IEEE/ACM Trans. Networking*, Feb. 1997.

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

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

[13] E. M. Gafni and D. P. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Comm.*, vol. COM-29, no. 1, pp. 11–18, Jan. 1981.

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

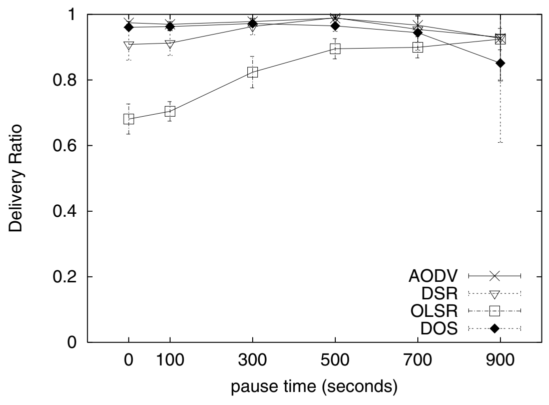

Fig. 2. Delivery ratio 50-nodes, 10-source

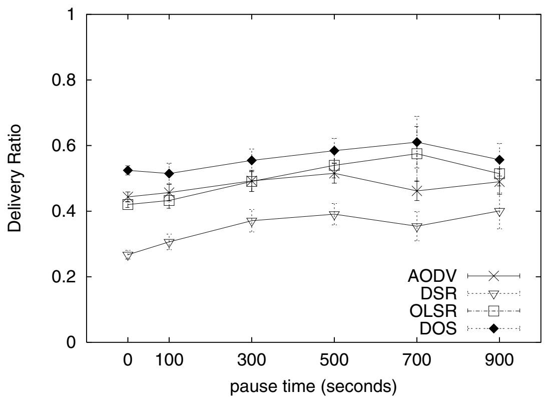

Fig. 3. Delivery ratio 50-nodes, 30-source

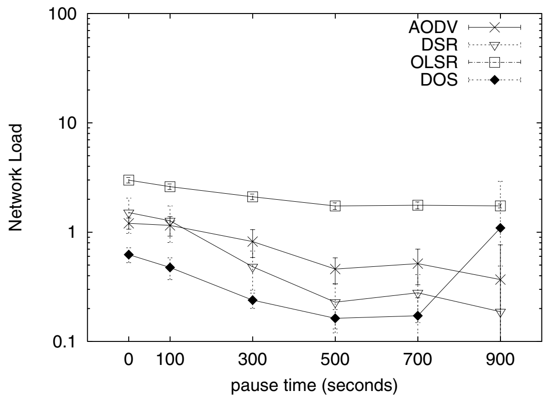

Fig. 4. Control 50-nodes, 10-source

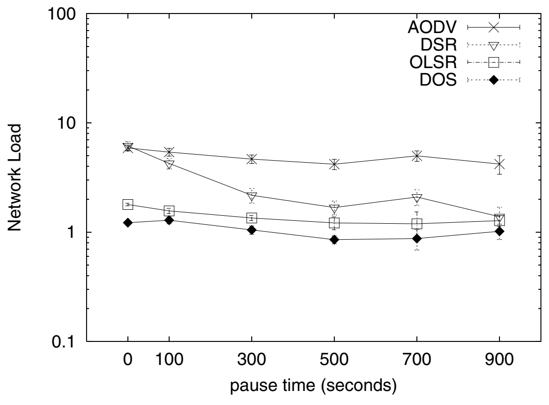

Fig. 5. Control 50-nodes, 30-source

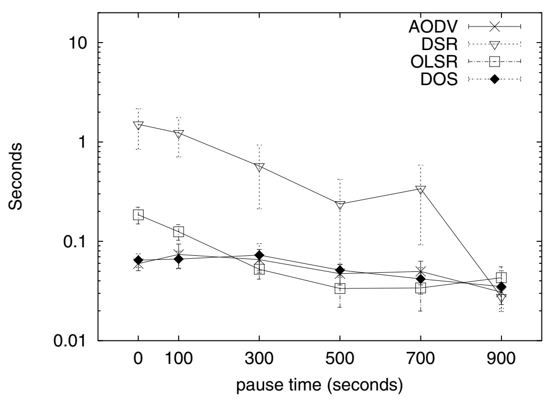

Fig. 6. Latency 50-nodes, 10-source

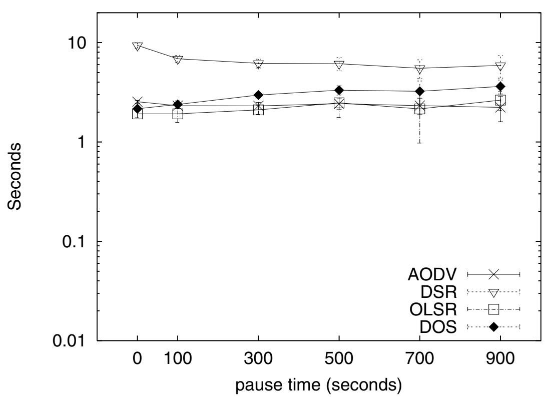

Fig. 7. Latency 50-nodes, 30-source

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

[16] M. Mosko and J. J. Garcia-Luna-Aceves, "Loop-free routing using a dense label set in wireless networks," in *ICDCS 2004*, Mar. 2004.

[17] B. S. Schröder, *Ordered Sets: an introduction*. Boston: Birkhäuser, 2003.

[18] S.-J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *IEEE ICC*, 2001, pp. 3201–3205.

[19] K. e. Fall and K. e. Varadhan, "The ns manual," 2003, http://www.isi.edu/nsnam/ns/doc/index.html.

[20] L. Christensen and G. Hansen, "INRIA OLSR draft 3 ns2 patch," 2001, http://hipercom.inria.fr/olsr/olsrpatch.diff.gz.

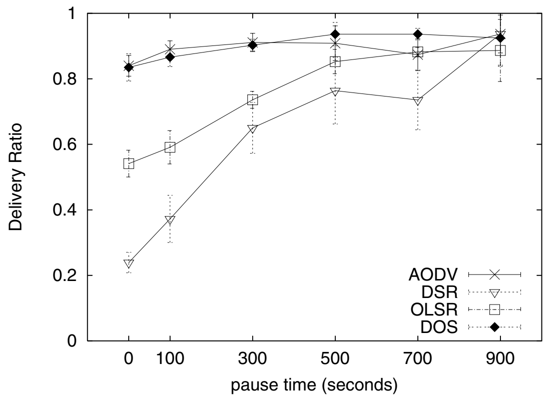

Fig. 8. Delivery ratio 100-nodes, 10-source

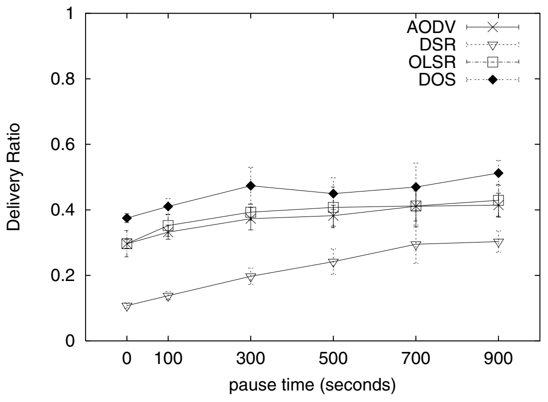

Fig. 9. Delivery ratio 100-nodes, 30-source

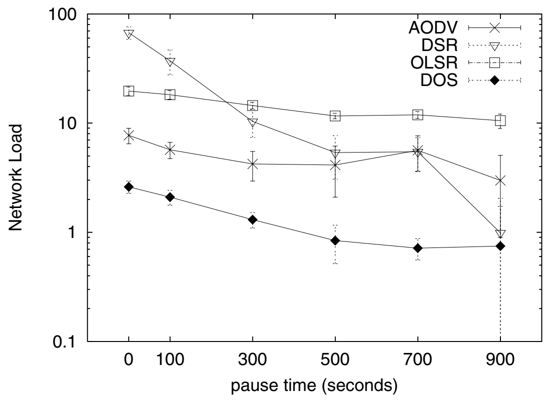

Fig. 10. Control 100-nodes, 10-source

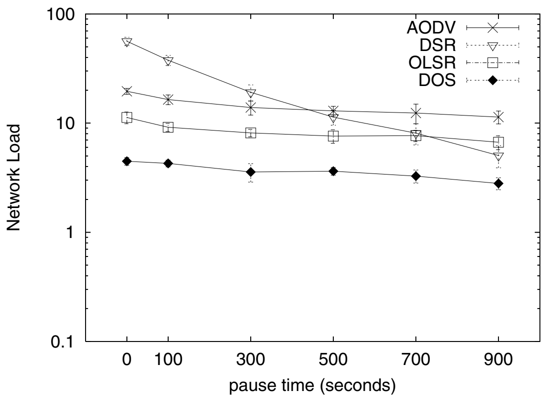

Fig. 11. Control 100-nodes, 30-source

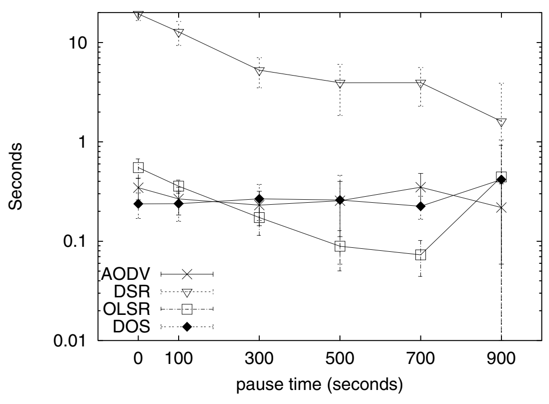

Fig. 12. Latency 100-nodes, 10-source

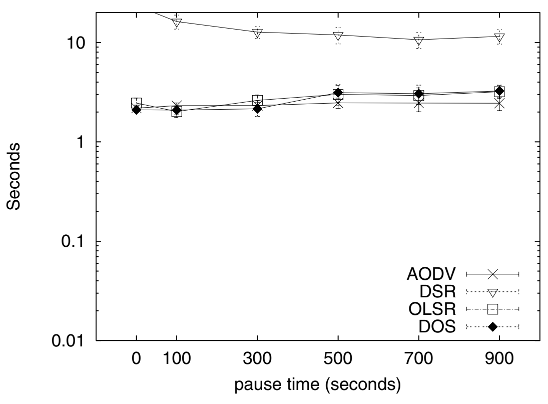

Fig. 13. Latency 100-nodes, 30-source

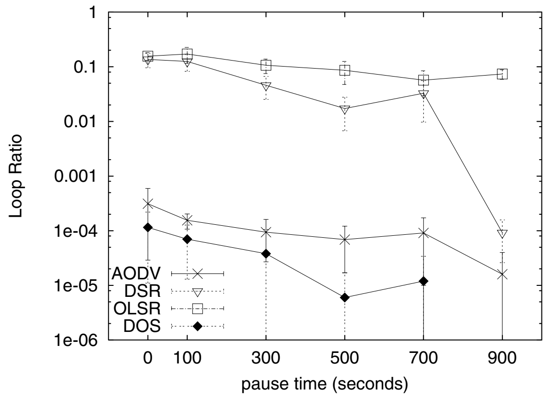

Fig. 14. Loop ratio 50-nodes, 10-source

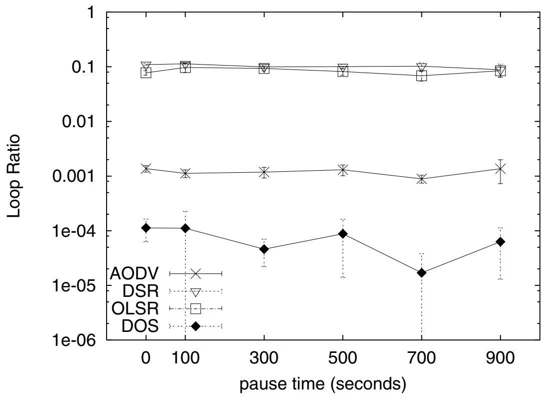

Fig. 15. Loop ratio 50-nodes, 30-source

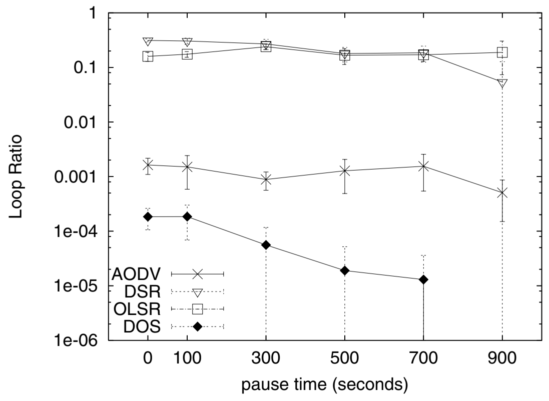

Fig. 16. Loop ratio 100-nodes, 10-source

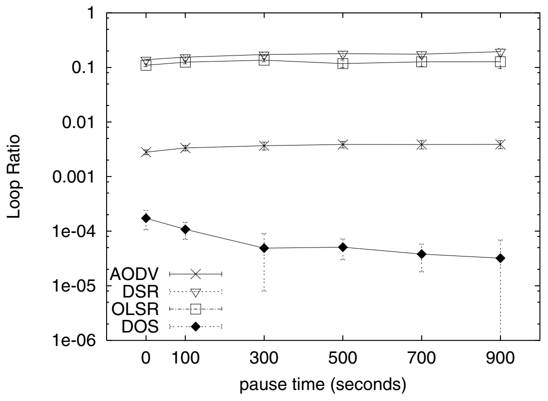

Fig. 17. Loop ratio 100-nodes, 30-source