UC Santa Cruz

UC Santa Cruz Previously Published Works

Title

ODVR: A Unifying Approach to On-Demand and Proactive Loop-Free Routing in Ad-Hoc Networks

Permalink

<https://escholarship.org/uc/item/33j51124>

Author

Garcia-Luna-Aceves, J.J.

Publication Date

2019

Peer reviewed

ODVR: A Unifying Approach to On-Demand and Proactive Loop-Free Routing in Ad-Hoc Networks

J.J. Garcia-Luna-Aceves^{*,†} and Ehsan Hemmati^{*}

^{*}Computer Science and Engineering Department, University of California, Santa Cruz, CA 95064

[†]PARC, Palo Alto, CA 94304

jj@soe.ucsc.edu, ehsan@soe.ucsc.edu

Abstract—We introduce **Ordered Distance Vector Routing (ODVR)**, which is the first approach that enables on-demand and proactive loop-free routing on a per-destination basis. ODVR establishes a strict total ordering of nodes with respect to any given destination using the distances to that destination and reference distances that nodes responding to route requests must have in order to be allowed to send responses. Destinations send gratuitous route replies to enact proactive routing. In contrast to all prior routing protocols, ODVR does not require source or destination sequence numbers, sequence numbers for messages, path information, source routing, or requiring a router to wait for replies from all its neighbors before making changes to its routing table. It is shown that loop-free on-demand routing cannot be attained simply by using destination sequence numbers and the type of signaling used in AODV (Ad-hoc On-Demand Distance Vector) if messages can be lost and nodes may lose routing state for any reason. It is also shown that ODVR provides loop-free routes at every instant. Simulation experiments using ns3 show that ODVR is more efficient than AODV and OLSR.

I. INTRODUCTION

Starting with the routing protocol of the DARPA packet-radio network [14], [15], many routing protocols have been proposed for ad-hoc networks over the years [1], [4]. One important aspect of the way in which such protocols operate is whether they are proactive or reactive, also called on-demand. A proactive routing protocol maintains routing state for destinations independently of whether there is data traffic intended for the destinations, and a reactive routing protocol establishes routing state on demand.

The fundamental tradeoff between the two approaches is well known. Proactive routing protocols result in shorter delays finding viable routes to destinations but incur signaling overhead for all destinations and hence scale poorly with increasing numbers of participating nodes. In contrast, on-demand routing protocols incur signaling overhead only for destinations for which there is traffic but suffer from increased latencies in finding viable routes through route discovery procedures that also scale poorly with increasing numbers of link failures resulting from node mobility or other causes.

As Section II describes, the routing protocols defined for ad-hoc networks to date work either proactively or on-demand for all network destinations, or consist of hybrid schemes that incorporate proactive and on-demand routing and attempt to reduce signaling overhead. Not all destinations are equally popular in a network and only a few destinations (e.g., a node with a server attached that supports a popular service) need to

communicate with many other network nodes. Furthermore, the characteristics of an ad-hoc network vary dynamically and a signaling modality that appears to be efficient for some conditions may behave inefficiently as the network characteristics and application behavior change. Hence, no routing modality is inherently better than the other, even for a given network.

The main contribution of this paper is introducing a unifying approach to on-demand and proactive loop-free routing that enables the dynamic choice of routing modality on a per-destination basis using the same signaling.

It could be argued that developing an integrated approach to on-demand and proactive loop-free routing could be attained by modifying the basic destination-based sequence numbering approach used in AODV (Ad-hoc On-Demand Distance Vector) [23] and DSDV [22]. However, as Section III proves, routing-table loops can occur in AODV. More importantly, it is shown that loop-free routing *cannot* be attained based on the signaling and destination sequence numbering used in AODV in a network in which relay nodes send back replies, messages can be lost, and nodes may lose routing state.

On the other hand, all other prior loop-free routing approaches incur much more communication overhead than simply communicating distances on a best-effort basis. The known schemes are: using source routes in routing updates and data packets [13], including path information or topology information in routing updates [5], [20], and requiring a router that sends a route request to wait for replies from all its neighbors before making changes to its routing table [24]. Accordingly, a new approach is needed for loop-free multipath routing.

Section IV introduces *Ordered Distance Vector Routing (ODVR)* to provide multiple loop-free routes at every instant based on distances to destinations maintained on-demand or proactively in an ad-hoc network. Routing state is maintained on demand or proactively for a given destination using route requests and route replies. The need for destination sequence numbers is eliminated by means of *reference distances* included in route requests and route replies. A reference distance states the maximum distance to a destination that a node responding to a route request is allowed to have, and is set equal to 0 when the destination must answer. As needed, a destination can set the routing modality for itself to be proactive by issuing gratuitous route replies for itself stating a 0 reference distance. The use of sequence numbers to identify

route requests is eliminated by having each route request state the prior node that handled the request and by having each node use a pending-request table to store a record of the route requests it has forwarded.

Section V proves that the use of reference distances introduced in ODVR for proactive and on-demand routing results in loop-free multi-path routing at every instant independently of the state of the topology, the amount of time a node stores routing state, or the reliability or timing with which signaling messages are exchanged among nodes. The use of reference distances to enable multi-path loop-free routing is important because of its remarkable simplicity, and because ODVR is provably loop-free, which is not the case of routing protocols based destination-based sequence numbers.

Section VI compares the performance of ODVR with the performance of AODV and OLSR (Optimized Link State Routing) [5], which are well-known examples of proactive routing and on-demand routing for MANET's. The simulation experiments use scenarios aimed at considering the impact of node speed, pause times, number of sources, and network size on the packet-delivery ratio, average end-to-end delays, and signaling overhead. The performance comparison is intended only to illustrate the efficiency of ODVR using scenarios in which no routing-table loops occur in AODV and where OLSR sequence numbering does not induce additional signaling due to node failures. The results of the simulations indicate that ODVR is a more efficient approach for on-demand routing and proactive routing in ad-hoc networks. Furthermore, it can be used in combination with such techniques as hierarchical routing and location-based routing to support routing in large-scale networks. Section VII presents our conclusions.

II. RELATED WORK

Many MANET routing protocols have been proposed since the introduction of the routing protocol for the DARPA packet-radio network [14], [15] and many surveys and comparative studies exist (e.g., [1], [4], [29]). These protocols can be categorized as proactive or reactive, with hybrid approaches using proactive and on-demand mechanisms. Proactive routing protocols maintain routing state for all destinations, regardless of whether traffic exists for them. By contrast, on-demand or reactive routing protocols maintain routing state for only those destinations for which data traffic exists and rely on flood search mechanisms to establish routes to destinations.

A number of proactive and on-demand routing protocols use partial or complete link-state information or second-to-last hop information to establish routing state for destinations. Examples of this approach are OLSR [5], STAR [9], OLIVE [10], and WRP [20]. Other routing protocols such as DSR [13] go even further and obtain source routes to destinations and use them in packet forwarding. A few routing protocols like ROAM [24] rely on synchronizing routing-table updates among nodes so that no node changes its next hop to a destination before receiving feedback from all its neighbors about its current distance.

Corson and Ephremides [6] introduced one of the first proposals for on-demand routing. Their specific approach incurs excessive signaling overhead because replies to route requests are flooded. However, their use of sequenced route requests has been adopted in many subsequent on-demand routing protocols based on distances.

There are several routing protocols today that use sequence numbering of distances to address loop freedom, and DSDV [22] was arguably the first to use of destination sequence numbers. A node accepts a next hop to a destination only if the destination sequence number stated by that neighbor is smaller than the number held by the node or the destination sequence number is the same but the distance offered by the neighbor is smaller than the current node distance to the destination. AODV [23] adapts the approach in DSDV to on-demand routing. To find a route to an intended destination, a source broadcasts a route request stating the source and destination nodes, the most recent sequence number known for each, a broadcast identifier and a hop count to the source. Nodes maintain state for the requests they originate or forward, and discard subsequent copies of requests that they have forwarded. The destinations or nodes with valid routes to destinations reply to route requests following the paths traversed by the requests in reverse. A reply states the destination and source of the request, the destination sequence number, and the hop count to the destination. A node receiving a reply establishes a route to the destination stating the destination sequence number, the next hop, and the neighbors using the route (precursors). Nodes forward only the first copies of replies based on the destination sequence numbers.

Several hybrid routing protocols have been proposed over the years that use both on-demand and proactive routing modalities and in general use proactive routing within regions of the network and on-demand routing to find routes to destinations outside the regions to which sources belong [1]. The main motivation for these protocols has been attempting to reduce the signaling overhead incurred for routing, and some approaches use backbones or virtual backbones to further reduce overhead. Hybrid routing schemes have been proposed based on distance information and link-state information, and the many examples include advertising gateways [18], ZRP [11], ZHLS [12], VBR [19], HARP [21], SHARP [25], and NEST [26].

A common characteristic of all prior hybrid routing approaches is that they combine two types of signaling, with some using existing on-demand or proactive routing protocols as part of the solution. To date, no routing protocol has been proposed that: (a) uses the same signaling to provide efficient routing using either proactive or on-demand modalities on a per-destination basis, and (b) allows the routing modality used for a given destination to change dynamically.

Clearly, an on-demand, proactive, or hybrid routing approach can take advantage of a number of mechanisms reported over the years to reduce signaling overhead. These mechanisms include using geographical or virtual coordinates (e.g., LAR [16]), establishing connected dominating sets [28],

exploiting address aggregation, maintaining backbones connecting network regions and establishing hierarchical clustering [1], [3], [4]. The same can be said for ODVR, which we present in the next section, especially given that its routing modalities are dynamic and operate on a per-destination basis. Accordingly, the rest of this paper focuses on routing in networks small networks, and compares the new approach against two basic on-demand and proactive routing schemes.

III. DESTINATION SEQUENCE NUMBERS AND ROUTING-TABLE LOOPS

A number of looping problems in the original AODV proposal in [23] and subsequent versions of AODV [2], [8], [27] have been identified over the years. Various approaches have been proposed to make destination sequence numbering more resilient [8] than in the original AODV version [23], and proposals have also been made to provide multiple paths per destination [17]. However, AODV and subsequent proposals based on destination sequence numbers do not address why the protocols should work correctly even when nodes experience the loss of routing state and signaling messages are lost.

The proof of loop freedom provided by Perkins and Royer [23], which is similar to the proof in [17], attempts to establish a contradiction on the existence of a routing-table loop L involving $n > 0$ nodes. The proof first states that $sn_d^1 \leq sn_d^2 \leq \dots \leq sn_d^n \leq sn_d^1$ and hence every node $i \in L$ must hold the same sequence-number value. Given that proposition, the proof states that node i makes node $i + 1$ its next hop only if $D_d^i = D_d^{i+1} + 1$. Based on the two propositions, the conclusion is reached that $D_d^1 = D_d^n + (n - 1)$, which together with the fact that $D_d^n = D_d^1 + 1$ leads to the contradiction that $n = 0$.

The fallacy in the previous argument, as well as in the proof in [17], is that it *assumes* that the protocol signaling must enforce the condition that $sn_d^1 \leq sn_d^2 \leq \dots \leq sn_d^n \leq sn_d^1$ if loop L is created. However, this assumption is not true if at least one node i that helps create L has erased its routing state for d for any reason just before it helps create L by choosing a new next hop. According to the operation of AODV in [23] and subsequent versions of AODV, $sn_d^i = 0$ if node i has no routing state for d . This state allows node i to accept a response from node $i + 1$ if $sn_d^{i+1} > 0$ independently of the value of D_d^{i+1} . As a consequence, as prior work discusses [2], [8], [27], a node can select a neighbor with $D_d^i \neq D_d^{i+1} + 1$ when a node deletes routing state due to failures or timeouts, which can lead to routing-table loops.

Bhargavan et al. [2] showed that the looping problems they identified in AODV can be eliminated, provided that nodes *never* delete routing-table entries and *immediately* detect when a neighbor node restarts its routing process. While their proof is valid, the conditions needed to guarantee loop freedom cannot be implemented in practice.

Zhou et al. [30] provided the sketch of a proof showing that AODV signaling cannot create routing-table loops, provided that the destinations are the only nodes that answer route

requests. This restriction does render failsafe behavior; however, it makes the signaling overhead of on-demand routing far more onerous than the overhead incurred if relays are allowed to send replies. More importantly, forcing destinations to answer all route requests negates the need to use destination sequence numbers in the first place! This becomes apparent by observing that each node identifies the route requests it originates with source sequence numbers to avoid duplicate forwarding; therefore, destinations can simply use the source sequence numbers carried in the requests to send replies only to the new requests they receive, and relays can use the same source sequence numbers to forward the replies.

Given the prior work showing the possibility of loops in AODV, it may be surprising that no valid proof exists showing that AODV avoids routing-table loops in any possible ad-hoc networking scenario. However, the following theorems show that on-demand loop-free routing cannot be attained using destination sequence numbers and the type of signaling assumed in AODV if relay nodes can answer route requests, and signaling messages and routing state may be lost.

A distributed routing algorithm (DRA) is said to be loop-free if it is able to maintain loop-free routes at every instant while the network is subject to any sequence of link or node failures and recoveries, message losses, and loss of routing state at any node. For a DRA to obtain shortest paths and be loop-free, it must establish a *strict total ordering* of the nodes with respect to any given destination d based on a *strict ordering constraint* that establishes an asymmetric relation between each node k and any neighbor that it chooses as a next hop to d , which we denote by s_d^k .

We denote by m_d^k the value of the metric at node k for destination d used to define a strict ordering constraint as follows:

$$\mathcal{C} : m_d^k \succ m_d^{s_d^k} \quad (1)$$

Theorem 1: A DRA is loop-free if and only if the strict ordering constraint $\mathcal{C}$ is satisfied by every node k at every instant.

Proof: Assume first that $\mathcal{C}$ is always satisfied by every node. For the sake of contradiction assume that a set of nodes $\{v_1, v_2, \dots, v_h, v_1\}$ different than destination d create a routing-table loop L of h hops by setting $s_d^{v_i} = v_{i+1}$ for $1 \leq i \leq h - 1$ and $s_d^{v_h} = v_1$. This means that $m_d^{v_i} \succ m_d^{v_{i+1}}$ for $1 \leq i \leq h - 1$ and $m_d^{v_h} \succ m_d^{v_1}$. However, this is a contradiction, because it implies that $m_d^{v_i} \succ m_d^{v_i}$ for $1 \leq i \leq h - 1$ and $m_d^{v_h} \succ m_d^{v_h}$. Next assume that a DRA is loop free and for the sake of contradiction assume that $\mathcal{C}$ is not satisfied by at least one node k with respect to a neighbor j at some point. This is a contradiction, because then k can use node j as a successor and node j can use node k as a successor, which is a loop. Therefore, the theorem is true. ■

We define a distributed on-demand routing algorithm (DORA) to be any DRA in which route requests can be answered by nodes other than the destinations of the requested routes. We denote by sn_d^k the destination sequence number currently adopted by node k for destination d and by $sn_{s(k)}^k$

the destination sequence number reported by neighbor s_d^k to node k . In a DORA in which destination sequence numbers are used, a node k assumes the sequence number reported by its successor. Therefore, $sn_d^k = sn_{s(k)d}^k$ and it follows from Theorem 1 that no loop-free DORA can be designed that uses only destination sequence numbers as the ordering constraint.

It is clear from the above that a strict ordering constraint that uses destination sequence numbers *must* include an additional metric to enforce a strict ordering of nodes for a destination d . Based on this, we consider the ordering constraint introduced in DSDV and AODV assuming a second metric r_d^k , which can be stated as follows:

$$\mathcal{T} : m_d^k \succ m_d^{s_d^k} \equiv [sn_d^k < sn_{s_d^k}^k] \vee [(sn_d^k = sn_{s_d^k}^k) \wedge (r_d^k \succ r_{s_d^k}^k)] \quad (2)$$

The intent of $\mathcal{T}$ is to attain loop-free routing on demand by forcing nodes to trust only the most recent sequence numbers and use a routing metric (the distance to a destination for the case of AODV or DSDV) to establish the strict ordering needed for loop freedom. To ensure that every network node k can obtain shortest paths to a given destination d in this context, if node k does not have any information about destination d , it must assume the largest possible value for m_d^k , so that it can accept any non-zero sequence number value associated with some finite value of the secondary metric used. Therefore, we must have

$$sn_d^u = 0 \wedge r_d^u = \infty \text{ after reset or routing-state initialization} \quad (3)$$

The following theorem assumes a DORA that operates using the same signaling used in AODV and such that a node without routing information for destination d that satisfies Eq. (3) and sends a route request accepts any reply that satisfies Eq. (2).

Theorem 2: Loop-free routing cannot be attained if routing state and signaling messages may be lost using a DORA in which nodes without routing state for a destination d satisfy Eq. (3) and use $\mathcal{T}$ to avoid routing-table loops.

Proof: Assume for the sake of contradiction that DORA is loop-free even when routing state and signaling messages are lost, i.e., that it enforces strict total ordering based on $\mathcal{T}$.

Consider a sub-path of path P_d^k that exists at time t_0 consisting of n nodes denoted by the hop count they have along P_d^k from k , i.e., $\{k \rightarrow k+1 \rightarrow \dots \rightarrow k+n\}$, where $n < |P_d^k|$.

Because signaling messages may be lost, it is possible for node $k+n$ to delete its routing state at some time $t_1 > t_0$ without changing the routing state of nodes k or $k+n-1$. In this case we have that $sn_d^{k+n} = 0$ and $r_d^{k+n} = \infty$, while $sn_d^{k+i} > 0$ and $r_d^{k+i} < \infty$ with $k \leq i \leq k+n-1$.

Assume that node $k+n$ sends a route request at time $t_2 > t_1$ to obtain a route to d . It is possible for the request to not be received by node $k+n-1$ and be received by some of the other nodes in the subpath $\{k \rightarrow k+1 \rightarrow \dots \rightarrow k+n-2\}$. Therefore, it is possible for node $k+n$ to select as its next hop some node j from which node $k+n$ receives a response at time $t_3 > t_2$ and such that j is in the same subpath, because $sn_d^{k+n} < sn_d^j$ and $r_d^j < \infty$ satisfies Eq. (2), where $k \leq j \leq k+n-2$.

However, this is a contradiction to the assumption that DORA is loop-free, because $r_d^{k+n-1} \not\prec r_d^{k+n}$ after time t_3 . ■

Theorem 2 does not mean that an on-demand routing protocol based on destination sequence numbers cannot operate without creating routing-table loops most of the time by means of additional mechanisms addressing the loss of routing state. This is similar to the case of the distributed Bellman-Ford algorithm (DBF), which incurs routing-table loops and yet routing protocols based on DBF can use hold-down timers to reduce the possibility of routing-table loops or counting-to-infinity from occurring.

A clear implication of Theorem 2 is that the designers of any on-demand routing protocol based on destination sequence numbers are left with the necessary task of adding mechanisms to cope with the inherent inability of destination sequence numbering to prevent routing-table loops in certain cases. Theorem 1 indicates that, at least in theory, loop-free on demand routing could be attained with some approach that modifies Eq. (2). The question is whether this effort is worthwhile. The previous result demonstrates that *destination sequence numbers are not sufficient* to provide loop-free routing. On the other hand, the existence of loop-free on-demand routing protocols that do not use destination sequence numbers (e.g., [24]) demonstrates that *destination sequence numbers are not necessary* to provide loop-free routing based on distance information.

IV. ODVR

A. ODVR Overview

The design of ODVR is motivated by three facts: (a) Destination sequence numbers are neither necessary nor sufficient to attain on-demand or proactive loop-free routing; (b) no prior loop-free routing protocol exists that uses only distances to destinations and does not require synchronizing routing-table updates; and (c) no routing modality is inherently better than the other.

The operation of ODVR has three novel components: (a) Distance-based multi-path loop-free routing without destination sequence numbering; (b) route-request forwarding without sequence numbering of requests; and (c) dynamic choice of routing modality on a per destination basis.

Reference Distances for Loop-Free Routing: ODVR operates by establishing a strict total ordering of nodes with respect to individual destinations using only distance values. This strict total ordering is enforced by having each route request and reply state a *reference distance* to the intended destination. A node must forward a route request stating a smaller reference distance than its current distance or reference distance, and adopts the smaller reference distance stated in the forwarded request as its new reference distance. A node trusts only replies stating a reference distance smaller than or equal to the reference distance it requires. A node that sends a route request regarding a destination for which it has no routing state must use a reference distance of 0 to force the destination to answer the request. This allows nodes to lose routing state without causing routing-table loops when they

recover. To save bandwidth, a relay node with a valid routing-table entry for a destination that receives a route request with a reference distance of 0 forwards the request only to its best next hop. This is done by indicating in a route request either the specific neighbor that should process the request or a broadcast address.

Pending Request Tables (PRT): Using PRT's eliminates the need for sequence numbers to identify route requests. Each route request states the previous hop traversed by the request, and PRT's are used by nodes to process route requests and to add or update routing-table entries. A node adds a routing-table entry for a destination only when the node answers a route request or receives a reply to a pending route request. A node can update an existing routing-table entry when it receives a reply regarding that entry. A node forwards a reply only if it is a proactive reply or the node has neighbors that forwarded requests stating prior hops for their requests that are different than the node itself. To avoid deadlocks when requests are lost, a node forwards all retransmissions of requests received from its neighbors, provided that they state previous hops traversed by the requests other than the node itself. It is up to the origins of requests to moderate the rate at which they retransmit their requests.

Dynamic Per-Destination Routing Modality: By default, routes to destinations are maintained on demand through route requests and replies. A destination can change to proactive mode by issuing *proactive route replies* for itself periodically, with each reply stating a 0 reference distance. The destination returns to the on-demand routing modality simply by not originating more proactive route replies for itself. A router forwards a proactive reply for a destination received from a neighbor k if k is closer to the destination and provides the shortest distance to it. The rest of the signaling and mechanisms are the same for on-demand and proactive routing.

B. Information Exchanged

Signaling messages are transmitted in broadcast mode among neighboring nodes. A signaling message sent by node i at time t is denoted by $SM^i(t)$ and contains the identifier of the node (i) and one or multiple update entries, each of which can be either a request for a route (*REQ*) or a reply to a request (*REP*).

$REQ_d^i[Q, d, o, RD_d^i, D_o^i, p_d^i, f_d^i]$ denotes a route request sent by node i for destination d originated by node o . In this tuple, Q states that the entry is a request, d is the destination identifier, o is the identifier of the origin of the request, RD_d^i is a reference distance to d used to prevent routing loops, D_o^i is the distance attained by node i to the origin of the request, p_d^i denotes the *request predecessor* (i.e., the neighbor from which node i received the request before forwarding it), and f_d^i states the neighbor(s) of node i that must process the request. If $f_d^i = 0$, the request should be processed by every node that receives the request.

$REP_d^i[R, d, o, RD_d^i, D_d^i, D_o^i]$ denotes a reply sent by node i for destination d . The value R states that the entry is a reply, d is the destination identifier, o is the origin of

the request, RD_d^i is the reference distance of the node that originated the reply, D_d^i is the distance attained by node i to destination d , and D_o^i is the distance attained by node i to the origin of the request o .

When node i receives a signaling message from neighbor k , it processes each query and reply in the message independently of the others. We describe the operation of ODVR by focusing on a particular destination d .

C. Information Stored

Node i maintains four tables to operate, and a maximum lifetime (LT) is allowed for any table entry other than self entries, with the lifetime of each entry decremented from the moment it is created or updated. The set of neighbors of node i is denoted by N^i .

Link-Cost Table (LCT^i): The table states the cost of each link from node i to each known neighbor. The cost of link (i, k) is denoted by l_k^i . By definition, $(i, k) > 0$ for $i \neq k$.

Distance Table (DT^i): The entry for destination d of DT^i is denoted by $DT^i(d)$ and specifies: the identifier of destination d and the distance to d reported by each known neighbor. The distance reported by node $k \in N^i$ for destination d maintained by node i is denoted by D_{kd}^i . If a known neighbor q has not reported a distance for d , then it is assumed that $D_{qd}^i = \infty$. The vector of distances reported by neighbor k and stored by node i is denoted by DT_k^i .

Routing Table (RT^i): The entry for destination d of RT^i is denoted by $RT^i(d)$ and specifies: the identifier of d , the shortest distance to d (D_n^i), the set of valid successors (S_d^i), a proactive update timeout (PUT_d^i), and a lifetime for the routing-table entry (LR_d^i).

The *valid successor set* S_d^i is defined as follows:

$$S_d^i = \{ k \in N^i \mid (D_{dk}^i < D_d^i) \wedge (D_{dk}^i + l_k^i = D_{min}^i(d)) \}$$

$$\text{where } D_{min}^i(d) = \text{Min}\{D_{dn}^i + l_n^i \mid n \in N^i\} \quad (4)$$

The node with the smallest identifier in set S_d^i is denoted by s_d^i . By definition, $D_d^i = 0$, $S_d^i = \{d\}$, $r_d^i = 0$, and $LR_d^i = \infty$.

The proactive update timeout PUT_d^i is needed because routing state for a proactive destination d must be refreshed with enough frequency so that its state is not erased after a maximum lifetime LT . Hence, $PUT_d^i < LR_d^i$ for all $i \neq d$.

LR_d^i in an existing routing-table entry for d is assumed to be reset to its maximum value of LT every time node i receives a data packet for or from destination d .

Pending-Request Table (PRT^i): This table keeps track of the route requests waiting for replies. The entry for destination d is denoted by $PRT^i(d)$ and states: the identifier of node d , the smallest reference distance received in a pending request from a neighbor (RD_d^i), a request list RL_d^i , and a lifetime for the entry (LP_d^i). The list RL_d^i contains of one or more tuples, with each tuple consisting of the identifier of the origin of a route request and a list of identifier-distance pairs for all neighbors from which the request was received, which we call *pending-request neighbors*. The tuple in RL_d^i corresponding to origin o is denoted by $RL_d^i(o)$ and its content is $[o, PRN_d^i(o)]$.

The pair in $PRN_d^i(o)$ corresponding to neighbor k is denoted by $PRN_{dk}^i(o)$ and equals the pair $[k, PD_{ok}^i]$, where PD_{ok}^i denotes the distance to origin o reported by k in its route request regarding destination d .

Modality Flag: The routing modality adopted by a destination is stated by the value of its *modality flag*, which is denoted by mod^d , with $mod^d = 0$ denoting on-demand routing and $mod^d = 1$ denoting proactive routing. The actual setting of the modality flag of a destination is carried out outside of the operation of ODVR. Possible mechanisms that would update the value of mod^d include a system administrator setting the value or a daemon monitoring data traffic to the destination and determining that the popularity of the node, the type of traffic to the node, or the applications running at the node merit proactive routing.

D. Maintaining Routing State

Maintaining routing state for a destination d involves creating and processing route requests and replies.

Originating Requests: A node that has data for a destination d or has an empty valid successor set after processing a proactive route reply originates a request $REQ_d^i[Q, d, o = i, RD_d^i, D_o^i = 0, p_d^i = i, f_d^i = 0]$ when the following condition is satisfied. Given that there is no request predecessor for the route request, $p_d^i = i$, and the value $f_d^i = 0$ requests all the neighbors of node i to process the request.

ORC (Originating Request Condition): Node creates the request REQ_d^i if

$$(\neg PRT^i(d)) \wedge ([\neg RT^i(d)] \vee [(\exists RT^i(d)) \wedge (S_d^i = \emptyset)]) . \square$$

ORC states that a node originates a route request if it is not already waiting for a reply regarding destination d and either it has no routing state for d or has routing state but has no valid successor for destination d .

Node i creates $PRT^i(d)$ with the identifier of node d , a value for RD_d^i that depends on the routing state for d , a lifetime $LP_d^i = LT$, and $RL_d^i(i) = [i, PRN_d^i(i)]$.

The list $PRN_d^i(i)$ consists of the tuple $[i, PD_{ii}^i = 0]$.

Node sets $RD_d^i = 0$ if there is no entry $RT^i(d)$ to force destination d to respond to its request. On the other hand, if $RT^i(d)$ exists but $S_d^i = \emptyset$ then node i sets $RD_d^i = D_d^i$ to allow a node with a distance smaller than D_d^i to respond. Node i adds the request REQ_d^i to its signaling message after updating PRT_d^i .

Processing Route Requests: If node i receives a signaling message from neighbor k containing $REQ_d^k = [Q, d, o, RD_d^k, D_o^k, p_d^k, f_d^k]$, it updates DT^i with $D_{ok}^i = D_o^k$ to remember the distance to the origin of the request reported by k , and determines whether to forward, answer, ignore, or remember the request using one of the following sufficient conditions that prevent routing-table loops by ordering nodes based on their distances to destination d .

FRC (Forwarding Request Condition): Node i forwards request REQ_d^i to all its neighbors if

$$[(\exists PRT^i(d)) \wedge$$

$$([0 < RD_d^k < RD_d^i] \vee [\exists PRN_{dk}^i(o)] \vee [\neg PRN_d^i(o)])] \vee [(\neg PRT^i(d)) \wedge ([S_d^i - \{k\} = \emptyset] \vee [0 < RD_d^k < D_d^i])] . \square$$

FRC states that node i should forward a route request received from a neighbor in two cases. If node i is expecting a reply to a prior request for the same destination d , then node i forwards the request if the request states a shorter reference distance than the one currently assumed, or the request is a retransmission of a prior request from the same origin, or the request was originated from an origin from which requests have not been received. A request retransmission is recognized because k is listed as having sent a request for d from origin o . Alternatively, if node i does not expect a reply to a prior request for the same destination d , then node i forwards the request if either node i has no valid successor for the destination excluding neighbor k or the request states a non-zero reference distance smaller than the current distance from node i to d .

We observe that FRC cannot be satisfied when $p_d^k = i$ because $RD_d^k \geq RD_d^i$ in that case. Hence, node i can only forward requests that are not the result of a prior request from i itself. If FRC is satisfied, node i must create or update $PRT^i(d)$. Accordingly, node i sets $RD_d^i = \min\{RD_d^i, RD_d^k\}$, sets $LP_d^i = LT$, and updates $RL_d^i(o)$ by adding k to the list of neighbors requiring a reply, which means that $PRN_{dk}^i(o) = PRN_{dk}^i(o) \cup \{[k, PD_{ok}^i = D_o^k]\}$.

Once PRT^i is updated, node i computes $D_o^i = D_o^k + l_k^i$ and adds the request $REQ_d^i[Q, d, o, RD_d^i, D_o^i, p_d^i = k, f_d^i = 0]$ to its signaling message.

RFC (Reset Forwarding Condition): Node i forwards REQ_d^i only to $s_d^i \in S_d^i - \{k\}$ if

$$(\neg PRT^i(d)) \wedge (RD_d^k = 0) \wedge (S_d^i - \{k\} \neq \emptyset) . \square$$

RFC states that a node with a valid routing-table entry for destination d that receives a “reset request” (a request with a zero reference distance) forwards the request directly to the neighbor other than k that has the smallest identifier among the nodes in its valid successor set.

If RFC is satisfied, node i eliminates k as a next hop to d by updating $RT^i(d)$ with $S_d^i = S_d^i - \{k\}$. Node i then creates $PRT^i(d)$ with the identifier of node d , $RD_d^i = RD_d^k = 0$, $LP_d^i = LT$, and RL_d^i consisting of the tuple $RL_d^i(o) = [o, PRN_d^i(o)]$ with $PRN_d^i(o) = \{[k, PD_{ok}^i = D_o^k]\}$. Node i computes $D_o^i = D_o^k + l_k^i$ and adds the request $REQ_d^i[Q, d, o, RD_d^i = 0, D_o^i, p_d^i = k, f_d^i = s_d^i]$ to its signaling message.

IRC (Ignore Request Condition): Node i ignores REQ_d^k if $[(f_d^k \neq 0) \wedge (f_d^k \neq i)] \vee [(f_d^k = 0) \wedge (p_d^k = i)] . \square$

IRC states that node i ignores a route request explicitly intended for a different node or a request forwarded by a neighbor as a result of processing a request from node i itself.

SRC (Store Request Condition): Node i stores information from REQ_d^k with no other action if

$$(f_d^k = 0) \wedge (p_d^k \neq i) \wedge (RD_d^k \geq RD_d^i) \wedge$$

$$(\exists PRN_d^i(o)) \wedge (\neg PRN_{dk}^i(o)). \quad \square$$

SRC states that node i can simply add neighbor k as a node that needs a route for destination d if node i has already forwarded a route request for destination d from the same origin stated by k and k is not already listed as a neighbor from which a request from origin o and with a previous hop other than i has been received.

If SRC is satisfied node i updates PRT_d^i . It sets $LP_d^i = LT$ and updates $RL_d^i(o)$ with $PRN_d^i(o) = PRN_d^i(o) \cup \{[k, PD_{ok}^i = D_o^k]\}$.

RRC (Reply Request Condition): Node i sends a reply to request REQ_d^k if

$$[d = i] \vee [(RD_d^k > D_d^i) \wedge (\neg PRT^i(d))] \vee [(f_d^k = i) \wedge (S_n^i = \emptyset)]. \quad \square$$

According to RRC node i can answer a request in three cases. Node i can reply independently of the reference distance stated in the request if it is the intended destination. Node i can also reply if it is not waiting for a reply for a route to d and it is closer to d than the reference distance stated in the request. Node i also replies if the request is explicitly directed to itself (i.e., $f_d^k = i$) and node i does not have a valid successor to destination d (i.e., $S_d^i = \emptyset$), in which case node i informs k that $D_d^i = \infty$.

If RRC is satisfied node i can update its routing table regarding origin o . To do so node i uses Eq. (4) to compute $D_{min}^i(o)$ and S_o^i and updates $D_o^i = D_{min}^i(o)$. Node i then includes the reply $REP_d^i[R, d, o, RD_d^i = D_d^i, D_d^i, D_o^i]$ in its signaling message.

Originating Proactive Replies: Any destination can become a proactive destination simply by transmitting proactive replies for itself periodically, and can become an on-demand destination again simply by stopping its transmission of such replies. The following conditions define the transmission of proactive replies by nodes.

PRC (Proactive Reply Condition): Destination d originates the proactive reply $REP_d^d[R, d, o = d, RD_d^d = 0, D_d^d = 0, D_o^d = 0]$ if $(mod^d = 1) \wedge (PUT_d^d = 0)$. $\square$

PRC simply states that a proactive destination issues a proactive reply for itself periodically every RT_d^d seconds. Node d sets $PUT_d^d = 0$ the first time that node d sets $mod^d = 1$, so that PRC is satisfied. Node d resets the value of PUT_d^d to the maximum value after it sends its proactive reply, so that periodic reply transmissions take place.

Processing Replies: Nodes process replies opportunistically, given that all signaling messages are sent in broadcast mode. Nodes are allowed to add new routing-table entries only after receiving proactive replies or replies that answer requests they have originated or forwarded. Nodes can update existing routing-table entries after receiving subsequent replies.

When node i receives the reply $REP_d^k[R, d, o, RD_d^k, D_d^k, D_o^k]$ from neighbor k , node i first updates DT^i by setting

$D_{dk}^i = D_d^k$ and $D_{ok}^i = D_o^k$. Node i then can determine if $k \in S_d^i$ and uses the following condition to determine if it can use the reply from k to answer its own request, improve its distances to either destination d or origin o if they are already instantiated in RT^i , or update its routing state for a proactive destination d .

ARC (Accept Reply Condition): Node i accepts the reply $REP_d^k[R, d, o, RD_d^k, D_d^k, D_o^k]$, if

$$[d = o] \vee [(d \neq o) \wedge (\neg PRT^i(d)) \wedge (D_d^k < D_d^i)] \vee [(d \neq o) \wedge (\exists PRT^i(d)) \wedge (RD_d^k < RD_d^i)]. \quad \square$$

ARC states that node i accepts a reply from neighbor k if the reply is proactive, the reply is for an on-demand destination and node k is closer to the destination and node i is not waiting for replies, or the reply is for an on-demand destination and satisfies the reference distance stated in the existing request from node i . The steps taken when ARC is satisfied are the following.

Case 1 ($d = o$ and $PRT^i(d)$ does not exist):

If $RT^i(d)$ does not exist, node i creates an entry for it. Node i uses Eq. (4) to compute $D_{min}^i(d)$ and S_d^i . If $D_{min}^i(d) \neq D_d^i$ node i updates $D_d^i = D_{min}^i(d)$. Given that the proactive reply was created by d , node i can trust the reply even if $D_{min}^i(d) > D_d^i$ and k must be in S_d^i . Node i adds the proactive reply $REP_d^i[R, d, o = d, RD_d^i = 0, D_d^i, D_d^i]$ to its signaling message if $D_{min}^i(d) \neq D_d^i$.

Case 2 ($d = o$ and $PRT^i(d)$ exists):

Node i uses REP_d^k as an answer to its own request in this case because $RD_d^k = 0$ and hence $RD_d^k \leq RD_d^i$. Because $D_d^k < \infty$ node i sets $D_{min}^i(d) = D_{jk}^i + l_k^i$ to reset the distance used to define valid successors, which results in $S_d^i = \{k\}$ according to Eq. (4). Node i updates RT^i with $D_d^i = D_{jk}^i + l_k^i$ and S_d^i . Node i also adds the proactive reply $REP_d^i[R, d, o = d, RD_d^i = 0, D_d^i, D_d^i]$ to its signaling message.

Case 3 ($d \neq o$ and $PRT^i(d)$ does not exist):

Node i does nothing else in this case if $RT^i(d)$ and $RT^i(o)$ do not exist. Alternatively, node i uses Eq. (4) to compute $D_{min}^i(d)$ and S_d^i if $RT^i(d)$ exists and to compute $D_{min}^i(o)$ and S_o^i if $RT^i(o)$ exists. If $D_{min}^i(d) \neq D_d^i$ or $D_{min}^i(o) \neq D_o^i$ node i updates $D_d^i = D_{min}^i(d)$ and $D_o^i = D_{min}^i(o)$ as needed. Node i adds the gratuitous reply $REP_d^i[R, d, i, RD_d^i = D_d^i, D_d^i, D_i^i]$ to its signaling message if $RT^i(d)$ is updated, and adds the gratuitous reply $REP_o^i[R, o, i, RD_o^i = D_o^i, D_o^i, D_i^i]$ to its signaling message if $RT^i(o)$ is updated.

Case 4 ($d \neq o$, and $PRT^i(d)$ exists):

Node i uses REP_d^k as an answer to its own request in this case. If $D_d^k < \infty$, node i updates RT^i with $D_d^i = D_{jk}^i + l_k^i$ and $S_d^i = \{k\}$, else $D_d^k = \infty$ and hence node i updates RT^i with $D_d^i = \infty$ and $S_d^i = \emptyset$.

Node i updates or creates $RT^i(o)$ by computing $D_{min}^i(o)$ and S_o^i using Eq. (4) and sets $D_o^i = D_{min}^i(o)$. Furthermore, for each origin p for which there is an entry $RL_d^i(p)$ in PRT^i , node i updates or creates $RT^i(p)$ by computing $D_{min}^i(p)$ and S_p^i using Eq. (4) and sets $D_p^i = D_{min}^i(p)$. Node i also adds the

reply $REP_d^i[R, d, o, RD_d^i = RD_d^k, D_d^i, D_o^i]$ to its signaling message if either $PRN_{dq}^i(o)$ exists for $q \in N^i$ or node i updated D_d^i or D_o^i . Node i then deletes $PRT^i(d)$.

E. Handling Link Changes and Unreliable Transmissions

To expedite neighbor discovery and make ODVR resilient in the presence of unreliable transmissions, each node transmits a signaling message periodically containing a *gratuitous route reply* $REP_i^i[R, i, o = i, RD_o^i, D_i^i, D_o^i]$ for itself with $RD_i^i = D_i^i = D_o^i = 0$.

To address the fact that signaling messages may be lost, node i also adds a gratuitous route reply $REP_d^i[R, d, o, RD_d^i, D_d^i, D_o^i]$ with $o = i, RD_d^i = D_d^i$ and $D_o^i = 0$ for each destination d in RT^i . To reduce the size of signaling messages, node i can simply include a hash value of the content of RT^i if it makes no changes to RT^i from the time it sent its previous signaling message.

Link additions are detected through the reception of signaling messages. Link failures are detected either through the link layer or as a result of a node not receiving any data packets or signaling messages from a neighbor for a period of time corresponding to the time needed to transmit two or three consecutive signaling messages periodically.

If node i detects a change in the cost of link (i, k) , it computes $D_{min}^i(d)$ and S_d^i for each destination d for which an entry $RT^I(d)$ exists using Eq. (4). In addition, for each destination d for which there is an entry $RT^I(d)$ such that $S_d^i \neq \emptyset$ and $D_{min}^i(d) \neq D_d^i$ node i updates $D_d^i = D_{min}^i(d)$ and adds a gratuitous reply $REP_d^i[R, d, o = i, RD_d^i = D_d^i, D_d^i, D_o^i = 0]$ to its signaling message.

If node i detects that the link (i, k) has failed, it sets $l_k^i = \infty$ in LCT^i and treats the link failure as a link-cost change. The entry D_{kd}^i for each destination d is deleted after its lifetime expires.

F. Handling Soft-State and Node Reboots

ODVR is a soft-state protocol. Entries in DT^i , RT^i , and PRT^i have finite lifetimes to allow router i to delete entries that become obsolete as a result of topology changes. Node i renews the lifetime of $RT^i(d)$ with every data packet it processes for destination d and deletes the entry if no data traffic is received for d during the lifetime of the entry. If entry $RT^i(d)$ is deleted, node i also deletes entry $DT^i(d)$. The entry $PRT^i(d)$ expires when LP_d^i reaches 0. Node i deletes the entry if this is the case and sets $S_d^i = \emptyset$ if $RT^i(d)$ exists. If node i is a source of data for destination d , node i can schedule the transmission of a new route request for d according to ORC.

If node i is asked to forward a data packet to destination d and $RT^i(d)$ does not exist or $S_d^i = \emptyset$, node i adds the gratuitous route reply $REP_d^i[R, d, o, RD_d^i, D_d^i, D_o^i]$ with $o = i, RD_d^i = D_d^i = \infty$ and $D_o^i = 0$ to its signaling message.

If node i initializes or reboots, it only has routing state for itself. It is assumed that $S_d^i = \emptyset$ and $D_d^i = \infty$ for any destination for which node i has no routing state.

V. LOOP FREEDOM IN ODVR

The following theorems prove that ODVR maintains loop-free routing tables at every instant. We assume the same notation introduced in Section IV.

Theorem 3: No routing-table loops can exist for destination d if the following ordering constraint is satisfied by each node i with a finite distance to d at every instant:

$$\mathcal{O} : (D_d^i = \infty) \vee (D_{ds^i}^i < D_d^i < \infty) \quad (5)$$

Proof: Consider a set of nodes $\{v_1, v_2, \dots, v_h, v_1\}$ different than destination d . Assume that this set of nodes create a routing-table loop L of h hops by setting $s_d^{v_i} = v_{i+1}$ for $1 \leq i \leq h-1$ and $s_d^{v_h} = v_1$. For the sake of contradiction, further assume that each node in loop L satisfies the ordering constraint $D_d^{v_i} = \infty$ or $D_{dv_{i+1}}^{v_i} < D_d^{v_i} < \infty$ for $1 \leq i \leq h-1$ and $D_d^{v_h} = \infty$ or $D_{dv_1}^{v_h} < D_d^{v_h} < \infty$.

Loop L cannot exist if $D_d^{v_i} = \infty$ for any i with $1 \leq i \leq h-1$ because then v_i has no next hop to d . It thus follows that $D_{dv_{i+1}}^{v_i} < D_d^{v_i} < \infty$ for $1 \leq i \leq h-1$ and $D_{dv_1}^{v_h} < D_d^{v_h} < \infty$. This, however, is a contradiction, because it implies that $D_d^{v_i} > D_d^{v_i}$ for $1 \leq i \leq h-1$ and $D_d^{v_h} > D_d^{v_h}$. Therefore, the theorem is true. ■

Theorem 3 states that if a routing protocol satisfies $\mathcal{O}$ at every instant then it ensures loop-free routing. The proof of the following theorem demonstrates that ODVR enforces $\mathcal{O}$ at every instant.

Theorem 4: ODVR ensures that $\mathcal{O}$ is satisfied by every network node i for any given destination d at every instant.

Proof: The definition of $\mathcal{O}$ in Eq. (5) can be expressed as:

$$\mathcal{O} \equiv [D_d^i = \infty] \vee [(D_{ds^i}^i < D_d^i) \wedge (D_d^i < \infty)] \quad (6)$$

Given that no distance can be larger than infinity, we have:

$$\begin{aligned} \mathcal{O} &\equiv [D_d^i = \infty] \vee [(D_{ds^i}^i < D_d^i) \wedge \neg(D_d^i = \infty)] \\ &\equiv ([D_d^i = \infty] \vee [D_{ds^i}^i < D_d^i]) \wedge ([D_d^i = \infty] \vee [\neg(D_d^i = \infty)]) \\ &\equiv [D_d^i = \infty] \vee [D_{ds^i}^i < D_d^i] \end{aligned} \quad (7)$$

Assume for the sake of contradiction that every node executes ODVR correctly and $\mathcal{O}$ is *not* satisfied at some moment t . Then there must be at least one node i that executes ODVR correctly and such that $\neg\mathcal{O}$ is true at time t . From Eq. (7) and DeMorgan's law our assumption means that node i updates its routing state for destination d at time t such that

$$\neg\mathcal{O} \equiv [D_d^i(t) < \infty] \wedge [D_{ds^i}^i(t) \geq D_d^i(t)] \quad (8)$$

From the operation of ODVR, if node i has no routing state for destination d at time t then $D_d^i(t) = \infty$. Hence, it must be true that node i has routing state for d at time t so that $D_d^i(t) < \infty$ and $\neg\mathcal{O}$ in Eq. (8) can be satisfied.

Let $D_d^i(t) < \infty$. According to the correct operation of ODVR, node i can select a successor for destination d at time t only if ARC is satisfied when the node processes a reply. However, whether node i selects a successor as a result of a proactive reply or a reply for an on-demand destination,

Eq. (4) is used to select a valid successor to destination d , which requires that $D_{ds_i}^i(t) < D_d^i(t)$. This implies that the correct operation of ODVR is a contradiction to $\neg\mathcal{O}$ stated in Eq. (8). Therefore, the theorem is true. ■

Theorem 5: ODVR provides loop-free routing at every instant.

Proof: The proof follows from Theorems 3 and 4. ■

VI. PERFORMANCE COMPARISON

We compare the performance of ODVR, AODV, and OLSR using simulations in the network simulator (ns3.24). We implemented ODVR and used the ns3 implementations of AODV and OLSR without modifications. OLSR uses HELLO messages to discover and check neighbor connectivity and Topology Control (TC) messages to disseminate link-state information throughout the network. To reduce signaling overhead, OLSR takes advantage of connected dominating sets. Some nodes are elected as multipoint relays (MPR's) and only MPRs forward TC messages, and only link-state information needed to connect MPR's is advertised in the network. Nodes do not fail and never lose the routing state they acquire during the simulation runs. This is done to illustrate the efficiency of ODVR relative to AODV and OLSR even when routing-table loops are unlikely to occur in AODV, and when no resets of sequence numbers occur in OLSR due to node failures.

Figure 1 shows the summary of simulation-environment settings for the three routing protocols. We considered different scenarios to study the behavior of routing protocols with respect to mobility and number of data flows. The scenarios were chosen to stress the signaling of all three protocols, rather than to attain good packet delivery rates.

Topology Area	500 x 1500 m
Number of Nodes	50
Transmission Range	250 m
Simulation time	1000 sec
Bandwidth	2 Mbps
Hello Interval	2 sec
TC Interval (OLSR)	5 sec
Update Interval (ODVR)	2 sec
Route TimeOut	3 sec
Queue Length	64
REQ Rate	10 req/sec
Max Retries	2

Fig. 1. Simulation parameters

We considered three metrics to analyze the performance of routing protocols: Data packet delivery ratio (DPDR), signaling overhead (overhead for short), and end-to-end delay. DPDR indicates the number of packets received by destination routers divided by number of packets sent by the source routers. The Overhead is the total number of routing message (in bytes) during the simulation. The signaling overhead in AODV includes its five types of packets: requests, replies, error messages, ACKs to replies, and HELLO messages. The signaling overhead in OLSR includes topology change (TC) messages and HELLO messages. The signaling overhead of ODVR includes all the request and reply messages it broadcasts. Average delay is the average delay incurred in transmitting a data packet until it reaches its destination, and

includes all delays incurred by buffering the packet during the route discovery phase, queuing at the output queues, propagation delays and transmission times.

The simulated network consists of 50 nodes spread uniformly and randomly in a 500×1500 m area at the beginning of the simulation run. Nodes start each experiment at random locations within the simulation area and move as defined by the random waypoint model, in which each node selects a random destination within the working area and moves linearly to that location at a predefined speed. After reaching its destination, it pauses for a specified time period (pause time) and then selects a new random location and repeats the same process. The scenarios include 25 data flows from 25 different sources to different destinations chosen randomly. Source nodes generate traffic at a rate of 15 packets per second each. The number of flows varies in the scenario in which we measure the effect of the number of flows.

A subset of nodes are selected as popular destinations. We assume that six nodes are popular and 75% of the traffic is targeted to the popular nodes. The popular nodes are selected randomly at the beginning of each simulation run. Traffic sources are on-off applications with on and off times of one second each, and generate packets of size 512 bytes at a rate of 15 packets per second. The data packets flow for 200 seconds and then new data flows are generated by selecting new sources and new destinations. New data flows are generated every 200 seconds after that.

We present simulation results for two versions of ODVR. The results labeled as ODVR-od correspond to the performance of ODVR when only on-demand routing is used, and the results labeled ODVR correspond to ODVR with integrated proactive and on-demand routing. As the simulation results illustrate, ODVR is far more efficient than traditional on-demand and proactive routing exemplified by AODV and OLSR, and the integrated routing approach in ODVR results in better performance than the exclusive use of on-demand signaling in ODVR.

A. Effect of Mobility

We study the effect of mobility on the performance of ODVR, AODV and OLSR by changing the pause time and speed of nodes. The speed with which nodes move is another parameter in mobility, and higher speed means more topology changes in the network. In the first experiment, the speed of the nodes in the mobility model varies from 1 m/s to 20 m/s to illustrate low and high mobility. Figure 2 shows the performance results as functions of node speed. The packet delivery rate decreases as the speed of nodes increases because more topology changes occur. Routes break due to topology changes and packets are dropped until the routing protocol finds a new route.

OLSR detects a link failure after failing to receive HELLO messages and sends TC messages to inform all routers of the topology so that new routes can be established. Given that TC messages contains all changes that take place between periodic updates, the signaling overhead increases to address

Fig. 2. Performance comparison as a function of node speed

Fig. 3. Performance comparison as a function of node pause time

Fig. 4. Performance comparison as a function number of sources

topology changes in the network. A link failure is detected in AODV by the absence of a number of consecutive HELLO messages, and a route discovery process is performed to establish new routes between sources and destinations. Link failures in ODVR can be detected by the absence of a number of consecutive gratuitous replies when proactive routing takes place, and a route discovery process is performed to establish new route between source and destinations for the proactive route. The discovery process for an on-demand destination is initiated in ODVR only when there is data traffic for the destination. Because of the delays incurred in detecting link failures and in establishing new routes after that, as router speed increases more and more data packets traversing failed routes end up being dropped.

As the results in Figure 2 indicate, ODVR incurs less overhead compared to AODV and OLSR. This is in part due to the fact that ODVR has smaller signaling packets compared to AODV, which needs the sequence number of both source and destination. More importantly, in OLSR, TC messages

must be disseminated by MPRs throughout the network, and each RREQ in AODV is flooded throughout the network. By contrast, a route request in ODVR is propagated only to a small part of the network, because a relay can reply if it is closer to the destination than the reference distance stated in the route request. This also results in shorter search times and smaller end-to-end delays in ODVR compared to OLSR and AODV. The signaling overhead in ODVR-od is even less than in ODVR, because only those nodes that have active route to destinations send route requests.

In the next scenario, we address the effect of pause times on performance. Longer pause times result in fewer topology changes. Figure 3 shows the packet delivery, delay, and signaling overhead as functions of node pause time. Pause times vary from 20 seconds (high mobility) to 200 seconds (low mobility), and the speed of individual nodes is chosen randomly between 0 and 20 m/s. The other parameters remain the same. The packet-delivery ratios of ODVR and ODVR-od are higher than the delivery ratios for AODV and OLSR.

ODVR and ODVR-od also attain shorter delays compared to OLSR and AODV, which means that ODVR converges faster and forwards packets faster than the other two protocols. As it was the case in the variable-speed scenario, ODVR and ODVR-od, introduce less overhead compared to AODV and OLSR. This is because ODVR uses smaller messages and fewer nodes transmit signaling messages. The extra overhead on ODVR compared to ODVR-od is due to the routing state maintained for all popular destinations by every node.

B. Effect of Number of Flows

Figure 4 shows the effect that data traffic and number of data flows have on performance. Sources and destinations are selected randomly and the number of source-destination data-flow pairs varies between 20 to 40. The delivery rate decreases and the average end-to-end delays increase for all the protocols as the number of flows increases. This is expected, because more traffic causes more congestion in the network, and a link failure affects more packet transmissions when the number of data flows increases. ODVR outperforms AODV and OLSR in both cases. The packet-delivery ratio of ODVR-od is close to the packet-delivery ratio in OLSR, but ODVR-od attains lower delays than OLSR.

The signaling of OLSR is independent of the number of data flows in the network; hence, the total signaling overhead remains almost the same. By contrast, signaling overhead increases in AODV, ODVR, and ODVR-od as the number of data flows increases, because more data flows means that more on-demand routes are established and the protocols have to fix more routes impacted by topology changes.

VII. CONCLUSIONS

We introduced ODVR, the first solution that supports on-demand or proactive loop-free multi-path routing on a per-destination basis using the same signaling for both routing modalities. ODVR introduces a novel approach for multi-path loop-free routing based on the use of reference distances stated in route requests and route replies. Routing state is established on demand or proactively by means of route replies that are either proactively originated by destinations, or are issued by relay nodes that are allowed to answer because their distances to destinations are smaller than the reference distances stated in route requests. We showed that ODVR is loop-free at every instant independently of how long routing state is maintained for a destination or the reliability of signaling messages. We compared the performance of ODVR with AODV and OLSR, which are well-known examples of on-demand and proactive routing. The simulation results show that the integrated routing approach introduced in ODVR results in much better performance than traditional on-demand and proactive routing. Our future work focuses on using clustering and other techniques in ODVR to enable integrated on-demand and proactive routing in large-scale networks.

ACKNOWLEDGMENTS

This work was supported in part by the Baskin Chair of Computer Engineering at UCSC.

REFERENCES

- [1] M. Abolhasan et al., "A Review of Routing Protocols for Mobile Ad Hoc Networks," *Ad Hoc Networks*, June 2003.
- [2] K. Bhargavan et al., "Formal Verification of Standards for Distance Vector Routing Protocols," *Journal of the ACM*, July 2002.
- [3] J. Behrens and J.J. Garcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," *Proc. IEEE INFOCOM '98*, 1998.
- [4] A. Boukersche, *Handbook of Algorithms for Wireless and Mobile Computing*, Chapman and Hall, 2006.
- [5] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)" *RFC 3626*, 2003.
- [6] M.S. Corson and A. Ephremides, "A Distributed Routing Algorithm for Mobile Wireless Networks," *Wireless Networks*, Jan 1995.
- [7] T. Clausen, "Comparative Study of Routing Protocols for Mobile Ad-hoc networks," Research Report RR-5135, INRIA, 2004.
- [8] J.J. Garcia-Luna-Aceves and H. Rangarajan, "A New Framework for Loop-Free On-Demand Routing Using Destination Sequence Numbers," *Proc. IEEE MASS '04*, 2004.
- [9] J.J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks," *Proc. of IEEE ICNP '99*, Oct. 1999.
- [10] J.J. Garcia-Luna-Aceves and S. Roy, "On-Demand Routing in Ad Hoc Networks Using Link Vectors," *IEEE JSAC*, March 2005.
- [11] Z. Haas and M. Pearlman, "Providing Ad-hoc Connectivity with the Reconfigurable Wireless Networks," *Proc. IEEE ICUPC '97*, Oct. 1997.
- [12] M. Joa-Ng and I. Lu, "Peer-to-peer Zone-based Two-level Link State Routing for Mobile Ad Hoc Networks," *IEEE JSAC*, Aug 1999.
- [13] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, Chapter 5, Kluwer Academic Publishers, 1996.
- [14] J. Jubin and J. Tornow, "The DARPA Packet Radio Network Protocols," *Proc. of The IEEE*, January 1987.
- [15] R. E. Kahn, "The Organization of Computer Resources into a Packet Radio Network," *IEEE Trans. Commun.*, Jan.1977.
- [16] Y. Ko and N.Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Wireless Networks*, Vol. 6, No. 4, 2000.
- [17] M. Marina and S. Das, "Ad Hoc On-Demand Multipath Distance vector Routing," *Wireless Commun. Mob. Comput.*, 2006.
- [18] J. Lee et al., "Hybrid Gateway Advertisement Scheme for Connecting Mobile Ad Hoc Networks to the Internet," *Proc. IEEE VTC 2003*, April 2003.
- [19] B. Liang and Z. Haas, "Hybrid Routing in Ad Hoc Networks with a Dynamic Virtual Backbone," *IEEE Trans. Wireless Comm.*, June 2006.
- [20] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *Mobile Networks and Applications*, 1996.
- [21] N. Navid, W. Shiyi, and C. Bonnet, "HARP: Hybrid Ad hoc Routing Protocol," *Proc. Int'l Symp. on Telecommunications '01*, 2001.
- [22] C. E. Perkins and P. Bhagwat, "Routing over Multihop Wireless Network of Mobile Computers," *Proc. ACM SIGCOMM '94*, 1994.
- [23] C. E. Perkins and E. M. Royer, "Ad-Hoc On-Demand Distance Vector Routing," *Proc. IEEE WMCSA '99*, 1999.
- [24] J. Raju and J.J. Garcia-Luna-Aceves, "A New Approach to On-Demand Loop-Free Multipath Routing," *Proc. IEEE ICCCN '99*, Oct. 1999.
- [25] V. Ramasubramanian et al., "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," *Proc. ACM MobiHoc '03*, June 2003.
- [26] S. Roy and J.J. Garcia-Luna-Aceves, "Node-Centric Hybrid Routing for Ad Hoc Networks," *Proc. 10th IEEE/ACM MASCOTS '02*, 2002.
- [27] R. Van Glabbeek et al., "Sequence Numbers Do Not Guarantee Loop Freedom—AODV Can Yield Routing Loops," *Proc. ACM MSWiM '13*, Nov. 2013.
- [28] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. ACM DIALM '99*, 1999.
- [29] X. Wu et al., "A Unified Analysis of Routing Protocols in MANETs," *IEEE Trans. on Communications*, March 2010.
- [30] M. Zhou et al., "The Proof of AODV Loop Freedom," *Proc. IEEE WCS '09*, 2009.