OWL: Towards Scalable Routing in MANETs Using Depth-First Search On Demand

Stephen Dabideen*
*Department of Computer Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
Email: dabideen@soe.ucsc.edu

J.J. Garcia-Luna-Aceves†*
† Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
Email: jj@soe.ucsc.edu

***Abstract*—Most routing protocols designed for MANETs to date employ breadth-first search (BFS), usually in the form of flooding of route requests or updates, to establish and maintain routes between source-destination pairs. This usually incurs significant overhead, which degrades the performance of the network. In this paper we present a new paradigm for routing protocols operating in MANETs, such that flooding is not required and paths from sources to destinations can be established on demand with time complexity comparable to that of flooding but with significantly less overhead. We introduce the concept of *ordered walk* as a depth-first based search (DFS) that does not rely on geographical or virtual coordinate information and is more efficient than mere random walks. Using the Ordered Walk Search Algorithm (OSA), we demonstrate the potential of using DFS as the building block of the signaling of MANET routing protocols. We introduce the OWL protocol (ordered walk with learning) as an example of efficient DFS-based routing in MANETs, and use simulation experiments to compare its performance against that of three well-known MANET routing protocols based on BFS (OLSR, DSR and AODV). The results show that OWL can achieve comparable these protocols while incurring up to ten times less overhead than AODV.**

I. Introduction

The proliferation of cost-effective mobile networking technology has fueled enormous interest in mobile ad hoc networks (MANETs), which in turn has lead to a very large number of proposals for MANET routing protocols. As varied as the routing proposals for MANETs are, none of the common proactive or on-demand schemes are very efficient under all traffic-load scenarios. We argue that, at least in part, this is due to the way in which signaling packets are disseminated in these protocols. In a proactive routing protocol, signaling packets must find each network node in order to update the node with the state of a link or destination, independently of whether or not the node needs a route that either uses the link or reaches the destination. Similarly, in an on-demand routing protocol, a route request is flooded to reach nodes that may or may not become part of the route sought by the route request. Section II summarizes prior work aimed at reducing the signaling overhead incurred in routing protocols for MANETs. As our brief survey indicates, the prior work has been aimed at reducing the number of nodes engaged in signaling of routing algorithms based on breadth-first search (BFS), reducing the amount of BFS signaling information that must be disseminated, or establishing virtual topologies that may be maintained more efficiently.

As an alternative, depth-first search (DFS) has been studied extensively in the past and many distributed algorithms for DFS have been reported (e.g., [1], [2], [3]). Surprisingly, however, DFS has not been used much to support the signaling of routing protocols in MANETs. To our knowledge, the only efforts that have addressed DFS focus on random walks [4], [5] or routing using location information (e.g., GPSR [6], WSR [7]).

BFS schemes would flood the network, or at least a very large number of nodes to find a destination that is far away. If too many nodes are performing BFS simultaneously, the routing overhead can saturate the network making it impossible to deliver any packets. Route computations based on DFS involve a much smaller number of nodes and can incur significantly lower overhead resulting in much less disruption than BFS and therefore more efficient routing as the size of the network and the number of flows increases. However, a DFS scheme may incur much longer delays in finding the desired routes to the destination.

The main contribution of this paper is to show that (a) routing in MANETs using DFS is not only feasible but can in fact be much more efficient than routing based on BFS schemes, and (b) efficient DFS-based routing does not need to rely on (geographical or virtual) coordinate information. Section III motivates the use of DFS instead of BFS as the basis for route signaling in MANETs in more detail.

Section IV presents the *ordered walk search algorithm* (OSA) and analyzes its potential in terms of its time complexity and the signaling involved. Instead of performing a search in a completely random manner, or assuming knowledge of the relative position of the destinations, OSA distributively constructs an approximated minimum-depth spanning tree. Then OSA searches this tree efficiently such that the number of search messages is minimized and the resulting path is of reasonable length. The only requirement of this search is two-hop neighborhood information. We demonstrate that, given some neighborhood information, a minimum-depth spanning tree can be approximated, and that this approximation converges to the actual minimum-depth spanning tree as the size of the known neighborhood increases.

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

Section V introduces an example of efficient routing in MANETs based on DFS without the need for location information, which uses OSA to establish routes on demand based on DFS and we call the *Ordered Walk with Learning* protocol (OWL).

Section VI shows the results of simulation experiments illustrating that OWL attains better performance than popular on-demand and proactive routing protocols based on BFS, with respect to the end-to-end delay of data packets and the signaling overhead incurred. Concluding remarks are presented in Section VII.

II. Related Work

Reactive routing schemes were developed to reduce routing overhead in mobile ad-hoc networks. Today, reactive (or on-demand) routing protocols have become synonymous with the flooding of route requests (RREQ) when a path needs to be established. While this approach may be the fastest solution in a network that is not bandwidth-limited, it leads to the *broadcast storm* problem as identified by Ni et.al. [8], especially in volatile routing environments. This inefficiency has been identified by many in the past, and several optimizations over this *blind* flooding have been proposed. These approaches include the use of an expanding ring search [9], the use of a connected dominating set heuristics to reduce the number of nodes retransmitting the flood packet [10], the use of geographical information to direct the flooding [6] and probabilistically reducing the number of retransmissions [8]. While these schemes alleviate the broadcast-storm problem, they do not address the need for flooding that is inherent in any BFS approach.

The most common strategy to reducing the overhead of route signaling in the past has been hierarchical routing, which dates back to the design of the DARPA packet radio network (PRNET), starting with the scheme proposed by Kleinrock and Kamoun [11]. Since then, there have been many other hierarchical routing schemes (e.g., [12], [13]). The limitation with hierarchical and hybrid routing schemes is that they do not address the inherent need for flooding and the need to update the affiliation of nodes to clusters or zones when nodes move away from their home clusters or clusters are partitioned into two or more components due to mobility.

Broadcast backbone networks ([14], [15]) have been used to mitigate the cost of flooding in the network. Instead of having every node flood the network, a connected subset of nodes which together dominate all the nodes in the network are responsible for forwarding overhead packets. Once such an infrastructure is in place, searching the network can potentially be more efficient than having all nodes flood search packets. The problem with this approach is in establishing and maintaining the infrastructure, especially in the face of mobility, where the cost can be comparable to flooding. Connected dominating sets make most sense in dense networks where there would otherwise be too much redundant broadcast of control packets but sparse networks would see little benefit from such infrastructure.

There have been only a few attempts to solve the problems incurred with flooding by using DFS instead of BFS. These approaches have focused on the use of *random walks* [4], [5] in which a route request starts at the source and travels along a single path found by consecutive random next-hop choices in search for the destination. The limitation of this prior work is that, as we discuss below, when packets traverse random walks, the communication complexity incurred in reaching destinations may be comparable to that of flooding, but with much longer delays.

Approaches that have improved over random walks in the past use location information for the routing of packets. (e.g., GPSR [6], WSR [7]). The many proposed schemes have made strong cases on performance benefits that can be attained but their limitations are that each node must know its location by some external means (e.g., GPS), and that the sources need to know the locations of their target destinations.

III. Motivation for Using DFS in Route Signaling

BFS has remained the most popular choice for route discovery in MANETs because, at least in principle, a well designed scheme based on BFS is the fastest approach to establishing the desired routes requiring on average $O(log_k N)$ time, where k is the average network connectivity. However, the price paid for this search speed is the signaling overhead incurred, which is $O(N)$. In practice, given that MANETs are bandwidth limited, this large communication complexity means that BFS may not succeed at establishing routes quickly, because signaling packets may suffer long queuing delays or even losses.

DFS may be a viable option to avoid the problems introduced by the broadcast storm associated with flooding. However, applying a DFS strategy to on-demand routing means that route requests are propagated from one node to a single successor and thus travel a single path from the source to the destination. Accordingly, using DFS in a complete graph in which each step is completely random would mean that the time required to find a path to a given destination is $O(N)$, which is the case of the destination being the last node searched, and the average complexity would be $O(N/2)$ for both time and number of messages. Hence, compared to BFS, DFS offers only a constant factor improvement but with a very large penalty in the time complexity it incurs. Clearly, DFS schemes based on random walks make sense only in networks where bandwidth is at a premium and delays in finding paths are not too important, which *may* be the case of some sensor networks with static topologies.

We advocate a new approach to DFS applied to on-demand routing in MANETs that takes advantage of two important characteristics of MANETs. Firstly, some local topology information is readily available to nodes due to the broadcast nature of radio links. In particular, a node can hear over time about the presence of neighboring nodes, and even the presence of its neighbors' neighbors. Secondly, MANETs

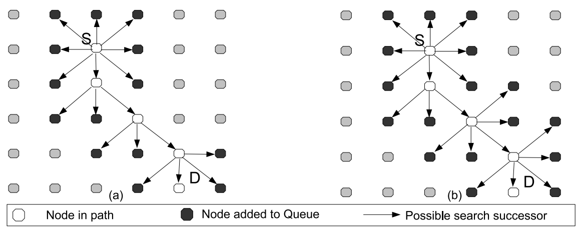

Fig. 1. DFS pruning in OSA

are not completely random and, more importantly, source-destination dialogues follow patterns of interest, which means that, over time, sources will be able to find destinations more effectively than by random walking if their searches choose "children" in the DFS tree based on this prior knowledge that incorporates information gathered from past searches, some of which may have even failed. Accordingly, our approach to using DFS in on-demand routing is based on the concept of "ordered walks," which we describe in the next section, and complement this DFS approach with learning gained from prior walks.

IV. Ordered Walk: A Hybrid Distributed Search Algorithm

With the ordered walk search algorithm (OSA), we aim to take advantage of the smaller time complexity of BFS and combine it with the low communication complexity of DFS to further improve the efficiency of the search through the use of known topology (i.e., path) information. The basic idea is to approximate the construction of a minimum-depth spanning tree rooted at the source (as in BFS) and then performing DFS on this tree. If there is information about the past location of the destination, then this can be used to guide the search. We call this an *informed search*, and while such information can help the search, it not necessary for the OSA.

A. Efficient Uninformed Searches

It is often the case in the establishment of new paths that no information about the destination is available at the node making a search decision. To make the search efficient, nodes can take advantage of topology information they have about their neighborhood. In particular, the number of nodes covered (in the known neighborhood) needs to be maximized while the number of nodes relaying the query needs to be minimized by the choice of next hop in the search. To achieve this, successive nodes in the search should have as few neighbors in common as possible. This is possible if two-hop neighborhood topology information is known by performing set comparisons between the current search node and its neighbors. This would usually favor choosing nodes physically far apart, rather than close together, and consequently results in shorter paths than a random walk, where a near node and a far node have the same probability of being the successor in the search. While set comparisons can be computationally intensive, it should not be an issue with modern technology.

At times it would be desirable to direct the search in a physical direction without knowing the relative positions of the nodes. A notion of direction can also be attained using two-hop neighborhood information. To guide the search away from a node, the neighbors of that node should not be allowed to be a successor in the search tree. To guide the search along the current trajectory, the neighbors of all previously searched nodes should be pruned from the search tree.

B. Approximating a Minimum-Depth Spanning Tree

Given global topology information, any node can construct a minimum-depth spanning tree in a connected network. The source becomes the root for the search. Nodes connected to the source are then at depth 1, their neighbors not yet included are at depth 2, and this process continues until all nodes are added.

Performing the search on a minimum-depth spanning tree results in paths with a shorter expected length than those obtained with a random walk search. Fortunately, such a tree can be approximated and constructed distributively requiring only two hop neighborhood information.

At each step in the search, a node must choose a successor that is at a greater depth in the search tree than itself. The neighbors of nodes which are already in the search path cannot be at a greater depth than the current node, and therefore such nodes should not be chosen as successors.

Figure 1 shows two executions of OSA. In the first case, global information is known and in the second case, only two-hop information is known. The difference between these two cases is that the search is guided away from the source with more information, which is the nature of DFS. With less information, fewer neighbors are pruned from the search tree, because they are not identified as being closer to the source.

C. The Ordered Walk Search Algorithm

The pseudocode for the *ordered-walk search algorithm* (OSA) is provided in Algorithm 1. It takes as input a graph G, a source node s, and the destination node d that is being searched, and delivers a path P from the source to the destination, should one exist. It also assumes that any node knows its n-hop neighborhood in the network, where $n \geq 1$.

OSA proceeds as DFS, using the approximated minimum-depth spanning tree. At each step in the ordered walk, a node must choose its successor in the search tree. An informed search would be prioritized if the relevant topology information were available, but if this is not the case, an efficient uninformed choice is made along the current trajectory of the search.

OSA uses a function N_k(x) that returns the set of nodes within k hops of node x, including node x itself and all of these nodes are precluded from the search tree at the current point of the search. Firstly, this prevents loops in the search as a node will never be able to choose a node already visited in the search. Secondly, all nodes in N_k(x) will be at a depth less than or equal to node x and therefore cannot be children of x in a DFS tree. In the algorithm ”j” represents the current depth of the search and ”n” is the span of the known neighborhood.

Algorithm 1 : OSA(G, s, d)

```
1: $Q \leftarrow \phi$
2: $P \leftarrow s$
3: $T \leftarrow N_1(s)$
4: $SORT(T)$
5: $Q \leftarrow T$
6: **while** $Q \neq \phi$ **do**
7:     $x \leftarrow$ first member of $Q$
8:     **if** $x = d$ **then**
9:         **return** $P$
10:    **end if**
11:    $T \leftarrow N_1(x)$
12:    **for all** $x_i \in P$ **do**
13:        j $\leftarrow$ number of hops of $x$ from $x_i$ in $P$
14:        **if** $j > n$ **then**
15:            $k \leftarrow n$
16:        **else**
17:            $k \leftarrow (n - j)$
18:        **end if**
19:        $T \leftarrow T - N_k(x_i)$
20:    **end for**
21:    $SORT(T)$
22:    Add $T$ to front of $Q$ in order
23:    Add $x$ to end of $P$
24: **end while**
25: **return** $\phi$
```

OSA also uses the SORT() function, which takes as input a set of nodes and places them in ascending order of distance to d, such that the node closest to d are searched first. The notion of proximity to the destination is based on past known locations of the destination. Nodes that are viewed as equidistant are placed in random order.

D. OSA Correctness

The following proofs show that OSA obtains finite paths from sources to destinations, if they exist, using only local information regarding a node's neighborhood and path information accumulated over time.

Theorem 4.1: OSA terminates for any finite graph with N nodes.

Proof: Once a node is visited in the search, it is added to P. For any node, $N_0(x) = x$, and thus any node in the path is never added to the queue Q, because it is never added to T in line 18 of Algorithm 1. Given that the number of nodes is finite and each node can only be searched at most once, the algorithm terminates when d is found or when there are no more nodes in the queue Q, which happens in at most N steps. ■

Theorem 4.2: For a finite graph G, OSA returns a path from the source s to destination d, if one exists.

Proof: It can be seen from Algorithm 1 that a node is not added to the search tree T at a particular instant if and only if it is either already in the search tree or in the known neighborhood $N_k(x)$ of a node already in the search tree.

Assume for contradiction that s and d are in the same connected component of the graph G and OSA terminates without returning a path from s to d. Then it must be the case that d was never added to the search tree T. If there was a path, it would have been found when d was searched. If d was never added to the search tree, then none of its neighbors could have been added to T, because then d would have been added when its first neighbor was searched. This same argument can be applied to the neighbors of d and can be repeated. Accordingly, it must be that the connected component containing d cannot contain any member of the search tree T and therefore cannot contain s, which contradicts the initial hypothesis. Therefore, the theorem is true. ■

E. OSA Complexity

For the sake of discussion, consider a network G with node degree k. If a path from a source s to the destination d is known, then the complexity of OSA is $O(log_k(N))$. This corresponds to the depth of a k-ary tree, and is in fact the best upper bound on the run time of any search algorithm; however, it is not really a search if a path is already known. It is well-known that the complexity of a centralized DFS algorithm is $O(V + E)$, which in this case is reduced to $O(N + k * N)$, which is $O(N)$.

The accuracy of the ordering determines how far the actual search deviates from this worse-case value and how close it approaches the optimal value of $O(log_k(N))$.

We note that the search space in OSA is reduced by a factor of k for every step that is known to be correct. This is illustrated in Figure 2, which shows OSA being executed on a binary tree. The direction of the arrows shows known paths. It is clear that, with every step taken towards the destination, the number of nodes in the search tree is halved. For pure DFS or pure BFS only one node is removed from the search tree in any give step. In real MANETs, the situation is not as simple as in Figure 2, because nodes would form a mesh instead of a tree and the nodal degree is likely to be variable and greater than two. However, the reduction in the search space at each step will still be significant.

V. OWL: DFS WITH LEARNING

OSA relies on topology information to enhance the performance of the depth first search. To demonstrate the effectiveness of OSA, we present the *ordered walk with learning* (OWL) routing protocol, which uses DFS to establish and repair paths from the source to the destination with minimal

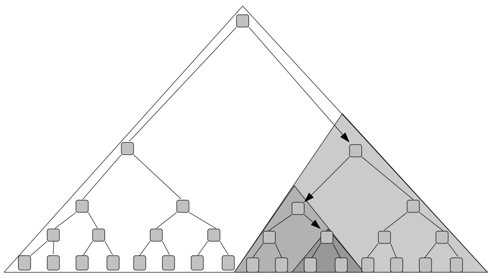

Fig. 2. An ordered walk example

signaling overhead and fast convergence. OWL is an on-demand routing protocol, which means that paths are established only when needed. However, there is a proactive component that is used to maintain up-to-date two-hop neighborhood information.

A. Routing Information

OWL relies on two-hop neighborhood information and this is achieved using periodic Hello packets. Hello packets contain the one hop neighbors of a node and are sent periodically with the frequency being dependent on the mobility of the nodes. The default hello packet interval in OWL was set to thirty seconds and is then additively increased and decreased based on the number of changes in the neighborhood. If the neighborhood is stable, then there is less need to update the information.

OWL uses two structures to store the information necessary for route discovery and routing data packets. The first is a neighbor table which stores each of the node's neighbors and keeps a list of each of their one-hop neighbors (as advertised in Hello Messages).

The second structure is a destination table. It stores the latest sequence number and corresponding distance to that destination. The information in the destination table is updated using path information carried in route requests and route replies. Each node also maintains a *preferred neighbor* to each active destination it knows. This preferred neighbor is the first neighbor from which the node receives the latest route reply for the given destination, and therefore the closest neighbor to that destination. Associated with each preferred neighbor is the number of times that neighbor has been used as the preferred neighbor in a DFS since the last successful search (we call this variable PNUse).

B. Route Establishment and Maintenance

Route requests (RREQs), route replies (RREPs), route errors (RERRs) and hello messages are the basic building blocks of OWL. These packets are used in the same way as many on-demand routing protocols.

RREQs are initiated by a source node with the intention of finding a destination. These packets carry the destination address, the source address, the current sequence number of the source, the distance to the source and the path traversed by the RREQ. These packets are relayed (as long as the TTL is greater than zero) to a single node determined using OSA. The precise manner in which route requests are handled are given in Algorithm 2. The basic idea is to forward the route request to a valid preferred neighbor if one exists, else use known topology information to select a successor which is at greater depth in the search tree and has the most unexplored neighbors.

RREPs can only be issued by the destination upon reception of a RREQ. The RREPs are sent along the reverse path to the source and carry the destination address, source address, current destination sequence number and the path traveled by the RREP. Nodes store the distance and sequence number carried in the RREQs and RREPs in their routing table. This distance information is useful in ordering the nodes for future searches in the event of link failures.

This ordering information can be used for up to two future searches (as determined by the variable PNUse), after which it expires and is removed from the routing table. This value can be changed depending on the mobility of the network. Whenever a node receives a RREP with a higher sequence number, it refreshes PNUse, so the preferred neighbor can be used two more times, regardless or prior use.

An ordered walk would fail if the TTL expires or a leaf node is reached. A RERR is sent back to the source in both cases, and the source then initiates an ordered walk in a different direction, with the notion of relative direction being derived from two-hop neighborhood information as perviously discussed. This would result in a different branch of the approximated minimum depth spanning tree being searched in a DFS manner and increases the probability of discovering the destination since it maximizes the number of new nodes being searched.

In a mobile environment, link failures will be inevitable and if the path being used no longer exists, the last node before the point of path failure can make up to two attempts to repair the path using neighbors with a known route to the destination. If no such neighbor exists, or the path using the selected neighbor(s) fail, a RERR is sent to the source, which then starts a new ordered walk. This time, there will be some ordering of the nodes with respect to the destination from the previous search and this information will be used in the OSA algorithm to make the DFS more efficient. The further from the source the failure occurs, the easier it will be repaired, because the search tree is significantly pruned with each successful step.

C. Path Reinforcement

Hello packets are also used to advertise ordering to destinations for which the node or one of its immediate neighbors is on the path to the destination. When a path to the destination is set up, the nodes along that path and their neighbors that overhear the route reply designate themselves as *active* for that path for up to two hello intervals after first overhearing the route reply. This reinforces the ordering to the destination and provides alternate routes for routing data and future searches.

D. Learning

We have shown how to make efficient uninformed DFS decisions, but it is even better to make informed search decisions when possible. To accomplish this, nodes need to learn about the topology, but this must be done with little cost.

OWL uses a couple of mechanisms to gain topology information. Periodic "Hello" messages, which contain a list of one-hop neighbors and destinations for which it is active are broadcasted by all nodes in the network and this is used to provide two-hop neighborhood information as in the Neighborhood Aware Source Routing (NSR) [16]. Also, once a node finds the destination once it uses the past location information to guide future searches.

In a mobile environment, past position of the destination will become outdated (more so as the search approaches the destination). Nonetheless, some of the information may be useful.

Algorithm 2 : HandleRREQ($RREQ$)

```
1: $P \leftarrow RREQ(Path)$
2: $PN \leftarrow PreferredNeighbor$
3: $ProcessPath(P)$
4: if $(Node = Destination)$ then
5:     $InitiateRREP()$
6:     return
7: end if
8: if (TTL $= 0$ or hasNoNewNeighbors) then
9:     $InitiateRERR()$
10:    return
11: end if
12: if PN and PNUse $> 0$ and PN $\notin$ P) then
13:    $PNUse = PNUse - 1$
14:    $RelayRREQ(PN)$
15: end if
16: $O \leftarrow BestOrderedHop(P)$
17: $RelayRREQ(O)$
18: return
```

At the start, most nodes may have no ordering for an intended of destination. After each search some nodes are ordered with respect to other nodes, be it the desired destinations or not. This ordering is important to the success of OWL. Promiscuous listening allows for ordering of nodes near the paths set up. Due to mobility, distance information is assumed to be valid for up to two searches per sequence number.

Over time, the ordering in the network can become robust, thus reducing the search time. This claim is substantiated by the simulation results presented in Section VI.

E. Learning and Mobility

WIth ordered walks, the search is directed towards the last known position of the destination. This is beneficial as long as the destination is in the same position or at least very close to that position. In a mobile environment, this is not always the case and, for this reason, a known ordering with respect to the destination is valid only for a few (set to three in OWL), searches. Such an event indicates substantial movement of the destination and an ordered walk is initiated and preformed using uninformed routing decisions.

However, in most of today's MANETs, nodes do not move very quickly and are likely to gradually move from one neighborhood to another. Once the known path is broken, it is likely to find an alternate route that contains most of the nodes as the original route and a few new nodes. The results indicates that OWL performs well in the face of mobility.

F. OWL versus Random Walk

We argue that Ordered Walks are better than Random Walks in the context of MANETs.

The expected path that results from an ordered walk should be shorter than that of a random walk. The TTL used in OSA forces an upper bound on the resulting path length. The fact that the OSA is performed on an approximated minimum depth spanning tree ensures that the TTL is sufficiently large to find a path without allowing the path to be excessively long. Furthermore, successive nodes in the ordered walk would be, on average, physically further apart than that of random walks. In a random walk all neighbors have an equal probability of being the successor in the search, but ordered walks favor successors that are further apart since these nodes would have neighbors in common. Because the average distance between successive nodes is larger, the average number of hops will be smaller for ordered walks than for random walks.

The expected number of search messages in ordered walks should be smaller than that of random walks. Each successive node in an ordered walk is chosen so as to maximize the number of nodes not yet covered by the search. For any given number of search messages, a random walk would cover at most, as many nodes as covered by an ordered walk. If the destination is randomly chosen, then each node is equally likely to be the destination. Therefore, the more nodes covered by a search, the greater the probability of finding the destination. Hence, for the same number of search messages, an ordered walk would have a greater than or equal to that of a random walk.

Furthermore, because the OSA operates on a minimum-depth spanning tree, the search is guaranteed to venture into to new neighborhoods of nodes as it is forced to move away from the source with each step. In a random walk however, there is no such restriction and the search can move through all the nodes in a same neighborhood, and therefore proceed much slower.

G. A Simple Example

Consider the simple network in Figure 3, where node S needs to send data to node node D and there is no previous routing information about D in the neighborhood of S. Node S chooses a neighbor with the fewest neighbors in common with itself. In this example, it first chooses A, which in turn selects B as its successor because they have they fewest neighbors in common. Node B does not have any neighbor that is not a child of a node already in the search path because nodes X and Y are neighbors of node A. Node B cannot proceed to a

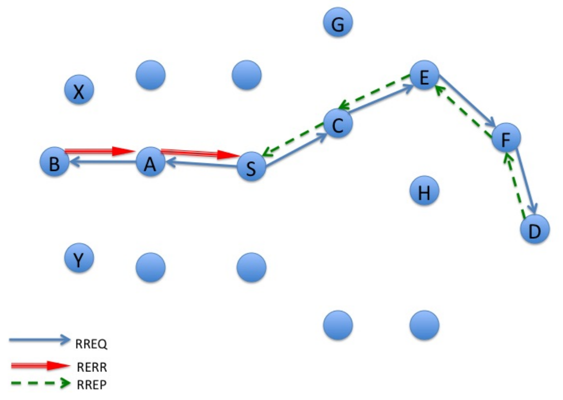

Fig. 3. An OWL example

greater depth of the search tree so it must sent a route error to node S.

Upon receiving this route error, node S must choose a new successor with as few neighbors in common with both itself and node A (in an attempt to guide the search in a different direction). In this example, node S chooses node C which then chooses node E. Once the RREQ arrives at node E, a path to the destination is found because the destination, node D, is in node E's two-hop neighborhood. Node E will forward the RREQ along the known path to verify that it still exists. Node D then initiates a route reply in response to the route request and this route reply travels along the same path as the route request. Nodes close to this path, such as nodes G and H, overhear the route replies and become active for destination D. Accordingly, they record their preferred neighbor (which is E for the case of node G) and set the number of future searches for which the information is useable to 2. Node H becomes active for destination D and therefore includes the current distance and sequence number for D in its Hello Messages which will provide node C with an alternate (and in this case a shorter) route to node D though node H.

VI. Experimental Results

We compared the performance of OWL with that of representative protocols for on-demand and proactive routing based on BFS in MANETs. OLSR was used as an example of proactive routing, and AODV and DSR were used as examples of on-demand routing. Routing in AODV and OLSR is incremental, meaning routing decisions are taken on a hop by hop basis, and DSR uses source routing. There are more recent protocols that outperforms OLSR, AODV and DSR; however, the unavailability of standardized code of such protocols prevents meaningful comparisons with OWL. Furthermore, our comparison highlights the ability to attain on-demand routing using a DFS approach without relying on location information, and illustrates the fact that such an approach can render comparable results to those attained with the traditional BFS scheme used in OLSR, AODV and DSR, but with only a fraction of the signaling overhead.

A. Simulation Environment

Three scenarios were used in the simulations. Scenario A, was designed to rigorously test the performance of the protocols in a dynamic environment with volatile links. Scenario B, uses a greater radio range to add more stability to the links and create more multi-path opportunities. Scenario C, tests the performance in larger networks under heavy loads.

Scenario A is consists of 100 nodes uniformly distributed in a grid of size 1000m x 1000m with the transmission range of the radios set to 150m. This choice of parameters satisfies the minimum standards for rigorous MANET protocol evaluation as prescribed by Kurkowski, et al [17] as it results in an *average shortest path hop count* [17] of 4.03 and *average network partitioning* [17] of 3.9%. Other relevant simulation parameters are summarized in Table I and Table II. This scenario ensures that packets travel several hops from source to the destination and thus tests the robustness of the protocols. The mobility model chosen was that of random waypoint with minimum speed of 1m/s and maximum speed of 10m/s with a pause time of 30s. Each experiment lasted for 900s.

Parameter	Value
Simulation time	900s
Node Placement	Uniform
Mobility Model	Random Waypoint
Min-Max Speed	1-10m/s
Pause time	30s
Propagation model	Two-ray
Physical layer	802.11
Antenna model	Omnidirectional
MAC Protocol	802.11 DCF
Data Source	constant bit rate (CBR)
Number of packets per flow	400
Packet rate	4 packets per second

TABLE I
Constant Simulation Parameters

Parameter	Scenario A	Scenario B	Scenario C
Number of Nodes	100	100	400
Number of Flows	10	10	100
Network diameter	9.4 hops	7.07	14.14
neighbor count	7.06 nodes	12.56 nodes	12.56 nodes
Transmission range	150 m	200m	200m
Simulation Area	1000m x1000m		2000m x 2000m

TABLE II
Varied Simulation Parameters

Ten nodes were chosen at random to be sources of CBR flows and ten nodes were chosen at random to be destination of these flows. Care was taken to avoid the case where a node was both the source and destination of any particular flow. There were no restrictions on nodes being multiple sources, multiple destinations or a source of one flow and a destination of another. Each source would send a maximum of 400 packets of size 512 Bytes at a rate of 4 packets per second. The start time of each flow was randomly determined using a uniform distribution and was within the duration of the experiment.

TABLE III
SIMULATION RESULTS

	Scenario A				
	Delivery Ratio	Latency	Net Load	R/R	Avg. Path Length
AODV	0.62±0.09	0.048±0.014	48.4±4.7	33.86	3.74
DSR	0.20±0.08	0.62±0.52	5.8±0.6	0.64	6.1
OLSR	0.32±0.08	0.07±0.02	238.6±2.3	NA	3.6
OWL	0.68±0.12	0.076±0.031	3.6± 0.4	11.1	4.5
	Scenario B				
	Delivery Ratio	Latency	Net Load	R/R	Avg. Path Length
AODV	0.85 ± 0.07	0.041 ± 0.010	25.5 ± 1.4	21.61	3.2
DSR	0.42± 0.20	0.50 ± 0.55	4.3 ± 2.83	0.56	5.6
OLSR	0.56 ± 0.07	0.044 ± 0.01	131.2 ± 1.79	NA	3.2
OWL	0.91±0.06	0.051±0.020	1.81±0.12	8.0	4.1
	Scenario C				
	Delivery Ratio	Latency	Net Load	R/R	Avg. Path Length
AODV	0.42 ± 0.03	0.14 ± 0.02	67.1± 6.04	130.6	4.8
DSR	0.02 ± 0.01	1.5 ± 0.1	18.5 ± 1.2	0.68	9.3
OLSR	0.09 ± 0.02	0.12 ± 0.01	180.3± 1.3	NA	4.9
OWL	0.49±0.05	0.17±0.04	6.5±0.32	61.5	5.2

Using a time-based seed, 20 random scenarios were generated with the above specifications and the results were used to compare the performance of the protocols. The large number of randomly generated scenarios were used to avoid bias in the results. Each series of random numbers was generated using the *lrand48* command in C and using the current time as a seed for the random number generator.

Scenario B is similar to A, except that the transmission range is increased to 200m. The purpose of this was to increase the average neighbor count from 7.06 to 12.6 nodes. Also, the increased range makes some of the links more stable as nodes take longer to move out of range of each other. The value of the average shortest path would certainly be less than 4 nodes while the average network partition would be less than 5%.

Scenario C is similar to Scenario B with the main difference being that the area is four times larger and there are 100 CBR flows, each of which beginning at a random time. The objective of this scenario is to demonstrate the performance of the protocols in larger networks. Each flow in Scenario C has 200 packets and sends them at a rate of 2 packets per second.

The differences between the three scenarios are summarized in Table II.

Five metrics were used to evaluate and compare the performance of the protocols and they are discussed below. The hello interval in OWL was set to 30 seconds and for comparability, the TC interval in OLSR was set to 30 seconds and the hello interval in OLSR was set to 15 seconds.

The mean and a 95% confidence interval were obtained. The simulation results for the four routing protocols are summarized in Table III.

B. Delivery Ratio

Delivery ratio is the fraction of packets that arrive at the corresponding destination by the end of the simulation. The reasons packets are not delivered to the destination is dependent on the specific protocol. Data packets can be dropped at the source if they cannot find a route to the destination. Data packets in transit can be dropped upon link failure, especially if the protocol does not perform local route repair. In OWL, when a link fails, a node can try up to two different paths to send the data packet along if such information is available. If no alternate routes are available the packet is dropped.

For Scenario A, the performance in terms of delivery ratio is poor for all protocols. This is due to the dynamic nature of the network, and the possibility of network partitions. OWL performs marginally better than AODV, and these two perform significant better than OLSR and DSR. In Scenario B, where the likelihood of nodes being unreachable is much smaller than in Scenario A due to the increased radio range, all the protocols delivered a greater number of packets. In Scenario C, all protocols experience much smaller delivery ratio as is expected. As the average length of the path increases the frequency at which the path breaks increase and more packets are lost. This performance reflects the need for protocols which scale better as the size of the network increases. OLSR and DSR experience relatively poor performance. This can be due to the dynamic nature of the network and source routes and topology information quickly becoming invalid. When this happens packets will be dropped due to a lack of a path to the destination and this is reflected in the results.

An important result is that the DFS approach discovers paths from the source to the destination and deliver comparable, if not more, packets. The length of the paths the packets travel from sources to the destinations and slightly larger than those obtained from BFS schemes, but this is mitigated by the overall performance.

C. Network Load

D. Latency

Latency is the average end-to-end delay experienced by a data packet. The main factor that affects latency is the time taken to find a path from the source to the destination. For a proactive protocol like OLSR, this time is expected to be very small, because there is no delay involved in setting up a path

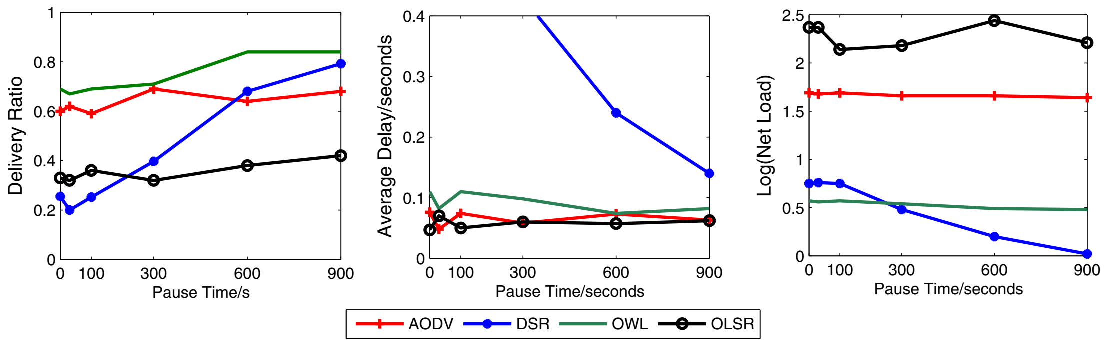

Fig. 4. Performance variation with Pause Time

when a data packet arrives at a node, given that one is already known. The time it takes to repair links also affects latency. In mobile networks, link failures are inevitable. Protocols should be able to repair links with minimal delay to ensure the timely delivery of packets. This is usually done with local route repair, but local route repairs may not always succeed in obtaining new routes, and a failed local route repair incurs a greater delay penalty compared to sending an immediate route error (RERR) to the source upon link failure.

BFS would, at least according to intuition, lead to faster route discovery and faster route repair than depth first search. However, in networks with multiple flows the flooding of search packets can result in the broadcast storm problem and this results in lost packets which can lead to fail attempts of route discovery. If there are too many nodes flooding the network, it may be possible to saturate the network which will lead to all packets being dropped. With DFS on the other hand, only a small fraction of the network is involved in route computations at any given time which significantly reduces the load in the network and fewer packets are lost.

OWL makes up to two attempts of local route repair. When a local repair is successful, a RERR is not sent to the source. If the intermediate node does not already know an alternate paths, or the alternate paths are broken, it sends a RERR to the source. Only nodes along the path of the RERR and those within radio range learn of the link failure encoded in the RERR. Therefore, the topology information stored in nodes is not uniform in OWL. However, for the purposes of routing RREQs, absolute accuracy is not necessary. A node is still likely to be in the same vicinity of its last known location, and this is sufficient for the purposes of guiding RREQs.

In terms of latency, the relative performance of the four protocols is the same in all scenarios, with AODV enjoying the lowest latency and DSR incurring significantly greater delays. Flooding the network is guaranteed to quickly find a path if one exists. OWL may require several ordered walks before a route to the destination is found. OLSR also incurs lower latencies than OWL, because paths are available when data packets arrive at sources and relays.

Net load is the number of overhead packets (RREQs, RREPs, RERRs, Hellos, etc.) which were initiated or forwarded divided by the number of data packets sent. This takes into account packets that were sent into the network and were dropped or did not make it to the destination for any reason. Net load gives an indication of the average number of overhead packets needed to send a single data packet from the source to the destination. The number of overhead packets needed would depend on the volatility of the network. The more frequently links are broken, the more control packets would be needed to establish new paths.

The advantage of ordered walks become quite clear when this metric is considered. OWL requires significantly fewer overhead packets than the other protocols, because it efficiently searches the network without flooding and therefore sources find their destinations without having to search every node. Even in the large network of 400 nodes, OWL incurs ten times less overhead than AODV which vouches for its scalability. DSR also appears to incur relatively small overhead; however, this is due to the fact that DSR drops many data packets due to stale source routes stored in nodes, and those packets do not cause RREQs to be sent, even though they should have been issued.

E. Mobility

We now show OWL's performance as the mobility of the network is varied. Using the parameters from Scenario A, which is more taxing on all protocols, we vary the pause time of the nodes from 0 seconds to 900 seconds. The results are illustrated in Figure 4. The results are as expected, the protocols performs better as mobility decreases. In the graph showing the average end to end delay, the line for DSR does not appear because it was beyond the range of the axis. It is important to note that the learning mechanisms in OWL still work effectively with shorter pause times.

F. Towards Scalability

OWL requires far fewer route requests and route replies to deliver a comparable number of packets than routing protocols based on traditional BFS searches. While the route discovery process may take marginally longer, the DFS approach used in OWL avoids the broadcast storm problem and causes fewer disruptions to other flows in the network. This becomes particularly important in larger networks as the number of flows increases. Considering all the results together, it becomes clear than ordered walks can be used to replace the flooding mechanism of reactive routing.

VII. Conclusion

We argued that most routing schemes designed for MANETs rely on some form of BFS, and presented the ordered walk search algorithm as a replacement for flooding. An ordered walk is a distributed approximation of DFS that is aided by known topology information to reduce the search tree. We introduced the OWL routing protocol as an example of the great potential for using DFS in route signaling for MANETs. We presented the results of simulation experiments illustrating that OWL provides comparable or better delivery and end-to-end delay than AODV, DSR and OLSR, but with significantly less signaling overhead. The use of ordered walks, as presented in this paper, is a promising tool in achieving minimum-signaling routing in MANETs. While OWL is a step towards achieving this goal, more work is needed to fully exploit the advantages of ordered walks in routing protocols.

VIII. Acknowledgements

Work partially sponsored by the U.S. Army Research Office (ARO) under grant W911NF-05-1-0246, by the National Science Foundation under grant CNS-0435522, and by the Baskin Chair of Computer Engineering.

References

[1] B. Awerbuch, "A new distributed depth-first-search algorithm," *Information Processing Letters, Vol. 20, No. 3, pp. 147–150*, 1985.

[2] I. Cidon, "Yet another distributed depth-first-search algorithm," *Information Processing Letters, Vol. 26, No. 6*, pp. 301–305, 1988.

[3] S. Makki and G. Havas, "Optimal distributed algorithms for constructing a depth first search tree," *Proc. ICPP*, 1994.

[4] H. Tian, H. Shen, and T. Matsuzawa, "Randomwalk routing for wireless sensor networks," *Proc. PDCAT*, 2005.

[5] S. D. Servetto and G. Barrenechea, "Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks," *Proc. WSNA*, 2002.

[6] B. Karp and H. Kung, "Greedy perimeter stateless routing for wireless networks," *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 243–254, August 2000.

[7] U. Acer, S. Kalyanaraman, and A. A. Abouzeid, "Weak State Routing for Large Dynamic Networks," *Proc. MobiCom*, September 2007.

[8] S.-Y. Ni, Y.-C. Tseng, Y. S. V. Hen, and J.-P. Sheu, "The boradcast storm problem in mobile ad-hoc networks," *Proc. MobiCom*, 2001.

[9] A. Segall, "Distributed network protocols," *IEEE Transactions on Information Theory*, 1983.

[10] M. A. Spohn, "Domination in graphs in the context of mobile ad hoc networks," Ph.D. dissertation, University of California, Santa Cruz, 2005.

[11] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks: Performance Evaluation and Optimization," *Computer Networks, Vol. 1, No. 3, pp. 155-174*, January 1977.

[12] M. G. Guangyu Pei and T.-W. Chen, "Fisheye state routing in mobile ad hoc networks," *ICDCS Workshop in Wireless Networks and Mobile Computing*, 2000.

[13] R. Ramanathan and C. Santivanez, "Hazy sighted link state (hsls) routing: A scalable link state algorithm," *BBM Technical Memo BBN-TM-130I, BBN Technologies*, 2001.

[14] K. Xu, X. Hong, and M. gerla, "An ad hoc network with mobile backbones," *IEEE Internation Conference on Communications*, 2002.

[15] I. Rubin and P. Vincent, "Topological synthesis of mobile backbone networks for managing ad hoc wireless networks," *IEEE International Conference on Management of Multimedia Networks and Services: Management of Multimedia on the Internet*, 2001.

[16] M. Spohn and J. J. Garcia-Luna-Aceves, "Neighborhood aware source routing," *Proc. ACM MobiHoc*, 2001.

[17] S. Kurkowski, T. Camp, and W. Navidi, "Minimal Standards for Rigorous MANET Routing Protocol Evaluation," *Technical Report MCS 06-02, Colorado School of Mines*, 2006.