An Efficient Path Selection Algorithm for On-Demand Link-State Hop-by-Hop Routing

Soumya Roy and J.J.Garcia-Luna-Aceves

Computer Engineering Department

University of California

Santa Cruz, CA 95064

soumya, jj@cse.ucsc.edu

Abstract—

Traditional routing protocols based on link-state information form a network topology through the exchange of link-state information by flooding or by reporting partial topology information and compute shortest routes to each reachable destination using a path-selection algorithm like Dijkstra's algorithm or the Bellman-Ford algorithm. However, in an on-demand link-state routing protocol, no one node needs to know the paths to every other node in the network. Accordingly, when a node chooses a next hop for a given destination, it must be true that the next hop has reported a path to the same destination; otherwise, packets sent through that node would be dropped. In this paper, we present a new path-selection algorithm that unlike traditional shortest path algorithms, computes shortest paths with the above on-demand routing constraint.

I. INTRODUCTION

To minimize control overhead in mobile ad-hoc networks, on-demand routing protocols (e.g., dynamic source routing (DSR) [1], ad-hoc on-demand distance vector (AODV) [2] routing, temporally ordered routing algorithm (TORA) [3], source-tree on-demand adaptive routing(SOAR) [4])) maintain paths to only those destinations to which data must be sent and the paths to such destinations need not be optimum.

In link-state routing protocols meant for mobile ad-hoc networks, partial link-state information can be used for computation of paths to destinations, because all nodes need not have to compute paths to every other destination. Hence, each node may not know how to reach every other node in the network, even when all nodes remain connected. For correct hop-by-hop routing, every node that receives a data packet for forwarding should have a correct route for the destination. Therefore, while computing routes, a node should be allowed to choose a neighbor as the next hop for certain destinations only if that neighbor has advertised routes for those destinations; otherwise, packet forwarding would be incorrect. Unfortunately, the Bellman-Ford algorithm or Dijkstra's algorithm do not place any constraint for the computation of routes, and new path selection algorithms are needed to account for the on-demand routing constraint. In this paper, we present such a new path selection

algorithm that computes shortest paths with on-demand routing constraint.

Section II describes how the path selection algorithm should be specified for on-demand link-state routing protocols. Section III describes the details of the path selection algorithm. Section IV proves that the given path selection algorithm is correct, i.e. it correctly computes the best path to reach any destination under the on-demand routing constraint. Section V concludes the paper.

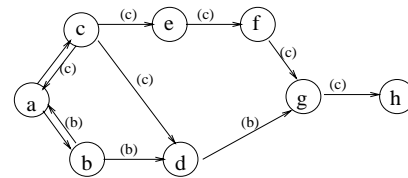## II. PATH SELECTION FOR ON-DEMAND LINK-STATE ROUTING

Fig. 1. Example explaining the requirements of a new path selection algorithm for routing protocols using link-state information on-demand

Considerable effort has been devoted for using distance-vector information or path-vector on-demand (e.g., AODV [2], DST [5], DSR [1]), but not much work has been done exploring the use of link-sate information on-demand in routing. Most of the link-state routing protocols that have been devised for mobile ad-hoc networks are pro-active, like OLSR [6], STAR [7], FSR [8], TBRPF [9] while the source tree on-demand adaptive routing (SOAR) [4] is the only protocol reported to date that uses link-state information on-demand.

The key idea in SOAR is for wireless routers to exchange *minimal source trees*, consisting of the state of the links that are in the paths used by the routers to reach only *important* destinations. Important destinations are active receivers of data packets, relays, or possible relays. Minimal source trees can be reported incrementally or atomically, and updates to individual links in source trees are validated using sequence numbers. A wireless router uses its outgoing links and the minimal source trees received from its neighbors to get a partial view of the

		Form Approved
Report Documentation Page		OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**
4. TITLE AND SUBTITLE **An Efficient Path Selection Algorithm for On-Demand Link-State Hop-by-Hop Routing**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **4**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

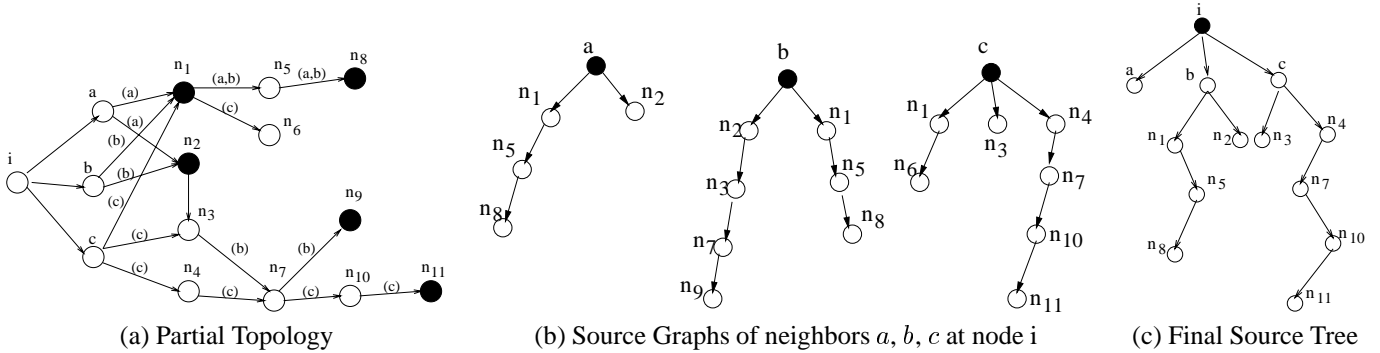(a) Partial Topology   (b) Source Graphs of neighbors a, b, c at node i   (c) Final Source Tree

Fig. 2. Partial Topology (a) and Source Graphs of neighbors (b) are the inputs to the path selection algorithm, (c) is the output i.e. the final source tree for packet forwarding

network topology and computes its source tree using a local path selection algorithm.

A key difference in the local path-selection algorithm needed in on-demand link-state routing protocols compared to traditional shortest-path algorithms used in proactive routing protocols is that each router advertises only those links necessary for data forwarding, rather than all links in the network. Fig. 1 illustrates this situation. The figure shows the partial topology at node a, which has been built based on its outgoing links and the inputs from its two neighbors, b and c. Each link in the partial topology lists the neighbors of node a that have reported the links. For example, node b has only reported the link $b - d$, while link $g - h$ has been advertised by node c only. A traditional path-selection algorithm (e.g., Dijkstra's shortest path first algorithm or the Bellman-Ford algorithm) finds the shortest path from a source to any destination. However, when any of these algorithms is run on the partial topology at node a, node b will be selected as the next hop to reach node g or node h. Node b has not advertised any path to node h, and node b may not know about node $h's$ existence. In such a case, packets for node h forwarded to node b will be dropped. The valid path along which packets for node h can get forwarded is $[acefgh]$ and the path selection algorithm should find that valid path, rather than the shortest path. Correct hop-by-hop routing would then be possible.

Of course, if source routing is enabled as in DSR, then the packets can always be forced on the shortest path $[abdgh]$, without the intermediate routers setting up any route for destination h. However, our focus is on correct route establishment at each node for hop-by-hop packet forwarding, rather than source routing because source routing requires modification of current IP forwarding mechanism and incurs extra overhead in each data packet. Accordingly, we need a new path-selection algorithm for on-demand link-state routing, which should satisfy the following property for choosing a route for a destination.

Property A : *All links in the computed path to a destination through a neighbor should be advertised by the neighbor itself.*

Because our objective is to choose the shortest among the possible valid paths, the path selection algorithm should satisfy the following rules :

Rule 1: If p_i^j be the path computed to reach j at i through neighbor k, then $p_i^j = [l_i^k p_k^j]$, i.e., there should be a path p_k^j, which has been reported by k to i (l_i^k : link from i to k), which becomes a subpath of p_i^j. This rule follows from Property A.

Rule 2: There can be several potential paths that satisfy Rule 1. However, for final route computation paths with the smallest length should be chosen.

Rule 3: Only a single path is chosen to reach any node in the network. For example, in Fig. 1 there are two valid paths for node g, namely $abdg$ to reach node d and $acefg$ while reaching node h. Path $abdg$ through neighbor b has to be selected because that is the shortest path to node d. In that case, node h will become unreachable, because the only valid path to reach h has been advertised by neighbor c and neighbor b is the next hop for the predecessor of node h.

The optimal path selection algorithm should choose the appropriate predecessor and successor with the objective to minimize the number of nodes becoming unreachable, i.e., to maximize the number of nodes for which routes can be obtained. Finding the optimal solution is an NP-complete problem[1] and hence we propose a heuristic.

III. DETAILS OF THE ALGORITHM

Dijkstra's algorithm or the Bellman-Ford algorithm cannot be used directly as the path selection algorithm for on-demand link-state routing because they do not satisfy Rule1. The proposed path selection algorithm for selecting shortest paths in on-demand link-state routing protocols consists of selecting the valid paths for a destination, and then choosing the shortest among the valid paths with the objective of having finite cost paths for maximum number of destinations. The following two sections describe each phase of the path selection process.

A. Finding Valid Paths for a Destination

The first step of the proposed algorithm is to rebuild the source trees advertised by each neighbor based on the links

---

[1] Proving that the problem of finding the optimal solution is an NP-complete is beyond the scope of this paper

Node	{next_hop, predecessor, distance}	min_dist	best_options	c_x	bnh	p_x	[selec_best_options, count, {nh, dist}]	[nh, pred, dist]
a	{a, i, 1}	1	1	n_1, n_1	a	i	X	[a, a, 1]
b	{b, i, 1}	1	1	n_1, n_2	b	i	X	[b, b, 1]
c	{c, i, 1}	1	1	n_1, n_3, n_4	c	i	X	[c, c, 1]
n_1	{a, a, 2}, {b, b, 2}, {c, c, 2}	2	3	n_5, n_6	a, b, c	a, b, c	[2, 3, {a, 3}, {b, 3}]	[b, b, 2]
n_2	{a, a, 2}, {b, b, 2}	2	2	X	a, b	a, b	[2, 1, {a, 1}, {b, 1}]	[b, b, 2]
n_3	{b, b, 2}, {c, c, 2}	2	1	X	c	c	[1, 1, {c, 1}]	[c, c, 1]
n_4	{c, c, 2}	2	1	n_7	c	c	[1, 4, {c, 4}]	[c, c, 1]
n_5	{a, n_1, 3}, {b, n_1, 3}	3	2	n_8	a, b	n_1, n_1	[2, 2, {a, 2} {b, 2}]	[b, n_4, 3]
n_6	{c, n_1, 3}	3	1	X	c	n_1	[1, 1, {c, 1}]	[NULL, NULL, ∞]
n_7	{b, n_3, 4}, {c, n_4, 3}	3	1	n_9, n_{10}	c	n_4	[1, 3, {c, 3}]	[c, n_4, 3]
n_8	{a, n_5, 4}, {b, n_5, 4}	4	2	X	a, b	n_5, n_5	[2, 1, {a, 1} {b, 1}]	[b, n_5, 4]
n_9	{b, n_7, 5}	5	1	X	b	n_7	[1, 1, {b, 1}]	[NULL, NULL, ∞]
n_{10}	{c, n_7, 4}	4	1	n_{11}	c	n_7	[1, 2, {c, 2}]	[c, n_7, 4]
n_{11}	{c, $n_1$0, 5}	5	1	X	c	n_{10}	[1, 1, {c, 1}]	[c, n_{10}, 5]

Fig. 3. Table depicting the step-wise execution of the path selection algorithm. (*min_dist* $\Rightarrow$ minimum length of the best paths to a destination x, *best_options* $\Rightarrow$ total number of neighbors that have advertised paths of smallest length, $c_x \Rightarrow$ nodes which are directly reachable along the least-cost paths through node x, bnh (best next hop) $\Rightarrow$ neighbors that have advertised least-cost paths to node x, $p_x \Rightarrow$ predecessors through which node x can be reached, *selec_best_options* $\Rightarrow$ number of best choices among the shortest paths, $count \Rightarrow$ total number of nodes farther from a node that can be potentially included in the final source tree when nh is chosen as next hop)

in the given topology and the list of neighbors, which have reported each link. The source trees become the inputs for the path selection algorithm. The complexity of this step is $O(nd^2)$, where n is the number of nodes in the network and d is the neighbor density. Fig. 2(a) shows the partial topology at node i, where corresponding to each link the neighbors who have advertised that link have been listed. Fig. 2(b) shows the source trees of neighbors a, b and c rebuilt from the partial topology of node i. Table 3 shows the step-by-step execution of the path selection algorithm.

Next, the valid paths for each destination in the form of tuples $\{distance, nexthop, predecessor\}$ are determined by doing depth first traversal on the source graph of each neighbor. Atmost d valid paths are theoretically possible for each destination, depending on whether a neighbor has advertised a path for it or not. The complexity of a depth-first traversal is $O(n)$ for a tree with n nodes; therefore, the total complexity is $O(nd)$ for d neighbors. Column 2 of Table 3 shows the different valid paths possible for each node in the network of Fig. 2 in the form of $\{nexthop, predecessor, distance\}$ tuples. For example, corresponding to node n_8 the valid paths are: (1) through neighbor a, with predecessor n_5 and distance four, and (2) through neighbor b, with predecessor n_5 and distance four. By making paths belonging to the source trees of neighbors eligible for path selection, Rule 1 is automatically satisfied.

B. Choosing The Best Routes

After computing all valid paths for a particular destination, the least-cost paths to any destination are only considered for the final route selection, thereby satisfying Rule 2. That process requires two operations: (a) finding the minimum cost among the possible options ($O(nd)$), and (b) selecting the paths which are of least cost ($O(nd)$). Columns 5, 6, and 7 of Table 3 show the least-cost paths possible based on the source graphs of neighbors a, b and c. As shown in Table 3, the least-cost path to reach node n_7 among the two valid paths through neighbors b and c (as shown in Table 3) is through neighbor c, with the predecessor being n_4 and the direct children being n_9 and n_{10}.

The next step of the operation is to choose among the valid least-cost paths only those paths aggregation of which will form the source tree with maximum number of nodes in it. This operation can be formally described as follows.

Problem statement: *Given a set of least-cost paths by which the nodes in a network can be reached, the successor in the route for each destination in the network topology has to be determined such that finite-cost paths for the maximum number of nodes can be obtained.*

Finding the optimal solution is an NP-complete problem. Hence, a heuristic is proposed for the final path selection. The heuristic can be divided into two distinct operations, the steps of which have been depicted in Fig. 4. During the first operation, choices are made regarding predecessors and next hops for nodes, starting from the farthest nodes towards the ones nearer to the source, while maximizing the count of children at each node. The count of children at a node x refers to the total number of nodes farther from x that can be included in the final source tree when a particular next hop is chosen. To illustrate, if node n_1 is reached via c, then the total number of children which can be reached from n_1 is two, while if either a or b is chosen the count becomes three. Hence, c is excluded as a next hop choice for n_1 and the total number of paths of minimum cost, (referred to as $selec_best_options$ in Table 3) becomes two.

During the second operation, traversals are made from the source towards the nodes farther away and final selections of next hops among $selec\_best\_options$ are made. For example, node n_2 can be reached via neighbor a, as well as via neighbor b. Any one of a or b can be chosen as $next\_hop$. However, if any of the nodes is the previous successor, then to prevent route flapping that node will be chosen automatically as the current successor. Node b also becomes next hop for node n_1's successors n_5 and n_8. Node n_5 chooses node b, since its only possible predecessor n_1 has chosen that also. Node n_6's only path is through node c with predecessor n_1, but n_1 is only reachable through node b. Therefore, node n_6 would be excluded from the final tree. If node b would have advertised link n_1-n_6, then n_6 could have been included in the final source tree. Through

message exchange in the routing protocol, intermediate nodes b and n_1 can be forced to advertise a path to n_6. Fig. 2(c) shows the minimal source tree computed drawn from the final results in the last column of Table 3.

The complexity of the above two operations is $O(nd+2nd^2)$. Accordingly, by considering each step taken for the path selection the complexity of the entire path selection algorithm becomes $O(nd^2)$. In comparison, the complexity of Dijkstra's algorithm is $O(n^2)$.
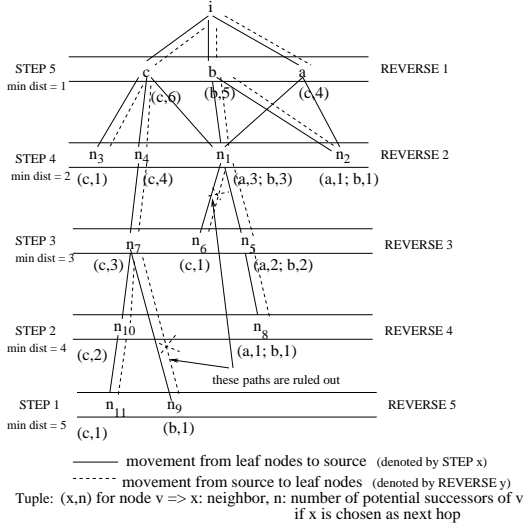
Fig. 4. Depiction of the process of final selection of next hop for each node

IV. CORRECTNESS OF PATH SELECTION

Theorem 1: **If** $ST_i(=\ (V', E'))$ **is the final computed source tree, the distance to each** $v \in V'$**, according to** ST_i **is the shortest path based on the given topology** ($G = (V, E)$) **under the constraint of Rule 1.**

Proof: Let d_v be the distance to vertex $v \in V'$ according to ST_i and $\delta^*(i, v)$ be the shortest distance from i to v under the constraint of Rule 1, given a topology at node i. Using depth-first traversals through each neighbor's source graph, we compute $\delta^n(i, v)$, i.e., the distance from i to v through neighbor n.

Let $\qquad d_v^n = \delta^n(i, v) \quad \forall n \in N_i$ (N_i : neighbor set).

According to the algorithm, the best next hops ($bnh \in N_i$) for any destination v are chosen such that the following is satisfied,
$$d_v^{bnh} = \min[\delta^n(i, v)] \quad \forall n \in N_i.$$
Because of Rule 1, $d_v^{bnh} = \delta^*(i, v)$ for each bnh. In ST_i, snh is the selected next hop for reaching v and snh is selected from the set of bnhs, which implies that $d_v = d_v^{snh} = \delta^*(i, v)$. ∎

Theorem 2: **Using a given topology if one run of path selection algorithm does not yield optimal solution, the optimal solution can be obtained by message passing with certain relevant nodes.**

Proof: Let v be any node that is included in ST_i, i.e., $v \in V'$. Let nh_v be the next hop to reach v according to ST_i. This implies that $count_v.nh_v = \max[count_v.n_i] \quad \forall n_i \in N_i$, where $count_v.n$ is the total number of nodes in the subtree rooted at v and advertised by neighbor n.

Let us assume that a node u has been left out of ST_i (i.e., $u \in (V - V')$). Let an upstream node of u according to one possible least cost path from i to u (through neighbor nh_v) be v and let v be the last node in the path to u that belongs to ST_i.

An on-demand routing protocol based on link-state information can be defined such that, a node could ask its neighbor(s) to enact a form of *forced routing* along the path $[nh_v, ..., v]$ such that $v, ..., nh_v$ would be forced to advertise to i the subtree ($SUBT^v$), rooted at v, containing path to u and that has been excluded from ST_i.

Let c be the total nodes in $SUBT^v$ excluding v. Then the new count value, $count'_v.nh_v = (count_v.nh_v + c) > \max[count_v.n_i] = \max[count'_v.n_i]$, because forced routing through nh_v increases $count_v.nh_v$ only.

This implies that nh_v would be selected as the next hop for v, and any node u left out in ST_i before would be included, hence giving the optimal solution. ∎

V. CONCLUSIONS

Traditional shortest-path algorithms work correctly only when all nodes maintain routes to all destinations. However, in on-demand routing protocols, a node need not maintain routes to all destinations. Accordingly, such algorithms as Dijkstra's shortest path first or the Bellman-Ford algorithm cannot be applied for computing paths when there is an on-demand routing constraint dictating that a neighbor can be chosen as the next hop to a destination only if that neighbor has advertised a path to the destination. In this paper, we have presented a new path selection algorithm that enables correct path computation in routing protocols based on the exchange of link-state information on-demand and on hop-by-hop packet forwarding.

REFERENCES

[1] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," *Mobile Computing*, 1994.

[2] C. E. Perkins and E. M. Royer, "Ad Hoc On-Demand Distance Vector Routing," in *Proc. of IEEE WMCSA'99*, New Orleans, LA, 1999.

[3] V. D. Park and M. S. Corson, " A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," in *Proc. IEEE Infocom'97*, Kobe, Japan, April 1997.

[4] S. Roy and J.J. Garcia Luna Aceves, "Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks ," in *IEEE Infocom*, Anchorage, Alaska, 2001.

[5] J. Raju and J. J. Garcia-Luna-Aceves, "Efficient On-Demand Routing Using Source-Tracing in Wireless Networks," in *IEEE Globecom*, 2000.

[6] P. Jacquet and et al., "Optimized Link State Routing Protocol," in *draft-ietf-manet-olsr-04.txt*, March, 2001.

[7] J.J.Garcia-Luna-Aceves and M.Spohn, "Transmission-Efficient Routing in Wireless Networks using Link-State Information," *ACM Mobile Networks and Applications Journal*, vol. 6, pp. 223–238, 2001.

[8] M. Gerla, X. Hong, and G. Pei, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks," in *draft-ietf-manet-fsr-02.txt*, December 17, 2001.

[9] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis, "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)," in *draft-ietf-manet-tbrpf-05.txt*, 1 March 2002.