An Extended Dynamic Source Routing Scheme in Ad Hoc Wireless Networks*

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
jie@cse.fau.edu

Abstract

In this paper we consider a multipath extension to the dynamic source routing *(DSR) protocol proposed by Johnson and Maltz, an on-demand routing protocol for ad hoc wireless networks. This extension keeps two node disjoint paths between the source and the destination of a routing process without introducing extra overhead. Several optimization options are also considered. Simulation is conducted on the success rate of finding node disjoint paths.*

Key words*: Ad hoc wireless networks, dynamic source routing (DSR), multipath routing, optimization, simulation.*

1 Introduction

Recent advances in technology have provided portable computers with wireless interfaces that allow networked communication among mobile users. The resulting computing environment, which is often referred to as mobile computing, no longer requires users to maintain a fixed and universally known position in the network and enables almost unrestricted mobility. An *ad hoc wireless network* [1] is a special type of wireless mobile network in which a collection of mobile hosts with wireless network interfaces may form a temporary network, without the aid of any established infrastructure or centralized administration. The applications of ad hoc wireless networks range from civilian (e.g., distributed computing, sensor networks) to disaster recovery (search-and-rescue), and military (battlefield).

An ad hoc wireless network can be represented as a simple directed graph $G = (V, E)$, where V is the vertex set representing a set of wireless mobile hosts (also called nodes) and E is a set of edges representing a set of links (channels). An edge (v, u) from v to u indicates that host u is within the wireless transmitter range of host v. Such a graph is also called *unit disk graph*, or simply, unit graph.

*This work was supported in part by NSF grant CCR 9900646 and grant ANI 0073736.

Routing is a process of sending a message from one mobile host in the network to another (it is also called *unicast*). Routing protocols for ad hoc wireless networks normally call for *mobility management* and *scalable design*. Mobility management is done through information exchanges between moving hosts in the ad hoc wireless network. In general, when information exchanges occur frequently, the network maintains accurate information of host locations and other relevant information. However, frequent information exchanges can be costly, because they consume communication resources including bandwidth and power. With less frequent information exchanges, these costs diminish but there is more uncertainty about the host's location. Scalable design (one that works for large size networks) requires both routing protocols and resource consumptions to be scalable.

Routing in the ad hoc wireless network poses special challenges because of its infrastructureless network and its dynamic topology. The tunnel-based triangle routing of mobile IP [9] works well if there is a fixed infrastructure to support the concept of the "home agent". However, when all hosts move (including the home agent), such a strategy cannot be directly applied. Traditional routing protocols for wired networks, that generally use either *link state* [5] or *distance vector* [2], are no longer suitable for ad hoc wireless networks. In an environment with mobile hosts as routers, convergence to new, stable routes after dynamic changes in network topology may be slow and this process could be expensive due to low bandwidth. Routing information has to be localized to adapt quickly to changes such as hosts movement.

Routing protocols for ad hoc wireless networks can be roughly divided into *proactive* and *reactive*. In proactive routing, each host continuously maintains complete routing information of the network. Both link state and distance vector belong to proactive routing. The reactive scheme, on the other hand, invokes a route determination procedure only on demand through a query/reply approach. Dynamic source routing protocol (DSR) [4] is a reactive routing pro-

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

tocol. The source determines the complete path for each routing process. The approach consists of *route discovery* and *route maintenance*. Route discovery allows any host to dynamically discover a route to a destination host. Each host also maintains a route cache in which it caches source routes that it has learned. Unlike regular routing-table-based approaches that have to perform periodic routing updates, route maintenance only monitors the routing process and informs the sender of any routing errors. Without the use of routing tables to keep track of routes, mobility management and scalable design objectives can be relatively easy to achieve.

However, the efficiency of DSR depends largely on the "hit ratio" of route cache. That is, the probability a route to the destination exists in the cache. When a miss occurs, the system has to invoke a relatively expensive route discovery process via flooding. In this paper, we propose a novel approach to reduce the frequency of invoking the routing discovery process. The idea is to keep two node disjoint routes to each destination in the route cache. One route is designated as the primary and the other the backup. When the primary route fails, the alternative route can be used without invoking a route discovery process. Two node disjoint routes are constructed during the route discovery process by restricting the way the query packet is flooded in the network in the query phase and the way routes are stored in the route cache of each node in the route reply phase. Other than the color mark of each node along a route and a dirty bit associated with each route in the route cache, the construction process does not introduce additional overhead, compared with the regular route discovery process. DSR also provides an option of constructing edge disjoint paths, so that an alternate path can be used when the primary path fails. However, too many paths are maintained in DSR in a trivial matter, without any regard to their ultimate usefulness.

The paper is organized as follows: Section 2 gives preliminaries and related work on multipath routing. Section 3 proposes the extended dynamic source routing. Several optimization options are studied in Section 4. Section 5 shows some simulation results on the effectiveness of the proposed approach in finding two node disjoint paths. Section 6 concludes this paper.

2 Related Work

Dynamic source routing protocol (DSR) [4] is a reactive routing protocol. Unlike other protocols, DSR requires no periodic packets of any kind at any level within the network. The approach consists of route discovery and route maintenance. Route discovery allows any host to dynamically discover a route to a destination host. Each host also maintains a route cache in which it caches source routes that

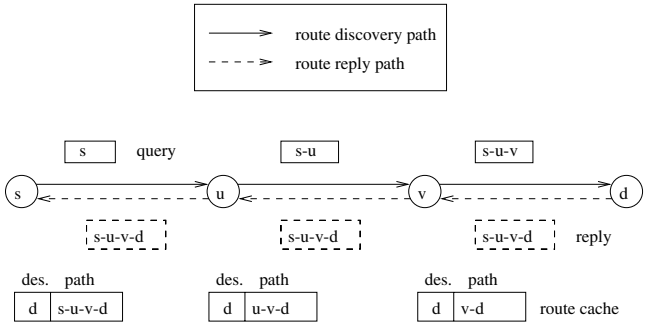

Figure 1. The query/reply phase in the route discovery process.

it has learned. The source determines the complete path for each routing process. When the source cannot find a route to the destination from its cache, it initiates route discovery which consists of two phases: query and reply. First, a *query* packet initiated from the source floods the network in seeking of a route to the destination. When the destination receives a query packet, it replies with a *reply* packet that copies the route from the query packet and traverses it backwards (or via a different route back to the source). Route information to the destination are stored in route cache of each node as learned from the reply packet. Note that more than one route reply packet may be generated at the destination. However, the destination can control the number of reply packets. Route reply packets can also be generated at an immediate node where the route to the destination exists in the route cache. In this case, the number of reply packets is difficult to control. Route maintenance maintains source routes to arbitrary destinations. Details of route maintenance are of no interest here and they will not be discussed further.

Figure 1 shows a routing example in a simple linear network. The source s first sends out an query packet. A sequence of network hops is accumulated during the query process. Once the packet reaches the destination d, d replies with a reply packet that copies the complete route $s-u-v-d$ and traverses the route backwards. Route information to d is stored in route cache of each intermediate node (including the source). For example, u includes route $u-v-d$ in its route cache. Since links in the network can be unidirectional, the reply phase may have to use a different route to send out the reply packet to the source.

Multipath routing is one of the favorite mechanisms to balance network traffic and to provide fault tolerance. Mathematical analysis has proven that splitting the traffic over the two paths is more efficient, provides shorter delays overall [3]. Zaumen and Garcia-Luna-Aceves [10] proposed a multipath routing using diffusing computation for

packet switching networks. In general, multipath routing is based on constructing first either *edge disjoint paths* or *node disjoint paths* with former being a special case of latter. The multipath routing is also captured in OSPF [6], a link state routing protocol in Internet, by the notion of "equal cost multipath", where traffic should be split equally between all the equal cost paths.

Among protocols for ad hoc wireless networks, the Temporally Ordered Routing Algorithm (TORA) [8] maintains a directed acyclic graph (DAG) for each destination with the destination being the sink of the DAG. In this way, edge disjoint paths are maintained for each destination.

Nasipuri, Cartaneda, and Das [7] proposed a multipath extension to DSR by constructing node disjoint paths. The destination keeps a record of the first arrived packet (including the complete path record initiated from a particular source). The subsequent packets arrived will be discarded until a packet with a node disjoint path with respect to the first one arrives. All subsequent packets are discarded. Unlike the approach in [7] where the destination selects two node disjoint paths, *our approach generates two node disjoint paths during the query phase of the route discovery process by restricting the way the query packet is flooded*.

3 Extended Dynamic Source Routing

In DSR, route discovery tries to find a path to a destination by first sending out a query through flooding. If the route discovery is successful the source receives a route reply packet listing a sequence of network hops through which it may reach the destination. In extended dynamic source routing (EDSR), the probe/reply phases are constructed in a special way so that two routes (if they exist) are constructed as the result of a route discovery. Specifically, one route is called the *black route* with all nodes along the route colored black and another one is called the *white route* with all nodes along the route colored white. A node that has a white color, a black color, or both colors is said to be *marked* (for a particular route request); otherwise, it is *unmarked*. Initially, all nodes except the source node are unmarked. The source is initially marked both white and black. Color white is said to be complement to color black, and vice versa. We denote $white^{'} = black$ and $black^{'} = white$. In order to detect duplicate route requests received, each host in the ad hoc wireless network maintains a list of $(source, destination, request\_id, color)$ triple that it has received. $request\_id$ is a sequence number maintained at the sender. Each intermediate node can be colored only once $(source, destination, request\_id)$ and the destination can be colored twice with one for each color.

Each sender s initially broadcasts two requests (s, d, id, $black$) and (s, d, id, $white$). Both requests have the same id, i.e., the same sequence number maintained locally. When an intermediate host v (excluding destination) receives a route request packet (s, d, id, $color$), it processes the request as follows:

- If v has been marked for (s, d, id), then the request will be discarded.
- If v is unmarked for (s, d, id) and v received the request for the first time, then the request is kept for Δ units of time before marking v for (s, d, id).
  - If v does not receive a route request packet of (s, d, id, $color^{'}$) before the expiration of Δ units of time, v is marked $color$ for (s, d, id).
  - If v receives a route request packet of (s, d, id, $color^{'}$) before the expiration, v is randomly marked either white or black without further delay.

Once it is marked, v performs one of the following actions:

- If a route to destination d can be determined at v, a *route reply* packet to the source of the route discovery is sent from v. A route can be determined if v is the destination or a route white a matching color from v to the destination exists in the route cache of v (such a route is called *cached route*).
- Otherwise, v appends v's host's own address to the route record in the route request packet, and forwards the request with the committed color.

The destination node accepts the first black route and the first white route (i.e., the destination node is colored twice). Then a route reply packet to the source is sent from the destination node. A route reply packet (sent from either the destination or an intermediate node with a cached route to the destination) keeps the complete route information. Route information to the destination are stored in route cache of each intermediate node (including source) as learned from the reply packet.

In the above algorithm, the use of Δ is to ensure that the selection of a color is a random process, like flipping a coin. Note that the color of each node for each request is a conceptual notion. Color information for a particular request can be removed once the corresponding query phase is completed.

Like DSR, EDSR uses its caches routes to avoid propagating a route request packet received all the way to the destination. The color of the selected route in the route cache is important in this case. Suppose each route can select any cached route without considering its marked color, a white route may stop at u with a black cached route $u - v - d$ and a black route may stop at v with a black cache route $v - d$

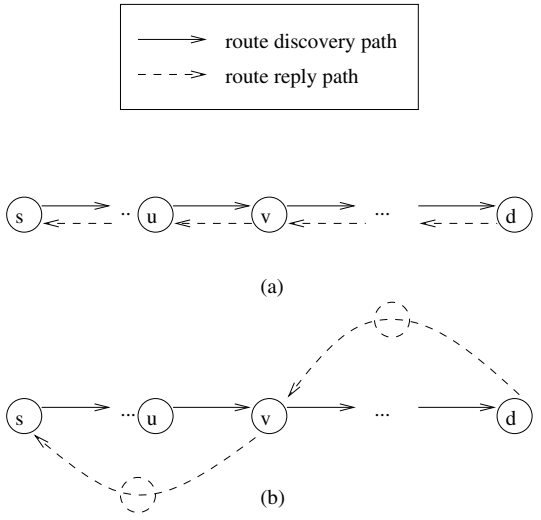

Figure 2. The relationship between route discovery path and route reply path: (a) route reply path is the reverse of route discovery path and (b) route reply path is not the reverse of route discovery path.

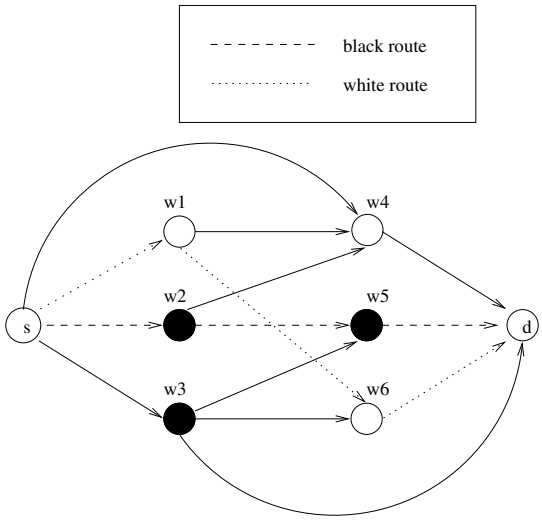

Figure 3. A sample multipath routing.

which is a subpath of $v - u - d$ (see Figure 2 for such an example). In this case, two routes are not node disjoint.

Therefore, a black route request initiated from s to d will stop at v only if a *black route* from v to s exists in the route cache. Similarly, a white route request initiated from s and with the same id will stop at u only if a *white route* from u to s exists in the route cache. When a node v has both black and white routes (e.g., v has initiated a route discovery process to the current destination), then pick the one with the matching color.

Figure 3 shows a sample ad hoc wireless network with eight nodes. Initially, all nodes excepts source s are unmarked. To simplify the discussion, it is assumed that the route cache of each node is empty, i.e., each route will not send its route reply before reaching the destination node. Assume that among neighbors of source s, w_2 and w_3 are marked black and w_1 and w_4 are marked white. The subsequent broadcasts mark w_5 black (by w_2) and w_6 white (by w_4). Finally, destination d is marked white by w_6 and black by w_5. The resultant node disjoint paths are $s - w_1 - w_6 - d$ and $s - w_2 - w_5 - d$ as shown in Figure 3.

During the route reply phase, route information to destination d is stored in route cache of each intermediate node (including source) as shown in Figure 1. When a network has unidirectional links, not all nodes on the route discovery path can cache route information during the route reply phase, since a different route might be used in the reply phase. Figure 2 (b) shows such an example. It is clear that only nodes (v in the figure) that are on both route reply path and route discovery path can cache route information. Such nodes are called *cache nodes*. In the subsequent discussion, we assume that links in the network are bidirectional. Therefore, the route reply path is the reverse of the route discovery path and all nodes on the path are cache nodes. When a route discovery path is disconnected (because of host movement) during the route reply phase, the path is simply discarded. Note that when a node caches route information, $(color, TTL)$ is associated with each route. That is, each route to a particular node in the route is colored. For example, in Figure 1, when node s caches $s - u - v - d$ with a white color, routes $s - u$, $s - u - v$. and $s - u - v - d$ are all considered white. TTL (Time-To-Live) is a timed value. When time-out occurs, the corresponding route entry is removed from the route cache.

4 Optimizations

When route cache is used, the node disjoint property of black and white routes is difficult to enforce. Consider an example shown in Figure 4 where route query for white route stops at v since v has a white cached route to destination d. Similarly, route query for black route stops at u with a matching cached route at u. However, the matching route requirement is still not enough. In Figure 4, paths $s - v - d$ and $s - u - d$ are not necessarily node disjoint. In fact, subpaths $s - v$ and $s - u$ are guaranteed to be node disjoint since they are generated from the current query phase, but not necessarily for $v - d$ and $u - d$ because they can be generated from two different sources, say s' and s'' (see Figure 4). Therefore, $v - d$ and $u - d$ are not guaranteed to be

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

node disjoint even though they have different colors. Two paths are said to be *overlapped* if they have the same destination and they share at least one intermediate node that is not the source of each path. In Figure 4 paths $u-w-d$ and $v-w-d$ are two overlapped paths and node w is called an *overlapped node*.

To ensure the property of node disjointness, we consider in the following two solutions. In the first solution, overlapped paths are prevented from being generated; while in the second solution, overlapped paths are allowed, but they are tagged with "dirty bits".

In the first solution, we forbid cache nodes to cache route information whenever two overlapped paths of different colors are detected: *During the route reply phase for a path from d to s, if at intermediate node v (excluding source s) a route of the other color to destination d exists in the route cache of v, any cache node on the path after v (closer to source s than v) is not allowed to cache route information*. Note that the detecting node v is still allowed to cache the route information.

Consider the example shown in Figure 4. Suppose route information of white route $s^{'}-v-w-d$ initiated from $s^{'}$ has been cached on each node on the path before the reply phase of black route $s^{''}-u-w-d$ (initiated from $s^{''}$) starts, If w is the *last overlapped* node (the closest overlapped node to the destination), any node between w and d (including w) will be able to cache route information, but not the other nodes on the path.

In the second solution, instead of disallowing caching routing information, an extra bit called "dirty bit" is added to each route record. The dirty bit is set after the last overlapped node w is detected in the route reply phase. In the example of Figure 4, nodes in subpath $s^{''}-u-w$ (excluding w) still cache route information with their dirty bits being set. Note that the notion of dirty bit is different from the one used in traditional cache memory. A route with its dirty bit set indicates the existence of another route of different color to the same destination and that these two routes are not node disjoint.

The dirty bit alone does not distinguish the second approach from the first one. The difference occurs when each dirty bit is associated with a TTL to indicate the life time of the dirty bit. When the TTL of a dirty bit expires, the dirty bit is removed and the corresponding path becomes clean. Note that each route still has its own TTL. TTL of a dirty bit of black (white) route at node u is the maximum TTL value of any white (white) route to the same destination d cached at a overlapped node in subpath $u-w$ (see Figure 4). In other word, TTL is the longest time a overlapped white (black) cached route can live!

To generate two node disjoint routes, the route discovery protocol can be slightly modified as follows:

- The white route accepts only *clean white route* (route with its dirty bit unset) while the black route accepts *clean black route* (route with its dirty bit unset).
- A clean white (black) route is found at v if v is destination d or a clean white (black) route from v to d exists in the route cache of v.

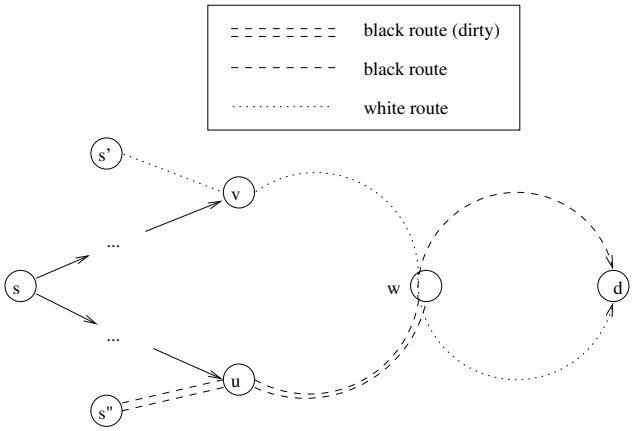

Figure 4. Overlapping paths.

Note that if the above approach fails, the routing policy can be dynamically switched back to the regular DSR where any route, without considering its color and dirty bit status, to the destination can be selected. However, information about route color and dirty bit status is still maintained in case the routing policy is switched back to EDSR later.

When route cache is used more than one black (white) route can be generated at the end of the query phase. A special delay mechanism similar to the one used in DSR can be used to avoid too many simultaneous replies for both white and black routes. This is done by selecting a delay period which is a monotone function of h, the number of hops for the route to be returned to the source. That is, the host that is farther from the source has a longer delay than the one that is closer to the source. In this way, a black (white) route can terminate its route reply if this node can infer that the source has already received a reply giving an equally good or better route (e.g., this node has initiated or passed a shorter route to the same source).

5 Simulation

We have conducted a simulation study without using route cache. The simulation is conducted in a 100 $\times$ 100 2-D free-space by randomly allocating a given number of hosts ranging from 10 to 100. Four radii of transmitter ranges are considered: 15, 25, 50, and 75. Assume that $rand(0,1)$ is a random number generator that generates a random number in [0...1]. The transmission time

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

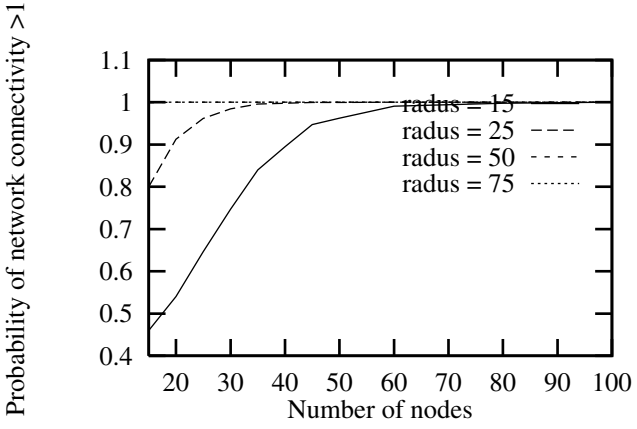

Figure 5. Probability of $K(G) > 1$.

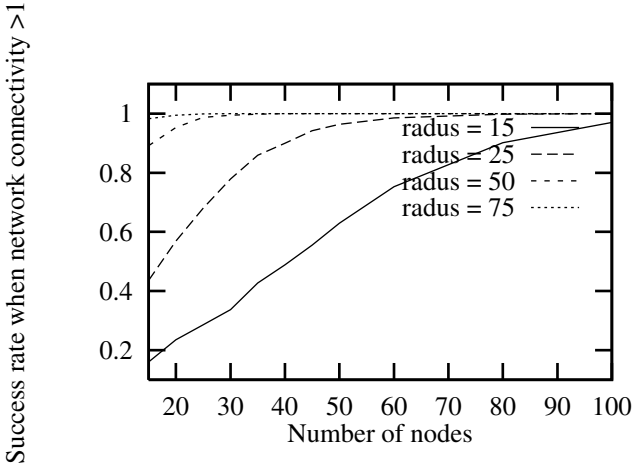

Figure 6. Success rate of finding two node disjoint paths when $K(G) > 1$.

(among neighbors) is $T_{min} + T_{range} \times rand(0,1)$, where $T_{min} = 0.2$ and $T_{range} = 0.3$. Delay time Δ is $D_{min} + D_{range} \times hop\_count \times rand(0,1)$, where $D_{min} - 0.3$ and $D_{range} = 0.5$. The hop_count is the number of hops from the source. Intuitively, the farther a node from the source, the longer the delay.

Based on the Menger's node-connectivity (or simply connectivity) theorem, a graph G is k-connected, represented as $K(G) = k$, if and only if any two distinct nodes are connected buy at least k-(internally) node disjoint paths. Therefore, if $K(G) > 1$ then two node disjoint paths exist. Two sets of simulation are conducted:

- The probability of $K(G) > 1$ is first calculated through simulation, given that G is connected. Then the success rate of finding two node disjoint paths when $K(G) > 1$ is determined.
- Given the source and the destination, the probability of the existence of node disjoint paths between them is determined. Note that such a node disjoint path may exist even though $K(G) \leq 1$. Then the success rate of finding two node disjoint paths when they exist is determined.

Figure 5 shows the probability of $K(G) > 1$ for unit graphs given the number of nodes and the radius of transmission range. It is clear that this probability is high for the radius of over 25 in a 100×100 2-D free-space. Converting to the diameter of the network, it is up to 7 hops. The probability increases (and is close to 1) when unit graphs become dense. This confirms the applicability of our approach in networks with relatively small diameters.

Figure 6 shows the success rate of finding two node disjoint paths when they exist. Again the results show the approach is effective for networks with small diameters. The success rate quickly reaches 1 as the unit graphs become dense.

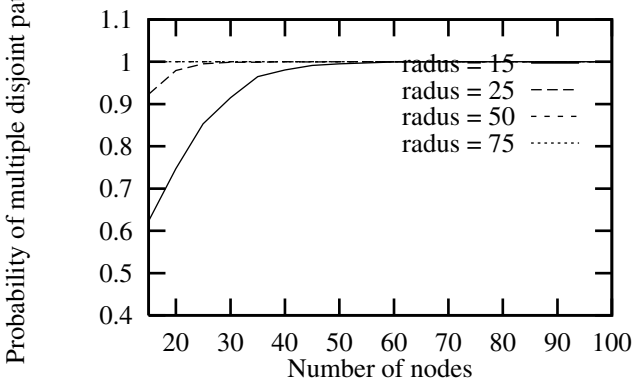

Figure 7. Probability of node disjoint paths between two nodes.

Figures 7 and 8 show similar results as Figures 5 and 6, respectively, but they are under a different condition. In this simulation, unit graphs under consideration are not necessarily 2-connected, but the source and the destination are connected by two node disjoint paths.

6 Conclusions

We have proposed an extension to the DSR by maintaining two node disjoint paths as a result of each route discovery process. The virtue of this approach is that it does not introduce additional overhead other than maintaining

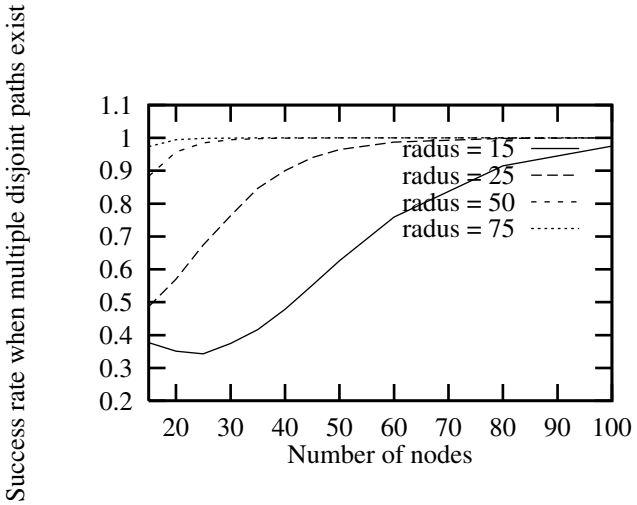

Figure 8. Success rate of finding two node disjoint paths when they exist.

the color of each node and a dirty bit associated with each route stored in the route cache. The simulation results have shown the effectiveness of the proposed multipath construction process. Our future work includes an in-depth simulation on the effectiveness of the proposed approach when route cache is used.

References

[1] D. J. Goodman. *Wireless Personal Communications Systems*. Addison Wesley, 1997.

[2] C. Hedrick. Routing information protocol. *Internet Request For Comments RFC 1058*, June 1988.

[3] A. Jean-Marie and L. Gun. Parallel queues with resequencing. *Journal of ACM*. Vol. 40, No. 5, Nov. 1993, 1188-1208.

[4] D. B. Johnson and D. A. Malts. Dynamic source routing in ad hoc wireless networks. In T. Imielinski, H. Korth, editor, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[5] J. M. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for ARPANET. *IEEE Transactions on Communications*. 28, 5, 1980, 711 - 719.

[6] J. Moy. OSPF version 2. *Internet Request For Comments RFC 1247*, July 1991.

[7] A. Nasipuri, R. Cartaneda, and S. R. Das. On-demand multipath routing for mobile ad hoc networks. *Proc. of IEEE INFOCOM 99*. 1999.

[8] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *Proc. of IEEE INFOCOM'97*. 1997, 1405-1413.

[9] C. E. Perkins. *Mobile IP: Design Principles and Practices*. Addison Wesley, 1997.

[10] W. T. Zaumen and J. H. Garcia-Luna-Aceves. Shortest multipath routing using generalized diffusing computations. *Proc. of IEEE INFOCOM 98*. 1998.