# An Adaptive Routing Protocol for Ad-hoc Networks using Multiple Disjoint Paths *

Young-ki Hwang, Hyungkeun Lee and Pramod K. Varshney, *Fellow*, IEEE
Department of Electrical Engineering and Computer Science
Syracuse University, 121 Link Hall, Syracuse, NY 13244, USA
youngki@ecs.syr.edu, hklee@cat.syr.edu, varshney@syr.edu

**Abstract**

*In this paper, we present the design and simulation of a source-initiated on-demand routing protocol for ad-hoc networks, which is known as the Signal-Power Adaptive Fast Rerouting (SPAFAR) protocol. SPAFAR employs efficient route discovery and maintenance mechanisms to increase bandwidth utilization and to decrease data flow disruption which occurs due to high node mobility. For the route discovery phase, we propose an algorithm to find multiple disjoint paths with longer-lived connections. This provides an efficient approach for data dispersion and for route maintenance. A longer-lived path generally involves less route maintenance. In the route maintenance phase, we propose a fast rerouting algorithm to significantly reduce data flow disruptions. Multiple paths provided by the route discovery phase significantly reduce the delay in finding an alternate path. Rerouting is proactivly carried out before path unavailability occurs. A simulation study is carried out where the performance of our SPAFAR algorithm is compared with several other routing algorithms. SPAFAR exhibits a significantly better performance, due to the coupled operation of route discovery and maintenance mechanisms.*

## 1. Introduction

An *ad-hoc* network is a collection of mobile nodes dynamically organizing themselves for communication without requiring existing infrastructure. Each node in such a network operates not only as an endpoint but also as a switch that has the functionality to forward data over the next hop while maintaining route information. Unlike the infrastructured network like a cellular system where routing is relatively straightforward, routing between two nodes is much more complex and challenging in an ad-hoc network. The routing protocol in an ad-hoc network needs to react promptly to frequent network topology changes that occur due to node mobility.

As interest in finding a solution framework for ad-hoc networks has grown, several routing solutions have been proposed for ad-hoc networks recently. They are mainly categorized into three broad classes depending on the method of maintaining route information; *table-driven* [10][2][8], *source-initiated on-demand* [11][6][9][13][3] and *hybrid* [5] routing protocols. The advantage of a table-driven protocol is that route information is always available whenever needed, resulting in a short delay before transmitting data. In contrast, an on-demand protocol eliminates the need for periodic broadcasts of route information, resulting in high bandwidth utilization compared to a table-driven protocol. Hybrid protocols reduce the control traffic significantly by limiting the broadcast only to the intrazone nodes, but they need efficient maintenance of the zones.

For successful communications in ad-hoc networks, a routing protocol should deal with the typical characteristics of the networks, such as limited bandwidth and high error rates, limited power supply and node mobility. Therefore, a routing algorithm designed for an ad-hoc network should have the following properties: low overhead for route discovery and maintenance mechanisms in terms of delay and bandwidth, multiple disjoint routes from a source to destination and fast rerouting mechanism to avoid data flow disruptions. In this paper, we propose an adaptive routing protocol to meet these goals for ad-hoc networks. We present an on-demand routing protocol to achieve high bandwidth utilization. This is desirable because the bandwidth is one of the most expensive resources in ad-hoc networks. Our protocol is known as the *Signal-Power Adaptive Fast Rerouting* (SPAFAR) protocol. SPAFAR consists of two phases namely the route discovery phase and the route maintenance phase. SPAFAR uses signal power strength and defines two signal power strength values for an efficient operation during both phases.

Multiple path routing has previously been proposed for the best-effort data traffic in ad-hoc networks [6][9][13]. Even though multiple paths were provided in the previous algorithms, they need not be disjoint, intermediate links are shared by multiple paths, and only one of the paths is used at one time. Multiple disjoint paths are important in that they can provide data dispersion. This results in an increase in the effective bandwidth, and reduction of congestion and the probability of a dropped packet in a network [12]. In this paper, we propose a route discovery algorithm to find multiple disjoint paths. These paths provide an efficient way for data dispersion and for route maintenance. When finding paths from a source to desti-

*This work is supported in part by the National Science Foundation under grant number ECS-9901361

nation, a link between physically closer nodes is selected for inclusion in a path over links that connect nodes that are far apart. This is because it is more likely to result in a longer-lived connection and path. A longer-lived path may not be shorter than some other paths in terms of number of hop counts, but it is likely to involve less route maintenance. Power strength of the received signal is used to determine the distance between two adjacent nodes.

Rerouting is needed whenever an established path is broken. When a path is broken, data transfers are disrupted until a new path is established. If multiple paths are provided, the delay in finding an alternate path is significantly reduced [1]. In previous algorithms, a rerouting mechanism is initiated when a broken path is detected, even though multiple paths have been found already. Therefore, these algorithms still suffer from some delays while rerouting. In this work, we propose a fast rerouting algorithm to significantly reduce data flow disruptions. Rerouting is done before path unavailability occurs. Received signal power strength is used to determine the time at which rerouting is initiated.

In Section 2, the new routing protocol is described including the route discovery and maintenance algorithms. In Section 3, we present some simulation results on the performance of our routing protocol. Section 4 summarizes the results of this paper.

## 2 The Routing Protocol

Each node of the ad-hoc network keeps a *Neighbors_Table* (NT) which has an updated list of its neighbors. The NT can be easily obtained by periodic broadcasts of the link layer BEACON. Each node also keeps a *Routing_Table* (RT) which has an updated list of all the possible routes to all the potential destinations. The RT is constructed by an on-demand routing algorithm. Route information is computed whenever a node has data to send to a destination that is not in the RT. Then, it is maintained by a separate maintenance mechanism.

SPAFAR uses the received signal power strength, which can be obtained by periodically broadcasting BEACONs and control packets, for both route discovery and maintenance. Using the minimum signal strength, denoted by $S_R$, that results in the satisfactory reception of a packet, SPAFAR determines the maximum transmission range R. SPAFAR defines two other threshold values in terms of signal power strengths, $Th_1$ and $Th_2$ as follows.

- The first threshold value, $Th_1 = S_{(R-\alpha)}$
- The second threshold value, $Th_2 = S_{(R-\beta)}$

where, $0 \leq \alpha, \beta \leq R$ , $\beta \leq \alpha$ *and* $S_R \leq Th_2 \leq Th_1$. These two thresholds are used to start preparations for potential rerouting and to actually reroute packets as described later.

In the route discovery phase, SPAFAR finds multiple disjoint paths from a source to destination. Neighboring nodes are divided into three groups. First group consists of nodes whose received signal power strength is above the first threshold $Th_1$. The second group consists of nodes whose received signal power strength is between the first threshold $Th_1$ and the second threshold $Th_2$. Finally the remaining nodes whose received signal power strength is between the second threshold $Th_2$ and $S_R$ belong to the third group. Neighbors in the first group are selected first as a part of the route to the destination followed by nodes of the second and third groups. Thus paths which are most likely longer-lived are found first.

Once the routes have been established for data communication, SPAFAR keeps monitoring the connection status in the route maintenance phase. The two threshold values are used to process the rerouting of a path. When the signal power strength between two neighbor nodes in an active path reaches a value between $Th_1$ and $Th_2$, preparation for potential rerouting begins by identifying the next best route between the source and destination that is not active. When the signal power strength between the two neighbor nodes reaches a value between $Th_2$ and $S_R$, traffic is rerouted on the new route. This process provides efficient and fast rerouting, and results in less data flow disruptions. When a connection between two nodes is broken due to reduced received signal power strength, SPAFAR deletes/updates the route information and searches for new paths if needed. These two protocols operate together continuously at each node.

More specifically, the second threshold $Th_2$ is the actual point at which rerouting is actually done. When a node is moving with velocity $X(meter)/S(sec)$, and the time for a packet to traverse from end to end in the network is $T$, $\beta$ should be greater than or equal to $(4 \times T \times X)/S$ in the worst case in order to reroute packets without data flow disruptions, i.e. $\beta \geq$ ( Time to notify the source node about the need for rerouting + Time for the source to complete the route discovery task + Time for the last packet to traverse from the source to destination, before rerouting occurs) $\times (X/S) = (T + 2T + T) \times (X/S)$ . $Th_1$ is the point at which rerouting preparation begins. The value of $\alpha$ needs to be found by experimentation.

### *2.1 Route Discovery Protocol*

A route discovery mechanism enables a node to find paths and to forward packets to a desired destination node. When a source wants to send data to a destination and it's RT does not have route information to the destination, the source initiates the route discovery mechanism to find all possible paths to the destination. The route discovery mechanism is based on request-reply operations. *R_Req* and *R_Rep* packets are used for the request and reply operations, respectively. They carry < *src, dest,* *req_id, int_nodes, hop_cnt* > information. Flooding is used for the request operation.

• Processing a R_Req Packet: when a source wants to find multiple disjoint paths to a destination, the source broadcasts a *R_Req* packet to its neighbors. When a node receives a *R_Req* packet, the following procedures are invoked.

```
/* If destination node of the path */
IF (dest of the R_Req packet = it's own address)
   IF (int_node of the R_Req is disjoint from the others found earlier)
      -Copy the reverse of the int_nodes field of the R_Req
      packet into the int_nodes field of a R_Rep packet.
      -Send the R_Rep packet to the source node along int_nodes
      in the R_Rep packet.
      -Insert Route info. from the destination to source into the RT.
   ELSE
      -Discard the R_Req packet.
/* if intermediate node of the path */
ELSE
   IF (the pair <src, req_id> in the R_Req packet)
   has been received already
      -Discard the R_Req packet.
   ELSE IF (it's own address is already listed in the int_nodes in the
   R_Req packet)
      -Discard the R_Req packet.
   ELSE
      -Append it's own address to the int_node in the R_Req packet.
      -Increase the hop_cnt by one in the R_Req Packet.
   IF (signal strength received ≥ Th1)
      -Broadcast the R_Req packet to its neighbor nodes without delay.
   ELSE IF (Th2 ≤ signal strength received < Th1)
      -Broadcast the R_Req packet to its neighbor nodes with delay D1.
   ELSE IF (SR ≤ signal strength received < Th2)
      -Broadcast the R_Req packet to its neighbor nodes with delay D2.
```

Here, $D_1 < D_2$, so that a link between closer neighbor nodes can be selected first as a part of the path. Hence, a path that is expected to be a longer-lived path is always found first.

• Processing a R_Rep packet: Upon receiving a *R_Req* packet, the destination sends a *R_Rep* packet which corresponds to the received *R_Req* packet along *int_nodes* in the *R_Rep* packet. When a node receives the *R_Rep* packet, the following procedure is invoked.

```
/* If source node of the path */
IF (dest of the R_Rep packet = it's own address)
   IF (This route is not in it's RT)
      -Insert the route info. from the source to destination into the RT.
/* if intermediate node of the path */
ELSE
   -Insert route info. from the source to destination into the RT.
   -Insert route info. from the destination to source into the RT.
   -Forward the R_Rep packet along int_nodes in the R_Rep packet.
```

Based on the procedures above, multiple *R_Req* packets which are not discarded at the intermediates nodes are guaranteed to reach the destination along a loop free sequence of nodes. Also multiple *R_Rep* packets are sent back to the source, if there are multiple disjoint paths between the source and destination nodes.

### *2.2 Route Maintenance Protocol*

Once the paths have been found, they need to be monitored and updated continuously, because frequent node movement results in path breakage. Three different packets are used for route maintenance; *R_Prepare*, *R_Reroute* and *R_Error* packets. Each node classifies the paths into active or non-active in its RT. The active paths are the ones that are currently being used for data transfer. *R_Prepare* and *R_Reroute* packets are used only for active paths, while *R_Error* packet is for all non-active or active paths. The threshold $Th_1$ is used in conjunction with *R_Prepare* packets, while the threshold $Th_2$ is used in conjunction with *R_Reroute* packets.

• Preparing Alternate Paths

When the received signal power strength from a neighbor falls below $Th_1$, but is larger than $Th_2$, and the link to the neighbor is on the active path list in RT, the node sends a *R_Prepare* packet to the source node of the path. The purpose of the *R_Prepare* packet is to have the source node find alternate paths in advance before the current path becomes unavailable. When the source receives the *R_Prepare* packet, it scans the RT to find out whether or not there are other non-active paths to the destination. If there are, the source node does not do anything further. If not, the source node initiates a route discovery mechanism to find alternate paths to the destination. The source continues to transfer data over the current path, because the path is not broken yet. When an intermediate node receives the *R_Prepare* packet, it forwards the packet to the source node. A timer for the *R_Prepare* packet needs to be used at the intermediate node, so that it retransmits the packet to the source node after timeout value expires. It prevents the source node from failing to find alternate paths when the source node does not receive the *R_Prepare* packet successfully. The intermediate node continuously retransmits the *R_Prepare* packet to the source node every time the timeout value expires, until the signal strength from a neighbor falls below $Th_2$. When the source node receives the redundant packets, it just ignores them if it already has alternate non-active paths to the destination.

• Rerouting

When the received signal power strength from a neighbor falls below $Th_2$ but is still larger than $S_R$, and the link to the neighbor is part of the active path in RT, the node sends a *R_Reroute* packet to the source node of this path. The purpose of the *R_Reoute* packet is to have the source node change the path to another one in advance of the path becoming unavailable. When the source node receives the *R_Reroute* packet, it checks whether or not there are other inactive paths to the destination for which no *R_Reroute* packets have been received. If there are, it changes the path to the path that is likely to be most longer lived from the set of available paths, and makes the old path inactive. If not, it continues to transfer data on the old path while it initiates a route discovery mechanism to find alternate paths. Traffic is not diverted on those paths which have received a *R_Reroute* packet already because they are likely to become unavailable soon. When an intermediate node receives the *R_Reroute* packet, it forwards the packet to the source node. Once again, a timer for the *R_Reroute* packet needs to be applied, so that the intermediate node retransmits the packet to the source node after the timeout value expires. It prevents the source node from failing to reroute traffic when the source node does not receive the *R_Reroute* packet successfully. The intermediate node continuously retransmits the *R_Reroute* packet to the source node every time the timeout value expires, until the neighbor is not reachable any more. When the source node receives the redundant packets, it just ignores them if it has already rerouted traffic to the destination.

- Reporting Broken Paths

When a neighbor is not reachable any more, and the link to the neighbor is part of some paths in RT, the node sends a *R_Error* packet to the source node of these paths. Each intermediate node including the source node which is part of the paths deletes the route information from their RT.

## 3 Simulation Results

The SPAFAR protocol was simulated using GLOMOSIM, which is a mobile network simulator developed using event-driven simulator PARSEC [7]. A network consisting of 50 mobil nodes over a fixed size $100m \times 100m$ terrain was considered. The maximum transmission power range was assumed to be $35m$ between nodes. Nodes were initially placed randomly within the fixed-size physical terrain. They move to a randomly selected destination within the terrain with a speed of MOVING-SPEED(meter/sec). After reaching the destination, they stay there for 3sec time period before they move again. The source and destination are selected randomly, and they also move and stop continuously in random directions during the whole simulation.

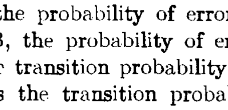

Figure 1. Two-state Markov model for fading channel

A two-state Markov model [4] is used to represent the error behavior of slowly fading channels in wireless networks. The Markov chain model consists of two states, Good(G) and Bad(B) as shown in Figure 1. In the good state G, the probability of error ($P_{eg}$) is 0. In the bad state B, the probability of error ($P_{eb}$) is 0.5. $P$ represents the transition probability from state G to B, whereas $p$ is the transition probability from state B to G. The probabilities to remain in each state are $Q = 1 - P$, $q = 1 - p$ respectively. It can be shown that $P = (p * (ber - P_{eg})) / (P_{eb} - ber)$, where *ber* is the channel bit error rate. In our simulation, $p$ is assumed to be 0.25.

The purpose of the simulation is to compare the performance of the SPAFAR protocol with other routing protocols that include two other on-demand routing protocols (DSR, AODV), one table-driven protocol (WRP) and the distributed Bellman-Ford (DBF) protocol in ad-hoc networks. The simulation was done for a Constant Bit Rate (CBR) application, which generates data traffic from source to destination at a constant rate, i.e. 400Kbps with 512 bytes packet size in this simulation. CBR runs over UDP transport layer protocol which does not employ retransmission schemes for lost data packets. Packets are dropped at the intermediate nodes, whenever they are corrupted due to channel fading.

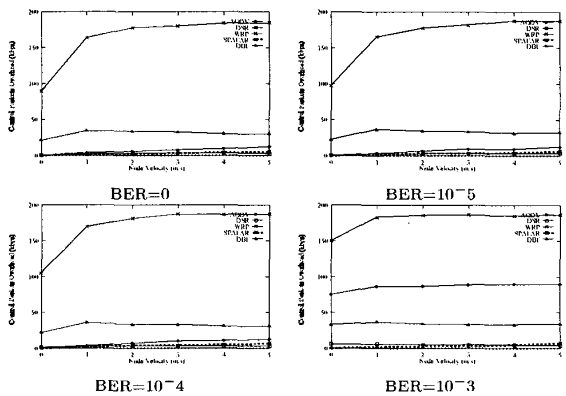


Figure 2. Control packets overhead for different bit error rates

Figure 2 shows the overhead incurred by control packets sent for route discovery and maintenance during the whole simulation. They are measured in terms of the bandwidth(kbps) consumed by the control traffic in the network. WRP sends route updates periodically and whenever the topology changes, which results in excessive control packets and, therefore, is not acceptable in a limited bandwidth wireless environment. AODV, DSR and SPAFAR generally generate significantly fewer control packets than WRP and DBF. AODV, however, generates many more control packets when bit error rate becomes severe and node moving speed increases, therefore consumes more network bandwidth in that environment. DSR generates the least number of control packets always, because it does not seem to react promptly to the changed network topologies. Even though DSR generates much fewer control packets, it achieves lower data goodput and throughput as shown in Figures 3 and 4. SPAFAR generally involves less route discovery operations than other on-demand routing protocols, because it finds multilple paths. Less route discovery operations significantly reduce the delay for acquiring new paths and, therefore result in performance improvement.

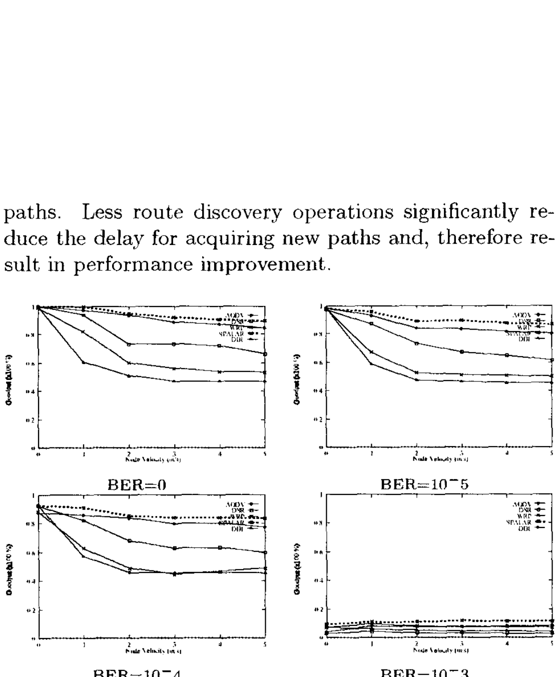

Figure 3. Data Goodput for different bit error rates

Figure 3 compares goodput of five routing protocols. Here, goodput is defined as the ratio of the number of correctly transferred packets to the total number of transferred packets during the whole simulation. Goodput represents how quickly a routing protocol reacts to the changed network topologies. The result shows that DSR, WRP and DBF fail to react promptly to the changed network topologies and lose lots of data packets when bit error rate becomes severe. SPAFAR shows higher goodput than any other protocols, because it performs rerouting before the path for data transfer is broken, which results in less data flow disruption and performance improvement.

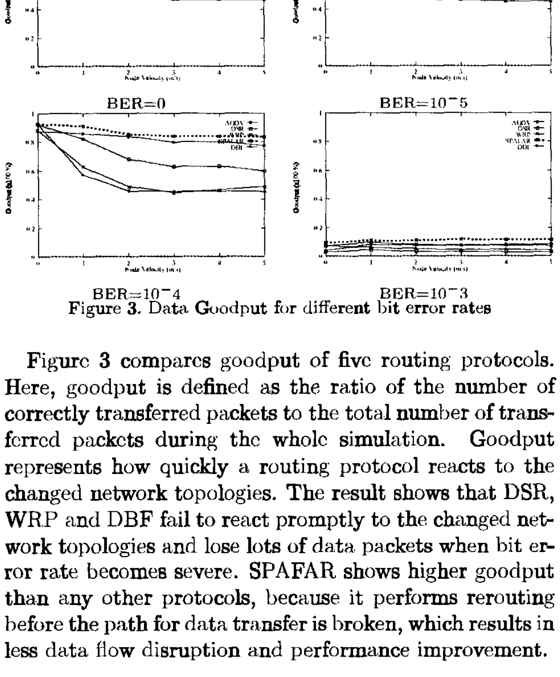

Figure 4. Data Throughput for different bit error rates

Figure 4 compares data throughput of five routing protocols. Unlike the data goodput, data throughput here is defined as the actual data rate (number of bits) transfered to the destination per second during the whole simulation. Based on combined route discovery and maintenance operations using two threshold values, SPAFAR shows the higher data throughput than other protocols.

## 4 Conclusion

In this paper, a new source-initiated on-demand routing algorithm for ad-hoc networks has been proposed. The SPAFAR algorithm consists of two phases, the route discovery phase and the route maintenance phase. SPAFAR uses received signal power strength and two threshold values to implement both route discovery and maintenance phases. In the route discovery phase, SPAFAR finds multiple disjoint paths. A path that is more likely longer-lived is found earlier. It results in less route maintenance and provides a way to fast rerouting. In the route maintenance phase, SPAFAR performs rerouting before the paths become unavailable. It results in less data flow disruptions. By employing dispersity over multiple disjoint paths and initiating rerouting proactively with these route discovery and maintenance mechanisms, we are able to attain significantly improved performance.

## References


[1] S. Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE JSAC*, 17(8):1488–1505, August 1999.

[2] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. *Proc. IEEE Singapore International Conference on Networks (SICON)*, pages 197–211, April 1997.

[3] R. Dube, C. Rais, K. Wang, and S. Tripathi. Singal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks. *IEEE Personal Communications*, pages 36–45, February 1997.

[4] E. Gilbert. The Capacity of a Burst-Noise Channel. *The Bell System Technical Journal*, pages 1253–1265, September 1960.

[5] Z. Hass and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *SIGCOMM*, pages 167–177, October 1998.

[6] D. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–168, December 1994.

[7] U. P. C. Laboratory. *http://pcl.cs.ucla.edu.*

[8] S. Murthy and J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications*, pages 183–197, October 1996.

[9] V. Park and M. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. *IEEE INFOCOM*, pages 158–168, April 1997.

[10] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM*, pages 234–244, August 1994.

[11] C. Perkins and E. Royer. Ad-hoc On Demand Distance Vector (AODV) Routing. *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[12] D. Sidhu, R. Nair, and S. Abdallah. Finding Disjoint Paths in Networks. *SIGCOMM*, pages 43–51, Septermber 1991.

[13] C. Toh. Associativity-Based Routing For Ad-Hoc Mobile Networks. *Wireless Personal Communications*, pages 1–36, March 1997.