Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks*

Rohit Dube Cynthia D. Rais Kuang-Yeh Wang Satish K. Tripathi
Institute for Advanced Computer Studies
Mobile Computing and Multimedia Laboratory
Department of Computer Science
University of Maryland
College Park, MD 20742
{*rohit,cldavis,kwang,tripathi*}*@cs.umd.edu*

CS-TR-3646, UMIACS-TR-96-34

August 28, 1996

Abstract

Unlike static networks, ad-hoc networks have no spatial hierarchy and suffer from frequent link failures which prevent mobile hosts from using traditional routing schemes. Under these conditions, mobile hosts must find routes to destinations without the use of designated routers and also must dynamically adapt the routes to the current link conditions. This paper proposes a distributed adaptive routing protocol for finding and maintaining stable routes based on signal strength and location stability in an ad-hoc network and presents an architecture for its implementation.

1 Introduction

Ad-hoc networks consist of mobile hosts in a dynamic network bereft of base-stations and pre-designated routers. These mobile hosts can communicate directly with neighboring hosts through the shared wireless media, but communication with non-neighbor hosts requires a distributed routing algorithm. Traditional routing algorithms which depend on hierarchical addressing schemes and pass detailed routing tables between designated routers, are not effective in ad-hoc networks due to the high rate of topology change [LNT87], [RT87]. In a dynamic network, the routing tables are soon out-of-date and the propagation of the route information is too slow to be accurate. In addition, the changing topology makes looping routes a problem. Hence, new routing algorithms need be developed to support ad-hoc networks.

Ad-hoc networks require a highly adaptive routing scheme to deal with the frequent host migrations. In the worst case of a highly dynamic topology, the rate of change is too fast to enable effective routing and data packets must be flooded through the network. In less dynamic cases, it is possible to take advantage of the stable parts of the network to establish routes and communicate using unicast data.

*This work is supported in part by NSF grant CCR 9318933 and IBM equipment grants.

Not all hosts in an ad-hoc network are equally mobile at a given time. Hosts that have been stationary for a period of time are more likely to remain stationary than those currently in motion. In addition, some pairs of hosts receive stronger signals due to their physical proximity and the propagation environment. It is better to choose routes that pass through hosts which receive a stronger signal and which have been stationary for a longer period of time.

We propose a routing protocol that utilizes the above observations to choose stable routes within a dynamic ad-hoc network. In this protocol, a host initiates route discovery on demand—when a route is needed to send data. The source broadcasts route-search packets which will propagate to the destination, allowing the destination to choose a route and return a route-reply. Signal strength and location stability are used as the criteria for choosing between various next-hop candidates for a route. Selecting the most stable links leads to longer-lived routes and requires less route maintenance.

The simulations we performed demonstrate the benefits and costs of using signal strength and location stability as the route selection criteria. The results show that the use of signal strength consistently decreases the route maintenance required by providing longer-living routes. Our results also show that location stability should be used only in certain scenarios since misinformation about stability patterns is very costly and has a negative impact on the routing performance.

The next section of this paper discusses the basic signal stability based adaptive routing protocol and gives an example of the use of this protocol. Section 3 presents details of the protocols and the architecture, and Section 4 discusses extensions to the protocol and the inter-operability with Mobile-IP. Section 5 presents our simulation environment and results. Section 6 discusses other ad-hoc routing proposals, and the final section presents conclusions and some suggestions for future work.

2 Signal Stability based Adaptive Routing (SSA) Protocol

The Signal Stability based Adaptive Routing protocol (SSA) performs on demand route discovery by selecting longer lived routes based on signal strength and location stability. The signal strength criteria allows the protocol to differentiate between *strong* and *weak* channels, each channel being so characterized by the average signal strength at which packets are exchanged between the hosts at either end of the channel. The location stability criteria biases the protocol toward choosing a channel which has existed for a longer period of time. Together, these two concepts form the signal stability criteria which chooses strong channels which have been in existence for a time greater than some threshold.

2.1 Protocol Overview

A source initiates a route discovery request when it has data to send to a destination which is not in the routing table. The route search is broadcast to all neighboring hosts. These hosts propagate the broadcast if (1) it is received over a strong channel and (2) the request has not been propagated previously (to avoid looping). The route search stores the address of each intermediate host in the route taken. The destination chooses the route recorded in the first arriving request, since this route is probably shorter and/or less congested than routes for slower arriving requests. The destination returns the route reply along the selected route, and each intermediate node includes the new next hop, destination pairs in its routing table.

Host	Signal Strength	Last	Clicks	Set
Y				
Z				

Table 1: The Signal Stability Table (SST)

Destination	Next Hop
Y	
Z	

Table 2: The Routing Table (RT)

Functionally, the Signal Stability based Adaptive Routing (SSA) protocol consists of two protocols, the Static Routing Protocol (*SRP*) and the Dynamic Routing Protocol (*DRP*), which utilize the *extended device driver interface*. This interface is responsible for making available to the routing protocols the signal strength information from the device. DRP maintains the routing table by interacting with the DRP on other hosts. SRP performs the actual routing table lookup to forward a packet onto the next hop.

2.2 Protocol Modules

The Static Routing Protocol (SRP) and the Dynamic Routing Protocol (DRP) work together to route packets in the ad-hoc network. The extended device driver interface enables communication between these routing protocols and the link layer for the sending and receiving of packets and receiving wireless link quality information. Two tables are maintained to enable SSA routing: the Signal Stability Table (table 1) and the Routing Table (table 2).

Each host sends out a link layer beacon[1] to its neighbors once every time quanta, denoted by a *click*. Every host receiving this beacon records the *signal strength* at which the beacon was received in the Signal Stability Table (**SST**). Each host also classifies its neighbors as strongly connected (*SC* and hence belonging to the *SC-set*) if the host has been receiving strong beacons from the neighbor for the past few clicks. The neighbor is otherwise classified as weakly connected (*WC* and hence belonging to the *WC-set*). A host marked as SC in the SST, also has an entry in the Routing Table (**RT**) which stores destination and next hop pairs for each known route. The SST also has a column to indicate that a beacon was received from a host within the last click, *Last*, and a column to record how long beacons have been continuously received from each neighboring host.

The availability and processing of signal strength information is made possible by the **extended device driver interface** which provides the **DRP** with the average signal strength at which a packet was received and the immediate sender. The DRP uses the extended interface to maintain the statistics in the SST. It then uses the SST to maintain routes to neighboring hosts in the RT and the non-neighbor routes via information provided by route-

[1] Beacons are 'I am alive messages' which are exchanged between wireless devices at regular intervals to maintain connectivity. SSA does not add overhead by defining any new beacons.

search, route-reply, error, and erase messages.

The **SRP** functions by looking up the destination in the RT and forwarding the packet on the next-hop for the destination. When there is no entry for the destination in the RT, the SRP initiates a *route-search* to find a route to this destination. The route-search message has a hop-list which records the path taken by the message. Each intermediate DRP uses this list to prevent loops and adds its own address to the hop-list. Although the destination DRP may receive multiple copies of a route-search messages, it simply selects the route contained in the first arriving route-search message and tunnels a *route-reply* message on the reverse path to the source. The DRP at each intermediate host installs the appropriate next-hop entry for the destination in its RT. When a route-reply message is received, the DRP at the source updates the RT and the SRP routes the data via the next hop found in the RT.

A route may become unavailable due to migration of the hosts along the route's path. When a host moves out of range of its neighbors or shuts down, the neighbors will recognize that the host is unreachable since they no longer receive beacons from that host. The DRP will modify the SST and RT to reflect the changes. Any routes that have this unreachable host as the next hop will become invalid. When the host receives a packet to forward along an invalid route, SRP will determine the lack of a route and will notify the source via an error message. The source SRP will initiate a new route discovery to find an available route, and it will send a message to erase the invalid route.

2.3 An Example

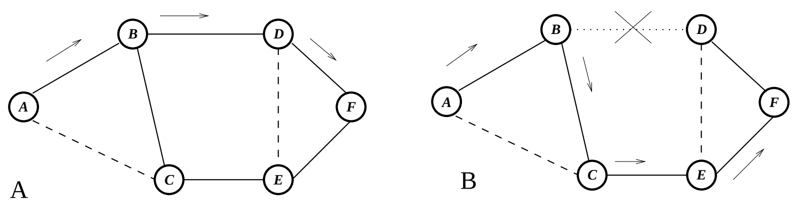

Figure 1: An Example Network

Consider the example ad-hoc network shown in Figure 1. A solid edge indicates that the vertices that make up the edge are in each others' SC-sets. The dashed edges indicate that the corresponding vertices are in the WC-set.

In part A of Figure 1, host A has data to send to host F and therefore wants to find a route to destination F. The protocol starts by sending out a broadcast route-search message seeking destination F. Route-search packets that arrive at any host over WC links are dropped, such as the packet sent from A to C. The route-search packet sent to B over the SC link will be broadcast by B to its neighbors. A will drop the packet, having already forwarded it, D and C will forward the packet and mark that they have seen it. Note that C has not already marked the packet as seen since it dropped the packet over the WC from A link without processing it. Packets will arrive at the destination F via two paths: A, B, D, F and A, B, C, E, F. Assuming that the links have similar latencies and traffic levels, the route-search will arrive over the shorter path first : A, B, D, F. The destination, F, will select this route and return the route-reply along the reverse path, F, D, B, A, installing entries in the RT at each intermediate host along the path and at the source, D, B, A. Once the route is installed in the RT, A forwards the data packets via the next-hop to destination F, which is host B.

Assume now that the link B - D disappears, as shown in part B of Figure 1. When B realizes that host D is unreachable, it sends an error message to A. A then initiates another route-search and sends an erase message to erase the invalid route. F again selects the first arriving route-search, which will be the route A, B, C, E, F.

If, in the future, the link between B and C disappears, then a strong route no longer exists to the destination. In this case, no route-search packets will be propagated to the destination, since packets are not forwarded which arrive over the WC links. When A does not receive a reply after some timeout period, it must decide whether it wants to find any route or wait and try to find a strong route at a later time. If it wants any route, it will send a route search message specifying *any* route, and will find the route A, C, E, F, which has one WC link.

3 SSA Protocol Details

We present the architecture, the packet format, and the protocol details in this section. This provides a clearer understanding of the protocol and also illustrates the architecture that could be used to implement such a protocol. For clarity, the algorithm discussed below presents a synchronous processing scenario, but an implementation of the algorithm could also be asynchronous.

Since ad-hoc networks exhibit no spatial hierarchy, we assume a flat addressing scheme. For simplicity of exposition we use the MAC addresses of the wireless device as the address of a node.

3.1 Protocol Stack

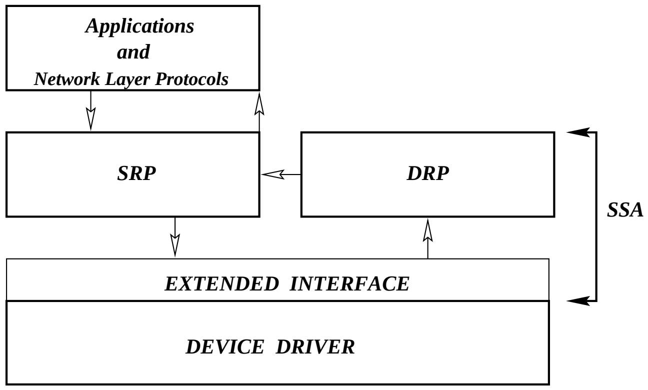

Figure 2: The Protocol Stack at each Mobile Host

The SRP and DRP are located between the network layer and the link layer, as shown in Figure 2. This makes the SSA routing protocol inter-operable with Mobile-IP, as we discuss in section 4. All incoming packets pass from the device driver extended interface to DRP. The DRP updates the SST and the RT and relays the appropriate types of packets to the SRP. The SRP then either passes a packet up the stack or forwards the packet though the wireless device driver on to the next hop. All transmissions go out via SRP and all receives in through DRP. This division simplifies the protocol by exporting a single interface for outgoing packets and also separating out the filter for incoming packets.

3.2 Packet Format

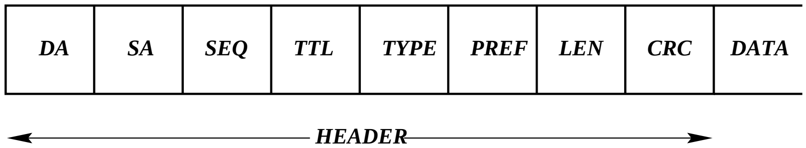

Figure 3: SSA Packet Format

Figure 3 shows the packet format expected by the protocols. *SA* and *DA* refer to the source and destination addresses respectively. *SEQ* is a sequence number assigned by the source, which is useful for route-searches. *TTL* is a time-to-live field used to eliminate erroneous packets looping in the network. *TYPE* distinguishes between messages and is one of the following: UNICASTDATA, FLOODDATA, ROUTESEARCH, ROUTEREPLY, ERROR or ERASE. *PREF* allows a host initiating a route-search to specify the quality of the route desired. This field can be STRONGLINKONLY or NOPREFERENCE. *LEN* is the length of the entire packet and *CRC* is the checksum.

The rest of the packet contains either data (for a packet of type UNICASTDATA or FLOODDATA); the recorded route hop list (for a packet of type ROUTESEARCH or ROUTEREPLY); or the destination address of a stale route (for a packet of type ERROR or ERASE).

3.3 SSA Broadcast and Flooding

It is important to distinguish between MAC level broadcasting and SSA broadcasting. Every SSA packet is encapsulated in a MAC frame before transmission, and every MAC frame is de-capsulated to an SSA packet on reception. If the MAC address of the encapsulated frame is the broadcast address, then all hosts on that shared wireless media which receive the frame (immediate neighbors) will pass the packet up to the DRP for processing.

On the other hand, an SSA broadcast has its *DA* equal to the broadcast address. This type of broadcast packet will be delivered to all hosts within this ad-hoc network and passed up to the network layer on each receiving host as well as being forwarded. To achieve this, SSA has to tell its lower layer to use MAC level broadcast, i.e. the MAC address of this packet is also the broadcast address.

A flooded packet, of *TYPE* FLOODDATA, must be forward by any host that receives it and is not the SSA destination. Such a packet is forwarded through a MAC level broadcast even if a route to the destination is unknown. The packet is not passed up to the network layer until it reaches the SSA destination.

In summary, SSA broadcast reaches the entire ad-hoc network; flooding tries to reach a specific destination by propagating through the entire network; and MAC level broadcast only ensures that the packet reaches the SSA at all the immediate neighbors.

3.4 SRP

SRP accepts packets from the DRP and from higher layer protocols. If the destination address of the packet matches that of the host, the packet is pushed up the protocol stack. SRP forwards all other packets through the device driver. Broadcasts are sent out without checking the RT, and the SRP performs a Routing Table lookup for unicast packets. If no entry is found for the unicast destination, then a route needs to be found. If this host is the source of the packet, then a route-search is initiated. When a route is found, the DRP will receive the route-reply and install the route in the RT. The SRP then forwards the original packet to the next hop. Alternatively, if this host is not the source of the packet, then a packet of type ERROR is sent back to the source, which will send a message of type ERASE to tear down the old route and will initiated a new route-search. In this case, the original packet will be dropped by the intermediate host.

Table 3 shows the details of the SRP packet processing. `initiateRouteSearch()` produces and broadcasts a route-search packet with an empty data portion, a unique sequence number in *SEQ*, and a *TYPE* field of ROUTESEARCH. The source can choose an appropriate *PREF* value depending on the needs of the upper layer application or protocol. We choose STRONGLINKONLY for the first try and NOPREFERENCE for any ensuing request retry after a time-out. If after several attempts no satisfactory route is found, the data packet is broadcast and reaches the destination via flooding. An alternate approach to flooding would be be to report an exception to the application which generated the packet.

The routine `sendRouteError()` initiates a route-error packet, with *DA* equal to `srcAddr` and *TYPE* ERROR. It sends the error packet to the original source for which forwarding was being attempted. The route maintenance packets (of *TYPE* ERROR or ERASE) are unicast with best-effort delivery. This means that any hosts which don't have an entry for the forwarding destination will drop the packet without further action. Section 3.7 explains this further.

All outgoing SSA packets are encapsulated with the current MAC address and the next-hop MAC address, or the MAC broadcast address if every neighboring host should receive the packet. In the pseudo-code, `forward(pkt)` is defined to be a unicast to a particular host's MAC address. `broadcast(pkt)` sends the packet to all reachable hosts by using the MAC broadcast address at the destination. Unicast data is processed only by the host with the matching address while broadcasts are processed by every host which receives the packet. Packets with MAC broadcast addresses may have unicast higher layer addresses, and the SRP forwards the packet and/or passes it up to the higher layers depending on the packet type and destination address.

3.5 DRP

The DRP is more complex than the SRP since it processes incoming packets and maintains the RT and the SST. On receiving a packet from the device driver, the DRP deciphers the packet type, updates the tables, modifies some of the header fields, and then passes it to the SRP. The DRP pseudo-code is given in Table 4.

In addition to the packet, the device driver passes the signal strength (`sig`) at which the packet is received and the address of the host (`sender`) that transmitted this packet (not the original source). The `updateTables()` function then updates the signal strength field in the SST according to the following formula:

$$SS_{cumulative} = \alpha \times SS_{cumulative} + (1 - \alpha) \times SS$$

$SS_{cumulative}$ here is the quantity recorded in the SST and SS is the value of average signal strength for the the packet supplied by the device driver. α is an experimentally determined constant. `updateTables()` also marks the *Last* field in the SST to indicate that a beacon was received from the sending host during the current time click.

```
/* input : packet pkt */
destAddr = pkt->header.DA;
if (destAddr == myAddr || isBroadcast(destAddr) == YES)
    /* pkt for this host */
    /* pass packet to application */
    enQ (pkt);
if (destAddr != myAddr) { /* pkt not for this host */
    if (isBroadcast(destAddr) == YES || pkt->header.type == ROUTESEARCH
        || pkt->header.type == FLOODDATA)
        /* pkt forwarded by DRP for broadcast */
        broadcast(pkt);
    else {
        if (isInRt (destAddr) == YES) {
            /* if the destAddr is in the RT, forward the packet */
            next = nextHop (destAddr); /* get the next hop */
            forward(next, pkt);
        }
        else {
            if (pkt->header.type != ERASE && pkt->header.type != ERROR) {
                if (srcAddr == myAddr) {
                    /* initiate route request. At the completion of this call
                       there will be an entry for the destAddr in the RT */
                    initiateRouteSearch(destAddr);
                    next = nextHop (destAddr) ; /* get the next hop */
                    if (next != NULL) forward(next, pkt);
                    else {
                        pkt->header.type = FLOODDATA;
                        broadcast(pkt);
                    }
                }
                else {
                    /* route fails; inform the source and drop packet */
                    sendRouteError(srcAddr);
                    discardPkt(pkt);
                    return;
                }
            else {
                discardPkt(pkt);
                return;
            }
        }
    }
}
```

Table 3: Pseudo-code for SRP

```
/* input : signal_info sig, packet pkt */
updateTables(sender, sig);
/* link layer has removed the lower level header from pkt */
if (pkt != NULL) {
    if (isStale(pkt) == YES) {
        discardPkt(pkt);
        return;
    }
    switch (pkt->header.type) {
    case (ROUTESEARCH):
        if (isFromWC(pkt) == YES &&
            pkt->header.pref == STRONGLINKONLY) {
            discardPkt(pkt);
            return;
        }
        if (isSearchForMe(pkt->header) == YES) {
            outPkt = constructRouteReply(pkt);
            if (outPkt == NULL) break ; /* reply not yet constructed */
            else installHop(outPkt);  /* update RT */
        }
        else {
            outPkt = constructRouteSearchForward (pkt) ;
        }
        seenRequest(pkt->header.SA, pkt->header.seq);
        SRP(outPkt);
        break;
    case (ROUTEREPLY):
        installHop(pkt);  /* update RT */
        outPkt = constructRouteReplyForward(pkt);
        SRP(outPkt);
        break;
    case (UNICASTDATA):
    case (FLOODDATA):
        SRP(pkt);
        break;
    case (ERROR):
        if (pkt->header.DA == myAddr) {
            outPkt = constructRouteErase(pkt);
            SRP(outPkt);
            deleteHop(pkt);
        }
        else {
            SRP(pkt);
        }
        break;
    case (ERASE):
        SRP(pkt);
        deleteHop(pkt);  /* update RT */
        break;
    }
}
```

Table 4: Pseudo-code for DRP

```
/* clicks assumed to be initialized to 0 */
if (last not marked) {
    delete entry from SST and RT;
    return;
}
unmark last;
if (SS_cumulative >= SS_threshold) clicks++;
else clicks = 0;
if (clicks == clicks_threshold) {
    set = SC;
    add host to RT;
 /* if clicks > clicks_threshold, then host is already in the RT */
}
else set = WC;
```

Table 5: Pseudo-code updating SST

When no packet is received during a certain period of time, the device driver may still pass signal information obtained through its beacon. In this case the DRP processes only the signal information.

Periodically, an asynchronous process runs through the SST comparing $SS_{cumulative}$ to an experimentally determined quantity $SS_{threshold}$. The calculation shown in Table 5 is then carried out.

$clicks_{threshold}$ is another experimentally determined quantity. The above piece of code ensures that a mobile host which exhibits strong signal strength for $clicks_{threshold}$ consecutive clicks is included in the SC-set and is added to the RT (with itself as the next hop).

The DRP processes each packet depending on its packet type. For route-search packets, `isStale()` determines whether the packet should be dropped. It is dropped if the packet has been previously seen by this host or if the *TTL* has expired. The host records route-search packets that it has seen in a table of source (*SA*), sequence number (*SEQ*) pairs. The routine `seenRequest()` records the pair when the host processes the route-search packet for the first time.

`constructRouteSearchForward()` modifies the route-search packet by adding the address of the resident host to the hop list to construct a new route-search packet for forwarding.

When the intended destination receives a route-search packet, a route-reply is produced by `constructRouteReply()` and sent back to the source along the reverse path of the route contained in the route-search packet. The next hop for route-reply is then installed in the RT and the reply is forwarded. Every intermediate node echoes this procedure (`installHop()` and `constructRouteReplyForward()`). `installHop()` installs *all* possible routes that are implied by the route-reply. For example, Figure 4 shows a route found from A to G. When host D receives a route-reply from G, it installs routes to A and B (with a next hop of C) and installs routes to F and G (with a next hop of E). D already has routes for its neighboring hosts, C and sc e, from beaconing information.

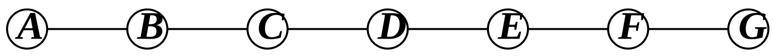

Figure 4: Establishment of a Route

Once the route installation is completed, data packets starting from the source are forwarded along the next hops installed in the RTs of the intermediate nodes. In the cases where all hops are along an SC-set path, the packets would be routed quickly and efficiently. If a host cannot forward a packet due to link failure, SRP sends a route-error packet back to the source. When the source receives such a route-error packet, it sends a route-erasure packet as constructed by the function `constructRouteErase()` with the DA equal to the destination of the stale route. The stale route is deleted from its RT (`deleteHop()`). Any intermediate host receiving this route-erasure packet forwards it to the next hop, and deletes the stale route. If the host is unable to forward the packet, the packet is simply dropped.

3.6 Extended Interface

The extended device driver interface provides the `updateTable()` call by which the higher layer protocols update the SST. The interface allows changes to the the time period between beacons. It also allows $SS_{threshold}$ and $click_{threshold}$ to be controlled. The $SS_{threshold}$ determines the extent of the host's coverage area within which the neighbor nodes have strong signals. The $click_{threshold}$ regulates the impact of location stability considerations on the protocol. It should be determined based on known mobility patterns of hosts in the ad-hoc network. In the case where it is set to one, the protocol's routing decisions are based solely on signal strength.

3.7 Route Maintenance

Route maintenance is triggered when a host has data to send over a failed link. Intermediate nodes send an error message to the source when such a failure occurs. The source host sends a route search packet to find a new route and sends an erase message to remove the old route. The erase message should reach the intermediate host which discovered the failed next hop. Error and erase messages are unicast with best-effort delivery and are dropped if a host is unable to forward the packet. This prevents a cycle of error and erase messages from wasting network resources.

In cases of multiple failures, some routing messages may not reach their destinations. This may result in the existence of stale routes, but it will not cause any routing errors or loops. The stale routes will be discovered and erased by the next data packet that tries to use the invalid route.

If a link failure prevents the route reply from reaching the source, the source will time out and retry the route request. The intermediate hosts between the failed link and the destination will have incorrect routes to the source. If any of these hosts use these routes, error and erase messages will be generated to correct the routes.

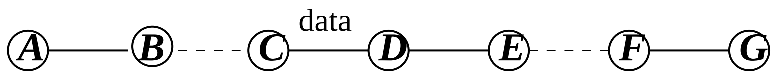

Figure 5: Failure of route erase packet

In cases where multiple link failures occur nearly simultaneously, erase and error messages may not reach their destinations. If an ERROR packet cannot be delivered to the source, it must be due to another link failure which will also trigger an *error* packet closer to the source. This second *error* packet will inform the source of the second failure so that the source can take appropriate action. If a data packet is between the links when they both fail, as shown in Figure 5, the resulting error message from E will not reach the source A, since it will be dropped due to the failed link at C. When A sends a data packet again, B will send the error message to inform A of the route failure. The erase message from A will erase the route at B, but the stales routes to G at C and D will only be erased if either C or D tries to send data to G.

If the second link failure occurs after the error message arrives at A, then the erase message from A will be dropped at B. This creates the same situation where C and D have an invalid route to G which will be erased when either C or D sends a packet toward G.

Since hosts relay route-search packets even if they already know a route to the desired destination, the algorithm works correctly in the presence of stale routes. The source is informed of the error and initiates a new route-search. SSA uses route erasure only to avoid wasting resources by forwarding data packets over routes known to be stale. The cost of this simple error and erase method is a few stale routes. Since the multiple failure cases are rare, the best-effort unicast of error and erase packets is an effective method of reducing excessive packet transmissions.

4 Other Issues

4.1 Enhanced SSA

The protocol presented in the previous sections provides a basic routing function. However, there may be several enhancements to give the hosts more options to deal with varying needs. One is the route quality option (the *PREF* field in the header) which the source sets when searching for a route. The host may prefer routes with more strong links but not want to exclude routes with any weak links. To accommodate this need, the hosts may implement another *PREF* option: STRONGPREFERRED. Any intermediate host receiving such a route-search should forward it (by broadcast) unless its hop list contains a loop or its time-to-live has expired, even if the host has seen this request before (through a different path). The destination host should wait for a period of time to allow several route-search packets to arrive via different routes. The destination then selects the best one according to a certain criteria, such as shortest path with minimal weak links. The source host may choose this option to find a route if it is probable that no strong route exists.

Another optimization decreases route discovery latency and route-request propagation by allowing intermediate hosts to participate in route discovery. If an intermediate host receiving a route-request already has a route to the destination, it may send a route-reply immediately back to the source to decrease the latency in addition to forwarding the route-request to the destination. If this route is non-optimal or stale, it will be overwritten later by a route-reply from the destination. The source may decide whether it wants to risk losing a few packets to start transmitting data sooner.

4.2 Inter-operability

Assuming that some hosts in the ad-hoc network are base-stations with both wired and wireless connectivity, SSA can be easily integrated into the global Internet through Mobile-IP ([Per96]). These base-stations serve as the home and foreign agents for Mobile-IP.

Conceptually, Mobile-IP sits on top of SSA (Figure 6). As far as SSA is concerned, Mobile-IP is just another higher layer protocol. On the other hand, Mobile-IP treats SSA as a link-layer protocol, and from the viewpoint of Mobile-IP, a base station can "directly" reach any host that resides in the same ad-hoc network. SSA will take the responsibility of delivering packets between them. The base station will take care of any encapsulation or de-capsulation required by Mobile-IP.

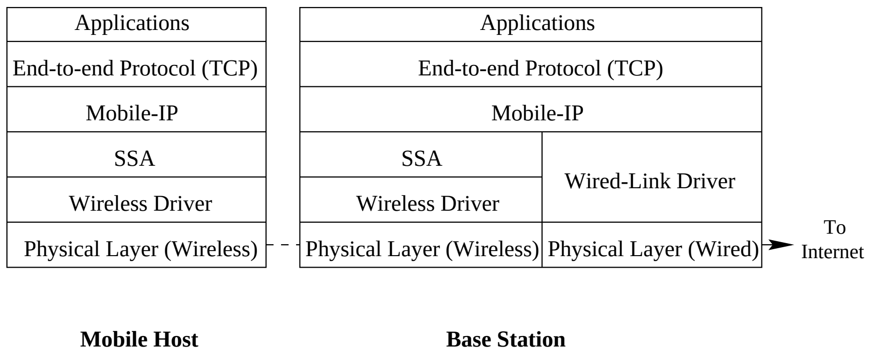

Figure 6: Integration of Mobile-IP and SSA

Mobile-IP on a base station may broadcast its agent advertisement, which should be heard by any potential client, in this case by any host within the ad-hoc network. In order to achieve this, all hosts in this ad-hoc network would be required to re-broadcast the agent advertisement, which takes a lot of resources. We suggest that a base station broadcast agent advertisements only rarely, or not at all, to conserve the limited bandwidth. If access to the wired network is not always available in the ad-hoc network, then occasional advertisements may be worthwhile. Otherwise, a mobile host should use agent solicitations when it desires the service of a base-station.

When sending an agent solicitation, the SSA on the mobile host should send a packet addressed to the SSA broadcast address. Non-base station hosts will simply re-broadcast the packet, while any reachable base-stations will send out a reply if willing to serve as an agent. The base-station's reply would trigger SSA to find a route to the base-station. After this handshake, the mobile host can continue with the usual registration process required by Mobile-IP and start sending and receiving datagrams to and from the Internet through the base station.

If a mobile host becomes separated from its agent, it may send out a new agent solicitation to try to find another reachable base-station. Alternatively, it may occasionally send agent solicitations to find other base-stations while maintaining a connection with the current one. Having this list of base-stations would decrease the latency of a base-station switch in case the first base-station becomes unreachable.

5 Simulation

We performed simulations to evaluate the benefits and costs of the SSA routing approach. The simulations quantify the length and longevity of the routes determined by SSA under various node densities and mobility rates. They also determine the relative efficacy of using signal strength and location stability as selection criteria for routing.

We studied a large range of cases by varying many of the input parameters, including: area, host density, transmission range, rate of topology change, pattern of individual host mobility, session length, and the routing algorithm criteria for route selection. We measured and compared the number of route reconstructions required, the average route hop length, the percentage of strong routes available, and the transmission cost of the SSA routing algorithm. We saw improved reconstruction costs for most sets of parameters. We present only a representative set of simulations in the following section.

5.1 Simulation Setup

Our simulation is based on the environment of a closed 1500×1500 unit area in which there are a number of randomly distributed mobile hosts. A signal is considered strong if it comes from a host *strong-radius* units or less away, and two hosts separated by more than *weak-radius* units are considered disconnected. Note that in a real environment, these quantities would be dependent on and controlled by the $SS_{threshold}$ set through the device driver interface and the capability of the wireless device respectively. In our simulation, we assume that the signal strength depends solely on the distance between the sending and receiving hosts. If a signal is weaker than that from a host weak-radius units away, then it would be considered noise and dropped at the physical layer. $SS_{threshold}$ is assumed to be equal to the strength of a signal from a host strong-radius units away. Since $SS_{threshold}$ is experimentally determined, we ran simulations with *strong-radius* 200 and 300 units. The weak-radius is kept constant at 400 units as it is a physical quantity dependent on the wireless device.

Twenty percent of the hosts are stationary during the simulation. The other eighty percent move for a number of clicks and then stay for some number of clicks and then move again, continuing the cycle. The lengths of the moving periods are normally distributed with average 10 clicks and standard deviation 1 click. There are two classes of staying period lengths. The short-stay class is normally distributed with average 3 clicks and standard deviation 1 click, and the long-stay class with average 150 and standard deviation 10. Initially every non-stationary host is assigned a probability that determines whether it falls into the short-stay or long-stay class each time it enters a staying period. The initial staying probability is chosen from a normal distribution with standard deviation 0.05 and a mean that ranges from 0 to 1 with step 0.1. A simulation is run for each step of the staying probability to obtain data for a range of mobility rates.

If a host moves during a certain click, it moves 20 units of distance in a randomly chosen direction. If it didn't move during the previous click, the direction is chosen from a uniformly distributed random number. Otherwise, the direction is chosen from a normally distributed random number with average equal to the previous direction and standard deviation of 10°. Hence, a host is likely to continue to move in the same general direction as previous movement. If a host hits the boundary of the area, it will bounce back (mirror reflection) so that the total moving distance during this time click is still 20 units.

We run simulations for networks of sizes 50, 100 and 200 hosts with $click_{threshold}$ equal to 5 and 1. $click_{threshold}$ determines the threshold above which routes are considered stable. Note that $click_{threshold}$ equal to 1 means that location stability is not considered. When considering location stability, this threshold should be slightly greater than the mean of the short-stay period so that only hosts which have a long-stay period are considered stable. A $click_{threshold}$ of 5 is chosen since the mean of the short-stay period equals 3.

During each run, we randomly generate the initial positions of the hosts and let them move and exchange beacon signals for 10 time clicks to establish an initial state. Then we randomly choose a source and a destination and run our algorithm. After each time click, we send a data packet through the established route to trigger route maintenance actions (route erasure and re-discovery) if any of the links failed. After each session (of length 300 clicks), we observe the average number of hops in a route and the number of required route reconstructions. These quantities are averaged over several hundred runs for each combination of the input parameters.

5.2 Simulation Results

We compare the results from SSA (with and without location stability) with those from an imaginary routing algorithm in which a shortest path is chosen, regardless of the strength of its links. We call the later approach the *simple algorithm.* The performance parameters are plotted against the average mobility rate, which is the average of all the host mobility rates. The mobility rate of a host is the number of clicks during which the host moves divided by the total number of clicks in a session. Clearly, this average mobility rate is inversely dependent on the average initial staying probability. As the mobility rate increases, the number of route reconstructions consistently increases, as shown in figures 7 to 9. The number of reconstructions also increases as the number of hosts decreases, due to the increasing sparseness of the topology. In the following graphs, we compare the simple algorithm to the SSA algorithm for the same number of hosts.

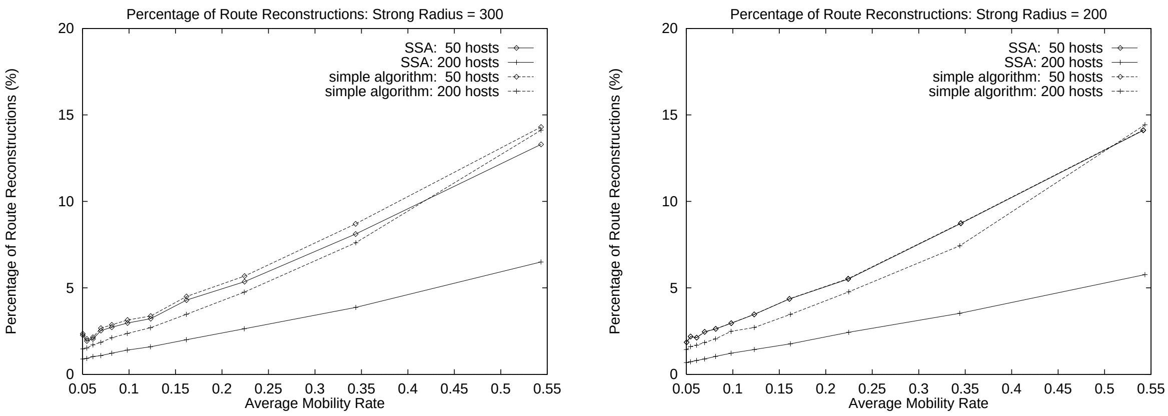

Figure 7: Number of Route Reconstructions: Stability Considered

Figures 7 and 8 show that the fraction of sessions requiring route reconstructions is consistently lower for SSA both with and without stability as compared to the simple algorithm. However SSA with location stability performs worse than that without location stability. At first glance, this is somewhat surprising. However figures 10 and 11 reveal that taking stability into account increases the probability of non-existence of strong routes considerably. This is because location stability introduces a much stronger criteria for a link to be SC. If we are unable to finds a strong route, route discovery takes long and the route likely fails sooner. The advantage of SSA arises from the *buffer-zone* effect. If a SC link is chosen as part of a route, will become WC before breaking, and this tends to give the individual links, and therefore the entire route, a longer life. The buffer-zone allows mobile hosts to roam within a certain vicinity of each other without triggering a route reconstruction.

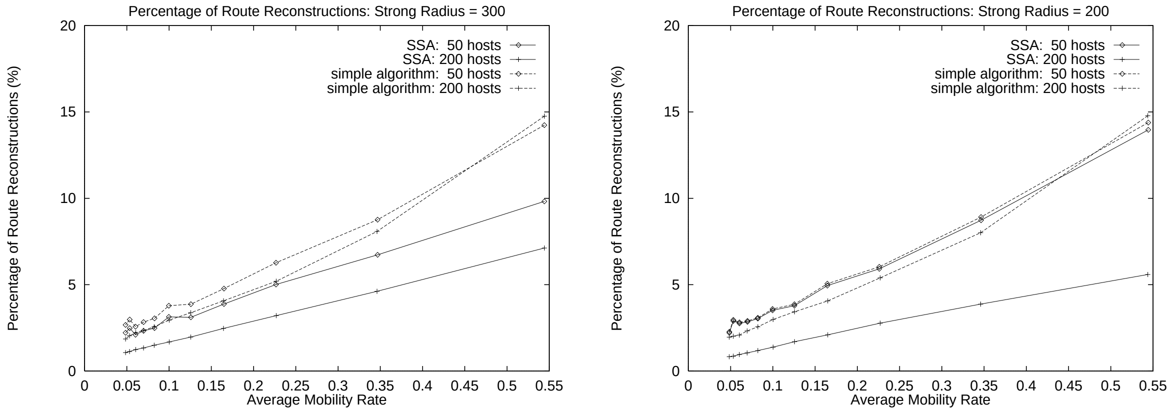

Figure 8: Number of Route Reconstructions: Stability Not Considered

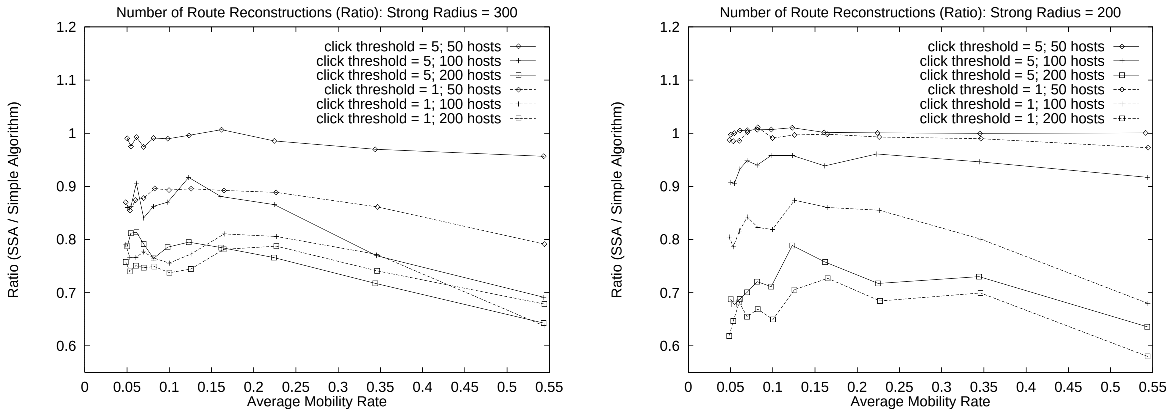

Figure 9: Improvement of SSA Over the Simple Algorithm

Comparing the SSA curves for 50 hosts in figure 8, we see that the performance with the larger strong radius is superior to that of the smaller one. This is somewhat contrary to the buffer-zone effect just described since the smaller radius allows the mobile hosts to travel a longer distance in the weak region before the link breaks. Figure 11 offers insight by showing that the percentage of non-existence of strong routes is more than 80% - 90% for a strong-radius of 200, whereas it is only about 20% - 30% for a strong-radius of 300. The decreased number of strong routes more than offsets any gains due to the increase in the buffer-zone.

Clearly, SSA performs well when there are an adequate number of strong routes. This, in turn, depends on the node density (the number of hosts in our environment), the strong radius, the mobility rate, and the criteria defining a strong link. Many combinations of these parameters result in a configuration where SSA drops the number of route reconstructions required. Figure 9 indicates that SSA reduces the route reconstructions needed by up to 40% and never performs worse than the simple algorithm. A careful comparison of the SSA curves with and without location stability considerations indicates that in most cases not taking stability into account results in better performance.

Since SSA prefers routes with strong links which are likely to be between two hosts close to each other, we tend to get routes with more hops as compared to the simple algorithm. On the other hand, strong links are less vulnerable to interference and hence result in less

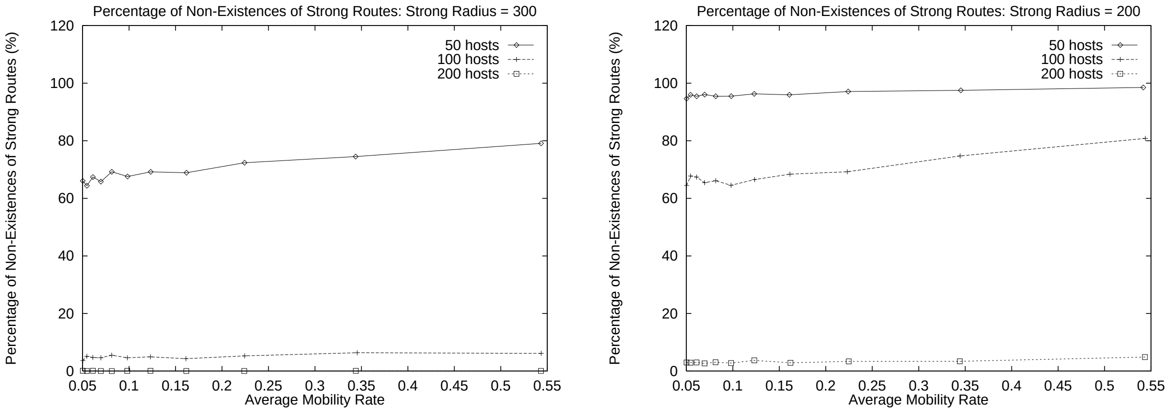

Figure 10: Probability of No Strong Routes: Stability Considered

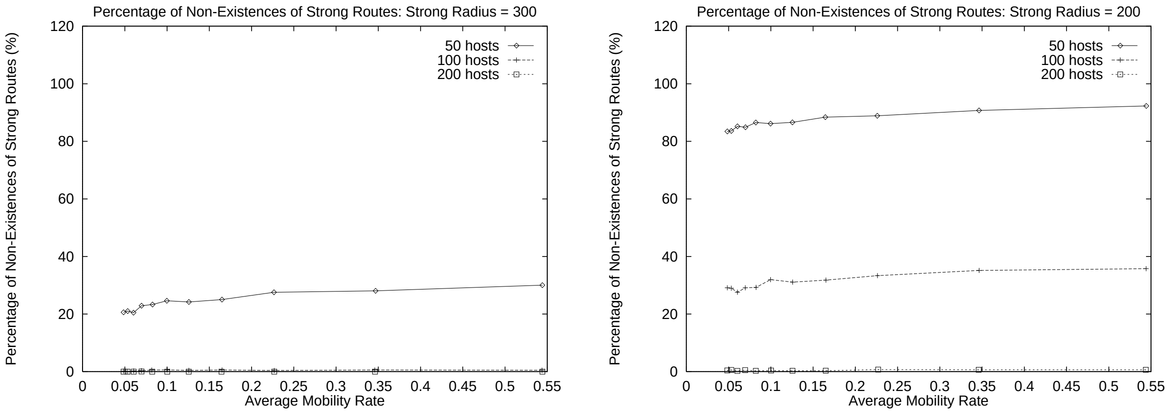

Figure 11: Probability of No Strong Routes: Stability Not Considered

packet loss and corruption. As a rough weight, we count each weak link as 1.25 hops to reflect its vulnerability. Figures 12 and 13 show the hop count ratio between SSA and the simple algorithm. Through the mobility rate range, the hop count ratio usually ranges between 1.1 and 1.5. Since this added hop length cost is small, SSA is beneficial since it consistently reduces the route reconstruction cost.

6 Related Work

For networks with static topologies, traditional routing techniques include link-state routing and distance vector routing. Routing in an ad-hoc network presents a challenge because the dynamic topology requires frequent updates and efficient routing information. If the rate of topology change is extremely dynamic then flooding becomes the only effective method for data transmission. Most routing approaches for ad-hoc networks assume a rate of change that is not so fast as to make flooding the only alternative. The requirements and difficulties of ad-hoc routing are discussed in [CMB96].

One type of ad-hoc routing methods seeks to modify existing routing algorithms for use in a dynamic topology. The Destination Sequenced Distance Vector algorithm [PB94] modifies the distributed Bellman-Ford algorithm to prevent looping by including sequence number to

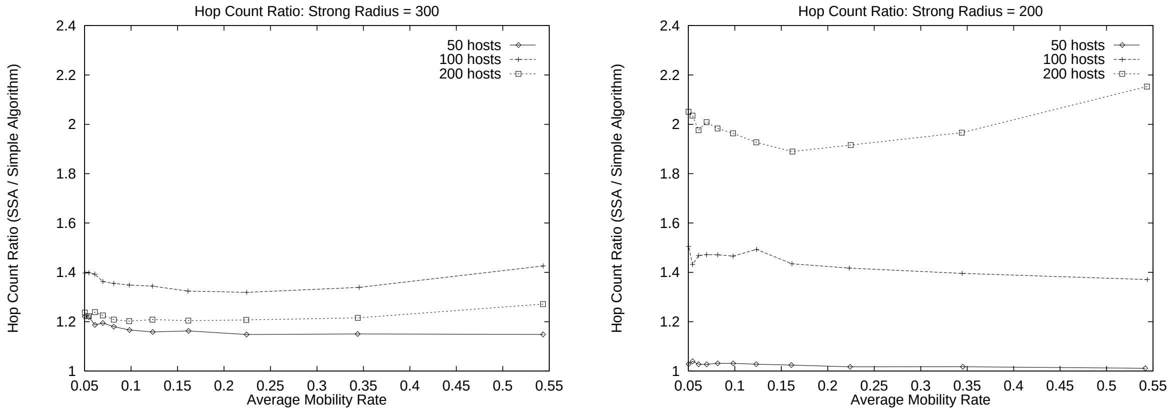

Figure 12: Hop Counts: Stability Considered

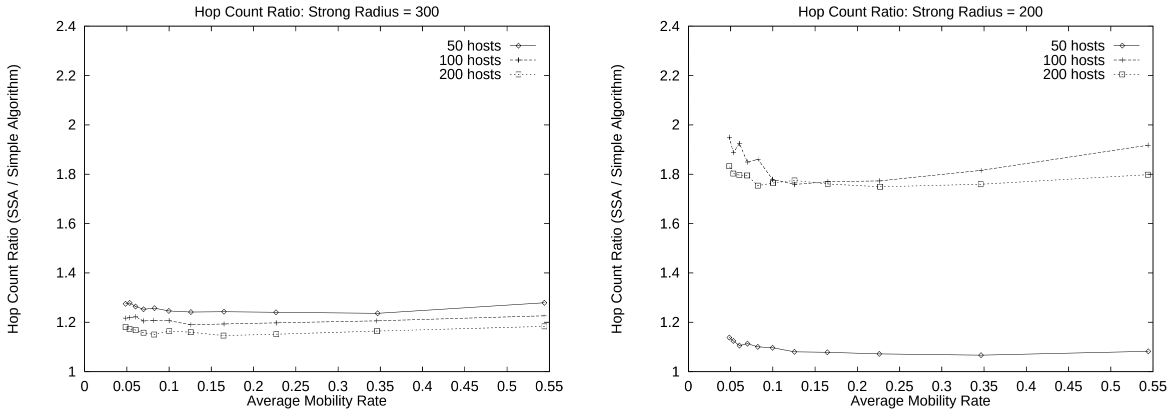

Figure 13: Hop Counts: Stability Not Considered

order the routing information. The Path Find algorithm in [MGLA95] is a distance vector routing protocol, which uses second-to-last hop to a destination to identify a route and prevent looping. The approach presented in [KCVP96] finds and maintains clusters in the ad-hoc network. The boundary nodes connect clusters and perform routing using a traditional routing algorithm. For all of these methods, each node, or the boundary nodes, needs to know the topology of the entire network at all times. Information is propagated through the network to achieve this goal.

Another type of ad-hoc routing algorithms uses an on demand philosophy. Routes to a destination are only sought if the node has something to send to that destination. The dynamic source routing proposed in [JM96] uses broadcasts to propagate the route request. Each node which forwards the route request adds its address to the source list. When the request reaches the destination a complete route is listed in the packet. The Lightweight Mobile Routing approach proposed in [CE95] floods the network with a query broadcast when the route is desired. The nodes receiving this query broadcast either forward the broadcast or broadcast a reply to the destination with the requested route. This approach uses link status to create routes and prevent looping. Route erasure is required when the topology changes. The Associativity Based Routing approach [Toh] also uses broadcast queries to find desired routes, but the optimal route is selected by the destination based on the stability of the route and shortest path. The main criteria is the intermediate node stability, which is based on the idea that nodes which have been stationary for a threshold period are less likely to move. This method also uses route erasure and maintenance when topology changes cause a route failure.

The main difference between these approaches and our approach is our utilization of the information available at the link level to choose routes. The quality of the channel is used to determine whether the topology is stable or fluctuating at any given time. Like the second type of algorithms, routes are determined only on demand. However, we do not limit the rate of change of the topology or suggest that all parts of the topology are equally stable. We select routes through the most stable areas of the network, using an an adaptive algorithm to ensure successful data transmission in a highly dynamic topology.

7 Conclusion and Future Work

The SSA protocol proposed in this paper focuses on obtaining the most stable routes through an ad-hoc network. This approach seeks to maximize the duration of the discovered routes. Our simulations have shown significant savings in the number of route reconstructions as a result of using signal strength to select routes. Using location stability, on the other hand, is shown to be very sensitive to the particular configuration of the ad-hoc network being considered. Since a general ad-hoc network is likely to have unpredictable and variable mobility patterns, we propose the adoption of signal strength as a criteria for routing with configurable parameters to take location stability into account where applicable. We plan to do further simulations using a packet level simulator to determine the costs and benefits of this approach more accurately and to study the effect of this approach on various transport protocols, including TCP.

Although it is intuitively clear that the algorithm is loop free and converges, we plan to prove these properties using theoretical constructs. We also plan to analyze convergence time and routing overhead for this algorithm.

Acknowledgements

We would like to thank Sambit Sahu (University of Massachusetts, Amherst) for his comments.

References

[CE95] S. Corson and A. Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *ACM/Baltzer Wireless Networks Journal*, 1(1), 1995.

[CMB96] S. Corson, J. Macker, and S. Batsell. Architectural Considerations for Mobile Mesh Networking. Internet Draft RFC Version 2—work in progress, May 1996.

[JM96] D.B. Johnson and D.A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, 3(1), February 1996.

[KCVP96] P. Krishna, M. Chatterjee, N.H. Vaidya, and D.K. Pradhan. A Cluster-based Approach for Routing in Ad-Hoc Networks. In *USENIX Technical Conference Proceedings*, January 1996.

[LNT87] B.M. Leiner, D.L. Nielson, and F.A. Tobagi. Issues in Packet Radio Network Design. *Proceedings of the IEEE*, 75(1), January 1987.

[MGLA95] S. Murthy and J.J. Garcia-Luna-Aceves. A Routing Protocol for Packet Radio Networks. In *MOBICOM Conference Proceedings*, November 1995.

[PB94] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *SIGCOMM Conference Proceedings*, 1994.

[Per96] C.E. Perkins, editor. *IP Mobility Support, draft-ietf-mobileip-protocol-15.txt.* Internet Engineering Task Force, February 1996. Internet draft—work in progress.

[RT87] J. Rubin and J.D. Tarnow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1), January 1987.

[Toh] C-K. Toh. Associativity-Based Routing for Ad-Hoc Mobile Networks. Personal communication.

Appendix: More Simulation Results

We also ran our simulation with strong radius equal to 100 units, an even higher signal strength threshold. As shown in Figures 14, 15 and 16, in this case there are virtually no strong routes, and this becomes so dominant a factor that SSA loses almost all of its advantage over the simple algorithm despite the larger "buffer zone".

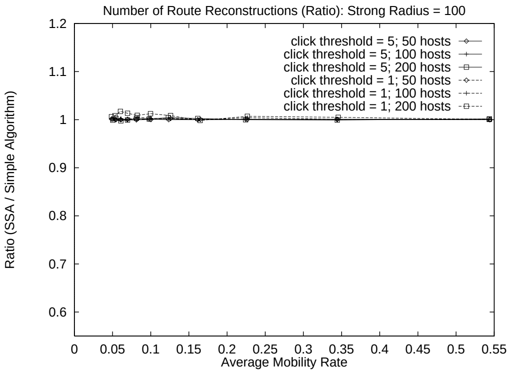

Figure 14: More Results: Improvement of SSA Over the Simple Algorithm

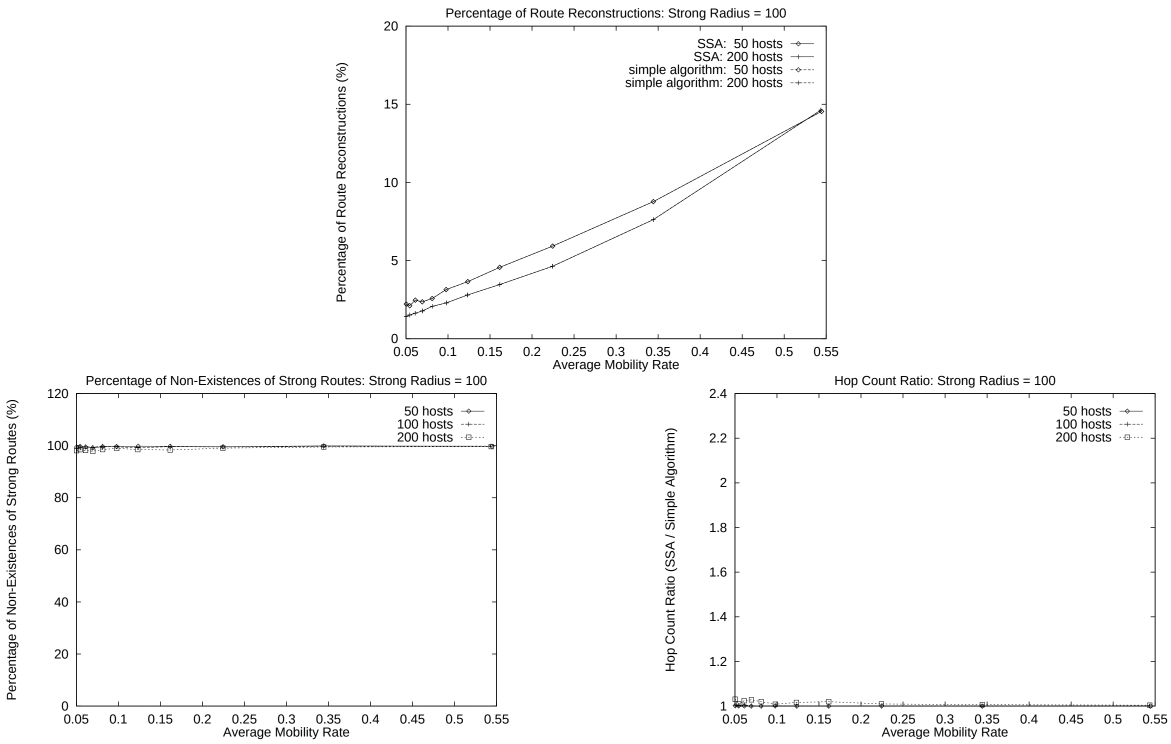

Figure 15: More Results: Stability Considered

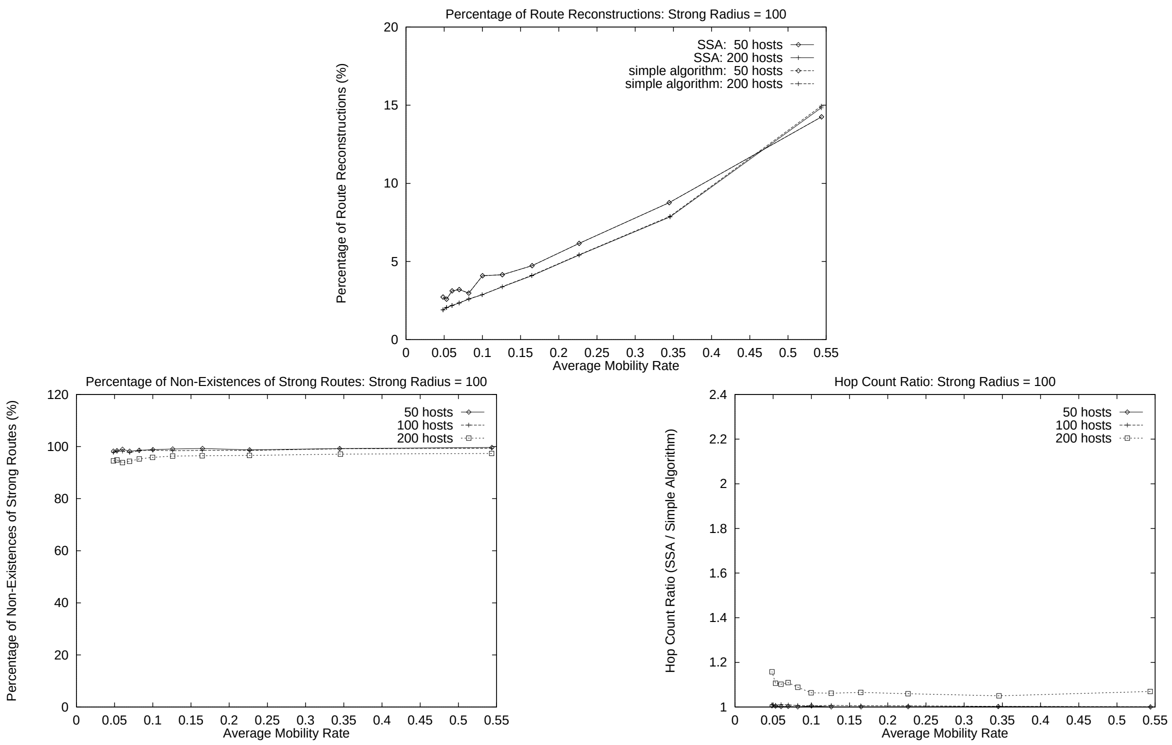

Figure 16: More Results: Stability Not Considered