IETF MANET Working Group David B. Johnson, Rice University
INTERNET-DRAFT David A. Maltz, AON Networks
21 February 2002 Yih-Chun Hu, Rice University
Jorjeta G. Jetcheva, Carnegie Mellon University

The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)

<draft-ietf-manet-dsr-07.txt>

Status of This Memo

This Internet-Draft is a submission to the IETF Mobile Ad Hoc Networks (MANET) Working Group. Comments on this draft may be sent to the Working Group at manet@itd.nrl.navy.mil, or may be sent directly to the authors.

Abstract

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the ad hoc network. The use of source routing allows packet routing to be trivially loop-free, avoids the need for up-to-date routing information in the intermediate nodes through which packets are forwarded, and allows nodes forwarding or overhearing packets to cache the routing information in them for their own future use. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use. This document specifies the operation of the DSR protocol for routing unicast IPv4 packets in multi-hop wireless ad hoc networks.

The DSR protocol is designed for mobile ad hoc networks with up to around two hundred nodes, and is designed to cope with relatively high rates of mobility.

Contents

1. Introduction

The Dynamic Source Routing protocol (DSR) [13, 14] is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. Using DSR, the network is completely self-organizing and self-configuring, requiring no existing network infrastructure or administration. Network nodes cooperate to forward packets for each other to allow communication over multiple "hops" between nodes not directly within wireless transmission range of one another. As nodes in the network move about or join or leave the network, and as wireless transmission conditions such as sources of interference change, all routing is automatically determined and maintained by the DSR routing protocol. Since the number or sequence of intermediate hops needed to reach any destination may change at any time, the resulting network topology may be quite rich and rapidly changing.

The DSR protocol allows nodes to dynamically discover a source route across multiple network hops to any destination in the ad hoc network. Each data packet sent then carries in its header the complete, ordered list of nodes through which the packet will pass, allowing packet routing to be trivially loop-free and avoiding the need for up-to-date routing information in the intermediate nodes through which the packet is forwarded. By including this source route in the header of each data packet, other nodes forwarding or overhearing any of these packets can also easily cache this routing information for future use.

In designing DSR, we sought to create a routing protocol that had very low overhead yet was able to react very quickly to changes in the network. The DSR protocol provides highly reactive service in order to help ensure successful delivery of data packets in spite of node movement or other changes in network conditions.

The DSR protocol is composed of two main mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

- Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D.

- Route Maintenance is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke Route Discovery again to find a new route for subsequent packets to D.

Route Maintenance for this route is used only when S is actually sending packets to D.

In DSR, Route Discovery and Route Maintenance each operate entirely "on demand". In particular, unlike other protocols, DSR requires no periodic packets of any kind at any layer within the network. For example, DSR does not use any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. This entirely on-demand behavior and lack of periodic activity allows the number of overhead packets caused by DSR to scale all the way down to zero, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR automatically scales to only that needed to track the routes currently in use. Network topology changes not affecting routes currently in use are ignored and do not cause reaction from the protocol.

In response to a single Route Discovery (as well as through routing information from other packets overheard), a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail. This caching of multiple routes also avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks.

The operation of both Route Discovery and Route Maintenance in DSR are designed to allow unidirectional links and asymmetric routes to be easily supported. In particular, as noted in Section 2, in wireless networks, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or sources of interference. DSR allows such unidirectional links to be used when necessary, improving overall performance and network connectivity in the system.

This document specifies the operation of the DSR protocol for routing unicast IPv4 packets in multi-hop wireless ad hoc networks. Advanced, optional features, such as Quality of Service (QoS) support and efficient multicast routing, and operation of DSR with IPv6 [6], are covered in other documents. The specification of DSR in this document provides a compatible base on which such features can be added, either independently or by integration with the DSR operation specified here.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [4].

2. Assumptions

We assume in this document that all nodes wishing to communicate with other nodes within the ad hoc network are willing to participate fully in the protocols of the network. In particular, each node participating in the ad hoc network SHOULD also be willing to forward packets for other nodes in the network.

The diameter of an ad hoc network is the minimum number of hops necessary for a packet to reach from any node located at one extreme edge of the ad hoc network to another node located at the opposite extreme. We assume that this diameter will often be small (e.g., perhaps 5 or 10 hops), but may often be greater than 1.

Packets may be lost or corrupted in transmission on the wireless network. We assume that a node receiving a corrupted packet can detect the error and discard the packet.

Nodes within the ad hoc network MAY move at any time without notice, and MAY even move continuously, but we assume that the speed with which nodes move is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. In particular, DSR can support very rapid rates of arbitrary node mobility, but we assume that nodes do not continuously move so rapidly as to make the flooding of every individual data packet the only possible routing protocol.

A common feature of many network interfaces, including most current LAN hardware for broadcast media such as wireless, is the ability to operate the network interface in "promiscuous" receive mode. This mode causes the hardware to deliver every received packet to the network driver software without filtering based on link-layer destination address. Although we do not require this facility, some of our optimizations can take advantage of its availability. Use of promiscuous mode does increase the software overhead on the CPU, but we believe that wireless network speeds are more the inherent limiting factor to performance in current and future systems; we also believe that portions of the protocol are suitable for implementation directly within a programmable network interface unit to avoid this overhead on the CPU [14]. Use of promiscuous mode may also increase the power consumption of the network interface hardware, depending on the design of the receiver hardware, and in such cases, DSR can easily be used without the optimizations that depend on promiscuous receive mode, or can be programmed to only periodically switch the interface into promiscuous mode. Use of promiscuous receive mode is entirely optional.

Wireless communication ability between any pair of nodes may at times not work equally well in both directions, due for example to differing antenna or propagation patterns or sources of interference around the two nodes [1, 18]. That is, wireless communications between each pair of nodes will in many cases be able to operate bidirectionally, but at times the wireless link between two nodes may be only unidirectional, allowing one node to successfully send packets to the other while no communication is possible in the reverse direction. Although many routing protocols operate correctly only over bidirectional links, DSR can successfully discover and forward packets over paths that contain unidirectional links. Some MAC protocols, however, such as MACA [17], MACAW [2], or IEEE 802.11 [11], limit unicast data packet transmission to bidirectional links, due to the required bidirectional exchange of RTS and CTS packets in these protocols and due to the link-layer acknowledgment feature in IEEE 802.11; when used on top of MAC protocols such as these, DSR can take advantage of additional optimizations, such as the ability to reverse a source route to obtain a route back to the origin of the original route.

The IP address used by a node using the DSR protocol MAY be assigned by any mechanism (e.g., static assignment or use of DHCP for dynamic assignment [7]), although the method of such assignment is outside the scope of this specification.

3. DSR Protocol Overview

3.1. Basic DSR Route Discovery

When some source node originates a new packet addressed to some destination node, the source node places in the header of the packet a source route giving the sequence of hops that the packet is to follow on its way to the destination. Normally, the sender will obtain a suitable source route by searching its "Route Cache" of routes previously learned; if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to this destination node. In this case, we call the source node the "initiator" and the destination node the "target" of the Route Discovery.

For example, suppose a node A is attempting to discover a route to node E. The Route Discovery initiated by node A in this example would proceed as follows:

```
            ^    "A"    ^   "A,B"   ^  "A,B,C"  ^ "A,B,C,D"
            |   id=2    |   id=2    |   id=2    |   id=2
         +------+    +------+    +------+    +------+    +------+
         |  A   |---->|  B   |---->|  C   |---->|  D   |---->|  E   |
         +------+    +------+    +------+    +------+    +------+
            |           |           |           |
            v           v           v           v
```

To initiate the Route Discovery, node A transmits a "Route Request" as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of A, including node B in this example. Each Route Request identifies the initiator and target of the Route Discovery, and also contains a unique request identification (2, in this example), determined by the initiator of the Request. Each Route Request also contains a record listing the address of each intermediate node through which this particular copy of the Route Request has been forwarded. This route record is initialized to an empty list by the initiator of the Route Discovery. In this example, the route record initially lists only node A.

When another node receives this Route Request (such as node B in this example), if it is the target of the Route Discovery, it returns a "Route Reply" to the initiator of the Route Discovery, giving a copy of the accumulated route record from the Route Request; when the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination.

Otherwise, if this node receiving the Route Request has recently seen another Route Request message from this initiator bearing this same request identification and target address, or if this node's own address is already listed in the route record in the Route Request, this node discards the Request. Otherwise, this node appends its own address to the route record in the Route Request and propagates it by transmitting it as a local broadcast packet (with the same request identification). In this example, node B broadcast the Route Request, which is received by node C; nodes C and D each also, in turn, broadcast the Request, resulting in a copy of the Request being received by node E.

In returning the Route Reply to the initiator of the Route Discovery, such as in this example, node E replying back to node A, node E will typically examine its own Route Cache for a route back to A, and if found, will use it for the source route for delivery of the packet containing the Route Reply. Otherwise, E SHOULD perform its own Route Discovery for target node A, but to avoid possible infinite recursion of Route Discoveries, it MUST piggyback this Route Reply on the packet containing its own Route Request for A. It is also possible to piggyback other small data packets, such as a TCP SYN packet [28], on a Route Request using this same mechanism.

Node E could instead simply reverse the sequence of hops in the route record that it is trying to send in the Route Reply, and use this as the source route on the packet carrying the Route Reply itself. For MAC protocols such as IEEE 802.11 that require a bidirectional frame exchange as part of the MAC protocol [11], the discovered source route MUST be reversed in this way to return the Route Reply since it tests the discovered route to ensure it is bidirectional before the Route Discovery initiator begins using the route; this route reversal also avoids the overhead of a possible second Route Discovery. However, this route reversal technique will prevent the discovery of routes using unidirectional links, and in wireless environments where the use of unidirectional links is permitted, such routes may in some cases be more efficient than those with only bidirectional links, or they may be the only way to achieve connectivity to the target node.

When initiating a Route Discovery, the sending node saves a copy of the original packet (that triggered the Discovery) in a local buffer called the "Send Buffer". The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the Send Buffer is logically associated with the time that it was placed into the Send Buffer and is discarded after residing in the Send Buffer for some timeout period; if necessary for preventing the Send Buffer from overflowing, a FIFO or other replacement strategy MAY also be used to evict packets even before they expire.

While a packet remains in the Send Buffer, the node SHOULD occasionally initiate a new Route Discovery for the packet's destination address. However, the node MUST limit the rate at which such new Route Discoveries for the same address are initiated, since it is possible that the destination node is not currently reachable. In particular, due to the limited wireless transmission range and the movement of the nodes in the network, the network may at times become partitioned, meaning that there is currently no sequence of nodes through which a packet could be forwarded to reach the destination. Depending on the movement pattern and the density of nodes in the network, such network partitions may be rare or may be common.

If a new Route Discovery was initiated for each packet sent by a node in such a partitioned network, a large number of unproductive Route Request packets would be propagated throughout the subset of the ad hoc network reachable from this node. In order to reduce the overhead from such Route Discoveries, a node SHOULD use an exponential back-off algorithm to limit the rate at which it initiates new Route Discoveries for the same target, doubling the timeout between each successive Discovery initiated for the same target. If the node attempts to send additional data packets to this same destination node more frequently than this limit, the subsequent packets SHOULD be buffered in the Send Buffer until a Route Reply is received giving a route to this destination, but the node MUST NOT initiate a new Route Discovery until the minimum allowable interval between new Route Discoveries for this target has been reached. This limitation on the maximum rate of Route Discoveries for the same target is similar to the mechanism required by Internet nodes to limit the rate at which ARP Requests are sent for any single target IP address [3].

3.2. Basic DSR Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that data can flow over the link from that node to the next hop. For example, in the situation shown below, node A has originated a packet for node E using a source route through intermediate nodes B, C, and D:

```
      +------+     +------+     +------+     +------+     +------+
      |  A   |---->|  B   |---->|  C   |-->? |  D   |     |  E   |
      +------+     +------+     +------+     +------+     +------+
```

In this case, node A is responsible for the link from A to B, node B is responsible for the link from B to C, node C is responsible for the link from C to D, node D is responsible for the link from D to E.

An acknowledgment can provide confirmation that a link is capable of carrying data, and in wireless networks, acknowledgments are often provided at no cost, either as an existing standard part of the MAC protocol in use (such as the link-layer acknowledgment frame defined by IEEE 802.11 [11]), or by a "passive acknowledgment" [16] (in which, for example, B confirms receipt at C by overhearing C transmit the packet when forwarding it on to D).

If a built-in acknowledgment mechanism is not available, the node transmitting the packet can explicitly request a DSR-specific software acknowledgment be returned by the next node along the route; this software acknowledgment will normally be transmitted directly to the sending node, but if the link between these two nodes is unidirectional, this software acknowledgment could travel over a different, multi-hop path.

After an acknowledgment has been received from some neighbor, a node MAY choose to not require acknowledgments from that neighbor for a brief period of time, unless the network interface connecting a node to that neighbor always receives an acknowledgment in response to unicast traffic.

When a software acknowledgment is used, the acknowledgment request SHOULD be retransmitted up to a maximum number of times. A retransmission of the acknowledgment request can be sent as a separate packet, piggybacked on a retransmission of the original data packet, or piggybacked on any packet with the same next-hop destination that does not also contain a software acknowledgment.

After the acknowledgment request has been retransmitted the maximum number of times, if no acknowledgment has been received, then the sender treats the link to this next-hop destination as currently "broken". It SHOULD remove this link from its Route Cache and SHOULD return a "Route Error" to each node that has sent a packet routed over that link since an acknowledgment was last received. For example, in the situation shown above, if C does not receive an acknowledgment from D after some number of requests, it would return a Route Error to A, as well as any other node that may have used the link from C to D since C last received an acknowledgment from D. Node A then removes this broken link from its cache; any retransmission of the original packet can be performed by upper layer protocols such as TCP, if necessary. For sending such a retransmission or other packets to this same destination E, if A has in its Route Cache another route to E (for example, from additional Route Replies from its earlier Route Discovery, or from having overheard sufficient routing information from other packets), it can send the packet using the new route immediately. Otherwise, it SHOULD perform a new Route Discovery for this target (subject to the back-off described in Section 3.1).

3.3. Additional Route Discovery Features

3.3.1. Caching Overheard Routing Information

A node forwarding or otherwise overhearing any packet SHOULD add all usable routing information from that packet to its own Route Cache. The usefulness of routing information in a packet depends on the directionality characteristics of the physical medium (Section 2), as well as the MAC protocol being used. Specifically, three distinct cases are possible:

- Links in the network frequently are capable of operating only unidirectionally (not bidirectionally), and the MAC protocol in use in the network is capable of transmitting unicast packets over unidirectional links.

- Links in the network occasionally are capable of operating only unidirectionally (not bidirectionally), but this unidirectional restriction on any link is not persistent, almost all links are physically bidirectional, and the MAC protocol in use in the network is capable of transmitting unicast packets over unidirectional links.

- The MAC protocol in use in the network is not capable of transmitting unicast packets over unidirectional links; only bidirectional links can be used by the MAC protocol for transmitting unicast packets. For example, the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol [11] is capable of transmitting a unicast packet only over a bidirectional link, since the MAC protocol requires the return of a link-level acknowledgment packet from the receiver and also optionally requires the bidirectional exchange of an RTS and CTS packet between the transmitter and receiver nodes.

In the first case above, for example, the source route used in a data packet, the accumulated route record in a Route Request, or the route being returned in a Route Reply SHOULD all be cached by any node in the "forward" direction; any node SHOULD cache this information from any such packet received, whether the packet was addressed to this node, sent to a broadcast (or multicast) MAC address, or overheard while the node's network interface is in promiscuous mode. However, the "reverse" direction of the links identified in such packet headers SHOULD NOT be cached.

For example, in the situation shown below, node A is using a source route to communicate with node E:

```
      +-----+     +-----+     +-----+     +-----+     +-----+
      |  A  |---->|  B  |---->|  C  |---->|  D  |---->|  E  |
      +-----+     +-----+     +-----+     +-----+     +-----+
```

As node C forwards a data packet along the route from A to E, it SHOULD add to its cache the presence of the "forward" direction links that it learns from the headers of these packets, from itself to D and from D to E. Node C SHOULD NOT, in this case, cache the "reverse" direction of the links identified in these packet headers, from itself back to B and from B to A, since these links might be unidirectional.

In the second case above, in which links may occasionally operate unidirectionally, the links described above SHOULD be cached in both directions. Furthermore, in this case, if node X overhears (e.g., through promiscuous mode) a packet transmitted by node C that is using a source route from node A to E, node X SHOULD cache all of these links as well, also including the link from C to X over which it overheard the packet.

In the final case, in which the MAC protocol requires physical bidirectionality for unicast operation, links from a source route SHOULD be cached in both directions, except when the packet also contains a Route Reply, in which case only the links already traversed in this source route SHOULD be cached, but the links not yet traversed in this route SHOULD NOT be cached.

3.3.2. Replying to Route Requests using Cached Routes

A node receiving a Route Request for which it is not the target, searches its own Route Cache for a route to the target of the Request. If found, the node generally returns a Route Reply to the initiator itself rather than forwarding the Route Request. In the Route Reply, this node sets the route record to list the sequence of hops over which this copy of the Route Request was forwarded to it, concatenated with the source route to this target obtained from its own Route Cache.

However, before transmitting a Route Reply packet that was generated using information from its Route Cache in this way, a node MUST verify that the resulting route being returned in the Route Reply, after this concatenation, contains no duplicate nodes listed in the route record. For example, the figure below illustrates a case in

which a Route Request for target E has been received by node F, and node F already has in its Route Cache a route from itself to E:

```
      +-----+     +-----+                 +-----+     +-----+
      |  A  |---->|  B  |-               >|  D  |---->|  E  |
      +-----+     +-----+ \             / +-----+     +-----+
                           \           /
                            \ +-----+ /
                             >|  C  |-
                              +-----+
                                | ^
                                v |
        Route Request         +-----+
        Route: A - B - C - F  |  F  |  Cache: C - D - E
                              +-----+
```

The concatenation of the accumulated route record from the Route Request and the cached route from F's Route Cache would include a duplicate node in passing from C to F and back to C.

Node F in this case could attempt to edit the route to eliminate the duplication, resulting in a route from A to B to C to D and on to E, but in this case, node F would not be on the route that it returned in its own Route Reply. DSR Route Discovery prohibits node F from returning such a Route Reply from its cache; this prohibition increases the probability that the resulting route is valid, since node F in this case should have received a Route Error if the route had previously stopped working. Furthermore, this prohibition means that a future Route Error traversing the route is very likely to pass through any node that sent the Route Reply for the route (including node F), which helps to ensure that stale data is removed from caches (such as at F) in a timely manner; otherwise, the next Route Discovery initiated by A might also be contaminated by a Route Reply from F containing the same stale route. If node F, due to this restriction on returning a Route Reply based on information from its Route Cache, does not return such a Route Reply, node F propagates the Route Request normally.

3.3.3. Preventing Route Reply Storms

The ability for nodes to reply to a Route Request based on information in their Route Caches, as described in Section 3.3.2, could result in a possible Route Reply "storm" in some cases. In particular, if a node broadcasts a Route Request for a target node for which the node's neighbors have a route in their Route Caches, each neighbor may attempt to send a Route Reply, thereby wasting bandwidth and possibly increasing the number of network collisions in the area.

For example, the figure below shows a situation in which nodes B, C, D, E, and F all receive A's Route Request for target G, and each has the indicated route cached for this target:

```
             +-----+                 +-----+
             |  D  |<               >|  C  |
             +-----+ \             / +-----+
   Cache: C - B - G   \           /  Cache: B - G
                       \ +-----+ /
                        -|  A  |-
                         +-----+\     +-----+     +-----+
                          |   |  \--->|  B  |     |  G  |
                         /     \      +-----+     +-----+
                        /       \     Cache: G
                       v         v
                 +-----+         +-----+
                 |  E  |         |  F  |
                 +-----+         +-----+
            Cache: F - B - G     Cache: B - G
```

Normally, each of these nodes would attempt to reply from its own Route Cache, and they would thus all send their Route Replies at about the same time, since they all received the broadcast Route Request at about the same time. Such simultaneous Route Replies from different nodes all receiving the Route Request may cause local congestion in the wireless network and may create packet collisions among some or all of these Replies if the MAC protocol in use does not provide sufficient collision avoidance for these packets. In addition, it will often be the case that the different replies will indicate routes of different lengths, as shown in this example.

In order to reduce these effects, if a node can put its network interface into promiscuous receive mode, it MAY delay sending its own Route Reply for a short period, while listening to see if the initiating node begins using a shorter route first. Specifically, this node MAY delay sending its own Route Reply for a random period

$$d = H * (h - 1 + r)$$

where h is the length in number of network hops for the route to be returned in this node's Route Reply, r is a random floating point number between 0 and 1, and H is a small constant delay (at least twice the maximum wireless link propagation delay) to be introduced per hop. This delay effectively randomizes the time at which each node sends its Route Reply, with all nodes sending Route Replies giving routes of length less than h sending their Replies before this node, and all nodes sending Route Replies giving routes of length greater than h sending their Replies after this node.

Within the delay period, this node promiscuously receives all packets, looking for data packets from the initiator of this Route Discovery destined for the target of the Discovery. If such a data packet received by this node during the delay period uses a source route of length less than or equal to h, this node may infer that the initiator of the Route Discovery has already received a Route Reply giving an equally good or better route. In this case, this node SHOULD cancel its delay timer and SHOULD NOT send its Route Reply for this Route Discovery.

3.3.4. Route Request Hop Limits

Each Route Request message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the Route Request. This hop limit is implemented using the Time-to-Live (TTL) field in the IP header of the packet carrying the Route Request. As the Request is forwarded, this limit is decremented, and the Request packet is discarded if the limit reaches zero before finding the target. This Route Request hop limit can be used to implement a variety of algorithms for controlling the spread of a Route Request during a Route Discovery attempt.

For example, a node MAY use this hop limit to implement a "non-propagating" Route Request as an initial phase of a Route Discovery. A node using this technique sends its first Route Request attempt for some target node using a hop limit of 1, such that any node receiving the initial transmission of the Route Request will not forward the Request to other nodes by re-broadcasting it. This form of Route Request is called a "non-propagating" Route Request; it provides an inexpensive method for determining if the target is currently a neighbor of the initiator or if a neighbor node has a route to the target cached (effectively using the neighbors' Route Caches as an extension of the initiator's own Route Cache). If no Route Reply is received after a short timeout, then the node sends a "propagating" Route Request (i.e., with no hop limit) for the target node.

As another example, a node MAY use this hop limit to implement an "expanding ring" search for the target [14]. A node using this technique sends an initial non-propagating Route Request as described above; if no Route Reply is received for it, the node originates another Route Request with a hop limit of 2. For each Route Request originated, if no Route Reply is received for it, the node doubles the hop limit used on the previous attempt, to progressively explore for the target node without allowing the Route Request to propagate over the entire network. However, this expanding ring search approach could have the effect of increasing the average latency of

Route Discovery, since multiple Discovery attempts and timeouts may be needed before discovering a route to the target node.

3.4. Additional Route Maintenance Features

3.4.1. Packet Salvaging

When an intermediate node forwarding a packet detects through Route Maintenance that the next hop along the route for that packet is broken, if the node has another route to the packet's destination in its Route Cache, the node SHOULD "salvage" the packet rather than discarding it. To salvage a packet, the node replaces the original source route on the packet with the route from its Route Cache. The node then forwards the packet to the next node indicated along this source route. For example, in the situation shown in the example of Section 3.2, if node C has another route cached to node E, it can salvage the packet by replacing the original route in the packet with this new route from its own Route Cache, rather than discarding the packet.

When salvaging a packet, a count is maintained in the packet of the number of times that it has been salvaged, to prevent a single packet from being salvaged endlessly. Otherwise, it could be possible for the packet to enter a routing loop, as different nodes repeatedly salvage the packet and replace the source route on the packet with routes to each other.

As described in Section 3.2, an intermediate node, such as in this case, that detects through Route Maintenance that the next hop along the route for a packet that it is forwarding is broken, the node also SHOULD return a Route Error to the original sender of the packet, identifying the link over which the packet could not be forwarded. If the node sends this Route Error, it SHOULD originate the Route Error before salvaging the packet.

3.4.2. Queued Packets Destined over a Broken Link

When an intermediate node forwarding a packet detects through Route Maintenance that the next-hop link along the route for that packet is broken, in addition to handling that packet as defined for Route Maintenance, the node SHOULD also handle in a similar way any pending packets that it has queued that are destined over this new broken link. Specifically, the node SHOULD search its Network Interface Queue and Maintenance Buffer (Section 4.5) for packets for which the next-hop link is this new broken link. For each such packet currently queued at this node, the node SHOULD process that packet as follows:

- Remove the packet from the node's Network Interface Queue and Maintenance Buffer.

- Originate a Route Error for this packet to the original sender of the packet, using the procedure described in Section 6.3.4, as if the node had already reached the maximum number of retransmission attempts for that packet for Route Maintenance. However, in sending such Route Errors for queued packets in response to a single new broken link detected, the node SHOULD send no more than one Route Error to each original sender of any of these packets.

- If the node has another route to the packet's IP Destination Address in its Route Cache, the node SHOULD salvage the packet as described in Section 6.3.6. Otherwise, the node SHOULD discard the packet.

3.4.3. Automatic Route Shortening

Source routes in use MAY be automatically shortened if one or more intermediate nodes in the route become no longer necessary. This mechanism of automatically shortening routes in use is somewhat similar to the use of passive acknowledgments [16]. In particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in promiscuous receive mode), then this node examines the unexpended portion of that source route. If this node is not the intended next-hop destination for the packet but is named in the later unexpended portion of the packet's source route, then it can infer that the intermediate nodes before itself in the source route are no longer needed in the route. For example, the figure below illustrates an example in which node D has overheard a data packet being transmitted from B to C, for later forwarding to D and to E:

```
      +-----+     +-----+     +-----+     +-----+     +-----+
      |  A  |---->|  B  |---->|  C  |     |  D  |     |  E  |
      +-----+     +-----+     +-----+     +-----+     +-----+
                     \                       ^
                      \                     /
                       ---------------------
```

In this case, this node (node D) SHOULD return a "gratuitous" Route Reply to the original sender of the packet (node A). The Route Reply gives the shorter route as the concatenation of the portion of the original source route up through the node that transmitted the overheard packet (node B), plus the suffix of the original source route beginning with the node returning the gratuitous Route Reply (node D). In this example, the route returned in the gratuitous Route Reply message sent from D to A gives the new route as the sequence of hops from A to B to D to E.

When deciding whether to return a gratuitous Route Reply in this way, a node MAY factor in additional information beyond the fact that it was able to overhear the packet. For example, the node MAY decide to return the gratuitous Route Reply only when the overheard packet is received with a signal strenth or signal-to-noise ratio above some specific threshold. In addition, each node maintains a Gratuitous Route Reply Table, as described in Section 4.4, to limit the rate at which it originates gratuitous Route Replies for the same returned route.

3.4.4. Increased Spreading of Route Error Messages

When a source node receives a Route Error for a data packet that it originated, this source node propagates this Route Error to its neighbors by piggybacking it on its next Route Request. In this way, stale information in the caches of nodes around this source node will not generate Route Replies that contain the same invalid link for which this source node received the Route Error.

For example, in the situation shown in the example of Section 3.2, node A learns from the Route Error message from C, that the link from C to D is currently broken. It thus removes this link from its own Route Cache and initiates a new Route Discovery (if it has no other route to E in its Route Cache). On the Route Request packet initiating this Route Discovery, node A piggybacks a copy of this Route Error, ensuring that the Route Error spreads well to other nodes, and guaranteeing that any Route Reply that it receives (including those from other node's Route Caches) in response to this Route Request does not contain a route that assumes the existence of this broken link.

4. Conceptual Data Structures

This document describes the operation of the DSR protocol in terms of a number of conceptual data structures. This section describes each of these data structures and provides an overview of its use in the protocol. In an implementation of the protocol, these data structures MAY be implemented in any manner consistent with the external behavior described in this document.

4.1. Route Cache

All ad hoc network routing information needed by a node implementing DSR is stored in that node's Route Cache. Each node in the network maintains its own Route Cache. A node adds information to its Route Cache as it learns of new links between nodes in the ad hoc network; for example, a node may learn of new links when it receives a packet carrying a Route Request, Route Reply, or DSR source route. Likewise, a node removes information from its Route Cache as it learns that existing links in the ad hoc network have broken; for example, a node may learn of a broken link when it receives a packet carrying a Route Error or through the link-layer retransmission mechanism reporting a failure in forwarding a packet to its next-hop destination.

Anytime a node adds new information to its Route Cache, the node SHOULD check each packet in its own Send Buffer (Section 4.2) to determine whether a route to that packet's IP Destination Address now exists in the node's Route Cache (including the information just added to the Cache). If so, the packet SHOULD then be sent using that route and removed from the Send Buffer.

It is possible to interface a DSR network with other networks, external to this DSR network. Such external networks may, for example, be the Internet, or may be other ad hoc networks routed with a routing protocol other than DSR. Such external networks may also be other DSR networks that are treated as external networks in order to improve scalability. The complete handling of such external networks is beyond the scope of this document. However, this document specifies a minimal set of requirements and features necessary to allow nodes only implementing this specification to interoperate correctly with nodes implementing interfaces to such external networks. This minimal set of requirements and features involve the First Hop External (F) and Last Hop External (L) bits in a DSR Source Route option (Section 5.7) and a Route Reply option (Section 5.3) in a packet's DSR header (Section 5). These requirements also include the addition of an External flag bit tagging each link in the Route Cache, copied from the First Hop

External (F) and Last Hop External (L) bits in the DSR Source Route option or Route Reply option from which this link was learned.

The Route Cache SHOULD support storing more than one route to each destination. In searching the Route Cache for a route to some destination node, the Route Cache is indexed by destination node address. The following properties describe this searching function on a Route Cache:

- Each implementation of DSR at any node MAY choose any appropriate strategy and algorithm for searching its Route Cache and selecting a "best" route to the destination from among those found. For example, a node MAY choose to select the shortest route to the destination (the shortest sequence of hops), or it MAY use an alternate metric to select the route from the Cache.

- However, if there are multiple cached routes to a destination, the selection of routes when searching the Route Cache MUST prefer routes that do not have the External flag set on any link. This preference will select routes that lead directly to the target node over routes that attempt to reach the target via any external networks connected to the DSR ad hoc network.

- In addition, any route selected when searching the Route Cache MUST NOT have the External bit set for any links other than possibly the first link, the last link, or both; the External bit MUST NOT be set for any intermediate hops in the route selected.

An implementation of a Route Cache MAY provide a fixed capacity for the cache, or the cache size MAY be variable. The following properties describe the management of available space within a node's Route Cache:

- Each implementation of DSR at each node MAY choose any appropriate policy for managing the entries in its Route Cache, such as when limited cache capacity requires a choice of which entries to retain in the Cache. For example, a node MAY chose a "least recently used" (LRU) cache replacement policy, in which the entry last used longest ago is discarded from the cache if a decision needs to be made to allow space in the cache for some new entry being added.

- However, the Route Cache replacement policy SHOULD allow routes to be categorized based upon "preference", where routes with a higher preferences are less likely to be removed from the cache. For example, a node could prefer routes for which it initiated a Route Discovery over routes that it learned as the result of promiscuous snooping on other packets. In particular, a node SHOULD prefer routes that it is presently using over those that it is not.

Any suitable data structure organization, consistent with this specification, MAY be used to implement the Route Cache in any node. For example, the following two types of organization are possible:

- In DSR, the route returned in each Route Reply that is received by the initiator of a Route Discovery (or that is learned from the header of overhead packets, as described in Section 6.1.4) represents a complete path (a sequence of links) leading to the destination node. By caching each of these paths separately, a "path cache" organization for the Route Cache can be formed. A path cache is very simple to implement and easily guarantees that all routes are loop-free, since each individual route from a Route Reply or Route Request or used in a packet is loop-free. To search for a route in a path cache data structure, the sending node can simply search its Route Cache for any path (or prefix of a path) that leads to the intended destination node.

  This type of organization for the Route Cache in DSR has been extensively studied through simulation [5, 9, 12, 19] and through implementation of DSR in a mobile outdoor testbed under significant workload [20, 21, 22].

- Alternatively, a "link cache" organization could be used for the Route Cache, in which each individual link (hop) in the routes returned in Route Reply packets (or otherwise learned from the header of overhead packets) is added to a unified graph data structure of this node's current view of the network topology. To search for a route in link cache, the sending node must use a more complex graph search algorithm, such as the well-known Dijkstra's shortest-path algorithm, to find the current best path through the graph to the destination node. Such an algorithm is more difficult to implement and may require significantly more CPU time to execute.

  However, a link cache organization is more powerful than a path cache organization, in its ability to effectively utilize all of the potential information that a node might learn about the state of the network. In particular, links learned from different Route Discoveries or from the header of any overheard packets can be merged together to form new routes in the network, but this is not possible in a path cache due to the separation of each individual path in the cache.

  This type of organization for the Route Cache in DSR, including the effect of a range of implementation choices, has been studied through detailed simulation [9].

The choice of data structure organization to use for the Route Cache in any DSR implementation is a local matter for each node and affects only performance; any reasonable choice of organization for the Route Cache does not affect either correctness or interoperability.

Each entry in the Route Cache SHOULD have a timeout associated with it, to allow that entry to be deleted if not used within some time. The particular choice of algorithm and data structure used to implement the Route Cache SHOULD be considered in choosing the timeout for entries in the Route Cache. The configuration variable RouteCacheTimeout defined in Section 9 specifies the timeout to be applied to entries in the Route Cache, although it is also possible to instead use an adaptive policy in choosing timeout values rather than using a single timeout setting for all entries; for example, the Link-MaxLife cache design (below) uses an adaptive timeout algorithm and does not use the RouteCacheTimeout configuration variable.

As guidance to implementors, Appendix A describes a type of link cache known as "Link-MaxLife" that has been shown to outperform other types of link caches and path caches studied in detailed simulation [9]. Link-MaxLife is an adaptive link cache in which each link in the cache has a timeout that is determined dynamically by the caching node according to its observed past behavior of the two nodes at the ends of the link; in addition, when selecting a route for a packet being sent to some destination, among cached routes of equal length (number of hops) to that destination, Link-MaxLife selects the route with the longest expected lifetime (highest minimum timeout of any link in the route). Use of the Link-MaxLife design for the Route Cache is recommended in implementations of DSR.

4.2. Send Buffer

The Send Buffer of a node implementing DSR is a queue of packets that cannot be sent by that node because it does not yet have a source route to each such packet's destination. Each packet in the Send Buffer is logically associated with the time that it was placed into the Buffer, and SHOULD be removed from the Send Buffer and silently discarded after a period of SendBufferTimeout after initially being placed in the Buffer. If necessary, a FIFO strategy SHOULD be used to evict packets before they timeout to prevent the buffer from overflowing.

Subject to the rate limiting defined in Section 6.2, a Route Discovery SHOULD be initiated as often as possible for the destination address of any packets residing in the Send Buffer.

4.3. Route Request Table

The Route Request Table of a node implementing DSR records information about Route Requests that have been recently originated or forwarded by this node. The table is indexed by IP address.

The Route Request Table on a node records the following information about nodes to which this node has initiated a Route Request:

- The Time-to-Live (TTL) field used in the IP header of the Route Request for the last Route Discovery initiated by this node for that target node. This value allows the node to implement a variety of algorithms for controlling the spread of its Route Request on each Route Discovery initiated for a target. As examples, two possible algorithms for this use of the TTL field are described in Section 3.3.4.

- The time that this node last originated a Route Request for that target node.

- The number of consecutive Route Discoveries initiated for this target since receiving a valid Route Reply giving a route to that target node.

- The remaining amount of time before which this node MAY next attempt at a Route Discovery for that target node. When the node initiates a new Route Discovery for this target node, this field in the Route Request Table entry for that target node is initialized to the timeout for that Route Discovery, after which the node MAY initiate a new Discovery for that target. Until a valid Route Reply is received for this target node address, a node MUST implement a back-off algorithm in determining this timeout value for each successive Route Discovery initiated for this target using the same Time-to-Live (TTL) value in the IP header of the Route Request packet. The timeout between such consecutive Route Discovery initiations SHOULD increase by doubling the timeout value on each new initiation.

In addition, the Route Request Table on a node also records the following information about initiator nodes from which this node has received a Route Request:

- A FIFO cache of size RequestTableIds entries containing the Identification value and target address from the most recent Route Requests received by this node from that initiator node.

Nodes SHOULD use an LRU policy to manage the entries in their Route Request Table.

The number of Identification values to retain in each Route Request Table entry, RequestTableIds, MUST NOT be unlimited, since, in the worst case, when a node crashes and reboots, the first RequestTableIds Route Discoveries it initiates after rebooting could appear to be duplicates to the other nodes in the network. In addition, a node SHOULD base its initial Identification value, used for Route Discoveries after rebooting, on a battery backed-up clock or other persistent memory device, in order to help avoid any possible such delay in successfully discovering new routes after rebooting; if no such source of initial Identification value is available, a node after rebooting SHOULD base its initial Identification value on a random number.

4.4. Gratuitous Route Reply Table

The Gratuitous Route Reply Table of a node implementing DSR records information about "gratuitous" Route Replies sent by this node as part of automatic route shortening. As described in Section 3.4.3, a node returns a gratuitous Route Reply when it overhears a packet transmitted by some node, for which the node overhearing the packet was not the intended next-hop node but was named later in the unexpended hops of the source route in that packet; the node overhearing the packet returns a gratuitous Route Reply to the original sender of the packet, listing the shorter route (not including the hops of the source route "skipped over" by this packet). A node uses its Gratuitous Route Reply Table to limit the rate at which it originates gratuitous Route Replies to the same original sender for the same node from which it overheard a packet to trigger the gratuitous Route Reply.

Each entry in the Gratuitous Route Reply Table of a node contains the following fields:

- The address of the node to which this node originated a gratuitous Route Reply.

- The address of the node from which this node overheard the packet triggering that gratuitous Route Reply.

- The remaining time before which this entry in the Gratuitous Route Reply Table expires and SHOULD be deleted by the node. When a node creates a new entry in its Gratuitous Route Reply Table, the timeout value for that entry should be initialized to the value GratReplyHoldoff.

When a node overhears a packet that would trigger a gratuitous Route Reply, if a corresponding entry already exists in the node's Gratuitous Route Reply Table, then the node SHOULD NOT send a gratuitous Route Reply for that packet. Otherwise (no corresponding entry already exists), the node SHOULD create a new entry in its Gratuitous Route Reply Table to record that gratuitous Route Reply, with a timeout value of GratReplyHoldoff.

4.5. Network Interface Queue and Maintenance Buffer

Depending on factors such as the structure and organization of the operating system, protocol stack implementation, network interface device driver, and network interface hardware, a packet being transmitted could be queued in a variety of ways. For example, outgoing packets from the network protocol stack might be queued at the operating system or link layer, before transmission by the network interface. The network interface might also provide a retransmission mechanism for packets, such as occurs in IEEE 802.11 [11]; the DSR protocol, as part of Route Maintenance, requires limited buffering of packets already transmitted for which the reachability of the next-hop destination has not yet been determined. The operation of DSR is defined here in terms of two conceptual data structures that together incorporate this queueing behavior.

The Network Interface Queue of a node implementing DSR is an output queue of packets from the network protocol stack waiting to be transmitted by the network interface; for example, in the 4.4BSD Unix network protocol stack implementation, this queue for a network interface is represented as a "struct ifqueue" [33]. This queue is used to hold packets while the network interface is in the process of transmitting another packet.

The Maintenance Buffer of a node implementing DSR is a queue of packets sent by this node that are awaiting next-hop reachability confirmation as part of Route Maintenance. For each packet in the Maintenance Buffer, a node maintains a count of the number of retransmissions and the time of the last retransmission. The Maintenance Buffer MAY be of limited size; when adding a new packet to the Maintenance Buffer, if the buffer size is insufficient to hold the new packet, the new packet SHOULD be silently discarded. If, after MaxMaintRexmt attempts to confirm next-hop reachability of some node, no confirmation is received, all packets in this node's Maintenance Buffer with this next-hop destination SHOULD be removed from the Maintenance Buffer; in this case, the node also SHOULD originate a Route Error for this packet to each original source of a packet removed in this way (Section 6.3) and SHOULD salvage each packet removed in this way (Section 6.3.6) if it has another route to that packet's IP Destination Address in its Route Cache. The definition of MaxMaintRexmt conceptually includes any retransmissions that might be attempted for a packet at the link layer or within the network interface hardware. The timeout value to use for each transmission attempt for an acknowledgment request depends on the type of acknowledgment mechanism used for Route Maintenance for that attempt, as described in Section 6.3.

4.6. Blacklist

When a node using the DSR protocol is connected through an interface that requires physically bidirectional links for unicast transmission, it MUST keep a blacklist. A Blacklist is a table, indexed by neighbor address, that indicates that the link between this node and the specified neighbor may not be bidirectional. A node places another node's address in this list when it believes that broadcast packets from that other node reach this node, but that unicast transmission between the two nodes is not possible. For example, if a node forwarding a Route Reply discovers that the next hop is unreachable, it places that next hop in the node's blacklist.

Once a node discovers that it can communicate bidirectionally with one of the nodes listed in the blacklist, it SHOULD remove that node from the blacklist. For example, if A has B in its blacklist, but A hears B forward a Route Request with a hop list indicating that the broadcast from A to B was successful, A SHOULD remove B from its blacklist.

A node MUST associate a state with each node in the blacklist, specifying whether the unidirectionality is "questionable" or "probable." Each time the unreachability is positively determined, the node SHOULD set the state to "probable." After the unreachability has not been positively determined for some amount of time, the state should revert to "questionable." A node MAY expire nodes from its blacklist after a reasonable amount of time.

5. DSR Header Format

The Dynamic Source Routing protocol makes use of a special header carrying control information that can be included in any existing IP packet. This DSR header in a packet contains a small fixed-sized, 4-octet portion, followed by a sequence of zero or more DSR options carrying optional information. The end of the sequence of DSR options in the DSR header is implied by total length of the DSR header.

For IPv4, the DSR header MUST immediately follow the IP header in the packet. (If a Hop-by-Hop Options extension header, as defined in IPv6 [6], becomes defined for IPv4, the DSR header MUST immediately follow the Hop-by-Hop Options extension header, if one is present in the packet, and MUST otherwise immediately follow the IP header.)

To add a DSR header to a packet, the DSR header is inserted following the packet's IP header, before any following header such as a traditional (e.g., TCP or UDP) transport layer header. Specifically, the Protocol field in the IP header is used to indicate that a DSR header follows the IP header, and the Next Header field in the DSR header is used to indicate the type of protocol header (such as a transport layer header) following the DSR header.

If any headers follow the DSR header in a packet, the total length of the DSR header (and thus the total, combined length of all DSR options present) MUST be a multiple of 4 octets. This requirement preserves the alignment of these following headers in the packet.

5.1. Fixed Portion of DSR Header

The fixed portion of the DSR header is used to carry information that must be present in any DSR header. This fixed portion of the DSR header has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |   Reserved    |        Payload Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                            Options                            .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Header

8-bit selector. Identifies the type of header immediately following the DSR header. Uses the same values as the IPv4 Protocol field [29].

Reserved

MUST be sent as 0 and ignored on reception.

Payload Length

The length of the DSR header, excluding the 4-octet fixed portion. The value of the Payload Length field defines the total length of all options carried in the DSR header.

Options

Variable-length field; the length of the Options field is specified by the Payload Length field in this DSR header. Contains one or more pieces of optional information (DSR options), encoded in type-length-value (TLV) format (with the exception of the Pad1 option, described in Section 5.8).

The placement of DSR options following the fixed portion of the DSR header MAY be padded for alignment. However, due to the typically limited available wireless bandwidth in ad hoc networks, this padding is not required, and receiving nodes MUST NOT expect options within a DSR header to be aligned.

The following types of DSR options are defined in this document for use within a DSR header:

- Route Request option (Section 5.2)

- Route Reply option (Section 5.3)

- Route Error option (Section 5.4)

- Acknowledgment Request option (Section 5.5)

- Acknowledgment option (Section 5.6)

- DSR Source Route option (Section 5.7)

- Pad1 option (Section 5.8)

- PadN option (Section 5.9)

5.2. Route Request Option

The Route Request option in a DSR header is encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Option Type  |  Opt Data Len |         Identification        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Target Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[1]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[2]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              ...                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[n]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP fields:

Source Address

MUST be set to the address of the node originating this packet. Intermediate nodes that retransmit the packet to propagate the Route Request MUST NOT change this field.

Destination Address

MUST be set to the IP limited broadcast address (255.255.255.255).

Hop Limit (TTL)

MAY be varied from 1 to 255, for example to implement non-propagating Route Requests and Route Request expanding-ring searches (Section 3.3.4).

Route Request fields:

Option Type

2

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification

A unique value generated by the initiator (original sender) of the Route Request. Nodes initiating a Route Request generate a new Identification value for each Route Request, for example based on a sequence number counter of all Route Requests initiated by the node.

This value allows a receiving node to determine whether it has recently seen a copy of this Route Request: if this Identification value is found by this receiving node in its Route Request Table (in the cache of Identification values in the entry there for this initiating node), this receiving node MUST discard the Route Request. When propagating a Route Request, this field MUST be copied from the received copy of the Route Request being propagated.

Target Address

The address of the node that is the target of the Route Request.

Address[1..n]

Address[i] is the address of the i-th node recorded in the Route Request option. The address given in the Source Address field in the IP header is the address of the initiator of the Route Discovery and MUST NOT be listed in the Address[i] fields; the address given in Address[1] is thus the address of the first node on the path after the initiator. The number of addresses present in this field is indicated by the Opt Data Len field in the option (n = (Opt Data Len - 6) / 4). Each node propagating the Route Request adds its own address to this list, increasing the Opt Data Len value by 4 octets.

The Route Request option MUST NOT appear more than once within a DSR header.

5.3. Route Reply Option

The Route Reply option in a DSR header is encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                   |  Option Type  |  Opt Data Len |L|   Reserved  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[1]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[2]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              ...                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[n]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP fields:

Source Address

Set to the address of the node sending the Route Reply. In the case of a node sending a reply from its Route Cache (Section 3.3.2) or sending a gratuitous Route Reply (Section 3.4.3), this address can differ from the address that was the target of the Route Discovery.

Destination Address

MUST be set to the address of the source node of the route being returned. Copied from the Source Address field of the Route Request generating the Route Reply, or in the case of a gratuitous Route Reply, copied from the Source Address field of the data packet triggering the gratuitous Reply.

Route Reply fields:

Option Type

3

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Last Hop External (L)

Set to indicate that the last hop given by the Route Reply (the link from Address[n-1] to Address[n]) is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the Route Reply. Nodes caching this hop in their Route Cache MUST flag the cached hop with the External flag. Such hops MUST NOT be returned in a cached Route Reply generated from this Route Cache entry, and selection of routes from the Route Cache to route a packet being sent MUST prefer routes that contain no hops flagged as External.

Reserved

MUST be sent as 0 and ignored on reception.

Address[1..n]

The source route being returned by the Route Reply. The route indicates a sequence of hops, originating at the source node specified in the Destination Address field of the IP header of the packet carrying the Route Reply, through each of the Address[i] nodes in the order listed in the Route Reply, ending with the destination node indicated by Address[n]. The number of addresses present in the Address[1..n] field is indicated by the Opt Data Len field in the option (n = (Opt Data Len - 1) / 4).

A Route Reply option MAY appear one or more times within a DSR header.

5.4. Route Error Option

The Route Error option in a DSR header is encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Option Type  |  Opt Data Len |   Error Type  |Reservd|Salvage|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Error Source Address                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Error Destination Address                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                   Type-Specific Information                   .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

4

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

For the current definition of the Route Error option, this field MUST be set to 10, plus the size of any Type-Specific Information present in the Route Error. Further extensions to the Route Error option format may also be included after the Type-Specific Information portion of the Route Error option specified above. The presence of such extensions will be indicated by the Opt Data Len field. When the Opt Data Len is greater than that required for the fixed portion of the Route Error plus the necessary Type-Specific Information as indicated by the Option Type value in the option, the remaining octets are interpreted as extensions. Currently, no such further extensions have been defined.

Error Type

The type of error encountered. Currently, the following type value is defined:

1 = NODE_UNREACHABLE

Other values of the Error Type field are reserved for future use.

Reservd

Reserved. MUST be sent as 0 and ignored on reception.

Salvage

A 4-bit unsigned integer. Copied from the Salvage field in the DSR Source Route option of the packet triggering the Route Error.

The "total salvage count" of the Route Error option is derived from the value in the Salvage field of this Route Error option and all preceding Route Error options in the packet as follows: the total salvage count is the sum of, for each such Route Error option, one plus the value in the Salvage field of that Route Error option.

Error Source Address

The address of the node originating the Route Error (e.g., the node that attempted to forward a packet and discovered the link failure).

Error Destination Address

The address of the node to which the Route Error must be delivered For example, when the Error Type field is set to NODE_UNREACHABLE, this field will be set to the address of the node that generated the routing information claiming that the hop from the Error Source Address to Unreachable Node Address (specified in the Type-Specific Information) was a valid hop.

Type-Specific Information

Information specific to the Error Type of this Route Error message.

Currently, the Type-Specific Information field is defined only for Route Error messages of type NODE_UNREACHABLE. In this case, the Type-Specific Information field is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Unreachable Node Address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Unreachable Node Address

The address of the node that was found to be unreachable (the next-hop neighbor to which the node with address Error Source Address was attempting to transmit the packet).

A Route Error option MAY appear one or more times within a DSR header.

5.5. Acknowledgment Request Option

The Acknowledgment Request option in a DSR header is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |         Identification        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

5

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification

The Identification field is set to a unique value and is copied into the Identification field of the Acknowledgment option when returned by the node receiving the packet over this hop.

An Acknowledgment Request option MUST NOT appear more than once within a DSR header.

5.6. Acknowledgment Option

The Acknowledgment option in a DSR header is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Option Type  |  Opt Data Len |         Identification        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       ACK Source Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ACK Destination Address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

6

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification

Copied from the Identification field of the Acknowledgment Request option of the packet being acknowledged.

ACK Source Address

The address of the node originating the acknowledgment.

ACK Destination Address

The address of the node to which the acknowledgment is to be delivered.

An Acknowledgment option MAY appear one or more times within a DSR header.

5.7. DSR Source Route Option

The DSR Source Route option in a DSR header is encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Option Type  |  Opt Data Len |F|L|Reservd|Salvage| Segs Left |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[1]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[2]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              ...                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Address[n]                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Type

7

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields. For the format of the DSR Source Route option defined here, this field MUST be set to the value (n * 4) + 2, where n is the number of addresses present in the Address[i] fields.

First Hop External (F)

Set to indicate that the first hop indicated by the DSR Source Route option is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the DSR Source Route option. Nodes caching this hop in their Route Cache MUST flag the cached hop with the External flag. Such hops MUST NOT be returned in a Route Reply generated from this Route Cache entry, and selection of routes from the Route Cache to route a packet being sent MUST prefer routes that contain no hops flagged as External.

Last Hop External (L)

Set to indicate that the last hop indicated by the DSR Source Route option is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the DSR Source Route option.

Nodes caching this hop in their Route Cache MUST flag the cached hop with the External flag. Such hops MUST NOT be returned in a Route Reply generated from this Route Cache entry, and selection of routes from the Route Cache to route a packet being sent MUST prefer routes that contain no hops flagged as External.

Reserved

MUST be sent as 0 and ignored on reception.

Salvage

A 4-bit unsigned integer. Count of number of times that this packet has been salvaged as a part of DSR routing (Section 3.4.1).

Segments Left (Segs Left)

Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

Address[1..n]

The sequence of addresses of the source route. In routing and forwarding the packet, the source route is processed as described in Sections 6.1.3 and 6.1.5. The number of addresses present in the Address[1..n] field is indicated by the Opt Data Len field in the option (n = (Opt Data Len - 2) / 4).

When forwarding a packet along a DSR source route using a DSR Source Route option in the packet's DSR header, the Destination Address field in the packet's IP header is always set to the address of the packet's ultimate destination. A node receiving a packet containing a DSR header with a DSR Source Route option MUST examine the indicated source route to determine if it is the intended next-hop node for the packet and determine how to forward the packet, as defined in Sections 6.1.4 and 6.1.5.

5.8. Pad1 Option

The Pad1 option in a DSR header is encoded as follows:

```
+-+-+-+-+-+-+-+-+
|  Option Type  |
+-+-+-+-+-+-+-+-+
```

Option Type

0

A Pad1 option MAY be included in the Options field of a DSR header in order to align subsequent DSR options, but such alignment is not required and MUST NOT be expected by a node receiving a packet containing a DSR header.

If any headers follow the DSR header in a packet, the total length of a DSR header, indicated by the Payload Length field in the DSR header MUST be a multiple of 4 octets. In this case, when building a DSR header in a packet, sufficient Pad1 or PadN options MUST be included in the Options field of the DSR header to make the total length a multiple of 4 octets.

If more than one consecutive octet of padding is being inserted in the Options field of a DSR header, the PadN option, described next, SHOULD be used, rather than multiple Pad1 options.

Note that the format of the Pad1 option is a special case; it does not have an Opt Data Len or Option Data field.

5.9. PadN Option

The PadN option in a DSR header is encoded as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
|  Option Type  |  Opt Data Len |   Option Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

Option Type

1

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Option Data

A number of zero-valued octets equal to the Opt Data Len.

A PadN option MAY be included in the Options field of a DSR header in order to align subsequent DSR options, but such alignment is not required and MUST NOT be expected by a node receiving a packet containing a DSR header.

If any headers follow the DSR header in a packet, the total length of a DSR header, indicated by the Payload Length field in the DSR header MUST be a multiple of 4 octets. In this case, when building a DSR header in a packet, sufficient Pad1 or PadN options MUST be included in the Options field of the DSR header to make the total length a multiple of 4 octets.

6. Detailed Operation

6.1. General Packet Processing

6.1.1. Originating a Packet

When originating any packet, a node using DSR routing MUST perform the following sequence of steps:

- Search the node's Route Cache for a route to the address given in the IP Destination Address field in the packet's header.

- If no such route is found in the Route Cache, then perform Route Discovery for the Destination Address, as described in Section 6.2. Initiating a Route Discovery for this target node address results in the node adding a Route Request option in a DSR header in this existing packet, or saving this existing packet to its Send Buffer and initiating the Route Discovery by sending a separate packet containing such a Route Request option. If the node chooses to initiate the Route Discovery by adding the Route Request option to this existing packet, it will replace the IP Destination Address field with the IP "limited broadcast" address (255.255.255.255) [3], copying the original IP Destination Address to the Target Address field of the new Route Request option added to the packet, as described in Section 6.2.1.

- If the packet now does not contain a Route Request option, then this node must have a route to the Destination Address of the packet; if the node has more than one route to this Destination Address, the node selects one to use for this packet. If the length of this route is greater than 1 hop, or if the node determines to request a DSR network-layer acknowledgment from the first-hop node in that route, then insert a DSR header into the packet, as described in Section 6.1.2, and insert a DSR Source Route option, as described in Section 6.1.3. The source route in the packet is initialized from the selected route to the Destination Address of the packet.

- Transmit the packet to the first-hop node address given in selected source route, using Route Maintenance to determine the reachability of the next hop, as described in Section 6.3.

6.1.2. Adding a DSR Header to a Packet

A node originating a packet adds a DSR header to the packet, if necessary, to carry information needed by the routing protocol. A packet MUST NOT contain more than one DSR header. A DSR header is added to a packet by performing the following sequence of steps (these steps assume that the packet contains no other headers that MUST be located in the packet before the DSR header):

- Insert a DSR header after the IP header but before any other header that may be present.

- Set the Next Header field of the DSR header to the Protocol number field of the packet's IP header.

- Set the Protocol field of the packet's IP header to the Protocol number assigned for a DSR header (TBA???).

6.1.3. Adding a DSR Source Route Option to a Packet

A node originating a packet adds a DSR Source Route option to the packet, if necessary, in order to carry the source route from this originating node to the final destination address of the packet. Specifically, the node adding the DSR Source Route option constructs the DSR Source Route option and modifies the IP packet according to the following sequence of steps:

- The node creates a DSR Source Route option, as described in Section 5.7, and appends it to the DSR header in the packet. (A DSR header is added, as described in Section 6.1.2, if not already present.)

- The number of Address[i] fields to include in the DSR Source Route option (n) is the number of intermediate nodes in the source route for the packet (i.e., excluding address of the originating node and the final destination address of the packet). The Segments Left field in the DSR Source Route option is initialized equal to n.

- The addresses within the source route for the packet are copied into sequential Address[i] fields in the DSR Source Route option, for i = 1, 2, ..., n.

- The First Hop External (F) bit in the DSR Source Route option is copied from the External bit flagging the first hop in the source route for the packet, as indicated in the Route Cache.

- The Last Hop External (L) bit in the DSR Source Route option is copied from the External bit flagging the last hop in the source route for the packet, as indicated in the Route Cache.

- The Salvage field in the DSR Source Route option is initialized to 0.

6.1.4. Processing a Received Packet

When a node receives any packet (whether for forwarding, overheard, or as the final destination of the packet), if that packet contains a DSR header, then that node MUST process any options contained in that DSR header, in the order contained there. Specifically:

- If the DSR header contains a Route Request option, the node SHOULD extract the source route from the Route Request and add this routing information to its Route Cache, subject to the conditions identified in Section 3.3.1. The routing information from the Route Request is the sequence of hop addresses

  ```
  initiator, Address[1], Address[2], ..., Address[n]
  ```

  where initiator is the value of the Source Address field in the IP header of the packet carrying the Route Request (the address of the initiator of the Route Discovery), and each Address[i] is a node through which this Route Request has passed, in turn, during this Route Discovery. The value n here is the number of addresses recorded in the Route Request option, or (Opt Data Len - 6) / 4.

  After possibly updating the node's Route Cache in response to the routing information in the Route Request option, the node MUST then process the Route Request option as described in Section 6.2.2.

- If the DSR header contains a Route Reply option, the node SHOULD extract the source route from the Route Reply and add this routing information to its Route Cache, subject to the conditions identified in Section 3.3.1. The source route from the Route Reply is the sequence of hop addresses

  ```
  initiator, Address[1], Address[2], ..., Address[n]
  ```

  where initiator is the value of the Destination Address field in the IP header of the packet carrying the Route Reply (the address of the initiator of the Route Discovery), and each Address[i] is a node through which the source route passes, in turn, on the route to the target of the Route Discovery. Address[n] is the address of the target. If the Last Hop External (L) bit is set in the Route Reply, the node MUST flag the last hop from the Route Reply (the link from Address[n-1] to Address[n]) in its Route Cache as External. The value n here is the number of addresses in the source route being returned in the Route Reply option, or (Opt Data Len - 1) / 4.

  After possibly updating the node's Route Cache in response to the routing information in the Route Reply option, then if the packet's IP Destination Address matches one of this node's IP addresses, the node MUST then process the Route Reply option as described in Section 6.2.5.

- If the DSR header contains a Route Error option, the node MUST process the Route Error option as described in Section 6.3.5.

- If the DSR header contains an Acknowledgment Request option, the node MUST process the Acknowledgment Request option as described in Section 6.3.3.

- If the DSR header contains an Acknowledgment option, then subject to the conditions identified in Section 3.3.1, the node SHOULD add to its Route Cache the single link from the node identified by the ACK Source Address field to the node identified by the ACK Destination Address field.

  After possibly updating the node's Route Cache in response to the routing information in the Acknowledgment option, the node MUST then process the Acknowledgment option as described in Section 6.3.3.

- If the DSR header contains a DSR Source Route option, the node SHOULD extract the source route from the DSR Source Route and add this routing information to its Route Cache, subject to the conditions identified in Section 3.3.1. If the value of the Salvage field in the DSR Source Route option is zero, then the routing information from the DSR Source Route is the sequence of hop addresses

  ```
  source, Address[1], Address[2], ..., Address[n], destination
  ```

  and otherwise (Salvage is nonzero), the routing information from the DSR Source Route is the sequence of hop addresses

  ```
  Address[1], Address[2], ..., Address[n], destination
  ```

  where source is the value of the Source Address field in the IP header of the packet carrying the DSR Source Route option (the original sender of the packet), each Address[i] is the value in the Address[i] field in the DSR Source Route, and destination is the value of the Destination Address field in the packet's IP header (the last-hop address of the source route). The value n here is the number of addresses in source route in the DSR Source Route option, or (Opt Data Len - 2) / 4.

  After possibly updating the node's Route Cache in response to the routing information in the DSR Source Route option, the node MUST then process the DSR Source Route option as described in

Section 6.1.5.

- Any Pad1 or PadN options in the DSR header are ignored.

Finally, if the Destination Address in the packet's IP header matches one of this receiving node's own IP address(es), remove the DSR header and all the included DSR options in the header, and pass the rest of the packet to the network layer.

6.1.5. Processing a Received DSR Source Route Option

When a node receives a packet containing a DSR Source Route option (whether for forwarding, overheard, or as the final destination of the packet), that node SHOULD examine the packet to determine if the receipt of that packet indicates an opportunity for automatic route shortening, as described in Section 3.4.3. Specifically, if this node is not the intended next-hop destination for the packet but is named in the later unexpended portion of the source route in the packet's DSR Source Route option, then this packet indicates an opportunity for automatic route shortening: the intermediate nodes after the node from which this node overheard the packet and before this node itself, are no longer necessary in the source route. In this case, this node SHOULD perform the following sequence of steps as part of automatic route shortening:

- The node searches its Gratuitous Route Reply Table for an entry describing a gratuitous Route Reply earlier sent by this node, for which the original sender of the packet triggering the gratuitous Route Reply and the transmitting node from which this node overheard that packet in order to trigger the gratuitous Route Reply, both match the respective node addresses for this new received packet. If such an entry is found in the node's Gratuitous Route Reply Table, the node SHOULD NOT perform automatic route shortening in response to this receipt of this packet.

- Otherwise, the node creates an entry for this overheard packet in its Gratuitous Route Reply Table. The timeout value for this new entry SHOULD be initialized to the value GratReplyHoldoff. After this timeout has expired, the node SHOULD delete this entry from its Gratuitous Route Reply Table.

- After creating the new Gratuitous Route Reply Table entry above, the node originates a gratuitous Route Reply to the IP Source Address of this overheard packet, as described in Section 3.4.3.

  If the MAC protocol in use in the network is not capable of transmitting unicast packets over unidirectional links, as discussed in Section 3.3.1, then in originating this Route Reply, the node MUST use a source route for routing the Route Reply packet that is obtained by reversing the sequence of hops over which the packet triggering the gratuitous Route Reply was routed in reaching and being overheard by this node; this reversing of the route uses the gratuitous Route Reply to test this sequence of hops for bidirectionality, preventing the gratuitous Route Reply from being received by the initiator of the Route Discovery unless each of the hops over which the gratuitous Route Reply is returned is bidirectional.

- Discard the overheard packet, since the packet has been received before its normal traversal of the packet's source route would have caused it to reach this receiving node. Another copy of the packet will normally arrive at this node as indicated in the packet's source route; discarding this initial copy of the packet, which triggered the gratuitous Route Reply, will prevent the duplication of this packet that would otherwise occur.

If the packet is not discarded as part of automatic route shortening above, then the node MUST process the option according to the following sequence of steps:

- If the value of the Segments Left field in the DSR Source Route option equals 0, then remove the DSR Source Route option from the DSR header.

- Else, let n equal (Opt Data Len - 2) / 4. This is the number of addresses in the DSR Source Route option.

- If the value of the Segments Left field is greater than n, then send an ICMP Parameter Problem, Code 0, message [26] to the IP Source Address, pointing to the Segments Left field, and discard the packet. Do not process the DSR Source Route option further.

- Else, decrement the value of the Segments Left field by 1. Let i equal n minus Segments Left. This is the index of the next address to be visited in the Address vector.

- If Address[i] or the IP Destination Address is a multicast address, then discard the packet. Do not process the DSR Source Route option further.

- If the MTU of the link over which this node would transmit the packet to forward it to the node Address[i] is less than the size of the packet, the node MUST either discard the packet and send an ICMP Packet Too Big message to the packet's Source Address [26] or fragment it as specified in Section 8.

- Forward the packet to the IP address specified in the Address[i] field of the IP header, following normal IP forwarding procedures, including checking and decrementing the Time-to-Live

(TTL) field in the packet's IP header [27, 3]. In this forwarding of the packet, the next-hop node (identified by Address[i]) MUST be treated as a direct neighbor node: the transmission to that next node MUST be done in a single IP forwarding hop, without Route Discovery and without searching the Route Cache.

- In forwarding the packet, perform Route Maintenance for the next hop of the packet, by verifying that the next-hop node is reachable, as described in Section 6.3.

Multicast addresses MUST NOT appear in a DSR Source Route option or in the IP Destination Address field of a packet carrying a DSR Source Route option in a DSR header.

6.2. Route Discovery Processing

Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D. The node initiating a Route Discovery is known as the "initiator" of the Route Discovery, and the destination node for which the Route Discovery is initiated is known as the "target" of the Route Discovery.

Route Discovery operates entirely on demand, with a node initiating Route Discovery based on its own origination of new packets for some destination address to which it does not currently know a route. Route Discovery does not depend on any periodic or background exchange of routing information or neighbor node detection at any layer in the network protocol stack at any node.

The Route Discovery procedure utilizes two types of messages, a Route Request (Section 5.2) and a Route Reply (Section 5.3), to actively search the ad hoc network for a route to the desired destination. These DSR messages MAY be carried in any type of IP packet, through use of the DSR header as described in Section 5.

Except as discussed in Section 6.3.5, a Route Discovery for a destination address SHOULD NOT be initiated unless the initiating node has a packet in its Send Buffer requiring delivery to that destination. A Route Discovery for a given target node MUST NOT be initiated unless permitted by the rate-limiting information contained in the Route Request Table. After each Route Discovery attempt, the interval between successive Route Discoveries for this target SHOULD be doubled, up to a maximum of MaxRequestPeriod, until a valid Route Reply is received for this target.

6.2.1. Originating a Route Request

A node initiating a Route Discovery for some target creates and initializes a Route Request option in a DSR header in some IP packet. This MAY be a separate IP packet, used only to carry this Route Request option, or the node MAY include the Route Request option in some existing packet that it needs to send to the target node (e.g., the IP packet originated by this node, that caused the node to attempt Route Discovery for the destination address of the packet). The Route Request option MUST be included in a DSR header in the packet. To initialize the Route Request option, the node performs the following sequence of steps:

- The Option Type in the option MUST be set to the value 2.

- The Opt Data Len field in the option MUST be set to the value 6. The total size of the Route Request option when initiated is 8 octets; the Opt Data Len field excludes the size of the Option Type and Opt Data Len fields themselves.

- The Identification field in the option MUST be set to a new value, different from that used for other Route Requests recently initiated by this node for this same target address. For example, each node MAY maintain a single counter value for generating a new Identification value for each Route Request it initiates.

- The Target Address field in the option MUST be set to the IP address that is the target of this Route Discovery.

The Source Address in the IP header of this packet MUST be the node's own IP address. The Destination Address in the IP header of this packet MUST be the IP "limited broadcast" address (255.255.255.255).

A node MUST maintain in its Route Request Table, information about Route Requests that it initiates. When initiating a new Route Request, the node MUST use the information recorded in the Route Request Table entry for the target of that Route Request, and it MUST update that information in the table entry for use in the next Route Request initiated for this target. In particular:

- The Route Request Table entry for a target node records the Time-to-Live (TTL) field used in the IP header of the Route Request for the last Route Discovery initiated by this node for that target node. This value allows the node to implement a variety of algorithms for controlling the spread of its Route Request on each Route Discovery initiated for a target. As examples, two possible algorithms for this use of the TTL field are described in Section 3.3.4.

- The Route Request Table entry for a target node records the number of consecutive Route Requests initiated for this target since receiving a valid Route Reply giving a route to that target node, and the remaining amount of time before which this node MAY next attempt at a Route Discovery for that target node.

  A node MUST use these values to implement a back-off algorithm to limit the rate at which this node initiates new Route Discoveries for the same target address. In particular, until a valid Route Reply is received for this target node address, the timeout between consecutive Route Discovery initiations for this target node with the same hop limit SHOULD increase by doubling the timeout value on each new initiation.

The behavior of a node processing a packet containing DSR header with both a DSR Source Route option and a Route Request option is unspecified. Packets SHOULD NOT contain both a DSR Source Route option and a Route Request option.

Packets containing a Route Request option SHOULD NOT include an Acknowledgment Request option, SHOULD NOT expect link-layer acknowledgment or passive acknowledgment, and SHOULD NOT be retransmitted. The retransmission of packets containing a Route Request option is controlled solely by the logic described in this section.

6.2.2. Processing a Received Route Request Option

When a node receives a packet containing a Route Request option, that node MUST process the option according to the following sequence of steps:

- If the Target Address field in the Route Request matches this node's own IP address, then the node SHOULD return a Route Reply to the initiator of this Route Request (the Source Address in the IP header of the packet), as described in Section 6.2.4. The source route for this Reply is the sequence of hop addresses

  initiator, Address[1], Address[2], ..., Address[n], target

  where initiator is the address of the initiator of this Route Request, each Address[i] is an address from the Route Request, and target is the target of the Route Request (the Target Address field in the Route Request). The value n here is the number of addresses recorded in the Route Request, or (Opt Data Len - 6) / 4.

  The node then MUST replace the Destination Address field in the Route Request packet's IP header with the value in the Target Address field in the Route Request option, and continue processing the rest of the Route Request packet normally. The node MUST NOT process the Route Request option further and MUST NOT retransmit the Route Request to propagate it to other nodes as part of the Route Discovery.

- Else, the node MUST examine the route recorded in the Route Request option (the IP Source Address field and the sequence of Address[i] fields) to determine if this node's own IP address already appears in this list of addresses. If so, the node MUST discard the entire packet carrying the Route Request option.

- Else, if the Route Request through a network interface that requires physically bidirectional links for unicast transmission, the node MUST check if the Request was last forwarded by a node on its blacklist. If such an entry is found, and the state of the unidirectional link is "probable," then the Request MUST be silently discarded.

- Else, if the Route Request through a network interface that requires physically bidirectional links for unicast transmission, the node MUST check if the Request was last forwarded by a node on its blacklist. If such an entry is found, and the state of the unidirectional link is "questionable," then the node MUST create and unicast a Route Request packet to that previous node, setting the IP Time-To-Live (TTL) to 1 to prevent the Request from being propagated. If the node receives a Route Reply in response to the new Request, it MUST remove the blacklist entry for that node, and SHOULD continue processing. If the node does not receive a Reply within some reasonable amount of time, MUST silently discard the Route Request packet.

- Else, the node MUST search its Route Request Table for an entry for the initiator of this Route Request (the IP Source Address field). If such an entry is found in the table, the node MUST search the cache of Identification values of recently received Route Requests in that table entry, to determine if an entry is present in the cache matching the Identification value and target node address in this Route Request. If such an (Identification, target address) entry is found in this cache in this entry in the Route Request Table, then the node MUST discard the entire packet carrying the Route Request option.

- Else, this node SHOULD further process the Route Request according to the following sequence of steps:

  o Add an entry for this Route Request in its cache of (Identification, target address) values of recently received Route Requests.

  o Conceptually create a copy of this entire packet and perform the following steps on the copy of the packet.

  o Append this node's own IP address to the list of Address[i] values in the Route Request, and increase the value of the Opt Data Len field in the Route Request by 4 (the size of an IP address).

  o This node SHOULD search its own Route Cache for a route (from itself, as if it were the source of a packet) to the target of this Route Request. If such a route is found in its Route Cache, then this node SHOULD follow the procedure outlined in Section 6.2.3 to return a "cached Route Reply" to the initiator of this Route Request, if permitted by the restrictions specified there.

- o If the node does not return a cached Route Reply, then this node SHOULD link-layer re-broadcast this copy of the packet, with a short jitter delay before the broadcast is sent. The jitter period SHOULD be chosen as a random period, uniformly distributed between 0 and BroadcastJitter.

6.2.3. Generating a Route Reply using the Route Cache

As described in Section 3.3.2, it is possible for a node processing a received Route Request to avoid propagating the Route Request further toward the target of the Request, if this node has in its Route Cache a route from itself to this target. Such a Route Reply generated by a node from its own cached route to the target of a Route Request is called a "cached Route Reply", and this mechanism can greatly reduce the overall overhead of Route Discovery on the network by reducing the flood of Route Requests. The general processing of a received Route Request is described in Section 6.2.2; this section specifies the additional requirements that MUST be met before a cached Route Reply may be generated and returned and specifies the procedure for returning such a cached Route Reply.

While processing a received Route Request, for a node to possibly return a cached Route Reply, it MUST have in its Route Cache a route from itself to the target of this Route Request. However, before generating a cached Route Reply for this Route Request, the node MUST verify that there are no duplicate addresses listed in the route accumulated in the Route Request together with the route from this node's Route Cache. Specifically, there MUST be no duplicates among the following addresses:

- The IP Source Address of the packet containing the Route Request,

- The Address[i] fields in the Route Request, and

- The nodes listed in the route obtained from this node's Route Cache, excluding the address of this node itself (this node itself is the common point between the route accumulated in the Route Request and the route obtained from the Route Cache).

If any duplicates exist among these addresses, then the node MUST NOT send a cached Route Reply. The node SHOULD continue to process the Route Request as described in Section 6.2.2.

If the Route Request and the route from the Route Cache meet the restriction above, then the node SHOULD construct and return a cached

Route Reply as follows:

- The source route for this reply is the sequence of hop addresses

  initiator, Address[1], Address[2], ..., Address[n], c-route

  where initiator is the address of the initiator of this Route Request, each Address[i] is an address from the Route Request, and c-route is the sequence of hop addresses in the source route to this target node, obtained from the node's Route Cache. In appending this cached route to the source route for the reply, the address of this node itself MUST be excluded, since it is already listed as Address[n].

- Send a Route Reply to the initiator of the Route Request, using the procedure defined in Section 6.2.4. The initiator of the Route Request is indicated in the Source Address field in the packet's IP header.

If the node returns a cached Route Reply as described above, then the node MUST NOT propagate the Route Request further (i.e., the node MUST NOT rebroadcast the Route Request). In this case, instead, if the packet contains no other DSR options and contains no payload after the DSR header (e.g., the Route Request is not piggybacked on a TCP or UDP packet), then the node SHOULD simply discard the packet. Otherwise (if the packet contains other DSR options or contains any payload after the DSR header), the node SHOULD forward the packet along the cached route to the target of the Route Request. Specifically, if the node does so, it MUST use the following steps:

- Copy the Target Address from the Route Request option in the DSR header to the Destination Address field in the packet's IP header.

- Remove the Route Request option from the DSR header in the packet, and add a DSR Source Route option to the packet's DSR header.

- In the DSR Source Route option, set the Address[i] fields to represent the source route found in this node's Route Cache to the original target of the Route Discovery (the new IP Destination Address of the packet). Specifically, the node copies the hop addresses of the source route into sequential Address[i] fields in the DSR Source Route option, for i = 1, 2, ..., n. Address[1] here is the address of this node itself (the first address in the source route found from this node to the original target of the Route Discovery). The value n here is the number of hop addresses in this source route, excluding the destination of the packet (which is instead already represented in the Destination Address field in the packet's IP header).

- Initialize the Segments Left field in the DSR Source Route option to n as defined above.

- The First Hop External (F) bit in the DSR Source Route option is copied from the External bit flagging the first hop in the source route for the packet, as indicated in the Route Cache.

- The Last Hop External (L) bit in the DSR Source Route option is copied from the External bit flagging the last hop in the source route for the packet, as indicated in the Route Cache.

- The Salvage field in the DSR Source Route option MUST be initialized to some nonzero value; the particular nonzero value used SHOULD be MAX_SALVAGE_COUNT. By initializing this field to a nonzero value, nodes forwarding or overhearing this packet will not consider a link to exist between the IP Source Address of the packet and the Address[1] address in the DSR Source Route option (e.g., they will not attempt to add this to their Route Cache as a link). By choosing MAX_SALVAGE_COUNT as the nonzero value to which the node initializes this field, nodes furthermore will not attempt to salvage this packet.

- Transmit the packet to the next-hop node on the new source route in the packet, using the forwarding procedure described in Section 6.1.5.

6.2.4. Originating a Route Reply

A node originates a Route Reply in order to reply to a received and processed Route Request, according to the procedures described in Sections 6.2.2 and 6.2.3. The Route Reply is returned in a Route Reply option (Section 5.3). The Route Reply option MAY be returned to the initiator of the Route Request in a separate IP packet, used only to carry this Route Reply option, or it MAY be included in any other IP packet being sent to this address.

The Route Reply option MUST be included in a DSR header in the packet returned to the initiator. To initialize the Route Reply option, the node performs the following sequence of steps:

- The Option Type in the option MUST be set to the value 3.

- The Opt Data Len field in the option MUST be set to the value (n * 4) + 3, where n is the number of addresses in the source route being returned (excluding the Route Discovery initiator node's address).

- The Last Hop External (L) bit in the option MUST be initialized to 0.

- The Reserved field in the option MUST be initialized to 0.

- The Route Request Identifier MUST be initialized to the Identifier field of the Route Request that this reply is sent in response to.

- The sequence of hop addresses in the source route are copied into the Address[i] fields of the option. Address[1] MUST be set to the first-hop address of the route after the initiator of the Route Discovery, Address[n] MUST be set to the last-hop address of the source route (the address of the target node), and each other Address[i] MUST be set to the next address in sequence in the source route being returned.

The Destination Address field in the IP header of the packet carrying the Route Reply option MUST be set to the address of the initiator of the Route Discovery (i.e., for a Route Reply being returned in response to some Route Request, the IP Source Address of the Route Request).

After creating and initializing the Route Reply option and the IP packet containing it, send the Route Reply. In sending the Route Reply from this node (but not from nodes forwarding the Route Reply), this node SHOULD delay the Reply by a small jitter period chosen randomly between 0 and BroadcastJitter.

When returning any Route Reply in the case in which the MAC protocol in use in the network is not capable of transmitting unicast packets over unidirectional links, the source route used for routing the Route Reply packet MUST be obtained by reversing the sequence of hops in the Route Request packet (the source route that is then returned in the Route Reply). This restriction on returning a Route Reply enables the Route Reply to test this sequence of hops for bidirectionality, preventing the Route Reply from being received by the initiator of the Route Discovery unless each of the hops over which the Route Reply is returned (and thus each of the hops in the source route being returned in the Reply) is bidirectional.

If sending a Route Reply to the initiator of the Route Request requires performing a Route Discovery, the Route Reply Option MUST be piggybacked on the packet that contains the Route Request. This piggybacking prevents a loop wherein the target of the new Route Request (which was itself the initiator of the original Route Request) must do another Route Request in order to return its Route Reply.

If sending the Route Reply to the initiator of the Route Request does not require performing a Route Discovery, a node SHOULD send a unicast Route Reply in response to every Route Request it receives for which it is the target node.

6.2.5. Processing a Received Route Reply Option

Section 6.1.4 describes the general processing for a received packet, including the addition of routing information from options in the packet's DSR header to the receiving node's Route Cache.

If the received packet contains a Route Reply, no additional special processing of the Route Reply option is required beyond what is described there. As described in Section 4.1 anytime a node adds new information to its Route Cache (including the information added from this Route Reply option), the node SHOULD check each packet in its own Send Buffer (Section 4.2) to determine whether a route to that packet's IP Destination Address now exists in the node's Route Cache (including the information just added to the Cache). If so, the packet SHOULD then be sent using that route and removed from the Send Buffer. This general procedure handles all processing required for a received Route Reply option.

When a MAC protocol requires bidirectional links for unicast transmission, a unidirectional link may be discovered by the propagation of the Route Request. When the Route Reply is sent over the reverse path, a forwarding node may discover that the next-hop is unreachable. In this case, it MUST add the next-hop address to its blacklist.

6.3. Route Maintenance Processing

Route Maintenance is the mechanism by which a source node S is able to detect, while using a source route to some destination node D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates that a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke Route Discovery again to find a new route for subsequent packets to D. Route Maintenance for this route is used only when S is actually sending packets to D.

Specifically, when forwarding a packet, a node MUST attempt to confirm the reachability of the next-hop node, unless such confirmation had been received in the last MaintHoldoffTime. Individual implementations MAY choose to bypass such confirmation for some limited number of packets, as long as those packets all fall within MaintHoldoffTime within the last confirmation. If no confirmation is received after the retransmission of MaxMaintRexmt acknowledgment requests, after the initial transmission of the packet, and conceptually including all retransmissions provided by the MAC layer, the node determines that the link for this next-hop node of the source route is "broken". This confirmation from the next-hop node for Route Maintenance can be implemented using a link-layer acknowledgment (Section 6.3.1), using a "passive acknowledgment" (Section 6.3.2), or using a network-layer acknowledgment (Section 6.3.3); the particular strategy for retransmission timing depends on the type of acknowledgment mechanism used. When passive acknowledgments are being used, each retransmitted acknowledgment request SHOULD be explicit software acknowledgment requests. If no acknowledgment is received after MaxMaintRexmt retransmissions (if necessary), the node SHOULD originate a Route Error to the original sender of the packet, as described in Section 6.3.4.

In deciding whether or not to send a Route Error in response to attempting to forward a packet from some sender over a broken link, a node MUST limit the number of consecutive packets from a single sender that the node attempts to forward over this same broken link for which the node chooses not to return a Route Error; this requirement MAY be satisfied by returning a Route Error for each packet that the node attempts to forward over a broken link.

6.3.1. Using Link-Layer Acknowledgments

If the MAC protocol in use provides feedback as to the successful delivery of a data packet (such as is provided by the link-layer acknowledgment frame defined by IEEE 802.11 [11]), then the use of the DSR Acknowledgment Request and Acknowledgment options is not necessary. If such link-layer feedback is available, it SHOULD be used instead of any other acknowledgment mechanism for Route Maintenance, and the node SHOULD NOT use either passive acknowledgments or network-layer acknowledgments for Route Maintenance.

When using link-layer acknowledgments for Route Maintenance, the retransmission timing and the timing at which retransmission attempts are scheduled are generally controlled by the particular link layer implementation in use in the network. For example, in IEEE 802.11, the link-layer acknowledgment is returned after the data packet as a part of the basic access method of of the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol; the time at which the acknowledgment is expected to arrive and the time at which the next retransmission attempt (if necessary) will occur are controlled by the MAC protocol implementation.

When a node receives a link-layer acknowledgment for any packet in its Maintenance Buffer, that node SHOULD remove that packet, as well as any other packets in its Maintenance Buffer with the same next-hop destination, from its Maintenance Buffer.

6.3.2. Using Passive Acknowledgments

When link-layer acknowledgments are not available, but passive acknowledgments [16] are available, passive acknowledgments SHOULD be used for Route Maintenance when originating or forwarding a packet along any hop other than the last hop (the hop leading to the IP Destination Address node of the packet). In particular, passive acknowledgments SHOULD be used for Route Maintenance in such cases if the node can place its network interface into "promiscuous" receive mode, and network links used for data packets generally operate bidirectionally.

A node MUST NOT attempt to use passive acknowledgments for Route Maintenance for a packet originated or forwarded over its last hop (the hop leading to the IP Destination Address node of the packet), since the receiving node will not be forwarding the packet and thus no passive acknowledgment will be available to be heard by this node. Beyond this restriction, a node MAY utilize a variety of strategies in using passive acknowledgments for Route Maintenance of a packet that it originates or forwards. For example, the following two strategies are possible:

- Each time a node receives a packet to be forwarded to a node other than the final destination (the IP Destination Address of the packet), that node sends the original transmission of that packet without requesting a network-layer acknowledgment for it. If no passive acknowledgment is received within PassiveAckTimeout after this transmission, the node retransmits the packet, again without requesting a network-layer acknowledgment for it; the same PassiveAckTimeout timeout value is used for each such attempt. If no acknowledgment has been received after a total of TryPassiveAcks retransmissions of the packet, network-layer acknowledgments (as described in Section 6.3.3) are used for all remaining attempts for that packet.

- Each node keeps a table of possible next-hop destination nodes, noting whether or not passive acknowledgments can typically be expected from transmission to that node, and the expected latency and jitter of a passive acknowledgment from that node. Each time a node receives a packet to be forwarded to a node other than the IP Destination Address, the node checks its table of next-hop destination nodes to determine whether to use a passive acknowledgment or a network-layer acknowledgment for that transmission to that node. The timeout for this packet can also be derived from this table. A node using this method SHOULD prefer using passive acknowledgments to network-layer acknowledgments.

In using passive acknowledgments for a packet that it originates or forwards, a node considers the later receipt of a new packet (e.g., with promiscuous receive mode enabled on its network interface) to be an acknowledgment of this first packet if both of the following two tests succeed:

- The Source Address, Destination Address, Protocol, Identification, and Fragment Offset fields in the IP header of the two packets MUST match [27], and

- If either packet contains a DSR Source Route header, both packets MUST contain one, and the value in the Segments Left field in the DSR Source Route header of the new packet MUST be less than that in the first packet.

When a node hears such a passive acknowledgment for any packet in its Maintenance Buffer, that node SHOULD remove that packet, as well as any other packets in its Maintenance Buffer with the same next-hop destination, from its Maintenance Buffer.

6.3.3. Using Network-Layer Acknowledgments

When a node originates or forwards a packet and has no other mechanism of acknowledgment available to determine reachability of the next-hop node in the source route for Route Maintenance, that node SHOULD request a network-layer acknowledgment from that next-hop node. To do so, the node inserts an Acknowledgment Request option in the DSR header in the packet. The Identification field in that

Acknowledgment Request option MUST be set to a value unique over all packets transmitted by this node to the same next-hop node that are either unacknowledged or recently acknowledged.

When a node receives a packet containing an Acknowledgment Request option, then that node performs the following tests on the packet:

- If the indicated next-hop node address for this packet does not match any of this node's own IP addresses, then this node MUST NOT process the Acknowledgment Request option. The indicated next-hop node address is the next Address[i] field in the DSR Source Route option in the DSR header in the packet, or is the IP Destination Address in the packet if the packet does not contain a DSR Source Route option or the Segments Left there is zero.

- If the packet contains an Acknowledgment option, then this node MUST NOT process the Acknowledgment Request option.

If neither of the tests above fails, then this node MUST process the Acknowledgment Request option by sending an Acknowledgment option to the previous-hop node; to do so, the node performs the following sequence of steps:

- Create a packet and set the IP Protocol field to the protocol number assigned for a DSR header (TBA???).

- Set the IP Source Address field in this packet to the IP address of this node, copied from the source route in the DSR Source Route option in that packet (or from the IP Destination Address field of the packet, if the packet does not contain a DSR Source Route option).

- Set the IP Destination Address field in this packet to the IP address of the previous-hop node, copied from the source route in the DSR Source Route option in that packet (or from the IP Source Address field of the packet, if the packet does not contain a DSR Source Route option).

- Add a DSR header to the packet, and set the DSR header's Next Header field to the "No Next Header" value.

- Add an Acknowledgment option to the DSR header in the packet; set the Acknowledgment option's Option Type field to 6 and the Opt Data Len field to 10.

- Copy the Identification field from the received Acknowledgment Request option into the Identification field in the Acknowledgment option.

- Set the ACK Source Address field in the Acknowledgment option to be the IP Source Address of this new packet (set above to be the IP address of this node).

- Set the ACK Destination Address field in the Acknowledgment option to be the IP Destination Address of this new packet (set above to be the IP address of the previous-hop node).

- Send the packet as described in Section 6.1.1.

Packets containing an Acknowledgment option SHOULD NOT be placed in the Maintenance Buffer.

When a node receives a packet with both an Acknowledgment option and an Acknowledgment Request option, if that node is not the destination of the Acknowledgment option (the IP Destination Address of the packet), then the Acknowledgment Request option MUST be ignored. Otherwise (that node is the destination of the Acknowledgment option), that node MUST process the Acknowledgment Request option by returning an Acknowledgment option according to the following sequence of steps:

- Create a packet and set the IP Protocol field to the protocol number assigned for a DSR header (TBA???).

- Set the IP Source Address field in this packet to the IP address of this node, copied from the source route in the DSR Source Route option in that packet (or from the IP Destination Address field of the packet, if the packet does not contain a DSR Source Route option).

- Set the IP Destination Address field in this packet to the IP address of the node originating the Acknowledgment option.

- Add a DSR header to the packet, and set the DSR header's Next Header field to the "No Next Header" value.

- Add an Acknowledgment option to the DSR header in this packet; set the Acknowledgment option's Option Type field to 6 and the Opt Data Len field to 10.

- Copy the Identification field from the received Acknowledgment Request option into the Identification field in the Acknowledgment option.

- Set the ACK Source Address field in the option to be the IP Source Address of this new packet (set above to be the IP address of this node).

- Set the ACK Destination Address field in the option to be the IP Destination Address of this new packet (set above to be the IP address of the node originating the Acknowledgment option.)

- Send the packet directly to the destination. The IP Destination Address MUST be treated as a direct neighbor node: the transmission to that node MUST be done in a single IP forwarding hop, without Route Discovery and without searching the Route Cache. In addition, this packet MUST NOT contain a DSR Acknowledgment Request, MUST NOT be retransmitted for Route Maintenance, and MUST NOT expect a link-layer acknowledgment or passive acknowledgment.

When using network-layer acknowledgments for Route Maintenance, a node SHOULD use an adaptive algorithm in determining the retransmission timeout for each transmission attempt of an acknowledgment request. For example, a node SHOULD maintain a separate round-trip time (RTT) estimate for each to which it has recently attempted to transmit packets, and it SHOULD use this RTT estimate in setting the timeout for each retransmission attempt for Route Maintenance. The TCP RTT estimation algorithm has been shown to work well for this purpose in implementation and testbed experiments with DSR [20, 22].

6.3.4. **Originating a Route Error**

When a node is unable to verify reachability of a next-hop node after reaching a maximum number of retransmission attempts, a node SHOULD send a Route Error to the IP Source Address of the packet. When sending a Route Error for a packet containing either a Route Error option or an Acknowledgment option, a node SHOULD add these existing options to its Route Error, subject to the limit described below.

A node transmitting a Route Error MUST perform the following steps:

- Create an IP packet and set the Source Address field in this packet's IP header to the address of this node.

- If the Salvage field in the DSR Source Route option in the packet triggering the Route Error is zero, then copy the Source Address field of the packet triggering the Route Error into the Destination Address field in the new packet's IP header; otherwise, copy the Address[1] field from the DSR Source Route option of the packet triggering the Route Error into the Destination Address field in the new packet's IP header

- Insert a DSR header into the new packet.

- Add a Route Error Option to the new packet, setting the Error Type to NODE_UNREACHABLE, the Salvage value to the Salvage value from the DSR Source Route option of the packet triggering the Route Error, and the Unreachable Node Address field to the address of the next-hop node from the original source route. Set the Error Source Address field to this node's IP address, and the Error Destination field to the new packet's IP Destination Address.

- If the packet triggering the Route Error contains any Route Error or Acknowledgment options, the node MAY append to its Route Error each of these options, with the following constraints:

  - The node MUST NOT include any Route Error option from the packet triggering the new Route Error, for which the total salvage count (Section 5.4) of that included Route Error would be greater than MAX_SALVAGE_COUNT in the new packet.

  - If any Route Error option from the packet triggering the new Route Error is not included in the packet, the node MUST NOT include any following Route Error or Acknowledgment options from the packet triggering the new Route Error.

  - Any appended options from the packet triggering the Route Error MUST follow the new Route Error in the packet.

  - In appending these options to the new Route Error, the order of these options from the packet triggering the Route Error MUST be preserved.

- Send the packet as described in Section 6.1.1.

6.3.5. Processing a Received Route Error Option

When a node receives a packet containing a Route Error option, that node MUST process the Route Error option according to the following sequence of steps:

- The node MUST remove from its Route Cache the link from the node identified by the Error Source Address field to the node identified by the Unreachable Node Address field (if this link is present in its Route Cache). If the node implements its Route Cache as a link cache, as described in Section 4.1, only this single link is removed; if the node implements its Route Cache as a path cache, however, all routes (paths) that use this link are removed.

- If the option following the Route Error is an Acknowledgment or Route Error option sent by this node (that is, with

Acknowledgment or Error Source Address equal to this node's address), copy the DSR options following the current Route Error into a new packet with IP Source Address equal to this node's own IP address and IP Destination Address equal to the Acknowledgment or Error Destination Address. Transmit this packet as described in Section 6.1.1, with the salvage count in the DSR Source Route option set to the Salvage value of the Route Error.

In addition, after processing the Route Error as described above, the node MAY initiate a new Route Discovery for any destination node for which it then has no route in its Route Cache as a result of processing this Route Error, if the node has indication that a route to that destination is needed. For example, if the node has an open TCP connection to some destination node, then if the processing of this Route Error removed the only route to that destination from this node's Route Cache, then this node MAY initiate a new Route Discovery for that destination node. Any node, however, MUST limit the rate at which it initiates new Route Discoveries for any single destination address, and any new Route Discovery initiated in this way as part of processing this Route Error MUST conform to this limit.

6.3.6. Salvaging a Packet

When an intermediate node forwarding a packet detects through Route Maintenance that the next-hop link along the route for that packet is broken (Section 6.3), if the node has another route to the packet's IP Destination Address in its Route Cache, the node SHOULD "salvage" the packet rather than discarding it. To do so using the route found in its Route Cache, this node processes the packet as follows:

- If the MAC protocol in use in the network is not capable of transmitting unicast packets over unidirectional links, as discussed in Section 3.3.1, then if this packet contains a Route Reply option, remove and discard the Route Reply option in the packet; if the DSR header in the packet then contains no DSR options, remove the DSR header from the packet. If the resulting packet then contains only an IP header, the node SHOULD NOT salvage the packet and instead SHOULD discard the entire packet.

  When returning any Route Reply in the case in which the MAC protocol in use in the network is not capable of transmitting unicast packets over unidirectional links, the source route used for routing the Route Reply packet MUST be obtained by reversing the sequence of hops in the Route Request packet (the source route that is then returned in the Route Reply). This restriction on returning a Route Reply and on salvaging a packet that contains a Route Reply option enables the Route Reply to test this sequence of hops for bidirectionality, preventing the Route Reply from being received by the initiator of the Route

Discovery unless each of the hops over which the Route Reply is returned (and thus each of the hops in the source route being returned in the Reply) is bidirectional.

- Modify the existing DSR Source Route option in the packet so that the Address[i] fields represent the source route found in this node's Route Cache to this packet's IP Destination Address. Specifically, the node copies the hop addresses of the source route into sequential Address[i] fields in the DSR Source Route option, for i = 1, 2, ..., n. Address[1] here is the address of the salvaging node itself (the first address in the source route found from this node to the IP Destination Address of the packet). The value n here is the number of hop addresses in this source route, excluding the destination of the packet (which is instead already represented in the Destination Address field in the packet's IP header).

- Initialize the Segments Left field in the DSR Source Route option to n as defined above.

- The First Hop External (F) bit in the DSR Source Route option is copied from the External bit flagging the first hop in the source route for the packet, as indicated in the Route Cache.

- The Last Hop External (L) bit in the DSR Source Route option is copied from the External bit flagging the last hop in the source route for the packet, as indicated in the Route Cache.

- The Salvage field in the DSR Source Route option is set to 1 plus the value of the Salvage field in the DSR Source Route option of the packet that caused the error.

- Transmit the packet to the next-hop node on the new source route in the packet, using the forwarding procedure described in Section 6.1.5.

As described in Section 6.3.4, the node in this case also SHOULD return a Route Error to the original sender of the packet. If the node chooses to salvage the packet, it SHOULD do so after originating the Route Error.

7. Multiple Interface Support

A node in DSR MAY have multiple network interfaces that support ad hoc network routing. This section describes special packet processing at such nodes.

A node with multiple network interfaces MUST have some policy for determining which Request packets are forwarded out which network interfaces. For example, a node MAY choose to forward all Requests out all network interfaces.

When a node with multiple network interfaces propagates a Route Request on an network interface other than the one it received the Request on, it MUST modify the address list between receipt and re-propagation as follows:

- Append the address of the incoming interface

- If the incoming interface and outgoing interface differ in whether or not they require bidirectionality for unicast transmission, append the address 127.0.0.1

- If the incoming interface and outgoing interface differ in whether or not unidirectional links are common, append the address 127.0.0.2

- Append the address of the outgoing interface

When a node forwards a packet containing a source route, it MUST assume that the next hop is reachable on the incoming interface, unless the next hop is the address of one of this node's interfaces, in which case this node MUST process the packet in the same way as if the node had just received it from that interface.

If a node which previously had multiple network interfaces receives a packet sent with a source route specifying an interface change to an interface that is no longer available, it MAY send a Route Error to the source of the packet without attempting to forward the packet on the incoming interface, unless the network uses an autoconfiguration mechanism that may have allowed another node to acquire the now unused address of the unavailable interface.

Source routes MUST never contain the special addresses 127.0.0.1 and 127.0.0.2.

8. Fragmentation and Reassembly

When a node using DSR wishes to fragment a packet that contains a DSR header not containing a Route Request option, it MUST perform the following sequence of steps:

- Remove the DSR header from the packet.

- Fragment the packet.

- IP-in-IP encapsulate each fragment.

- Add the DSR header to each fragment. If a Source Route header is present in the DSR header, increment the Salvage field.

When a node using the DSR protocol receives an IP-in-IP encapsulated packet destined to itself, it SHOULD decapsulate the packet and reassemble any fragments contained inside, in accordance with RFC 791 [27].

9. Protocol Constants and Configuration Variables

Any DSR implementation MUST support the following configuration variables and MUST support a mechanism enabling the value of these variables to be modified by system management. The specific variable names are used for demonstration purposes only, and an implementation is not required to use these names for the configuration variables, so long as the external behavior of the implementation is consistent with that described in this document.

For each configuration variable below, the default value is specified to simplify configuration. In particular, the default values given below are chosen for a DSR network running over 2 Mbps IEEE 802.11 network interfaces using the Distributed Coordination Function (DCF) MAC with RTS and CTS [11, 5].

BroadcastJitter	10	milliseconds
RouteCacheTimeout	300	seconds
SendBufferTimeout	30	seconds
RequestTableSize	64	nodes
RequestTableIds	16	identifiers
MaxRequestRexmt	16	retransmissions
MaxRequestPeriod	10	seconds
RequestPeriod	500	milliseconds
NonpropRequestTimeout	30	milliseconds
RexmtBufferSize	50	packets
MaintHoldoffTime	250	milliseconds
MaxMaintRexmt	2	retransmissions
TryPassiveAcks	1	attempt
PassiveAckTimeout	100	milliseconds
GratReplyHoldoff	1	second

In addition, the following protocol constant MUST be supported by any implementation of the DSR protocol:

MAX_SALVAGE_COUNT	15	salvages

10. IANA Considerations

This document proposes the use of a DSR header, which requires an IP Protocol number.

In addition, this document proposes use of the value "No Next Header" (originally defined for use in IPv6) within an IPv4 packet, to indicate that no further header follows a DSR header.

11. Security Considerations

This document does not specifically address security concerns. This document does assume that all nodes participating in the DSR protocol do so in good faith and without malicious intent to corrupt the routing ability of the network. In mission-oriented environments where all the nodes participating in the DSR protocol share a common goal that motivates their participation in the protocol, the communications between the nodes can be encrypted at the physical channel or link layer to prevent attack by outsiders.

Appendix A. Link-MaxLife Cache Description

As guidance to implementors of DSR, the description below outlines the operation of a possible implementation of a Route Cache for DSR that has been shown to outperform other other caches studied in detailed simulations. Use of this design for the Route Cache is recommended in implementations of DSR.

This cache, called "Link-MaxLife" [9], is a link cache, in that each individual link (hop) in the routes returned in Route Reply packets (or otherwise learned from the header of overhead packets) is added to a unified graph data structure of this node's current view of the network topology, as described in Section 4.1. To search for a route in this cache to some destination node, the sending node uses a graph search algorithm, such as the well-known Dijkstra's shortest-path algorithm, to find the current best path through the graph to the destination node.

The Link-MaxLife form of link cache is adaptive in that each link in the cache has a timeout that is determined dynamically by the caching node according to its observed past behavior of the two nodes at the ends of the link; in addition, when selecting a route for a packet being sent to some destination, among cached routes of equal length (number of hops) to that destination, Link-MaxLife selects the route with the longest expected lifetime (highest minimum timeout of any link in the route).

Specifically, in Link-MaxLife, a link's timeout in the Route Cache is chosen according to a "Stability Table" maintained by the caching node. Each entry in a node's Stability Table records the address of another node and a factor representing the perceived "stability" of this node. The stability of each other node in a node's Stability Table is initialized to InitStability. When a link from the Route Cache is used in routing a packet originated or salvaged by that node, the stability metric for each of the two endpoint nodes of that link is incremented by the amount of time since that link was last used, multiplied by StabilityIncrFactor (StabilityIncrFactor >= 1); when a link is observed to break and the link is thus removed from the Route Cache, the stability metric for each of the two endpoint nodes of that link is multiplied by StabilityDecrFactor (StabilityDecrFactor < 1).

When a node adds a new link to its Route Cache, the node assigns a lifetime for that link in the Cache equal to the stability of the less "stable" of the two endpoint nodes for the link, except that a link is not allowed to be given a lifetime less than MinLifetime. When a link is used in a route chosen for a packet originated or salvaged by this node, the link's lifetime is set to be at least

UseExtends into the future; if the lifetime of that link in the

Route Cache is already further into the future, the lifetime remains unchanged.

When a node using Link-MaxLife selects a route from its Route Cache for a packet being originated or salvaged by this node, it selects the shortest-length route that has the longest expected lifetime (highest minimum timeout of any link in the route), as opposed to simply selecting an arbitrary route of shortest length.

The following configuration variables are used in the description of Link-MaxLife above. The specific variable names are used for demonstration purposes only, and an implementation is not required to use these names for these configuration variables. For each configuration variable below, the default value is specified to simplify configuration. In particular, the default values given below are chosen for a DSR network where nodes move at relative velocities between 12 and 25 seconds per transmission radius.

InitStability	25	seconds
StabilityIncrFactor	4	
StabilityDecrFactor	2	
MinLifetime	1	second
UseExtends	120	seconds

Appendix B. Location of DSR in the ISO Network Reference Model

When designing DSR, we had to determine at what layer within the protocol hierarchy to implement ad hoc network routing. We considered two different options: routing at the link layer (ISO layer 2) and routing at the network layer (ISO layer 3). Originally, we opted to route at the link layer for several reasons:

- Pragmatically, running the DSR protocol at the link layer maximizes the number of mobile nodes that can participate in ad hoc networks. For example, the protocol can route equally well between IPv4 [27], IPv6 [6], and IPX [32] nodes.

- Historically [13, 14], DSR grew from our contemplation of a multi-hop propagating version of the Internet's Address Resolution Protocol (ARP) [25], as well as from the routing mechanism used in IEEE 802 source routing bridges [24]. These are layer 2 protocols.

- Technically, we designed DSR to be simple enough that it could be implemented directly in the firmware inside wireless network interface cards [13, 14], well below the layer 3 software within a mobile node. We see great potential in this for DSR running inside a cloud of mobile nodes around a fixed base station, where DSR would act to transparently extend the coverage range to these nodes. Mobile nodes that would otherwise be unable to communicate with the base station due to factors such as distance, fading, or local interference sources could then reach the base station through their peers.

Ultimately, however, we decided to specify and to implement [20] DSR as a layer 3 protocol, since this is the only layer at which we could realistically support nodes with multiple network interfaces of different types forming an ad hoc network.

Appendix C. Implementation and Evaluation Status

The initial design of the DSR protocol, including DSR's basic Route Discovery and Route Maintenance mechanisms, was first published in December 1994 [13], with significant additional design details and initial simulation results published in early 1996 [14].

The DSR protocol has been extensively studied since then through additional detailed simulations. In particular, we have implemented DSR in the ns-2 network simulator [23, 5] and performed extensive simulations of DSR using ns-2 (e.g., [5, 19]). We have also conducted evaluations of different caching strategies documented in this draft [9].

We have also implemented the DSR protocol under the FreeBSD 2.2.7 operating system running on Intel x86 platforms. FreeBSD [8] is based on a variety of free software, including 4.4 BSD Lite from the University of California, Berkeley. For the environments in which we used it, this implementation is functionally equivalent to the version of the DSR protocol specified in this draft.

During the 7 months from August 1998 to February 1999, we designed and implemented a full-scale physical testbed to enable the evaluation of ad hoc network performance in the field, in an actively mobile ad hoc network under realistic communication workloads. The last week of February and the first week of March of 1999 included demonstrations of this testbed to a number of our sponsors and partners, including Lucent Technologies, Bell Atlantic, and DARPA. A complete description of the testbed is available as a Technical Report [20].

We have since ported this implementation of DSR to FreeBSD 3.3, and we have also added a preliminary version of Quality of Service (QoS) support for DSR. A demonstration of this modified version of DSR was presented in July 2000. These QoS features are not included in this draft, and will be added later in a separate draft on top of the base protocol specified here.

DSR has also been implemented under Linux by Alex Song at the University of Queensland, Australia [31]. This implementation supports the Intel x86 PC platform and the Compaq iPAQ.

The Network and Telecommunications Research Group at Trinity College Dublin have implemented a version of DSR on Windows CE.

Several other independent groups have also used DSR as a platform for their own research, or and as a basis of comparison between ad hoc network routing protocols.

Changes from Previous Version of the Draft

This appendix briefly lists some of the major changes in this draft relative to the previous version of this same draft, draft-ietf-manet-dsr-06.txt:

- Added a blacklist mechanism for handling unidirectional links when the network interface requires bidirectionality.

- Added language describing multiple interface support.

- Described fragmentation and reassembly processing.

- Updated the implementation and evaluation list.

Acknowledgements

The protocol described in this draft has been designed and developed within the Monarch Project, a research project at Rice University (previously at Carnegie Mellon University) that is developing adaptive networking protocols and protocol interfaces to allow truly seamless wireless and mobile node networking [15, 30].

The authors would like to acknowledge the substantial contributions of Josh Broch in helping to design, simulate, and implement the DSR protocol. Josh is currently on leave of absence from Carnegie Mellon University at AON Networks. We thank him for his contributions to earlier versions of this draft.

We would also like to acknowledge the assistance of Robert V. Barron at Carnegie Mellon University. Bob ported our DSR implementation from FreeBSD 2.2.7 into FreeBSD 3.3.

Many valuable suggestions came from participants in the IETF process. We would like to acknowledge Fred Baker, who provided extensive feedback on our previous draft, as well as the working group chairs, for their suggestions of previous versions of the draft.

References

[1] David F. Bantz and Frederic J. Bauchot. Wireless LAN Design Alternatives. IEEE Network, 8(2):43--53, March/April 1994.

[2] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In Proceedings of the ACM SIGCOMM '94 Conference, pages 212--225, August 1994.

[3] Robert T. Braden, editor. Requirements for Internet Hosts---Communication Layers. RFC 1122, October 1989.

[4] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, March 1997.

[5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 85--97, October 1998.

[6] Stephen E. Deering and Robert M. Hinden. Internet Protocol Version 6 (IPv6) Specification. RFC 2460, December 1998.

[7] Ralph Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.

[8] The FreeBSD Project. Project web page available at http://www.freebsd.org/.

[9] Yih-Chun Hu and David B. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking, August 2000.

[10] Yih-Chun Hu, David B. Johnson, and David A. Maltz. Flow State in the Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsrflow-00.txt, February 2001. Work in progress.

[11] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[12] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In Proceedings

of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 195--206, August 1999.

[13] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, pages 158--163, December 1994.

[14] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153--181. Kluwer Academic Publishers, 1996.

[15] David B. Johnson and David A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. IEEE Personal Communications, 3(1):34--42, February 1996.

[16] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. Proceedings of the IEEE, 75(1):21--32, January 1987.

[17] Phil Karn. MACA---A New Channel Access Method for Packet Radio. In ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pages 134--140, September 1990.

[18] Gregory S. Lauer. Packet-Radio Routing. In Routing in Communications Networks, edited by Martha E. Steenstrup, chapter 11, pages 351--396. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[19] David A. Maltz, Josh Broch, Jorjeta Jetcheva, and David B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. IEEE Journal on Selected Areas of Communications, 17(8):1439--1453, August 1999.

[20] David A. Maltz, Josh Broch, and David B. Johnson. Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 1999.

[21] David A. Maltz, Josh Broch, and David B. Johnson. Quantitative Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. In Proceedings of the IEEE Wireless Communications and Networking Conference, September 2000.

[22] David A. Maltz, Josh Broch, and David B. Johnson. Lessons From a Full-Scale MultiHop Wireless Ad Hoc Network Testbed. IEEE Personal Communications, 8(1):8--15, February 2001.

[23] The Network Simulator -- ns-2. Project web page available at

http://www.isi.edu/nsnam/ns/.

[24] Radia Perlman. Interconnections: Bridges and Routers. Addison-Wesley, Reading, Massachusetts, 1992.

[25] David C. Plummer. An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Addresses for Transmission on Ethernet Hardware. RFC 826, November 1982.

[26] J. B. Postel, editor. Internet Control Message Protocol. RFC 792, September 1981.

[27] J. B. Postel, editor. Internet Protocol. RFC 791, September 1981.

[28] J. B. Postel, editor. Transmission Control Protocol. RFC 793, September 1981.

[29] Joyce K. Reynolds and Jon Postel. Assigned Numbers. RFC 1700, October 1994. See also http://www.iana.org/numbers.html.

[30] Rice University Monarch Project. Monarch Project Home Page. Available at http://www.monarch.cs.rice.edu/.

[31] Alex Song. picoNet II: A Wireless Ad Hoc Network for Mobile Handheld Devices. Submitted for the degree of Bachelor of Engineering (Honours) in the division of Electrical Engineering, Department of Information Technology and Electrical Engineering, University of Queensland, Australia, October 2001. Available at http://student.uq.edu.au/~s369677/main.html.

[32] Paul Turner. NetWare Communications Processes. NetWare Application Notes, Novell Research, pages 25--91, September 1990.

[33] Gary R. Wright and W. Richard Stevens. TCP/IP Illustrated, Volume 2: The Implementation. Addison-Wesley, Reading, Massachusetts, 1995.

Chair's Address

The MANET Working Group can be contacted via its current chairs:

M. Scott Corson
Flarion Technologies, Inc.
Bedminster One
135 Route 202/206 South
Bedminster, NJ 07921
USA

Phone: +1 908 947-7033
Email: corson@flarion.com

Joseph Macker
Information Technology Division
Naval Research Laboratory
Washington, DC 20375
USA

Phone: +1 202 767-2001
Email: macker@itd.nrl.navy.mil

Authors' Addresses

Questions about this document can also be directed to the authors:

David B. Johnson   Phone: +1 713 348-3063
Rice University   Fax: +1 713 348-5930
Computer Science Department, MS 132   Email: dbj@cs.rice.edu
6100 Main Street
Houston, TX 77005-1892
USA

David A. Maltz   Phone: +1 650 688-3128
AON Networks   Fax: +1 650 688-3119
3045 Park Blvd.   Email: dmaltz@cs.cmu.edu
Palo Alto, CA 94306
USA

Yih-Chun Hu   Phone: +1 412 268-3075
Rice University   Fax: +1 412 268-5576
Computer Science Department, MS 132   Email: yihchun@cs.cmu.edu
6100 Main Street
Houston, TX 77005-1892
USA

Jorjeta G. Jetcheva   Phone: +1 412 268-3053
Carnegie Mellon University   Fax: +1 412 268-5576
Computer Science Department   Email: jorjeta@cs.cmu.edu
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
USA