IETF MANET Working Group
INTERNET-DRAFT
2 March 2001

David B. Johnson, Rice University
David A. Maltz, AON Networks
Yih-Chun Hu, Rice University
Jorjeta G. Jetcheva, Carnegie Mellon University

The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks

<draft-ietf-manet-dsr-05.txt>

Status of This Memo

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 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 IP packets in multi-hop wireless ad hoc networks.

Contents

1. Introduction

The Dynamic Source Routing protocol (DSR) [12, 13] 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 may 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 quickly to changes in the network. The DSR protocol provides highly reactive service 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 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 level 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 uni-directional 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 uni-directional 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 IP packets in multi-hop wireless ad hoc networks. Advanced, optional features, such as Quality of Service (QoS) support and efficient multicast routing, 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 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 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 [13]. 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, 17]. That is, wireless communications between each pair of nodes will in many cases be able to operate bi-directionally, but at times the wireless link between two nodes may be only uni-directional, 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 bi-directional links, DSR can successfully discover and forward packets over paths that contain uni-directional links. Some MAC protocols, however, such as MACA [16], MACAW [2], or IEEE 802.11 [10], limit unicast data packet transmission to bi-directional links, due to the required bi-directional exchange of RTS and CTS packets in these protocols and due to the link-level acknowledgement 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 easy ability to reverse a source route to obtain a route back to the origin of the original route.

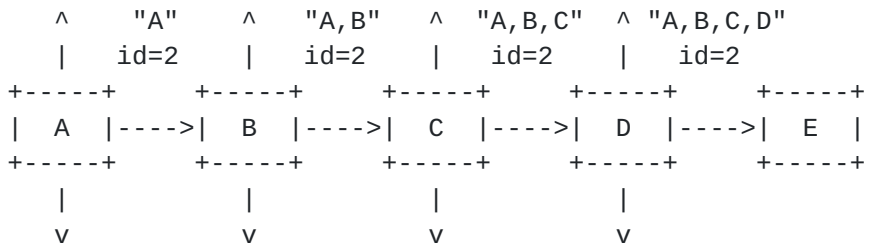
The IP address used by a node using the DSR protocol MAY be assigned by any mechanism (e.g., static assignment or use of DHCP for dynamic assignment [8]), although the method of such assignment is outside the scope of this specification.

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, but 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:



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 [25], 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 bi-directional frame exchange as part of the MAC protocol [10], this route reversal is preferred, as it avoids the overhead of a possible second Route Discovery, and it tests the discovered route to ensure it is bi-directional before the Route Discovery initiator begins using the route. However, this technique will prevent the discovery of routes using uni-directional links. In wireless environments where the use of uni-directional links is permitted, such routes may in some cases be more efficient than those with only bi-directional 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

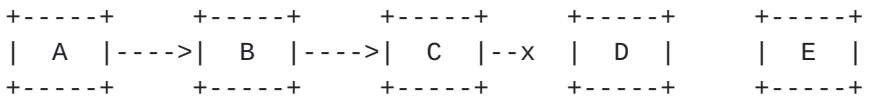
destination address. However, the node MUST limit the rate at which such new Route Discoveries for the same address are initiated, since

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 MUST use an exponential back-off algorithm to limit the rate at which it initiates new Route Discoveries 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 the packet has been received by the next hop along the source route; the packet SHOULD be retransmitted (up to a maximum number of attempts) until this confirmation of receipt is received. 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:



In this case, node A is responsible for receipt of the packet at B, node B is responsible for receipt at C, node C is responsible for receipt at D, and node D is responsible for receipt finally at the destination E.

This confirmation of receipt in many cases may be provided at no cost to DSR, either as an existing standard part of the MAC protocol in use (such as the link-level acknowledgement frame defined by IEEE

802.11 [10]), or by a "passive acknowledgement" [15] (in which, for example, B confirms receipt at C by overhearing C transmit the

packet when forwarding it on to D). If neither of these confirmation mechanisms are available, the node transmitting the packet can explicitly request a DSR-specific software acknowledgement be returned by the next hop; this software acknowledgement will normally be transmitted directly to the sending node, but if the link between these two nodes is uni-directional, this software acknowledgement may travel over a different, multi-hop path.

If no receipt confirmation is received after the packet has been retransmitted the maximum number of attempts by some hop, this node SHOULD return a "Route Error" to the original sender of the packet, identifying the link over which the packet could not be forwarded. For example, in the example shown above, if C is unable to deliver the packet to the next hop D, then C returns a Route Error to A, stating that the link from C to D is currently "broken". 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 exponential back-off described in Section 3.1).

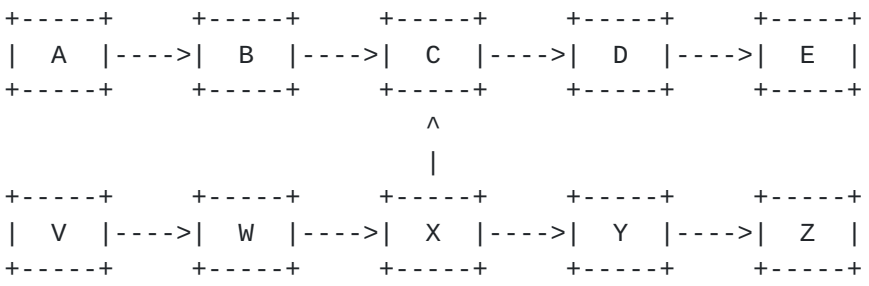
3.3. Additional Route Discovery Features

3.3.1. Caching Overheard Routing Information

A node forwarding or otherwise overhearing any packet MAY add the routing information from that packet to its own Route Cache. In particular, 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 MAY all be cached by any node. Routing information from any of these packets received can be cached, whether the packet was addressed to this node, sent to a broadcast (or multicast) MAC address, or received while the node's network interface is in promiscuous mode.

One limitation, however, on caching of such overheard routing information is the possible presence of uni-directional links in the

ad hoc network (Section 2). For example, in the situation shown below, node A is using a source route to communicate with node E:



As node C forwards a data packet along the route from A to E, it MAY 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. However, the "reverse" direction of the links identified in the packet headers, from itself back to B and from B to A, may not work for it since these links might be uni-directional. If C knows that the links are in fact bi-directional, for example due to the MAC protocol in use, it could cache them but otherwise SHOULD not.

Likewise, node V in the example above is using a different source route to communicate with node Z. If node C overhears node X transmitting a data packet to forward it to Y (from V), node C SHOULD consider whether the links involved can be known to be bi-directional or not before caching them. If the link from X to C (over which this data packet was received) can be known to be bi-directional, then C MAY cache the link from itself to X, the link from X to Y, and the link from Y to Z. If all links can be assumed to be bi-directional, C MAY also cache the links from X to W and from W to V. Similar considerations apply to the routing information that might be learned from forwarded or otherwise overheard Route Request or Route Reply packets.

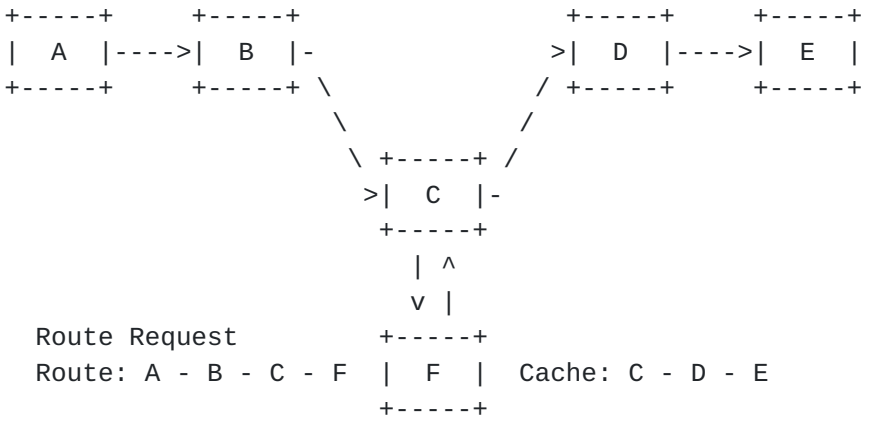
3.3.2. Replying to Route Requests using Cached Routes

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:



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 for two reasons. First, this limitation 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. Second, this limitation means that a 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 the Route Request does not meet these restrictions, the node (node F in this example) discards the Route Request rather than replying to it or propagating it.

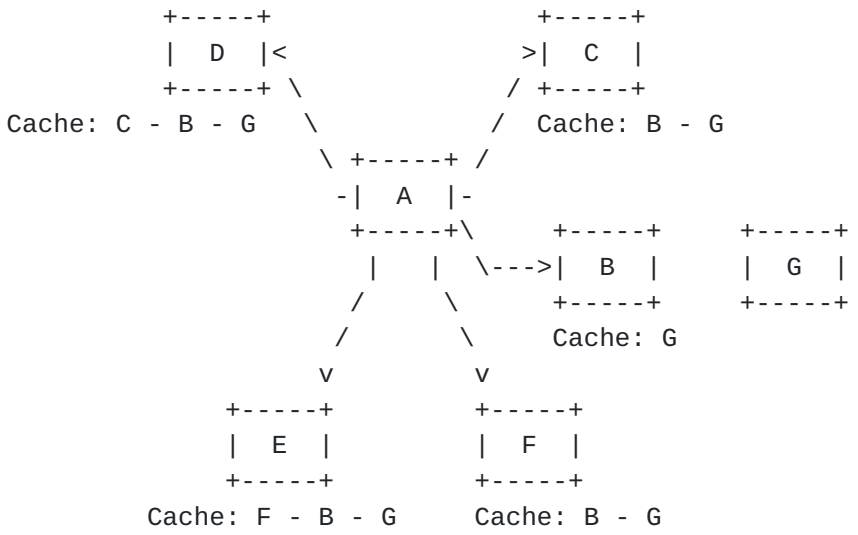
3.3.3. Preventing Route Reply Storms

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:



Normally, these nodes would all attempt to reply from their own Route Caches, and would 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 replies from different nodes all receiving the Route Request may create packet collisions among some or all of these Replies and may cause local congestion in the wireless network. In addition, it will often be the case that the different replies will indicate routes of different lengths, as shown in this example.

If a node can put its network interface into promiscuous receive mode, it SHOULD delay sending its own Route Reply for a short period, while listening to see if the initiating node begins using a shorter route first. That is, this node SHOULD 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 send 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 rebroadcasting 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 a "propagating" Route Request (i.e., with no hop limit) MAY be sent.

Another possible use of the hop limit in a Route Request is to implement an "expanding ring" search for the target [13]. For example, a node could send an initial non-propagating Route Request as described above; if no Route Reply is received for it, the node could initiate another Route Request with a hop limit of 2. For each Route Request initiated, if no Route Reply is received for it, the node could double 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

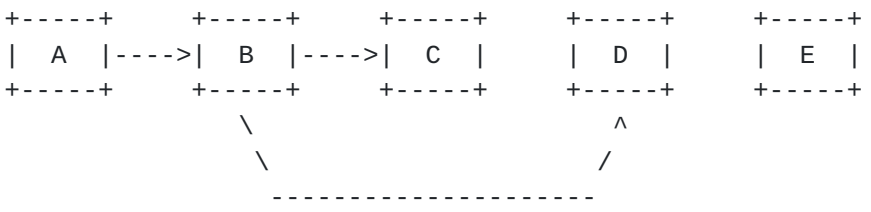
After sending a Route Error message as part of Route Maintenance as described in Section 3.2, a node MAY attempt to "salvage" the data packet that caused the Route Error rather than discarding the packet. To attempt to salvage a packet, the node sending a Route Error searches its own Route Cache for a route from itself to the destination of the packet causing the Error. If such a route is found, the node MAY salvage the packet after returning the Route Error by replacing 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 in this way, 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.

3.4.2. Automatic Route Shortening

Source routes in use MAY be automatically shortened if one or more intermediate hops in the route become no longer necessary. This mechanism of automatically shortening routes in use is somewhat similar to the use of passive acknowledgements [15]. 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 unused portion of that source route. If this node is not the intended next hop for the packet but is named in the later unused 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:



In this case, this node (node D) returns 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.

3.4.3. 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 routing information needed by a node participating in an ad hoc network using 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 either a Route Reply or a DSR Routing header. 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.

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 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 node in the Route Cache, copied from the First Hop External (F) and Last Hop External (L) bits in the Source Route option or Route Reply option from which the link to this node 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 SHOULD prefer routes that do not have the External flag set on any node. 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 nodes other than possibly the first node, the last node, 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, 11, 18] and through implementation of DSR in a mobile outdoor testbed under significant workload [19, 20, 20].

- 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: 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.

4.2. Route Request Table

The Route Request Table 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 that this node last originated a Route Request for that target node.
- The number of consecutive Route Requests 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.
- The Time-to-Live (TTL) field used in the IP header of last Route Request initiated by this node for that target node.

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 REQUEST_TABLE_IDS 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, REQUEST_TABLE_IDS, MUST NOT be unlimited, since, in the worst case, when a node crashes and reboots, the first REQUEST_TABLE_IDS 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 SHOULD base its initial Identification value after rebooting on a random number.

4.3. 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 SEND_BUFFER_TIMEOUT seconds 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.

4.4. Retransmission Buffer

The Retransmission Buffer of a node implementing DSR is a queue of packets sent by this node that are awaiting the receipt of an acknowledgment from the next hop in the source route (Section 5.7). For each packet in the Retransmission Buffer, a node maintains (1) a count of the number of retransmissions and (2) the time of the last retransmission.

Packets are removed from the Retransmission Buffer when an acknowledgment is received or when the number of retransmissions exceeds DSR_MAXRXTSHIFT. In the later case, the removal of the packet from the Retransmission Buffer SHOULD result in a Route Error being returned to the original source of the packet (Section 6.3).

5. DSR Header Format

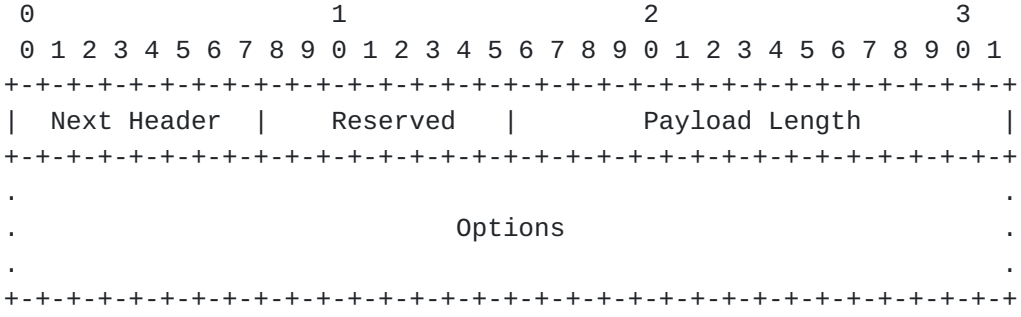
The Dynamic Source Routing protocol makes use of a special header carrying control information that can be included in any existing IP packet. This DSR header in a packet contains a small fixed-sized, 4-octet portion, followed by a sequence of zero or more DSR options carrying optional information. The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

The DSR header is inserted in the packet 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.

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 any 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:



Next Header

8-bit selector. Identifies the type of header immediately following the DSR header. Uses the same values as the IPv4 Protocol field [26].

Reserved

Sent as 0; 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. A node inserting a DSR header into a packet MUST set the Don't Fragment (DF) bit in the packet's IP header.

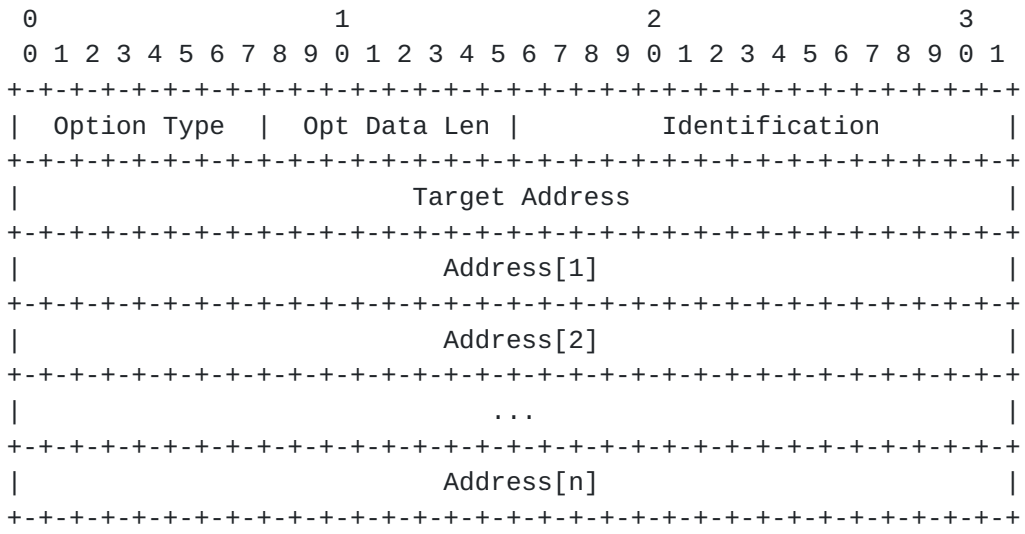
The following types of DSR options are defined in this document for

use within a DSR header:

- Route Request option (Section 5.2)
- Route Reply option (Section 5.3)
- Route Error option (Section 5.4)
- Acknowledgement Request option (Section 5.5)
- Acknowledgement option (Section 5.6)
- Source Route option (Section 5.7)
- Pad1 option (Section 5.8)
- PadN option (Section 5.9)

5.2. Route Request Option

The Route Request DSR option is encoded as follows:



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.

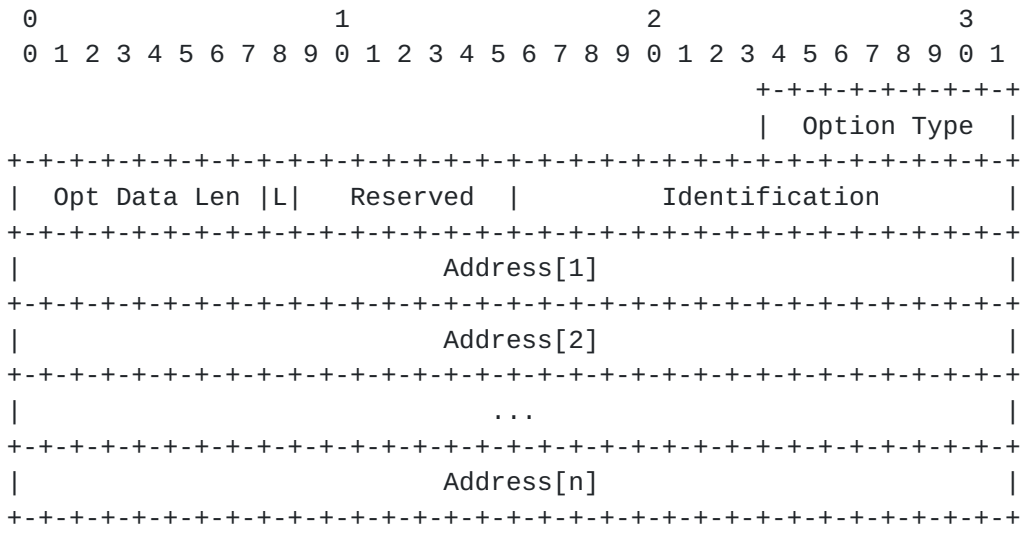
Address[1..n]

Address[i] is the address of the i-th hop 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 - 2) / 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 DSR option is encoded as follows:



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 3.4.2), 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 node indicated by the Route Reply (Address[n]) is actually in a network external to the DSR network; the exact sequence of hops leading to it 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 SHOULD prefer routes that contain no hops flagged as External.

Reserved

Sent as 0; ignored on reception.

Identification

Copied from the Identification field of the Route Request for which this Reply is sent in response. Sent as 0 if the Route Reply is not sent in response to a Route Request (a gratuitous Route Reply).

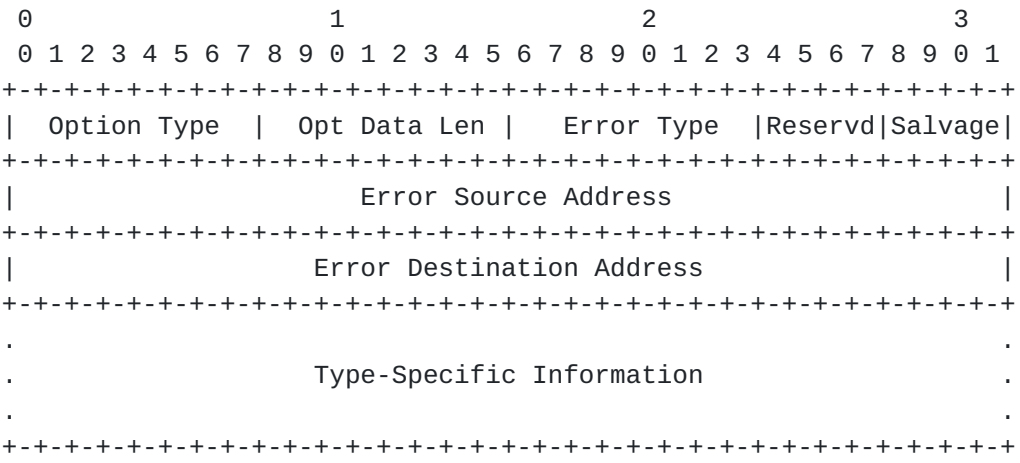
Address[1..n]

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 - 3) / 4).

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

5.4. Route Error Option

The Route Error DSR option is encoded as follows:



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. Sent as 0; ignored on reception.

Salvage

A 4-bit unsigned integer. Copied from the Salvage field in the Source Route option of the packet triggering the Route Error, incremented by the node returning the Route Error.

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).

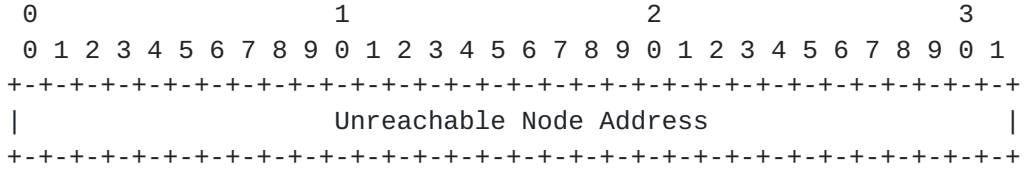
Error Destination Address

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:



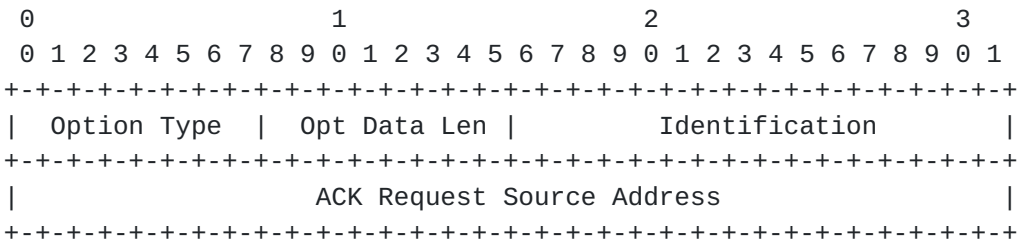
Unreachable Node Address

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 DSR option is encoded as follows:



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 nonzero value and is copied into the Identification field of the Acknowledgement option when returned by the node receiving the packet over this hop.

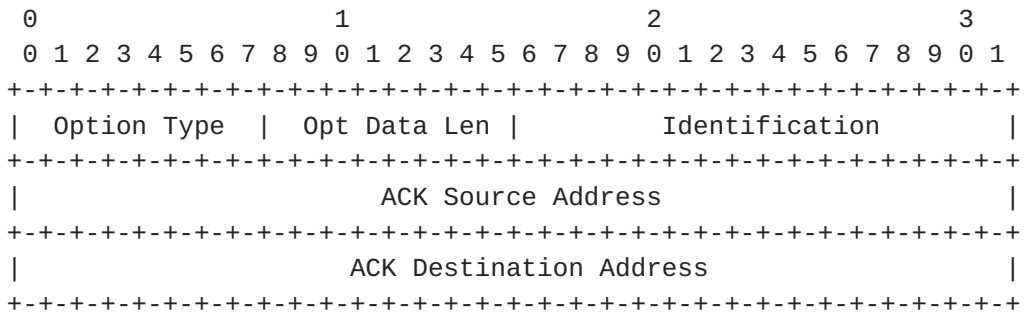
ACK Request Source Address

The address of the node requesting the acknowledgment.

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

5.6. Acknowledgment Option

The Acknowledgment DSR option is encoded as follows:



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 Acknowledgement Request option of the packet being acknowledged.

ACK Source Address

The address of the node originating the acknowledgment.

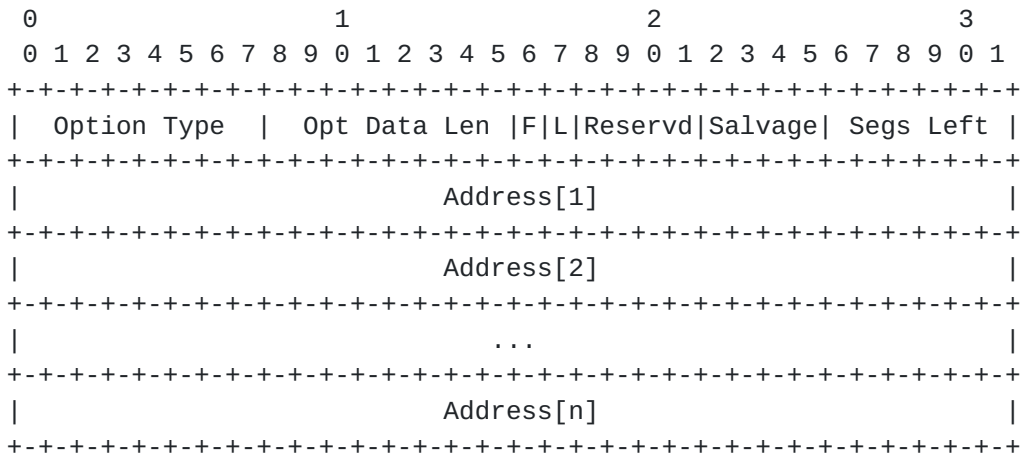
ACK Destination Address

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

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

5.7. Source Route Option

The Source Route DSR option is encoded as follows:



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 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 node indicated by the Source Route option is actually in a network external to the DSR network; the exact sequence of hops leading from it outside the DSR network are not represented in the 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 SHOULD prefer routes that contain no hops flagged as External.

Last Hop External (L)

Set to indicate that the last hop indicated by the Source Route option is actually in a network external to the DSR network; the exact sequence of hops leading to it outside the DSR network are not represented in the 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 SHOULD prefer routes that contain no hops flagged as External.

Reserved

Sent as 0; 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.

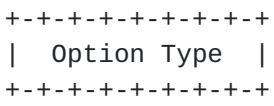
Address[1..n]

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.

When forwarding a packet along a DSR source route using a Source Route option in the packet's DSR header, the Source 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 Source Route option MUST examine the indicated source route to determine if it is the intended next hop 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 DSR option is encoded as follows:



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 nodes receiving packets containing a DSR header.

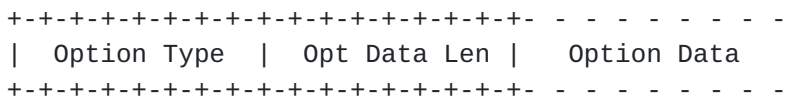
The total length of a DSR header, indicated by the Payload Length field in the DSR header MUST be a multiple of 4 octets. When building a DSR header in a packet, sufficient Pad1 or PadN options MUST be included in the Options field of the DSR header to make the total length a multiple of 4 octets.

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 DSR option is encoded as follows:



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 nodes receiving packets containing a DSR header.

The total length of a DSR header, indicated by the Payload Length field in the DSR header MUST be a multiple of 4 octets. 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.
- If the packet contains a Route Request option, then replace the IP Destination Address field with the IP "limited broadcast" address (255.255.255.255) [3].
- Else, this node must have a route to the Destination Address of the packet (since otherwise a Route Request would have been added to the packet). If the length of this route is greater than 1 hop, or if the node determines to request a DSR network-layer acknowledgement from the first hop of the route, then insert a DSR header as described in Section 6.1.2, and insert a Source Route option, as described in Section 6.1.3. The source route in the packet is initialized from the route to the Destination Address found in the Route Cache.
- Transmit the packet to the address given in the next hop, using Route Maintenance to retransmit the packet if necessary, 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 (???).

6.1.3. Adding a Source Route Option to a Packet

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

- A Source Route option, as described in Section 5.7, is created and appended 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 Destination Address from the IP header is copied into Address[n] in the DSR Source Route option.

- The first hop of the source route for the packet is copied into the Destination Address field in the IP header.

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

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

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

6.1.4. Receiving a Packet

When a node receives any packet containing a DSR header, it MUST process the packet according to the following sequence of steps:

- 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.

- Examine and process each of the options (if any) in the DSR header in the order in which they occur in the packet, skipping over any Pad1 or PadN options.

Any DSR routing information carried in a packet SHOULD be examined and reflected in the node's Route Cache, even if the options in the packet are not otherwise processed as described above. In particular, the following routing information SHOULD be handled in this way:

- In a Route Request option, the accumulated route record, represented by the IP Source Address of the packet and by the sequence of Address[i] entries in the Route Request option SHOULD be added to the node's Route Cache.

- In a Route Reply option, the route record being returned, represented by the sequence of Address[i] entries in the Route Request option and by the Destination Address in the packet's IP header SHOULD be added to the node's Route Cache.

- In an Acknowledgement option, the single link from the ACK Source Address to the ACK Destination Address SHOULD be added to the node's Route Cache.

- In a Route Error option, the single link from the Error Source Address to the Unreachable Node Address MUST be removed from the node's Route Cache.

- In a Source Route option, the indicated source route SHOULD be added to the node's Route Cache, subject to the conditions identified in Section 3.3.1. The full sequence of hops in the DSR Source Route option is as follows:

  * The Source Address in the packet's IP header is the first hop (the sender of the packet).

  * The sequence of hops

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

    follow immediately after the IP Source Address in the source route, where n is the number of addresses in the packet, or (Opt Data Len - 2) / 4.

  * The Destination Address in the packet's IP header is the final destination of the packet and is the last hop of the

source route.

In addition to the processing of received packets described above, a node SHOULD examine the packet to determine if the receipt of this packet indicates an opportunity for automatic route shortening, as described in Section 3.4.2. If the received packet satisfies the tests described there, then this node SHOULD perform the following sequence of steps:

- Return a gratuitous Route Reply to the IP Source Address of the packet, as described in Section 3.4.2.

- Discard the received 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.

6.1.5. Processing a Received Source Route Option

If a received packet contains a DSR header with a DSR Source Route option, the Source Route option MUST be examined and processed (even though this node is not indicated in the Destination Address field of the packet's IP header).

If, after processing a Source Route option in a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node MUST discard the packet and send an ICMP Packet Too Big message to the packet's Source Address [23].

A Source Route option in a DSR header for IPv4 is processed according to the following sequence of steps:

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

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

- If the value of the Segments Left field is greater than n, then send an ICMP Parameter Problem, Code 0, message [23] to the IP Source Address, pointing to the Segments Left field, and discard the packet. Do not process the 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 Source Route option further.

- 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 [24, 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 packet was received by that next hop, as described in Section 6.3.

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

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 MUST be doubled, up to a maximum of MAX_REQUEST_PERIOD, 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 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 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 last Route Request 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.

  These values MUST be used to implement an exponential back-off algorithm to limit the rate at which this node initiates new Route Discoveries for the same target address. Until a valid Route Reply is received for this target node address, the timeout between consecutive Route Discovery initiations for this target node 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 Source Route option and a Route Request option is unspecified.

Packets SHOULD NOT contain both a Source Route option and a Route Request option.

Packets containing a Route Request option SHOULD NOT be retransmitted, SHOULD NOT request a DSR acknowledgment by including an Acknowledgement Request option, SHOULD NOT expect a passive acknowledgment, and SHOULD NOT be placed in the Retransmission Buffer. The repeated transmission 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, the 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 hops

  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 node MUST then continue processing the rest of the packet normally. The node in this case MUST NOT retransmit the Route Request to propagate it to other nodes. Do not process the Route Request option further.

- 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, 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:

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

  * Create a copy of this entire packet and perform the following steps on the copy of the packet.

  * 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).

  * 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.

  * 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 BROADCAST_JITTER.

6.2.3. Generating Route Replies 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 hops

  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 hops 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.

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 addresses of the source route are copied into the Address[i] fields of the option. Address[1] MUST be set to the first hop of the route after the initiator of the Route Discovery, Address[n] MUST be set to the last hop 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 rely by a small jitter period chosen randomly between 0 and BROADCAST_JITTER milliseconds.

If the MAC layer above which DSR is operating requires bidirectionality for unidirectional transmissions, the Route Reply MUST be sent by reversing the sequence of hops that are stored in it.

If sending a Route Reply to the originator 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 originator of the original Route Request) must do another Route Request in order to return its Route Reply.

If sending the Route Reply to the originator of the Route Request does not require performing Route Discovery, a node SHOULD send a unicast Route Reply in response to every received Route Request targeted at it.

6.2.5. Processing a Route Reply Option

Upon receiving a Route Reply, a node SHOULD extract the source route from the Route Reply and add this routing information to its Route Cache. The source route from the Route Reply is the sequence of hops

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 hop Address[n] in its Route Cache as External.

Each packet in the Send Buffer SHOULD then be checked to see whether the information in the Route Reply and now in the Route Cache allows it to be sent immediately.

6.3. Route Maintenance Processing

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.

When forwarding a packet, a node MUST attempt to receive an acknowledgement for the packet from the next hop. If no acknowledgement is received, the node SHOULD return a Route Error to the IP Source Address of the packet, as described in Section 6.3.3. A node's algorithm for deciding whether or not to return a Route Error MUST NOT allow any node to attempt to send an unbounded number of packets along a broken link without receiving a Route Error.

6.3.1. Using Network-Layer Acknowledgments

When a node retransmits a packet or has no other way to ensure successful delivery of a packet to the next hop, it MUST request a network-layer acknowledgement by placing inserting an Acknowledgement Request the DSR header. The Identification value contained in that header MUST be unique over all packets delivered to the same next hop which are either unacknowledged or recently acknowledged.

A node receiving an Acknowledgement Request MUST send an acknowledgement to the previous hop by performing the following sequence of steps:

- Create a packet and set the IP Source Address to the address of this node, the IP Destination Address to the address of the previous hop, and the IP Protocol field to the protocol number reserved for DSR headers.

- Set the DSR header's Next Header field to be the "No Next Header" value.

- Set the Acknowledgement option's Option Type field to 6, and the Opt Data Len field to 10.

- Copy the Identification field from the Acknowledgement Request option into the Identification field in the Acknowledgement option. Set the ACK Source Address field in the option to be the IP Source Address and the ACK Destination Address field to the IP Destination Address.

- Send the packet as described in Section 6.1.1.

6.3.2. Using Link Layer Acknowledgments

If explicit failure notifications are provided by the link layer, then all packets are assumed to be correctly received by the next hop, and a Route Error is sent only when an explicit failure notification is made from the link layer.

Nodes receiving a packet without an Acknowledgement Request Option do not need to send an explicit Acknowledgment to the packet's originator, since the link layer will notify the originator if the packet was not received properly.

6.3.3. Originating a Route Error

When a node is unable to verify successful delivery of a packet to the next hop after a maximum number of retransmission attempts, a node SHOULD send a Route Error to the IP Source Address of the packet. In addition, a node's algorithm for deciding whether or not to return a Route Error MUST NOT allow any node to attempt to send an unbounded number of packets along a broken link without receiving a Route Error. When sending a Route Error for a packet containing either a Route Error option or an Acknowledgement option, a node SHOULD add these options to its Route Error, subject to some limit on lifetime. Specifically, we define the "salvage count" of an option to be the sum of one plus the salvage count recorded in the Source Route option plus the sum of the salvage counts of any Route Errors preceding that option.

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

- Create a packet and set the IP Source Address to the address of this node, the IP Destination Address to the address IP Source Address of the packet experiencing the error.

- Insert a DSR header into the packet.

- Add a Route Error Option, setting the Error Type to NODE_UNREACHABLE, the Reserved bits to 0, the Salvage value to one plus the Salvage value from the DSR Source Route option, and the Unreachable Node Address to the address of the next hop. Set the Error Source Address to the IP Source Address and the Error Destination to the IP Destination Address.

- The node MAY append each Route Error and Acknowledgement option, in order, from the packet experiencing the error,

though it MUST exclude options with salvage counts greater than MAX_SALVAGE_TIMES.

- Send the packet as described in Section 6.1.1.

6.3.4. Processing a Route Error Option

A node receiving a Route Error MUST process it as follows:

- Delete all routes from the Route Cache that have a link from the Route Error Source Address to the Unreachable Node Address.

- If the option following the Route Error is an Acknowledgement or Route Error option sent by this node (that is, with Acknowledgement 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 Acknowledgement or Error Destination Address. Transmit this packet as described in Section 6.1.1, with the salvage count in the Source Route option set to the Salvage value of the Route Error.

6.3.5. Salvaging a Packet

When a node is unable to verify successful delivery of a packet to the next hop after a maximum number of retransmission attempts and has transmitted a Route Error to the sender, it MAY attempt to salvage the packet by examining its route cache. If the node can find a route to the packet's IP Destination Address in its own Route Cache, then this node replaces the packet's Source Route option with a new Source Route option in the same way as described in Section 6.1.3, except that Address[1] MUST be set to the address of this node and the Salvage field MUST be set to 1 plus the value of the Salvage field in the Source Route option that caused the error.

7. Constants

BROADCAST_JITTER	10	milliseconds
MAX_ROUTE_LEN	15	nodes
MAX_SALVAGE_TIMES	15	salvages
Route Cache		
ROUTE_CACHE_TIMEOUT	300	seconds
Send Buffer		
SEND_BUFFER_TIMEOUT	30	seconds
Route Request Table		
REQUEST_TABLE_SIZE	64	nodes
REQUEST_TABLE_IDS	16	identifiers
MAX_REQUEST_REXMT	16	retransmissions
MAX_REQUEST_PERIOD	10	seconds
REQUEST_PERIOD	500	milliseconds
NONPROP_REQUEST_TIMEOUT	30	milliseconds
Retransmission Buffer		
DSR_RXMT_BUFFER_SIZE	50	packets
Retransmission Timer		
DSR_MAXRXTSHIFT	2	

8. 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.

9. 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. 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 [24], IPv6 [7], and IPX [27] nodes.

- Historically [12, 13], DSR grew from our contemplation of a multi-hop propagating version of the Internet's Address Resolution Protocol (ARP) [22], as well as from the routing mechanism used in IEEE 802 source routing bridges [21]. 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 [12, 13], 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 [19] 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 B. Implementation and Evaluation Status

The DSR protocol has been implemented under the FreeBSD 2.2.7 operating system running on Intel x86 platforms. FreeBSD 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 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 a actively mobile ad hoc network under realistic communication workloads. The last week of February and the first week of March 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 [19].

The software was ported to FreeBSD 3.3, and a preliminary version of Quality of Service (QoS) support was added. A demonstration of this modified version of DSR was presented in July 2000. Those 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.

The DSR protocol has been extensively studied using simulation; we have implemented DSR in the ns-2 simulator [5, 18] and conducted evaluations of different caching strategies documented in this draft [9].

Several 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.

Acknowledgements

The protocol described in this draft has been designed and developed within the Monarch Project, a research project at Rice University and Carnegie Mellon University which is developing adaptive networking protocols and protocol interfaces to allow truly seamless wireless and mobile node networking [14, 6].

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.

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] Carnegie Mellon University Monarch Project. CMU Monarch Project Home Page. Available at http://www.monarch.cs.cmu.edu/.

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

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

[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] 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.

[11] 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.

[12] 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.

[13] 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.

[14] David B. Johnson and David A. Maltz. Protocols for adaptive wireless and mobile networking. IEEE Personal Communications, 3(1):34--42, February 1996.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

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

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

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

[27] Paul Turner. NetWare communications processes. NetWare Application Notes, Novell Research, pages 25--91, September 1990.

Chair's Address

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

M. Scott Corson
Institute for Systems Research
University of Maryland
College Park, MD 20742
USA

Phone: +1 301 405-6630
Email: corson@isr.umd.edu

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

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

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

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