IETF MANET Working Group
INTERNET-DRAFT

Josh Broch
David B. Johnson
David A. Maltz
Carnegie Mellon University
13 March 1998

The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks

<draft-ietf-manet-dsr-00.txt>

Status of This Memo

Abstract

Dynamic Source Routing (DSR) is a routing protocol designed specifically for use in mobile ad hoc networks. The protocol allows nodes to dynamically discover a source route across multiple network hops to any destination in the ad hoc network. When using source routing, each packet to be routed carries in its header the complete, ordered list of nodes through which the packet must pass. A key advantage of source routing is that intermediate hops do not need to maintain routing information in order to route the packets they receive, since the packets themselves already contain all of the necessary routing information. This, coupled with the dynamic, on-demand nature of Route Discovery, completely eliminates the need for periodic router advertisements and link status packets, reducing the overhead of DSR, especially during periods when the network topology is stable and these packets serve only as keep-alives.

Contents

1. Introduction

This document describes Dynamic Source Routing (DSR) [6, 7], a protocol developed by the Monarch Project [8, 14] at Carnegie Mellon University for routing packets in a mobile ad hoc network [3].

Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet; the sender explicitly lists this route in the packet's header, identifying each forwarding "hop" by the address of the next node to which to transmit the packet on its way to the destination host.

DSR offers a number of potential advantages over other routing protocols for mobile ad hoc networks. First, DSR uses no periodic routing messages (e.g., no router advertisements and no link-level neighbor status messages), thereby reducing network bandwidth overhead, conserving battery power, and avoiding the propagation of potentially large routing updates throughout the ad hoc network. Our Dynamic Source Routing protocol is able to adapt quickly to changes such as host movement, yet requires no routing protocol overhead during periods in which no such changes occur.

In addition, DSR has been designed to compute correct routes in the presence of asymmetric (uni-directional) links. In wireless networks, links may at times operate asymmetrically due to sources of interference, differing radio or antenna capabilities, or the intentional use of asymmetric communication technology such as satellites. Due to the existence of asymmetric links, traditional link-state or distance vector protocols may compute routes that do not work. DSR, however, will find a correct route even in the presence of asymmetric links.

2. Assumptions

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

We refer to the minimum number of hops necessary for a packet to reach from any host located at one extreme edge of the network to another host located at the opposite extreme, as the diameter of the network. We assume that the diameter of an ad hoc network will 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. A host receiving a corrupted packet can detect the error and discard the packet.

We assume that hosts can enable a promiscuous receive mode on their wireless network interface hardware, causing 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, it is for example common in current LAN hardware for broadcast media including wireless, and some of our optimizations take advantage of it if available. 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 believe that portions of the protocol are also suitable for implementation directly within a programmable network interface unit to avoid this overhead on the CPU.

3. Terminology

3.1. General Terms

node

A device that implements IP.

router

A node that forwards IP packets not explicitly addressed to itself.

host

Any node that is not a router.

link

A communication facility or medium over which nodes can communicate at the link layer, such as an Ethernet (simple or bridged). A link is the layer immediately below IP.

interface

A node's attachment to a link.

prefix

A bit string that consists of some number of initial bits of an

address.

interface index

An 8-bit quantity which uniquely identifies an interface among a given node's interfaces.

link-layer address

A link-layer identifier for an interface, such as IEEE 802 addresses on Ethernet links.

packet

An IP header plus payload.

home address

An IP address that is assigned for an extended period of time to a mobile node. It remains unchanged regardless of where the node is attached to the Internet [9]. If a node has more than one home address, it SHOULD select and use a single home address when participating in the ad hoc network.

source route

A source route from node A to node B is an ordered list of home addresses, starting with the home address of node A and ending with the home address of the node B. Between A and B, the source route includes an ordered list of all the intermediate hops between A and B, as well as the interface index of the interface through which the packet should be transmitted to reach the next hop. Note that the packet formats defined in Section 5.1 encode the Target Address (node B) separately, instead of encoding it as the last hop on the source route.

Route Discovery

The method in DSR by which a node A dynamically obtains a source route to node B that will carry packets through the network from A to B. Performing a route discovery involves sending one or more Route Request packets.

Route Maintenance

The process in DSR of monitoring the status of a source route while in use, so that any link-failures along the source route can be detected and the broken source route removed from use.

3.2. Specification Language

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 [2].

4. Protocol Overview

4.1. Route Discovery and Route Maintenance

A source routing protocol must solve two challenges, which DSR terms Route Discovery and Route Maintenance. Route Discovery is the mechanism whereby a node S wishing to send a packet to a destination D obtains a source route to D.

Route Maintenance is the mechanism whereby 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 hop 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.

To perform Route Discovery, the source node S broadcasts a Route Request packet with a recorded source route listing only itself. Each node that hears the Route Request forwards the Request if appropriate, adding its own address to the recorded source route in this copy of the Request and rebroadcasts the packet. The forwarding of Requests is constructed so that copies of the Request propagate hop-by-hop outward from the node initiating the Route Discovery, until either the target of the Request is found or until another node is found that can supply a route to the target.

The basic mechanism of forwarding Route Requests forwards the Request if the node (1) is not the target of the Request and (2) is not already listed in the recorded source route in this copy of the Request. In addition, however, each node maintains an LRU cache of recently received Route Requests and does not propagate any copies of a Request after the first, avoiding the overhead of forwarding additional copies that reach this node along different paths. Also, the Time-to-Live field in the IP header of the packet carrying the Route Request MAY be used to limit the scope over which the Request will propagate, using the normal behavior of Time-to-Live defined by IP [12, 1]. Additional optimizations on the handling and forwarding of Route Requests are also used to further reduce the Route Discovery overhead. When the target of the Request (e.g., node D) receives the Route Request, it copies the recorded source route into a Route Reply packet which it then sends this Reply back to the initiator of the Route Request (e.g., node S).

All source routes learned by a node are kept in a Route Cache, which is used to further reduce the cost of Route Discovery. When a node wishes to send a packet, it examines its own Route Cache and performs Route Discovery only if no suitable source route is found in its Cache.

Further, when a node B receives a Route Request from S for another node D, B searches its own Route Cache for a route to D. If B finds such a route, it does not propagate the Route Request, but instead returns a Route Reply to node S based on the concatenation of the recorded source route from S to B in the Route Request and the cached route from B to D. The details of replying from a Route Cache in this way are discussed in Section 7.1.

As a node overhears routes being used by others, either by promiscuously snooping on them or when forwarding packets, the node MAY insert those routes into its Route Cache, leveraging the Route Discovery operations of the other nodes.

4.2. Packet Forwarding

To represent a source route within a packet's header, DSR uses a Routing Header that conforms to the Routing Header format specified for IPv6, adapted to the needs of DSR and to the use of the DSR in IPv4 (or in IPv6 in the future). The DSR Routing Header uses a unique Routing Type field value to distinguish it from the existing Type 0 Routing Header defined within IPv6 [4].

To forward a packet, a receiving node N simply processes the Routing Header as specified in the IPv6 [4] and transmits the packet to the next hop. If a forwarding error occurs along the link to the next hop in the route, this node N sends a Route Error back to the originator S of the packet informing S that this link is "broken". If node N's Route Cache contains a different route to the destination, then the packet is retransmitted using the new source route. Each node overhearing or forwarding a Route Error packet also removes from its Route Cache the link indicated to be broken, thereby cleaning the stale cache data from the network.

4.3. Conceptual Data Structures

All information a node needs for participation in an ad hoc network using the Dynamic Source Routing Protocol can be organized conceptually into four data structures: a Route Cache, a Node Information Cache, a Send Buffer, and a Retransmission Buffer. These data structures MAY be implemented in any manner consistent with the external behavior described in this document.

4.3.1. Route Cache

All routing information needed by a node participating in an ad hoc network is stored in a Route Cache. Each node in the network

maintains its own Route Cache. The node adds information to the

cache as it learns of new links between nodes in the ad hoc network, for example through packets carrying either a Route Reply or a Routing Header. Likewise, the node removes information from the cache as it learns existing links in the ad hoc network have broken, for example through packets carrying a Route Error or through the link-layer retransmission mechanism reporting a failure in forwarding a packet to its next-hop destination. The Route Cache is indexed logically by destination node, and supports the following operations:

void Insert(Route RT)

Information extracted from source route RT is inserted into the Route Cache.

Route Get(Node DEST)

A source route from this node to DEST (if it exists) is returned.

void Delete(Node FROM, Node TO)

Any routes in the cache that assume the existence of a unidirectional link from node FROM to node TO are removed from the cache.

Each implementation MAY choose the cache replacement and cache search strategies most appropriate for its particular network environment. For example, some environments may choose to return the shortest route to a node (the shortest sequence of hops), while others may select an alternate metric for the Get() operation.

The Route Cache SHOULD support storing more than one source route for each destination.

If node S is using a source route to destination D that includes intermediate node I, S SHOULD shorten the route to destination D when it learns of a shorter route to node I. A node S using a source route to destination D through node I, MAY shorten the source route if it learns of a shorter path from node I to node D.

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. In particular, a node SHOULD prefer routes that it is presently using over those that it is not.

The Route Cache SHOULD time-stamp each route as it is inserted into the cache. If the route is not used within ROUTE_CACHE_TIMEOUT seconds, it SHOULD be removed from the cache.

4.3.2. Node Information Cache

The Node Information Cache is a collection of records indexed by home address. A record maintained on node N1 for node N2 contains the following:

- The time that N1 last began a Route Discovery for N2.
- The interval of time that N1 must wait before the next attempt at a Route Discovery for N2.
- The Time-to-live (TTL) field in the IP header of last Route Request transmitted by N1 for N2.
- A FIFO cache of the last ID_FIFO_SIZE Identification values observed in Route Request packets initiated by N2.

Nodes SHOULD use an LRU policy to manage the entries of the Node Information Cache.

4.3.3. Send Buffer

The Send Buffer is a queue of packets that cannot be transmitted because the transmitting node does not yet have a source route to the packets' destinations. Each packet in the Send Buffer is stamped with the time that it is placed into the Buffer, and SHOULD be removed from the Send Buffer and 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 6.1, a Route Discovery SHOULD be initiated as often as possible for any packets residing in the Send Buffer.

4.3.4. Retransmission Buffer

The Retransmission Buffer is a queue of packets that are awaiting the receipt of an explicit acknowledgment from the next hop in the source route (Section 5.2).

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.

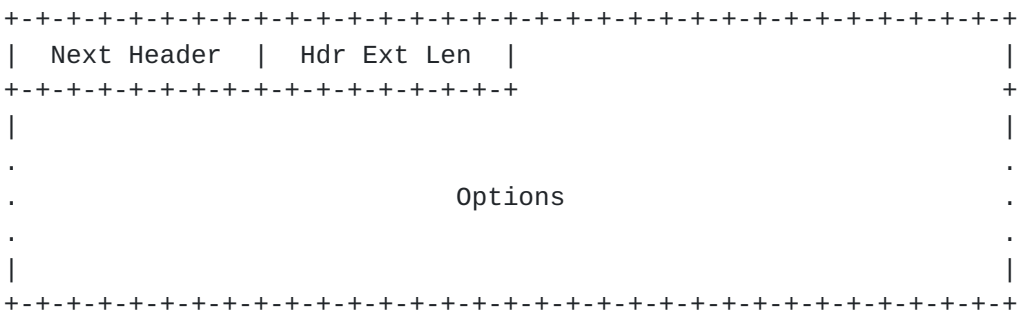
Packets are removed from the buffer when an acknowledgment is received, or when the number of retransmissions exceeds MAX_EXPLICIT_REXMIT. In the later case, the removal of the packet from the Retransmission Buffer should result in a Route Error being returned to the initial source of the packet (Section 6.2).

5. Packet Formats

5.1. Destination Options Headers

Dynamic Source Routing makes use of four options carrying control information that can be piggybacked in any existing IP packet. The mechanism used for these options is based on the design of the Destination Option mechanism in IPv6 [4]. This notion of a Destination Option must be build in to a IPv4 protocol stack. Specifically, the Protocol field in the IP header should be used to indicate that a Destination Options header exists between the IP header and the remaining portion of a packet's payload (such as a transport layer header). The Next Header field in the Destination Options header will then indicate the type of header that follows it in a packet.

The Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header, and has the following format:



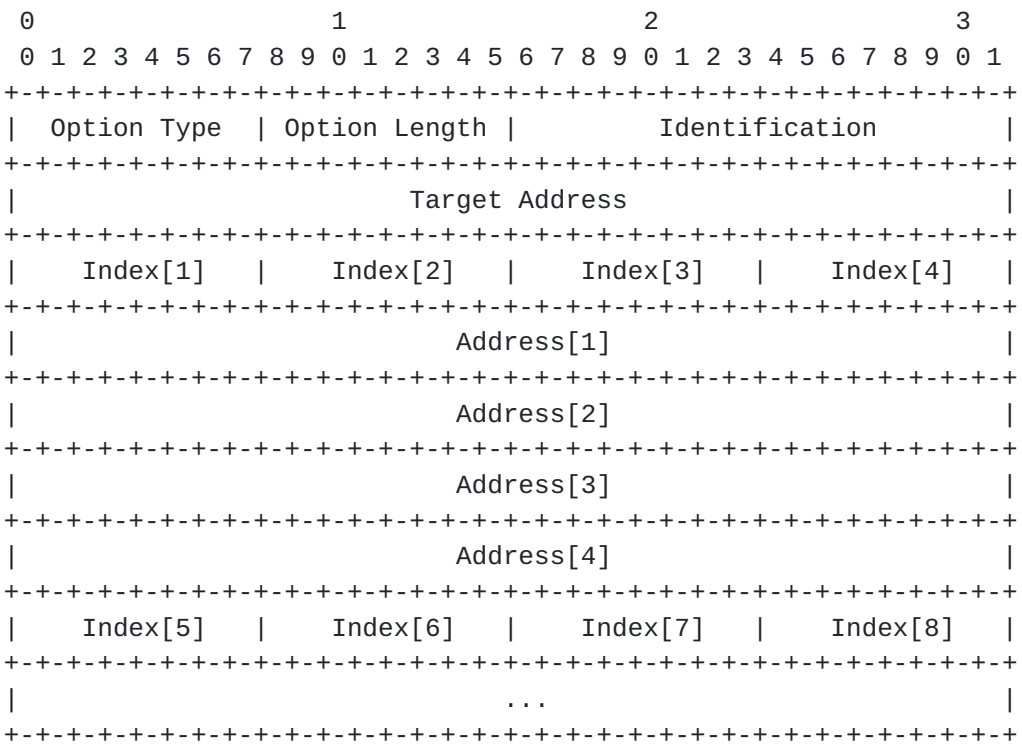
The following destination options are used by the Dynamic Source Routing protocol:

- DSR Route Request option (Section 5.1.1)
- DSR Route Reply option (Section 5.1.2)
- DSR Route Error option (Section 5.1.3)
- DSR Acknowledgement option (Section 5.1.4)

All of these destination options MAY appear multiple times within a single Destination Options header.

5.1.1. DSR Route Request Option

The DSR Route Request destination option is encoded in type-length-value (TLV) format as follows:



IP fields:

Source Address

MUST be the home address of the node transmitting this packet.

Destination Address

MUST be the limited broadcast address (255.255.255.255).

Hop Limit (TTL)

Can be varied from 1 to 255, for example to implement expanding-ring searches.

Route Request fields:

Option Type

???. A node that does not understand this option MUST discard the packet (the top two bits must be 01).

Option Length

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

Identification

A unique value generated by the initiator (original sender) of the Route Request. This value allows a recipient to determine whether or not it has recently seen this a copy of this Request; if it has, the packet is simply discarded. When propagating a Route Request, this field MUST be copied from the received copy of the Request being forwarded.

Target Address

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

Index[1..n]

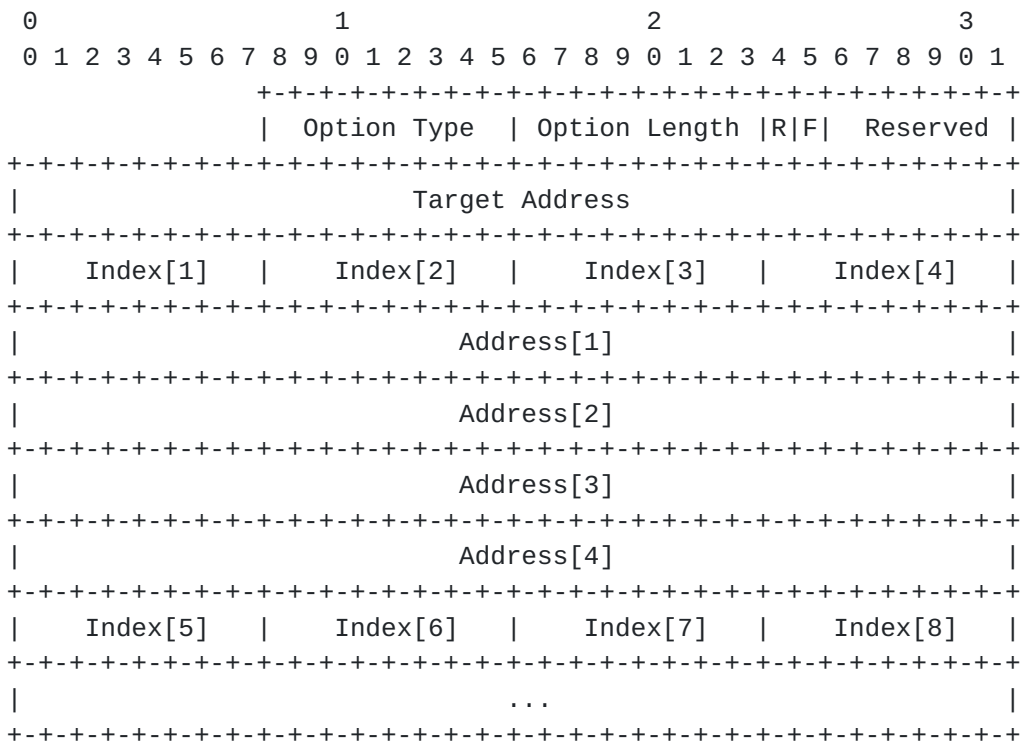
Index[i] is the interface index of the ith hop recorded in in the Route Request option (in Address[i]).

Address[1..n]

Address[i] is the home address of the ith hop recorded in the Route Request option.

5.1.2. DSR Route Reply Option

The DSR Route Reply destination option is encoded in type-length-value (TLV) format as follows:



Option Type

???. A node that does not understand this option should ignore this option and continue processing the packet (the top two bits should be 00).

Option Length

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

Router (R)

If the Router (R) bit is set, the last address recorded in this header is the home address of a router that believes it can reach the Target Address specified in the Route Request packet.

Foreign Agent (F)

If the Foreign Agent (F) bit is set, the last address recorded in this header is the home address of an IETF Mobile IP [9] Foreign Agent. The Router (R) bit and the Foreign Agent (F)

bit are mutually exclusive as (F) implies (R).

Reserved

Sent as 0; ignored on reception.

Target Address

The home address of the node that is the ultimate destination of the source route contained in the Route Reply.

Index[1..n]

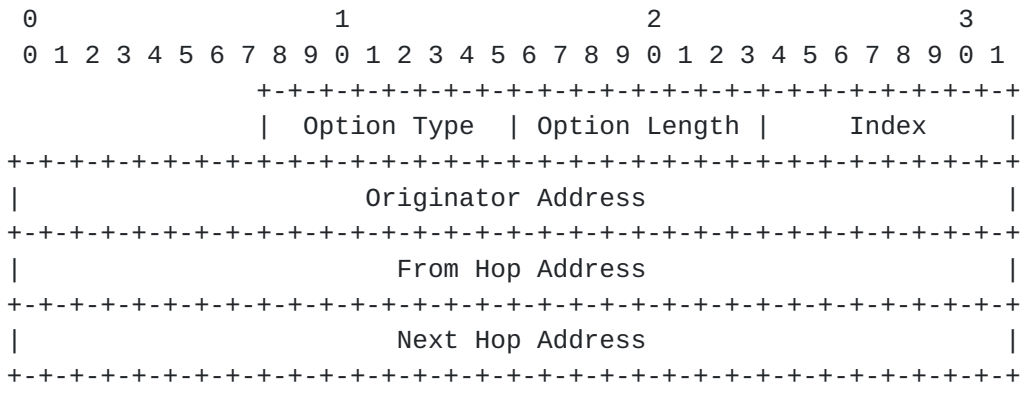
Index[i] is the interface index of the ith hop listed in the Route Reply option (in Address[i]).

Address[1..n]

Address[i] is the home address of the ith hop listed in the Route Reply option.

5.1.3. DSR Route Error Option

The DSR Route Error destination option is encoded in type-length-value (TLV) format as follows:



Option Type

???. A node that does not understand this option should ignore the option and continue processing the packet (the top two bits must be 00).

Option Length

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

Index

The interface index of the network interface over which the link from the From Hop Address node to the Next Hop Node is being reported as broken by this Route Error option. This Index refers to an interface on the From Hop Address node.

Originating Address

The home address of the node which originated the packet that could not be forwarded.

From Hop Address

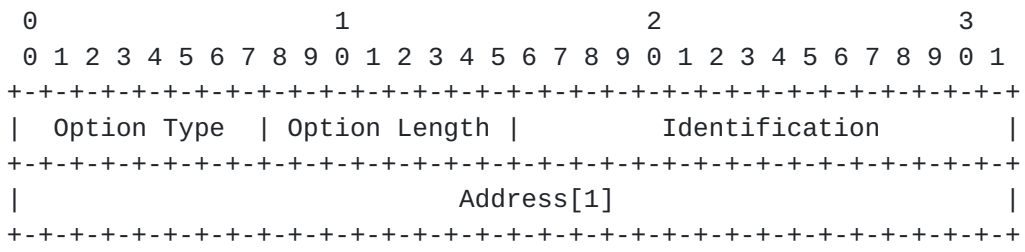
The home address of the node that attempted to forward a packet and discovered the link failure.

Next Hop Address

The home address of the node that was found to be unreachable (the next hop neighbor to which the node at Originating Address was attempting to transmit the packet).

5.1.4. DSR Acknowledgment Option

The DSR Acknowledgment destination option is encoded in type-length-value (TLV) format as follows:



Option Type

???. A node that does not understand this option should ignore the option and continue processing the packet (the top two bits must be 00).

Option Length

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

Identification

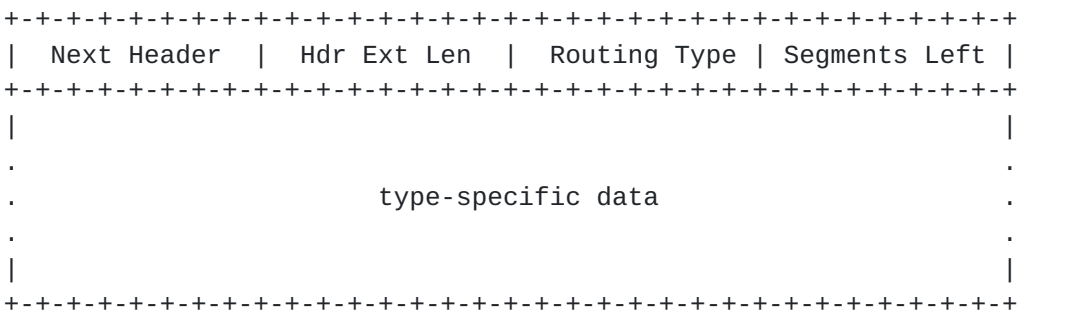
A unique value assigned by the originator of the packet. This value is used to match explicit acknowledgments to the corresponding packet.

Address[1]

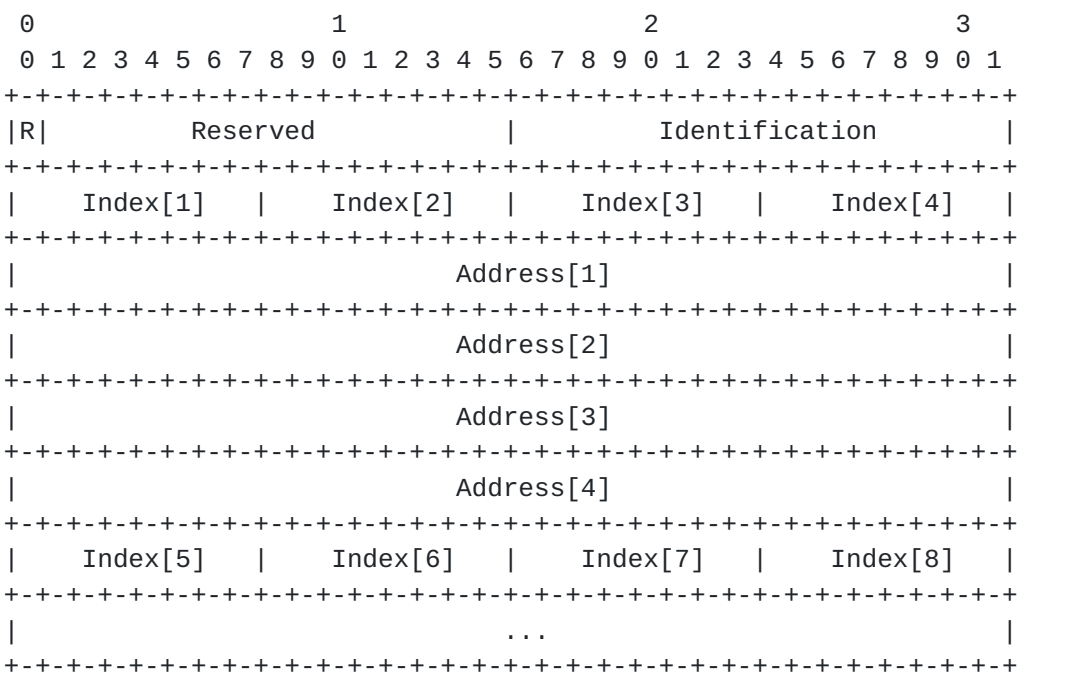
The home address of the original source of the IP packet.

5.2. DSR Routing Header

As specified for IPv6 [4], a Routing header is used by a source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is similar to IPv4's Loose Source and Record Route option, but the Routing header does not record the route taken as the packet is forwarded. The specific processing steps required to implement the Routing header must be added to an IPv4 protocol stack. The Routing header is identified by a Next Header value of 43 in the immediately preceding header, and has the following format:



The type specific data for a Routing Header carrying a DSR Source Route is:



Routing Header Fields:

Next Header

8-bit selector. Identifies the type of header immediately following the Routing Request header.

Hdr Ext Len

8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets.

Routing Type

???

Segments Left

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

Type Specific Fields:

Acknowledgment Request (R)

The Acknowledgment Request (R) bit is set to request an explicit acknowledgment from the next hop.

Reserved

Sent as 0; ignored on reception.

Identification

A unique value assigned by the originator of the packet. This value is used to match acknowledgments (passive or explicit) to the appropriate packet.

Index[1..n]

Index[i] is the interface index of the ith hop in the Routing header.

Address[1..n]

Address[i] is the home address of the ith hop in the Routing header.

6. Detailed Operation

6.1. Route Discovery

Route Discovery is the demand-driven process by which nodes actively obtain source routes to destinations to which they are actively attempting to send packets. The destination node for which a Route Discovery is initiated to discover a route is known as the "target" of the Route Discovery. A Route Discovery for a destination SHOULD NOT be initiated unless the initiating node has an unexpired packet to be delivered to that destination.

A Route Discovery for a given target node MUST NOT be initiated unless the difference between the current time and the time that a Route Discovery was last initiated for destination D is greater than the backoff interval currently listed in the Node Information Cache for node D. After each Route Discovery attempt, the interval between successive Route Discoverys must be doubled, up to a maximum of MAX_RTDISCOV_INTERVAL.

The basic Route Discovery algorithm is to originate a single Route Request packet as described below that targets the desired destination and has a maximum hop limit set to MAX_ROUTE_LEN.

6.1.1. Originating a Route Request

A node originates a Route Request for a particular host when it has no route to that host. The Option Length field in the Route Request option MUST be set to 6, the Identification field MUST be set to a unique number, and the Target Address field MUST contain the Home Address of the node for which a route is being requested.

6.1.2. Processing a Route Request Option

Let P1 be the received packet containing a Route Request option. Let P2 be a packet containing a corresponding Route Reply. A Route Request option is processed as follows:

1. Determine the originator of the Route Request.

   If no addresses are presently listed in P1.REQUEST.Address[], then P1.Source_Address identifies the originator of the Route Request. Otherwise, P1.REQUEST.Address[1] identifies the originator of the Route Request.

2. If the combination (Originator Address, P1.REQUEST.Identification) is in the node's cache of recently seen (Address, Identification)

pairs, then discard the packet. DONE.

3. If the home address of this node is already listed in P1.REQUEST.Address[], then discard the packet. DONE.

4. If P1.REQUEST.Target_Address matches the home address of this node, then this packet contains a complete source route describing the path from the initiator of the Route Request to this node.

   (a) Send a Route Reply as described in Section 6.1.3.

   (b) If P1.REQUEST.Next_Header indicates No Next Header, DONE.

   (c) Otherwise, swap P1.REQUEST.Target_Address and P1.Source_Address and pass the packet up the protocol stack. DONE.

5. Set P1.REQUEST.Address[n+1] = home address of this node. Re-broadcast the Route Request packet jittered by T milliseconds, where T is a uniformly distributed, random number between 0 and BROADCAST_JITTER. DONE.

6.1.3. Originating a Route Reply

Let P1 be the received packet containing a Route Request option. Let P2 be a packet containing a corresponding Route Reply. A Route Reply is transmitted in response to a Route Request as follows:

1. If P1.REQUEST.Address[] does not contain any hops, then this node is only a single hop from the originator of the Route Request. Build a Route Replay packet as follows:

```
P2.Destination_Address = P1.Source_Address
P2.Source_Address = P1.REQUEST.Target_Address
```

   GOTO 3.

2. Otherwise, build a Route Reply packet as follows:

```
P2.Destination_Address = P1.REQUEST.Address[1]
P2.Source_Address = P1.REQUEST.Target_Address
P2.REPLY.Address[1..n] = P1.REQUEST.Address[1..n]
```

3. Transmit the Route Reply jittered by T milliseconds, where T is a uniformly distributed, random number between 0 and BROADCAST_JITTER. DONE.

If sending a Route Reply packet to the originator of the Route Request requires performing a Route Discovery, the Route Reply

destination option MUST be piggybacked on the packet that contains

the Route Request. This prevents a loop wherein the target of the 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, nodes SHOULD send a unicast Route Reply in response to every Route Request targeted at them.

6.1.4. Processing a Route Reply Option

Upon receipt of a Route Reply, a node should extract the source route (Address[1..n] + Target Address) and insert this route into its Route Cache. Any packets in the Send Buffer that are addressed to Target Address SHOULD be processed.

6.2. Route Maintenance

6.2.1. Originating a Route Error

If while forwarding a packet with a Routing Header, the next hop specified in the source route is found to be unreachable, a Route Error packet (Section 5.1.3) MUST be returned to the originator (Address[1]) of the packet.

The forwarding node SHOULD consider the next hop to be unreachable if any of the following conditions occurs:

- The failure to receive a passive acknowledgment when such passive acknowledgments had been received previously.
- The failure to receive an explicitly requested acknowledgment after MAX_EXPLICIT_REXMIT retransmissions.
- In link layers providing retransmissions and acknowledgments (e.g., 802.11), a signal from the link layer that it is unable to deliver the packet.

6.2.2. Processing a Route Error Option

Upon receipt of a Route Error via any mechanism, a node SHOULD remove any route from its Route Cache that uses the hop (From Hop Address, Next Hop Address).

When the Route Error is returned to the Originator Address, the

originator must verify that the source route in the Route Error

packet (From Hop Address...Originator Address) includes the same hops as the working prefix of the original packet's source route (Originator Address...From Hop Address). If any hop listed in the working prefix is not included in the Route Error's source route, then the originator must transmit the Route Error back along the working prefix (Originator Address...From Hop Address) so that each node along the working prefix will remove the invalid route from its Route Cache.

If the node processing a Route Error option discovers its home address equals the Router Error's Originator Address and the packet contains an additional nested Route Error, the node MUST perform the following steps:

1. Remove the Route Error being processed from the packet.

2. Copy the Originator Address from the next nested Route Error to the IP destination field of the packet.

3. Attach a source route and send the packet to the IP destination, performing Route Discovery if needed.

6.2.3. Processing a DSR Acknowledgment Option

Upon receipt of a DSR Acknowledgment, a node should remove any packet in its Retransmission Buffer matching the (Address, Identification) pair found in the Acknowledgment option. If no match is found, the Acknowledgment should be silently discarded.

[I'm supposed to say something intelligent here, but I can't remember what... -josh]

6.3. Processing a Routing Header

A DSR Routing Header should be processed in accordance with the steps outlined for Routing Headers in [4]. The Routing Header is only processed by the node whose address appears as the IP destination of the packet. A few additional rules apply to processing the type specific data of a DSR Source Route:

1. The interface used to transmit the packet MUST be the interface denoted by Index[n] where Address[n] is the home address of this node.

2. If the Acknowledgment Request (R) bit is set, the node MUST create and transmit a packet containing the DSR Acknowledgment option to the IP Source of the packet, performing Route Discovery

if necessary.

3. If the node chooses to set the Acknowledgment Request (R) bit in the packet when it forwards it, it must first make a copy of the packet and insert this copy into its Retransmission Buffer.

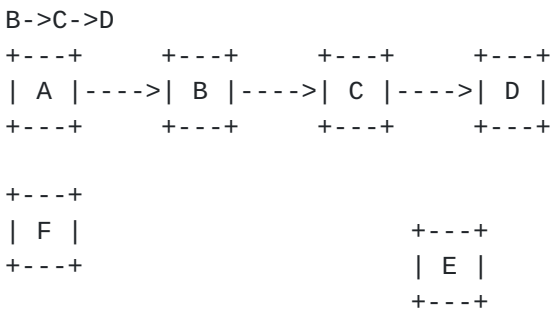
4. If a node finds the next hop in the Routing Header to be unreachable, it MUST send a Route Error packet to the originator of the packet, denoted by ROUTING.Address[1].

7. Optimizations

A number of optimizations can be added to the basic operation of Route Discovery and route maintenance as described in Section 4.1 that can reduce the number of overhead packets and improve the average efficiency of the routes used on data packets. This section discusses some of those optimizations.

7.1. Leveraging the Route Cache

The data in a node's Route Cache may be stored in any format, but the active routes in its cache form a tree of routes, rooted at this node, to other nodes in the ad hoc network. For example, the illustration below shows an ad hoc network of six mobile nodes, in which mobile node A has earlier completed a Route Discovery for mobile node D and has cached a route to D through B and C:



Since nodes B and C are on the route to D, node A also learns the route to both of these nodes from its Route Discovery for D. If A later performs a Route Discovery and learns the route to E through B and C, it can represent this in its Route Cache with the addition of the single new hop from C to E. If A then learns it can reach C in a single hop (without needing to go through B), A SHOULD use this new route to C to also shorten the routes to D and E in its Route Cache.

7.1.1. Promiscuous Learning of Source Routes

A node can add entries to its Route Cache any time it learns a new route. In particular, when a node forwards a data packet as an intermediate hop on the route in that packet, the forwarding node is able to observe the entire route in the packet. Thus, for example, when node B forwards packets from A to D, B SHOULD add the route information from that packet to its own Route Cache. If a node forwards a Route Reply packet, it SHOULD also add the route information from the route record being returned in the Route Reply, to its own Route Cache.

Finally, since all wireless network transmissions are inherently broadcast, a node MAY configure its network interface into promiscuous receive mode, and add to its Route Cache the route information from any packet it can overhear.

7.1.2. Answering Route Requests using the Route Cache

A node SHOULD use its Route Cache to avoid propagating a Route Request packet received from another node. In particular, suppose a node receives a Route Request packet for which it is not the target and which it does not discard on based on the logic of section 6.1.1. If the node has a Route Cache entry for the target of the request, it may append this cached route to the accumulated route record in the packet, and may return this route in a Route Reply packet to the initiator without propagating (re-broadcasting) the Route Request. Thus, for example, if node F in the example network shown in Section 7.1 needs to send a packet to node D, it will initiate a Route Discovery and broadcast a Route Request packet. If this broadcast is received by A, A can simply return a Route Reply packet to F containing the complete route to D consisting of the sequence of hops A, B, C, and D.

Before transmitting a Route Reply packet that was generated using information from its Route Cache, a node MUST verify that:

1. The resulting route does not contain any loops.

2. The node issuing the Route Reply is listed in the route that it is replying with. This increases the probability that the route is valid, since the node in question should have received a Route Error if this route stopped working.

7.2. Route Discovery Using Expanding Ring Search

The propagating nature of a basic Route Request packet means that potentially every node in the ad hoc network will be disturbed whenever one is originated. To reduce this network-wide cost, all nodes SHOULD limit the maximum propagation of their Route Requests in some way, and MAY use the following algorithm.

1. Whenever the backoff algorithm permits the initiation of a Route Discovery, initially send a Route Request with a hop limit of one (we refer to this as a non-propagating Route Request).

2. If no Route Reply is received from the non-propagating Route Request after RING0_TIMEOUT seconds, send a new Route Request with the hop limit set to MAX_ROUTE_LEN.

A single attempt at Route Discovery for destination node D may therefore involve sending two Route Request packets. Nodes MUST not backoff between the sending a Route Request with a hop limit of one and the subsequent sending of Route Request with a hop limit of MAX_ROUTE_LEN. This procedure uses the hop limit on the Route Request packet to inexpensively check if the target is currently within wireless transmitter range of the initiator, or if another node within range has a Route Cache entry for this target (effectively using the caches of this node's neighbors as an extension of its own cache). Since the initial request is limited to one network hop, the timeout period before sending the propagating request can be quite small.

7.3. Preventing Route Reply Storms

The ability for nodes to reply to a Route Request not targeted at them using their Route Caches can result in a Route Reply storm. If a node broadcasts a Route Request for a node that its neighbors have in their Route Caches, each neighbor may attempt to send a Route Reply thereby wasting bandwidth and increasing the rate of collisions in the area. For example, in the network shown in Section 7.1, if both A and B receive F's Route Request, they will both attempt to reply from their Route Caches. Both will send their replies at about the same time since they receive the broadcast at about the same time. Particularly when more than the two mobile nodes in this example are involved, these simultaneous replies from the mobile nodes receiving the broadcast 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. For example, A's reply will indicate a route to D that is one hop longer than that in B's reply.

For interfaces which can promiscuously listen to the channel, mobile nodes SHOULD use the following algorithm to reduce the number of simultaneous replies by slightly delaying their Route Reply:

1. Pick a delay 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 reply, r is a random number between 0 and 1, and H is a small constant delay to be introduced per hop.

2. Delay transmitting the Route Reply from this node for a period of d.

3. Within the delay period, promiscuously receive all packets at this node. If a packet is received by this node during the delay period that is addressed to the target of this Route Discovery (the target is the final destination address for the packet, through any sequence of intermediate hops), and if the length of the route on this packet is less than h, then cancel the delay and do not transmit the Route Reply from this node; this node may infer that the initiator of this Route Discovery has already received a Route Reply, giving an equal or better route.

7.4. Piggybacking on Route Discoveries

As described in Section 4.1, when one node needs to send a packet to another, if the sender does not have a Route Cached to the destination node, it must initiate a Route Discovery, either buffering the original packet until the Route Reply is returned, or discarding it and relying on a higher-layer protocol to retransmit it if needed. The delay for Route Discovery and the total number of packets transmitted can be reduced by allowing data to be piggybacked on Route Request packets. Since some Route Requests may be propagated widely within the ad hoc network, though, the amount of data piggybacked must be limited. We currently use piggybacking when sending a Route Reply or a Route Error packet, since both are naturally small in size, and small data packets such as the initial SYN packet opening a TCP connection [13] could easily be piggybacked.

One problem, however, arises when piggybacking on Route Request packets. If a Route Request is received by a node that replies to the request based on its Route Cache without propagating the request (Section 7.1), the piggybacked data will be lost if the node simply discards the Route Request. In this case, before discarding the packet, the node must construct a new packet containing the piggybacked data from the Route Request packet. The source route in this packet MUST be constructed to appear as if the new packet had been sent by the initiator of the Route Discovery and had been forwarded normally to this node. Hence, the first portion of the route is taken from the accumulated route record in the Route Request packet and the remainder of the route is taken from this node's Route Cache. The sender address in the packet should also be set to the initiator of the Route Discovery. Since the replying node will be unable to correctly recompute an Authentication header for the split off piggybacked data, data covered by an Authentication header SHOULD NOT be piggybacked on Route Request packets.

7.5. Discovering Shorter Routes

Once a route between a packet source and a destination has been discovered, the basic DSR protocol MAY continue to use that route for

all traffic from the source to the destination, even if the nodes move such that a shorter route becomes possible. In many cases, the basic route maintenance procedure will discover the shorter route, since if a node moves enough to create a shorter route, it will likely also move out of transmission range of at least one hop on the existing route.

When operating in promiscuous receive mode, a node SHOULD use the following algorithm to process a received packet. Whenever possible, this algorithm shortens routes that already exist in the Route Cache.

1. If the packet is not a data packet containing a Routing Header, drop the packet. DONE.

2. If the IP destination is the home address of this node, then follow the normal steps to process the packet. DONE.

3. If the home address of this node does not appear in the portion of the source route that has not yet been processed (indicated by Segments Left), then drop the packet. DONE.

4. The node S indicated by the Source Address field in the IP header can communicated directly with this node N. Create a Route Reply. The Route Reply MUST list the entire source routing contained in the received packet with the exception of the intermediate nodes between node S and node N.

7.6. Rate Limiting the Route Discovery Process

One common error condition that must be handled in an ad hoc network is the case in which the network effectively becomes partitioned. That is, two nodes that wish to communicate are not within transmission range of each other, and there are not enough other mobile nodes between them to form a sequence of hops through which they can forward packets. If a new Route Discovery was initiated for each packet sent by a node in this situation, 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, we use exponential backoff to limit the rate at which new route discoveries may be initiated from any node for the same target. If the node attempts to send additional data packets to this same node more frequently than this limit, the subsequent packets SHOULD be buffered in the Send Buffer until a Route Reply is received, but it 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 to any single IP address [1].

7.7. Improved Handling of Route Errors

All nodes SHOULD process all of the Route Error messages they receive, regardless of whether the node is the destination of the Route Error, is forwarding the Route Error, or promiscuously overhears the Route Error.

Since a Route Error packet names both ends of the hop that is no longer valid, any of the nodes receiving the error packet may update their Route Caches to reflect the fact that the two nodes indicated in the packet can no longer directly communicate. A node receiving a Route Error packet simply searches its Route Cache for any routes using this hop. For each such route found, the route is truncated at this hop. All nodes on the route before this hop are still reachable on this route, but subsequent nodes are not.

An experimental optimization to improve the handling of errors is to support the caching of "negative" information in a node's Route Cache. The goal of negative information is to record that a given route was tried and found not to work, so that if the same route is discovered again shortly after the failure, the Route Cache can ignore or downgrade the metric of the failed route.

We have not currently included this caching of negative information in our simulations, since it appears to be unnecessary if nodes also promiscuously receive Route Error packets.

8. Constants

BROADCAST_JITTER	10	milliseconds
ID_FIFO_SIZE	8	identifiers
INVALID_INTERFACE_INDEX	0xFF	
MAX_EXPLICIT_REXMIT	3	attempts
MAX_RTDISCOV_INTERVAL	120	seconds
MAX_ROUTE_LEN	15	nodes
RING0_TIMEOUT	30	milliseconds
ROUTE_CACHE_TIMEOUT	300	seconds
SEND_BUFFER_TIMEOUT	30	seconds

9. IANA Considerations

This document defines four new types of IPv6 destination option, each of which must be assigned an Option Type value:

- The DSR Route Request option, described in Section 5.1.1
- The DSR Route Reply option, described in Section 5.1.2
- The DSR Route Error option, described in Section 5.1.3
- The DSR Acknowledgment option, described in Section 5.1.4

DSR also requires a routing header Routing Type be allocated for the DSR Source Route defined in section 5.2.

In IPv4, we require two new protocol numbers be issued to identify the next header as either an IPv6-style destination option, or an IPv6-style routing header. Other protocols can make use of these protocol numbers as nodes that support them will processes any included destination options or routing headers according to the normal IPv6 semantics.

10. 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 with out 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.

Location of DSR Functions in the ISO Model

When designing DSR, we had to determine at what level within the protocol hierarchy to implement source 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 the following 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 IP [12], IPv6 [4], and IPX [5] nodes.

- Historically, DSR grew from our contemplation of a multihop ARP protocol [6, 7] and source routing bridges [10]. ARP [11] is a layer 2protocol.

- Technically, we designed DSR to be simple enough that that it could be implemented directly in network interface cards, well below the layer 3 software within a mobile node. We see great potential for DSR running between clouds of mobile nodes around fixed base stations. DSR would act to transparently fill in the coverage gaps between base stations. 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 design DSR as a layer 3 protocol since this is the only layer at which we could realistically support nodes with multiple interfaces of different types.

Implementation Status

We have implemented Dynamic Source Routing (DSR) under the FreeBSD 2.2.2 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.

Acknowledgments

The protocol described in this draft has been designed within the CMU Monarch Project, a research project at Carnegie Mellon University which is developing adaptive networking protocols and protocol interfaces to allow truly seamless wireless and mobile host networking [8, 14]. The current members of the CMU Monarch Project include:

- Josh Broch
- Yih-Chun Hu
- Jorjeta Jetcheva
- David B. Johnson
- David A. Maltz

Areas for Refinement

We are currently working to refine the DSR protocol in the following ways:

- Improve the algorithms and data structures used by the Route Cache. We currently represent the Route Cache as a directed acyclic tree of paths branching out from a root that represents the node owning the Route Cache. However, each source route learned by the Route Cache effectively describes the interconnectedness of all the hops listed on the route, and can be treated as a type of partial information Link State Packet as one would find in a Link State routing algorithm. By generalizing the Route Cache to a graph of all known links between all known nodes, it may be possible to better leverage the information a node overhears.

- Support for better route selection. In order to select the best source route to send a packet with, nodes need be able to evaluate the costs/benefits of each of their cached routes to the destination. If those routes involve forwarding through nodes with more than one interface, some routes may be better suited to the traffic type because the bandwidth/range/latency/error-rate characteristics of of the interfaces used on those routes best match the needs of the traffic type. The Route Request and Route Reply option format must be extended to enable node to report the properties of the interfaces on the route, as well as the interface index used in basic DSR forwarding.

- Improved Route Discovery algorithms. We are investigating ways to cancel a propagating Route Request if the target of the request has already been found in another part of the network. Similarly, we are studying various ring-search algorithms in case a more sophisticated algorithm might perform better than the 2-step algorithm we currently use.

References

[1] R. Braden, editor. Requirements for Internet Hosts -- Communication Layers. RFC 1122, October 1989.

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

[3] Scott Corson and Joseph Macker. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Internet-Draft, draft-ietf-manet-issues-00.txt, September 1997. Work in progress.

[4] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. Internet-Draft, draft-ietf-ipngwg-ipv6-spec-v2-01.txt, November 1997. Work in progress.

[5] IPX Router Specification. Novell Part Number 107-000029-001, Document Version 1.30, March 1996.

[6] 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.

[7] 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.

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

[9] Charles Perkins, editor. IP Mobility Support. RFC 2002, October 1996.

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

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

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

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

[14] The CMU Monarch Project. http://www.monarch.cs.cmu.edu/.
Computer Science Department, Carnegie Mellon University.

Chair's Address

The 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:

Josh Broch
Carnegie Mellon University
Electrical and Computer Engineering Department
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
USA

Phone: +1 412 268-3056
Email: broch@andrew.cmu.edu

David B. Johnson
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
USA

Phone: +1 412 268-7399
Fax: +1 412 268-5576
Email: dbj@cs.cmu.edu

David A. Maltz
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
USA

Phone: +1 412 268-3621
Fax: +1 412 268-5576
Email: dmaltz@cs.cmu.edu