INTERNET-DRAFT Zygmunt J. Haas, Cornell University
Marc R. Pearlman, Cornell University
Expires in six months August 1998

# The Zone Routing Protocol (ZRP) for Ad Hoc Networks

<draft-ietf-manet-zone-zrp-01.txt>

## Status of this Memo



## Abstract

This document describes the Zone Routing Protocol (ZRP), a hybrid routing protocol suitable for a wide variety of mobile ad-hoc networks, especially those with large network spans and diverse mobility patterns. Each node proactively maintains routes within a local region (referred to as the routing zone). Knowledge of the routing zone topology is leveraged by the ZRP to improve the efficiency of a reactive route query/reply mechanism. The proactive maintenance of routing zones also helps improve the quality of discovered routes, by making them more robust to changes in network topology. The ZRP can be configured for a particular network by proper selection of a single parameter, the routing zone radius.

This version of the ZRP internet draft describes a number of features which have been added to the protocol. The distance-vector based IARP algorithm has been enchanced to prevent the 'counting to infinity problem'. Route caching is now supported by the IERP route discovery process. By allowing discovered route information to be distributed in caches, route accumulation in the IERP query/reply packets can be avoided, thereby reducing the amount of route discovery traffic, and improving the query response time. Two additional (optional) stages have also been added to the route discovery process, to support the acquisition and optimization of

source routes.

Contents

## 1. Introduction

One of the major challenges in designing a routing protocol for the ad hoc networks stems from the fact that, on one hand, to determine a packet route, a node needs to known at least the reachability information to its neighbors. On the other hand, in an ad hoc network, the network topology can change quite often. Furthermore, as the number of network nodes can be large, the potential number of destinations is also large, requiring large and frequent exchange of data (e.g., routes, routes updates, or routing tables) among the network nodes. Thus, the amount of update traffic can be quite high. This is in contradiction with the fact that all updates in a wirelessly interconnected ad hoc network travel over the air and, thus, are costly in resources.

In general, the existing routing protocols can be classified either as proactive or as reactive. Proactive protocols attempt to continuously evaluate the routes within the network, so that when a packet needs to be forwarded, the route is already known and can be immediately used. The family of Distance-Vector protocols is an example of a proactive scheme. Reactive protocols, on the other hand, invoke a route determination procedure on demand only. Thus, when a route is needed, some sort of global search procedure is employed. The family of classical flooding algorithms belong to the reactive group. Some examples of reactive (also called on-demand) ad hoc network routing protocols are [Johnson], [AODV], [TORA] (see also [Park]).

The advantage of the proactive schemes is that, once a route is needed, there is little delay until the route is determined. In reactive protocols, because route information may not be available at the time a datagram is received, the delay to determine a route can be quite significant. Furthermore, the global flood-search procedure of the reactive protocols requires significant control traffic. Because of this long delay and excessive control traffic, pure reactive routing protocols may not be applicable to real-time communication. However, pure proactive schemes are likewise not appropriate for the ad hoc networking environment, as they continuously use a large portion of the network capacity to keep the routing information current. Since nodes in an ad hoc networks move quite fast, and as the changes may be more frequent than the route requests, most of this routing information is never even used! This results in a further waste of the wireless network capacity. What is needed is a protocol that, on one hand, initiates the route-determination procedure on-demand, but at limited search cost. The protocol described in this draft, termed the "Zone Routing Protocol (ZRP)" ([Haas-1], [Haas-2]), is an example of a such a hybrid proactive/reactive scheme.

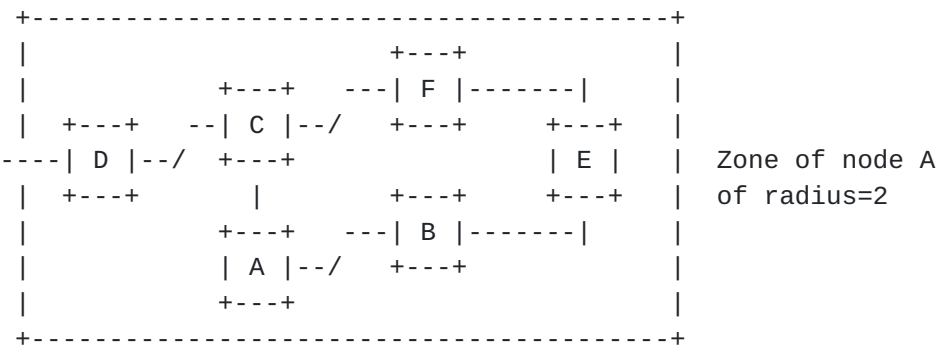
The ZRP, on one hand, limits the scope of the proactive procedure only to the node's local neighborhood. On the other hand, the search throughout the network, although global in nature, is done by efficiently querying selected nodes in the network, as opposed to querying all the network nodes.

A related issue is that of updates in the network topology. For a routing protocol to be efficient, changes in the network topology should have only a local effect. In other words, creation of a new link at one end of the network is an important local event but, most probably, not a significant piece of information at the other end of the network. Proactive protocols tend to distribute such topological changes widely in the network, incurring large costs. The ZRP limits propagation of such information to the neighborhood of the change only, thus limiting the cost of topological updates.

## 2. Overview of the Zone Routing Protocol

### 2.1 The Notion of a Routing Zone and the Intrazone Routing Protocol (IARP)

A routing zone is defined for each node X, and includes the nodes whose minimum distance in *hops* from X is at most some predefined number, which is referred to as the Zone Radius. An example of a routing zone (for node A) of radius 2 is shown below.




Note that in this example nodes B through E are within the routing zone of A. Node G is outside A's routing zone. Also note that E can be reached by two paths from A, one with length 2 hops and one with length 3 hope. Since the minimum is less or equal than 2, E is within A's routing zone.

Peripheral nodes are nodes whose minimum distance to the node in question is equal exactly to the zone radius. Thus, in the above figure, nodes D, F, and E are A's peripheral nodes.

An important consequence of the routing zone construction is the ability of a node to deliver a packet to its peripheral nodes.

This service, which we refer to as bordercasting, allows for more efficient network-wide searching than simple neighbor broadcasting. Bordercasting could be implemented either through a series of IP unicasts or an IP multicast (Distance Vector Multicast Routing

Protocol [RFC-1075])) to the peripheral nodes. (In cases where multicasting is supported, the multicasting approach is preferred to reduce the amount of traffic over the air.) However, as will be explained later, efficient ZRP operation requires that these unicast or multicast services be provided by the ZRP, with IP providing best-effort delivery to the specified ZRP next hops.

The ZRP supports the proactive maintenance of routing zones through its proactive Intrazone Routing Protocol (IARP). Through the IARP, each node learns the identity of and the (minimal) distance to all the nodes in its routing zone. The IARP may be derived from a wide range of proactive protocols, such as Distance Vector (e.g., [Murthy], [DSDV]) or Shortest Path First (e.g., OSPF [RFC-2178]). Whatever the choice of IARP is, the base protocol needs to be modified to ensure that the scope of this operation is restricted to the radius of a node's routing zone.

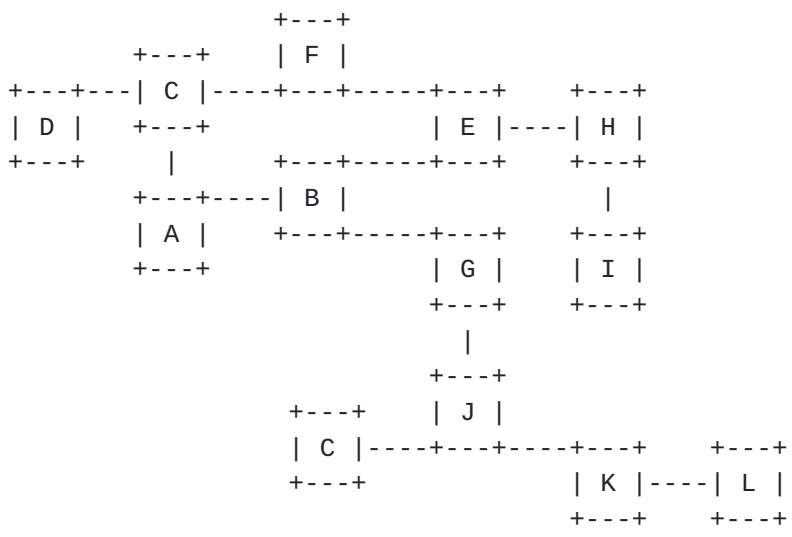
Because each node needs to proactively acquire route information only for the nodes within its zone, the total amount of IARP traffic does not depend on the size of the network, which may be quite large

## 2.2 The Interzone Routing Protocol (IERP)

While the IARP maintains routes within a zone, the IERP* is responsible for finding routes between nodes located at distances larger than the zone radius. The IERP is distinguished from standard flood-search query/response protocols by exploiting the routing zone topology. A node is able to respond positively to any queries for its routing zone nodes. For large networks, relatively few destinations lie within any particular node's routing zone. In this more likely case, the node can efficiently continue the propagation of the query, through the routing zone-based bordercast delivery mechanism.

### 2.2.1 Routing Zone Based Route Discovery

**The IERP operates as follows: The source node first checks whether** the destination is within its routing zone. (Again, this is possible as every node knows the content of its zone). If so, the path to the destination is known and no further route discovery processing is required. If, on the other hand, the destination is not within the source's routing zone, the source bordercasts a route query to all of its peripheral nodes. Now, in turn, all the peripheral nodes execute the same algorithm: check whether the destination is within their zone. If so, a route reply is sent back to the source indicating the route to the destination. If not, the peripheral node forwards the query to its peripheral nodes, which, in turn, executes the same procedure. An example of this Route Discovery procedure is demonstrated in the figure below.




The node A has a datagram to send to node L. Assume a uniform routing zone radius of 2 hops. Since L is not in A's routing zone (which includes B,C,D,E,F,G), A bordercasts a routing query to its peripheral nodes: D,F,E, and G. Each one of these peripheral nodes check whether L exists in their routing zones. Since L is not found in any routing zones of these nodes, the nodes bordercast the request to their peripheral nodes. In particular, G bordercasts to K, which realizes that L is in its routing zone and returns the requested route (L-K-G-A) to the query source, namely A.

* Some functions of the IERP, including bordercasting, route accumulation, and query control (see later), are performed by a special component of the IERP called the Bordercast Resolution Protocol (BRP). Sections 3.0 and 4.0 describe, in detail, the relationship between the BRP and the IERP proper.

### 2.2.2 Route Accumulation Procedure

The query propagation mechanism allows a route query to indirectly reach the desired destination (through some node Y, which discovers the destination in its routing zone.) To complete the route discovery process, Y needs to send a reply back to the query's source, S, providing S with the desired route. To perform the route reply, sufficient information needs to be accumulated during the query phase so that Y is provided with a route back to S. Providing routes from discovering node Y to query source S, and from the query source S back to the query destination D (through Y), is the role of the Route Accumulation procedure.

In the basic Route Accumulation, a node appends its IP address to a received query packet. The sequence of IP addresses specifies a route from the query's source to the current node. By reversing this

sequence, a route may also be obtained back to the source. In this way, Y may send a reply back to S, through strict source routing.

Given sufficient storage space, a queried node may cache routing information accumulated in the query packet, allowing the information to be remove from the packet. This has the benefit of reducing the length of the query packet, thereby decreasing the query traffic and query response time. The IP addresses that remain in the packet can be used to form a loose source route back to the query's source (If ALL nodes have cached the accumulated route information, then this effectively becomes next hop routing. If no nodes have cached accumulated route information, then this defaults to the basic case previously discussed). The same caching strategy can be applied to the reply packet on its way back to the source. In this case, a loose source route to the destination is formed.

### 2.2.3 Query Control Mechanisms

Bordercasting has the potential to support global querying schemes that are more efficient than flooding. To achieve this efficiency, the protocol should be able to detect and terminate a query thread when it appears in a previously queried region of the network (i.e. arrives at a node belonging to the routing zone of a previously queried node). This detection / termination capability is significantly limited when bordercasting is implemented directly through IP unicast or IP multicast.

By implementing bordercasting within the ZRP, the nodes that relay the query to the peripheral node are able to detect the passing query. If the underlying IP delivery is (neighbor) broadcast or if IP is operating in promiscuous mode, then nodes that overhear a query transmission are also able to detect the query, further strengthening the Query Detection (QD) mechanism. Upon detecting a query, the identifying query parameters (i.e. query source, query ID) are recorded in a Detected Queries Table, to provide a basis for termination of future threads of that query.

A node can consider a query to be redundant if it has already detected that query, bordercasted by a different node. If bordercasting is implemented directly through IP unicast/ multicast, then a query thread could only be terminated after being received by the peripheral node (bordercast destination). This could result in wasted transmissions as a query penetrates into a previously queried region. Implementing bordercasting in the ZRP allows the protocol to provide an Early Termination (ET), as the redundant query enters the previously queried region.

### 2.2.4 Route Maintenance

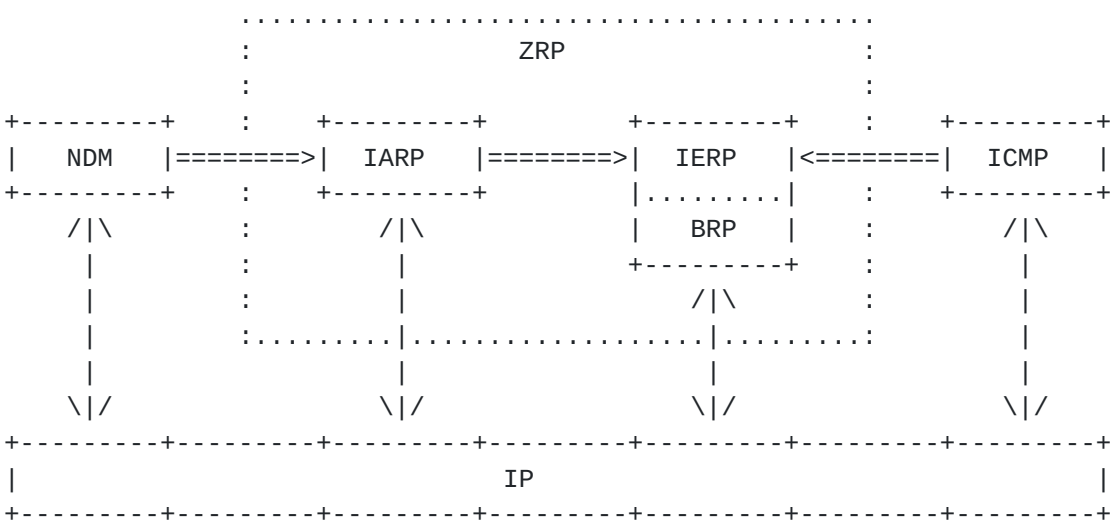
In addition to initially discovering routes, the IERP may also assume responsibility for monitoring the integrity of these routes and repairing failed routes as appropriate.

Route failures can be detected either proactively or reactively. Proactive route failure detection is triggered by the IARP, in response to a node leaving the routing zone. Any IERP routes

containing this node as the first hop can be considered invalid. Route failures may also be detected reactively, by IP, when the next hop in a datagram's source route is determined to be unreachable (i.e. does not appear in the (Intrazone) Routing Table).

Upon detection of a route failure, a node may choose to notify the route's source of the failure and / or attempt to repair the route. A route failure notification consists of a transmission back to the query source, indicating that the source route has failed, and possibly the hop at which it failed. This type of service is provided by protocols like ICMP. The node that detects the route failure may also try to repair the failed connection by discovering a route to bypass the failed connection. The repair discovery process is nearly identical to an initial route discovery. Route repairs should not be substantially longer than the original failed connection. Thus, the depth of a repair query can be limited, through the use of hop counter. This has the advantage of producing much less query traffic than an unrestricted initial route query. After a successful repair, the route's source MAY be notified so that routes are properly selected for use. Alternatively, the repair could go unreported without compromising the connectivity between source and destination.

## 3.0 The ZRP Architecture




Legend:

A <---> B exchange of packets between protocols A & B
A ===> B information passed from protocol A to protocol B

Existing Protocols

| | |
|---|---|
| IP | Internet Protocol |
| ICMP | Internet Control Message Protocol |

ZRP Entities

| | |
|---|---|
| IARP | IntrAzone Routing Protocol |
| IERP | IntErzone Routing Protocol |
| BRP | Bordercast Resolution Protocol (component of IERP) |

Additional Protocols

| | |
|---|---|
| NDM | Neighbor Discovery/Maintenance Protocol |

Note, it is assumed that basic neighbor discovery operation is implemented by the MAC/link-layer protocols. Thus the NDM protocol remains unspecified here.

## 4. Implementation Details

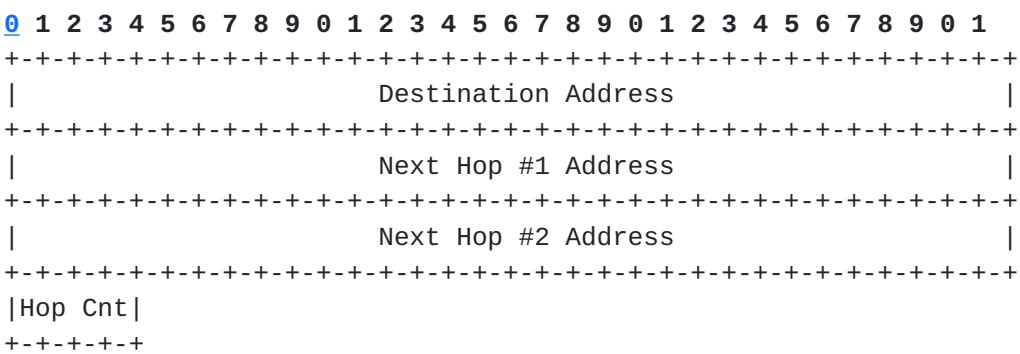
### 4.1 The IntrAzone Routing Protocol (IARP)

The IARP can be implemented through various proactive protocols. We present here an implementation based on a modified version of the Distance Vector algorithm. Routing information to a node is limited to the the routing zone of that node. In this version, routing information consists of the route cost and the IP addresses of the route destination and next two hops to the destination. The second hop is included to identify routes where a node is it's own second hop. This particular looping condition can result in the "counting to infinity" problem. By deleting these routes, this problem can be avoided, allowing the IARP to converge faster.

A node may receive new route information either from a received IARP packet or from an interrupt generated by its Neighbor Discovery / Maintenance (NDM) proces*. In the special case when a host has discovered a neighor, the IARP is responsible for sending to the new neighbor the shortest route to each host which is common to both of their routing zones (i.e. each host with a hop count less than it's routing zone radius). The node then records the new route information in its Intrazone Routing Table. If the shortest path to the host has changed, the terminal broadcasts an IARP packet reflecting the change.

* IARP relies on the services of a separate protocol (referred to here as the Neighbor Discovery/Maintenance Protocol) to provide current information about a host's neighbors. At the least, this information must include the IP addresses of all the neighbors. The IARP can be readily configured to support supplemental neighbor information, such as link cost.

A. Packet Format



Field Description:

* Destination Address (32 bits)
  IP address of the destination host
* Next Hop # 1 Address (32 bits)
  IP address of the "next hop" host to the destination host
* Next Hop # 2 Address (32 bits)
  IP address of the Next Hop #1 's "next hop" host to the destination host
* Hop Count (4 bits)
  Length of the route to the destination host, measured in hops

## B. Structures

### B.1 Intrazone Routing Table

| | Routes | | | | | |
|---|---|---|---|---|---|---|
| Host | 0 | | 1 | | ==> | M-1 | |
| | Next Hop | Hop Count | Next Hop | Hop Count | ==> | Next Hop | Hop Count |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

## C. Interfaces

C.1 Higher Layer (N/A)
C.2 Lower Layer (IP)
  C.1.1 Send() (specified in IP standard)
  C.1.2 Deliver() (specified in IP standard)
C.3 NDM
  C.3.1 Neighbor_Lost(host)
    Used by the NDM to notify the IARP that direct contact has been lost with "host".
  C.3.2 Neighbor_Found(host)
    Used by the NDM to notify the IARP that direct contact has been confirmed with "host".
C.4 IERP
  C.4.1 Zone_Node_Lost(host)
    Used by IARP to notify the IERP that a node no longer

exists within the routing zone.

## D. State Machine

The IARP protocol consists of only one state (IDLE). Therefore, no state transitions need to be specified. The IARP immediately acts upon an event and then returns back to the IDLE state.

Notes: 1) X is used as a label for the host running this state machine.
2) INF is a reserved field value corresponding to "infinity".

D.1

Event: An IARP packet is received containing route information to a destination D. The hop count associated with the received route is LESS THAN OR EQUAL TO the routing zone radius. The second next-hop is EQUAL to X.

Action: NONE

D.2

Event: An IARP packet is received containing route information to a destination D. The hop count associated with the received route is LESS THAN the routing zone radius. The second next-hop is NOT EQUAL to X.

Action: The received route is recorded in the Intrazone Routing Table. If the received route is shorter than the previous shortest route to D, then a new IARP packet containing route information to D through X is broadcasted.

D.3

Event: An IARP packet is received containing route information to a destination D. The hop count is EQUAL TO the routing zone radius. The second next-hop is NOT EQUAL to X.

Action: The received route is recorded in the Intrazone Routing Table.

D.4

Event: An IARP packet is received containing route information to a destination D. The hop count is equal to INF.

Action: The route to D is removed from the Intrazone Routing Table.

1) If the Intrazone Routing Table still contains a route to D and the length of the shortest route has increased due to the route removal, then the an IARP packet containing the shortest route to D through X is broadcasted.

2) If the Intrazone Routing Table contains no more routes to D, then an IARP packet containing a route to D through X with hop count of INF is broadcast. A "Host Lost" interrupt is generated to alert the IERP that D has moved beyond the routing zone.

D.5

Event: A "Neighbor Found" interrupt is received, indicating the discovery of a neighbor host N.

Action: For each destination in X's Intrazone Routing Table, an IARP packet is sent to N containing the best route to that destination. An IARP packet is then broadcasted containing the 1 hop route to N through X.

D.6

Event: A "Neighbor Lost" interrupt is received, indicating that host N is no longer a neighbor of X

Action: The one hop route to N is removed from the Intrazone Routing Table.

1) If the Intrazone Routing Table still contains a route to N and the length of the shortest route has increased due to the route removal, then the an IARP packet containing the shortest route to N through X is broadcasted.

2) If the Intrazone Routing Table contains no more routes to N, then an IARP packet containing a route to D through X with hop count of INF is broadcast. A "Host Lost" interrupt is generated to alert the IERP that D has moved beyond the routing zone.

## E. Pseudocode Implementation

### E.1 Update Intrazone Routing Table

```
if (packet arrived)
  {host, route->next_hop, next_next_hop, route->hop_count}
                                        <-- packet
else
{
  {host} <-- intrpt
  route->next_hop=host
  if (type(intrpt) == "Neighbor Found")
  {
    for dest = each host in Intrazone_Routing_Table
    {
      best_route = Intrazone_Routing_Table[dest,0]
      if (best_route->hop_count <  ROUTING_ZONE_RADIUS)
      {
        packet <--{dest,my_id,best_route->next_hop,
                                        best_route->hop_count+1}
        send(packet,host)
      }
    }
    route->hop_count=1
  }
  else
    route->hop_count=INF
}

former_best_route = Intrazone_Routing_Table[host,0]
if (route->hop_count < INF)
  if(route->next_next_hop != my_id
    add(Intrazone_Routing_Table[host], route)
else
    remove(Intrazone_Routing_Table[host], route)
best_route = Intrazone_Routing_Table[host,0]
if (best_route != NULL)
{
  if (best_route->hop_count != former_best_route->hop_count
        && best_route->hop_count < ROUTING_ZONE_RADIUS)
  {
    packet <-- {host, my_id, best_route->next_hop,
                                    best_route->hop_count+1}
    broadcast(packet)
  }
}
else
{
  force_intrpt("IERP","Zone Node Lost",{host})
```

```
            packet <-- {host, my_id, my_id, INF}
            broadcast(packet)
          }
```

### 4.2 IntErzone Routing Protocol (IERP)

The Interzone Routing Protocol (IERP) is responsible for discovering and maintaining routes to hosts which are beyond a node's routing zone. The IERP acquires routing information reactively, exploiting the knowledge of the routing zone topology through the bordercasting delivery service.

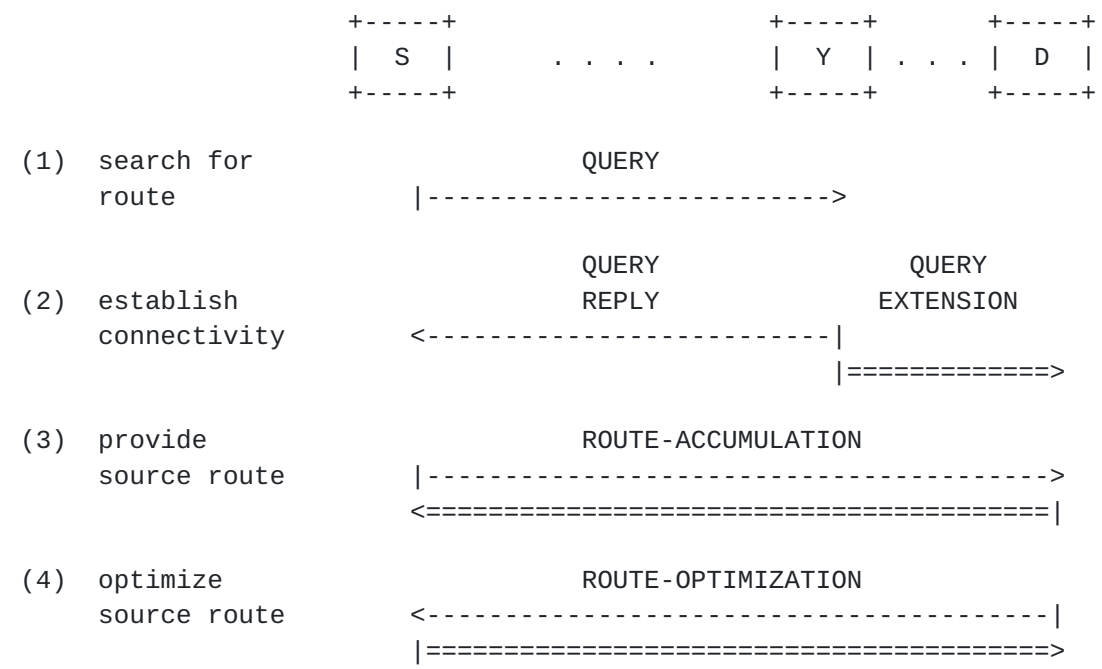
This version of the IERP extends the basic query / reply mechanism described in Section 3. In the basic protocol, queries are terminated before reaching the query destination. This provides a faster response to the route query than if the destination, D, were to respond directly. However, because D never receives the query, it does not acquire a route back to the source, S. If the D needs to send data back to S (which is often the case, given the bi-directional nature of many applications), a separate route query from D to S would have to be initiated. This substantial overhead is avoided by having the query forwarded to D, by the node Y that discovers D in its routing zone. We refer to this as a QUERY EXTENSION.

Both the source and destination can acquire routes to each other through the BRP route accumulation mechanisms (to be discussed later). Distributing route information in the caches of the route's nodes can significantly reduce the size of the IERP packets and provide for a faster query response. On the other hand, QOS requirements for a particular application may favor a source specified route. (rather than a distributed next-hop approach). Source routing requires that the discovered route information be accumulated in the IERP packets, rather than cached. This IERP implementation satisfies the demands for rapid query response and source routing support by two stages of route reporting. In the first two stages, route information is cached by the route's nodes (when possible). Next-hop route are quickly returned to the source in QUERY-REPLY packets and forwarded to the destination in QUERY-EXTENSION packets. At this point, bi-directional connectivity is established. In the third stage, the source and destination can provide each other with complete source routes, by sending each other ROUTE-ACCUMULATION packets. As these packets traverse the route, the cached route information is accumulated in the packets, thereby constructing the desired source route.

Variations on this IERP implementation can be realized by combining or eliminating some of these stages. For example, if source routing is not desired, the ROUTE-ACCUMULATION stage can be eliminated. Also, if all of the route's nodes elect not to cache the routing information (perhaps due to storage limitations), then the QUERY REPLY and QUERY-EXTENSION packets essentially serve the role of ROUTE-ACCUMULATION packets, obviating the need for a separate

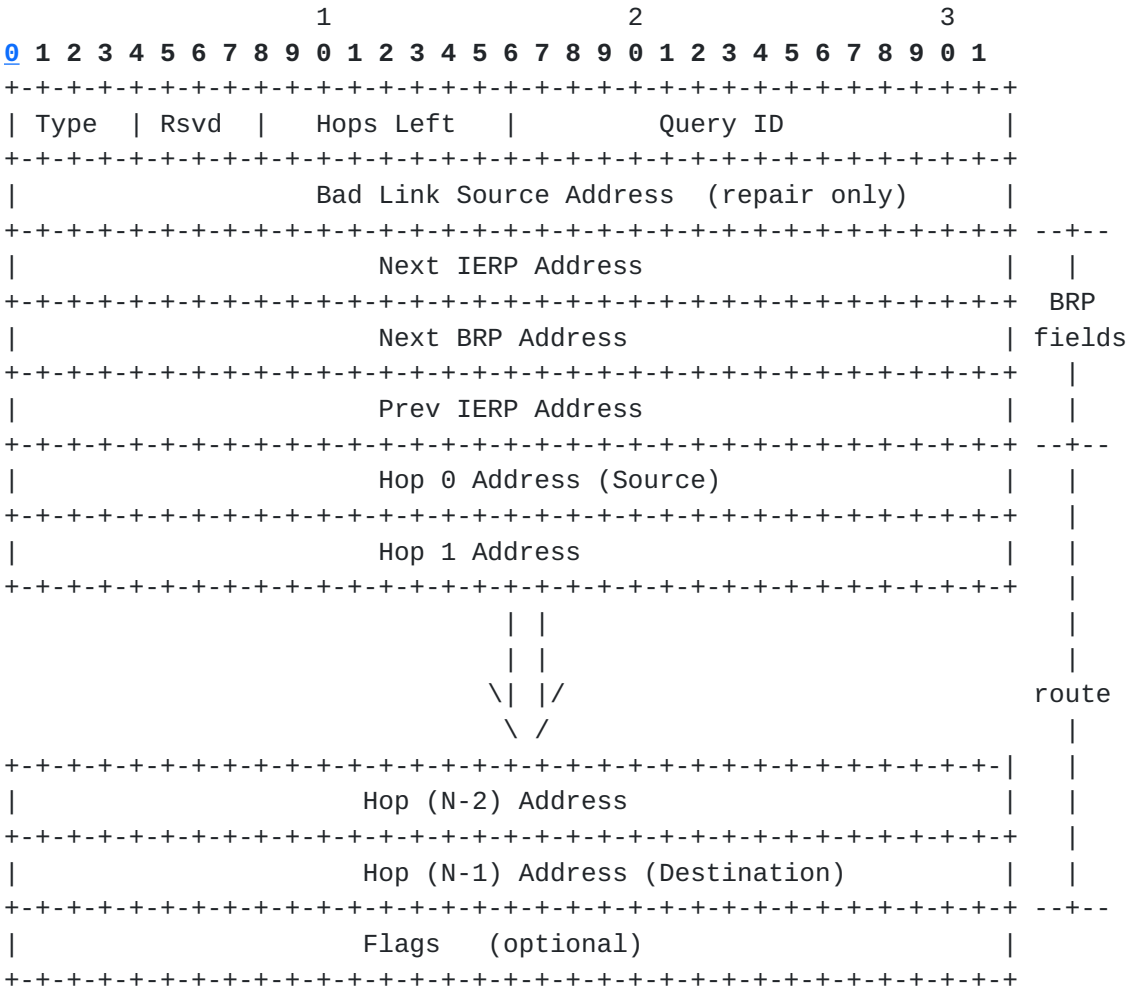
ROUTE-ACCUMULATION stage.

Because each node proactively maintains the local topology of its routing zone, it is not necessary for a source route to specify every hop along the route (i.e. strict source routing). A properly chosen subset of the complete source route can be used to specify the route to the destination, and where desirable, the reverse route back to the source. The IERP supports such an optimization through a final ROUTE-OPTIMIZATION stage. The details of the route optimization are described in the BRP specification. Upon completion of the ROUTE-OPTIMIZATION stage, sufficient routing zone membership is acquired for each node in the route so that the source route may be reduced (by translating this route reduction into an appropriate set covering problem, and employing a suitable heuristic).




Sequence of the IERP Route Discovery Stages

A. Packet Format




Field Description:

* Type (4 bits)

Identifies the type of IERP packet. The current version of IERP contains five packet types:

QUERY:

request for an Interzone source route to the destination specified by the Destination IP Address

QUERY-EXTENSION:

extension of a QUERY packet sent from the node that discovers the Destination to the Destination itself.

QUERY-REPLY:
response to a QUERY packet, sent from the node that discovers the Destination back to the Source. At a minimum, the QUERY-REPLY contains next-hop route information to the Destination. (In general, returns the source route up to the first node which has cached routing information. If no nodes have cached routing information, then the complete source route is returned.)

ROUTE-ACCUMULATION (optional):
sent by the Source to the Destination, in response to a QUERY-REPLY, and sent by the Destination to the Source, in response to a QUERY-EXTENSION. Routing information cached at the route's nodes is accumulated in this packet, providing a complete source route at the destination of this packet.

ROUTE-OPTIMIZATION (optional):
sent by the Source to the Destination, and by the Destination to the Source, in response to a ROUTE-ACCUMULATION. Flags are appended to this packet reflecting the mutual routing zone membership of each node in the source route.

* Query ID (16 bits)
Sequence number which, along with the Source Address (see below) uniquely identifies any route query in the network.

* Hops Left (8 bits)
Number of hops that a route query may continue to propagate. This field allows a querying node to configure the depth of a route search, to control the amount of IERP traffic generated.

* Bad Link Address (32 bits)
Used during route repairs. Contains the IP Address corresponding to the source of routes failed link.

* Next IERP Address (32 bits)
IP address of the next IERP recipient

* Next BRP Address (32 bits)
IP address of the next BRP recipient.
(i.e. next hop to the next IERP recipient)

* Prev IERP Address (32 bits)
IP address of the previous IERP recipient

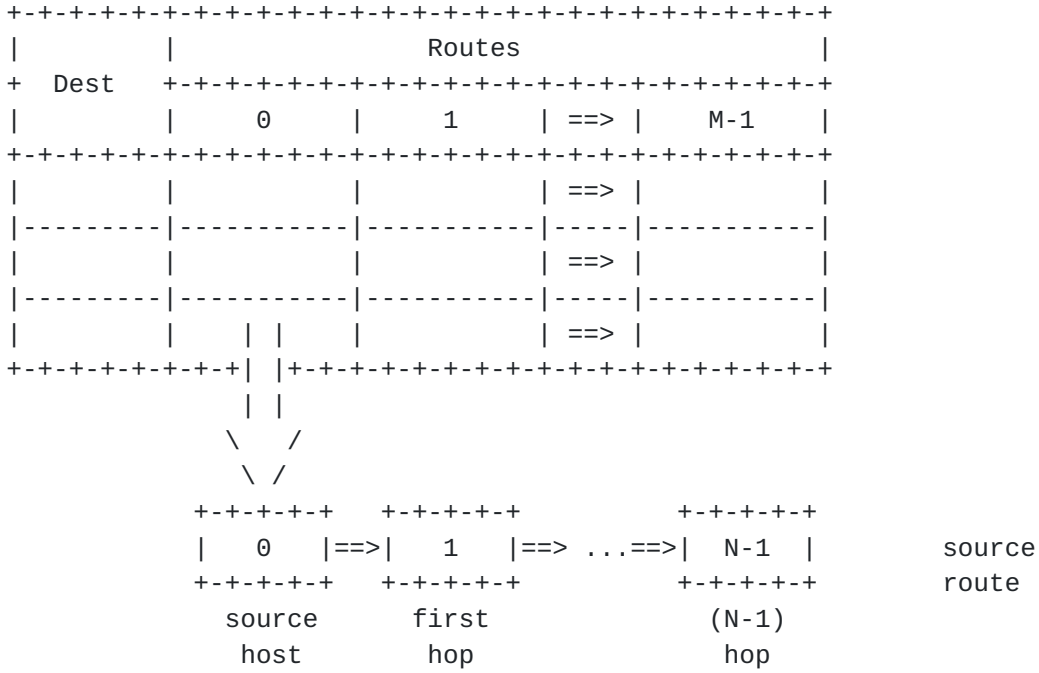
* Source Route (N * 32 bits)
  Variable length field that contains the IP addresses of the source route's nodes.
  - route(0) contains the IP address of the route's source.
  - route(N-1) contains the IP address of the route's destination
  - in general, route(n) contains the IP address of the n-th hop of the source route

* Flags (N * 32*ceil(N/32) bits)
  The k-th bit of the n-th flag subfield indicates whether route(k) is located within route(n)'s routing zone. This routing zone membership information, collected during the optional ROUTE-OPTIMIZATION stage, may be used to determine the shortest possible specification for the accumulated source route.

## B. Structures

### B.1 Intrazone Routing Table (See IARP Description)

### B.2 Interzone Routing Table



C. Interfaces

C.1 Higher Layer (N/A)
C.2 Lower Layer (BRP)
C.2.1 Send() (same interface as IP send())
Used by the IERP to request transmission of an IERP packet.
C.2.2 Deliver() (same interface as IP deliver())
Used by the BRP to alert the IERP of the arrival of a data packet.
C.3 IARP
C.3.1 Zone_Node_Lost(host)
Used by the IARP to notify the IERP that a node has left the routing zone
C.4 ICMP
C.4.1 Host_Unreachable() (specified in ICMP standard)
C.4.2 Source_Route_Failed() (specified in ICMP standard)

D. State Machine

The IERP protocol consists of only one state (IDLE). Therefore, no state transitions need to be specified. The IERP immediately acts upon an event and then returns back to the IDLE state.

Note: 1) X is used as a label for the host running this state machine.

D.1
Event: A "Zone_Node_Lost" interrupt is received, indicating that the node H has moved beyond the routing zone.

Action: Remove every route from the Interzone Routing Table which contains H. If any routes containing H are found, then a route repair (limited depth route discovery) to H is initiated.

D.2
Event: A "Host_Unreachable" interrupt is received from the ICMP, indicating an unknown destination host D.

Action: A full depth route discovery to D is initiated. The query_id is incremented and assigned to a new QUERY packet. The route is initialized with the IP addresses of the route's source and destination IP addresses (X and D). Finally, the QUERY packet is bordercasted.

D.3

Event: A "Source_Route_Failed" or "Proactive_Repair" interrupt is received, indicating that the next hop, H, specified in a source route does not appear within the routing zone.

Action: A limited depth route discovery to H is initiated. The query_id is incremented and assigned to a new QUERY packet. The route is initialized with the valid portion of the failed route, and the bad link address field is set with X's IP address, to indicate the location of the route failure. Finally, the QUERY packet is bordercasted.

D.4

Event: A QUERY packet is received with a destination that appears within X's routing zone.

Action: A QUERY-REPLY is sent back to the query source, and a QUERY-EXTENSION is sent to the query destination.

D.5

Event: A QUERY packet is received with a destination that DOES NOT appear within X's routing zone.

Action: The QUERY packet is bordercasted.

D.6

Event: A QUERY-EXTENSION packet is received.

Action: The packet contents are moved to a ROUTE-ACCUMULATION packet. The ROUTE-ACCUMULATION packet is sent to the query source.

D.7

Event: A QUERY-REPLY packet is received.

Action: The packet contents are moved to a ROUTE-ACCUMULATION packet. The ROUTE-ACCUMULATION packet is sent to the query destination.

D.8

Event: A ROUTE-ACCUMULATION packet is received. X is not the final destination of this packet (i.e. X != IERP_next). This only occurs when the accumulated route contains a repair

Action: The bad link is replaced by the path repair in the Interzone Routing Table.

D.9

Event: A ROUTE-ACCUMULATION packet is received. X is either the route source or (route destination).

Action: The (reversed) accumulated route is added to the Interzone Routing Table or replaces a failed route if the packet specifies a bad link. In addition, if X is the ROUTE-ACCUMULATION destination, the packet contents may be moved to a ROUTE-OPTIMIZATION packet, which is then sent to the query destination (query source).

D.10

Event: A ROUTE-OPTIMIZATION packet is received.

Action: The routing zone membership information that is collected in the ROUTE-OPTIMIZATION packet is processed. The resulting optimized route(s) are added to the Interzone Routing Table.

## E. Pseudocode Implementation

We define two complimentary operations on packets: extract(packet) and load(packet)

```
extract (packet)
    extracts the fields of the IERP packet to the following
    variables: {type, hops_left, query_id, bad_link_source,
                next_IERP, next_BRP, prev_IERP, route, flag}

load (packet)
    loads the values of the aforementioned variables into
    the fields of the IERP packet.
```

### E.1 Routing Zone Node Lost

```
{lost_host} <-- intrpt
repair_link = FALSE
for host = each host in Interzone Routing Table
{
  for (route = each Interzone route to host)
  {
    if (lost_host (EXISTS IN) route)
    {
        if (PROACTIVE_REPAIRS_ENABLED)
        {
            removal_timer = ROUTE_QUERY_TIMEOUT
            repair_link = TRUE
        }
        else
            removal_timer = 0
        schedule(
            remove(Interzone_Routing_Table[host]->route(m)),
            removal_timer)
    }
  }
}
if(repair_link)
{
    bad_link = my_id + lost_host
    force_intrpt("IERP","Proactive_Repair", bad_link)
}
```

### E.2 Initiate Route Discovery

```
{route} <-- intrpt

dest            = route(length(route)-1)
current_hop_ptr = get_index(route, my_id)
prev_hops       = route(0: current_hop_ptr-1)

route = prev_hops + my_id + dest
if (type(intrpt) == "Proactive_Repair" ||
    type(intrpt) == "Source_Route_Failed")
{
    hops_left        = MAX_REPAIR_HOPS
    bad_link_source  = my_id
}
else
{
    hops_left        = MAX_FULL_QUERY_HOPS
    bad_link_source  = NULL
}
```

```
            query_id ++
            load (packet)
            send (packet, BORDERCAST)
```

### E.3 Receive IERP Packet

```
extract(packet)
source=route(0)
dest = route(length(route)-1)
switch(pk_type)
{
    case:  QUERY
        if (dest (EXISTS IN) Intrazone_Routing_Table)
        {
            type = QUERY-REPLY
            if(bad_link_source)
                IERP_next = bad_link_source
            else
                IERP_next = source
            load(packet)
            send(packet)

            type = QUERY-EXTENSION
            IERP_next = dest
            load(packet)
            send(packet)
        }
        else
            bordercast(packet)
  break
  case:  QUERY-EXTENSION
        type = ROUTE-ACCUMULATION
        IERP_next=source
        send(packet)
  break
  case:   QUERY-REPLY
        type = ROUTE-ACCUMULATION
        IERP_next = dest
        load (packet)
        send (packet)
  break
```

```
            case:   ROUTE-ACCUMULATION
              if (bad_link_source)
              {
                path_source_ptr = get_index(route, bad_link_source)
                path_dest_ptr   = get_index(route, dest)
                if (IERP_next == source)
                {
                   bad_link     = bad_link_source + dest
                   path_repair  = route(path_source_ptr:path_dest_ptr)
                }
                if (IERP_next  == dest)
                {
                   bad_link     = dest + bad_link_source
                   path_repair  = reverse(route(path_source_ptr :
                                                path_dest_ptr))
                }
                replace(Interzone_Routing_Table, bad_link,path_repair)
              }
              else
              {
                if (my_id == source)
                  add(Interzone_Routing_Table, route)
                if (my_id == dest)
                  add(Interzone_Routing_Table, reverse(route))
              }

              if (my_id == IERP_next)
              {
                if (my_id == source)
                  IERP_next = dest
                if (my_id == dest)
                  IERP_next = source

                type = ROUTE-OPTIMIZATION
                load (packet)
                send (packet)
              }
            break
            case:   ROUTE-OPTIMIZATION
              if (my_id == source)
                  add(Interzone_Routing_Table, route, flags)
              if (my_id == dest)
                  add(Interzone_Routing_Table, reverse(route), flags)
            break
          }
```

### 4.3 Bordercast Resolution Protocol (BRP)

The BRP is a sublayer of the IERP that performs the operations of bordercasting, query control, route accumulation and routing zone labelling, which form the basis for the route discovery procedure.

In this BRP implementation, bordercasting is implemented as a series of unicasted messages to the peripheral nodes. The BRP is able to identify the peripheral based on the information contained in the Intrazone Routing Table. Rather than unicasting to the peripheral node directly through IP, the bordercasted packets are relayed to the peripheral nodes by the BRP layer. IP is used only to send these messages one hop toward the peripheral nodes. This allows the BRP to detect all QUERY packets that are received by that node.

To efficiently terminate the query, the BRP needs to record sufficient information from each detected query. The query's source and ID, which serve to uniquely identify a query, are added to the Detected Queries Table. In addition, the IP address of the last node to bordercast the query is also recorded in the Detected Queries table. Any threads of this query that are later received from a different bordercasting node are discarded. Each query also contains a hop counter that is decremented at each node. When the counter expires, the packet is discarded.

IERP packets (excluding ROUTE-OPTIMIZATION packets) that are not terminated next undergo a route accumulation procedure. As described earlier, route accumulation is used to construct routes, by recording the IP addresses of a route's nodes in the IERP packet or local cache. The Detected Queries Table, extended by two columns, serves as a convenient route accumulation cache.

The node begins the route accumulation procedure by adding its IP address to the IERP route. This is followed by the IP addresses of any cached nodes leading to the packet's destination (the "next hops"). This is sufficient processing for ROUTE-ACCUMULATION packets, where complete source routes are constructed. On the other hand, QUERY, QUERY-EXTENSION and QUERY-REPLY packets should carry as little of the route as possible. Therefore, if cache space is available, the route accumulation process records the IP addresses of the "previous hops" from the packet's source, and removes them from the IERP packet.

The final role of the BRP is contribute to the ROUTE-OPTIMIZATION process by indicating the mutual routing zone membership of the nodes in the source route. This is done by appending a special flag field to the ROUTE-OPTIMIZATION packet. The status of the k-th bit in the flag field indicates whether the k-th hop in the source route is a member of the node's routing zone. This

membership information is eventually processed at the source to determine the smallest set of routing zones that cover the route (and therefore the smallest set of nodes needed to specify this route through loose source routing.)

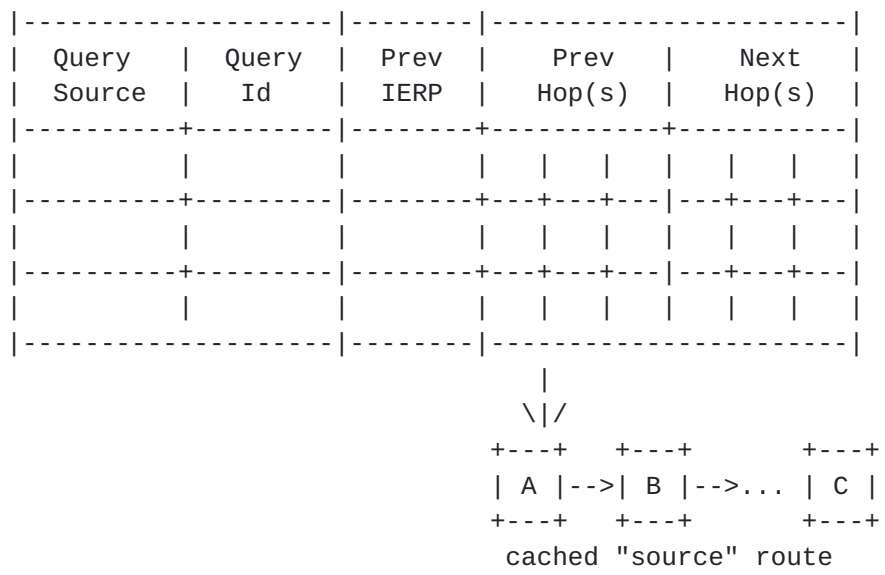
A. Packet Format

See IERP packet format in Section 4.2

B. Structures

B.1 Intrazone Routing Table (see IARP description)

B.2 Detected Queries Table




C. Interfaces

C.1 Higher Layer (i.e. IERP)

C.2.1 Send() (same interface as IP Send() primitive)
Used by higher layer protocol to request transmission of a data packet

C.2.2 Deliver() (same interface as IP Deliver() primitive)
Used by the BRP to alert the higher layer protocol of the arrival of a data packet

C.2 Lower Layer (IP)

C.2.1 Send() (specified in IP standard)

C.2.2 Deliver() (specified in IP standard)

D. State Machine

The BRP protocol consists of only one state (IDLE). Therefore, no state transitions need to be specified. The BRP immediately acts upon an event and then returns back to the IDLE state.

Notes: 1) X is used as a label for the host running this state machine.

2) NULL is a reserved field value, corresponding to an invalid IP address.

D.1

Event: A QUERY packet is received from the IERP

Action: If X is the query's source, then the identifying querying information is recorded in the Detected Queries Table. X adds its IP address to the packet's route. A copy of the packet is sent to the IERP layer of each peripheral node, by way of a BRP transmission to the next hop to that peripheral node.

D.2

Event: A QUERY packet is received from the IP. The hop counter has expired or the query was already detected from another bordercasting node.

Action: The packet is discarded.

D.3

Event: A QUERY packet is received from IP. The hop count has not expired and the query has not been previously detected (or was detected from the same bordercasting node). X is not the intended BRP recipient.

Action: If cache space is available, identifying query information SHOULD be recorded in the Detected Queries Table. The packet is then discarded.

D.4

Event: A QUERY packet is received from the IP. The hop count has not expired and the query has not been previously detected (or was previously detected from the same bordercasting node). X is the intended BRP recipient, but is not the intended IERP recipient and the query destination does not lie within X's routing zone.

Action: The hop counter is decremented. If cache space is available, identifying query information and accumulated route information should be recorded in the Detected Queries Table. The recorded route information is removed from the packet and X adds its IP address to the route.
The packet is then sent to the BRP of the next hop to the intended IERP recipient.

D.5

Event: A QUERY packet is received from the IP. The hop count has not expired and the query has not been previously detected (or was previously detected from the same bordercasting node). X is the intended BRP recipient, and either X is the intended IERP recipient, OR the query destination lies in X's routing zone.

Action: The hop counter is decremented. If cache space is available, identifying query information and accumulated route information should be recorded in the Detected Queries Table. The recorded route information is removed from the packet and X adds its IP address to the route.
The packet is then delivered up to the IERP.

D.6

Event: A QUERY-EXTENSION is received from the IERP.

Action: The packet is sent to the BRP layer of the next hop to the specified IERP recipient (in this case, the query destination).

D.7

Event: A QUERY-EXTENSION is received from the IP.
X is not the intended BRP recipient.

Action: If cache space is available, identifying query information SHOULD be recorded in the Detected Queries Table. The packet is then discarded.

D.8

Event: A QUERY-EXTENSION packet is received from the IP.
X is the intended BRP recipient, but is not the intended IERP recipient.

Action: If cache space is available, identifying query information and accumulated route information should be recorded in the Detected Queries Table. The recorded route information is removed from the packet and X adds its IP address to the route.
The packet is then sent to the BRP of the next hop to the intended IERP recipient.

D.9

Event: A QUERY-EXTENSION packet is received from the IP. X is the intended BRP recipient and the intended IERP recipient.

Action: If cache space is available, identifying query information and accumulated route information should be recorded in the Detected Queries Table. The recorded route information is removed from the packet and X adds its IP address to the route. The packet is then delivered up to the IERP.

D.10

Event: A QUERY-REPLY packet is received from the IERP.

Action: The IP addresses of X and the previous hops back to the query source (cached in the Detected Queries Table) are added to the route. The packet is sent back to the IERP layer of the query source, by way of a BRP layer transmission to the first hop back to the query source.

D.11

Event: A QUERY-REPLY packet is received from the IP. X is not the intended BRP recipient.

Action: The packet is discarded.

D.12

Event: A QUERY-REPLY packet is received from the IP. X is the intended BRP recipient, but not the intended IERP recipient.

Action: If cache space is available, accumulated route information to the destination should be recorded in the Detected Queries Table, and the recorded route information removed from the packet. The IP addresses of X and the previous hops back to the query source (cached in the Detected Queries Table) are added to the route.
The packet is then sent to the BRP layer of the previous hop back to the query source.

D.13

Event: A QUERY-REPLY packet is received from the IP. X is the intended BRP recipient and the intended IERP recipient.

Action: If cache space is available, accumulated route information to the destination should be recorded

in the Detected Queries Table, and the recorded
route information removed from the packet.
The packet is then delivered up to the IERP.

D.14

Event: A ROUTE-ACCUMULATION packet is received from the IERP.

Action: The packet is sent to the IERP layer of the specified IERP recipient by way of a BRP transmission to the next hop toward the IERP recipient.

D.15

Event: A ROUTE-ACCUMULATION packet is received from the IP. X is not the intended BRP recipient.

Action: The packet is discarded.

D.16

Event: A ROUTE-ACCUMULATION packet is received from the IP. X is the intended BRP recipient, but not the intended IERP recipient.

Action: The IP addresses of X and the next hops to the intended IERP recipient (which are cached in the Detected Queries Table) are added to the route. If this packet contains a route repair and the repair has already been accumulated, then a copy of the packet is delivered to the IERP. The packet is then sent to the BRP layer of the next hop toward the IERP recipient.

D.17

Event: A ROUTE-ACCUMULATION packet is received from the IP. X is the intended BRP recipient and the intended IERP recipient.

Action: The packet is delivered up to the IERP.

D.18

Event: A ROUTE-OPTIMIZATION packet is received from the IERP.

Action: X indicates (in its flag field) those route nodes that belong to its routing zone. The packet is then sent to the IERP layer of the specified IERP recipient, by way of a BRP transmission to the next hop toward the IERP recipient.

D.19

Event: A ROUTE-OPTIMIZATION packet is received from the IP. X is not the intended BRP recipient.

Action: The packet is discarded.

D.20

Event: A ROUTE-OPTIMIZATION packet is received from the IP. X is the intended BRP recipient, but not the intended IERP recipient.

Action: X indicates (in its flag field) those route nodes that belong to its routing zone. The packet is then sent to the IERP layer of the specified IERP recipient, by way of a BRP transmission to the next hop toward the IERP recipient.

D.21

Event: A ROUTE-OPTIMIZATION packet is received from the IP. X is the intended BRP recipient and the intended IERP recipient.

Action: X indicates (in its flag field) those route nodes that belong to its routing zone. The packet is then delivered up to the IERP

## E. Pseudocode Implementation

We define two complimentary operations on packets: extract(packet) and load(packet)

```
extract (packet)
    extracts the fields of the IERP packet to the following
    variables: {type, hops_left, query_id, bad_link_source,
                next_IERP, next_BRP, prev_IERP, route, flag}

load (packet)
    loads the values of the aforementioned variables into
    the fields of the IERP packet.
```

### E.1 Receive Packet from IERP

```
extract (packet)

source          = route(0)
dest            = route(length(route)-1)
current_hop_ptr = get_index(route, my_id)
prev_hops       = reverse(route(0: current_hop_ptr-1))
next_hops       = route(current_hop_ptr+1 : length(route)-1)

IERP_prev = my_id
```

```
            switch(type)
            {
              case:  QUERY
                if ((bad_link_source == my_id || source == my_id)
                    add(Detected_Queries,packet)
                for (IERP_next = each host in Intrazone_Routing_Table)
                {
                    if (IERP next is a peripheral node)
                    {
                      BRP_next=Intrazone_Routing_Table[host,0]->next_hop
                      load(pk_copy)
                      send(pk_copy,BRP_next)
                    }
                }
              break
              case:  QUERY-EXTENSION
                BRP_next=Intrazone_Routing_Table[IERP_next,0]->next_hop
                load(packet)
                send(packet,BRP_next)
              break
              case:  QUERY-REPLY
                if (prev_hops(0) == source)
                    prev_hops = Detected_Queries[source,query_id]
                                                          ->prev_hops
                route = prev_hops + my_id + next_hops
                BRP_next = prev_hops(0)
                load(packet)
                send(packet,BRP_next)
              break
              case:  ROUTE-ACCUMULATION
                if(my_id == source)
                    BRP_next = next_hops(0)
                if(my_id == dest)
                    BRP_next = prev_hops(0)
                load(packet)
                send(packet,BRP_next)
              break
              case:  ROUTE-OPTIMIZATION
                flag = NULL
                for (i = 0 to length(route)-1)
                {
                    if ((EXISTS) Intrazone_Routing_Table[route(i)])
                        flag = flag,1
                    else
                        flag = flag,0
                }
                if(my_id == source)
                    BRP_next = next_hops(0)
                if(my_id == dest)
```

```
                BRP_next = prev_hops(0)
              load(packet)
              send(packet,BRP_next)
            break
```

### E.2 Receive Packet from IP

```
    extract(packet)

    source          = route(0)
    dest            = route(length(route)-1)
    current_hop_ptr = get_index(route, my_id)
    prev_hops       = reverse(route(0: current_hop_ptr-1))
    next_hops       = route(current_hop_ptr+1 : length(route)-1)

    switch(type)
    {
      case QUERY:
        if (hops_left > 0 &&
            (!EXISTS(Detected_Queries[source,query_id] ||
             Detected_Queries[source,query_id]->from
                                              == IERP_prev))
        {
            hops_left --
            if (!FULL (Detected_Queries))
            {
                add(Detected_Queries, packet)
                prev_hops = source
            }
            else
                route = prev_hops + my_id + next_hops

            if(BRP_next == my_id)
            {
                if(IERP_next == my_id ||
                   dest (EXISTS IN) Intrazone_Routing_Table)
                    deliver(packet,IERP)
                else
                {
                    BRP_next=Intrazone_Routing_Table[IERP_next]
                                          ->route(0)->next_hop
                    load(packet)
                    send(packet, BRP_next)
                }
            }
            else
                discard(packet)
        }
        else
            discard(packet)
      break
```

```
            case QUERY-EXTENSION:
              {
                  if (!FULL (Detected_Queries))
                  {
                      add(Detected_Queries,packet)
                      prev_hops = source
                  }
                  route = prev_hops + my_id + next_hops

                  if(BRP_next == my_id)
                  {
                      if(IERP_next == my_id ||
                         dest (EXISTS IN) Intrazone_Routing_Table)
                          deliver(packet,IERP)
                      else
                      {
                          BRP_next=Intrazone_Routing_Table[IERP_next]
                                                  ->route(0)->next_hop
                          load(packet)
                          send(packet, BRP_next)
                      }
                  }
                  else
                      discard(packet)
              }
              else
                  discard(packet)
            break

            case QUERY-REPLY:
              if(my_id == BRP_next)
              {
                  if (!FULL (Detected_Queries))
                  {
                      add(Detected_Queries, packet)
                      next_hops = dest
                  }
                  if (prev_hops(0) == source)
                     prev_hops = Detected_Queries[source,query_id]
                                                          ->prev_hops
                  route = prev_hops + my_id + next_hops
                  if(my_id == IERP_next)
                      deliver(packet,IERP)
                  else
                  {
                      BRP_next = prev_hops(0)
                      load(packet)
                      send(packet,BRP_next)
                  }
```

```
            }
            else
                discard(packet)
          break
```

```
            case ROUTE-ACCUMULATION:
              if(my_id == BRP_next)
              {
                  if(my_id == IERP_next)
                      deliver(packet, IERP)
                  else
                  {
                      if (bad_link_source && IERP_next == source)
                      {
                        bad_link_ptr=get_index(route,bad_link_source)
                        if (current_hop_ptr <= bad_link_ptr)
                              deliver(packet, IERP)
                      }
                      if (IERP_next == dest)
                      {
                        if(next_hops(0) == dest)
                           next_hops=Detected_Queries[source,query_id]
                                                        ->next_hops
                        BRP_next = next_hops(0)
                      }
                      if (IERP_next == source)
                      {
                        if(prev_hops(0) == source)
                           prev_hops=Detected_Queries[source,query_id]
                                                        ->prev_hops
                        BRP_next = prev_hops(0)
                      }
                      route = prev_hops + my_id + next_hops
                      load(packet)
                      send (packet,BRP_next)
                  }
              }
              else
                  discard(packet)
            break
```

```
              case ROUTE-OPTIMIZATION:
                if(my_id == BRP_next)
                {
                    f = NULL
                    for (i = 0: length(route)-1)
                    {
                      if ((EXISTS) Intrazone_Routing_Table[route(i)])
                        f = f,1
                      else
                        f = f,0
                    }
                    if (IERP_next == source)
                      flag = f , flag
                    else
                      flag = flag , f

                    if(my_id == IERP_next)
                        deliver(packet, IERP)
                    else
                    {
                      if(IERP_next == source)
                        BRP_next = prev_hops(0)
                      else
                        BRP_next = next_hops(0)
                      load(packet)
                      send (packet,BRP_next)
                    }
                }
                else
                    discard(packet)
              break
```

# 5. Other Considerations

## 5.1 Sizing the Zone Radius

The performance of the Zone Routing Protocol is determined by the routing zone radius. In general, dense networks consisting of a few fast moving nodes favor smaller routing zones. On the other hand, a sparse network of many slowly moving nodes operates more efficiently with a larger zone radius.

The simplest approach to configuring the routing zone radius is to make the assignment once, prior to deploying the network. This can be performed by the network administration, if one exists, or by the manufacturer, as a default value. This may provide acceptable performance, especially in situations where network characteristics do not vary greatly over space and time. Alternatively, the ZRP can adpat to changes in network behavior, through dynamnic configuration of the zone radius [Haas-3]. In [Haas-4], it was shown that a node can accurately estimate its optimal zone radius, on-line, based on local measurements of ZRP traffic. The re-sizing of the routing zone can be carried out by a protocol that conveys the change in zone radius to the members of the routing zone. The details of such an update protocol will be provided in a future version of this draft.

## 6.0 References

[AODV] Perkins, C.E., "Ad-Hoc On-Demand Distance Vector Routing", MILCOM'97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 1997

[Haas-1] Haas, Z.J., "A New Routing Protocol for the Reconfigurable Wireless Networks", ICUPC'97, San Diego, CA, Oct. 12,1997.

[Haas-2] Haas, Z.J. and Pearlman, M.R., "The Performance of Query Control Schemes for the Zone Routing Protocol," SIGCOMM'98, Vancouver, BC, Sept. 2-4, 1998.

[Haas-3] Haas, Z.J. and Tabrizi, S., "On Some Challenges and Design Choices in Ad-Hoc Communications", MILCOM'98, Boston, MA, October 18-21, 1998.

[Haas-4] Haas, Z.J. and Pearlman, M.R., "Determining the Optimal Configuration for the Zone Routing Protocol", submitted for journal publication.

[Johnson] Johnson, D.B., and Maltz, D.A., "Dynamic Source Routing in Ad-Hoc Wireless Networks," in Mobile Computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer, 1996.

[Murthy] Murthy, S., and Garcia-Luna-Aceves, J.J., "An Efficient Routing Protocol for Wireless Networks", MONET, vol. 1, no. 2, pp. 183-197, October 1996.

[Park] Park, V.D., and Corson, M.S., "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," IEEE INFOCOM'97, Kobe, Japan, 1997.

[Perkins] Perkins, C.E., and Bhagwat, P., "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers",ACM SIGCOMM, vol.24, no.4, October 1994.

[TORA] Macker, J., "Mobile Ad Hoc Internetworking", MILCOM'97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 1997.

[RFC-0791] Postel, J., Editor, "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC-1075] Waitzman, D., Partridge, C., Deering, S.E., "Distance Vector Multicast Routing Protocol",RFC 1075, Nov. 1, 1988.

[RFC-2178] Moy, J., "OSPF Version 2", INTERNET DRAFT STANDARD, RFC 2178, July 1997.

Authors' Information

Zygmunt J. Haas
Wireless Networks Laboratory
323 Frank Rhodes Hall
School of Electrical Engineering
Cornell University
Ithaca, NY 14853
United States of America
tel: (607) 255-3454, fax: (607) 255-9072
Email: haas@acm.org or haas@ee.cornell.edu

Marc R. Pearlman
319 Frank Rhodes Hall
School of Electrical Engineering
Cornell University
Ithaca, NY 14853
United States of America
Email: pearlman@ee.cornell.edu

The MANET Working Group contacted through its chairs:

M. Scott Corson
Institute for Systems Research
University of Maryland
College Park, MD 20742
(301) 405-6630
corson@isr.umd.edu

Joseph Macker
Information Technology Division
Naval Research Laboratory
Washington, DC 20375
(202) 767-2001
macker@itd.nrl.navy.mil