PROVIDING AD-HOC CONNECTIVITY WITH RECONFIGURABLE WIRELESS NETWORKS[1]

Zygmunt J. Haas and Marc R. Pearlman
School of Electrical Engineering, Cornell University, Ithaca, NY, 14853
Tel: (607) 255-3454, fax: (607) 255-9072,
e-mail: {haas,pearlman}@ee.cornell.edu
URL: http://www.ee.cornell.edu/~haas/wnl.html

Abstract
In this paper, we introduce a novel routing protocol for a special class of ad-hoc networks, termed by us the Reconfigurable Wireless Networks (RWNs). The main features of RWNs are: increased mobility of the network nodes, larger number of nodes, and a large network span. We argue that the current routing protocols do not provide a satisfactory solution for routing in this type of an environment. We propose a scheme, called the *Zone Routing Protocol (ZRP),* which dynamically adjusts itself to the operational conditions by sizing a single network parameter – the *Zone Radius*. More specifically, ZRP reduces the cost of frequent updates of the constantly changing network topology by limiting the scope of the updates to the immediate neighborhood of the change. We study the performance of the scheme, evaluating the average number of control messages required to discover a route within the network. Furthermore, we compare the scheme's performance with reactive flooding search and proactive link-state classes of routing protocols.

1. Introduction

1.1 The Notion of the Reconfigurable Wireless Networks

A Reconfigurable Wireless Network (RWN) is an ad-hoc network architecture that can be rapidly deployed without relying on preexisting fixed network infrastructure. The nodes in a RWN can dynamically join and leave the network, frequently, often without warning, and possibly without disruption to other nodes' communication. Finally, the nodes in the network can be highly mobile, thus rapidly changing the node constellation and the presence or absence of links. Examples of the use of the RWNs are:

- tactical operation - for fast establishment of military communication during the deployment of forces in unknown and hostile terrain;
- rescue missions - for communication in areas without adequate wireless coverage;
- national security - for communication in times of national crisis, where the existing communication infrastructure is non-operational due to a natural disaster or a global war;
- law enforcement - for fast establishment of communication infrastructure during law enforcement operations;
- commercial use - for setting up communication in exhibitions, conferences, or sales presentations;
- education - for operation of wall-free (virtual) classrooms; and
- sensor networks - for communication between intelligent sensors (e.g., MEMS[2]) mounted on mobile platforms.

Nodes in the RWN exhibit nomadic behavior by freely migrating within some area, dynamically creating and tearing down associations with other nodes. Groups of nodes that have a common goal can create formations (clusters) and migrate together, similar to military units on missions or to guided tours on

---

[1] This work is supported by the US Air Force/Rome Labs, under the contract number C-7-2544 and a grant from Motorola Corporation, the Applied Research Laboratory
[2] **M**icro-**E**lectro-**M**echanical-**S**ystems

excursions. Nodes can communicate with each other at any time and without restrictions, except for connectivity limitations and subject to security provisions. Examples of network nodes are pedestrians, soldiers, or unmanned robots. Examples of mobile platforms on which the network nodes might reside are cars, trucks, buses, tanks, trains, planes, helicopters or ships.

RWNS are intended to provide a data network that is immediately deployable in arbitrary communication environments and is responsive to changes in network topology. Because ad-hoc networks are intended to be deployable anywhere, existing infrastructure may not be present. The mobile nodes are thus likely to be the sole elements of the network. Differing mobility patterns and radio propagation conditions that vary with time and position can result in intermittent and sporadic connectivity between adjacent nodes. The result is a time-varying network topology.

RWNs are distinguished from other ad-hoc networks by *rapidly* changing network topologies, influenced by the network size and node mobility. Such networks typically have a large span and contain hundreds to thousands of nodes. The RWN nodes exist on top of diverse platforms that exhibit quite different mobility patterns. Within an RWN, there can be significant variations in nodal speed (from stationary nodes to high-speed aircraft), direction of movement, acceleration/deceleration or restrictions on paths (e.g. a car must drive on a road, but a tank does not. A pedestrian is restricted by built objects while airborne platforms can exist anywhere in some range of altitudes. In spite such volatility, the RWN is expected to deliver diverse traffic types, ranging from pure voice to integrated voice and image, and even possibly some limited video.

1.2 The Communication Environment and the RWN Model

The following are a number of assumptions about the communication parameters, the network architecture, and the network traffic in a RWN.

- Nodes are equipped with portable communication devices. These may be powered by lightweight batteries. Limited battery life can impose restrictions on the transmission range, communication activity (both transmitting and receiving) and computational power of these devices.
- Connectivity between nodes is **not** a transitive relation; i.e., if node *A* can communicate directly with node *B* and node *B* can communicate directly with node *C,* then node *A* **may not,** necessarily, be able to communicate directly with node *C.* This leads to as *the hidden terminal problem* [Tobagi85].
- A hierarchy in the network routing and mobility management procedures could improve network performance measures, such as the latency in locating a mobile. However, a <u>physical</u> hierarchy may lead to areas of congestion and is very vulnerable to frequent topological reconfigurations.
- We assume that nodes are identified by fixed IDs (based on IP [Postel81] addresses, for example).
- All the network nodes have equal capabilities. This means that all nodes are equipped with identical communication devices and are capable of performing functions from a common set of networking services. However, all nodes do not necessarily perform the same functions at the same time. In particular, nodes may be assigned specific functions in the network, and these roles may change over time.
- Although the network should allow communication between **any** two nodes, it is envisioned that a large portion of the traffic will be between geographically close nodes. This assumption is clearly justified in a hierarchical organization. For example, it is much more likely that communication will take place between two soldiers in the same unit, rather than between two soldiers in two different brigades.

A RWN is a *peer-to-peer* network that allows direct communication between any two nodes, when adequate radio propagation conditions exist between these two nodes and subject to transmission power limitations of the nodes. If there is no direct link between the source and the destination nodes, *multi-hop* routing is used. In multi-hop routing, a packet is forwarded from one node to another, until it reaches the destination. Of course, appropriate routing protocols are necessary to discover routes between the source and the destination, or even to determine the presence or absence of a path to the destination node. Because of the lack of central elements, distributed protocols have to be used.

The main challenges in the design and operation of the RWNs, compared to more traditional wireless networks, stem from the lack of a centralized entity, the potential for rapid node movement, and the fact that all communication is carried over the wireless medium. In standard cellular wireless networks, there are a number of centralized entities (e.g., the base-stations, the Mobile Switching Centers (MSC-s), the Home Location Registry (HLR), and the Visitor Location Registry (VLR)). In ad-hoc networks, there is no preexisting infrastructure, and these centralized entities do not exist. The centralized entities in the cellular networks perform the function of coordination. The lack of these entities in the RWNs requires distributed algorithms to perform these functions. In particular, the traditional algorithms for mobility management, which rely on a centralized HLR/VLR, and the medium access control schemes, which rely on the base-station/MSC support, become inappropriate.

All communications between all network entities in our ad-hoc networks are carried over the wireless medium. Due to the radio communications being vulnerable to propagation impairments, connectivity between network nodes is not guaranteed. In fact, intermittent and sporadic connectivity may be quite common. Additionally, as the wireless bandwidth is limited, its use should be minimized. Finally, as some of the mobile devices are expected to be hand-held with limited power sources, the required transmission power should be minimized as well. Therefore, the transmission radius of each mobile is limited, and channels assigned to mobiles are typically spatially reused. Consequently, since the transmission radius is much smaller than the network span, communication between two nodes often need to be relayed through intermediate nodes; i.e., multi-hop routing is used.

Because of the possibly rapid movement of the nodes and variable propagation conditions, network information, such as a route table, becomes obsolete quickly. Frequent network reconfiguration may trigger frequent exchanges of control information to reflect the current state of the network. However, the short lifetime of this information means that a large portion of this information may never be used. Thus, the bandwidth used for distribution of the routing update information is wasted. In spite of these attributes, the design of the RWNs still needs to allow for a high degree of reliability, survivability, availability, and manageability of the network.

Based on the above discussion, we require the following features for the RWNs:

- **Robust routing and mobility management algorithms** to increase the network's reliability and availability; e.g., to reduce the chances that any network component is isolated from the rest of the network
- **Adaptive algorithms and protocols** to adjust to frequently changing radio propagation, network, and traffic conditions
- **Low-overhead algorithms and protocols** to preserve the radio communication resource
- **Multiple (distinct) routes** between a source and a destination - to reduce congestion in the vicinity of certain nodes, and to increase reliability and survivability
- **Non-hierarchical physical network architecture** to avoid susceptibility to network failures, congestion around high-level nodes, and the penalty due to inefficient routing

2. The Zone Routing Protocol

In this paper, we concentrate on the issue of designing a routing protocol for the RWN. The proposed protocol, the *Zone Routing Protocol (ZRP)*, allows efficient and fast route discovery in the RWN communication environment (i.e., large geographical network size, large number of nodes, fast nodal movement, and frequent topological changes). It requires a small amount of routing information to be maintained at each node and the cost in wireless resources for maintaining routing information of inactive routes is limited as well. The protocol identifies multiple, loop-free routes to the destination, increasing reliability and performance. Routing is flat, rather than hierarchical, reducing organizational overhead, allowing optimal routes to be discovered and reducing the threat of network congestion. However, the most appealing feature of the protocol is that its behavior is adaptive, based on the mobility and calling patterns of the mobile users.

In what follows, we explain the elements of the proposed scheme. However, first, we clarify the difference between *reactive* and *proactive* routing schemes.

2.1 Reactive vs. Proactive Routing

The challenge in designing a routing protocol for the RWNs stems from the fact that, on one hand, to determine a packet route, at least the reachability information[3] for the source's neighbors needs to be known to the source node. On the other hand, in a RWN, this topology may 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, route updates, or routing tables) among the network nodes. Thus, the amount of update traffic is quite high. This is appropriate since all updates in the wireless communication environment travel over the air and use scarce bandwidth.

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. OSPF [Moy97], WRP [Murthy96] and DSDV [Perkins94] are examples of proactive schemes. On the other hand, reactive protocols, like TORA [Park97], AODV [Perkins99] and DSR [Johnson96], invoke a route determination procedure only on demand.

The advantage of 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 route request is received, the delay to determine a route can be quite significant. Furthermore, the global 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 RWN environment, as they continuously use a large portion of the network capacity to keep the routing information current. Since in a RWN nodes 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 again in an excessive waste of the network capacity. What is needed is a protocol that, on one hand, initiates the route-determination procedure on-demand, but at limited search cost.

[3] The reachability information indicates whether a destination node could be reached from the node in question, what is the next neighbor on that path, and what is the "cost" of the path. The "cost" may be based on different criteria, such as, for example, delay, number of hops, or traffic congestion along the path.

Our protocol, the *Zone Routing Protocol (ZRP)* [Haas98] [Pearlman99], is an example of a hybrid reactive/proactive routing protocol. On one hand, it limits the scope of the proactive procedure to the <u>node's local neighborhood.</u> On the other hand, the search throughout the network, although it is global, is done by querying only a subset of the network nodes. A related issue regards updates in the network topology. For a routing protocol to be efficient, changes in the network topology should have local effect only. In other words, creation of a new link at one end of the network is an important local event but, usually, 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. ZRP limits propagation of such information to the neighborhood of the change only, thus limiting the cost of topological updates. ZRP is based the notion of a *Routing Zone,* which we introduce next.

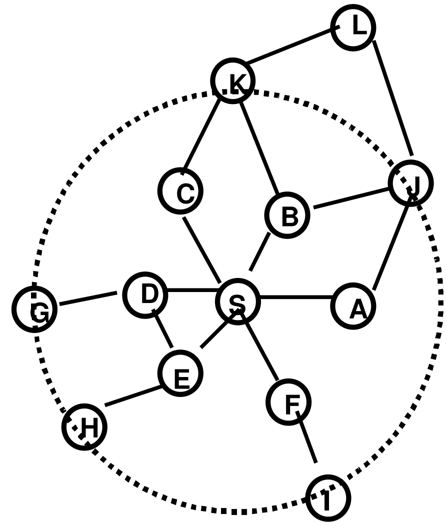


Figure 1: A Routing Zone of Radius 2 Hops

2.2 The Notion of a Routing Zone and Intrazone Routing

In ZRP, a node proactively maintains routes to destinations within a local neighborhood, which we refer to as a *routing zone*. More precisely, a node's routing zone is defined as a collection of nodes whose minimum distance (in hops) from the node in question is no greater than a parameter referred to as the *zone radius*.
An example of a routing zone (for node *S*) of radius 2 is shown in Figure 1. Nodes A through K are members of node S's routing zone. Node *L* lies outside *S*'s routing zone. *Peripheral nodes* are nodes whose minimum distance to the node in question is equal <u>exactly</u> to the zone radius. Nodes G-K are peripheral nodes, whereas nodes A-F are interior nodes. We note that each node maintains its own routing zone. As a result, routing zones of nearby nodes may overlap heavily.

Related to the definition of a zone is the coverage of a node's transmitter, which is the set of nodes that are in direct communication with the node in question. These nodes are referred to as *neighbors.* The transmitter's coverage depends on the propagation conditions, on the transmitter power, and on the receiver sensitivity. In our simulation, we define a radius, d_{xmit}, which is the maximal distance that a node's transmission will be received without errors. Of course, it is important that each node be connected to at least one other node. However, more is not necessarily better. As the transmitter's coverage include all the nodes with distance 1 hop from the node in question, the larger the d_{xmit} is, the larger is the population of its routing zone. A large routing zone requires a larger amount of update traffic. For the purpose of illustration, we will depict zones as circles around the node in question. However, one should keep in mind that the zone is not a description of distance, but rather node connectivity (hops).

Each node is assumed to maintain the routing information to all nodes that are within its routing zone and those nodes only. Consequently, in spite of the fact that a network can be quite large, the updates are only locally propagated. The node learns the topology of its routng zone trough a localized proactive scheme, which we refer to here as the *IntrAzone Routing Protocol (IARP).* In this paper, we use a basic link state algorithm, where links state updates are localized within the routing zone radius. Other link state and distance vector variants may be modified to serve as an IARP, by properly restricting the propagation of topology updates to the extent of a node's routing zone.

2.3 Interzone Routing and the *Zone Routing Protocol*

The IntErzone Routing Protocol (IERP) is responsible for reactively discovering routes to destinations located beyond a node's routing zone. IERP is distinguished from standard flooding-based query/response protocols by exploiting the structure of the routing zone. The routing zones increase the probability that a node can respond positively to a route request. This is beneficial for traffic that is destined for geographically close nodes. More importantly, knowledge of local routing zone topology can be exploited to efficiently relay a query through the network. This is achieved through a packet delivery service, called *bordercasting*, which directs messages from a node out to its peripheral nodes.

The nature of bordercasting lends itself to a multicast implementation. One approach is for a node to compute its bordercast (multicast) tree and append the corresponding packet forwarding instructions to the bordercast packet. Alternatively, each node may re-construct the bordercast tree of its interior routing zone members, by proactively maintaining the topology of an *extended* zone. In particular, if IARP maintains an extended zone of radius 2ρ-1 (where the "basic" routing zone radius is ρ), bordercast messages can be relayed without the need for explicit directions from the bordercast source.

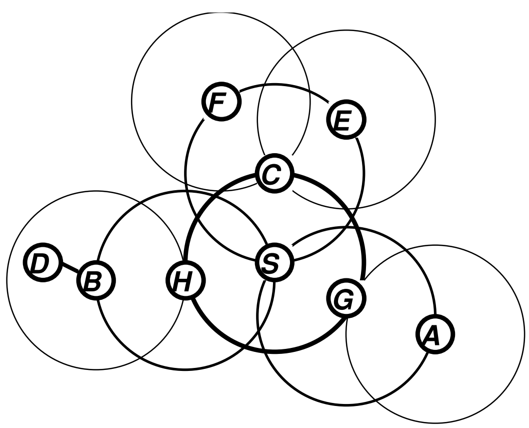


Figure 2: An example of IERP operation

IERP route discovery operates as follows: the source node first checks whether the destination is within its zone.[4] If so, the path to the destination is known and no further route discovery processing is required. If the destination is not within the source Routing Zone, the source bordercasts a *route request* to all its peripheral nodes.[5] Now, in turn, all the peripheral nodes execute the same algorithm: checking whether the destination is within their zone. If so, a *route reply* is sent back to the source, indicating the route to the destination (more about this in moment). Otherwise, the peripheral node forwards the route request to *its* peripheral nodes, which, in turn, execute the same procedure.

An example of this *Route Discovery* procedure is shown in Figure 2. The source node *S* sends a packet to the destination *D.* To find a route within the network, *S* first checks whether *D* is within its routing zone. Since D does not lie within S's routing zone, *S* bordercasts a route request to all of its peripheral nodes: that is, to nodes *C, G,* and *H.* Upon receiving the route request, nodes C,H and G determine that *D* is not in their routing zones and they therefore bordercast the request to their peripheral nodes. One of H's peripheral nodes, B, recognizes *D* as being in its routing zone and responds to the route request, indicating the forwarding path: *S-H-B-D.*

To complete the route discovery process, a reply is sent back to the route query's source, providing it with the desired route. It is the responsibility of the Route Accumulation procedure to acquire sufficient information during the route request phase so that the route response can be routed back to the source. In the basic Route Accumulation procedure, a node appends its address to a received route request packet. The sequence of 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, a route reply may

---

[4] Remember that a node knows the identity, distance to, and a route to all the nodes in its zone.

[5] Again, the identity of its zone peripheral nodes are known to the node in question.

be sent back to the source, through strict source routing. Given sufficient storage space, nodes may remove the routing information accumulated in the route request packet and store it in a temporary route query cache. This has the benefit of reducing the length of the request packet, thereby decreasing the query response time. When all nodes adopt this caching policy, the route reply can be relayed back to the source through next hop routing.

A nice feature of this distributed route discovery process is that a single route query can return multiple route replies. The quality of these returned routes can be determined based on hop count (or any other path metric accumulated during the propagation of the route request. The best route can be selected based on the relative quality of the route (i.e. choose the route with the smallest hop count, or shortest accumulated delay).

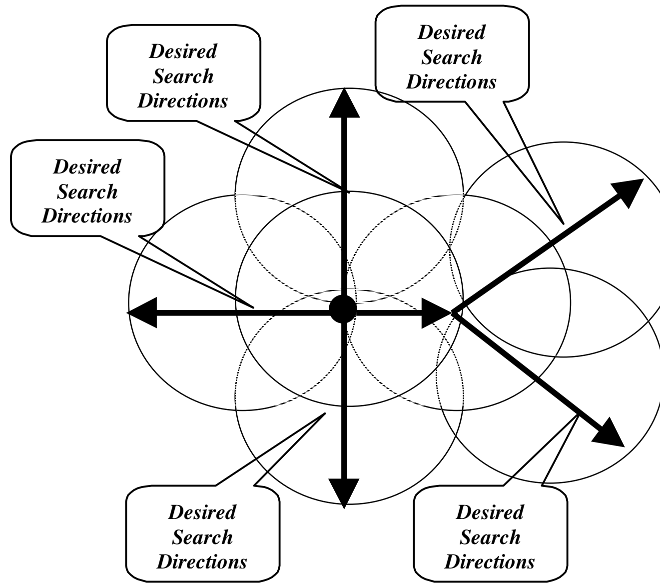


Figure 3: Guiding the search in desirable directions

The cost of route discovery can be significantly reduced by initiating a global route search only when there is a substantial change in the network topology. When a change in network topology causes a broken link within an active path, a <u>local</u> path repair procedure is initiated. The path repair procedure substitutes a broken link by a mini-path between the ends of the broken link. A path update is then generated and sent to the end points of the path. Path repair procedures tend to reduce the path optimality (e.g., increase the length for shortest path routing). Thus, after some number of repairs, the path end points will initiate a new Route Discovery procedure to replace the path with a new optimal one.

ZRP is based on the idea that querying can be performed more efficiently than flooding by directing route requests to target peripheral nodes. However, because neighboring routing zones heavily overlap, each node may forward a route request multiple times, resulting in more control traffic than flooding. The challenge is to “steer” the query outward from the original routing zone (see Figure 3), rather than returning to areas that were already covered by the query. This desired query propagation behavior is achieved by properly extending the query forwarding and termination strategies from traditional flooding algorithms to the context of a routing zone architecture.

Redundant querying occurs when a route request packet arrives in a previously queried routing zone. To prevent this situation from occurring, nodes should be able to detect when a routing zone that they belong to has been previously queried. Clearly, a node that bordercasts a route request is aware that its zone has been queried. If bordercasting is implemented below the routing protocol (i.e. by the network layer), then the query will not be detected again until it reappears at the target peripheral nodes. In order to notify the remaining routing zone nodes, without introducing additional control traffic, bordercasting needs to be performed by ZRP itself. The first level of Query Detection - QD1 - allows nodes to detect the queries as they relay the queries to the edge of the routing zone. In networks that use a single, broadcast channel, it may be possible for queries to be detected by any node within the range of a query-relaying node. This extended query detection capability - QD2 - can be implemented by using IP and MAC neighbor broadcast (or multicast) as a vehicle for relaying a route request to the downstream neighbors[6]. Figure 4 illustrates both levels of advanced query detection. In this example, node S bordercasts to two peripheral nodes: B and D. The intermediate nodes A and C are able to detect passing route request packets, and record that S's routing zone has been queried. In single channel networks, node E may also be able to receive A's transmission and record the query information as well.

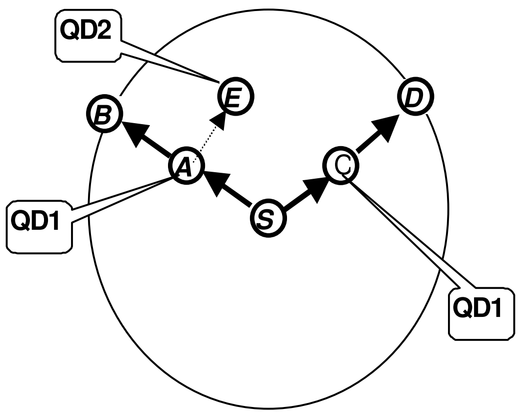


Figure 4: Advanced Query Detection (QD1/QD2)

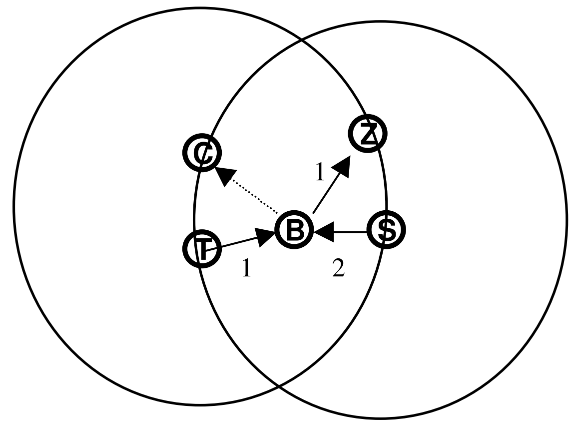


Figure 5: Early Termination (ET)

Standard flood search protocols terminate packets that are targeted for (or arrive at) previously queried nodes. We can extend this idea to ZRP by discarding route request packets before they arrive at bordercast recipients that belong to the routing zones previously queried nodes. More precisely, a node will not relay a query packet to a target bordercast recipient, if that potential recipient either lies *inside* the routing zone of a previously bordercast node, or if this node has already relayed the query to this potential recipient. This scheme, which we refer to as Early Termination (ET), relies on Query Detection to identify which local nodes have already bordercast the query. In order to identify the nodes that lie *inside* of the routing zones of these bordercast nodes, the topology of an extended zone of radius 2ρ-1 hops (where ρ is the radius of the "basic" routing zone) needs to be maintained. Conveniently, IARP already maintains this extended information in support of multicast-based bordercasting. Figure 5 illustrates the operation of ET. Node B first detects (and relays) a route request packet bordercast by node T. Later, B receives a route request packet to be relayed to the bordercast recipient node C. B recognizes that node C belongs to the previously queried routing zone of node T, and therefore withholds transmission.

3. ZRP - Formal Description

The placement of ZRP in the OSI protocol stack is shown in Figure 6. The proactive maintenance of the routing zone topology is performed by IARP, through exchange of route update packets. Route updates are triggered by the MAC-level NDP, which notifies IARP when a link to a neighbor is established or broken. IERP reactively acquires routes to nodes beyond the routing zone using a query-reply mechanism. IERP forwards queries to its peripheral nodes through the bordercast delivery service. IERP keeps track of the peripheral nodes through up-to-date routing zone topology information provided by IARP. IERP also makes use of IARP routing zone information to determine whether a queried-for-

[6] Even though all neighbors may receive a neighbor broadcast, only those neighbors that are designated by the ZRP as *downstream intermediate nodes* (i.e. nodes responsible for relaying a query packet to the target peripheral nodes) will process the query. The use of neighbor broadcast does not necessarily imply flood searching.

destination belongs to its routing zone. We will proceed now with formal definition of NDP, IARP, and IERP.

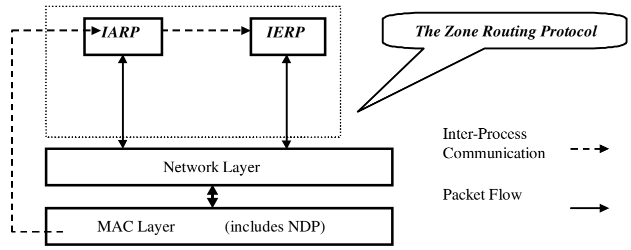


Figure 6: ZRP Architecture

3.1 Neighbor Discovery Protocol (NDP)

Nodes advertise their presence to their neighbors by periodically transmitting “HELLO” beacons. Upon receipt of a beacon, a node records the beacon source in its Neighbor Table. Each node scans its Neighbor Table at regular sampling intervals to check the status of each of its neighbors. If no beacon was received from a neighbor during the previous MAX_LAST_RECORDED sampling intervals, the neighbor is considered lost. If a beacon was received, and the neighbor was previously unrecorded, it is considered found. When a neighbor is either lost or found, IARP is notified of this new link status. The protocol is shown in Figures 7a-b.

3.2 The Intrazone Routing Protocol (IARP)

In this version of IARP (Figures 8a-c) nodes compute intrazone routes based on the link state of each (extended) routing zone node. A node may receive link state updates either from an IARP link state packet or from an interrupt generated by the Neighbor Discovery Protocol. Link states are maintained in a Link State Table. When all pending link state updates have been received (full link state updates may contain multiple links and span multiple packets), the routing table is recomputed, using a minimum spanning tree algorithm. The Link State Table is then updated to remove links that lie outside of the (extended) routing zone. Newly received link state updates for link sources inside the node's (extended) routing zone are forwarded to all the node's neighbors. In addition, if the node discovers a new neighbor, the new neighbor is sent the link states of all nodes that lie inside of the node's (extended) routing zone.

3.3 The Interzone Routing Protocol (IERP)

IERP (Figures 9a-e) is responsible for discovering routes to hosts that are beyond a node’s routing zone. A route request is triggered at the network layer, when a data packet is to be sent to a destination that is not recorded in its Routing Table. Each route request is assigned a query id that is unique to the source node. The combination of source node id and query id uniquely identifies any route query in the network. After recording the query source and query id in the request packet, the packet is bordercast to all peripheral nodes.

When a node receives a route request packet, the query id, query source, bordercasting node and previous hop are all recorded in the Detected Queries Table. The node then searches its Routing Table to see if the requested destination lies within its routing zone. If so, the node responds with a route reply returned to the query source, along a path specified by the previous hop information cached in the Detected Queries Table. If the destination does not belong to the node’s routing zone, the node forwards the route request down the bordercasting node's "bordercast tree" to the target peripheral nodes.

Without sufficient query control mechanisms, the bordercasting mechanism can result in far more traffic than flooding. A combination of advanced query detection (QD) and early query termination (ET) techniques are employed to realize the potential of zone based querying. Based on the information cached in the Detected Queries Table, a node will *not* relay a route request on an outgoing link if none of the intended bordercast recipients need to receive the route request packet. Specifically, a node can refrain from forwarding a route request to a bordercast recipient if the bordercast recipient lies inside of a previously queried routing zone <u>or</u> if the current node has already relayed a query packet to that bordercast recipient.

In addition to discovering new routes, IERP is also able to repair broken routes. When a broken link is detected, IERP is immediately notified, and a link repair is attempted (by means of a route discovery). Because it is very likely that the lost host is just beyond the routing zone, the depth of a link repair route discovery can be restricted. Upon receipt of link repair route reply, all routes containing the broken links can be updated. If no route reply is received within a reasonable amount of time, all routes containing the broken link are removed.

When the network layer detects invalid links in the routes of incoming data packets, they are re-routed based on the local route repairs. While this route repair process could be transparent to the packet source, hiding this information may result in a costly route selection by the packet source. To address this problem, IERP sends a link repair report to the packet source. The packet source can then update its routing tables accordingly.

Packet Format

8	16	24	32
source_node *(node_id)*			

Structures

Neighbor_Table

Neighbor (*node id*)	Arrival (*boolean*)	Last_recorded (*int*)

Initialization

beacon_xmit_timer = rand_uniform (2 $\cdot T_{beacon}$);

table_update_timer = rand_uniform (2 $\cdot T_{beacon}$) ;

Beacon Transmission

// Transmit periodic HELLO beacon
source_node = my_id;
load(packet); *ˆ/* load packet fields with local variables of the same name */*
broadcast(packet);
beacon_xmit_timer += T_{beacon} ;

Beacon Reception

// Record detected neighbor (from received beacon) in Neighbor Table
extract(packet); */* extract packet fields into local variables of the same name */*
if (source_node $\notin$ Neighbor_Table)
Neighbor_Table[source_node].Last_recorded = -1;
Neighbor_Table[source_node].Arrival = TRUE;

Figure 7a: The Neighbor Discovery Protocol (NDP)

Neighbor Table Update

```
// Identify lost neighbors and remove them from the Neighbor Table
for each neighbor in Neighbor_Table
{
        if (Neighbor_Table[neighbor].Arrival == FALSE)

           if (Neighbor_Table[neighbor].last_recorded ≥ MAX_LAST_RECORDED)
           {
                  // If a neighbor's beacon hasn't been received in MAX_LAST_RECORDED update cycles
                  // then remove neighbor from Neighbor Table
                  remove (Neighbor_Table[neighbor]);
                  load_intrpt_params(neighbor);
                  set_intrpt( IARP, "Neighbor Lost", "Update Intrazone Routing Table");
           }
           else
           {
                  // Increment number of cycles that neighbor's beacon has not been received
                  Neighbor_Table[neighbor].Last_recorded++;
           }
        }
        else
        {
           // If a new neighbor was found, alert the IARP
           if (Neighbor_Table[neighbor].Last_recorded == -1)
           {
                  load_intrpt_params(neighbor);
                  set_intrpt( IARP, "Neighbor Found", "Update Intrazone Routing Table");
                  Neighbor_Table[neighbor].last_recorded = 0;
           }
           Neighbor_Table[neighbor].arrival = FALSE;
        }
        table_update_timer += T_beacon;
}
```

Figure 7b: The Neighbor Discovery Protocol (NDP)

Packet Format

<table>
<tr><td colspan="2">8</td><td colspan="8">16</td><td>24 ... 32</td></tr>
<tr><td colspan="11">link_source
(node_id)</td></tr>
<tr><td colspan="11">link_dest
(node_id)</td></tr>
<tr><td colspan="11">pk_source
(node_id)</td></tr>
<tr><td colspan="2">link_state_id
(unsigned int)</td><td>F</td><td>C</td><td>S</td><td></td><td></td><td></td><td></td><td></td><td>RESERVED</td></tr>
</table>

Flags
F = full_update
A = updates_complete
S = link_status

Structures

Link State Table

Link Source (*node_id*)	Link Dest (*node_id*)	Link State ID (unsigned int)

Pending Link States

Pk Source (*node_id*)	Link Source (*node_id*)	Link Dest (*node_id*)	Link State Id (*unsigned int*)	Full Update (*boolean*)	Link Status (*boolean*)

Routing Table

Dest (*node_id*)	Is_zone_member (*boolean*)	Source_routes (*node_id list*)

Figure 8a: Intrazone Routing Protocol (IARP)

Update Intrazone Routing Table

```
// IARP may be triggered by either a link state packet update or an interrupt from the NDP
if (packet arrived)
{
    extract(packet);            /* extract packet fields into local variables of the same name */
    my_link_changed = FALSE;
}
else
{
    extract_intrpt_params(&link_dest);
    link_source        = my_id;
    pk_source          = my_id;
    link_state_id      = my_link_state_id;
    full_update        = FALSE;
    updates_complete   = TRUE;
    if (type(intrpt) == "Neighbor Found")
    {
        link_status = UP;
        // Send all link state information that lies inside my (extended) routing zone to each new neighbor
        if(!is_neighbor(Link_State_Table, my_id ,link_dest) )
        {
            send_link_state_table(Link_State_Table,  link_dest);
            my_link_state_id++;
        }
    }
    else
    {
        link_status = DOWN;
        if(is_neighbor(Link_State_Table, my_id, link_dest))
            my_link_state_id++;
    }
}

// Buffer link state updates until all updates from pk_source have arrived
add_to_Pending_Link_States(pk_source, link_source, link_dest, link_state_id, full_update, link_status);

// Once all link state updates have arrived from pk_source, transfer those link states  to the Link_State_Table
if(updates_complete)
{
    link_states_list  = remove_from_Pending_Link_States(pk_source);
    for each link_state ∈ link_states_list
        add_to_Link_State_Table(link_state);
}
```

Figure 8b: Intrazone Routing Protocol (IARP)

```
// The link state table should not contain links that lie beyond my (extended) routing zone.
// The routing table is recomputed until all outlying links have been removed from the routing table
rebuild = TRUE;
while (rebuild)
{
        Routing_Table = construct_spanning_tree(Link_State_Table);
        rebuild = prune(Link_State_Table);
}

// Broadcast all new link state information for link sources that lie inside my (extended) routing zone
broadcast_new_link_state_updates(Link_State_Table);
```

Figure 8c: Intrazone Routing Protocol (IARP)

Packet Format

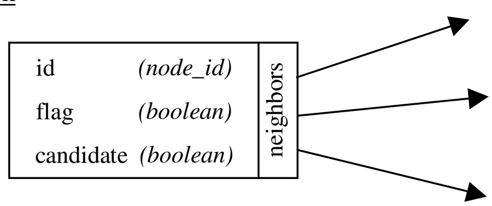
8	16	24	32
query_source *(node_id)*			
query_dest *(node_id)*			
prev_hop *(node_id)*			
bordercast_from *(node_id)*			
query_id *(unsigned int)*	pk_type *(char)*	link_marker *(unsigned int)*	max_hops *(unsigned int)*
source_route [node 0] *(node_id)*			
source_route [node 1] *(node_id)*			
↓			
source_route [node N] *(node_id)*			

Structures

Routing Table (see IARP description)

Link State Table (see IARP description)

Net Graph



Detected Queries Table

Source (*node_id*)	Query_id (*unsigned int*)	Prev_hops (*node_id list*)	Coverage (*Net Graph*)

Figure 9a: Interzone Routing Protocol (IERP)

```
Initiate Route Discovery
        extract_intrpt_params(&query_dest);
        pk_type              = ROUTE_REQUEST;
        query_id             = my_query_id++;
        query_source         = my_id;
        bordercast_from      = my_id;
        prev_hop             = NULL;
        source_route         = NULL;

        // the path repair is a route query with limited depth
        if(type(intrpt)= = "Path Repair")
        {
            max_hops         = MAX_REPAIR_HOPS;
            link_marker      = 0;
        }
        else
        {
            max_hops         = MAX_REQUEST_HOPS;
            link_marker      = NULL;
        }
        // bordercast route request to all peripheral nodes
        load(packet);                 /* load packet fields with local variables of the same name */
        load_intrpt_params(packet);
        set_intrpt( IERP, "Receive IERP Packet", " ");


Receive IERP Packet
        extract(packet);              /* extract packet fields into local variables of the same name */
        route_length = length(source_route);
        switch(pk_type)
        {
            case:   ROUTE_REQUEST
                // If a route to the query destination is known, forward route reply back to query source
                if (dest ∈ Routing_Table)
                {
                    source_route = get_route(Routing_Table, query_dest).;
                    pk_type = ROUTE_REPLY;
                    load(packet);                 /* load packet fields with local variables of the same name */
                    send(packet,  prev_hop);
                }
                else
                {
                    // If this is the first received copy of this query, add it to the Detected Queries Table,
                    // along with a copy of my extended routing zone graph (for recording the detected
                    / /coverage of this query)
                    if(! ∃ Detected_Queries_Table[query_source, query_id] )
                    {
                        add_to_Detected_Queries_Table(query_source,query_id);
                        t = Detected_Queries_Table[query_source, query_id];
                        t.coverage = create_graph_from_Link_State_Table(Link_State_Table);
                    }
```

Figure 9b: Interzone Routing Protocol (IERP)

```
            t = Detected_Queries_Table[query_source, query_id];

            // Add query packet's previous hop to the Detected_Queries_Table
            add_to_NodeID_List(&t.Prev_hops, source_route[prev_hop]);

            // Mark all nodes within bordercast_from's routing zone as having been covered
            // This is achieved simply by a depth-first search of the t.coverage graph, starting at
            // bordercast_from, and limited to a depth of zone_radius - 1
            mark_interior_nodes as_covered(t.coverage, bordercast_from);

            // If I am a peripheral node of bordercast_from, then I will re-bordercast the query
            if(is_peripheral_node(bordercast_from, my_id))
            {
                bordercast_from = my_id;
                mark_as_covered(t.coverage, bordercast_from, zone_radius-1);
            }

            // Determine the neighbors to whom I should relay the query
            downstream_neighbors = get_downstream_neighbors(t.coverage, bordercast_from);

            prev_hop = my_id;
            load(packet);
            for each neighbor ∈ downstream_neighbors
                send(packet, neighbor);
        }
    break;

    case:   ROUTE_REPLY
    case:   PATH_REPAIR_REPORT
        if ( pk_type == PATH_REPAIR_REPORT )
        {
            // replace broken link in routing table with path repair
            bad_link = source_route[link_marker] + query_dest;
            path_repair = source_route[link_marker : length(source_route)-1];
            replace(Routing_Table, bad_link, path_repair);
        }
        else
        {
            // record route in routing table
            add (Routing_Table[query_dest], route);
        }

        if (query_source ≠ my_id )
        {
            // look up the next hop back to the query source (from the Detected Queries cache)
            // and forward reply back to query source
            prev_hop = Detected_Queries_Table[query_source, query_id].prev_hops[0];
            source_route = prev_hop + source_route;
            load(packet);           /* load packet fields with local variables of the same name */
            send(packet, prev_hop);
        }
    break;
```

Figure 9c: Interzone Routing Protocol (IERP)

```
node_id list   get_downstream_neighbors (Net_Graph graph, node_id bordercast_from)
{
      root_node = find_node(graph, bordercast_from);
      bordercast_downstream(root_node, 0, my_downstream_neighbors);
      clear_visited_flags(root_node);

      return(my_downstream_neighbors);
}

boolean    bordercast_downstream (Net_Graph node, int depth,  node_id list downstream_neighbors)
{
      insert  = FALSE;

      if(!node.visited)
      {
         // mark nodes to prevent looping in this depth-first search
         node.visited = TRUE;

         if(depth == zone_radius)
         {
               // If this peripheral node is a candidate bordercast recipient, then the query should be relayed to it.
               if(!node.covered)
               {
                     node.candidate = FALSE;
                     insert = TRUE;
               }
         }
         else
         {
               // Nodes inside of the routing zones of previously bordercast nodes are not candidate bordercast recipients
               node.candidate = FALSE;

               // Determine which neighbors of node.id have downstream bordercast recipients.
               // If at least one neighbor has a downstream bordercast recipient, then this node is part
               // of the root's bordercast tree.
               for each neighbor ∈ node.neighbors
               {
                     if(bordercast_downstream(neighbor, depth+1, my_downstream_neighbors))
                     {
                           insert = TRUE;
                           // If this node is my node, then record the neighbors which belong the root's bordercast tree
                           if(node_id == my_id)
                                 add_to_node_id_list(my_downstream_neighbors, neighbor.id);
                     }
               }
         }
      }
      return(insert);
}
```

Figure 9d: Interzone Routing Protocol (IERP)

```
Report Path Repair
        extract_intrpt_params(&route);

        // locate "my_id's" broken link in source route [my_id, next_hop]
        last_hop = 0;
        while (route[last_hop] ≠ my_id)
            last_hop++;
        next_hop = route[last_hop+1];

        // replace broken link with new segment from Routing Table
        repaired_path = Routing_Table[next_hop].route(0);

        // relay route repair report back to source, through all nodes along the reverse route
        source_route = route[0:last_hop] + repaired_path;
        pk_type  = PATH_REPAIR_REPORT;
        link_marker  = last_hop;
        load(packet);               /* load packet fields with local variables of the same name */
        send(packet, route[last_hop-1]);
```

Figure 9e: Interzone Routing Protocol (IERP)

4. Evaluation of ZRP

We use the OPNET™ Network Simulator from MIL3, an event driven simulation package, to evaluate the performance of ZRP over a range of routing zone radii (ρ), from reactive flood-search routing (ρ=1) to proactive link state routing ($\rho \rightarrow \infty$). Performance is gauged by measuring the control traffic generated by ZRP and its effects on the average session delay. Our results can be used to determine the optimum ZRP routing zone radius for a given nodal velocity and for a given route query rate.

Control traffic includes intrazone route update packets and the interzone route request/reply/failure packets. While the neighbor discovery beacons could be considered control overhead, this additional traffic is independent of both mobile velocity and routing zone radius. Furthermore, the neighbor discovery process is not an exclusive component of ZRP; various MAC protocols are also based on neighbor discovery. As such, the beacons do not contribute to the relative performance of ZRP and are not accounted for in our analysis. We measure the average arrival rate of both the interzone and intrazone control packets separately, and then add the two to produce the total average arrival rate of control packets.

A meaningful measure of ZRP delay is the *average route query response time* ($\overline{T}_{rqr}$), which is defined as the average duration from the time a route is *initially*[7] requested by the *Network* layer until the route is discovered. If the destination appears in the routing tables (which will occur with probability $(1-\text{Prob[route discovery]})$), the query is immediately answered and the route query response time is assumed to be zero.[8] Otherwise, a route discovery is required (which will occur with probability $(\text{Prob[route discovery]})$) and the route query response time is measured as the time elapsed between the generation of the route request and the reception of the first route reply, $T_{route-reply}$.

$$\begin{aligned}\overline{T}_{rqr} &= \left(1-\text{Prob}\left[\text{route discovery}\right]\right)\cdot 0 + \text{Prob}\left[\text{route discovery}\right]\cdot \overline{T}_{route-reply} \\ &= \text{Prob}\left[\text{route discovery}\right]\cdot \overline{T}_{route-reply}\end{aligned}$$

For a fixed network size and fixed nodal density, the probability of a route discovery for an initial query is only dependent on the routing zone radius. The behavior of the route reply time is far more complicated. Not only is it dependent on the arrival rate of control packets, it is also affected by such factors as the network traffic load and the average length of IERP control packets. Our study provides some insight into the effect of these factors on the ZRP delay.

Our simulated RWN consists of 200 mobile nodes, whose initial positions are chosen from a uniform random distribution over an area of 1000 [m] by 1000 [m]. Each node j moves at a constant speed, *v*, and is assigned a new direction[9], θ_j, uniformly distributed between 0 and 2π. When a node reaches the edge of the simulation region, it is reflected back into the coverage area, by setting its direction to $-\theta$ (horizontal edges) or $\pi-\theta$ (vertical edges). The magnitude of the velocity is not altered.

---

[7] This delay metric does not reflect the delays associated with subsequent route repairs. We assume here that routes can be adequately repaired through the local route repair procedure described earlier. These *limited depth* queries produce much less control traffic and much lower delays compared with the initial full depth query.

[8] For simplicity, we assume that the local processing time (e.g., table lookup) is negligible, compared with transmission delays.

[9] Direction is measured as an angle relative to the positive x-axis.

The duration of each simulation is 125 seconds. No data is collected for the first 5 seconds of the simulation to avoid measurements during the transient period and to ensure that the initial intrazone route discovery process stabilizes.

In order to measure the delay resulting only from the ZRP overhead, the network load is assumed to be low. Route failures are detected and acted upon. The route queries are generated according to a Poisson arrival process, with the arrival rate being a simulation parameter. The route queries represent both the initial query performed at the beginning of a session and subsequent queries due to reported route failures. Each route query is for a destination selected from a uniform random distribution of all other nodes in the network. Since the average time between a node's query for the *same* destination is longer than the expected interzone route lifetime, discovered interzone routes are effectively used only once and then discarded.

In the absence of a packet collision, we assume that background channel interference and receiver noise limit the transmission range of packets and busy tones to a physical radius of d_{xmit} = 500 [m]. Within a range of d_{xmit}, the average power of the desired signal (and resulting average SIR) rapidly increase to support reliable packet transmission. As significant improvements can be realized through the addition of error control coding, we approximate the rapid increase of packet reliability by a simple threshold packet delivery model. *Once access to the channel has been established*, a packet can be delivered (error-free) to any receiver within d_{xmit} from the transmitting node. Receivers farther than d_{xmit} from the transmitting node will not receive the packet.

In single channel networks, nodes contend for the channel based on the Dual Busy Tone Multiple Access (DBTMA) protocol [Deng98]. Prior to transmitting a data packet, a node secures access to the channel through an RTS/CTS handshake (performed on a separate control channel). After completing the RTS/CTS handshake, the transmitter sends the data packet, while simultaneously activating a *transmit busy tone*. The intended receiver, in turn, activates a separate *receive busy tone* as soon as this data transmission is detected. The dual busy tones are used to block attempts by neighboring nodes to access a channel already in use. In particular, the transmit busy tone prevents neighbors of the transmitter from accepting incoming RTS requests. Likewise, the receive busy tone prevents the receiver's neighbors from initiating the RTS/CTS handshake. This effectively prevents the "hidden terminal problem" associated with wireless channel access. In addition, DBTMA inherently avoids the "exposed terminal problem", by permitting neighboring nodes to transmit data simultaneously to different (and available) receivers.

In contrast to the single channel networks, we assume that channel access in our multiple channel networks are contention free. The underlying media access control is responsible for assigning each incoming/outgoing link a locally unique channel (to avoid channel contention). Although there are no packet collisions, retransmissions are still possible, as a receiving node may be busy receiving or transmitting another packet.

In our model, neighbor discovery is based on the reception of HELLO beacons that are broadcast at the MAC layer. These short beacons (containing only source address) are transmitted at random intervals of mean T_{beacon} = 0.2 seconds. Neighbor connectivity is determined by the reception of the HELLO beacons. If a new beacon fails to arrive within $2 \cdot T_{beacon}$ of the most recent beacon, a link failure is reported. Because the links are bi-directional, the need for a more complex HELLO→I-HEAR-YOU packet exchange is eliminated. Furthermore, we assume that neighbor discovery are given highest transmission priority and are not destroyed by collisions. This prevents inaccurate reporting of link failures for the allowed $2 \cdot T_{beacon}$ window.

Fixed Simulation Parameters

Parameter	Symbol	Value
Number of nodes	N	200
Network coverage area	A	1000 m x 1000 m
Transmission radius	d_{xmit}	105 m
Beacon period	T_{beacon}	0.2 sec
Transmission rate	R_{xmit}	1.0 Mbps

5. Performance Results

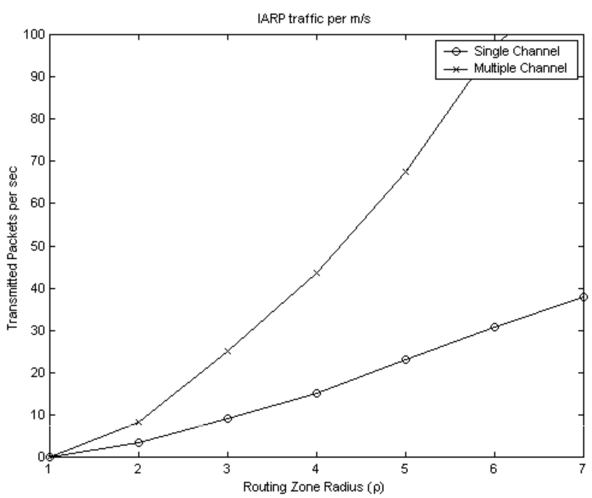


Figure 10

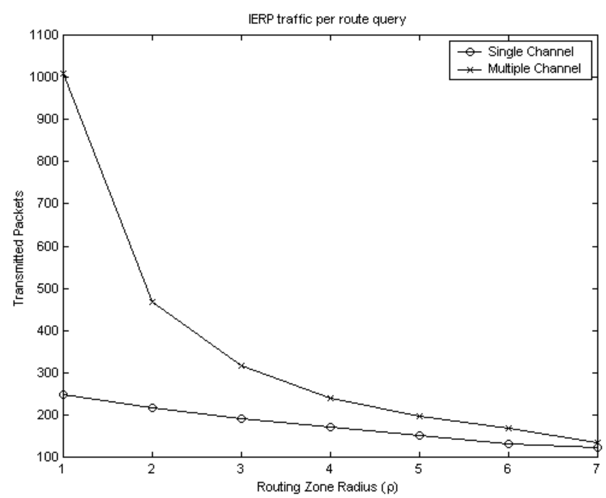


Figure 11

Results of our simulation are presented in the following figures. Figure 10 illustrates the dependence of proactive Intrazone Routing Protocol (IARP) control traffic on the routing zone radius, ρ. All else being equal, the rate at which the network reconfigures increases linearly with the speed of the nodes. Therefore, we express the amount of IARP traffic produced by each node, for each m/s of node mobility. Because each node needs to construct the bordercast trees for its interior nodes, a routing zone radius of ρ hops actually corresponds to an exchange of link state information over a range of 2ρ -1 hops. For unbounded networks with a uniform distribution of nodes, we expect the amount of intrazone control traffic to be $O(\rho^2)$ However, because our network is of finite size, the resulting boundary effect makes this dependence less than ρ^2. It should be noted that there is no intrazone control overhead for ρ = 1. For ρ = 1, the routing zone and extended routing zone both have a one hop radius and consist solely of neighbor nodes. Therefore, Neighbor Discovery Protocol provides all of the information needed to maintain connectivity within the routing zone. As we might expect, significantly more packet transmissions are required for the multiple channel networks, as the MAC level broadcast used to advertise link state information to neighbors must be replaced with individual unicasts to each neighbor.

The reactive IntErzone Routing Protocol (IERP) performance is demonstrated by the average traffic produced by each route query. In the case of ρ = 1, the peripheral nodes are neighbors. Consequently the IERP route query operates as a basic flood search. As we increase the routing zone radius, the reactive route discovery operates with increased efficiency. This improvement is due to the query detection (QD) and early termination (ET) query control mechanisms, which combine to guide the route requests outward from previously queried regions. For multiple channel networks, IERP can control which nodes relay route requests, and on which outgoing links the requests are forwarded. A routing zone radius of just 2 hops can produce 50% less route query traffic than a simple flood search. The improvement is less dramatic for single channel networks, since each node has only a single (broadcast) outgoing link. In these networks, a zone radius of 2 hops reduces the amount of route discovery traffic, but by only 15%.

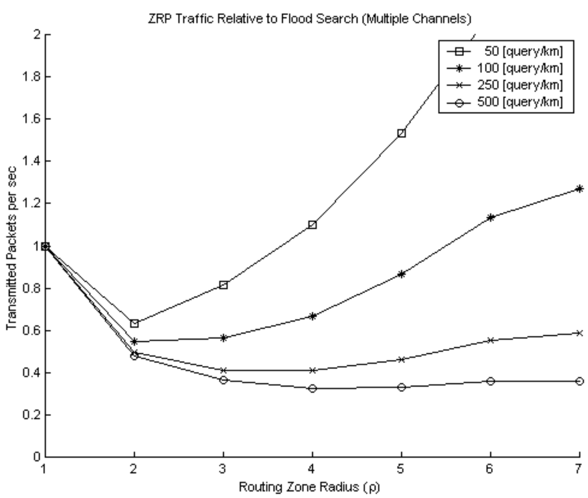


Figure 12a

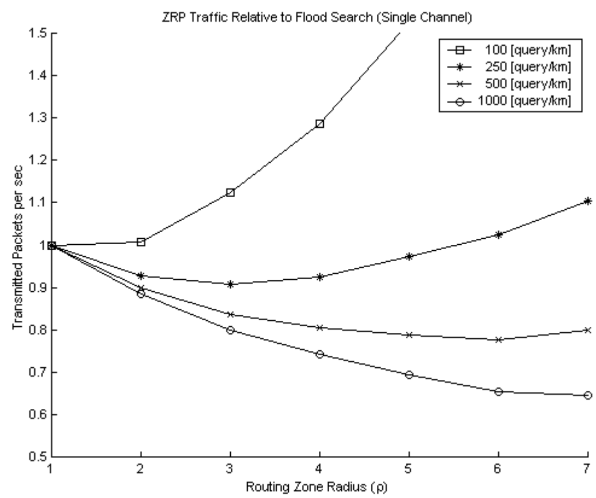


Figure 12b

The total ZRP control traffic (i.e., the sum of the control packets from the intrazone and interzone protocols), depicted in Figures 12a and 12b, gives an indication of the performance of our hybrid routing scheme. The amount of control traffic depends on both node mobility and route query rate. However, we can characterize the *relative* amounts of traffic based on the call to mobility ratio (CMR) (measured as the ratio of route query rate to node speed). As the CMR increases, the ZRP configuration favors larger routing zone radii. For large CMR scenarios, node mobility is relatively low and the cost of maintaining larger routing zones is justified by the resulting reduction in route discovery traffic. In contrast, lower CMR corresponds to relatively higher mobility scenarios, where routing zone maintenance becomes more costly. In the context of these simulated networks, a ZRP configuration of ρ = 1 (equivalent to flood searching) is appropriate for CMR below 15 [query/km] (multiple channels) or 100 [query/km]. For larger CMR, routing zones provide an improvement in the overall amount of routing traffic. For example, an optimally configured ZRP for a CMR of 500 [query/km], produces 70% less routing traffic than flood searching for multiple channel networks, and 20% less traffic than flood searching for single channel networks. As another example, we note that for multiple channel networks with a CMR of 100 [query/km], a routing zone radius of 2 hops produces 40% less routing traffic than flood searching (ρ = 1) and more than 50% less traffic than a purely proactive link state routing protocol ($\rho \rightarrow \infty$).

Figures 13a and 13b shows the performance of ZRP as measured by the average route query response time. For low network load, the delay characteristics appear to be heavily influenced by the traffic generated by the present route query. As the network topology becomes more volatile and route demand increases, the overall ZRP control traffic becomes large and begins to have a noticeable impact on the *instantaneous* network load. The *instantaneous* network load is generally more reactive than the *average* network load (due to the strong contribution of the current route query). The route query response time is therefore minimized by a routing zone radius that is at least as large as zone radius that minimizes overall ZRP control traffic. Neglecting the effects of additional data traffic, we find that ZRP can provide routes, on demand, in about half the time of flood search routing protocols. When optimizing ZRP for control traffic, some delay performance is sacrificed, but the ZRP response time is still noticeably less than the flood search response time

Increased route demand and node mobility contribute to additional routing protocol overhead traffic. In spite of the benefits that zone based querying outperforms traditional proactive and reactive routing protocols, there are limits to what can be achieved by routing protocols in the RWN environment. Given the network's bandwidth limitations, it may not be possible to support routing in extremely dynamic networks. However, the improvements provided by ZRP help to push this threshold higher.

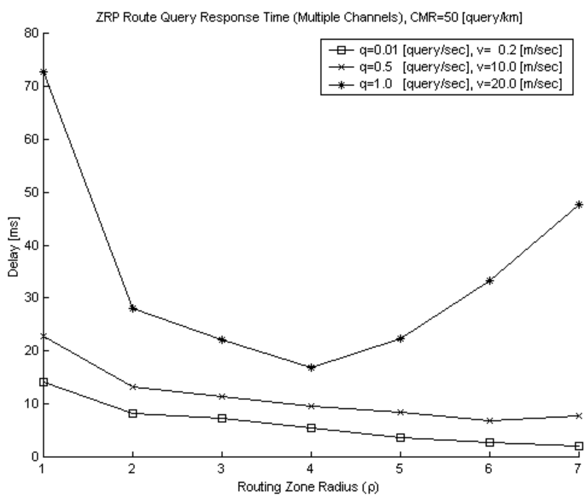


Figure 13a

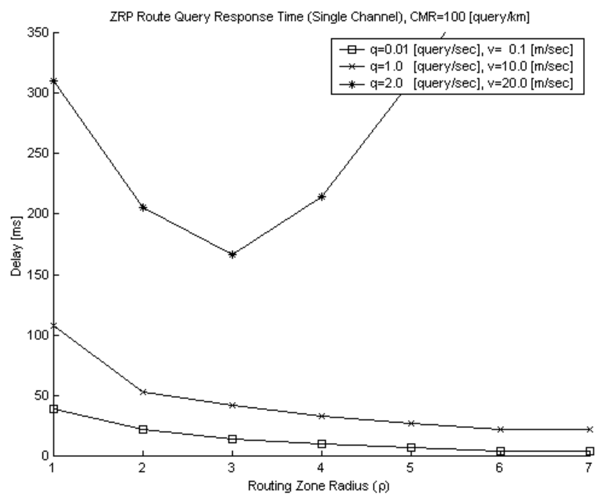


Figure 13b

6. Concluding Remarks

The Zone Routing Protocol (ZRP) provides a flexible solution to the challenge of discovering and maintaining routes in the Reconfigurable Wireless Network communication environment. ZRP combines two radically different methods of routing into one protocol. Interzone route discovery is based on a reactive route request/route reply scheme. By contrast, intrazone routing uses a proactive protocol to maintain up-to-date routing information to all nodes within its routing zone.

The amount of intrazone control traffic required to maintain a routing zone increases with the size of the routing zone. However, through a mechanism that we refer to as bordercasting, we are able to exploit the knowledge of the routing zone topography to significantly reduce the amount of interzone control traffic. For networks characterized by highly mobile nodes and very unstable routes, the hybrid proactive-reactive routing scheme ($\rho > 1$) produces less average total ZRP control traffic than purely reactive flood-search routing protocols ($\rho = 1$). Purely reactive schemes appear to be more suitable for networks with greater route stability. Furthermore, for highly active networks (frequent route requests), more proactive networks produce less overhead (i.e., larger routing zones are preferred).

We note that for networks with low activity, the instantaneous network load is generally dominated by the control traffic from a single route discovery. Consequently, ZRP exhibits minimum delay for relatively large routing zone radii, even for cases where reactive routing minimizes the average ZRP control traffic. By minimizing the amount of routing control traffic, ZRP can provide routes about 1.5 - 2.0 times faster than flood search routing protocols.

7. Appendix: Correctness Proof of ZRP

The following proofs demonstrate the correctness of the Zone Routing Protocol (ZRP). For analysis of the route query mechanism, we assume that the network topology remains static during the propagation of the query, and that IARP has already converged to prior topology changes.

These proofs indirectly apply to ZRP enhancements such as route caching and local route repair. It is true that cached routes are not guaranteed to be valid and limited depth route repair may not locate be successful. However, failure of these mechanisms will result in a new full depth route query, which discovers a route to any reachable network node. An important feature of this proof is that it does not rely on any assumptions about the size of any node's routing zone. Consequently, this correctness proof applies to networks where every node's routing zone radius is the same, and to networks where each node independently adjusts its own routing zone.

Let $X(t)$ be the set of reachable nodes that belong to the routing zones of all bordercast recipients, at time t. We refer to these nodes as *covered* nodes. Likewise, the complementary set of nodes, $X^c(t)$, represent the remaining set of *uncovered* nodes. Let $P(t)$ be a subset of $X(t)$ such that each node in $P(t)$ has at least one neighbor in $X^c(t)$. These nodes, which we refer to here as *frontier nodes*, are the covered nodes that form a boundary between covered regions and uncovered regions in the network.

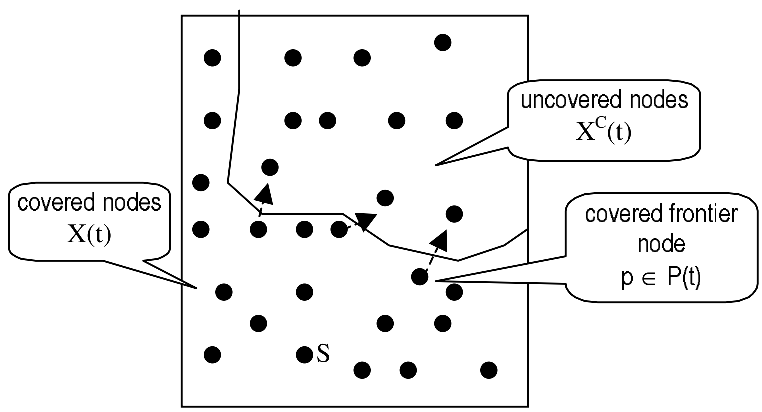


Figure 14

Theorem 1: $X(t_1) \subset X(t_2)$ and $X^c(t_2) \subset X^c(t_1)$, for $t_1 \leq t_2$

Proof:
Once a node has been covered by a query, it cannot be uncovered. Therefore, $\mathbf{X(t_1) \subset X(t_2)}$.

From basic set theory, it also follows that $\mathbf{X^c(t_2) \subset X^c(t_1)}$.

Theorem 2: If $|X^c(t)| > 0$, $|P(t)| > 0$

Proof:

Since all $x \in X^C(t)$ are reachable from any other node, there exists a node $x \in X^C(t)$ which has a neighbor $p \in P(t)$.

Theorem 3: Every $p \in P(t)$ is a peripheral node of a bordercast recipient.

Proof:
Since $p \in P(t)$, $p \in X(t)$ (p belongs to the set of covered nodes). Since p was covered by a query, it belongs to the routing zone of a bordercast recipient. Thus, p must either be a peripheral node or an interior node of the bordercast recipient's routing zone. If p was an interior node, then *all* of p's neighbors would belong to a queried routing zone and be considered covered as well. However, by definition, at least one of p's neighbors is uncovered ($x \in X^C(t)$). Therefore, p must be a peripheral node of a bordercast recipient.

Theorem 4: If there exists a node $p \in P(t_1)$ such that $p \notin P(t_2)$, then $|X^C(t_2)| < |X^C(t_1)|$.

Proof:
Let N_p be the set of nodes that are neighbors of node p.

Because $p \in P(t_1)$, $X^C(t_1) \cap N_p \neq \varnothing$
Because $p \notin P(t_2)$, $X^C(t_2) \cap N_p = \varnothing$

It follows that $X^C(t_1) \neq X^C(t_2)$. We've already shown that $X^C(t_2) \subset X^C(t_1)$, so $\mathbf{|X^C(t_2)| < |X^C(t_1)|}$.

Theorem 5: If a node p is a bordercast recipient and receives the route query message at t_2, then $p \notin P(t_2)$.

Proof:
When node p receives the bordercast route query at t_2, it will check its routing zone for the queried destination, thereby covering all members of its routing zone. Consequently, all of node p's neighbors will be covered and $p \notin P(t_2)$.

Our protocol is said to provide **full coverage** if a route query for a *non-reachable* destination results in every reachable node belonging to at least one queried routing zone (i.e. $X^C(t_k) = 0$, for $t_k<\infty$)

If our protocol provides full coverage, it follows that a route query for any *reachable* destination will discover at least one route to that destination.

Theorem 6: ZRP permits at least one node in $P(t_{k-1})$ to receive a route query bordercast by time $t_{k-1} < t_k < \infty$.

Proof:
For each covered frontier node, $p \in P(t_{k-1})$, there exists at least one bordercast recipient, n, that will launch a bordercast to p. If the query is relayed along the designated path between nodes n and p, then node p will receive the route query packet.

An intermediate relaying node, m, may terminate the query for one of two reasons. In the first case, m may determine (based on its detected query information) that p is no longer a covered *frontier* node. However, p *was* a covered frontier node at time t_{k-1}. The only way for p to have been removed from the frontier is if another covered frontier node, $p' \in P(t_{k-1})$, already received a route query bordercast at some time $t_{k-1} < t_k < \infty$.

In the second case, m recognizes that it has already relayed a query packet to node p. If nodes m and p are a distance of d hops apart, then the query packet has already progressed to d-1 hops from p. For the case of d=1, the query packet is relayed directly to p. It follows, by induction, that as long as p remains on the frontier, p will successfully receive a route query bordercast by time $t_{k-1} < t_k < \infty$.

Theorem 7: ZRP provides full coverage.

Proof:
From Theorem 6, we know that at least one node in $P(t_{k-1})$ will receive a route query by time $t_k < \infty$.

Each time, t_k, that a query arrives at a node $p \in P(t_{k-1})$, the number of uncovered nodes decreases ($|X^C(t_k)| < |X^C(t_{k-1})|$). Consequently, there exists an increasing sequence of times $\{t_k\}$ for which the corresponding number of uncovered nodes $\{|X^C(t_k)|\}$ decreases, ultimately reaching 0.

8. References

[Deng98] J. Deng. and Z.J. Haas, “Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks,” IEEE ICUPC’98, Florence, Italy, Oct. 5-9, 1998.

[Haas98] Z.J. Haas and M.R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," ACM SIGCOMM, Vancouver, BC, 1998.

[Johnson96] D.B. Johnson and D.A. Maltz, “Dynamic Source Routing in Ad-Hoc Wireless Networking,” in *Mobile Computing,* T. Imielinski and H. Korth, editors, Kluwer Academic Publishing, 1996.

[Moy97] J. Moy “OSPF Version 2”, *RFC 2178,* March 1997.

[Murthy96] S. Murthy and J.J. Garcia-Luna-Aceves, “An Efficient Routing Protocol for Wireless Networks,” *MONET,* vol.1, no.2, pp.183-197, October 1996.

[Park97] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," IEEE INFOCOM '97, Kobe, Japan, 1997.

[Pearlman99] M.R. Pearlman and Z.J. Haas, "Determining the Optimal Configuration of the Zone Routing Protocol," IEEE JSAC, special issue on Ad-Hoc Networks, vol.17, no. 8, August 1999.

[Perkins94] C.E. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” *ACM SIGCOMM,* vol.24, no.4, Oct. 1994, pp.234-244.

[Perkins99] C.E. Perkins and E.M. Royer, “Ad Hoc On-Demand Distance Vector Routing,” *IEEE WMCSA’99,* New Orleans, LA, Feb. 1999.

[Postel81] J. Postel, “Internet Protocol, “ RFC *791*, Sept. 1981.