Virtual Paths Routing
A Highly Dynamic Routing Protocol for Ad Hoc Wireless Networks

Abdulrahman H. Altalhi Golden G. Richard, III

Computer Science Department
University of New Orleans
New Orleans, LA 70148 U.S.A.
{aaltalhi,golden}@cs.uno.edu

Abstract

In this paper, we introduce the Virtual Paths Routing (VPR) Protocol for ad hoc wireless networks. VPR provides highly dynamic, correct, and efficient paths creation and maintenance between nodes. Innovatively, the protocol utilizes a technique to monitor the mobility of the nodes, and factorizes it in its operations. VPR is a distributed, on-demand, and adaptive protocol that comprises two phases. The first phase is path discovery, in which VPR discovers and creates a virtual path between source and destination nodes. The second phase is path maintenance, in which the protocol monitors the usability of all active paths.

Keywords: Ad Hoc Networks, Mobility, and Routing.

1. Introduction

Rapidly progressing technology for mobile devices, including laptops and handheld computers, and the availability of inexpensive wireless networking hardware has resulted in serious interest in wireless connectivity among mobile users. One approach to provide wireless connectivity is the formation of ad hoc wireless networks. One of the challenging problems in this type of network is the routing problem in view of the dynamic nature of the ad hoc wireless networks. We propose a highly dynamic, distributed, on-demand, and adaptive solution in which the mobility of the nodes and the impact of this mobility on the network topology are effectively taken into account. In our solution, all nodes potentially participate in forwarding messages between sources and destinations by the means of virtual paths. Each node maintains routing information to other nodes, discovering new routes as necessary, and performing maintenance on known paths. Our technique to determine the level of mobility by and around a node is an *operational based* technique.

This paper is organized as follows. In Section 2, we briefly review related works. In Section 3, we fully describe the Virtual Paths Routing (VPR) protocol. In Section 4, we detail the mobility management of VPR. In Section 5, we present a simulation study of VPR. Finally, in Section 6, we conclude the paper.

2. Related Work

The development of routing protocols for ad hoc networks is an active research field, and many protocols have been proposed. In this section, we briefly survey some of the closely related work, explain how each protocol handles the mobility of the nodes and the network topology changes, and highlight the drawbacks of each approach.

The first protocol is the **Destination Sequenced Distance-Vector (DSDV)** [1], which is a proactive hop-by-hop distance vector routing protocol that requires every node to repeatedly broadcast routing updates. DSDV requires a time to converge since it is based on periodical broadcasts of route information to cope with the dynamic mobility of the nodes. At a high mobility level, the time to converge is considerable, and causes many packets to be dropped [2]. Furthermore, the periodical broadcast of route updates from all the nodes consumes much of the bandwidth and adds a high overhead [3].

The second protocol is the **Adaptive Distance Vector Routing Algorithm for Mobile Ad Hoc Networks (ADV)** [4], which is based on the distance vector routing algorithm combined with a sequence number mechanism. This protocol adaptively changes the size and rate of route update exchanges. To deal with mobility and topology changes, the nodes of the network broadcast their views of the network to their immediate neighbors periodically. This strategy is well known for forming short-lived loops and adding considerable overhead.

The third protocol is the **Ad Hoc On-Demand Distance Vector (AODV)** [5], which is a reactive hop-by-hop routing protocol that uses sequence numbers and periodic beacons like DSDV. Although the use of the sequence number guarantees the protocol to be loop-free, it may get uncoordinated when the network is partitioned [2]. The use of periodic beacons to keep track of node movements and network topology changes adds a sizeable overhead. The protocol also dictates that all

nodes with active routes broadcast a HELLO message every second, which, again, adds another considerable overhead [3].

The fourth protocol is the **Dynamic Source Routing (DSR)** protocol [6,7,8], which is a reactive routing protocol that uses a source routing technique to deliver packets. Each packet to be routed carries in its header the full and ordered list of nodes through which the packet will be routed. The main disadvantage of DSR is the fact that the source route is contained in the header of every packet. The inclusion of the full source route, which increases in size as the number of hops to the destination increases, is a substantial overhead [2, 3, 9]. More importantly, the mobility levels of the nodes are not used in any of the operational aspects of DSR.

The fifth protocol is the **Implicit Source Routes for On-Demand Ad Hoc Network Routing (ISR)** [10], which is based on DSR. This protocol uses a technique called Implicit Source Routing to reduce the per-packet overhead that is associated with source routing. In this technique, each node maintains a flow table, with one entry for each logical flow serviced by that node. Beside source and destination addresses and the flow identifier, each entry records the next hop address to which a packet for this flow should be routed. Those logical flows are uni-directional from source to destination. Another logical flow is needed from the destination to the source if the communication is bi-directional. More notably, the timeout values associated with these flows are constant; regardless of the level of mobility a node is currently experiencing.

Last protocol is called **Hierarchically–Organized, Multihop Mobile Wireless Networks for Quality of Service Support** [11], which is a Link State based routing protocol with a hierarchically organized structure. This protocol creates *Virtual Circuits* to deliver messages between nodes. The only purpose of the circuits is to provide a Quality of Service support, while the underlying routing technique is based on Link State, in which each node periodically broadcasts the link costs of its outgoing links to all other nodes to handle mobility and network topology changes.

3. Virtual Paths Routing Protocol

In this section, we describe the Virtual Paths Routing protocol for ad hoc wireless networks in detail.

3.1. Overview

To deliver a packet between a source and destination node, a virtual path must be established to connect the two nodes. A virtual path is simply a route, or sequence of nodes, in which all the nodes are aware of the existence of the path and they monitor its activities. Each node on any given virtual path knows its predecessor and successor nodes on the path. All the packets to be delivered through a particular virtual path are marked with a key that uniquely identifies the virtual path to be used, and each node passes the packets to the next node on the path until they reach their final destinations.

Any given node operates as a router and it must be able to handle more than one virtual path. To accomplish that, each participating node maintains two virtual routing tables. Those tables are called primary and secondary tables. The primary table contains the currently active paths, which we define as the paths that are currently in use (i.e., to route packets of an established connection between source and destination nodes) and are operational. The structure of each entry in primary table is as follows:

1. *<Path Number>*
2. *<Full Path>*
3. *<Source Address>*
4. *<Destination Address>*
5. *<In-Bound Node Address>*
6. *<Out-Bound Node Address>*
7. *<Flag>*

The *Path Number* is a sequence number that is generated by the transmitting node at the creation time of the path (entry). The *Full Path* is a list of all nodes' addresses on the path from the transmitting node to the final receiving node (i.e., it is a source route). The *Source* and *Destination Addresses* are the IP addresses of the transmitting and receiving nodes respectively. The *In-Bound* node address is the address of the previous node on the path, with respect to the current node, from source to destination. The *Out-Bound* node is the address of the next node on the path, with respect to the current node, from source to destination. The *Flag* field is used to mark a path that is not operational but cannot be removed from the table yet; this occurs during attempts to repair a broken path. Each entry in this table is uniquely identified by the combination of *Source Address*, *Destination Address*, and *Path Number*.

The secondary table contains paths that are collected or learned by the node and are not used to route packets. We consider the secondary table as a repository of cached routes. Each entry in the secondary table contains the following fields:

1. *<Full Path>*
2. *<Timeout>*

The full path is as explained above. The timeout is the time when this entry would no longer be valid and must be deleted.

To communicate with a destination node, a source node must **establish** a virtual path to that destination. The process of creating such a path is fully described in a subsequent section. As a result of such a process, an entry is created in each primary table on each node in the path.

This virtual path is used to route traffic between the source and destination nodes bi-directionally. There are two possible ways to deactivate that virtual path. The first is the source node sending a deactivation message on that path. The deactivation message is a control message that is used by our protocol to close an established virtual path. The second is the detection of an un-repairable broken link along the path.

When a path is to be **deactivated**, it is removed from the primary table and placed in the secondary table. The secondary table contains paths that are not active but still valid for a certain period of time. When a path is deactivated due to a deactivation message, the whole path is copied into the analogous field in the secondary table. However, when a path is deactivated due to a broken link, the path is truncated from the broken link, and then copied into the corresponding field. The protocol packet header contains an 8 bit-flag (called the operation flag) that is used to control the operations of the protocol. One bit in the flag is used to indicate the type of deactivation of this path. If the bit is set, the deactivation is due to an end-of-dialog. Otherwise, the deactivation is due to a broken link.

The full path, as defined above, is a partial knowledge about the current topology of the network. Because this knowledge is valuable, it is copied into the secondary table rather than deleted. However, due to the dynamic nature of ad hoc wireless networks, a timeout is associated with each entry in the table. The timeout is used to protect the protocol from using outdated information about the network topology. The timeout values that are used in VPR are not constant; they vary based on the mobility level of the nodes.

3.2. Basic Protocol Operations

In order to send a packet to another node, the transmitting node checks its virtual paths table (primary table) to see if it has created such a path to the intended destination node. If no path is found, the node then inspects its secondary table for any possible path that was obtained earlier. If, again, no path is found, the node must create one before sending the packet. While the node is creating the path, it buffers the packet until it finishes the creation of the path to the destination.

If a valid path is found on the primary table, the transmitting node constructs a VPR routing header for the packet to be sent. This routing header includes the *Source* and *Destination Addresses* and the *Path Number* of that path. After constructing the header, the transmitting node forwards the packet to the *Out-Bound* node of the path.

Upon receiving a packet, a node on the path checks the routing header to see if it is the final destination of the packet. If the node is the final destination of the packet, it delivers it to the higher layer protocol. Otherwise, it forwards the packet to the next node on the path. The next node is determined by using the VPR routing header fields, *Source* and *Destination Addresses* and *Path Number*, to uniquely identify a path in the local virtual paths table (primary table). The *Out-Bound* node of that path is used as the next node. If the node does not have an entry for the path in its primary table, the node unicasts a "path unknown" message to the forwarder of the packet.

A destination node may use the same path to send packets to the source node. One bit in the operation flag is used to indicate the direction of the packet on this path. If the bit is set, the direction is considered downstream, from source to destination, and the packets are forwarded to the *Out-Bound* node of this path. Otherwise, the direction is considered upstream, from destination to source, and the packets are forwarded to *In-Bound* node of this path.

At the end of a dialog between two nodes, the creator of the path sends a deactivation message along the path to deactivate the path. Each node on the path that receives the deactivation forwards it to the *Out-Bound* node of that path, and then deletes the path entry from its primary table. Before deleting the path, the node moves some of the path information into the secondary table as described earlier.

3.3. Virtual Path Creation

Path creation is a two-phase process. The first is the **Path Discovery,** during which the source node determines the path through the ad hoc network to the destination. At the end of this phase, the source node should have a list of all the nodes through which it may reach the destination node. The second phase is the **Path Set Up** for all the nodes included on the list. At the end of this phase, each node on the path has an entry in its virtual paths routing table (primary table) for the newly created path.

3.3.1. Path Discovery. The source node broadcasts a path discovery packet to all of its immediate neighbors within its transmission range. The VPR header of this packet includes the *Source* and *Destination Addresses* and a *Path Number*. The *Path Number* is locally managed within the source node and is incremented whenever a new path discovery is initiated. Each node in the vicinity maintains a record of the recently received path discovery packets. The record contains sufficient data (*Source* and *Destination Addresses* and *Path Number*) to uniquely identify each packet received, and the node uses it to detect duplicate packets. The path discovery packet also contains a node list, which is used to record the route between source and destination as the packet is propagated through the ad hoc network. When the source node initiates the path discovery packet, the node list contains only the source node IP address. A path discovery packet that is received by any node is processed as follows (as in DSR [6,7,8]):

I. **Drop the packet and proceed no further:** The node will drop the packet if it is a duplicate packet or the node IP address is already on the packet's node list. The node uses its record of the recently received path discovery packets to determine if the packet is a duplicate. This step guarantees a loop-free protocol.

II. **Send a path discovery reply:**
If the node is the target of this path discovery packet or has a valid path to the intended destination node, then it generates a path discovery reply to the initiator.

III. **Rebroadcast the packet:**
If the packet is not a duplicate, nor the node is on packet's node list, nor it is the destination of the packet, then the node adds itself to packet's node list and rebroadcasts it.

Hence, the path discovery packet is disseminated through the vicinity until it reaches either its final destination node or a node that has a valid path to the destination. Valid paths are either operational paths on the primary table or the paths on the secondary table. In either case, the node generates a path discovery reply destined to the initiator of the search.

The VPR header of such a reply is constructed as follows. If the node, which generates the reply, is the intended destination of the search, then it copies the path discovery packet's node list to the reply packet and appends its own address to it. It also copies the path number from the discovery packet to the reply packet.

If the node which generates the reply is an intermediate node that has a valid path to the intended destination, then it copies the path discovery packet's node list to the reply packet and appends its own address. It also appends the valid path that was found locally in its tables (primary or secondary) to the list. To ensure loop freedom, the node must check for duplication of nodes on the newly constructed list. The rest of the fields are copied as described above. To deliver the path discovery reply to the initiator node, a source routing technique, which traverses the packet's node list as the path to the source, is used.

3.3.2. Path Set Up. When the node that initiated the path discovery process receives the path discovery reply, it starts the path set up phase. First, the node creates an entry in its primary table for the new path. The node list on the reply packet is copied into the *Full Path* field of this new entry. The source and destination addresses and the path number are copied into the corresponding fields of the new entry as well. The value of the *Out-Bound* field is determined by navigating through the node list (it is the second node in the list). The node's own IP address is assigned to the *In-Bound* field of the new entry. Second, the source node uses its first data packet to be sent to the destination to set up the path. It copies the path reply packet's node list to the VPR header of its data packet. The source node also includes its address as the *Source Node Address*, the *Destination Node Address*, and a *Path Number* in the header of this data packet. The newly created entry in the primary table is used to route this packet.

Each node on the path that receives this initial packet processes it as follows. If the node already has an entry for the path of the packet, it uses this entry to forward the packet to the next node on the path. However, the node must compare the node list attached to the data packet to the *full path* of the entry. If they match, the entry is used to route the packet. If not, the fields of the entry are updated and then used to route the packet. The latter case reflects a path recovery process, discussed in the path maintenance section. If the node has no entry, it creates an entry for the new path and uses this entry to route the packet to the next node.

The routing header of the first data packet contains all the necessary data for a node to create or update entries in the local primary table. The node copies the *Source* and *Destination Addresses* and the *Path Number* from the header of the packet into their corresponding fields in the table. The *In-Bound* and *Out-Bound* fields are filled with the predecessor and successor nodes of the current node in the packet's node list. Subsequent data packets, from the source to the destination, contain only the *Source* and *Destination Addresses* and the *Path Number*. Those packets will be routed as described previously. The source and destination addresses in the IP header of the packet could be used to reduce the size of the VPR header.

3.4. Path Maintenance

Links in an ad hoc wireless network may fail due to several reasons. When a node along an established path crashes, that path becomes unserviceable. Nodes of an ad hoc wireless network may move freely at any given time.

When a node on a certain path moves to a new location, that path may become unviable. To cope with such situations, the protocol uses path maintenance techniques to detect and repair link failures.

3.4.1. Link Failure Detection. Link failure detection is an important aspect of any routing protocol. It greatly impacts the performance of the protocol. To detect link failures, VPR uses Hello Messages and Link Layer acknowledgments. The use of the Hello Message provides the protocol with an early notice of the link breakage. This notice enables the protocol to notify the source node about the link breakage before it sends (and loses) many packets on that link. A Hello Message is a short message that is sent by a node within its vicinity to report its presence to its neighbors. Each message contains the identity of the node (IP address). Due to the overhead involved in sending Hello Messages, only those nodes on active paths may send the Hello Messages. A node is considered to be on an active path if it has an entry in its primary table. Hello Messages have an implicit Time To Live (TTL) of 1. A node must send a Hello Message if it has not sent or forwarded any packets to any of its Out-Bound and In-Bound nodes within a period of time known as the Hello Message Interval. This interval of time is a function of node mobility. Nodes monitor the presence of their neighbors through data and control packets and the Hello Messages they send. Whenever a node receives either one, it resets the time for the next Hello Message from that neighbor. When a node fails to receive three consecutive Hello Messages from one of its Out- or In-Bound nodes, it considers that node as unreachable, and initiates the repair algorithm to fix the link as we describe later.

IEEE 802.11 standard [12] requires that a receiving node to send an acknowledgment for each unicasted packet it receives. The continual absence of such ACKs is a clear indication of a link breakage. The use of this **Link Layer Support** will greatly improve performance if it is available [2]. Whenever the **Link Layer** is unable to deliver a packet to a neighbor, it notifies the protocol. However, in an ad hoc wireless network, two neighbors may lose connectivity for a short period of time and VPR should not be overly responsive to apparent link breaks. While responsiveness leads to rapidly repairing paths over broken links, it also leads to the unnecessary repairing of temporarily disrupted paths. To avoid that, VPR tries to deliver a packet to a neighbor three times, with an increasing back-off period for each attempt. After three consecutive unsuccessful attempts, VPR considers the link to that neighbor to be broken and invokes the repair algorithm to fix it.

The path repair algorithm is also invoked if a forwarder of a packet receives a “path unknown” message as described in Section 3.2. This situation can occur when a node along an established virtual path recovers from a momentarily loss of power in which it lost its state about virtual paths that had been established through it.

3.4.2. Path Repair. A virtual path consists of multiple links between numerous mobile nodes. When one of these links is broken, the entire path is considered to be broken. These broken links are usually detected by the two nodes that formed the link. The broken link divides any path into two portions. The first portion, from the source node to the first node that detected the broken link, is called the upstream portion of the path. The second portion, from the destination node to the second node that detected the broken links, is called the downstream portion of the path.

The **first** action in the repair process is carried out by the two nodes that detected the failure of the path. The node on the upstream portion sends a “temporarily out of service” message along the path. Every node that receives such a message sets the flag of that path to indicate that it is out of service. However, the path entry is not deleted from the primary table. Instead, a timeout value is associated with the entry in anticipation of either a path creation or a path error message. The node on the downstream portion sends a deactivation message along its portion of the path on behalf of the source node. Every node that receives this message, deactivate the path as described previously in the deactivation process.

The **second** action in the repairing process is to find a new path. This action is performed by the upstream portion of the broken path, specifically, by the node that detects the link failure on this portion. This node takes the following steps:

I. Find a new path:
The node searches for a route to the destination in its local cache; if a valid path is available, it extracts the node list from the path, appends the extracted list to the list of nodes on the upstream portion of the broken path from its own primary table, and sends this newly constructed list in an unsolicited path reply to the source node.

II. Send a path error:
If no path can be found to the destination, the node sends a path error message to the source node. The intermediate nodes that forward the path error message along the path to the source node deactivate the path as described before.

Since a path may experience multiple simultaneous broken links, we limit the node to only resort to its cached paths rather than initiating the path discovery process. By this constraint, the protocol avoids a situation in which multiple nodes initiate the path discovery at the same time to repair the same broken path. The protocol also dictates that upon finding a path in the local cache, the node must send the new path to the source node of the broken path in unsolicited path reply. The reason for this restriction is to avoid having multiple nodes concurrently repairing the

broken path without notifying the source node, which would set the path to inconsistent state.

The **third** and last action is executed by the source node of the broken path. At the end of this step, a determination is made about the final status of the path. The source node takes the following steps:

I. Initiate the path set up:

If the source node receives any unsolicited path replay, it initiates the path set up process as described previously to set up the new path.

II. Start the path discovery:

If the source node receives a path error message, it starts the path discovery process to find a new path.

The paths, which were marked as out of service in the first step, have timeout values associated with them. If these timeouts expire with no resolution, the paths are removed from the primary table and placed in the secondary (i.e., deactivated). While the path repair process is engaged, intermediate nodes on the broken path will attempt to salvage the packets in transit using its cached routes. When a valid path is available to salvage a packet, it will be used in a simple source routing technique. In addition, the source node ceases sending any more packets along the broken path until a resolution is made.

3.5. Path Selection

The path discovery process may yield more than one path to the same destination. The initiator node must wait for an interval of time known as the PATH_REPLY_WAIT after starting the path discovery process. This interval allows more than one reply to arrive at the initiator before it starts the path creation process.

Currently, the path to be utilized is selected based on the distance (number of hops) between the initiator and the destination nodes. The path with the least number of hops is chosen. However, the protocol is capable of finding paths based on other criteria. Since VPR is designed to be able to collect data about the network status during path discovery and the fact that every participating node is aware of the paths it maintains, VPR can provide paths that are **Power-Aware** or **Load-Aware**. More creatively, because of the mobility monitoring technique employed by VPR, it can provide **Mobility-Aware** paths in which the level of mobility is the criterion used for path selection. These criteria and others for path selection, is the subject of an ongoing research project.

The replies that are not selected will be inserted to the secondary table since they are valuable knowledge about the network topology. More criteria will be used in planned improvements to the protocol in the near future.

4. Mobility

The mobility of the nodes of an ad hoc wireless network greatly impacts the performance of the routing protocols designed for such a network. For instance, the mobility of the nodes alters the validity of cached routes that were collected by the nodes [13,14,15]. All the proposed protocols either use no expiry time for their cached routes or use a constant value. Both of these choices certainly do not tune with the dynamic environment of the ad hoc wireless networks. In fact, both choices would degrade the performance of the protocols. If no expiry time is used, stale routes may be used which results in a route errors and a corruption of other cached routes by other nodes. This situation was shown in [14, 15, 16]. If a constant value is used (as in ISR and AODV), it is possible to invalidate a viable route [15]. While the expiry times are static for most protocols, the dynamic nature of ad hoc networks suggests a dynamic approach. Similar works, about expiry time and mobility were studied in [17,18]. However, in [18] they studied a heuristic technique to adjust the expiry time and in [17] the emphasis was on the cache structure and management.

We will limit our discussion on the length of the timeout period of cached routes due to its significant impact on the performance. In VPR, the timeout value is varied in response to changes in the mobility level by, and around, a given node. When the level of mobility is high, the timeout period must be short to prevent the node from using invalid or out dated data such as its knowledge of the network topology. On the other hand, when the mobility level is low, the timeout period must be long to prevent the node from losing valid data or performing unnecessary overhead. The length of the timeout period should not be changed sharply. Rather, a gradual decreasing or increasing of the length is preferred. Also, the length should be limited by upper and lower bounds.

VPR defines a **mobility indicator** that reflects the current level of mobility. The indicator is a variable with possible values ranging between 0 and 1. A value of 1 indicates a high level of mobility in which the network nodes are in constant movement. Whereas, a value of 0 indicates a low level of mobility in which the nodes of the network are stationary. To capture the mobility level by and around a node, the protocol monitors the operations it performed on the primary table entries. Particularly, it monitors the deletion of entries from the table due to broken paths. These deletions are clear indications of a high level of mobility within a node's environment. The insertion operations are not monitored since they are indications of a high level of activity more than level of mobility.

When a node deletes entries from its primary table because of broken paths, the value of the mobility indicator should be increased to reflect a high level of mobility. In contrast, when the node does not delete any entries, the value of the mobility indicator should be decreased to reflect a low level of mobility. The values to be used to either increase or decrease the value of the mobility indicator are calculated as follows:

Let

MLI = *Mobility Level Indicator*
NED = *Number of Entries Deleted since last adjustment.*
CNOE = *Current Number of Entries.*
MAF = *Mobility Level Adjustment Factor (Constant =0.2)*
CTO = *Current Timeout.*
UB = *Timeout Upper Bound*
LB = *Timeout Lower Bound*

Then, the Mobility Level Indicator (*MLI*) is calculated as follows:

$$MLI = (NED / CNOE) \quad 1$$

Any node that deleted an entry, should decrease its current timeout value according to the following equation:

$$New\ TO = CTO - (MLI * (UB-LB) * MAF) \quad 2$$

And any node that did not delete an entry, should increase its timeout according to the following equation:

$$New\ TO = CTO + (MAF * (UB-LB)) \quad 3$$

In the first equation, the use of the of the quantity (NED / CNOE) is to allow the number of entries deleted and current number of entries to play a role in calculating the value to be used to set the mobility indicator. The indicator is used in equation 2 to adjust the timeout value. The higher the value of the Mobility Level Indicator, the higher the value to be used to decrease the timeout will be. The timeout period is increased when the node does not delete any entry from its table. Since some of the entries might be deleted due to **node failures** and not to a high level of mobility, the rate that is used to decrease the timeout is lower than the rate to increase it. If the number of current entry is 0 and the number of deleted entry is greater than 0, the new MLI is set to equal 1. This adjustment takes place **periodically**. When a node calculates a new timeout, it initializes *NED* to zero and waits for the same newly calculated timeout before it recalculates and adjusts the timeout again.

5. Simulation Study

We compare the VPR to ISR using the *ns-2* network simulator [19], which includes a mobility extension that was ported from the CMU Monarch Group's mobility extension to *ns*. The CMU Monarch mobility extension to *ns* allows the simulation of multi-hop ad hoc wireless networks. The extension includes functionalities to simulate node movements, and to transmit and receive on wireless channels with a realistic radio propagation model.

For the sake of fairness, we choose to compare VPR to ISR because ISR is the closest protocol to VPR with respect to basic functionality. Furthermore, ISR is based on a well-studied protocol that is DSR, which gives us the ability to indirectly compare VPR to DSR.

We modeled our network interfaces after the Lucent WaveLan DSSS IEEE 802.11 product with a transmission rate of 2 Mbps. The interfaces use the IEEE 802.11 Distributed Coordination Function (DCF) [12] MAC protocol, which utilizes carrier sensing for collision avoidance. For the ISR simulation, we used the latest version available from the VINT project that comes with *ns-2*. That version includes a DSR with a full implementation of the Implicit Source Routing (**ISR**) technique. We simulated ISR in the non-promiscuous mode. The promiscuous mode does not give DSR/ISR a significant improvement [9]. We also decided not to add this feature to VPR due to security issues addressed in [2]. We implemented and added VPR to *ns-2* as described in this paper. The following table summarizes the parameter values used by both protocols:

Table 1. Values used in the simulation.

Parameter	VPR	ISR
Send buffer size	64	64
Promiscuous mode	N/A	Off
Primary cache size[1]	64	30
Secondary cache size	128	64
Reply to Requests From The Cache	On	On
Salvaging packets using the cache	On	On
Interface queue size	50	50
Hello Interval	Dynamic	N/A
Allowed lost hello	3	N/A

5.1. Traffic and Mobility Models

The traffic used in our simulations was Constant Bit Rate (CBR) traffic. The source and destination nodes were randomly selected, and each simulation shows the results of 30 connections. The size of each CBR packet was 148 bytes, which included 128 bytes of data and 20 bytes for the IP header. We modeled a 5 packets/sec send rate. The mobility patterns in our simulation followed the *random waypoint* [3] model. In that model, each node starts at a random location, chooses a new location in a rectangular space (1500 m x 300 m) randomly, and starts its trip to the new location at a randomly chosen speed (uniformly distributed between 0–20 m/sec). After reaching the new location, a node pauses for a period of time called the *pause time*, and then starts a new trip to a new location. We varied the mobility of the nodes by varying the *pause time* values.

The results we present in this paper are based on simulation runs of 50 nodes. Each run is 500 seconds. Ten runs of different traffic and mobility scenarios are averaged to generate each data point. However, identical traffic and mobility scenarios were used for both protocols.

[1] The difference in the cache size is due to the difference in the cache implementations; it does not impact the protocol performance.

5.2. Results

We used three performance metrics to compare VPR to ISR. The first metric is the **Packet Delivery Ratio**, which is defined as the percentage of data packets delivered to their destination nodes of those sent by the source nodes. The second metric is the **Routing Overhead** of both protocols, which is defined as the number of routing packets "transmitted" per each data packet "delivered." On multi-hop routes, each transmission of the routing packets is counted as one transmission. We chose not to include the forwarding information carried in each data packet in our calculation of the overhead because the size is the same for both protocols. The third metric used is the **Average End-To-End Delay** of the data packets. We used eleven pause time values (0, 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500 s) to differ the mobility level (with 0 s pause time means continually moving nodes and 500 s representing stationary nodes).

5.2.1. Packet Delivery Ratio. VPR, in general, has a better delivery ratio than ISR (see Figure 1). VPR delivered an average of 7.12 % of the data packets sent higher than ISR. The difference in the delivery ratio between both protocols is considerable at the high level of mobility where the pause times are 0 s and 50 s. With pause times 0 s and 50 s, VPR delivered 16.58 % and 9.42 % higher than ISR, respectively. The mobility monitoring technique is the explanation behind these results. VPR expires the cached routes based on its monitoring of mobility level around the node. That technique prevents the usage and propagation of stale routes, which explains the better performance of VPR.

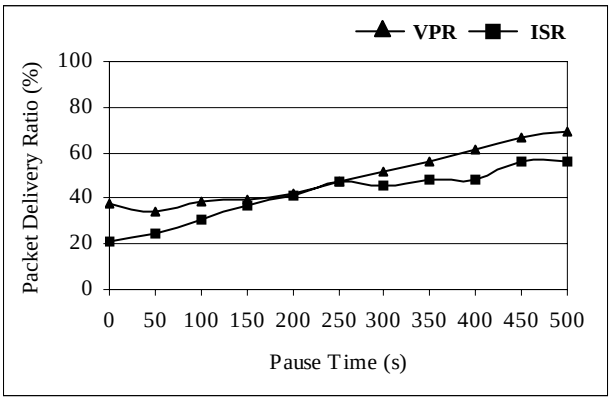

Figure 1. Packet delivery ratio.

5.2.2. Overhead. VPR also outperformed ISR in the routing overhead metric (see Figure 2). ISR incurred an average of 6.27 overhead packets per data packet higher than VPR. We found that the higher the level of mobility, the higher the difference in overhead between VPR and ISR. At the highest level of mobility, ISR incurred about 20.19 of overhead packets per data packet, whereas VPR incurred about 1.29 overhead packets per data packet. The difference in the overhead incurred by both protocols is insignificant at low mobility levels. The reason for such a high overhead is the additional Route Discoveries incurred by ISR through its salvaging process.

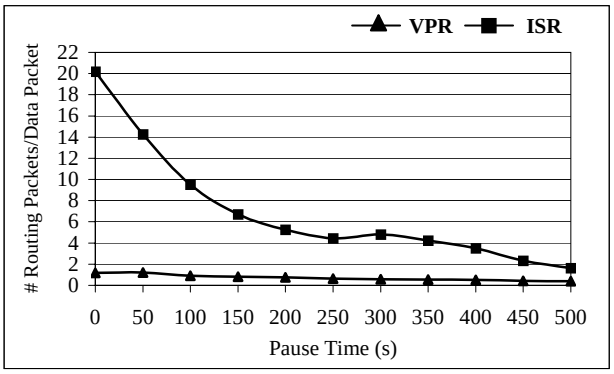

Figure 2. Overhead.

The mobilities of the nodes invalidate the cached routes that were collected by the protocols. While VPR uses its mobility monitoring technique to limit the validity of a cached route, ISR invalidates a route only when it receives a route error. In ISR, the node that recognizes the broken link on a route will attempt to salvage the data packets it had received on that route and usually uses the Path Discovery Process. If a node uses multiple invalid routes for the same destination, that results in multiple nodes initiating the Route Discovery Process for the same destination at the same time in order to salvage the data packets they had received on the broken routes.

5.2.3. Average End-to-End Delay. VPR surpassed ISR in the average end-to-end delay metric (see Figure 3). The average end-to-end delay of VPR was about three second less than that of ISR. At the highest level of mobility, the average end-to-end delays for VPR and ISR were 2.31 s and 6.70 s, respectively. Generally, the average of the end-to-end delay of VPR is less than that of ISR. We believe the reason for that is the high overhead traffic generated by ISR, which was discussed in Section 5.2.2. High overhead traffic congests the network and causes a long delay for the data packets.

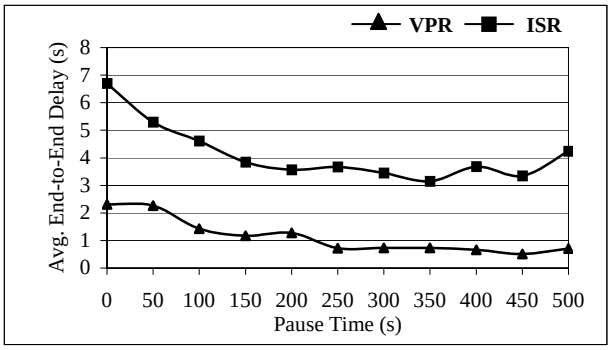

Figure 3. Average end-to-end delay.

6. Conclusion

In this paper, we introduced the Virtual Paths Routing (VPR) Protocol for Ad Hoc Wireless Networks. The key purpose of VPR is to provide correct, efficient, and highly dynamic route creation and maintenance between mobile nodes. The protocol utilizes a technique to monitor the mobility of the nodes and factorizes it in its operations. That technique is a distinguished feature that makes VPR a unique protocol among all of the proposed protocols. For comparison purposes, we presented a simulation study in which we compared VPR to ISR. The study showed the advantage of the mobility management utilized by VPR. Although, has technique has improved VPR, it can be further improved by creating a global Mobility Indicator for the entire network.

Acknowledgment

The authors would like to thank Ann Jones for her assistance in compiling this paper.

References

[1] C. Perkins and P. Bhagwat, Highly Dynamic Destination Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, In *Proceedings of the Conference on Communications Architectures, Protocol and Applications (SIGCOMM '94)*, London, United Kingdom, August 1994, 234-244.

[2] T. Larsson and N. Hedman, *Routing Protocols in Wireless Ad-Hoc Networks – A Simulation Study*, *Master Thesis*, (Stockholm: Luleå University of Technology, 1998).

[3] J. Broch, D. A. Maltz, D. B. Johnson, Yih-Chun Hu, and J. G. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, Dallas, Texas, USA, October 1998, 85–97.

[4] R. V. Boppana and Satyadeva, Adaptive Distance Vector Routing Algorithm for Mobile Ad Hoc Networks, In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (InFOCOM2001)*, Anchorage, Alaska, USA, April 2001, 1753-1762.

[5] C. E. Perkins, E. M. Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, Internet Draft, draft-ietf-manet-aodv-07.txt, November 2000.

[6] D. B. Johnson, D. A. Maltz, Yih-Chun Hu, and J. G. Jetcheva, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, Internet Draft, draft-ietf-manet-dsr-04.txt, November 2000.

[7] D. B. Johnson, Routing in Ad Hoc Networks of Mobile Hosts, In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA '94)*, Santa Cruz, California, USA, December 1994, 158-163.

[8] D. B. Johnson and D. A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, in T. Imielinski and H. Korth (Ed.) *Mobile Computing*, (Kluwer Academic Publishers, 1996) Ch 5, 153-181.

[9] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks, In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, Washington, USA, August 1999, 195–206.

[10] Yih-Chun Hu and D. B. Johnson, Implicit Source Routes for On-Demand Ad Hoc Network Routing, In *Proceedings of The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001)*, Long Beach, California, USA, October 2001, 1-10.

[11] R. Ramanathan, and M. Steenstrup, Hierarchically–Organized, multihop mobile wireless networks for quality of service support, *ACM/Baltzer Mobile Networks and Application*, Vol. 3, No 1, June 1998, 101-119.

[12] IEEE Computer Society LAN/MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, *IEEE Std. 802.11-1997* (New York: IEEE, 1997).

[13] S.R. Das, R. Castaneda, J. Yan, and R. Sengupta, Comparative Performance Evaluation of Routing Protocols for Mobile, Ad Hoc Networks, In *Proceedings of the 7th International Conference on Computer Communications and Networks (IC3N)*, Lafayette, Louisiana, USA, October 1998, 153-161.

[14] G. Holland and N. Vaidya, Analysis of TCP Performance over Mobile Ad Hoc Networks Wireless Networks. *In Wireless Networks, The Journal of Mobile Communication, Computation and Information*, Volume 8, Number 2-3, March-May 2002, Kluwer Academic Publishers, 275-288.

[15] S. R. Das, C. E. Perkins, and E. M. Royer, Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks, In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (InFOCOM 2000)*, Tell Aviv, Israel, 2000, 3–12.

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

[17] Hu, Yih-Chun., and Johnson, D, Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks, In *Proceeding of the Sixth Annual IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom'00)*, Boston, Massachusetts, USA, August 2001, 231-242.

[18] M. Marina, and S. Das, Performance of Routing Caching Strategies in Dynamic Source Routing, in *Proceedings of the International Workshop on Wireless Networks and Mobile Computing (WNMC)*, Phoenix (Mesa), Arizona, USA, Apr 2001, 425-432.

[19] K. Fall and K. Varadhan, *The ns Manual*, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, April 14, 2002. Available from http://www.isi.edu/nsnam/ns/.