A SIMULATION STUDY ON USING THE VIRTUAL NODE LAYER TO IMPLEMENT EFFICIENT AND RELIABLE MANET PROTOCOLS

by

JIANG WU

A thesis submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy,

The City University of New York

2011

This manuscript has been read and accepted by the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

	(Nancy Griffeth)
Date	Chair of Examining Committee
	(Theodore Brown)
Date	Executive Officer

Amotz Bar-Noy

Ping Ji

Bilal Khan

Nancy Lynch

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

A SIMULATION STUDY ON USING THE VIRTUAL NODE LAYER TO IMPLEMENT EFFCIENT AND RELIABLE MANET PROTOCOLS

by JIANG WU

Advisor: Nancy Griffeth

The Virtual Node Layer (VNLayer) is a cluster based programming abstraction for a Mobile Ad-Hoc Network. VNLayer defines fixed or predictably mobile geographical regions. In each region, a number of mobile nodes collectively emulate a virtual node, which provides services and relays packets for client processes.

As a clustering scheme with state replication, the VNLayer approach can theoretically improve the efficiency and reliability of MANET protocols. As a general programming abstraction, the VNLayer hides underlying complexities from protocol developers and can be shared by multiple applications. However, the VNLayer also introduces extra control overhead and prolongs data forwarding delay, which could be prohibitively expensive in terms of performance.

Based on an existing VNLayer implementation [1], we developed an ns-2 based software package, VNSim. VNSim can be used to simulate VNLayer based applications in a MANET of up to a few hundred mobile nodes, in order to better understand the impact of the VNLayer approach on efficiency and reliability.

With VNSim, we did our first case study on a VNLayer based MANET address allocation protocol, VNDHCP. Simulation results proved that the VNLayer approach can be used to adapt a wireline protocol to MANET.

We also did an extensive simulation study on VNLayer based MANET routing. A wireline routing protocol, RIP, was adapted to run over the VNLayer. With the support provided by the VNLayer, the adapted protocol, VNRIP, was implemented very quickly and can provide reasonable performance.

During the study of VNLayer based MANET routing, we identified a number of major performance limitations in the existing VNLayer implementation and the models it is based on. To tackle the limitations, we created a more realistic link layer model, extended the VNLayer model and optimized our VNLayer implementation.

With the optimized VNLayer implementation, we implemented VNAODV, an adapted version of AODV, over the new link and VNLayer models. Simulation results indicate that VNAODV delivers more packets and creates more stable routes than standard AODV in a dense MANET with high node motion rate and moderate data traffic.

This research validated the intuition that the VNLayer approach can be used to adapt wireline protocols quickly to MANET and to improve the performance of MANET protocols. This research also provided us some insights on how to implement and optimize cluster based MANET protocols.

Dedication

This is dedicated to my father, mother and wife. Their strong support made everything possible.

Acknowledgments

I would like to first thank Professor Nancy Griffeth for her guidance through this long journey. This is not only for her painstaking effort in advising me on my research, correcting papers together with me, but also for her strong support on my personal life. I've learned a lot more than scientific research from her. I would also like to thank Prof. Nancy Lynch and her research group for providing me the research topic, source code and inspiring discussions. I would especially thank Calvin Newport for his help on solving problems I had with simulations and his resourceful suggestions. I would also thank Mike Spindel for his great help in providing me his simulator for the Virtual Node Layer, based on which my simulator was created. His patient email responses at the early stage of research helped me to get started very quickly. In addition, I would thank Costas Djouvas, Yuri Cantor, Prof. Ping Ji, Prof. Bilal Khan and Prof. Amotz Barnoy on providing great comments and suggestions on my research.

Table of Contents

Contents

List of Tables

List of Figures

Figure Page

CHAPTER 1. Overview

Mobile Ad-hoc Networks (MANETs) are wireless networks set up temporarily among wireless devices, without the support of any infrastructure. Wireless devices in a MANET may move around continuously and each one of them may need to forward packets for other devices in the MANET. Because MANETs can be deployed quickly, they can be used for disaster rescue, battlefield communication and sensor networks.

1.1 Difficulties in Mobile Ad-hoc Networking

While MANET can be deployed easily, networking in a MANET faces many difficulties.

- **Absences of designated servers:** As the word "Ad-hoc" indicates, the first difficulty any MANET has to deal with is the absence of designated servers such as routers, DNS server and DHCP servers, etc. This is because any wireless device can leave the MANET or run out of battery power at any time. Therefore, to provide any service, a MANET protocol can't count on any specific node being able to work permanently. Any MANET service has to be supported by wireless devices in a distributed way so that the failure of a single device will not significantly affect the service.

- **Shared transmission medium:** Wireless data transmissions in a MANET are usually done in a shared radio channel. Given a radio channel, a packet sent out by a wireless device can reach every wireless device within the sender's radio

range. When more than one packet is heard by a wireless device from the same radio channel at the same time, neither packet can be received successfully. This kind of interference between packets can cause message collisions. Even with collision avoidance mechanisms such as CSMA/CA in 802.11, packet collisions can still happen due to issues like the “hidden terminal problem”. The result is that packets get dropped more frequently in MANETs. MANET protocols must be carefully designed to use the wireless channels efficiently while avoiding collisions.

- **Limited transmission range:** Unlike the long transmission ranges offered by cables in wireline networks, the radio range of wireless devices are typically on the order of a few hundred meters. Short radio ranges translate to larger network diameters and longer forwarding paths when a packet needs to be delivered from one node to another. Long forwarding paths in turn translate into long delivery latencies, high delivery failure rates and slow routing convergence.

- **Dynamic network topology:** The network topology of a MANET can be very dynamic due to the mobility of the wireless devices. Unlike wireline networks in which links between network devices can be very stable, the links between wireless devices break frequently. To keep a MANET protocol operational, a lot of control burden is involved in dealing with the dynamic topology. MANET protocols have to struggle to keep the balance between performance and efficiency.

The dynamic topology of a MANET also leads to the lack of network hierarchy in a MANET. A wireline network can be easily constructed as a hierarchical network, using routers/switches to carve the network into subnets. This is because devices in a wireline network don't move around. They can stay in a subnet set up for them. With the dynamic topology and the absence of fixed routers, there is no easy way to bind a mobile device to a specific subnet. Most MANETs are created as flat networks, in which the MANET protocols don't scale well. To improve the efficiency of MANET protocols, complex ways must be designed to create dynamic hierarchies in MANETs.

- **Limited battery life:** Mobile devices in a MANET are powered by batteries. This not only limits the lifetimes but also the radio transmission ranges of mobile devices. A MANET can split into components when a mobile device connecting the partitions together runs out of power. To deal with this problem, many MANET network protocols are designed to make efficient use of power and to prolong the lifetime of a MANET.

1.2 What is Virtual Node Layer

The Virtual Node Layer (VNLayer) [1] is a programming abstraction designed to alleviate the difficulties in MANET networking, as discussed above. The VNLayer creates mobile device clusters and defines "virtual" servers, called "virtual nodes", at fixed locations or predictably changing locations in a MANET.

Different definitions and implementations of the VNLayer have been discussed in theoretical literature [2][3][4][5][6][7]. In this thesis, we use a specific implementation of VNLayer called “the Reactive VNLayer” [1]. In the Reactive VNLayer, each virtual node’s operations are controlled by an automaton driven by incoming messages. For simplicity, for the rest of this thesis, we use the term VNLayer to refer to our VNLayer implementation.

In this thesis, we use the VNLayer abstraction with virtual nodes defined at fixed locations. A mobile ad-hoc network is divided into regions at fixed geographical locations. Within each region, a subset of the physical mobile devices elects a leader, which processes and responds to incoming protocol messages. Non-leaders maintain replicated states which are consistent with the leader’s state and work as backup servers. In each region, this set of nodes hence emulates a virtual node. To physical mobile devices in a region, a virtual node works as if it is a fixed local server. Now, within a MANET, we have a matrix of virtual nodes/servers defined at fixed locations, which can cooperatively provide services in a distributed way.

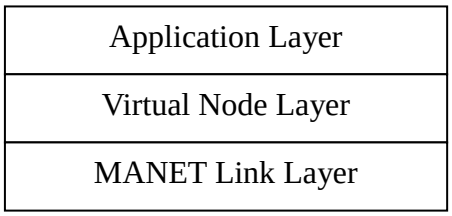

Figure 1-1 The VNLayer works between the MANET and Applications as a programming abstraction

Figure 1-1 shows the relationship between a MANET and the VNLayer. In this thesis, we define application layer protocols to be any protocols running on top of the VNLayer. This can include routing protocols (which would be considered network layer in the Internet or OSI model) and transport layer protocols.[1] As a programming abstraction, the VNLayer handles tasks such as node location checking, leader election, and state synchronization. It also provides a set of user interface functions that can be used by the application layer to pass packets and state to the VNLayer. The application layer also must implement a number of interface functions required by the VNLayer so that the VNLayer can use them to pass packets to the application layer and get/save state to the application layer. At the bottom, the VNLayer interfaces with the MANET link layer. It passes packets to the link layer and receives incoming packets from the link layer. It may also elect to use link layer services such as address resolution (ARP), RTS-CTS and data packet acknowledgement.

1.3 An Example of VNLayer based Data Forwarding

Figure 1-2 shows an example of a packet being forwarded through a 12-region MANET using VNLayer-based routers.

[1] This application layer is different from the application layer in the OSI 7 layer network model.

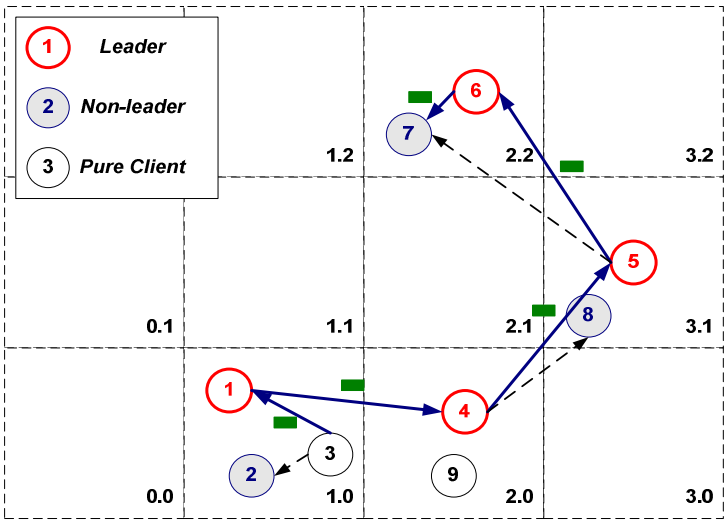

Figure 1-2 Illustration of VNLayer based MANET packet forwarding.

In each region, a leader node and a number of non-leader nodes emulate a router. For example, in region 2.2, the virtual node is emulated by node 1 and node 2 and node 1 is the leader node. Pure client node 3 (a client node that doesn't emulate a virtual node) in region 1.0 sends out a packet destined for client node 7 in region 2.2. The packet is first processed by local leader node 1 and non-leader node 2. Leader node 1 forwards the packet to region 2.0. As a backup router, non-leader node 2 buffers the packet it tries to forward to region 2.0 in its sending queue. When node 2 overhears the packet forwarded by leader node 1, it removes the matching packet from its sending queue. The virtual node emulated routers in region 2.0, 3.1 and 2.2 then forward the packet all the way to the destination node 7. Node 4, 5 and 6 are the nodes forwarding the packet. Node 8 works as a backup router for node 5. Node 7 is the destination node. It also works a backup router for node 6. The dotted arrows indicate forwarded packets heard by the backup routers. Node 9 is another pure client node that is not involved in the forwarding

at all. To a client node, a virtual node in its region works as a fixed router, although it is emulated by multiple physical nodes.

1.4 Benefits of Using the VNLayer

There are a number of benefits of using the VNLayer. First of all, the VNLayer works as a clustering scheme that creates a level of hierarchy in a MANET. This reduces the number of nodes that has to handle a distributed network service. It also reduces the work load each distributed server has to handle. Therefore, VNLayer based services can be more scalable than services that run over a flat MANET.

In addition, the virtual nodes are defined at known and fixed locations. This makes the topology of the overlay network formed by the virtual nodes stable and predictable. It can also make the communication between remote virtual nodes easier. For example, to forward a packet to a remote virtual node, when all the virtual nodes are up, a virtual node can simply relay the packet to a virtual immediate neighbor node that is closest to the destination region and expect a good delivery ratio[2].

As a generalized programming abstraction, the VNLayer hides many MANET complexities from programmers. Because programmers only need to deal with the VNLayer user interface, rather than dealing with a set of highly unpredictable physical nodes, they can deploy applications on both the client side and these virtual static servers with greater ease and efficiency.

[2] This is used in our address allocation protocol in CHAPTER 5.

Furthermore, on mobile nodes emulating the virtual node in a region, application states can be kept consistent between leader and non-leader nodes in a region using the state synchronization mechanism. Hence, the virtual node in a region can maintain persistent state and be fault tolerant even when individual physical nodes might fail or leave a region. This state replication capability also helps make the links between neighboring virtual nodes more reliable.

Finally, the geographical location based clustering makes the clustering job trivial and robust. In the VNLayer, mobile nodes check their geographical locations to determine the clusters they are in. Each time a node's region changes, it simply joins a new cluster. When a cluster's head leaves the region, the nodes left in the region still stay in the same cluster. In dynamic clustering schemes, the membership changes in a cluster can cause other clusters to re-cluster. This is known as the "rippling effect" [8]. This problem doesn't exist in VNLayer clustering due to the geographical location based cluster setting.

1.5 Limitations of the Virtual Node Layer

One assumption we make in our implementation of the VNLayer is that all mobile nodes can find their geographical locations at any time. While this assumption brings great convenience for clustering and gives VNLayer based clustering an advantage over other dynamic clustering schemes, equipping every mobile device with GPS capability is expensive. There is a lot of research [9][10][11] on node localization services that allows the majority of mobile nodes in a MANET to infer their locations based on the location of a small subset of mobile nodes in the MANET that are equipped with GPS. For example,

in [9], Lingxuan Hu et. al. designed a low cost Monte Carlo method in which mobile nodes make random predictions about their locations at the end of next time interval and use observed beacon messages from seed nodes who know their locations to filter bad predictions. The predicted locations finally converge on the actual locations of mobile nodes. Since precise localization is not necessary[3] for the operation of most VNLayer based applications, when not all nodes can have GPS capability, a low cost localization algorithm can be used by the VNLayer to determine a mobile node's region.

As a clustering scheme, the VNLayer improves the scalability of MANET protocols. However, clustering comes with its cost. Within each cluster, control messages have to be generated by mobile nodes to do leader election and maintain leadership. In addition, during leadership changes, a cluster could stop functioning for a period of time when the leader of the cluster is missing.

In addition, the VNLayer is designed to maintain consistent application state among leader and non-leader nodes in each region. This also requires exchange of control messages between the leaders and non-leaders to facilitate state synchronizations.

The VNLayer message overhead is composed of the clustering overhead (leader election overhead) and state synchronization overhead. The VNLayer needs to be carefully designed in order to keep the VNLayer message overhead low.

[3] We only need a node that resides in the geographical area set up for a region or that is very close to the area to identify itself with the region.

For applications running over the VNLayer, a VNLayer header needs to be added to each application message. This increases the traffic overhead when an application is implemented over the VNLayer.

Although the VNLayer state synchronization mechanism can guarantee a certain level of state consistency between leader and non-leader nodes, a non-leader node whose state is out of sync may have to take over a region before it is able to get its state synchronized. As we shall see later, tricky issues (for example, routing loops) can arise when this happens.

In the existing implementation of the VNLayer [1], local broadcast is extensively used to ensure all physical nodes emulating a virtual node in the same region can hear a message for the virtual node. Because link layer capabilities such as address resolution, RTS/CTS and data packet acknowledgment can't be used on broadcast messages, broadcast is more susceptible to transmission failures than unicast messages. As thoroughly discussed in [12], messages sent by broadcast not only are subject to higher loss rate by themselves, they also interfere with other packets, including other broadcast messages. Excessive use of local broadcast can limit a protocol's effectiveness in conveying messages to the whole network.

To deal with these limitations, we created a more realistic link layer mode, extended the existing VNLayer model and optimizations on our VNLayer implementation.

1.6 Research Objectives

The first research objective of this work is to find out whether the VNLayer approach is a practical way to adapting wireline based protocols to MANET and to improving the efficiency and reliability of existing MANET protocols. The second objective is to design and verify techniques that can be used to alleviate the impact of the limitations of the VNLayer, as introduced above in section 1.5 . To achieve these objectives, we designed an ns-2 based VNLayer simulator, VNSim, and conducted extensive simulation studies on a number of VNLayer based applications.

1.7 Overview of the Simulation Studies

In this section, we give an overview of the simulations studies we did on VNLayer based address allocation, reactive routing and proactive routing in MANET.

1.7.1 MANET Address Allocation over the VNLayer

To validate the intuition that the VNLayer approach can be used to adapt wireline protocols to MANET, a wireline protocol, DHCP, is adapted to run over the VNLayer. We pick DHCP for our first case study for the following reasons. First, DHCP is a simple and important wireline protocol. Second, based on our survey, address allocation in MANET is difficult. Existing address allocation protocols for MANET are either node robust enough or not scalable for large networks. Third, the state replication capability of the VNLayer suits the strict needs of address allocation application on state consistency very well.

The research goal here is to find out whether a VNLayer based distributed address allocation system can actually work in a MANET and whether the VNLayer generates too much control overhead.

1.7.2 Reactive MANET Routing over the VNLayer

Ad-hoc On-demand Distance Vector routing [13] (AODV) is one of the most popular MANET routing protocols. The main strength of AODV is its on-demand nature. Since forwarding paths are only created when there is a need for the path, there is no need to use periodic routing messages to maintain routes once there is no more traffic using them.

AODV also has some weaknesses that may be addressed by using the VNLayer approach. A major weakness is that AODV's route discovery is flooding based. Each route discovery involves every single physical node in a network. This makes AODV's routing overhead proportional to the number of nodes in a MANET. When a MANET contains a large number of physical nodes, AODV's route discoveries not only cause broadcast storms but also are unreliable. Many discovery failures can occur. The use of expanded ring search and local repair only partially alleviates the problem because every node receiving an RREQ message might still need to forward it or respond to it.

Another problem of AODV is that the protocol picks routes based only on the path length and path freshness, without considering the stability of the route. For example, a downstream router that has a shorter route could move away. When the motion rates of nodes in a MANET are high, the routes created by AODV might fail frequently, leading to large number of route discoveries.

A cluster-based scheme like the VNLayer approach is a natural solution to the two problems above. By implementing AODV over the VNLayer, we expect the protocol to generate less routing overhead and to create routes with better stability. This is because the on demand routing is conducted by a set of virtual nodes, each of which takes care of a region at a fixed location. Only the virtual nodes in a MANET, rather than every physical node, need to send and forward routing messages. The topology and connectivity among the virtual nodes at fixed locations are much more stable, especially when individual physical nodes are moving around quickly.

The goal of this simulation study is to find out whether a VNLayer based AODV (VNAODV) can provide better routing performance than standard AODV, in terms of data message delivery ratio, routing path length, delivery latency and traffic overhead.

In addition, during the implementation and performance evaluations, failure modes of the VNLayer based AODV protocol were investigated. For example, there are cases in which a region doesn't have a leader or has more than one leader. For example, the current leader in a region might leave and the leadership needs to be transferred to a non-leader. The state on different virtual nodes may not be synchronized. The systems need to be engineered to be resilient to such failures.

1.7.3 Proactive MANET Routing over the VNLayer

Due to the dynamic network topology of MANETs, traditional proactive routing protocols won't work in a MANET due to the heavy routing traffic overhead. The VNLayer approach provides a good way to adapt wireline based proactive routing

protocols to the MANET. With the VNLayer, MANET routing can be split into two levels, the intra-region routing and inter-region routing. The intra-region routing is trivial. Inside a region, each physical node uses the virtual node as the default gateway. Physical nodes can communicate with each other directly. The inter-region routing is handled by the overlay network of virtual nodes. Routing in the overlay network involves only the virtual nodes rather than all the physical nodes. In addition, the overlay network can have a stable topology when the MANET is dense. This is not only because regions are defined at fixed geographical locations. It is also because each virtual node is emulated by a number of physical nodes. The failure on a single physical node won't necessarily cause a virtual node to fail.

A simple version of RIP [14], is implemented over the VNLayer. The research objective here is to find out whether wireline based proactive routing application can be easily adapted to MANET using the VNLayer approach. Again, the major concern is whether the VNLayer based RIP protocol will cause high message overhead. As a proactive routing protocol, the routing traffic overhead of RIP will be proportional to the total number of mobile nodes in the network. This may hurt the performance of the VNLayer based RIP protocol. However, with the benefits brought by the VNLayer approach, we expect the protocol to perform reasonably well.

1.7.4 Scope of Optimizations

In order to improve the performance of VNLayer based applications, optimizations were done both in the VNLayer and in the application layer. The majority of our optimizations

are done inside the VNLayer. They can benefit any application running over the VNLayer.

In the MANET research community, many research are done on optimizing MANET routing protocols through techniques for improving the route maintenance [15], optimizing forwarding paths [16], etc. Since the objective of this research is to verify the feasibility of adapting routing protocols to MANET using the VNLayer approach, we limit our optimizations at the application layer to the ones that are used solve problems introduced by the VNLayer approach, rather than general techniques that can be applied to MANET routing. We believe the existing optimizations designed by other researcher can be easily adopted by virtual node emulated routers.

1.8 Structure of the Thesis

This dissertation is structured as follows. CHAPTER 2 discusses related works on MANET address allocation, MANET routing. CHAPTER 3 introduces the link layer model and VNLayer model we started this research with and the extended models we designed in order to improve the performance of VNLayer based applications. Major implementation choices we investigated in the simulation study on VNLayer are also discussed in this chapter. CHAPTER 4 presents the detailed design of our VNLayer implementation. CHAPTER 5 presents the design of our VNLayer based MANET address allocation protocol. CHAPTER 6 presents the design of VNAODV, a VNLayer based reactive MANET routing protocols adapted from AODV. CHAPTER 7 introduces the design of a proactive wireline routing protocol, RIP, adapted to MANET using the

VNLayer. CHAPTER 8 presents our simulation results on MANET address allocation and MANET routing over the VNLayer. Conclusions and future works are given in CHAPTER 9.

CHAPTER 2. Background

In this chapter, we go through a number of related works in the two areas our VNLayer based applications are developed for, MANET address allocation and MANET routing.

2.1 MANET Address Allocation

A robust address allocation scheme is critical to successful message delivery and correct routing operation. However, address allocation in a MANET is difficult. There is no centralized entity that can provide DHCP [17] service because any mobile node may leave the network at any time. Because most mobile devices are power constrained, using any single mobile device as a dedicated server would greatly shorten the lifetime of the device. Mobile devices may move around quickly and the wireless link between nodes may fail any time due to message collision and congestion. Portions of a MANET may separate and merge together frequently. During network partitions, the same address may be picked by different mobile devices (duplication) and addresses used by departing devices may never be recovered (address leakage). A good MANET address allocation protocol should generate little message overhead. It should be distributed, resilient to node/link failure, avoid both address duplication and leakage and be able to do all this in the presence of network partition and merging.

2.1.1 IP Address Auto-Configuration for Ad Hoc Networks (IAAC)

A simple solution to address allocation was presented by C. Perkins, et al. in [18]. The solution assumes a MANET uses DSR [19] or AODV [13] as its routing protocol. When a mobile node joining the network needs an address, it first picks a random address from a "temporary" address pool (1-2047 from the address block 169.254/16) as its IP address. Then, it picks an address from a "permanent" address pool (2048-65534 from the same address block above) and floods an AODV or DSR route request message (RREQ) to the network. The use of the temporary address is to allow RREP messages to be forwarded back to the new node. If there is any mobile node using the address picked, there is supposed to be a response (RREP) message from an AODV or DSR router. This way, the new node knows the permanent address is not available. It picks a new address and floods another RREQ message. Otherwise, to make sure the address is indeed not in use, the new node repeats the flooding of the RREQ message a few more rounds before it starts using the address as its permanent address.

IAAC is a simple solution to MANET address allocation. However, there are a number of limitations. First, IAAC relies on the assumption that some sort of on-demand routing[20] protocol is running in the MANET to support the use of RREQ messages. In addition, since IAAC doesn't detect and handle address duplications, when a network partitions, multiple mobile devices can have the same address. This becomes a problem when the partitions merge.

Finally, the use of the small "temporary" address pool can also result in the case in which multiple new node are using the same address during their search for available addresses.

When this happens, the RREP messages generated for the RREQ messages can be returned to the wrong node. Missing RREP messages destined for it, a node might start using the “permanent” address it picked. This leads to address duplication.

There is also a small chance that two new nodes, although with their temporary addresses picked differently, pick the same permanent address that is not used by any node in the network. Since there will be no response to either node’s RREQ messages, they’ll start using the same address.

As a summary, IAAC has problems with address duplications and network mergers. In addition, flooding is used by IAAC to send the RREQ messages. Each time when a new node enters a MANET, there will be a few rounds of RREQ storms in which each mobile node in the network has to forward the RREQ message. This can affect the performance of other applications in the network.

2.1.2 MANETConf: Configuration of hosts in a mobile ad hoc network

MANETConf [21] is more complex than IAAC. In MANETConf, each mobile node maintains two sets of addresses for address allocation. One set, “allocated”, holds the addresses that, to the knowledge of the mobile node, have been allocated. The other set, “pending”, holds addresses that are in the process of being allocated. Every mobile node knows the range of addresses that can be used by the whole network, therefore, any address that are not in the two sets above are “free” addresses.

In MANETConf, a new node entering the network doesn't launch its search for available address by itself. Instead, it picks one of its immediate neighbors as its address allocation proxy. If a proxy can't be found, the new node is in a new partition in the network. It picks a random address and starts using it.

Since the proxy node already has an address, it can communicate with other nodes in the network in a reliable way. The proxy handles the address allocation and eventually sends an available address to the new node. The communication between the new node and the proxy can use MAC address, rather than an IP address because it is a one hop communication.

The proxy node does the address allocation as follows. It picks a free address based on its knowledge about the "allocated" and "pending", puts the address in its record of "pending" and floods a QUERY message to the network, asking the entire network to confirm the selection. Receiving the query messages, a mobile node sends back a NO message to reject the selection if the address in question is already in use or marked as "pending", based on the two set of addresses it maintains. Otherwise, the mobile node sends back a YES message to confirm the selection. In addition, the proxy node puts the address in question in its record of "pending" to prevent the address from being picked locally.

The proxy node needs to collect YES messages from all the addresses in its "allocated" set before it can send the attempted address to the new node as an available address. If the proxy node receives YES messages from all the addresses in its "allocated" set, it means

every other node in the network has confirmed the selection. It puts the address it picked in the “allocated” set and tells the new node to start using the address. It also floods another message telling every node in the network to move the address it picked for the new node into their “allocated address” sets. Therefore, every node in the network knows the address is in use.

If at least one NO message is received or at least one YES messages is missing from some node, the proxy node picks a different “free” address and repeats the procedure above.

If not moved to the “allocated” set, an address in the “pending” eventually expires and becomes a free address and can be used in address allocations.

Because a new node can move around before it gets its address allocated, by the time its proxy node gets the address allocation done, it may have moved out of the radio range of the proxy node. To solve this problem, when a new node moves away from its proxy node, it picks a new proxy and asks the new proxy to contact the old proxy for the address allocated for it.

Address leakage can happen when a node using an address leaves a network without giving back its address to the network. In MANETConf, an AddressCleanup message is flooded by a node about to leave the network to ask all the nodes in the network to remove its address from their “allocated” set. If a node didn’t have a chance to send out the AddressCleanup message before it leaves the network, MANETConf can still reclaim the address. During address allocations, due to the absence of a mobile node, a proxy

node can never collect enough YES messages. If a proxy node notices that it can never get either a YES or NO message from an address after a number of QUERY messages are flooded. It sends out an AddressCleanup message for the “inactive” address, so that the address can be moved back to the set of free addresses on every node.

There is no need to handle network partition in MANETConf because when it happens, addresses used by a partition will eventually be set to free in other partitions. However, when partitions merge, there has to be a way to resolve duplicate addresses. To do this, in MANETConf, each partition is identified by a 2-tuple of (address, UUID). Where the address is the lowest address in a partition and UUID is a unique number generated by node with the lowest address in a partition.

Therefore, when a network partitions, one partition can preserve its identifier because it still has the node with the lowest address. The nodes in the other partition eventually realize that the node with the lowest address is gone and generate an AddressCleanup message. By checking the “allocated” set, the node with the lowest address in the other partition generates a new UUID and broadcast it to the partition. This partition gets a new identifier.

Each time two nodes discover each other as immediate neighbors, they exchange their partition identifiers. If their identifiers are different, a partition merging procedure starts. The two nodes first exchange their “allocated”. Then, they flood the “allocated” set received from the other partition in their own partition. Receiving the “allocated” set, each node in a region combines it with its own “allocated” set. If there is any duplicate

address allocations detected, the node with fewer TCP connections gives up its address and requests a new one.

MANETConf is a comprehensive solution that solved all the problems that can arise in MANET address allocation. However, each address allocation in MANETConf involves every node in the network. The protocol's complexity and extensive use of flooding on all kinds of message both limit its scalability.

2.1.3 Zero-Maintenance Address Allocation (ZAL)

As we have discussed, the main problem of IAAC and MANETConf are their inefficiency because each address allocation attempt in the two protocols has to be confirmed by all the nodes in the network through flooding. Zero-Maintenance Address Allocation is a protocol [22] aiming at improving the efficiency of address allocations in MANET.

Preventing and detecting duplicate address allocation quickly are the key difficulties in MANET address allocation that lead to the complexity. To reduce the difficulty of duplicate address detections (DAD), ZAL distributes addresses using a different way.

In a network, the first node owns the entire pool of addresses that can be used by the network. The first node picks an address from its address pool for itself. Then, each time when a new node joins the network, it asks its neighbors for addresses. Receiving the request from a new node, a mobile node offers a slice of its address pool to the new node. Receiving multiple offers, the new node accepts the offer with the largest number of addresses and takes one address from the offered address pool. Apparently, the control

overhead of this kind of one hop address allocation is very low because no DAD is necessary in ZAL.

There are a few problems ZAL has to handle though. First, since the way mobile nodes split their address pool is like a binary splitting, the size of the address pool shrinks quickly with the increase of the network diameter. If the first node has an address pool of 2^n addresses, a node that is more than n hops away from it might not be able to get any address. To solve this problem, a temporary address pool is set up for nodes to pick their addresses from when they can't receive any offer. Once a node meets a node that can offer permanent addresses to it, it gives up the temporary address.

Now, when multiple nodes pick their addresses from the temporary address pool, duplicate address allocation can happen. DAD is needed again. Since the chance that this happens is low, it is expected that the control cost of DAD is low.

ZAL also designed a one hop distribution equalization algorithm in order to optimize how mobile nodes split their address pools for their neighbors. Immediate neighbors exchange information about the size of their address pools. Based on this information collected, a node can find out the total number of addresses owned by its immediate neighbors and compare it with the size of its own address pool. If a node owns a large address pool compared with the number of addresses owned by its neighbors, the node distributes a portion of its addresses to its neighbors. By doing this, the address pool distribution among mobile node is fairer and address depletions happen less frequently. The control

overhead of this optimization is low because it only involves the message exchanges between 1 hop neighbors.

Second, ZAL has to deal with partition mergers. ZAL uses a similar procedure as the one used by MANETConf to handle partition mergers. Each partition is identified by an id generated by the first node of a network. When two nodes with different partition ids meet each other, ZAL uses the following way to give nodes in the smaller partition the addresses belong to the larger partition. Starting from the border between the two partitions, nodes in the smaller partition give up their address pools to neighbors that are still in the smaller partition and request for addresses from the larger partition. Recursively, all the nodes in the smaller partition get addresses from the large partition.

Compared with MANETConf, ZAL is solution with much lower control overhead. However, it doesn't use address efficiently. A network with diameter n requires the order of 2^n addresses. In addition, address leak can happen in ZAL when a node crashes before it returns the address pool it has. The author claims the probability that this happens is low under a given model of the lifetime a mobile node. However, if the node that fails happens to have a large chunk of addresses, its impact is big. Third, the handling of network merging is not efficient because collecting the information on network size could involve $O(n^2)$ complexity[4].

[4] The collection of this information is not explained clearly in the paper. The overhead of exchanging this information throughout a partition can be costly.

2.1.4 MANET Address Allocation in IPv6

Another solution to MANET address allocation [23] uses a combination of IPv6 MANET prefix[5] and a node's MAC address as its address. This solution takes advantage of the abundant address space provided by IPv6 and eliminates the need for dynamic address allocations in a MANET. As pointed out in [21](section III.C), this solution assumes that MAC addresses are unique for each mobile device, which is not always true. In addition, when IPv6 is not available and the address space is limited, dynamic address allocation is still necessary for MANET applications.

Another solution [24] uses a combination of a mobile node's MAC address and the network address provided by a designated gateway so that a mobile node in a MANET can communicate to the global Internet. This solution is no longer completely a MANET address allocation scheme since a fixed gateway node exists in the MANET as an address allocation server.

2.1.5 Summary

The current solutions are either not reliable or too expensive due to the use of message flooding. In our simulation studies, we have found that flooding introduces unacceptable overhead and causes large number of message collisions. Therefore, none of the current solutions can support large networks. In CHAPTER 5, we introduce our VNLayer based address allocation application, VNDHCP, which does address allocations in a clustered MANET and doesn't use flooding for control messages. On top of that, VNDHCP is also free of address leakage and duplication in face of network partitions and mergers.

[5] The prefix is FE:C0:0:0:FF:FF

2.2 MANET Routing

Compared with routing in wireline networks, routing in a MANET is difficult because of MANET's limited radio range and channel bandwidth, collision prone channel, flat network architecture and dynamic network topology. Popular routing protocols such as the distance vector routing protocol RIP[14] and the link state routing protocol OSPF[25] can't be used directly in MANET. There are three reasons. First, the wireline based routing protocols usually assume that the network topology is relatively stable. However, the highly dynamic network topology in a MANET leads to frequent route updates and slow routing table convergence. Second, the flat network architecture in MANET requires each router to have a route entry for every destination, the periodic routing information exchange in tradition routing protocols creates much heavier control traffic. Third, the wireless channel is shared between adjacent mobile nodes. The heavy routing overhead and limited channel bandwidth can cause frequent packet losses due to message collisions and congestions.

Therefore, new protocols that suit the special needs of MANETs must be designed. The routing problem has been studied by the MANET research community for many years. Various routing protocols in different categories have been proposed.

Using Zhou's classification in [20], routing protocols for MANET can be classified into the following categories.

- Topology-based routing protocols
    - Proactive routing protocols
    - Reactive routing protocols

- Geographical-based routing protocols

Topology-based routing protocols are routing protocols calculating the best route to a destination based on topology information collected from the network. Within this category, **proactive routing protocols** are routing protocols that calculate the routes to all the destinations before a transmission actually happens. **Reactive routing protocols** are routing protocols that calculate the route to a destination only when it's necessary for a transmission.

Geographical-based routing protocols are routing protocols that calculate routes based on the geographical locations of the destination node and neighboring nodes. This set of routing protocols requires that each mobile node can determine its current location or can access a distributed location service that can return the current location of any mobile node in the network. With this knowledge, a mobile node can make local forwarding decisions based on the geographical location of the destination. Geographical based routing suits MANET because it requires fewer routing information exchanges and is more scalable. The disadvantages of such routing scheme are: First, each node needs to have GPS-like capability, which can be power consuming, and, second, the location service may introduce extra message overhead. One of the most popular geographical based MANET routing protocol is Greedy Perimeter Stateless Routing (GPSR) [26]. GPSR uses greedy routing to relay packets to mobile nodes that are closer to the destination than the current router. When no such nodes can be found before a packet reaches the destination, GPSR uses face routing to relay packets toward the destination using mobile nodes that are farther away from the destination than the current router.

In this section, we first discuss a number of popular proactive and reactive MANET routing protocols. Then, we introduce two Cluster based routing protocols that are designed to improve the efficiency of MANET routing. Because our VNLayer based routing is closely related to this set of MANET routing protocols, I summarize the differences between the cluster based protocols and the VNLayer approach at the end of this section.

2.2.1 Proactive Routing Protocols

2.2.1.1 Destination Sequence Distance Vector (DSDV)

DSDV [27] is one of the earliest MANET routing protocols. As the name suggests, DSDV is a distance vector routing protocol based on the classical Bellman-Ford algorithm [28] (RIP is a wireline protocol using this algorithm). Its most important contribution is the use of a destination sequence number in the routing protocol.

Now, in the DSDV routing table, other than the destination id and metric, each route entry for a destination also contains a sequence number that is originally generated by the destination in order to indicate the freshness of a route.

In DSDV, each router periodically broadcasts Update messages, each of which contains its entire routing table or changes to its routing table, to its immediate neighbors. Routers update their routes with incoming Update messages. For a destination, if the router doesn’t have a route and the Update message contains a route, the route is installed. If both the router and the Update message contain a route for a destination, the router replaces its route with the one in the Update message if the latter is tagged with a greater

sequence number or the latter is tagged with the same sequence number and has a better metric.

In short, a newer route or a better route will be chosen.

Although DSDV is not much different from traditional wireline-based distance vector routing protocols, the use of the sequence numbers reduces the chance of routing loops. This feature is used by many other MANET routing protocols. As a proactive routing protocol, DSDV doesn't scale well because every single node in a network has to do the periodic broadcasting of routing tables.

2.2.1.2 Optimized Link State Routing (OLSR)

Another popular proactive MANET routing protocol is OLSR [29]. OLSR is a link state routing protocol similar to OSPF. However, in order to adapt to the MANET environment, important optimizations are done in OLSR to drastically reduce its control overhead. (Control overhead is the reason why traditional link state routing protocols can't be used in MANET)

As a link state routing protocols, OLSR routers still construct routing tables using flooded link states that are generated by each router in the network to announce its list of immediate neighbors. The difference is, Multi Point Relay (MPR) is used in OLSR.

MPR works as follows. Each router exchanges beacon messages with its immediate neighbors and maintains a list of its one hop neighbors. In addition, in the beacon messages, a router also gives its immediate neighbors its list of one hop neighbors.

Therefore, based on the beacon message exchange, a router can also maintain a list of its 2 hop neighbors and knows the 2 hop topology of routers around itself.

Based on this knowledge, a router picks a minimal subset of its one hop neighbors such that all of its two hop neighbors can be reached through (aka. covered by) this set of nodes. The one hop neighbors chosen are called the MPR nodes of a router. A router then informs its MPR nodes that they are chosen as its MPR nodes.

Now, when a router broadcasts its link state to the network, only the neighbors that are chosen by the router as MPR nodes re-broadcast the link state. The link state messages are in turn forwarded by the MPR nodes of the MPR nodes of the originator of the messages. This way, the number of nodes that participate in the flooding of the link states can be greatly reduced.

In addition, only routers that are chosen by at least one router as its MPR node generate link state messages. Finally, a router's link state message only contains the nodes that have chosen it as their MPR nodes (MPR selector nodes).

This way, both the number of routers originating link states and the size of each link state message can also be greatly reduced, while the route toward every single node in the network can still be found with the Dijkstra algorithm[30].

OLSR can greatly reduce the routing overhead. However, in a flat network with large number of nodes, its route overhead is still proportional to the size of the network, limiting the scalability of the protocol.

2.2.2 Reactive Routing Protocols

Reactive routing protocols only do routing when there is traffic. However, since routes are often discovered reactively, there will often be a route discovery delay before a route can be used to forward data packets. This category of routing protocols is more suitable when the network size and data traffic load are moderate, the network topology is very dynamic.

2.2.2.1 Dynamic Source Routing (DSR)

Dynamic Source Routing [19] is a source routing protocol that can work in a MANET with either undirected or directed links. Each DSR router maintains a route cache (as opposed to routing table) that records the entirety of the routes the router has learned for each destination. Each destination can have multiple route entries in the route cache. In DSR, when a router needs to send a data packet to destination but it doesn't have a route, it does a route discovery by flooding an RREQ messages to the network. The RREQ message carries the source of the route discovery, a discovery sequence number generated by the source of the route discovery and a vector recording the sequence of routers it traverses. The first two fields are used by routers to avoid forwarding RREQ messages of the same route discovery more than once. The last field is used to facilitate source routing.

When an RREQ message arrives at a router, the router first creates a routing table entry for the initiator of the RREQ by reversing the sequence of routers traversed by the RREQ messages.

Then, the router checks to see if it has a route to the destination asked for by the RREQ message or the router itself is the destination asked for. If so, it returns an RREP message to the initiator of the RREQ message. The route the router has will be combined with the list of routers traversed by the RREQ message and put in the RREP message. To forward the RREP message back to the originator of the RREQ message, a router can use the source route it just learned from the RREQ message, or, when the network links are directed, the router sends another RREQ message for the initiator, with the RREP message attached to it. This way, the RREP message can eventually reach the initiator of the route discovery.

Receiving an RREQ message, if a router doesn't have a route, it re-broadcasts the RREQ message with its id added to the source route carried in the RREQ message. Eventually, the RREQ message can reach either the destination or a router that has a route toward the destination. In the flooding of the RREQ messages, a router only forwards the RREQ messages for the same route discovery (identified by a discovery id and the initiator's node id) once.

Receiving an RREP message, a router puts the route carried in the message in its route cache. When the initiator router receives the RREP message for its route discovery, it sends out the data packets it buffered during the route discovery. Each forwarded data packet carries a source route, so that the data packets are forwarded along the path picked by the first hop router.

When a broken link is detected due the absence of acknowledgement for data packets, a router sends a ROUTE_ERROR message back to the sender of the data packet. The ROUTE_ERROR message carries both sides of the broken link. Receiving the ROUTE_ERROR message, each router that has route cache entries using either side of the broken link as a downstream router removes those entries. Receiving the ROUTE_ERROR message, the sender of the data packets starts another route discovery if there is no other route available.

DSR is simple and can support networks with directed links. Its use of source routing helps prevent loop formation. As a reactive routing protocol, no periodic beacons or routing updates are used in DSR. In addition, the use of route cache rather than routing table helps reduce the number of route discoveries needed because for each destination, multiple alternative routes can be cached.

DSR also has a number of problems. The use of source routing increases the size of both routing messages and data packets being forwarded. The use of route cache also uses more memory than other approaches. Finally, the use of flooding in a flat MANET is costly when the number of mobile nodes in the network is large.

2.2.2.2 Ad-hoc On-Demand Distance Vector routing (AODV)

AODV [13] is one of the most popular MANET routing protocols. The core algorithm of AODV is very close to DSR. The two protocols both operates in two stages, route discovery and route maintenance. However, as the name suggests, AODV is not a source routing algorithm. AODV routers use routing tables rather than route caches. That is, for

each destination, only one route is maintained a routing table. AODV was proposed by Charles E. Perkins, et al., who also proposed DSDV. This might be the reason why AODV is similar to DSDV, in that destination sequence numbers, rather than source routes are used to ensure freshness of routes and prevent loop formations.

In AODV, when a router needs to forward a data message but it doesn't have a route, it buffers the data message and sends out a RREQ message. An AODV RREQ message carries a 6-tuple including source id, source sequence number, destination id, destination sequence number, BCAST id, hop count. The source id is the address of the initiator of the route discovery, the source sequence number is a monotonically increasing number generated by the initiator to ensure the freshness of routes toward it. The destination id is the address of the destination of the data packet. The destination sequence number is the largest sequence number generated by the destination node known to the initiator of the route discovery (If an initiator knows nothing about a destination's sequence number, it uses 0). The BCAST id is a monotonically increasing number generated by an initiator node to uniquely identify a route discovery. The hop count starts with 1. It carries the number of hops the RREQ message has traversed.

When the RREQ arrives at an AODV router, as in DSR, the router first updates its route entry for the initiator's address using the route and information in the RREQ message. As in DSDV, if the router doesn't have a valid route or, compared with the router's route entry for the destination, if the incoming message carries a fresher route or a route that is the same fresh but shorter, the route in the incoming message will be used by the routing table.

Then, the router checks its routing table to see if there is a route fresh enough for the destination. (By fresh enough, we mean the router has a route with a sequence number no less than the destination sequence number carried in the RREQ message.) If so, the router returns an RREP message to the initiator of the route discovery, using the route it just learned through the RREQ message. Otherwise, it increase the hop count carried in the RREQ message by 1 and rebroadcasts it.

Eventually, the RREQ message can reach either the destination or a router that knows a route to the destination and an RREP message can be returned to the initiator of the route discovery. If it is the destination that receives the RREQ message, the RREP message carries a new sequence number generated by the destination node, indicating the route is the latest. Upon receiving an RREP message, a router updates its route entry for the destination the same way as it updates its route for the initiator node with incoming RREQ messages.

As in DSR, the address of the initiator and the BCAST id are used by AODV routers to avoid forwarding more than one RREQ message for the same route discovery. Route error (RERR) messages are also used to report broken links to upstream routers. When a router receives an RERR message, it checks if its routes for the destinations carried in the message uses the sender of the message as next hop. If so, it disables the route and report the error to its neighbors using another RERR message.

In addition, a “Ring Search” mechanism is used to control the scope of route discoveries in order to reduced the flooding overhead of route discoveries. The basic idea of ring

search is to try route discoveries with smaller TTLs used on the RREQ messages first before the search for routes is expand to greater scopes.

In AODV, a local repair mechanism is designed to allow intermediate routers to fix route failures locally by starting a route discovery themselves. Basically, when a broken link is detected at the 2nd half of a forwarding path, a router buffers the packets it is relaying; sets the routes affected to a “repair mode” and send RREQ packets for the affected destinations. This way, the need for reporting route errors all the way back to the sender of the data message and letting the sender node start a network wide route discovery can be reduced.

DSR and AODV are both on-demand routing protocols that work in similar ways. However, there are major differences between them. DSR uses source routing to avoid loop formations while AODV uses route sequence numbers. In addition, DSR routers use more bandwidth than AODV for routing and data forwarding also due to the use of source routing. On the other hand, AODV routers keep only the freshest route for each destination. DSR routers maintain a collection of alternative routes for each destination. While costly in terms of memory use, DSR responds better to topology changes. Performance comparisons on DSR and AODV in [31] proved that AODV scales better than DSR while performs worse than DSR in face of frequent network topology changes.

2.2.3 Cluster Based Routing

The MANET routing protocols we have discussed so far work with MANETs with no hierarchies. The routing process involves every single node in a network. In addition, due

the heavy control overhead and unreliability of message flooding, those routing protocols usually can't support a MANET that has over 100 nodes. As in wireline networks, hierarchical/cluster based [8] routing is the solution to this problem. By grouping mobile node into clusters each of which has a cluster leader, the inter-cluster routing can be handled by cluster heads/leaders. This way, the number of mobile nodes that has to be involved in the global routing can be reduced and the number of routing entries each router (now the cluster heads) has to maintain can also be much smaller. In this section, we discuss a number of routing protocols that create clusters in a MANET.

2.2.3.1 Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures (CEDAR)

In order to tackle the two problems faced by reactive MANET routing schemes without hierarchies, CEDAR[6] [32] is designed to provide a virtual infrastructure to on-demand routing protocols. CEDAR uses a core extraction algorithm to elect a set of "core" node. In essence, this set of core nodes is an approximated minimum dominating set[7] that cover every single node in a MANET. Each core node then acts as a cluster leader, does routing and forwarding for the mobile nodes in its domain (cluster). This way, the number of nodes need to be involved in routing is now the number of core nodes.

The core extraction algorithm results in a set of core nodes that are at most 3 hops from each other. To perform route discovery over this virtual overlay network, there needs to be a way to for a core node to flood RREQ messages to all the other core nodes. In order

[6] CEDAR stands for Core Extraction Distributed Ad-hoc Routing.
[7] In graph theory, a dominating set of a graph represented as G=(V, E) is a subset D of the set of vertices V such that every edge in E is connected with at least one member in D. A minimum dominating set is the smallest dominating set of a graph.

to solve the problem caused by broadcast based flooding, in CEDAR, a unicast channel is created and maintained between neighboring core nodes. Now, a Core Broadcast mechanism, rather than simple local broadcast, is used to propagate RREQ messages. On a core node, an RREQ message is flooded to neighbor core node using these channels by unicast. This way, the reliability of route discovery can be improved and the interference of the flooding on other packets in the channel can also be reduced.

Core Broadcast in CEDAR is unreliable because the maintenance of the unicast channel requires periodic beacon messages and the channels are subject to frequent failures in a dynamic topology.

An enhanced version of CEDAR, E-CEDAR (E for enhanced) [12] further improves the core broadcast mechanism. In E-CEDAR, every core node maintains a “forwarding set”, in which each address is a node that the core node has to deliver RREQ messages to during a route discovery. A forwarding set of a core node thus needs to include the core node’s 1 hop neighbors that are core nodes, a minimal subset of the core node’s 1 hop neighbors that can be used to cover core nodes that are two hops away, and a minimal subset of the core node’s 1 hop neighbors that can be used to cover core nodes that are three hops away, through nodes dominated by these remote core nodes. Therefore, this “forwarding set” serves similar purpose as the MPR set in OLSR. The difference here is that the “forwarding set” is used to reach core nodes that are within 3 hops.

The “forwarding set” is created using periodic beacon messages exchanged between nodes. However, the unicast channels are created by local computations rather than using

explicit message exchanges between core nodes. Therefore, the enhanced core broadcast is more efficient and more resilient to topology changes than the core broadcast in CEDAR.

In order to further reduce packet interferences, E-CEDAR also modified the 802.11 RTS-CTS mechanism. A NCTS (negative CTS) message is used by a node to reject an RTS request if it finds out the RTS is for a Core Broadcast message that it has already overheard.

E-CEDAR is shown to be able to improve the performance of the two reactive routing protocols, DSR and AODV, we introduced in the previous section. The core extraction algorithm can be used by any application to create a virtual infrastructure in a MANET. There are a few problems with E-CEDAR, though. First, the core-extraction algorithm creates dynamic clusters based on the edge degree of mobile nodes. The dynamic clusters are subject to frequent changes when mobile node moves around quickly. Frequent changes in the set of core nodes can lead to unreliable forwarding routes. In addition, in order to maintain the connection between the core nodes, periodic beacon messages (basically link state messages) still have to be used so that each core node can collect information on its 3 hop connectivity to neighboring nodes. This constant overhead is not correlated with the data traffic.

2.2.3.2 Cluster Overlay Broadcast (COB)

Cluster Overlay Broadcast (COB) [33] works similarly to AODV [13], but with RREQ messages and RREP messages flooded only by cluster heads. In COB, dynamic clusters

are formed using a 1-hop clustering algorithm, Least Cluster Change [34], in order to reduce the clustering changes in face of node mobility. Each data packet sender sends the data packet to its local leader first for routing service, using a short range radio transmission. Upon receiving a data packet, a cluster leader floods (a controlled flooding) a RREQ message to the network using a long range radio transmission. When the destination node receives the flooded RREQ, it responds with an ACK messages, which works similarly to RREP messages in AODV. Upon receiving the ACK message, a cluster leader marks itself as active for the session between the sender node and the destination node. When the originating cluster leader receives the ACK message, it also broadcasts data packet to the network using the long range radio transmission. At subsequent hops, cluster leaders that have been set as active for the session relay the data packet and set themselves as inactive for the session. This way, the data packet is forwarded hop by hop toward the destination node. COB is proved to perform better than DSR [19] by simulations. However, as mentioned above, COB requires mobile nodes to be able to switch between two transmission powers and uses broadcast at each forwarding hop. Furthermore, a connection created by COB through a route discovery can only be used once because a router marks itself inactive for a session between a source and a destination once it forwards a packet for the session. This is going to cause unnecessarily high control overhead when each session contains a large number of data packets.

2.2.3.3 Summary

Cluster based routing protocols such as COB and CEDAR improved the efficiency of MANET routing by creating hierarchies in a flat MANET. CEDAR is a more complex

and practical solution than COB. However, both COB and CEDAR/E-CEDAR uses dynamic clustering, which are subject rippling effects when cluster membership changes. In addition, in each cluster, there is only one cluster head. VNLayer based routing is also a cluster based routing scheme, in which the leader and non-leaders in each region emulate a virtual router. Compared with the existing cluster based MANET routing protocols, the VNLayer approach has the following advantages: First, the VNLayer approach is a generalized programming abstraction. It hides the complexities such as clustering, message buffering and state synchronization from the routing application. In addition, the services provided by the VNLayer can be shared by multiple applications, rather than just the routing application, so that the overall performance can be improved. Second, to our best knowledge, our VNLayer implementation is the first clustering scheme that has the capability of maintaining replicated states in a cluster. A virtual node emulated router can stay functional even when the cluster head leaves a region. Third, the geographical based VNLayer clustering is very efficient and is free of the rippling effect.

CHAPTER 3. Models for the Link Layer and the VNLayer

The Virtual Node Layer (VNLayer) is a general programming abstraction that hides MANET complexities from applications. With this abstraction, programmers only write programs for virtual nodes at fixed geographical locations, emulated by physical nodes nearby, so that they don't need to deal with node motion. In the TCP/IP model [35], the virtual node layer resides between the link layer and the Internet Layer. The VNLayer uses the service provided by the link layer and provides services to applications at the Internet Layer. To define the VNLayer approach, both the link layer and the VNLayer needs to be modeled. In the first section of this chapter, I introduce the models for the Link Layer and the VNLayer defined by Mike Spindel in [36]. The second section defines a more realistic model for the Link Layer and an extended VNLayer model that supports better performance in the presence of message losses. In the third section, I give a review on the VNLayer implementation by Mike Spindel. In the last section, the implementation choices we investigated in this research will be discussed.

3.1 The Basic Link Layer and VNLayer Models

Mike Spindel described models for the link[8] layer and the VNLayer in [36]. In this section, I give a review on the two models. In this thesis, these two models are called the Basic Link Layer model and the Basic VNLayer model.

[8] The term used in [36] is physical. However, in real implementation, the link layer is what the VNLayer is

3.1.1 The Basic Link Layer Model

3.1.1.1 Physical Node

A physical node is modeled as a timed input/output automaton [37] moving arbitrarily in a two dimensional plane with no obstacles[9]. The set of physical nodes is modeled as a finite set of automata. The location of a physical node, say node i, is referred to as $loc(i)$ and its motion rate is bounded by a constant v_{max}.

Location Determination: Each physical node is able to determine its current geographical location and the global time every τ time.

Node Clock: Every physical node has a local clock that runs at the rate of real time and is synchronized every τ time.

Each physical node is able to do arbitrary computation. It is assumed that local computations do not take any time[10]. Physical nodes may suffer stopping failures. That is, when a physical node stops, it stops all local computations and stops sending messages.

3.1.1.2 P-bcast Service

At the link layer, each physical node is assumed to have access to P-bcast, a broadcast service. Physical nodes have different broadcast ranges.

---

built upon. Therefore, we choose to call the layer which provides basic communication service to the VNLayer the link layer.

[9] In [32], obstacles are not considered. However, obstacles not only affect node motion but also affect radio range.

[10] This assumption is reasonable because for the protocols under study, network activities dominate the power and time requirement.

The maximum reliable transmission distance for a physical node i at geographical location src when sending toward geographical location dst is determined by *src* and *dst*.

The P-bcast service guarantees:

1. **Integrity Property:** Every message received was previous broadcast.
2. **Reliable Local Delivery Property:** Every message broadcast will be received by every physical node in-range in a timely manner. When physical node i sends a message, there exists a time$d$, if for a physical node j is located within the reliable transmission distance between *loc(i)* and *loc(j)*. for the entire transmission, then physical node j can receive the message in d seconds[11].

The basic link layer model here doesn't consider message losses. However, in reality, there is no wireless link layer without message losses. In the simulations[12] described in this thesis, message losses are allowed. Thus the second guarantee is not provided by the simulations.

3.1.2 The Basic VNLayer Model

3.1.2.1 Regions

The geographical area of a MANET is subdivided in to regions. It is assumed that the region configuration is known by every physical node. Regions are configured or chosen so that every physical node in a region can reliably send and receive data from every other physical node in the region and neighboring regions.

[11] In the simulation, the function is dependant only on the two locations, not on the node i.
[12] VNE simulated the basic VNLayer without collisions.

A physical node's region is uniquely determined by its location. The set of neighbors is also determined by the region. For all locations *i*, *j* and any physical node *p* at location *i*, if physical node *p* is in a region and *j* is in the same or a neighbor region, then node is within the reliable transmission range of node *p*.

3.1.2.2 Virtual Nodes

Each region hosts a virtual node. A virtual node is an automaton driven by incoming messages. A virtual node's operation is defined completely by a msgReceived handler. With an incoming message, a virtual node can change its state arbitrarily and send out a set of messages using V-bcast.

Virtual Node Clock: A virtual node doesn't have access to a real time clock. Instead, it simulates a clock by synchronizing to timestamps on incoming messages.

There are two failure modes of virtual nodes related to the behavior of client processes[13].

1. If there is no client process in a region, the virtual node for the region has failed.
2. There is a t_{start} associated with the system such that if a physical node stays in a failed region for more than t_{start}, the virtual node for the region restarts with its initial state[14].

3.1.2.3 Client Process

Client processes hosted by physical nodes solicit services from virtual nodes in each region. A physical node can host any number of client processes. For simplicity, for a

[13] Here, it is assumed that every physical node emulates a virtual node.
[14] This can also be regarded as a recovery mode. However, the virtual node state won't be recovered. As we are going to see, when a virtual node restarts, there are tricky complications.

single application, it is assumed that each physical node hosts at most one client process. Therefore, clients can be modeled as a set of timed input-output automata.

A client process has access to the location of its hosting physical node. It also has access to a real-time clock. A client process can communicate with its local virtual node only through the V-bcast service.

3.1.2.4 Virtual Broadcast

For VNLayer based communication, Clients and Virtual Nodes have access to another broadcast service, V-bcast. V-bcast is used by clients and virtual nodes to communicate with other clients and virtual nodes. The V-bcast guarantees the Integrity Property and Reliable Local Delivery Property.

V-bcast provides two additional guarantees. First, it guarantees that if a client or a virtual node is not in a region originating or neighboring a broadcast, it won't receive the message. Second, it guarantees that all broadcast messages will have a total ordering and will be received by all clients and virtual nodes in the same order.

The V-bcast service therefore requires that a virtual node or a client can communicate with any virtual node or client in the same region or in a neighbor region and only with those nodes.

This combination of the reliable delivery and inverse reliability requirements has the consequence that in any region, either all client processes and the virtual node in the

region receive a message or none do. In other words, the transmission is atomic with respect to the regions. This is the atomicity property of the basic VNLayer.

3.1.2.5 Virtual Node State

The virtual node state includes the clock and discrete variables at the application layer. In the absence of message collisions, the VNLayer guarantees that a virtual node maintains its current state as long as the virtual node doesn't fail. When a virtual node does fail and restarts, its state resets to its initial state.

3.1.2.6 Requirements on Applications

In order to use the VNLayer, an application needs to be able to handle messages passed up by the VNLayer. It also needs to allow the VNLayer to read and overwrite its state.

In addition, an application must be able to tolerate virtual node failures and virtual node resets (the two failure modes described in section 3.1.2.2).

3.2 The Extended Link Layer and VNLayer Models

3.2.1 The Extended Link Layer Model

In order to use a more realistic communication model, we extended the basic model for the link layer. The extended link layer model allows message collisions and losses.

3.2.1.1 Physical Nodes

The model for physical nodes is same as the one in the basic VNLayer except that the local clock has the current real time and doesn't need to be synchronized.

3.2.1.2 LL-bcast Service

Instead of P-bcast, each physical node has access to an 802.11 like link layer service, LL-bcast. Physical nodes have different broadcast ranges. The maximum reliable transmission distance between two physical nodes is determined by the sender's location and transmission power and the receiver's location and receiving capability[15].

Compared with the P-bcast service in the basic link layer model, LL-bcast takes message losses[16] into consideration. That is, messages may not be received by a destination node even if it is in-range. In addition, with LL-bcast, a message can be sent with either a broadcast destination address or a unicast destination address. When a broadcast address is used, it is not possible for the sender to determine whether a message has been received by a potential recipient. We call this kind of data transmission "**local broadcast**". When unicast address is used, message transmission is more reliable because the sender can determine before sending the message if the receiver is around and after sending the message if the message is received by the intended recipient[17]. However, using unicast destination addresses in LL-bcast require all physical nodes to listen to all messages and process or discard them appropriately. In addition, the improved reliability with unicast comes with an additional transmission delay because of the time for an acknowledgement from the receiver to the sender.

[15] Receiving capability on mobile devices can be different. For example, a device using a high-gain antenna can communicate with a device out of the regular radio range.
[16] Here, we assume all message losses are due to collisions.
[17] In 802.11, when unicast is used, Address Resolution, CMSA/CA (using RTS-CTS) and link layer acknowledgement and retransmission can be used to detect link failures quickly and improve the reliability of data transmission.

In the extended link layer model, when the sender of a message can't determine the intended recipient node, a broadcast destination address must be used. When the sender can determine the address of the intended recipient, the message uses the address of the recipient. Since all physical nodes listen to all messages, a message sent to a unicast destination can still be heard by every mobile in range. I will call this kind of transmission "**Directed Broadcast**".

Like P-bcast in the basic link layer model, LL-bcast guarantees the Integrity Property but only guarantees the Reliable Local Delivery property in the absence of collisions.

3.2.2 The Extended VNLayer Model

In order to improve the performance of VNLayer based applications, we created an extended VNLayer model. Now, each virtual node has access to a real-time clock. A virtual node or client process is allowed to communicate with other client processes or virtual nodes that are not in local or neighbor regions. This removes the inverse reliable delivery guarantee provided by the basic VNLayer model.

3.2.2.1 Regions

Regions are the same in the extended VNLayer model.

3.2.2.2 Virtual Nodes

Virtual nodes have access to a real-time clock. A virtual node in a region is an automaton driven not only by incoming message but also by timer events. Therefore, a virtual node's operation is no longer defined completely by a msgReceived handler. In addition to

incoming message, with a timer expiration event, a virtual node can also change its state arbitrarily and send out a set of messages using V-bcast.

The use of timers allows actions to be taken exactly at scheduled times. Because network applications often age state so that it expires and is discarded after a period of time, the message driven approach in the basic VNLayer model requires a lot of processing time to locate expired state and expired messages in the buffer. Using timers is much more efficient.

Here, we don't assume each physical node emulates a virtual node[18]. Each physical node that emulates a virtual node is defined as an ***emulator node***. This introduces a change to the failure modes. There are two failure modes of virtual nodes related to the behavior of emulator nodes.

1. If there is no active emulator node in a region, the virtual node for the region has failed.
2. If an emulator node stays in a failed region for more than t_{start}, the virtual node for the region restarts.

3.2.2.3 Client Process

One difference in the extended VNLayer model is that a client process is allowed to receive messages from a virtual node that is not in the client process's local region.

Another difference is that a client process doesn't need to provide the Virtual Nodes with clock information, because the Virtual Nodes already have it.

[18] This is to reduce the number of emulator nodes when a MANET is dense enough.

3.2.2.4 Virtual Broadcast

In the extended VNLayer model, Clients and Virtual Nodes still have access to V-bcast. Like LL-bcast, V-bcast here guarantees the integrity property and guarantees the reliable local delivery property only in the absence of collisions.

V-bcast still guarantees that all broadcast messages will have a total ordering and will be received by all clients and virtual nodes in the same order. However, the V-bcast service in the extended VNLayer model allows a virtual node or a client to communicate with any other virtual node or client in range. That is, V-bcast in the extended VNLayer model no longer provides the guarantee that in each region, either all nodes receive a message or none do (the atomicity property).

3.2.2.5 Virtual Node State

The virtual node state includes the clock and discrete variables at the application layer. A virtual node's clock is synchronized together with its state.

In the absence of message collisions, the VNLayer still guarantees that a virtual node maintains its current state as long as the virtual node doesn't fail. When a virtual node does fail and restart, its state resets to the initial state.

However, with a lossy channel, the extended VNLayer model can't guarantee that a virtual node retains its current state even when the virtual node doesn't fail. Due to state inconsistencies among emulator nodes, the state on a virtual node might have occasional arbitrary changes when leadership of its region changes. This model requires that an application tolerate such state changes, which are called "acceptable" state changes.

3.2.2.6 Requirements on Application

In addition to the requirements in the basic VNLayer model, applications over the extended VNLayer must be able to tolerate message losses. In addition, due to the possibility of arbitrary state changes in the extended VNLayer, applications must be able to tolerate acceptable state changes, as described above.

3.3 The Implementation of the VNLayer

Our implementation of the VNLayer simulator is based on the implementation of VNE, a python based simulator developed by Mike Spindel in [36] for the basic link layer model and VNLayer model. With VNE, a Virtual Node based system is implemented as follows.

A MANET is tiled with square shaped geographical regions at fixed locations. In each region, a simple algorithm is used for leader election. In this algorithm, all physical nodes that are in the same region have equal opportunity to become leader. Whichever physical node that sends out its request for leadership first will be chosen as the leader of the region. When a physical node becomes the leader of a region, it sends out periodic heartbeat messages to claim its leadership. When missed heartbeat messages exceed a threshold, a non-leader node sends out a leader request message and starts a leader re-election.

In each region, physical nodes collectively emulate a virtual node. Among the emulator nodes in a region, the leader node processes incoming message and sends out response messages. The non-leader nodes process incoming messages the same way and the leader node does. However, the non-leader nodes buffer their response messages in a sending

queue. When a non-leader node receives a response message from the leader node, it checks its sending queue for an identical response message. If a match can be found, it removes the matched packets from its sending queue. This way, non-leader emulator nodes work as backup servers in a region.

Non-leader emulator nodes maintain replicated virtual node state for the leader node. One way to do this is that when a non-leader node can't find a match for an incoming message from the local leader, it considers it a sign of a state inconsistency and synchronizes its state with the leaders. In addition, each time an emulator node is set to a non-leader in a leader election, it synchronizes its state with the leader.

In addition, when a node moves into a region and there is already a leader in the region, the node becomes a non-leader and synchronizes its state with the leader's. Therefore, in the absence of message collisions, we can guarantee consistent state on a virtual node.

3.4 Implementation Choices

In order to support the extended link layer and extended VNLayer model, we implemented our own ns-2 bases simulator, VNSim, which uses the service provided by the extended link layer model. VNSim supports both the basic VNLayer model and the extended VNLayer model. During our simulations, we also investigated possible optimizations that can be taken when implementing the VNLayer. In this section, I discuss the possible optimizations as implementation choices.

3.4.1 Region Shapes and Node Sending and Receiving Capabilities

For simplicity, the simulators for the basic VNLayer model and the extended VNLayer model both have used square shaped regions and uniform node sending and receiving capabilities. However, other region shapes might utilize radio range more efficiently. For example, a network can be tiled by hexagonal regions.

3.4.2 Leader Election

Receiving a LeaderRequest message, the leader node rejects the request by a LeaderReply message. Non-leader nodes in the same region can be set to send LeaderReply messages too. Doing so can reduce the chance that an arriving node falsely claim itself as a leader. However, it will also increase the leader election message overhead.

On top of the basic leader election algorithm, to speed the leadership switching, a LeaderLeft message can be added to the leader election algorithm to speed leadership switching. Now, when a leader leaves a region, it sends out a LeaderLeft message to ask the non-leaders in the region to start a leader re-election immediately.

In addition, in a leader election, nodes that move more slowly and nodes that have had their state synchronized with the leader are more likely to become the new leader of a region.

In addition, to reduce the chance of multi-leadership, we can require that a node that just arrived at a region wait longer after sending out its leadership request before it can claim leadership.

Any combination of these choices can be used to improve the performance of either model. The more complex leader election algorithm reduces the number of state resets in the basic VNLayer model and arbitrary state changes in the extended VNLayer. However, these options are tested by my simulations only for the extended VNLayer model.

3.4.3 Number of Emulator Nodes

When a network is dense, it is not necessary to use every physical node to emulate virtual nodes. Doing so would increase the burden on every physical node and increase the number of state synchronizations. One implementation choice is to allow a physical node to decide dynamically on whether it should be an emulator node. This option would put control on the total number of physical nodes that are emulating the virtual node in each region and increase the efficiency of the VNLayer approach.

When not every physical node is an emulator node, the guarantees on virtual node still hold as long as there is at least one emulator node in a region.

3.4.4 State to Be Synchronized

In the implementation of the basic VNLayer model, the entire virtual node state is synchronized when a non-leading emulator node detects a state inconsistency. In order to reduce the state synchronization traffic overhead, an option is to synchronize only the critical part of the virtual node state. We define hard state and soft state as follows.

Hard state is the virtual node state that is critical to the correct operation of an application. An example of hard state is the address allocation information maintained by a DHCP server. Incorrectness on this information can lead to duplicate address allocations for extended period of time.

Soft state is the virtual node state that is non-critical to the correct operation of an application. An application will run correctly in spite of incorrect soft state. An example of soft state is the non-critical parts of a routing table maintained by a MANET router. The non-critical parts include route lifetime, expired routes, etc. Since MANET routers are meant to tolerate frequent routing failures due to node mobility, inconsistencies on routing tables is not critical to the correct operation of MANET routing.

With hard state and soft state defined, an implementation options is to let the VNLayer synchronize the hard state only. To do this, the programmer of the application layer software needs to determine what hard state is and what soft state is. With this option turned on, the VNLayer guarantees on hard state remain.

3.4.5 Subtypes of State Synchronizations

There are two types of state synchronizations. When a node enters a region and becomes non-leader, it synchronizes its state with the leader's. We call this type of state synchronization motion sync because it is triggered by node motion. When a non-leader detects state inconsistency, it synchronizes its state with the leader's. We call this type of state synchronization message sync because it is triggered by a message receive event.

In the implementation of VNLayer, the two types of state synchronizations can be enabled or disabled[19]. When either subtype of state synchronization is disabled, there will be more state inconsistencies on emulator nodes.

3.4.6 Control Over State Synchronization Frequency

In order to reduce the number of state synchronizations a virtual node, a minimum inter-state-synchronization interval can be set up on virtual nodes. However, doing so increases the chance that emulator nodes can have out of sync state.

3.4.7 Use of Overheard State Synchronization Messages

A non-leader can use any overheard state synchronization message from the leader to synchronize its state even if it hasn't detected any state inconsistency[20]. This option reduces the number of state synchronizations needed in a region.

3.4.8 State Consistency Checks

A non-leader emulator node can choose to check every message received from its leader to check for state consistencies. In order to reduce the number state synchronizations caused by state inconsistencies detected on non-critical part of the virtual node state, an implementation can choose to check messages that are more likely to have affected hard state only. Doing so would increase the chance an emulator node having its state out of sync.

---

[19] It is expected that motion sync is more important than message sync because when an emulator node moves into a new region, it relies on a message sync to receive the entire copy of a virtual node state. It's also what guarantees consistent state when there are no message losses. Message sync is used when there are message losses to patch up the state.

[20] When a non-leader detects a state inconsistency, it drops all the messages in its sending buffer and synchronizes its state with the leader's. However, when a non-leader's state is synchronized before it detects the state inconsistency, some messages in its sending buffer might not be valid.

In addition to using incoming message to look for state inconsistencies, a hash of the virtual node state can be carried in each message sent by a virtual node. The state hash can be used by emulator nodes to look for state inconsistencies. This option would reduce the false positives in state inconsistency detection. However, it also increases the processing overhead on virtual nodes.

3.4.9 State Inferencing

This option allows non-leader emulator nodes to fix parts of its state by inferring the virtual node state from messages sent by the leader emulator node. Doing so can reduce the number of state synchronizations. However, this option breaks the abstraction because it requires an application on an emulator node to act differently based on its role (leader or non-leader) at the VNLayer.

3.4.10 Communication Rules

In the basic VNLayer model, a client process can only communicate with its local virtual node and virtual nodes can only communicate with neighbor virtual nodes. With the extended VNLayer model, an implementation has the option of keeping the rules above or allowing a client to receive messages from non-local virtual nodes and allowing a virtual node to communicate with any other virtual node that is in range. Doing so would reduce the number of forwarding hops needed for a transmission. However, allowing virtual nodes to communicate with any other virtual node breaks the atomic reception guarantee. State inconsistencies can happen even in the absence of collisions when non-neighboring virtual nodes communicate with each other through long links.

3.4.11 Powerful Emulators

Another implementation option allows an emulator node to act as the Server for a client process that resides on the same physical node, using the application state of the region. This implementation choice is called Powerful Emulator because it is like an emulator node is given the full power of a server. With this option, rather than being constrained to its own region, a client process on an emulator node can seek services from virtual nodes in any region. This option can be used when the efficiency of a protocol is critical to its performance. However, doing so breaks the abstraction because it requires an emulator node to act differently when processing messages from client processes on its host physical node and allows an emulator node to act alone.

3.4.12 Summary of Implementation Choices

There are two simulators for the VNLayer, VNE [36] simulates the basic VNLayer. VNSim simulates both the basic VNLayer and the extended VNLayer. However, it needs to be noted that VNSim simulates the basic VNLayer model with the extended link layer model. Table 3-1 summarizes the implementation choices investigated by the simulators for each model. The impacts of the choices are also listed.

Table 3-1: Implementation options investigated by simulations of the VNLayer approach

Features	**Options**	**Basic VNLayer Model**	**Extended VNLayer Model**	**Impacts**
Region shape	Square, hexagon	Square	Square	Using square regions and uniform radio ranges simplifies simulation
Node radio range, reception ability	Uniform, non-uniform	Uniform	Uniform	
Number of emulator nodes	All physical nodes, a selected subset	All physical nodes	A physical node can dynamically decide to be an emulator node or not	Reducing the number of emulator nodes reduces syncs and increases chance that regions can be empty.
Leader election	Taking node motion rate, node status, node state consistency into consideration.	Simple	LeaderLeft message, node status, node motion rate, node state consistency took into consideration	The more complex leader election algorithm reduces leader changes; reduces leader switching delay and reduces the number of state resets in the basic VNLayer model and arbitrary state changes in the extended VNLayer model.
Consistency Checking	Check all messages, check only critical messages, check state hash	Check all messages (For VNDHCP, messages with no impact on state excluded)	Check only critical messages	Checking only critical messages, syncing hard state only, doing motion syncs only and limiting the rate of state synchronization reduce syncs or sync traffic and increase state

State to be synchronized	Sync all state, sync hard state only	Sync all state	Sync hard state only	inconsistency
Synchronization subtypes	Motion Sync, Message Sync	Always sync.	Two sync subtypes can be turned on or off	
Limit rate of state synchronizations	Used, not used.	N/A	Rate limit used on state synchronization.	
On overheard sync messages	Ignore, use to sync local state	ignore	Use overheard state sync messages	Using overheard sync msgs reduces number of syncs.
Client communication rules	Only with local virtual node, can receive from any virtual node, can send to any virtual node	Only with local virtual node	Can choose between the first two options.	Non-atomic reception when long links are used. It increases syncs and state inconsistency.
Virtual node communication rules	Can only communicate with neighbors, can communicate with anyone	Communicate with neighbors only	Can choose between the two options	
Powerful Emulator	Emulator node can act as server for client processes on the same physical node	Not used	Investigated	Making protocols more efficient but breaks the abstraction.
State inferencing	Used, not used	N/A	Can be turned on or off	Using state inferencing reduces syncs but breaks abstraction

CHAPTER 4. Virtual Node Layer Implementation

In this chapter, I first give a review on a python based VNLayer simulator we used at the early stage of the research. Then, I present in detail the implementation of our ns-2 based simulator VNSim. Because VNSim is built over ns-2, it uses the extended link layer model introduced in the previous chapter, which considers packet losses. VNSim can be used to simulate both the basic VNLayer model and the extended VNLayer model.

4.1 Virtual Node Emulator

Virtual Node Emulator (VNE) [38] developed by M. Spindel, is a light weight VNLayer simulator. VNE supports the basic link layer model and the basic VNLayer model as introduced in the last chapter. VNE includes a Mobile Node (MN) layer, a Virtual Node Emulator (VNE) layer and an Application layer. At the bottom, the MN layer simulates a simplified wireless link layer. It supports functions such as node creation, node motion and packet transmissions. The VNE layer simulates the VNLayer. It keeps track of a node's current location, sets up a node's region id, does leader elections, buffers packets for the non-leader nodes and synchronizes a non-leader's state with the leader's state. On top of VNE, the Application layer supports the application servers and clients.

VNE works well as a tool for proof of concept studies. The coding for applications in VNE is easy. In our early work on VNLayer based address allocations, simulations using VNE provided quick results and useful insights. However, VNE has a number of limitations. First, VNE's link layer doesn't model the packet loss caused by message

collisions and congestions. This limits the validity of the simulation results. Second, Python runs slowly. This makes VNE unsuitable for large scale simulations. Third, VNE is time based. Periodic checking on flags, node locations, etc. is used to drive the simulation in VNE. This makes the simulation time of VNE scale badly with increasing network size. In addition, the fixed checking periods can affect latency related simulation results. For example, if the checking period for incoming messages is set to 0.1 second, the response time for a message observed by the simulator can appear to be 0.2 second even if the actual latency is much shorter.

4.2 Virtual Node Simulator (VNSim)

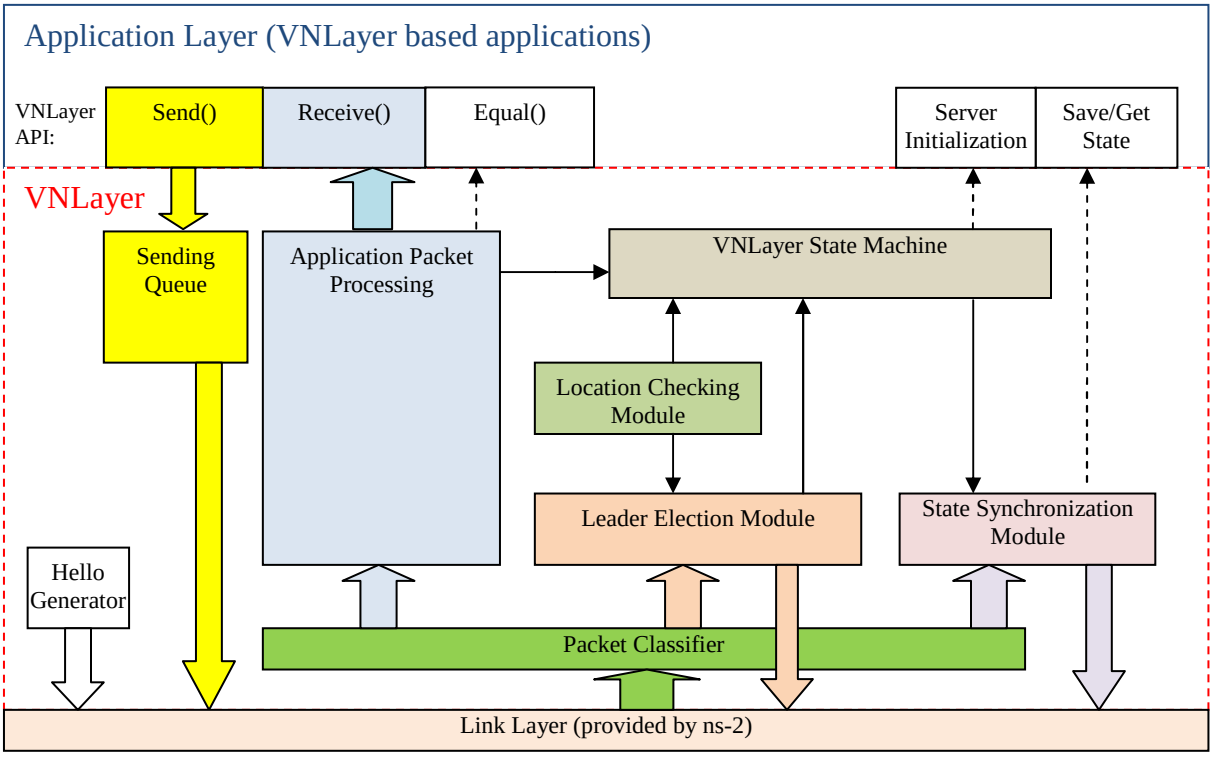

Figure 4-1 The Architecture of VNSim on a VNLayer equipped physical node

To deal with the problems with VNE, we created a discrete event-based VNLayer simulator, VNSim, on top of Network Simulator ns-2[39]. As shown in Figure 4-1, VNSim is built over the link layer provided by ns-2. It takes advantages of a mature simulator of the 802.11 link layer model, which is the same as the extended link layer model introduced in Chapter 3. VNSim can be used to simulate both the basic VNLayer model and the extended VNLayer model. However, it doesn't support the basic link layer model, which is an unrealistic model.

Programs for VNLayer based applications are developed at the application layer, which implements the following major interface functions required by the VNLayer.

- receive(): a function that the application layer uses to handle messages passed up from the VNLayer.
- send(): a function that the application layer uses to push messages down to the VNLayer.
- equal(): a function used by a non-leader node to check incoming messages from the local leader with the messages in its sending queue.
- save/get state: functions used by the VNLayer to retrieve or write to the Application Layer State.
- server initialization: a function used by a virtual node to initialize its state when it is restarted.

Except the send() function, all these functions are triggered by function modules at the VNLayer. The send() is initiated by the applications.

Figure 4-1 also shows the core function modules in the VNLayer.

- Application Packet Processing Module: a function module that handles application packets21 received by the VNLayer from the link layer and passes them to the application layer. (To be explained in section 0)
- Hello Generator: A function module that is used by a node to inform its neighbors about its presence. (To be explained in section 4.5.2)
- Location Checking Module: a function module that checks a physical node's geographical location and determines the region a node is in. (To be explained in section 4.6)
- Leader Election Module: a function module that determines and maintains a node's leader status by communicating with physical nodes in the same region. (To be explained in section 4.7)
- Packet Classifier: a function module that checks and passes incoming messages to appropriate function modules. It also keeps track of the activeness of neighbor nodes and neighbor regions. (To be explained in section 04.5)
- Sending Queue: A buffer at the VNLayer for packets sent down from the application layer. The VNLayer controls when the packets are sent to the link layer. (To be explained in section 4.9 0)

[21] Packets created by the application layer.

- VNLayer State Machine: the state machine that support state replication at the VNLayer. (To be explained in section 4.8)

In section 4.3 and section 4.4 , I introduce the VNLayer packet header and the state maintained by the VNLayer. Most of the terms used in this chapter are also defined in these two sections. Then, I introduce how each core function module works and how they interact with each other and with the application layer.

4.3 VNLayer Packet Header

The VNLayer inserts a 20 byte VNLayer header to every packet it relays to the link layer. The VNLayer header contains the following fields.

- Type (1 byte): The packet type.
- Subtype (1 byte): The packet subtype.
- Region ID (2 bytes): the sender region of the packet
- Source (4 bytes): the address of the sending physical node
- Destination (4 bytes): the address of destination physical node
- Send_time (4 bytes): the sending time of a packet
- Hash (4 bytes): a hash of the virtual node emulator's application state at the moment the packet is sent.

There are four types of packets that the VNLayer has to handle; Application messages, Leader Election messages, State Synchronization messages and Hello messages.

Application messages are the messages sent and received by the VNLayer based application. There are four subtypes of VNLayer application messages. Four subtypes provide finer grained control on what kind of messages should be used by the VNLayer to look for state inconsistencies.

- **Client messages**: application messages sent to the VNLayer by a client process. Local Client messages are the messages that a client process, which doesn't know anything about the VNLayer, sends to VNLayer on the same node. This kind of message is always considered an application message by the VNLayer. Therefore, when the VNLayer inserts the VNLayer packet header, the type of the message will also be set to "Application message" and the subtype of the message will be set to "Client message"
- **Server messages**: application messages originated from a virtual node.
- **Forwarded Server messages**: application messages forwarded by a virtual node.
- **Forwarded Client Messages**: A subset of client messages forwarded by a virtual node that neither use nor affect the application state. For example, when a client message is relayed by a virtual node to the neighbor that is closest to the destination region, the message forwarding doesn't use any application state, the message is a forwarded client message.

Leader Election messages are messages for leader elections and leadership maintenance. There are three subtypes of Leader Election messages.

- **LeaderRequest:** messages used to request for leadership
- **LeaderReply:** messages sent by a leader to decline a leader request

- **Heartbeat:** periodic messages used by a leader to claim its leadership.
- **LeaderLeft:** a new message type in the leader election algorithm for the extended VNLayer. It is used by a leader node to inform the non-leaders that it is leaving a region.

State Synchronization messages are messages for state synchronizations between leader nodes and non-leader nodes in the same region. There are two subtypes of State Synchronization messages.

- **SYN:** synchronization request messages sent by a non-leader.
- **SYN-ACK:** synchronization response messages sent by a leader.

Hello messages are generated by the VNLayer to help VNLayer based applications to maintain a list of immediate neighbors.

4.4 VNLayer State

The VNLayer operates on VNLayer state. The VNLayer state can be changed only by the VNLayer but is readable by the application layer[22].

The first part of the VNLayer state is **region ID**, which identifies the geographical region a VNLayer emulator node is in. A region ID is also used to identify the virtual node in a region.

[22] In VNSim, the VNLayer state is implemented as protected members of the base class for VNLayer based applications.

The second part of the VNLayer state is the **Leader Status**. It indicates the leadership status of a node in the region. The Leader Status can take one of the following seven values.

- INIT: The initial state before a node learns its region.
- Unknown: The node just enters a region and doesn't know about its role. A LeaderRequest message is scheduled but not sent out yet.
- Requested: The node has sent out a LeaderRequest, no response is received yet and the LeaderRequest timer hasn't timed out.
- Leader: The node is a leader of its current region.
- Non-leader: The node is a non-leader in its current region.
- Unstable: The node has missed at least one Heartbeat message from the Leader.

The third part of the state is **VN Status**. It is updated by the VNLayer State Machine and used by a physical node to determine its current role among the virtual node emulators of its region. The VN Status can take one value from the following values.

- UNKNOWN: The virtual node emulator hasn't learned its region id and does nothing. This is the initial state of a virtual node emulator.
- NEWNODE: The virtual node emulator just entered a region and hasn't determined its role.
- SERVER: The virtual node emulator plays the Server role. For the rest of this thesis, we refer to a node with this status as a *Server node*.

- BACKUP: The virtual node emulator plays the Backup Server role and it has its state synchronized with the Leader's. For the rest of this thesis, we refer to a node with this status as a *Backup Server* node.
- SYNC: The virtual node emulator just plays the non-leader role. The emulator either just entered a region or it detected a state inconsistency. It is synchronizing its state with the leader.
- PURECLIENT: The virtual node emulator, when not elected as a region leader, chooses to not work as a Backup Server. It acts as a *pure client* and doesn't process to any service request.

The fourth part of the VNLayer state is a **region activeness table** that keeps track of the activeness of regions from which messages can be heard by the virtual node. Each entry in the table maintains a region id, the address of the current leader of the region, the activeness of the region and a "lifetime".

The fifth part of the VNLayer state is a **neighbor list**. The list maintains the list of physical nodes from which messages have been heard recently. Each entry in the neighbor list maintains a physical node's node id, current region id and a "lifetime". An entry for a physical node in the list will be removed if no messages can be heard from it before its lifetime expires.

4.5 Packet Classifier

On an emulator node, the Packet Classifier is the first module that processes an incoming message from the link layer. It performs two tasks, Packet Classification and Neighbor/Region Activeness Maintenance.

4.5.1 Packet Classification

As shown in Figure 4-1, the Packet Classifier passes application messages to the application packet processing module; Leader Election messages to the Leader Election module and state synchronization messages to the State Synchronization Module. Hello messages are handled by the Packet Classifier and not passed to other function modules.

4.5.2 Neighbor/Region State Maintenance (NRSM)

When the Packet Classifier receives a Hello message or any other message, it uses it for its second functionality, Neighbor and Region Activeness Maintenance.

The Hello Generator Module in the VNLayer generates Hello messages. The interval between Hello messages can be adjusted by both the VNLayer and the Application Layer[23]. Each Hello Message carries the sending time, region id and node id of the physical node sending it. When it receives a Hello message, the NRSM on a node refreshes the lifetime of the corresponding entry in its neighbor list so the sending node stays as the node's immediate neighbor.

Since every VNLayer packet carries the sender's node id and region id, the NRSM uses every incoming VNLayer packet as a Hello message. To reduce the number of Hello

[23] The generation of Hello messages could be turned off if it is not needed.

messages, each time any message is sent by the VNLayer, the Hello Generator delays the next Hello message by a Hello interval.

Each time an entry in the neighbor list is refreshed by an incoming message, the VNLayer informs the application layer through an optional "Hello Handling" interface function. The application layer decides what to do with the event. In these simulations, the VNRIP application uses these events to update its table of immediate neighbors.

NRSM uses overheard messages generated by leader nodes to maintain its region activeness table. These messages include HeartBeat, LeaderReply, LeaderLeft, SYN-ACK and application messages. When NRSM hears a message from the leader of a region, it updates the leader id of the region and refreshes the timer associated with the region. If no leader message can be heard from a region before the timer expires, NRSM sets the leader id of the region to UNKNOWN and set the region to inactive.

There is an exception. When NRSM hears a LeaderLeft message from a region, it sets the leader id of the region to UNKNOWN and sets the region to inactive.

4.6 Location Checking Module

The Location Checking module in Figure 4-1 checks[24] a mobile node's current location and updates the node's region ID in the VNLayer state.

Location checking is the first thing an emulator node has to do when it starts running. An emulator node does nothing before it learns its region ID. When a node's Location

[24] The location checks can be done either periodically or done around the time a node is predicted to enter a different region.

Checking module finds out that the node has moved into a new region, it updates the Region ID and informs the Leader Election Module and the VNLayer State Machine about the region change.

4.7 Leader Election Module

As explained in section 3.3 the leader election algorithm works to ensure that the leader is the first node that requested leadership in a region without a current leader. Whenever a region change is detected, a node tries to become leader by sending a time stamped message requesting leadership (the LeaderRequest message). If it doesn't hear from a current leader (a LeaderReply message or a Heartbeat message) and it doesn't hear an earlier LeaderRequest message from another new node in the region, it becomes the region leader. One minor implementation choice we have here is whether or not to let the non-leaders to respond to LeaderRequest messages too. Doing so would increase the leader election traffic overhead while reducing the chance that a newly arrived physical node becomes a duplicate leader when the LeaderReply message is lost.

A number of timers are used to control how long a node waits to send its LeaderRequest message (leader request timer), to decide it is not going to hear from an earlier leader (request wait timer), to decide when a leader should send the next Heartbeat message (Heartbeat timer) and when a non-leader should start a leader election in the absence of Heartbeat message from its leader (leader timer).

Figure 4-3 illustrates the state machine that controls the leader election module. The input actions include: the expirations on the timers listed above, region changed event reported

by the location checking module, incoming messages such as the LeaderRequest, Heartbeat, LeaderReply and LeaderLeft. Figure 4-3 omits some reactions on incoming messages that don't result in state change. For example, when a leader node receives a LeaderRequest message, it sends back a LeaderReply message and stays as a leader.

The initial state of every mobile node is INIT, before it knows its region id. Every time a node learns it has entered a new region (the first time it learns its region is also treated as entering a new region), its state changes to UNKNOWN.

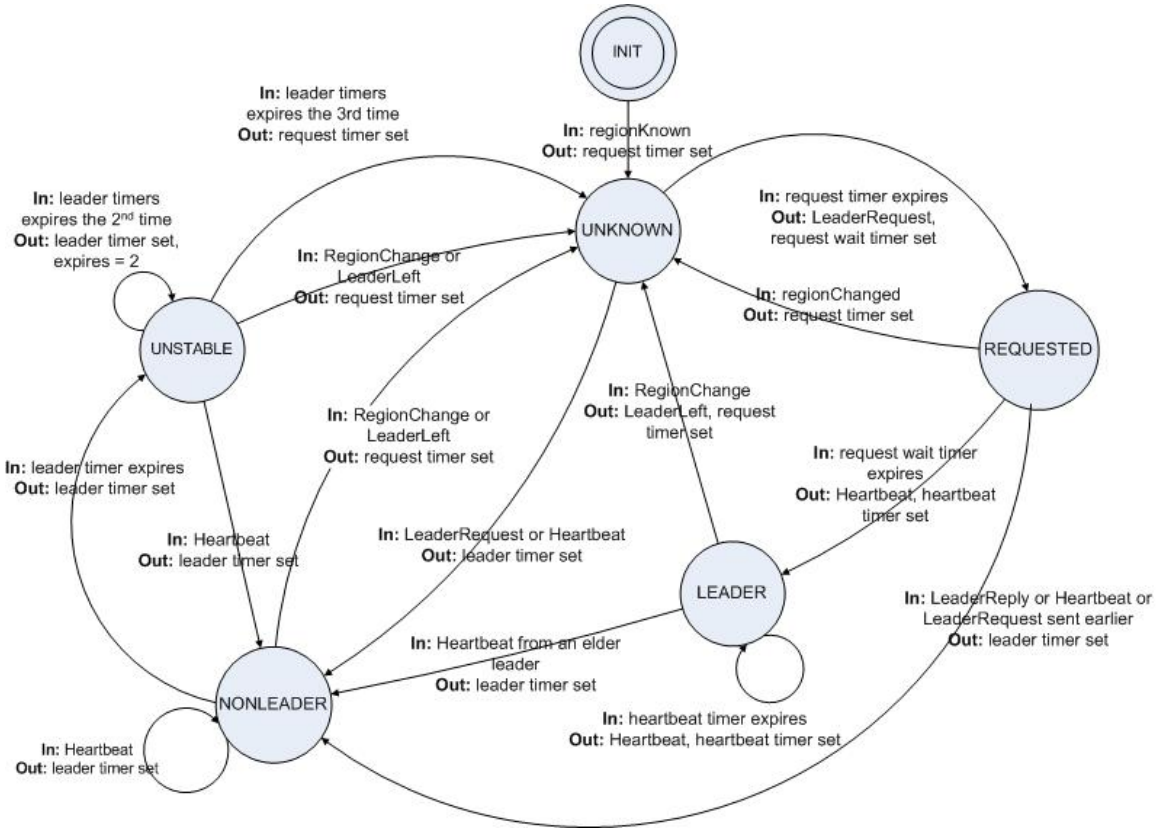

Figure 4-3 The Leader Election Module State Machine

The first thing a node has to do when it enters a new region is to determine its role in the region by sending out a LeaderRequest message and set up a request timer. If there is no

rejection from the region leader (by a LeaderReply) before the request timer expires, it changes its state to LEADER. Otherwise, it sets its state a NONLEADER. When a node becomes a leader, the Leader Election module sends a leader event to the VNLayer state machine[25].

In order to reduce the number of LeaderRequest messages when multiple nodes compete for leadership, each node schedules its LeaderRequest message with a random delay[26] using the leader request timer. Before the timer expires on a node, if a LeaderRequest message is heard from another node in the region, the node gives up its leader request and set its state as NONLEADER. Otherwise, the node sends out its LeaderRequest message; changes its state to REQUESTED and set the request wait timer.

A node starts sending periodic Heartbeat messages right away when it becomes a leader. The Heartbeat messages are tagged with the time when the sender becomes the leader of the region. Each time a Heartbeat message is sent by a leader node, the node sets its Heartbeat time with the Heartbeat interval.

When a node becomes a non-leader, it uses a leader timer (expiration time set to the Heartbeat interval plus the one hop transmission time) and a counter to tell when leader election needs to be done. A non-leader tolerates at most 2 Heartbeat misses. The first time a Heartbeat message is missing from the leader when the leader time expires, a non-leader node changes its state to UNSTABLE. When more than 2 HeartBeat messages are

[25] When a node becomes a non-leader, it may use a Coin Tosser Function (CTF) to decide whether the node shall become a backup server or a pure client node. Details on CTF will be given in section 4.7.5

[26] This delay is also used to give nodes that are moving slower and nodes that have their state synchronized with the leader precedence in leader election.

missing, a non-leader node assumes that its leader has gone and starts an attempt on leadership.

A good leader election algorithm should react to node mobility quickly; avoid duplicate leaders and excessive leadership changes. In section 3.4.2 , we discussed a few implementation choices on the Leader Election Module aimed at optimizing the leader election algorithm. As pointed out in section 3.4.2 , the implementation choices on the leader election algorithm don't affect the guarantees provided by the VNLayer abstraction. These optimizations will be explained in detailed in the following subsections.

4.7.1 Faster Leadership Switching

Because it takes 3 Heartbeat intervals for a region to decide that its leader is gone, Leadership switching is slow when waiting on missed HeartBeats. To solve this problem, we let a leader node send out a LeaderLeft message when it leaves its region. The LeaderLeft messages triggers a leader election right away in the leaders previous region.

The addition of this message greatly improves the delivery performance. It also lowers the requirement on the frequency of periodic Heartbeat messages. The Heartbeat interval is increased from 1 second to 5 seconds without affecting the performance of applications. The number of Heartbeat messages, the largest fraction of leader election messages, is reduced by 5 times.

In the case that the leader node crashes before it can send out a LeaderLeft message, the non-leader nodes can still detect, although much more slowly, the absence of the leader node in three Heartbeat intervals and start another leader election.

The simulations reported here assume no node failures. Frequent leader failures (as opposed to motion out of a region) would require a lower HeartBeat interval to keep the leadership switching delay down.

4.7.2 Reducing Duplicate Leaderships

Duplicate leaders can happen when messages are lost due to collisions. For example, when a node enters a region, it sends out a LeaderRequest message. If the LeaderRequest message couldn't be heard by the current leader of the region or the leader's LeaderReply message couldn't be heard by the requesting node, the requesting node would claim itself as a leader.

In a routing application, when duplicate leadership happens, a virtual node could forward the same data message multiple times toward the same next hop, causing amplified data traffic. Second, since the new self-claimed leaders don't have any route, data packets sent to them can trigger incorrect data packet drops and unnecessary route discoveries. This disrupts the data forwarding and increases the traffic overhead. Third, when there are multiple leaders in a region that have different application states, each incoming message could trigger a state synchronization in the region. This increases the state synchronization overhead. The increased traffic overhead can in turn cause even more duplicate leadership in the network. Therefore, duplicate leadership is very harmful.

In a lossy channel, it is impossible to prevent duplicate leadership. However, measures can be taken to reduce the chance that duplicate leadership happens and eliminate duplicate leadership quickly when it happens.

Since most duplicate leadership happen when a node enters a new region, we increased the delay before a newly arrived node can send its LeaderRequest message. Therefore, the newly arrived nodes have a greater chance of receiving a message from the current leader of the region and give up its attempt on leadership. In addition, after a newly arrived node sends out its LeaderRequest message, it also has to wait longer (than a non-leader node has to wait in a leader re-election) before it can claim leadership. This increase the chance a LeaderReply message can be heard from the current leader by the requesting node.

When a node whose state is LEADER receives a Heartbeat message from the same region, it checks when the sender became the leader. If the sender became the leader of the region earlier, the node gives up its leadership and sets its state to NONLEADER. If the sender became the leader of the region later, the node sends out a Heartbeat message right away to ask the other leader to give up its leadership.

With these optimizations, duplicate leadership rarely occurred in the simulations reported here.

4.7.3 Stabilizing Region Leaderships

Each time a region leader leaves a region, the services provided by a virtual node has to be paused for a period of time so that a new leader can be elected. In a leadership

switching, there is always a chance that the new leader doesn't have consistent state. As we are going to see later in this thesis, loops can form when this happens. Therefore, it is desirable that the number of region leadership switches be minimized.

Excessive leadership changes can happen when rapidly moving nodes become leaders, so that some penalty for rapid motion is useful in a leader election. In our implementation, nodes that can stay in a region longer are given an advantage in the competition for leadership. Here, we assume mobile nodes can find out their current motion rates and direction they are heading. Based on this information, a mobile node can find out how long it would take it to enter a different region. In a leader election, when mobile nodes decide their random delay before they can sends out their LeaderRequest messages, different random delays are used for nodes with the following 4 different levels of stability.

1. Static nodes: nodes that are not moving.
2. Stable nodes: nodes that can stay in the current region longer than 2 seconds.
3. Unstable nodes: nodes that can stay in the current region shorter than 2 seconds but longer than 0.1 seconds. (This is the waiting time before a requesting node can claim leadership, if there is no rejection.)
4. Very unstable nodes: nodes that are leaving the current region before it can claim their leadership if they send out LeaderRequest messages right away.

Table 4-1 shows the random backoff settings we used for nodes at different stability levels. This way, the slowest nodes send out their LeaderRequest message soonest. Therefore, they have the greatest chance of being elected as the leader of the regions they

are in. With this optimization used, the average number of leadership changes is reduced by 10%.

Table 4-1 Random backoff settings in leader election for nodes at different level of stabilities

Node Stability Level	Random Backoff Setting (picked uniformly random in the range)
1. Static Nodes	0~0.05 second
2. Stable Nodes	0.05~0.1 second
3. Unstable Nodes	0.1~0.15 second
4. Very Unstable nodes	0.2~0.25 second

4.7.4 Electing Better Leaders

When a node whose state is out of sync is elected as the leader, the virtual node in the region may operate incorrectly. In order to improve the performance of VNLayer based applications, nodes whose application state is out of sync (for example, newly arrived nodes) can be given lower precedence in leader elections.

The solution is, each time a state inconsistency is detected, a non-leader node sets its Sync Status to “out-of-sync”. Once its state is synchronized, the non-leader node clears the flag. During a leader election, if a node is flagged as “out of sync”, it delays itself longer before it can send out its LeaderRequest message. This way, if there are other Backup nodes in the region, the chance that the out of sync node takes over the region is lower.

4.7.5 Reducing the Number of Backup Servers

When a MANET is dense, there is no need for every region to have many Backup Servers. Having too many Backup Servers in a region can also lead to large number of state synchronizations[27]. As discussed in section 3.4.3 , an optimization can be done to control the number of Backup Servers in a region.

An optional Coin Tosser Function (CTF) is added to the leader election module to reduce the number of Backup Servers when the network is dense. Each time the Leader Election Module on a node finds out the node is to become a non-leader, it calls the CTF to decide whether the node will become a Backup Server or a Pure Client node. The CTF makes the decision using a preset threshold value and the current estimated number of nodes in the region. The preset threshold value is an integer between 0 and 1000. If the threshold is 1000, a non-leader node always sets itself to be a Backup Server. If the threshold is 0, a non-leader node always set itself as a Pure Client node. This actually means there is no Backup Server in any region. The CTF calculates the current number of physical nodes in the local region using the neighbor list maintained by the Neighbor and Region State Maintenance functionality in the Packet Classifier. Let this number be “size”, the following formula is used to calculate the probability p that the node will set itself as a Backup server. If the node density of the local region is 0, p is set to preset threshold/1000

$$if\ size > 0, p = \frac{preset\ threshold}{2000 \times size}$$

[27] Both MSG-SYNC and MOV-SYNCs

$$if\ size = 0, p = \frac{preset\ threshold}{1000}$$

The CTF makes a random Boolean decision with probability p and informs the VNLayer state machine about the decision on whether a non-leader node is to be a Pure Client node or a Backup Server.

4.8 The VNLayer State Machine

Figure 4-3 shows the state machine that controls the core VNLayer operations. The state transitions are triggered by input actions such as “regionChange” events generated by the Location Checking Module, “leader” event generated by the Leader Election Module (when a node becomes a region leader), “backup server” or “pureClient” event generated by Coin Tosser Module, state inconsistency detected by the Consistency Manager, synchronization waiting timer expiration and incoming messages such as the SYN message and SYN-ACK messages. The output action includes the sending of SYN and SYN-ACK messages.

The VNLayer of every mobile node starts with the initial state UNKNOWN before it finds out for the first time about its region id. Once its region id is known, a node enters the state NEWNODE. In addition, whenever a node enters a new region, the state of the node always transits into NEWNODE.

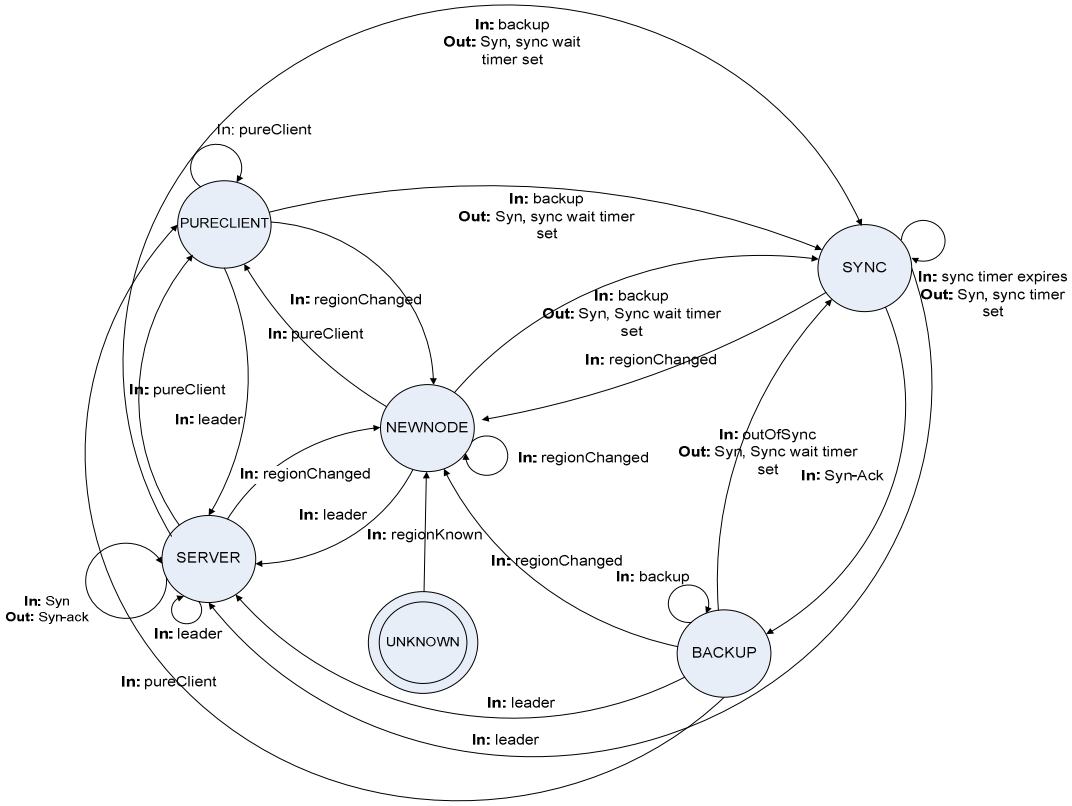

Figure 4-4 The VNLayer State Machine

Each time a node enters the state NEWNODE, the node resets the state at the application layer using the interface function “server initialization”. After this, the node waits for the events from the Leader Election module so that it can determine what role it will play in the new region. If the Leader Election Module decides that the node is the leader of the region, the node changes its state to SERVER. When the Leader Election Module decides that the node will be a non-leader, if the Coin Tosser Function decides that the node will

become a Backup Server, the node sets its state to SYNC. Otherwise, the node changes its state to PURECLIENT.

A Server node remains in the state SERVER as long as it stays in its region. In addition to responding to application messages, it also handles incoming requests for state synchronizations. However, when duplicate leadership happens and the node's Leader Election module decides to give up its leadership, the Coin Tosser Function will then decide whether the node shall act as a Pure Client node or a Backup Server.

When a node is to act as a Backup Server, the next thing it needs to do is to synchronize its state with the leader's state. Therefore, it asks the State Synchronization Module (SSM) to do a state synchronization. More details on the State Synchronization Module will be given in the next section. Once the node's state is synchronized, its state turns to BACKUP and it becomes an operational Backup Server of its region.

As mentioned before, a Backup Server node checks its state with the Server node's state (using the Consistency Manager). Once it determines that its state is out of sync, it changes its state to SYNC and asks the State Synchronization Module to do a state synchronization.

When the leader of a region leaves, a leader election will be done by the leader election module. This results in changing the role a node has to play in the region, even if the VNLayer status of a node is originally PURECLIENT.

4.9 Sending Queue

As discussed in chapter 3, the application layer on both Server nodes and Backup Server nodes put their response messages in a sending queue in the VNLayer. However, only the response messages on a Server node actually get sent.

If a node's VN Status is Server, the Sending Queue module is enabled to send. Packets in the queue are sent out one by one, with a small interval (10ms) between each sending. The small interval between each sending is added to reduce message collisions in the channel.

When a node's VN Status turns from Backup Server into Server, as a result of a leader election, the sending queue will also be enabled to send so that the packets remained there will be sent out. The packets stored in the sending queue are set to expire after a period of time (2 seconds). This prevents Backup Server nodes from sending out very old messages left in their sending queues, when they become Server nodes of its region.

As explained in section 3.2.1 , the extended link layer model allows physical nodes and virtual nodes to communicate using Directed Broadcast[28] whenever it is possible. When this implementation choice is used, if a packet to be sent is destined for a virtual node rather than a physical node, the current leader of the region hosting the virtual node will be looked up from the region activeness table. If the current leader is known, the packet is

[28] Direct Broadcast means when promiscuous mode is used by physical nodes, a packet can be broadcast to nearby physical nodes using a unicast destination address. The use of unicast destination address allows link layer acknowledgement and re-transmission.

sent out using a unicast destination address. Otherwise, the packet is sent out using a broadcast destination address.

In order to use Directed Broadcast, we need to consider packet transmissions in the following three cases:

Client process to virtual node transmission: Client processes use the region activeness table maintained by the Packet Classifier to find out the address of the leader of the virtual node. If the address is known, it sends the packet to the address of the leader node by unicast.

When the address of the leader node is unavailable, the client process sends its packet using a broadcast address. With the response packet sent out by leader node, the Packet Classifier will find out the address of the leader.

This implementation requires a client process to be able to keep track of the address of the leaders of regions. This breaks the abstraction and makes it harder to develop client code. One alternative implantation is to let virtual node emulator nodes to use two IP addresses, one unique address identifies itself, the other one identifies its region. When a client process needs to communicate with a virtual node, it uses the IP address for the region. The Server node in the region responds to unicast packets destined for the virtual node. However, doing so requires a more complicated link layer model.

Virtual node to virtual node transmission: When a virtual node needs to send a packet to another virtual node, from the region activeness table, it can find the address of the

current leader of the destination region. The packet is sent to the leader node of the destination region using unicast.

When the transmission of the packet fails due to leader changes, the sender virtual node replaces the address of the destination region to UNKNOWN. This transmission failure can be reported to the application layer together with the packet in question. The application layer then determines whether the packet shall be retransmitted or not.

When the address of a region's leader is unknown, a virtual node sends packets to the region by broadcast.

If a LeaderLeft message is missed by a virtual node, a node that sent the LeaderLeft may continue receiving packets for its old region's virtual node after it has left the region. In this case, the node sends another LeaderLeft message to inform the neighborhood again about the leadership change.

Virtual node to client process transmissions: When a virtual node needs to send a packet to a client process, the address of the client node is known to the virtual node. Using unicast destination address for this transmission is natural.

4.10 Application Packet Processing

In this section, we explain in detail how the Application Packet Processing Module works. This involves a number of sub-modules. Figure 4-5 shows a detailed illustration of the module.

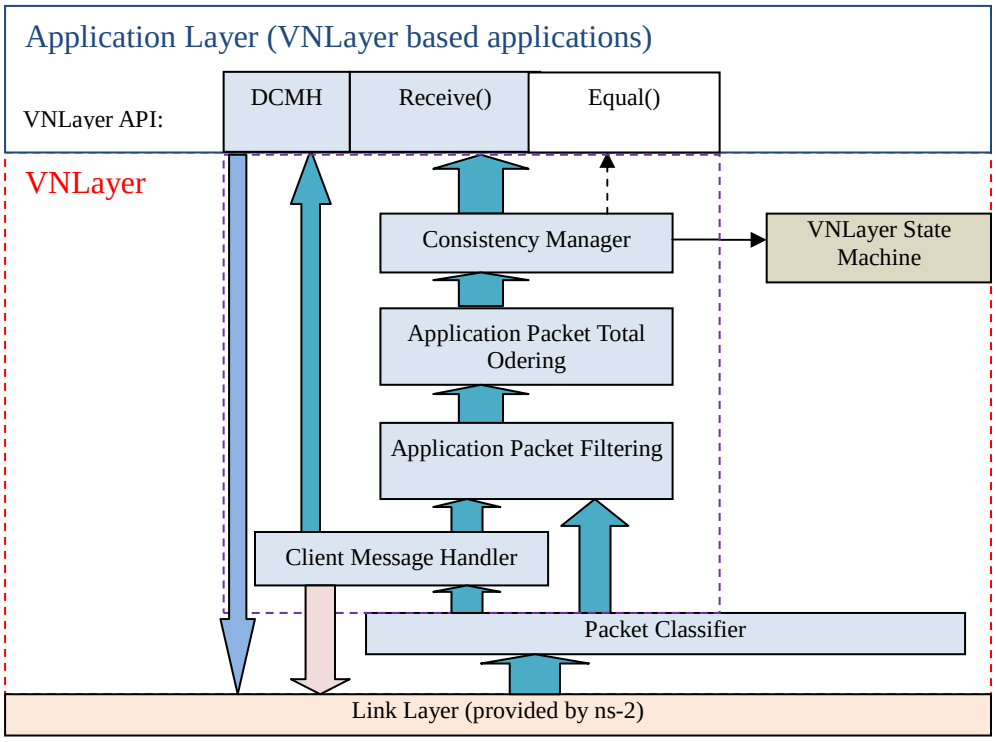

Figure 4-5 Details of the Application Packet Processing Module

4.10.1 Client Message Handler (CMH)

If a physical node is equipped with VNLayer capability, all local client messages are passed to the CMH by the Packet Classifier. Since other emulator nodes in the same region need to hear the message too, the CMH module makes a copy of the client message, and broadcasts it to the MANET. After this, the CMH module passes the client message to the next module, Application Packet Filtering (APF).

If a physical node doesn't support VNLayer, it still needs to implement the CMH module so that its local client message can have a VNLayer header, with type set as VNLayer

Application message and subtype set as Client Message. The CMH module then broadcasts the Client Message to the MANET.

4.10.1.1 Implementing the Powerful Emulator Option

This section presents the implementation of the Powerful Emulator option. As introduced in section 3.4.10 , an implementation choice called Powerful Emulator can be used to allow emulator nodes to act as independent servers and process local client messages directly. When this option is turned on, the CMH module does nothing except passing the local client messages to the server process at the Application Layer through an interface function called Direct Client Message Handling (DCMH). DCMH handles client messages alone.

Because the DCMH function at the application layer can communicate with other virtual nodes on its own, messages sent by the DCMH function always need to be sent directly to the link layer rather than controlled by the Sending Queue. Therefore, the DCMH function is given direct access to the link layer. Every packet sent out by the DCMH function is set as a Client Message, so that it won't trigger any state synchronization on other emulator nodes in the same region. Simulations on VNAODV with the Powerful Emulator option use this function.

4.10.2 Application Packet Filtering (APF)

When implementing the basic VNLayer model, a VNLayer application packet can only be passed on if it comes from the local region or from an immediate neighbor

region/virtual node. Hence, the APF module drops all messages coming from non-neighboring regions.

With our extended VNLayer model, virtual nodes are allowed to communicate with any virtual node it can reach. When implementing the extended VNLayer model, packets sent from any virtual node can be passed on by APF. Doing this reduces the reliability of the virtual node based network while increases the efficiency of it. In section 6.3.2 , we can see this implementation allows us to use fewer hops to forward data packets in a routing application.

In addition to filtering packets coming from remote regions, the APF also blocks application packets on emulator nodes whose state is out of sync. This is to ensure VNLayer emulator nodes whose state is out of sync don't process incoming application messages and generate bad response messages.

4.10.3 Application Packet Total Ordering (APTO)

This module buffers packets passed up from the APF module for a short period of time[29]. The packets buffered will be sorted using their sending times to increase the likelihood that different virtual node emulators receive packets in roughly the same order. Order matters because we want the state on each emulator node to change in the same sequence. This way, when the next module, Consistency Manager, uses the incoming messages to detect state inconsistencies, it makes fewer false positive detections of an out-of-sync state.

[29] We set this parameter to 10 milliseconds.

4.10.4 Consistency Manager (CM)

Consistency Manager is the last module an incoming packet has to go through before it is passed onto the application layer. The Consistency Manager cleans up the sending queue and detects state inconsistencies using incoming application packets, and passes packets to the application layer.

4.10.4.1 Sending Queue Clean Up and State Consistency Check

Since the Server node and Backup Server nodes in the same region are supposed to prepare the same sequence of responses message, a Backup Server should receive from the Server node a copy of every message in its sending buffer. On a Backup Server node, the consistency manager uses the messages it receives from the Server node to drop identical response messages from its sending queue and to detect state inconsistency when no identical response messages can be found.

On a Backup Server node, when the Consistency Manager receives a VNLayer application packet from the local region, it checks the subtype field in the VNLayer header.

Each Client message is sent by a single client process and it is not buffered in the sending queue. Therefore, this subtype of local application messages is ignored by the Consistency Manager. As defined earlier, Forwarded Client Messages have nothing to do with the application state. Therefore, we don’t do state consistency checks on Forwarded Client Messages either.

For Server Messages and Forwarded Server Messages, the Consistency Manager checks the sending queue to see if it holds a packet that is identical to the incoming packet. To tell whether two response packets are identical, the Consistency Manager calls the application layer user interface function equal(). In the VNLayer design in [1], two packets have to be exactly the same so that they can be considered the same. This way, there is no need to ask the application layer to check if two packets are the same. Since there are cases in which we might want to allow the response messages of from a Backup Server node and the Server node to be a little bit different. For example, the timestamp on each response message could be slightly different due to out of synch clocks on different node. However, triggering a state synchronization over such differences may not worth the cost. Therefore, we propose to allow the VNLayer to ask the application layer to check whether two packets are similar enough to be considered identical response to the same message.

If a match can be found, the Consistency Manager removes the matching packet from the sending queue. If no match can be found, the Consistency Manager asks the state synchronization module to do a state synchronization (This is a MSG-SYNC.).

When a VNLayer based application involves large number of Forwarded Server Messages (for example, routing application), in a lossy channel, the state synchronization overhead can be very heavy because Backup Server nodes in a region could miss many packets received by the Server node in the same region. Therefore, in such applications, to reduce the number of MSG-SYNCs, we only do state synchronization checks on Server Messages.

Another option provided by the Consistency Manager is to use the VNLayer header state hash field to check for state inconsistencies. To do this, each outgoing message shall carry a hash of the sending node's application state. Using an application interface function, getHash(), the CM can check if the local state is the same as the Server node's state. One disadvantage of the method is the computation cost involved in the state hashing. Also, many state synchronizations can be triggered by state inconsistencies on irrelevant parts of the state. To alleviate the impact these problems, the getHash() function in the application layer could be programmed to just do state Hashing a subset of the server's state that is deemed critical.

4.10.4.2 Passing Application Packets to the Application Layer

Depending on the VN Status of a node, the Consistency Manager decides whether an application packet needs to be passed to the application layer though the interface function receive(). If a node's VN Status is Server or Backup, the Consistency Manager passes the packet to the application layer. If a node's VN Status is PURECLIENT, the CM doesn't pass the packet to the application layer.

4.11 State Synchronization

Application layer state is maintained by the application layer. In order to do state synchronization, the VNLayer can read and change the application layer state through the API.

When the State Synchronization Module is asked[30] to do a state synchronization, it sends out a SYN message by unicast and sets a waiting timer for the response. When a Server node receives a SYN, it uses an interface function, getState(), to retrieve the local application state and creates a SYN-ACK message. When a Backup Server node receives a SYN-ACK message from the Server node, it uses another interface function saveState() to update its local state with the payload of the SYN-ACK message.

If the timer expires and no SYN-ACK message is received, the Backup Server tries again. The time interval between consecutive synchronization attempts increases linearly with each additional attempt.

In VNE, each non-leader checks all incoming messages from the local leader to look for state inconsistencies. Each packet missed by a non-leader node due to collision could trigger a state synchronization. Therefore, the state synchronization overhead increases quickly with heavier application traffic. In addition, in a state synchronization, a Server node has to send its state to the Backup Servers. The size of each SYN-ACK message can be large when the application layer state is large.

Because state synchronization messages can be large and consume a lot of bandwidth, we want to reduce the number of unnecessary state synchronizations. In the following subsections, I discuss in detail the optimizations/implementation choices we used to reduce the size of the SYN-ACK messages, to reduce the number of SYN and SYN-ACK

[30] by either the VNLayer State Machine or the Consistency Manager Module in the application message processing function.

messages and to reduce the number of state synchronizations that are triggered in the consistency manager.

4.11.1 State to be Synchronized

In section 3.4.4 , I introduced the optimization we used to reduce the state synchronization overhead by including only hard state in state synchronizations messages. The application layer decides what state is hard state and what state is soft state. The VNLayer creates SYN-ACK messages using the state passed down from the application layer.

If there is further need on reducing the size of the SYN-ACK messages, one solution is to synchronize only the portion of the state that is deemed critical. For example, in a routing application, the routes that are in use and the routes that are recently used can be deemed as critical state. In addition, the information maintained for a router in region, (for example, the local time) shall also be deemed critical.

4.11.2 Subtypes of State Synchronizations

As explained in section 3.4.5 , there are two types of state synchronizations in the VNLayer. The first type is motion sync (MOV-SYNC). MOV-SYNCs happen when a node enters a new region and becomes a Backup Server. The second type is message sync (MSG-SYNC). MSG-SYNCs happen when a Backup Server node detects state inconsistencies based incoming messages from the local Server node. Based on our observation, in routing applications, MOV-SYNCs are more important in keeping the state of the Backup Servers consistent with the Server's. This is easy to understand. When

a node just enters a new region, it has completely no idea about the current state of the virtual node emulated router of the region. The quickest way it can get its state updated is to do a state synchronization with the Server node. However, MSG-SYNCs are like doing patches on emulator states that has flaws. Routing application's requirement on state consistency is not as strict as in MANET address allocation. For example, a Backup Server may use a different viable next hop from the Server node's simply because it received RREP messages in a different sequence. When the Backup Server takes over the region, nothing bad will happen. A MSG-SYNC triggered by this kind of state inconsistency will be unnecessary. Therefore, in the VNLayer, we provide the option for the user to turn off MSG-SYNC completely when it is deemed necessary31.

4.11.3 Control Over State Synchronization Frequency

As introduced in section 3.4.6 , a limit can be set up for the maximum frequency a Server node can send out SYN-ACK messages. For example, a Server node can be set to send out at most one SYN-ACK message to its region per second.

4.11.4 Use Overheard SYN-ACK messages.

When a Backup Server's state is inconsistent with the Server's, it is likely that there are other Backup Servers in the region whose states are inconsistent too. Therefore, multiple non-leaders in the same region can send out their SYN messages together. Responding to each SYN message with a SYN-ACK is not necessary.

[31] With another option, MOV-SYNC can also be turned off. In the next chapter, simulations are done to test what happens when all state synchronizations are turned off.

As introduced in and section 3.4.7 , we can let Backup Servers use overheard SYN-ACK messages to update their state. This is done as follows. SYN-ACK messages sent out by the Server node as a broadcast message so that every Backup Server node can use it to update their state. A random backoff mechanism is used by the Backup Server nodes so that they delay their SYN messages with a random period of time before sending them. When a SYN message is heard from another node in the same region, a Backup Server whose SYN message hasn't been sent out cancels the SYN message.

4.11.5 State Consistency Checks

In VNE, a non-leader uses every incoming message from the local leader node to check for state inconsistencies. In a routing application, when the data traffic is heavy and the channel is congested, the Server node and Backup Server nodes in a region could receive very different sets of messages. Therefore, many MSG-SYNCs could be triggered. In order to reduce the number of state synchronizations, in section 3.4.8 , I introduced the optimization in which only messages that affects hard state are checked for state inconsistencies. In this section, I discuss this optimization in greater detail.

We classified VNLayer application messages into four subtypes, Client Messages, Server Messages, Forwarded Client Messages and Forwarded Server messages. The reason why we do this is because different application messages reflect the application state in different ways. In general, Server Message are the most connected to the application state because they are usually generated based on the application state. Client Messages have nothing to do with the application state because they are generated by the client process. When the application state is used to forward Forwarded Server Messages and Forwarded

Client Messages and can be affected by the forwarding operations, these two types of messages are connected to the application state too.

CHAPTER 5. MANET Address Allocation over the VNLayer

In this chapter, I present a VNLayer based address allocation protocol for MANET, VNDHCP (Virtual Node Dynamic Host Control Protocol). VNDHCP is adapted from the standard DHCP [17] for a wireline network, which operates basically as follows. When a client in a network needs an IP address, it broadcasts a DISCOVER message to the network. When a DHCP server receives the DISCOVER message and it has available IP addresses, it sends out an OFFER message by broadcast. When the client receives the OFFER, it confirms to the server that it wants to use the IP address by sending back a REQUEST message. When the server receives the REQUEST message, it sends back an ACK message, informing the client that it can start using the IP address it asked for, for a period of time (a lease time). Before the lease time expires, the client sends another REQUEST message to renew the lease with the server. The server confirms the renewal request with another ACK. By renewing the lease for its address periodically with the server, a client can use an IP address assigned by the DHCP server indefinitely.

VNDHCP was implemented over VNSim using the extended link layer model[32] and the basic VNLayer model. In VNDHCP, address allocation servers are emulated by the virtual node in each region. The addresses owned by the MANET are partitioned into pools[33] for allocation to the virtual node servers. Inside each region, a client process can

[32] This is due to the use of VNSim, which uses the link layer service provided by ns-2.
[33] In our implementation, the address pool owned by each virtual node is of the same size.

ask for addresses from the local virtual node, as if it is a fixed DHCP [17] server in the region. The operations of VNDHCP are therefore the same as DHCP when both the client and server are in the same region. However, this isn't always the case because a client process can leave the region after getting its address allocated. To solve this problem, VNDHCP lets a client node talk to the virtual node that originally supplied its address in order to renew its lease on the address. This inter-region communication is supported by a simple geographical routing algorithm.

As discussed in section 2.1 , a MANET address allocation server should allocate addresses to clients in such a way that each client can get an address until all addresses have been allocated; that no two clients have the same address; and an address can eventually be recovered for allocation to a second client if the client using it fails or leaves the network. It should be able to achieve this in the presence of message losses. In addition, since the address allocation servers in VNDHCP are emulated by virtual nodes, they must work correctly under the failure modes of the VNLayer, which include virtual node failures and resets.

The following sections explain how VNDHCP operates inside a single region and how a client node can get its address renewed or allocated from a remote virtual node. An important VNLayer implementation choice that reduces the state synchronization overhead is also explained. Section 5.5 discusses the advantages and disadvantages of VNDHCP, compared with other MANET address allocation protocols.

5.1 Address Allocation and Renewal inside a Single Region

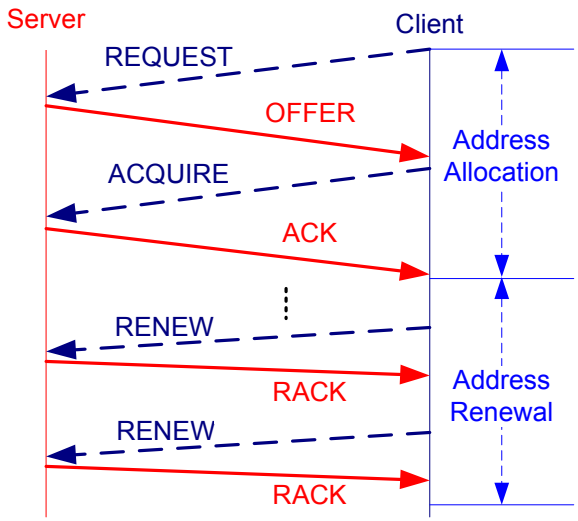

Figure 5-1 Address Allocation and Address Renewal in VNDHCP

Figure 5-1 illustrates the basic operations of the VNDHCP protocol, from a client's point of view[34]. In a region, a client process that needs an address broadcasts a REQUEST message to its region. Each REQUEST message is uniquely identified by the sender's id and a sequence number generated by the client process. When a REQUEST message is received by a virtual node, it responds with an available address if it has one, using an OFFER message. Receiving the OFFER message, the client process confirms that it wants to use the address by sending back an ACQUIRE message[35]. Upon receiving an ACQUIRE message from the client process to which it offered the address, the virtual node sends an ACK message to the client process. The ACK message carries the amount of time the address can be used (the ***lease time***) by the client process. Receiving the ACK

[34] The protocol is the same for address allocation across region borders. However, the messages would be relayed by the local virtual node to neighboring regions. However, from the point of view of a client, the interaction it has with a virtual server is the same as shown in the figure.
[35] The reason why this confirmation step is needed is because a client process can receive multiple OFFERs, as we'll see soon.

message, the client process can start using the address for a ***lease time***. These four steps conclude the address allocation stage of VNDHCP. If an offered address isn't allocated with an ACK, it remains available.

To prevent duplicate address assignment and to allow reallocation of unused addresses, each address in the address pool of a virtual node is associated with a flag indicating the allocation status of the address. The flag can take one of three values ***free***, ***pending***[36] and ***assigned***. Each address is also associated with a value, ***lifetime***[37], which indicates when a ***pending*** *or* ***assigned*** address should be set back to ***free***. In addition, each address is also associated with a value, ***owner***, which records the id[38] of node that is currently using the address.

In order to keep using an address, a client process has to renew the lease before the lease on its address expires. The renewal comes from the virtual node that originally assigned the address. To renew an address lease, a client process sends out a RENEW message to the virtual node (indentified by its region id). When a virtual node receives a RENEW message, it checks its state to see if the address in question is indeed assigned to the sender of the RENEW message. If so, it extends the lifetime of the assigned address by another lease time and sends an RACK message back to the client process. Like the ACK message, the RACK message carries the amount time the client process can keep using the address. On receiving the RACK message, the client process refreshes the lease time

[36] An address is "pending" state when it is offered to a client process but yet allocated to a client process.
[37] It is set to a lease time when an address is allocated. It is set to a smaller value when an address is set to "pending".
[38] MAC address, for example.

on its address. These two steps conclude an address renewal. This RENEW-RACK procedure repeats between a client process and a virtual node every lease time.

If an address renewal fails because the RENEW message from the client to the server is lost in the channel and the lease expires, the virtual node will set the address allocated to the client process back to *free*.[39] On the client process, when the lease on the address it uses expires, it gives up the address and start a new address allocation procedure.

If an address renewal fails because the RACK message from the server to the client is lost in the channel, the client process will give up the address when its lease expires. On the virtual node, the address remains as *allocated* for another lease time. Then, the lifetime on the address also expires. The address will also be set back to a free address.

5.2 Address Allocations and Renewals across Region Borders

If a client process never leaves the region where it gets its address assigned, the address allocation and renewal procedures are almost the same as DHCP's address allocation and renewal procedure. However, in VNDHCP, a client process may leave the region from which it originally got its address. A virtual node that assigned an address for a client process can also crash when the virtual node's region becomes empty. Therefore, some alterations are necessary to VNDHCP.

There are three cases in which address allocations and renewals need to be done across region borders. First, a virtual node may run out of address (Section 5.2.1). Second, a

[39] There is no extra attempt on renewal allowed in VNDHCP.

client node may leave the region from which it got its address (Section 5.2.2). Third, all nodes may leave a region so that the virtual node is "down" for a period (Section 5.2.3).

5.2.1 Local Address Depletion

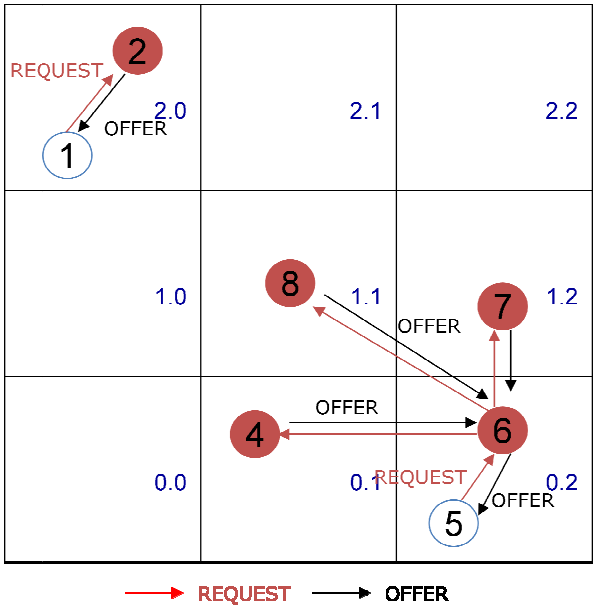

Figure 5-2 Address Allocation across Region Borders when a Region Runs Out of Addresses

In the first case, when a local virtual node runs out of free addresses for client processes in the same region, it forwards the REQUEST message it receives from the client process to its immediate neighbor regions. As shown in Figure 5-2, the REQUEST message from the client process on node 5 is forwarded by the virtual node in region 0.2 to its immediate neighbor regions. In the forwarded REQUEST message, the sequence of virtual nodes the message travels through is recorded. On receiving the forwarded REQUEST messages, virtual nodes in the neighbor regions send back OFFER messages when they have free addresses. These OFFER messages are forwarded back to the client

process by source routing, using the inverse of the forwarding path carried in the REQUEST message.

A client process may get multiple OFFERs relayed (by its local virtual node) to it from its immediate neighbor virtual nodes. The client process takes the first OFFER and sends back an ACQUIRE message toward the region that originated the OFFER message. The local virtual node forwards the ACQUIRE message by broadcast to its immediate neighbor regions. Like the forwarded REQUEST messages, the ACQUIRE message records the virtual nodes that have forwarded it. When the virtual node that made the offer receives the ACQUIRE message, it sends back an ACK message. The message is also forwarded back to the client process the inverse of the forwarding path carried in the ACQUIRE message.

5.2.2 Node Motion

In the second case, when a client process needs to renew its address lease, it might have already left the region from which it initially got its address. A client process must be able to renew its address lease with the remote virtual node that originally assigned it its address. Therefore, the RENEW message needs to be forwarded by virtual nodes toward a remote region and the RACK message also needs to be forwarded from a remote region back to the region where the client process is.

5.2.3 Virtual Node Reset

In the third case, in VNDHCP, when a virtual node is just booted up in an empty region, it sets all the addresses in its address pool as “assigned” for a whole lease time and set the

owner of all the addresses as “unknown”. This is because the newly booted virtual node doesn’t know whether there are client processes in other regions still using an address allocated by the region. Setting all the address to “assigned” therefore prevents duplicate address allocations.

During this one lease time waiting period, client processes in the newly booted region must rely on the virtual nodes in neighbor regions for address allocation. Address allocation and address renewals are done across region borders exactly as we have seen in the two cases above.

If there are indeed client processes using addresses originally assigned by a newly booted region, it would be ideal if they can keep using their addresses although the virtual node in the newly booted region has crashed before. Receiving a RENEW message for an address, if a virtual node finds out that the address is “assigned” and the owner of the address is “unknown”, the virtual node extends the lease for the client process and set the owner of the address to the original sender of the RENEW message. This way, the truly “assigned” addresses can be used by their owners without interruption. Otherwise, if a renewal request comes from a node that is not the owner of the address, as recorded by a virtual node, the virtual node ignores the RENEW message. This way, a client process that is using the address wrongly eventually gives up the address.

5.3 Application Layer Implementation Choices

When a virtual node is not able to assign addresses to a local client process due to address depletion or due to virtual node reset, as described above, it relays the REQUEST

message from the client process to neighbor virtual nodes. In our implementation, we allow the REQUEST messages to be relayed to the immediate neighbor regions only. This is because in case the local region runs out of addresses, a client process can have up to 8 neighbor regions that can help on assigning it an address. This is good enough when the address pool maintained by each region is large enough.

In addition, when an address renewal must be done across region borders, the RENEW and RACK messages need to be relayed by intermediate virtual nodes. The first method to forward the RENEW messages is to use flooding. Basically, a RENEW message from a client process is relayed by all the virtual nodes that hear it. The flooding of the RENEW message is controlled such that a virtual node forwards the same RENEW message only once. Flooding is simple and it is guaranteed to able deliver a RENEW message as long as there is a path and there is no message loss. However, flooding is also expensive in terms of traffic overhead and message collisions.

Because a client process knows the geographical location of the virtual node that gave it its address, the second method is to use a simple geographical based routing to forward the RENEW messages. Basically, a RENEW message is forwarded by each virtual node to its neighbor virtual node that is closest to the destination virtual node, no matter whether the virtual node's region is empty or not.

The forwarded RENEW message also records the virtual nodes it travels through, the RACK message sent back from the destination virtual node can be forwarded back to the client process using source routing.

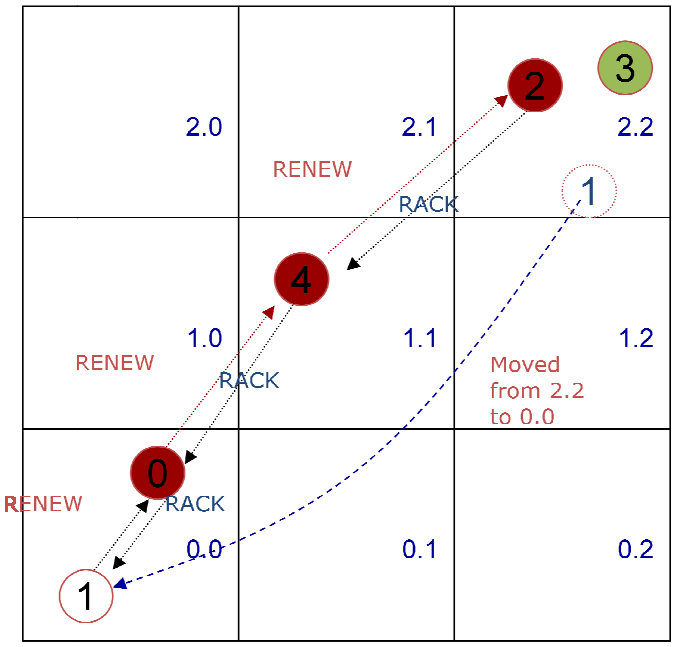

Figure 5-3 An address lease renewal across region borders (geographical based routing for RENEW messages)

As illustrated in Figure 5-3, the client process on node 1 initially got its address allocated in region 2.2. When the client process tries to renew its address lease, it has already moved into region 0.0. Its RENEW messages is first forwarded by virtual node in region 0.0 to region 1.1, the neighbor region that is closest to the destination region 2.2. The virtual node in region 1.1 then forwards the RENEW message to region 2.2. The RACK message sent back from the virtual node in region 2.2 is forwarded back along the route used by the RENEW message.

Thus a choice must be made between flooding or geographical routing for forwarding the RENEW messages. Flooding generates more messages than geographical routing. A flooded RENEW message, however, is more likely to reach the server region because the single path picked by geographical routing may fail due to node mobility or message loss.

A second choice is, how many hops a RENEW message is forwarded, either by flooding or geographical routing. Using a higher hop limit increases the chance a RENEW message reaches the intended virtual node and causes higher message overhead. Using a lower hop limit reduces the control traffic overhead and the chance the RENEW message can reach the intended virtual node.

5.4 Implementation Choices taken at the VNLayer

Based on the discussion so far, the VNDHCP protocol is implemented over the extended link layer model and the basic VNLayer model. However, performance can be improved by reducing synchronization checks based on message types. Messages are categorized into different subtypes by at the application layer. In VNDHCP, the messages types used in address allocation are REQUEST, OFFER, ACQUIRE and ACK messages. The other two types of packets are used for address renewal. When a message is originated from a client, it is marked as a Client Message or (when sent to neighbor virtual nodes by the local virtual node) a Forwarded Client Message. When a message is originated from a virtual node, it is a Server Message or a Forwarded Server message.

Based on the observation that all the forwarded address renewal messages, including the Forwarded Client messages and Forwarded Server message, are not using or affecting the VNDHCP state, we turned off state consistency checks in the VNLayer consistency manager on these two types of messages. This optimization reduced the number of state synchronizations without affecting either the state consistency among emulator nodes in a region or the address allocation performance.

5.5 Discussion: VNDHCP vs. Existing Address Allocation Solutions

Compared with existing MANET address allocation solutions introduced in Chapter 2, VNDHCP is a distributed address allocation server that has strong failover capability. The address allocation workload is shared by the virtual nodes covering separate geographical regions in a MANET. Because both the address allocation/renewal and inter-region message forwarding are handled by virtual nodes, the failure or movement of individual physical nodes won't affect the address allocation service as long as there are backup emulator nodes who can take over a virtual node after a leader moves to another region.

Although flooding is used for client messages such as REQUEST and ACQUIRE, the number of hops that these messages can be forwarded is limited to 2. This generates at most two times[40] the message overhead than the case that these messages are not forwarded at all. When RENEW messages need to be forwarded across region borders, if flooding is used, the forwarding of RENEW messages can cause heavy message overhead. However, when geographical based routing is used, the renewal message overhead is proportional to the hop distance between the client and the server region. In that case, our simulation results show that VNDHCP is scalable with increasing network sizes.

Although duplicate address allocations are very rare in VNDHCP, due to clock skew or state inconsistencies during region leadership changes, they can still happen. For example, due to message losses, a Backup Server doesn't know an address has been

[40] This is because the local virtual node only rebroadcast the messages to one extra hop by sending out one message when it can't do the address allocation.

allocated by the Server node. Before the Backup Server can synchronize its state with the Server node's state, it becomes the Server node of its region. It might allocate the address to another client process. Because any given address can only be allocated from a single virtual node and every client process needs to renew its lease for its address, address duplication can be detected within a lease time. When there are multiple nodes using the same address, the virtual node assigning the address acknowledges only one of the RENEW messages. The client processes that can't get an RACK message give up their addresses.

VNDHCP doesn't have the address leakage problem. When the lease on an ***assigned*** address is not renewed by RENEW messages from the client process, the status of the address always goes back to ***free***.

Due to the fixed setting of regions in the VNLayer implementation and the fixed address pool distribution among regions, network partitions and mergers won't cause any problems for the address allocation. When a network partition happens, a client process in a network partition using an address allocated from a separate network partition loses the address in one lease time. Before this happens, the virtual node in the other partition won't allocate the address to any other node. When two network partitions merge into one, the client processes in both partitions can keep using their current addresses. Therefore, the virtual node based protocol doesn't need any special handling for network partitions and mergers.

VNDHCP also has its disadvantages. First, the virtual node layer generates extra message overhead, although our simulation shows that this overhead is small compared to the channel bandwidth. Second, when the local virtual node and all the neighbor virtual nodes run out of addresses, a client may not be able to get an address even though there might still be addresses available in the system. In this case, the REQUEST messages may need to be flooded to more regions. Our simulation results show that when the address pool on each virtual node is large enough, the chance that a node can't get an address for extended period of is very low. Third, when a region becomes empty and the virtual node in the region is down, all the client processes who got their addresses from the region have to give up their addresses when the renewal fails. It would be better to back up the state of a region at the servers in neighboring regions.

The advantages over the existing solutions (for example, MANETConf, ZAL) described in CHAPTER 2 suggest the virtual node based address allocation protocol can be better suited for mobile ad hoc networks than all previous approaches.

CHAPTER 6. Reactive Routing over the VNLayer

Simulation results on VNDHCP show that the VNLayer approach is practical for simple protocols with little overhead. Can protocols involving continuous activity and generating significant overhead also be supported efficiently with the VNLayer approach? One rigorous test of this would be adaptation of a mature MANET routing protocol to the VNLayer approach. Can the adapted routing protocol deliver a packet, in the absence of message losses, with a bounded delay whenever there is a viable forwarding path? To answer this question, we created VNAODV, an adapted version of the popular reactive routing protocol, AODV, as introduced in section 2.2.2.2 .

VNAODV uses the core AODV algorithm[41]. However, there are a few major differences.

1) In VNAODV, the routing entities are virtual nodes running routing processes at the application layer. In the rest of this thesis, we call them ***vrouters***. Vrouters are identified by region ids, in contrast to physical nodes, which are identified by their IP addresses.

2) In addition, the routing table on each vrouter maintains routes for both physical nodes and other vrouters.

3) At the application layer, each VNLayer emulator node implements the functions required by the VNLayer API. These functions pass application layer messages to and

[41] The simulation code of VNAODV retains the core AODV algorithms, packet types (with additional fields) and the settings on most parameters. Therefore, the performance comparisons between AODV and VNAODV are fair.

from the link layer and allow the VNLayer to initialize, retrieve and synchronize the state at the application layer.

Now, in a MANET supported by VNAODV, the routing job is handled by vrouters at fixed locations in non-empty regions. The reduced number of routing entities, the relatively stable topology among the vrouters and the state replication capability provided by the physical nodes implementing vrouters would seem to give VNAODV an advantage over AODV. On the other hand, as we have pointed on in section 1.5 , the basic VNLayer model has its limitations. The small region setting leads to longer forwarding paths and the use of local broadcast in data delivery leads to heavier message losses. However, using the extended link layer model and the extended VNLayer model shortens the forwarding paths and greatly reduces the message loss rate in VNAODV. Simulation results presented in Section 8.2 show that VNAODV based on the extended link layer and VNLayer models performs better than AODV.

This chapter first presents the basic operations of VNAODV. Then, it describes how the implementation choices provided by the extended VNLayer model improve the performance of VNAODV. Finally, it explains a few optimizations at the application layer in VNAODV.

6.1 Basic Operations of VNAODV

The basic operations of VNAODV include three parts. Route Discovery refers to the operations taken by a vrouter to find a route for a data packet. Data Message Forwarding refers to the operations taken by vrouters to relay data packets to the destination. Route

Maintenance refers to the operations taken by vrouters to detect/fix link failures and recover lost data packets.

6.1.1 Route Discovery

As a reactive routing protocol, VNAODV doesn't do any route discovery or maintenance when there is no data traffic. When a data message (DMSG) is sent by a client process and received by a vrouter[42], the vrouter checks its routing table for a route. If a route is available, the DMSG is forwarded to the next hop vrouter identified by a region id. If there is no route, the DMSG is put in a buffer, namely, the RecvQueue. Then, the vrouter starts a route discovery by broadcasting a route request (RREQ) message to all the other vrouters.

An RREQ message carries the address of the destination node (destination address) and the last known route sequence number[43] for the destination (destination sequence number). It also carries the region id of the virtual node initiating the route discovery (the initiator address), a reverse route sequence number and a BCAST id. The last two fields are two non-decreasing integers generated by the initiator. The reverse route sequence number is used by other routers to update the reverse route for the initiator node. The BCAST id is used by other routers to avoid duplicate forwarding of RREQ messages.

[42] As we have discussed in chapter 3, when the node hosting the client process is a VNLayer emulator node, its VNLayer passes the data message up to the application layer and sends a copy of the data packet to the channel, by broadcast, so that all the other emulator nodes in the same region can hear it.
[43] The number is 0 initially, when there is no previously known route.

In each region, a vrouter responds to incoming RREQ messages the same way as standard AODV does, except that the router ids involved now are all vrouter ids, rather than the IP addresses of physical nodes. On receiving an RREQ message, a vrouter first uses the reverse route information from the RREQ message to update its route for the initiator of the route discovery, if necessary[44]. Then, the vrouter checks to see if it has a route for the destination with a sequence number that is no lower than the destination sequence number carried in the RREQ message. If so, it means the vrouter has a fresh route for the destination. The vrouter sends an RREP message back towards the initiator of the route discovery, using the reverse route just learned.

If the vrouter has no fresh route, it re-broadcasts the RREQ message, with its TTL field reduced by 1. The process goes on until every node in the network is reached or the TTLs on the RREQ messages reduce to zero.

Each route discovery is uniquely identified by the initiator address and the BCAST id carried in the RREQ messages. When a vrouter sends or forwards an RREQ message, the two fields above are saved in a queue for a period of time so that the vrouter doesn't forward the RREQ for the same route discovery again.

When the flooded RREQ message is received by the destination node or a vrouter that has a fresh route for the destination, an RREP message is generated and forwarded back toward the initiator of the route discovery. If it is the destination node that receives the RREQ message, the RREP message carries a destination sequence number newly

[44] If there are two or more alternative routes, the route with a greater destination sequence number or a route with the same the same destination sequence number but a smaller hop count is picked.

generated by the destination node (As in AODV, it is an even number that is no less than the destination sequence number in the RREQ message and greater than the sequence number maintained by the destination node for itself.).

When a vrouter receives an RREP message, it updates its route entry for the destination address, if the incoming route is fresher than its own or if the incoming route carries the same sequence number but is shorter than its own.

Unlike AODV, an extra field is added in the RREP message header to specify the next hop vrouter that is supposed to forward it, since the RREP messages are sent by broadcast at each hop. To take advantage of the broadcast RREP messages, every vrouter that can receive an RREP message uses the message to update its routing table. However, only the vrouter specified by an RREP message as its next hop forwards the RREP message,.

The AODV expanding ring search is also implemented in VNAODV. Using the TTL field carried in the messages, the expanding ring search puts a limit on how far the RREQ messages will be flooded in each route discovery attempt. Each time a route discovery is attempted by a vrouter, a wait time is set up in the route entry for the destination. The wait time is calculated based on the TTL used for the RREQ message and the estimated per hop delay. During the wait time, if more DMSGs are received by the vrouter, they are just buffered in the RecvQueue. When the wait timer expires and there is no response for the route discovery, another route discovery attempt will be triggered. This time, a greater TTL and longer wait time are used to expand the radius of the route discovery. As in AODV, at most 3 route discovery attempts are allowed for a DMSG.

6.1.2 Data Message Forwarding

The Data Message forwarding mechanism of VNAODV is also directly adapted from AODV. Vrouters forward each DMSG region by region toward the destination node. An extra field is added in the DMSG header to specify the next hop vrouter to relay a DMSG. Each time a vrouter forwards a DMSG, it extends the lifetime of the route entry used. This way, an active route doesn't expire unless a link failure is detected.

6.1.3 Route Maintenance

DMSGs are most frequently lost because a link fails (e.g., an empty region resulting from the leader node leaving the region cannot forward messages). Detecting link failures quickly is crucial to reducing DMSG delivery failures. AODV forwards DMSGs at each forwarding hop using unicast. This permits various mechanisms to be used to detect link failure quickly. For example, a failure can be reported when address resolution can't resolve the MAC address of the next hop router, the RTS/CTS mechanism can't reserve the channel with the next hop, or no ACK for the DMSG can be received and retransmission attempts also failed.

In VNAODV, detecting link failures is harder because the VNLayer requires that DMSGs be broadcast so that both Server and Backup Server nodes can hear them. None of the previously mentioned capabilities such as address resolution, RTS/CTS and data acknowledgement and retransmission can be used on broadcast messages. Therefore, the route maintenance mechanism used by VNAODV is different from AODV's.

The AODV specification [40] suggests two alternatives to link layer detection. One is periodic Hello messages to maintain a neighbor list on each router, using it to detect link failures. The other option is “passive acknowledgments”, i.e., if a vrouter overhears its next hop vrouter forwarding the message, it treats it as an acknowledgement.

Using Hello messages, the Hello interval determines how long it takes to detect a broken link. To respond to link failures quickly, the Hello interval has to be set to a small value. Doing so introduces a constant message overhead uncorrelated with the amount of data traffic. To avoid this, we use the passive acknowledgement mechanism[45] described next.

6.1.3.1 Passive Acknowledgement Mechanism

In AODV, each time a route entry is used to forward a DMSG, its lifetime is reset to 10 seconds[46]. In VNAODV, each time a route entry is used to forward a DMSG, we set its lifetime to 3 times the maximum estimated per hop Round Trip Time (RTT) and mark it as “unacked”. A “unacked” route entry is set back to “acked” by any data message overheard from the next hop vrouter, with its lifetime set back to 10 seconds. Therefore, the route entry expires quickly if there is no activity detected from the downstream region. When this happens, the link to the next hop router is considered unreliable and will be checked.

[45] The simulation does this at the application layer. However, it is also possible to move the implicit acknowledgement of local broadcast messages to the VNLayer

[46] Using the parameter ACTIVE_ROUTE_TIMOUT in AODV.

Thus a vrouter in VNAODV considers a DMSG that is forwarded by its next hop vrouter within three RTT's as a passive acknowledgement[47].

At the last forwarding hop, the destination node acknowledges a DMSG with an explicit DMSG acknowledgement so that the last hop vrouter can refresh the route used by the DMSG. The use of explicit acknowledgements at the last hop increases the message overhead proportionally to the data traffic.

6.1.3.2 Local Connectivity Check before Local Repair

The likelihood that a link failure reported by link layer detection reflects an actual link failure is much higher for AODV than VNAODV, because the vrouters are far more likely to miss a passive acknowledgment due to message collisions. Thus AODV assumes that the apparent link failure is real and either drops the DMSG being forwarded and sends a route error message (RERR) upstream, or else it starts a local route repair (it does the latter when the place the link breaks is closer to the destination than to the source of the DMSG being forwarded).

To avoid excessive route discoveries, VNAODV uses a different recovery mechanism from AODV. If a link failure is reported by link layer detection, VNAODV either sends an RERR message or starts a local repair. If the link failure was detected by passive DMSG acknowledgement, VNAODV first does a Local Connectivity Check (LCC) to find out whether the link is really broken. To do LCC, the route entry involved is marked as "route in repair" so that incoming DMSG messages are buffered in the RecvQueue,

[47] The DMSGs are forwarded by broadcast, making them audible to the previous hop vrouters.

waiting for the route to be verified. An RREQ message, with TTL set to 1, is broadcast to the neighborhood. Because LCC is not meant to find a fresher route, the RREQ message carries the current destination sequence number of the route involved. If the next hop vrouter is still working and the link is good, it responds to the message with an RREP message. On receiving this message, the router restores the route entry's status back to "up" and delivers the DMSGs buffered in the RecvQueue. If no such message can be heard within 2 RTTs, the vrouter considers the link broken and proceeds with one of the two AODV options.

In response to the one hop RREQ, other vrouters in the neighborhood can also provide alternative routes. This can reduce the service interruption when the next hop vrouter is indeed down. However, to prevent using alternative routes that are actually using the current next hop vrouter as a downstream router, upon receiving RREP messages from the neighborhood, a vrouter that is doing LCC only accepts alternative routes that are either fresher or no longer than the current one.

6.2 Preventing and Detecting Routing Loops

One important design objective of routing protocols is to provide loop free forwarding paths. In AODV, routing loops rarely[48] happen because the routes are tagged with sequence numbers to ensure freshness and each physical node creates and maintains its own routing tables independently.

[48] In our simulations, loops never happen in AODV because no AODV crashes. Therefore, the sequence number maintained by a router for a destination is never stale.

In VNAODV, routing loops are more likely. The root cause is that in VNAODV, routers are virtual nodes, each of which is emulated by a number of physical nodes. When the leadership in a virtual node switches and the new leader's state was inconsistent with the state of the old leader, the new leader could accept bad routes.. The following subsections explain three cases in which loops can happen in VNAODV and present solutions to them.

6.2.1 Restarted Regions

Figure 6-1 illustrates the first case in which routing loops can happen when a virtual node is booted up by a newly arrived mobile node. A forwarding path is present between node S and node D along region 0.1→1.1→2.1→3.1. When the last node (node B) in region 2.1 leaves, the virtual node in 2.1 is down. Then node C enters the region and boots up the region again. At this moment, node C doesn't know the latest sequence number used by the vrouter in region 2.1. Therefore, it may accept an RREP message from region 1.0[49] which uses region 1.1 as the next hop toward node D. When the vrouter in region 2.1 receives a DMSG from region 1.1, it forwards the DMSG to region 1.0. The vrouter in region 1.0 in turn forwards the DMSG back to region 1.1. A loop forms. The key problem here is that after the vrouter in region 2.1 went down, the vrouter in region 1.1 still forwards packets to it.

[49] When region 1.0 gets an RREQ message with a sequence number lower than its current sequence number, it sends out a RREP message if it has a route.

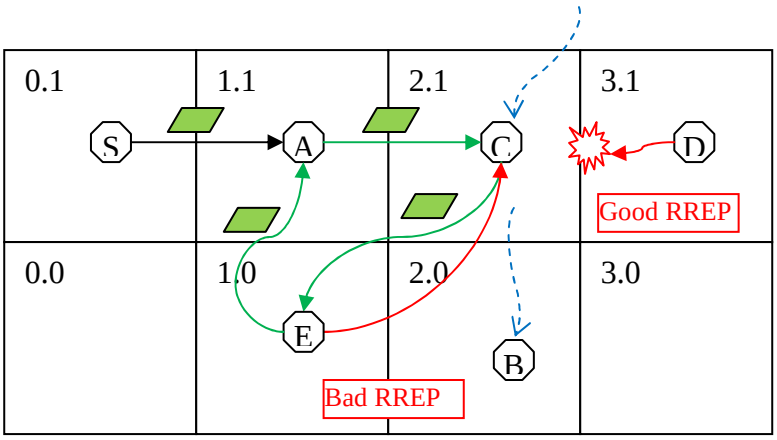

Figure 6-1 A routing loop in VNAODV when a region is booted

To solve this problem, when the vrouter for a region is restarted by a newly arrived node , the vrouter sends out a special RERR message right after it initializes its state. Receiving this special RERR message, neighbor vrouters tear down all local routes that are using the sender of the RERR message as the next hop vrouter. This way, no DMSGs will be delivered to the newly booted vrouter until it learns a valid route.

6.2.2 Out of Sync Nodes

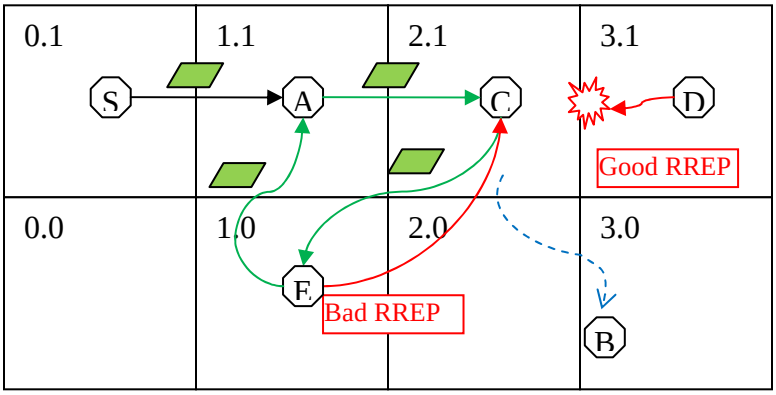

Figure 6-2 A routing loop in VNAODV when an out-of-sync node takes over a region

When a non-leader node loses track of the latest sequence number used by the vrouter in its region due to message losses, its state is out of sync. If the leader node leaves the region before the out-of-sync non-leader's state could be synchronized with the leader's state, and subsequently the non-leader node takes over the region, forwarding loops can form. Figure 6-2 shows an example. Node B was the leader of region 2.1 and node C's state was out of sync. When B leaves the region and C takes over, node C's sequence number for the route toward destination D could be smaller than the latest route sequence number used by node B. At some point later, during a route discovery, if an RREP message is received from Region 1.0, an out-of-date route could be accepted by the vrouter in region 2.1 because the RREP message has a greater sequence number. (A correct RREP message from the destination node D could fix the issue right away. However, sometimes the destination node D's RREP message is lost in the channel.) When the vrouter in region 1.1 sends a DMSG to region 2.1, the vrouter in region 2.1 forwards the DMSG to region 1.0. The vrouter in region 1.0 forwards the DMSGs back to region 1.1. A loop forms.

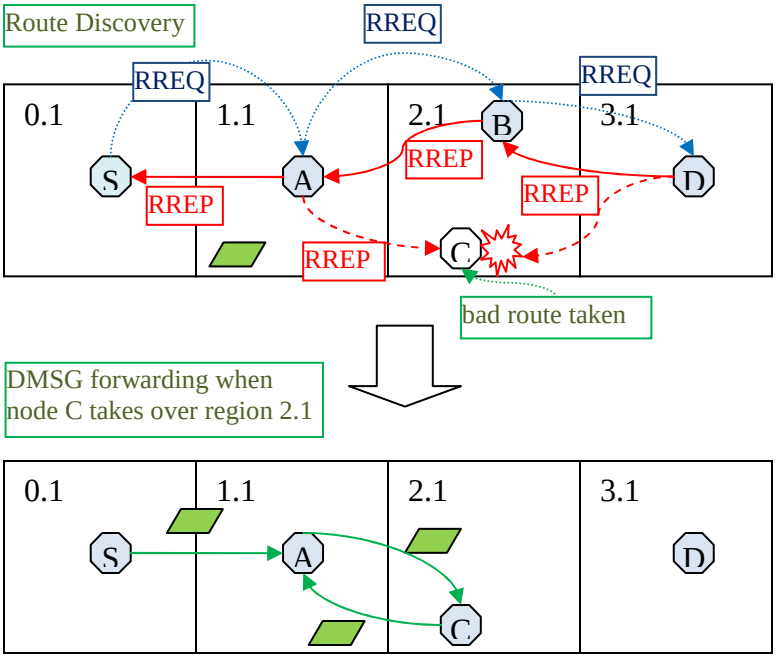

Figure 6-3 Routing loops in VNAODV when a Backup node learns a wrong route

Figure 6-3 shows another example. Here, node C used to be a Backup Server of region 2.1 while node B was the Server node. During a route discovery launched from the source node S for destination node D, the RREP message returned from the destination node D was received by the Server node B but was missed by node C due to collision. When the upstream Server node A forwards the RREP message it receives from Server node B, the RREP message is accepted by Backup Server node C as a viable route toward D through A. Before this problem could be fixed through state synchronization, Server node B leaves and node C takes over region 2.1. A loop forms when DMSGs are forwarded to region 2.1 from region 1.1.

The root cause of the two types of loops here is that a physical node whose state is out of sync could take over a region. As introduced in section 4.7.4 with the extended VNLayer model, the leader election can make sure that an emulator node with synchronized state is more likely to be chosen as region leader than one without synchronized state. However, this only makes loop formation less likely, it doesn't eliminate it. Therefore, we use additional loop prevention and loop detection methods in VNAODV.

In order to reduce the likelihood of loops, when a vrouter receives an RREQ message and is about to respond with an RREP message, it compares the next hop vrouter in its route with the sender of the RREQ message. If they are the same, it doesn't send the RREP message. This prevents 1 hop loops.

Loop detection is also used to detect and break loops quickly. First, when a vrouter learns that a DMSG is to be forwarded back to the vrouter where it comes from, the vrouter drops the DMSG and sends a RERR message upstream to tear down the loop. Second, when a DMSG's TTL field reaches 1 and still hasn't reached its destination, it indicates that something might be wrong with the route. When this happens, a vrouter drops the DMSG and sends an RERR message upstream to report the error.

As an alternative[50] to our last loop detection technique, a packet's expected hop count toward the destination could also be used to detect routing loops. The expected hopcount at each forwarding router should decrease monotonically as it gets closer and closer to the destination. Instead of decreasing the TTL field as it forwards a DMSG, a vrouter can fill

[50] This alternative is not tested in our simulations.

the TTL field of a DMSG with the hop count from the route it has for the destination. When a DMSG is received, if the local hopcount is greater than the TTL carried by the incoming DMSG, a possible loop is detected.

6.3 Taking Advantage of VNLayer optimizations

The purpose of the extended link layer model and extended VNLayer model is to improve the performance of VNLayer based applications. The last section described how an implementation choice (considering state synchronization status in leader election) at the VNLayer can help improve the performance of VNAODV. This section gives more examples of VNAODV benefits from the implementation choices in the extended models.

6.3.1 Selective State Synchronization and State Consistency Checks

6.3.1.1 Hard State vs. Soft State

As an implementation option, a virtual node can choose to keep only part of the application state (that is, the hard state) synchronized. This reduces the size of the state synchronization packets. This requires a design choice for the VNAODV implementer, who needs to determine which part of the application state is hard state and which part is soft state. On a vrouter, the most important state is the routing table, which contains routes that are up, down or under repair[51]. In addition, each route entry has a large number of fields and data structures (for example, a list of pre-cursor nodes is maintained for each route entry). The correctness of some state information only affects the

[51] The corresponding flags in AODV are RTF_UP, RTF_DOWN, RTF_IN_REPAIR.

performance, rather than the correctness of the routing application. For example, the last known hop count for a route that is down helps make the expanding ring search more effective in the next route discovery.

In the simulations reported here, for each route entry, the destination id, hop count, sequence number and next hop router id are considered hard state, because these values directly affect the correctness of route computations. On the other hand, since dead routes are only kept in the routing table for reference, they are considered soft state. In addition, the reverse route sequence number and the BCAST id are considered hard state because the physical node setting these fields for the vrouter in a region can change due to node mobility[52]. Incorrectness on these reverse route sequence numbers can lead to routing loops. Outdated BCAST id's can lead to router discovery failures.

6.3.1.2 State Consistency Checks

The extended VNLayer model provides for specifying which messages a Backup Server must use to detect state inconsistencies. This reduces number of state synchronizations. The simulations reported here specify state consistency checks on "Server Messages" only, i.e., those messages that are originated from a vrouter. These include the RREQ, RREP and RERR messages sent by a vrouter. These messages directly affect the routing table on vrouters. Therefore, they are considered more relevant to the application state than DMSGs, which only uses the routing tables on vrouters.

[52] These two pieces of information could get lost when a virtual node is down. Hence, when a vrouter is booted by an incoming node, the route discovery it launches can fail if the BCAST id it uses happens to match with a BCAST id recently used by the region. The Powerful Emulator option, discussed later, can alleviate this problem.

Using the two optimizations in this section provides a weaker guarantee on state consistency among emulator nodes. We can only guarantee each time a vrouter sends an RREQ, RREP or RERR message, if any Backup Server didn't prepare the same message in its sending queue, a state synchronization will be initiated to synchronize the hard state of all the Backup Servers.

6.3.2 Shortening Forwarding Paths

In the basic VNLayer Model, a virtual node only communicates with its immediate neighbor virtual nodes, even though it may be able to reach many additional virtual nodes. In our square region setting, a virtual node is therefore guaranteed to be able to reach every single virtual node emulator in its 8 immediate neighbor regions. While this setup ensures reliable communications between virtual nodes, it also requires that each pair of consecutive vrouters on a route created by a VNLayer based routing protocol must be immediate neighbors.

In addition, in the basic VNLayer Model, a client process can exchange messages only with its local virtual node for services. Therefore, at the first hop, a DMSG from a client process always has to be relayed by the local vrouter at the first hop. At the last forwarding hop, a DMSG has to be delivered to the destination node by the local vrouter in the destination node's region, even if a vrouter that is earlier in the route can reach the destination physical node.

These communication rules of the basic VNLayer model increase the length of the forwarding paths. Long forwarding paths lead to heavier DMSG forwarding traffic

overhead, longer forwarding delay, more frequent route discoveries and a higher chance for delivery failures due to broken links.

The extended VNLayer model relaxes the communication rules in the basic VNLayer model. It allows any pair of virtual nodes to communicate with each other and allows a virtual node to send to packets to any client process it can reach. This makes the use of longer links between vrouters and client processes possible In this section, we explain how this can be used to improve the performance of VNAODV.

6.3.2.1 Direct Receipt (DR)

When a vrouter sends a DMSG addressed to a destination that is in an immediate neighbor region, the destination node can receive the packet without the help of its local virtual node. The extended VNLayer model provides an implementation choice that allows a client process to receive messages from any virtual node. This option is called Direct Receipt (DR). A client process using DR in VNAODV can receive a message sent either by its local vrouter or by a vrouter in one of its neighbor regions. This way, a vrouter can deliver a DMSG directly to its final destination even if the destination is in a neighbor region.

As illustrated in Figure 6-4, with DR, the vrouter in region 2.1 delivers a DMSG directly to its destination D. To make DR work, at the VNLayer, we don’t allow a virtual node to receive a message sent to a client process in its region from another virtual node. Hence, the vrouter in region 3.1 doesn’t do anything with the DMSG. DR can reduce the length

of forwarding paths created by VNAODV by 1 by skipping the vrouter in the final destination's region.

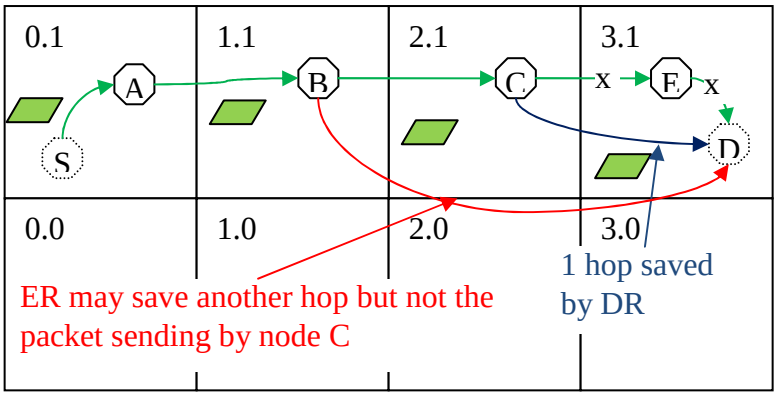

Figure 6-4 Optimizations in VNAODV that shorten forwarding paths

6.3.2.1.1 Direct Route Report from Destination (DRRD)

With the DR option, since vrouters don't relay DMSGs from other regions to local client processes anymore, their route entries for the destination node may expire even though there are still DMSGs forwarded to the destination node. When this happens, a vrouter can't respond to RREQ messages regarding a node in its region even though some neighbor vrouters may still know the route. This can slow down route discoveries.

To eliminate the need for a vrouter to respond to RREQ messages for nodes in its region, we allow the destination nodes themselves to respond to RREQ messages heard from immediate neighbor regions directly. In addition, vrouters update their routing tables using RREP messages heard from destination nodes in their immediate neighbor regions.

.

6.3.2.2 Early Receiving (ER)

When communication rules are relaxed so that clients can communicate with non-local vrouters, another optimization is to allow a client process to receive any DMSG for it is the destination, as soon as it can hear it. This means a DMSG can be received by the destination even before the vrouters on the forwarding path are done forwarding it. This optimization, called Early Receiving (ER), doesn't reduce the actual number of times a DMSG is relayed. However, it reduces the delivery latency. Figure 6-4 also shows the effect of ER.

ER allows a destination client process to continue receiving DMSGs for a while even after it leaves its original region. This gives a last hop vrouter more time to react before the destination client process leaves its radio range[53].

6.3.2.3 Long Links (LL)

The relaxation of communication rules allowing a virtual node to talk to any other virtual node within its radio range provides another VNAODV optimization called Long Links (LL).

With LL, the vrouters in VNAODV can work the same way as AODV routers. They can use any incoming RREP message they hear to update their routing tables and pick any vrouter within their reach as next hop vrouters toward the destination. The forwarding paths created by VNAODV can therefore be much shorter. Figure 6-5 shows an example in which when LL is used, a forwarding hop can be saved. Here, S is the source client

[53] The application layer optimization "route correction by destination nodes" in Section 6.4.3 takes advantage of this.

process. A, B, C and E are the vrouters in region 1.0, 1.1, 2.1 and 3.1, respectively. D is the destination client process. Because the vrouter in region 1.0 can reach the vrouter in region 2.1 directly, the vrouter in region 1.1 can be skipped. The forwarding path length is shortened from 4 to 3.

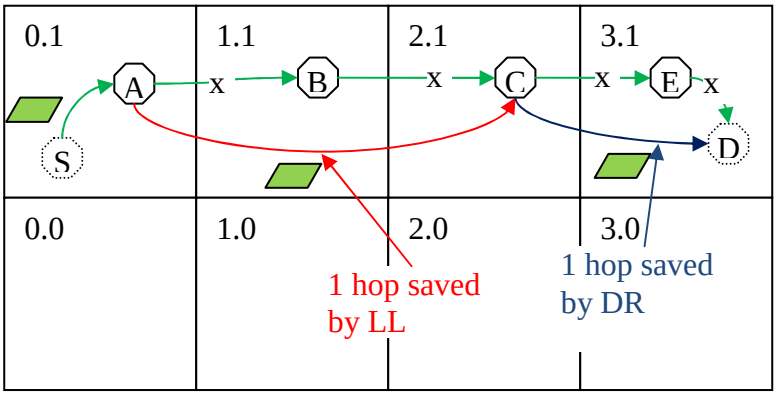

Figure 6-5 One forwarding hop saved by the Long Links option

However, this improvement in efficiency comes at the cost of degraded link stability. With LL, the guarantee of reliable transmission between two consecutive vrouters on a forwarding path no longer holds even in the absence of message losses. For example, if the next hop picked by a vrouter is not in an immediate neighbor region, it is possible that only a subset of emulator nodes in the next hop region can hear the messages sent by the vrouter. In the next hop region, when the node emulating the leader moves out of range or the leader switches to an emulator node that is out of range, the link between the two vrouters will break. When this happens, a local route repair or even a network route discovery has to be done to fix the route.

There is an interesting problem that arises with the LL option. This is illustrated in Figure 6-6. In the example, vrouter B can't reach client process D directly. In response to a

RREQ message, server node E sends an RREP message regarding D, which is in its own region. Vrouter B accepts this message and thinks it is just one hop away from D[54]. Later on, when it tries to forward DMSGs directly to node D, it can't. The reason this problem happens is that with LL, a good link to vrouter in the destination node's region may not be a good link to the destination node itself. To avoid this problem with the LL option, vrouters don't report routes to clients in their own regions. Instead, the clients use DRRD to respond directly to RREQ messages.

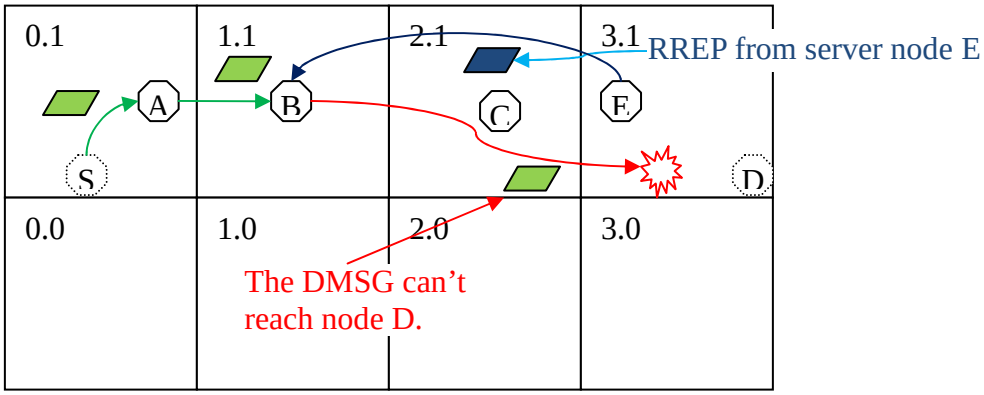

Figure 6-6 With LL option, a vrouter should not report routes for local destination nodes.

6.3.3 Directed Broadcast

To improve the reliability of data transmission in VNAODV, messages are sent by Directed Broadcast when the next hop is the final destination or when the address of the leader of the next hop region is known. Otherwise, local broadcast is used, together with passive acknowledgement and LCC for route maintenance.

[54] Here, we assume the RREP message originated from the destination node is lost.

When local broadcast has to be used to send a DMSG the sender can acquire the address of the next hop region's leader from the first passive acknowledgment (i.e., the forwarded DMSG). Because of this, Directed Broadcast and link layer detection can be used to send most DMSGs and to detect most broken links passive DMSG acknowledgement and LCC are rarely used at intermediate forwarding hops. Moreover, explicit DMSG acknowledgement at the last hop is not needed at all. Therefore, the use of Directed Broadcast makes data transmissions more reliable and route maintenance more efficient.

6.3.4 Powerful Emulator Option

Both the basic and extended VNLayer models require a client process to get service from its local virtual node only. This may lead to an extra forwarding hop in DMSG forwarding. This is because no matter whether a client process resides on a Server node or not, a message sent from the client process has to be copied by the VNLayer to its region so that all the other emulator nodes can get a copy of it.

Section 3.4.11 described Powerful Emulator, an implementation choice that allows an emulator node to act alone as the server for a client process that it hosts. When this option is used by VNAODV, an emulator node of a vrouter hosting a client process can act as an independent router, using the VNAODV state maintained by the server process on the virtual node. When an emulator node receives a DMSG from a client process running on the same node, if the destination is in its own region, it delivers the DMSG directly to the destination client process; otherwise, it relays the DMSG to a vrouter in a different region. Therefore, a client process on an emulator node doesn't have to rely on the local leader for services. At the first hop, the emulator node doesn't copy the DMSG to other

emulator nodes in the same region because they don't need to process the DMSG. As shown in Figure 6-7, with the Powerful Emulator option used, in addition to hops saved by other options, another forwarding hop at the beginning of the forwarding path for a DMSG can be saved because the source physical node S (an emulator node) sends its DMSG directly to the next hop vrouter C. The forwarding path is reduced from 3 hops to 2 hops.

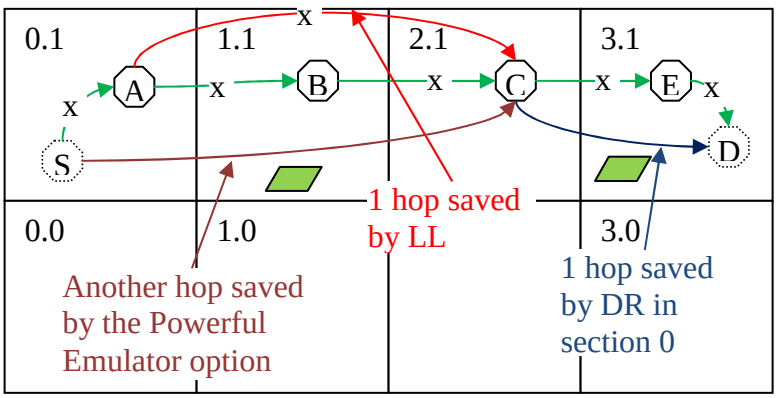

Figure 6-7 One forwarding hop saved by the Powerful Emulator Option in VNLayer

6.3.4.1 Basic Operations

As explained in section 4.10.1.1 , when the Powerful Emulator option is used, the VNLayer passes a locally generated Client message from directly to the application layer function DCMH. Therefore, to use the Powerful Emulator option, at the application layer, the VNAODV server implements the function DCMH[55], which does routing for client process on an emulator node[56] independently, using and updating the routing table it maintains.

[55] handleClientPacket() in our code.
[56] As opposed to a Pure Client node, this emulator node is the physical node that hosts the client process.

The DCMH function in VNAODV operates as follows. If DCMH learns that the routing table on the emulator node has a route for a locally generated DMSG, then it relays the DMSG directly to the next hop vrouter or the destination. Otherwise, the DMSG will be buffered in RecvQueue on the emulator node and route discoveries will be attempted. The DMSGs buffered in the RecvQueue by the Powerful Emulator Option are marked differently. They will be sent directly to the link layer, without using the sending queue in the VNLayer. If no route can be discovered, the DMSG will eventually be dropped.

The DCMH function processes routing messages and updates the routing table on an emulator node the same way as a regular VNAODV server process does, except that the messages sent out by the DCMH function go directly to the link layer.

If a client process is hosted by a pure client node, it can't take advantage of the Powerful Emulator option. As before, the client process just inserts the VNLayer header to DMSGs and broadcast them to its region.

Shortening the forwarding paths is not the only benefit of using the Powerful Emulator option in VNAODV. When Powerful Emulator option is not used, in a route discovery, the BCAST id and reverse route sequence number carried by the RREQ messages are set by the first hop vrouter, which is a virtual node. When a virtual node is taken over by an out of sync Backup node or booted by an incoming node, the first hop vrouter may lose track of the BCAST id and sequence number used by the region. When the Powerful Emulator option is used, in the route discoveries initiated by an emulator node for client processes on it, the BCAST id and reverse route sequence number carried by the RREQ

messages can be uniquely set up by the emulator node. Leadership switching in the region of the source node won't affect the correctness of the BCAST id and reverse route sequence number used by route discoveries launched by a "Powerful" emulator node for a client process on it.

6.3.4.2 Additional Considerations

There are a few additional considerations that must be taken into account with the Powerful Emulator option when it is used in VNAODV.

First, when an emulator node enters a different region, by our VNLayer implementation, the state on the node must be cleaned up and synchronized with the leader of the new region. Without the Powerful Emulator option, a client process entering a new region simply lets the vrouter in the new region handle its DMSGs. The vrouter in the new region usually has a route for the destination (through overheard RREP messages). However, when the Powerful Emulator option is used, an emulator node hosting a client process handles the routing for the client process. Each time such an emulator node enters a new region, it loses its routing table and has to wait a while[57] before it can determine its leader status and gets its routing table synchronized with the vrouter of the new region. During this period of time, if there are DMSGs to send for the client process, the emulator node has to do a network-wide route discovery. This increases the number of network-wide route discoveries.

[57] About 1 second.

To alleviate this problem, we let the DCMH function on an emulator node do a 1-hop route discovery each time it enters a new region while relaying DMSGs for a client process on it. The intuition behind this optimization is that when an emulator node relaying DMSGs enters a new region, it is very likely that a vrouter within its one hop neighborhood still has a viable route for the destination. This optimization greatly reduced the number of unnecessary network-wide route discoveries.

Second, with our original VNLayer implementation, each time an emulator node enters a new region, it should discard all the packets it has in its RecvQueue at the application layer because it no longer works for the old region. When the Powerful Emulator option is turned on, when an emulator node enters a new region, the DMSGs it buffers in its RecvQueue for a client process on it should not be discarded, because this emulator node is the only node who handles these packets. Therefore, when the Powerful Emulator option is used, we let an emulator node keep the DMSGs in its RecvQueue that are sent by a client process on it.

Third, when the Powerful Emulator option is used, the state maintained by the Server node in a region may interfere with the state maintained by a Backup Server node serving a client process on it independently. The reason is that such a Backup Server node can create routes that are not synchronized with the Server node in the region. For example, when a Backup Server node discovers a route for a client process on it, due to message losses, the Server node in the same region might still have an old route. When the Server node sends out a SYN-ACK message in response to a state synchronization request, the Backup Server will be forced to change its good route to the bad route the Server node

has. To solve this problem, we flag a route that is discovered by a Backup Server node for a client process on it a “client route”. As long as a “client route” is still up, messages heard from the Server node of the region can’t overwrite it.

In another case, when a Backup Server node serving a client process on it determines that a route is no longer good[58], it sets the route to “down” and starts a route discovery. However, if the Server node in the region thinks the route is up, messages from the Server node may force the Backup Server node to set the route back to “up”. To solve this problem, when a Backup Server set a “client route” to “down”, it increases its sequence number by 3. (The standard AODV increases the sequence number of a route by 1 when it is flagged as “down”.) In addition, we require that in order for a route learned from the Server node to turn a route on a Backup Server node from “down” to “up”, the route must have a sequence number that is no less than the local route’s sequence number minus 1. This way, only a fresh route from the Server node can restore a “client route” that is set to “down” by a Backup Server node.

As discussed before, the Powerful Emulator option provided by the VNLayer actually breaks the VNLayer abstraction because it requires an emulator node to act differently depending on whether a DMSG message comes from a client process on the node itself. It requires the application layer to implement another state machine handling local client messages and leads to tricky complications that have to be dealt with carefully at the application layer. The solutions to the complications further diversify the behavior of

[58] This can result from its own route maintenance.

emulator nodes in the same region. One might want to use this option only if doing so can greatly improve the efficiency of a VNLayer based application.

6.4 Optimizations at the Application Layer

The focus of this simulation study is to find out how to improve the performance of VNLayer based protocols using general solutions at the VNLayer. In addition to optimizations at the VNLayer, we also tried a number of application layer optimizations for VNAODV.

6.4.1 Local Recovery of DMSGs

In AODV, when a link failure is detected through link layer detection[59], the packet for which the link failed is re-buffered and can be re-transmitted once the route is repaired. In VNAODV, when local broadcast is used to deliver a DMSG[60], link layer detection can't be used. Instead, passive DMSG acknowledgement is used to detect broken links. Because of this, when a DMSG is sent by local broadcast, the sender can't recover it. This section describes how to add such a mechanism.

We use a Local Recovery (LR) mechanism to recover and retransmit DMSGs suspected of being lost when local broadcast is used to transmit a DMSG. With LR, the source client process tags each DMSG with a sequence number. Each time a DMSG is broadcast to a downstream vrouter[61], a copy of the DMSG is saved in a buffer called Backlog. When a passive DMSG acknowledgement is received, a vrouter checks its Backlog for

[59] AODV use unicast at the link layer to deliver DMSGs. A router keeps a copy of the packet that is just sent until it is acknowledged. Without an acknowledgement, the link is deemed broken.
[60] For example, when the address of next hop region's leader is unknown or promiscuous mode is not supported
[61] At the last hop, because unicast is used, LR is not needed when the next hop is the destination node.

DMSGs from the same session, using the source address, destination address and sequence number in each DMSG. The matching DMSG will be dropped from the Backlog. The DMSGs in the Backlog that have lower sequence numbers are the DMSGs not yet acknowledged. Retransmissions might be necessary for them.

The DMSG having a sequence number exactly 1 less than the sequence number of the passive DMSG acknowledgement will be resent. It is removed from the Backlog and re-buffered in the RecvQueue. The next time the router checks the RecvQueue, if the route for the DMSG's destination is "up", the re-buffered DMSG will be re-transmitted.

To avoid excessive re-transmissions, any other DMSG in the Backlog for the same session that has a lower sequence number than the sequence number of the passive DMSG acknowledgement is dropped.

In addition, when a link is suspected of being broken because of a timeout on DMSG acknowledgements, there will be a Local Connectivity Check. Before this is done, if there is any DMSG for the same destination in the Backlog, the one with the smallest sequence number is removed from the Backlog and re-buffered in RecvQueue. This DMSG, which must be unacknowledged, will be re-transmitted once the route is restored.

With subsequent passive DMSG acknowledgements, the DMSGs left in the Backlog will be either dropped or re-buffered. If a session terminates, DMSGs left in the Backlog will eventually be timed out and dropped.

Re-buffered DMSGs are marked differently so that they won't be put in the Backlog again when they are re-transmitted from the RecvQueue. This means a DMSG can be transmitted to the next hop at most twice. Also, to reduce routing traffic, re-buffered DMSGs in the RecvQueue won't trigger any route discoveries. If a re-buffered DMSG in the RecvQueue can't be forwarded, the DMSG will eventually be timed out and dropped.

Figure 6-8 shows an example of how Local Recovery works. In steps 1 and 2, packets 1 and 2 have been received successfully but passive acknowledgements have not been heard by vrouter A. Vrouter A sends out packet 3 in step 2. Therefore, at the end of step 2, vrouter A has 3 packets in its Backlog, packet 1, 2 and 3. In step 3, packet 4 is sent out but lost (e. g. due to collision). Vrouter A also received a passive acknowledgement for packet 3. It drops packet 1 and 3 from its Backlog and re-buffers packet 2. In step 4, packet 2 is retransmitted without being put into the Backlog and a new packet 5 is also transmitted and put into the Backlog. In step 5, acknowledgement for packet 5 is received. Vrouter drops packet 5 from its Backlog and re-buffers packet 4. In step 6, packet 4 is re-transmitted. The retransmission in step 4 is unnecessary and the one in step 6 is necessary.

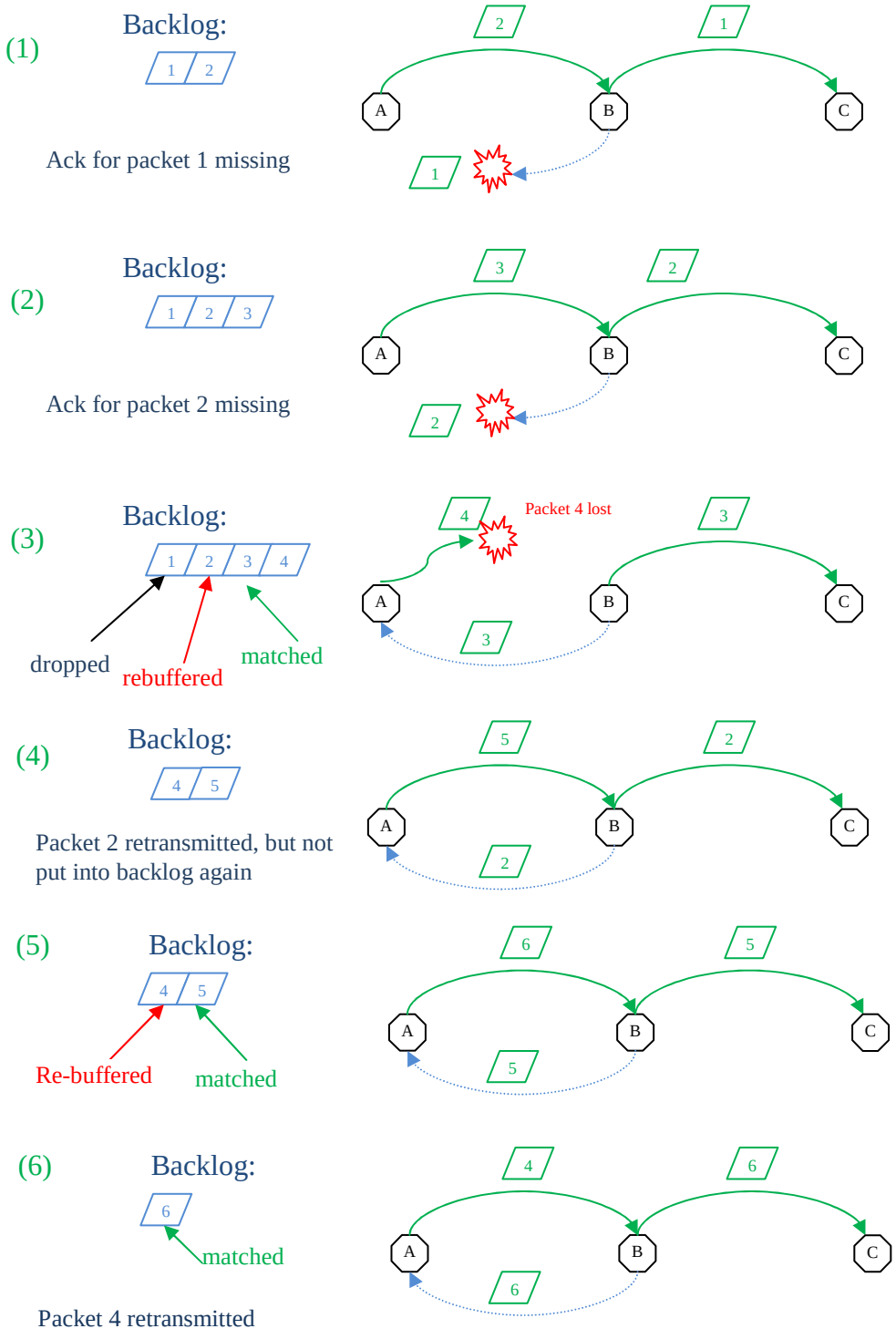

Figure 6-8 Example on How Local DMSG Recovery Works

Due to long transmission delay or out of sequence DMSG forwarding, sometimes the passive acknowledgement for a DMSG can come after it has already been re-buffered. In order to further reduce unnecessary DMSG retransmissions, when a passive DMSG acknowledgement matches with a re-buffered DMSG, the DMSG will be dropped from the buffer.

By doing Local Recovery, when the transmission failure rate is low, the mechanism can recover most packets dropped due to link failures. However, Local Recovery does introduce extra message forwarding overhead because some DMSGs may be wrongly considered lost. When the channel becomes congested, excessive re-buffering and re-transmission can make the congestion worse. To solve this problem, a minimum interval between consecutive DMSG re-buffers is used to put an upper bound on the frequency of retransmissions from a vrouter.

6.4.2 State Inferencing

To reduce the impact of not doing a state consistency check on all the messages, in VNAODV, we can allow the Backup Server nodes to use messages they hear from the local Server node to infer its state. We call this option State Inferring (SI).

With the SI option, when a Backup Server receives a server message from the same region, the message must be from its leader. If the message is an RREQ, the Backup Server can correct its BCAST id and the current route sequence number using values from the RREQ message. If the message is an RREP, the emulator node can correct its

sequence number, hop count and route entry flag[62] of the corresponding route in its routing table. If the message is an RERR, the corresponding route entry can be deleted. If the message is a DMSG, it can be used to correct route entry flags and the next hop used by the local vrouter for a destination.

The benefit of using SI is that it can patch the most relevant parts of a Backup Server's state piece by piece without resorting to explicit state synchronizations. The MSG-SYNC is costly compared to SI because each MSG-SYNC synchronizes the entire state of a Backup Server even when most parts of the state are still in sync. When a route on a Backup Server node is patched by SI, the messages generated by the node regarding the patched route are going to be the same as the messages generated by the vrouter in the future. Therefore, SI can reduce the costly MSG-SYNCs caused by state inconsistencies.

Reducing the number of consistency checks and synchronizations allows state inconsistencies. However, with the SI option, when there is a state inconsistency on a Backup Server route for a destination, each time the Server node sends out a DMSG toward the destination or a routing message regarding the destination, part of the inconsistency can be fixed. When the route in question is up and active on the local vrouter, eventually, the inconsistent route will be synchronized with the Server node's route.

[62] The status of the route must be RTF_UP.

6.4.3 Route Correction by Destination

Frequently, a destination node moves into a new region while a router is still delivering messages to its previous region. With the Early Receiving (ER) mechanism introduced in section 6.1.2 , the destination node can continue receiving DMSGs for a while. However, if the destination node keeps moving, eventually it will move out of the radio range of the last hop vrouter and the link will be broken. Then, a local repair might be needed. A simple application layer optimization can reduce the need for local repairs. When a node receives a DMSG destined for it and the DMSG is at its last forwarding hop but the message is delivered to a wrong region, the destination node sends out an unsolicited RREP message without specifying the next hop vrouter. Receiving this RREP message, vrouters in the neighborhood update their routes for the destination, without forwarding the message.

CHAPTER 7. Proactive Routing over the VNLayer

In order to verify that a wireline based routing protocol can be adapted to MANET using the VNLayer, we implemented VNRIP. At the application layer, VNRIP is a simplified version of RIP [14], a simple distance vector routing protocol suitable for small networks. RIP operates as follows. Each RIP router collects the number of subnets connected to it and builds its initial routing table. Then, each RIP router broadcast its routing table by a Response message to its neighbors. Routers use incoming Response messages to update their routing tables. The sender of a Response message that has the shortest route toward a subnet is picked as the next hop. Routers send their updated routing tables to their neighbors periodically. In the absence of message losses and router failures, eventually, every router will have a route to every subnet in the network. In addition to periodical routing table updates, RIP also allows a router to explicitly solicit routing tables from all of its neighbors or one of its neighbors using a Request message.

Clearly, the RIP protocol has to be modified to operate in a MANET situation because on vrouters, there is no directly connected subnet configured. Instead, each vrouter is directly connected with a set of client process in its region.

Because VNRIP is a proactive routing protocol, virtual node emulated routers (or vrouters, using the terminology introduced in CHAPTER 6) pre-calculate routes for all

the destinations in a network using route update messages flooded by other vrouters. Therefore, even when there is no data traffic at all, there is a constant routing overhead in a network running VNRIP. Therefore, the efficiency of VNRIP will be low when the data traffic is light. Implementing RIP over the VNLayer instead of on all MANET nodes, reduces the number of entities that are involved in the flooding of route updates and improves the reliability of the links between neighbor routers. Since the routing table on each vrouter contains route entries for each destination node, the state size on each vrouter can be large when the network contains a large number of physical nodes. To achieve reasonable performance, the protocol has to be carefully designed to reduce the state synchronization overhead and routing overhead. In this chapter, we present our implementation of VNRIP over the extended link layer model (with message losses) and extended VNLayer model[63].

7.1 Message Types

VNRIP uses three types of control messages. In addition to RIP's Request Messages and Response messages, it also uses Hello messages.

As described in CHAPTER 3, Hello message is actually a VNLayer message type. The message is broadcast by a client process to let the local vrouter and vrouters in the immediate neighbor regions know that they can reach the client process directly. From the VNLayer header of a Hello message, a vrouter can determine a client process's node

[63] We didn't optimize VNRIP using all the implementation choices provided by the extended VNLayer model.

id and the region it is in. The Hello message is not needed in RIP because directly connected subnets are manually configured on RIP routers.

Each Response message carries a set of route entries known to a vrouter based on its routing table. Each route entry in a Response message contains the address of a client process (rather than a subnet), the next hop used and number of hops needed by the vrouter to reach the client process. Here, the next hop is advertised in the Response message so that the next hop vrouter will not use the route entry to update its route for the same destination. In RIP, split horizon is used to prevent route loops. This can be done because in a wireline based network, Response Messages can be sent individually to a router's neighbors. In VNRIP, loop prevention is done differently[64] because every route on a vrouter is learnt from the same wireless interface card.

Request messages are used by a vrouter to look for routes from its immediate neighbor vrouters for a destination node. This can happen when a route entry expires or when a vrouter is just booted in an empty region. A Request message can be used by a vrouter to look for a route toward a single destination or solicit complete routing table updates from neighbor vrouters.

7.2 Routing Table

The routing table of a VNRIP vrouter includes a route for each client process in a MANET. Each route entry in a vrouter's routing table contains the following fields.

- Destination id: The address of a client process.

[64] It is done by attaching next hops with each route entry advertised. A vrouter rejects routes that uses itself as the next hop.

- Route flag: This Boolean flag indicates whether a route is valid or invalid.
- Hop count: The number of hops needed by a vrouter to forward a DMSG to a destination client process. If the hop count is 1, it means the destination is in the same region as the vrouter's. If the hop count is 2, it means the destination is in an immediate neighbor region. If the hop count is greater than or equal to 16, the route is treated as invalid.
- Next hop: The next hop (a vrouter) used by the current vrouter to forward packets to the destination. If the destination is in the local region, the next hop is the id of the local vrouter.
- Changed: This is a Boolean flag indicating whether a route entry has recently been changed or not. VNRIP vrouters use this flag to decide whether the entry needs to be included in a Response message.
- Lifetime: This field stores the expiration time of a route entry. With each incoming message, a vrouter checks each route entry's lifetime. If a route entry expires, the entry is set to invalid. And the route entry is flagged as "changed".

7.3 Routing Updates

Vrouters create and update route entries through four kinds of routing updates: Hello messages, Triggered Partial Updates, Complete Updates and On-demand Updates. Triggered Partial Updates and Complete Updates are based on the corresponding mechanisms in RIP. On receiving a Response message, a vrouter checks its routing table to see if any route can be updated with a better route from the message. On a vrouter, when a Response message for a destination is received from the next hop vrouter

currently used by the vrouter for the destination, the vrouter always updates the hop count and refreshes the lifetime of the route. When a route entry's hop count or next hop is changed, the route entry is flagged as "changed".

7.3.1 Hello Messages from Every Physical Node

Every second, every client process broadcasts a Hello message. Hello messages are used by vrouters to create or update route entries for directly connected client processes. In addition, each time a client process enters a new region, it also sends out a Hello message immediately. This is to ensure the vrouters in the neighborhood[65] can be informed about the region change quickly. This type of routing update is an essential difference between RIP and VNRIP. In RIP, routers have permanently attached networks, but in VNRIP, vrouters don't have permanently attached networks.

7.3.2 Triggered Partial Update

Each time a route entry is flagged as "changed" on a vrouter, the vrouter schedules a Triggered Partial Update (TPU) within a Triggered Update Interval (TUI). To reduce the number of routing updates, a vrouter doesn't schedule more TPUs until the scheduled TPU is sent. A TPU message carries all the "changed" routes on a vrouter. Once a TPU message is sent, all the routes on a vrouter are set back to "unchanged".

When the triggering event of a TPU is a Response message from another vrouter, the TUI is set to 1 second. When a TPU is triggered by a route change caused by an incoming

[65] A Hello message can be used by the vrouters in a client process's own region and immediate neighbor regions to update their routes to the client process.

Hello message, the TUI is set to a smaller value (0.5 second) so that the region change of a client process can propagate faster.

7.3.3 On-Demand Update

When a vrouter has just booted up in a region, it broadcasts a Request message. In response, neighbor vrouters set routes that are using the vrouter as next hop to "invalid" to prevent loop formation. In addition, the neighbor vrouters also send out On-Demand Updates containing their entire routing tables so the newly booted vrouter can construct its routing table.

When a vrouter receives a DMSG but there is no valid route for the destination, it broadcasts a Request message just for the destination of the DMSG. In response, neighbor vrouters that have routes toward the destination send back On-Demand Updates, carrying only the route asked for by the vrouter.

7.3.4 Complete Update

Because routes expire, complete routing table updates are necessary even when the network topology is static and there is no topology change. However, sending complete Updates frequently is costly because each complete Update contains the whole routing table a vrouter has. In order to reduce the routing overhead, a relatively long Complete Update Interval[66] (CUI) is used to control the minimum interval between two consecutive complete updates on each vrouter. During this interval, some potentially reachable destinations may be unreachable. When a vrouter doesn't have a route toward a

[66] 60 seconds in our implementation.

destination and none of its neighbor vrouters know a route to the destination, the vrouter might have to wait up to a CUI to receive a complete routing update from a remote vrouter before it can restore the route for the destination.

7.4 Data Message Forwarding

When a DMSG is received by a vrouter, the vrouter checks its routing table for a valid route. If there is a route available, the DMSG is forwarded to the next hop region by local broadcast. If there is no route available, the DMSG is buffered. As introduced above, a Request message is sent out by the vrouter to its neighbor vrouters. When a route is learned for the destination, the DMSG is moved to a second buffer. The vrouter delays the DMSGs moved into the second buffer a little while before they are forwarded using the routing table. The reason for this is to give the vrouter time to pick the best route based on the responses to the Request message. This way of handling undeliverable DMSGs is different from RIP, which simply drops the messages.

7.5 Route Maintenance

As in VNAODV, route maintenance is crucial to VNRIP's performance. (RIP doesn't do this because links between routers are assumed to be reliable most of the time.) Due to the use of local broadcast[67] on DMSG forwarding, the leading cause of end to end DMSG delivery failures in VNRIP is message collision. As in VNAODV, a passive DMSG acknowledgement mechanism is used in VNRIP to detect link failures. Each time a vrouter forwards a DMSG, the lifetime associated with the route entry used is shortened.

[67] This is the only option we used in our simulation for VNRIP.

If there is no passive DMSG acknowledgement from the next hop vrouter received, the route entry will expire soon.

At the last forwarding hop, upon receiving a DMSG, a destination client process sends back a Hello message as an explicit DMSG acknowledgement, so that the route entry used by the last hop vrouter can be refreshed.

When a link failure is detected on an entry for a route, the route is flagged as invalid. Within a Triggered Update Interval, the invalid route will be announced to neighbor vrouters so that they stop using the vrouter as next hop for the destination. Before this, when there are more DMSGs arriving for the same destination, the vrouter buffers the incoming DMSG and sends out a Request message. Each time a route entry is turned from “invalid” to “valid” by an incoming Response message, the DMSG buffer is checked. DMSGs in the buffer that can use the updated route will be sent out. If the route can’t be restored for the buffered DMSGs, they eventually time out and are dropped.

7.6 Loop Detection and Prevention

Loops can happen in VNRIP due to router and link failures and out of sync router state. In addition to the loop prevention and detection techniques introduced in section 6.2 VNRIP uses the following methods to detect loops.

First, if a vrouter learns from its routing table that the next hop vrouter of a DMSG is the vrouter that sent the DMSG, a loop is detected. The vrouter buffers the DMSG, sets the current route for the destination to “invalid” and informs its neighbors about the route

change right away[68] with a Response message. The vrouter also broadcasts a Request message to neighboring regions, asking for routes toward the destination.

Second, each DMSG carries a field recording the number of hops the message has traversed, if this hopcount value reaches 16 (the value for infinity), it is very likely the DMSG has been trapped in a loop. When such a DMSG is received, the message is dropped. As above, the vrouter also sets the route affected to "invalid", informs its neighbors about the change and broadcasts a Request message for alternative routes.

7.7 Optimizations based on VNLayer Implementation

VNAODV optimizations such as Direct Receipt and Early Receiving can also used by VNRIP at the last forwarding hop to shorten the forwarding paths and allow destination nodes to continue receiving DMSGs even after leaving its original region. In addition, as in VNAODV, Backup Server nodes to check for state inconsistencies on routing messages only. However, options such as reducing the state size by synchronizing hard state only, using Long Links to shorten forwarding paths, using Directed Broadcast for DMSG transmission, and route correction by the destination node are not implemented in VNRIP.

The following two subsections explain two cross layer optimizations of VNRIP based on the implementation of the Hello Message Generator and NRSM module in the Packet Classifier of our VNLayer implementation.

[68] As opposed to waiting for a Triggered Update Interval.

7.7.1 Hello Messages Sent and Managed by the VNLayer

To reduce the traffic overhead of Hello messages, on an emulator node, a client process can send Hello messages using the Hello Message Generator provided by the VNLayer[69]. As explained in section 4.5.2 , each time a message is sent from the VNLayer, the Hello Message Generator delays the next Hello message by another Hello Interval. With every incoming message with a VNLayer header (including the Hello messages) the VNLayer reports a Hello event to the application layer, which is used by VNRIP to update its routing table.

Compared with sending periodic Hello messages by the application layer itself, using the VNLayer to handle Hello messages reduces the Hello traffic overhead and allows a vrouter to use overheard messages to update its routing table. However, doing so requires a client process to have access to the Hello Message Generator at the VNLayer. This breaks the abstraction.

7.7.2 Neighbor Region Activeness

As presented in section 4.5.2 , by observing the messages that are sent out by virtual nodes, a virtual node keeps track of the active state of the virtual nodes in its immediate neighbor regions. Using this feature, in VNRIP, a vrouter treats a route as broken if the downstream region is inactive.

When a vrouter receives a DMSG, if it has a valid route for the DMSG, it checks the VNLayer state to see if the next hop virtual node is active. If not, the vrouter buffers the

[69]On a pure client process, this option can't be used.

DMSG, set the route entry used to “invalid” and sends out a Request message to try to fix the route. If the next hop virtual node is active, the vrouter forwards the DMSG and sets the timer used to determine the active state of the next hop virtual node to 1.5 times the maximum expected one hop Round Trip Time (RTT). This way, the neighbor virtual nodes that are used by a vrouter to relay DMSGs are set to inactive faster. This is a cross-layer optimization because it allows the application layer to modify the VNLayer settings.

7.8 Summary

In addition to the optimizations explained in the last section, state inferencing at the application layer, as introduced in section 6.4.2 can also be used by VNRIP so that Back Servers can use Server messages to patch their state.

This simple version of RIP protocol was implemented very quickly[70]. As we’ll see in the next chapter, although VNRIP generates heavy routing traffic and synchronization traffic, it provides reasonable delivery performance that is not much worse than VNAODV under similar settings. With the optimizations applied to VNAODV added to VNRIP, it is likely to perform much better. This verifies the intuition that the VNLayer approach can be used to adapt wireline protocols to MANET easily.

[70] The coding and debugging only took about 3 weeks.

CHAPTER 8. Performance Evaluation on VNLayer based Address Allocation and MANET Routing

So far, I have discussed our implementation and optimizations on VNLayer based Address Allocation and MANET routing. In this chapter, I present the simulation results on VNDHCP, VNAODV and VNRIP. As we are going to see, VNAODV performs very well. This proved that the VNLayer approach can be used to adapt wireline protocols to MANET. Routing applications pose greater challenge to the VNLayer approach. From the simulation results, we are going see how the optimizations on the VNLayer implementation based on the extended VNLayer model improved the performance of VNAODV.

8.1 Performance Evaluation on VNDHCP

8.1.1 Simulation Settings

For VNDHCP, we ran our simulations using ns-2.31 on a Linux machine with an Intel Pentium 4 3.20GHz CPU and 512M bytes memory. In ns-2, the wireless propagation model is set to “freespace”. Two network settings were used: a small network of 40 to 120 nodes that contains 16 87.5m×87.5m regions in a 350m×350m and a large network of 160 nodes that contains 64 87.5m×87.5m regions in a 700m×700m area. All the mobile nodes are set to emulate the VNLayer. The packet receiving range is set to 250

meters to make sure that a message sent from a region can reach every node in the immediate neighboring regions.

In each region, the address pool size is set to 30 to reduce the chance that a server runs out of addresses. The region leaders are set to send out a Heartbeat message every second. The lease time is set to 400 seconds. Each simulation ran for 40000 seconds, or 100 lease times. For each data point, the simulations are repeated 5 times with different node mobility traces. Error bars are created with confidence intervals with confidence level set to 95%.

Table 8-1 Settings for 5 Motion Speed Modes

	slow	medium slow	medium fast	fast
Minimum speed (m/s)	0.73	1.46	2.92	5.84
Maximum speed (m/s)	2.92	5.84	11.68	23.36
Minimum pause time (s)	400	200	100	50
Maximum pause time (s)	4000	2000	1000	500
Average cross time (s)	48	24	12	6

We evaluated the performance of the system with the nodes moving at various speeds. Using the random waypoint model, ns-2 mobility traces were generated for four speed modes: slow, medium slow, medium fast and fast. The settings used to generate the mobility traces for each speed mode are given in Table 8-1.

Slower speed means that it takes a node longer to travel across a region so that it is less likely to be far away from its server when it needs to renew its lease. For example, for speed mode “slow”, the 2200 second average pause time is 5.5 times of the lease time.

The average time for a moving node to travel across a region is 48 seconds. This means that during one lease period, a moving node on average may travel across 8 regions. For the speed mode “fast”, the average pause times and crossing times are 8 times shorter.

8.1.2 Simulation Time

We first compared the simulation speed of VNE and VNSim with the small network setting. Table 8-2 lists the simulation time of VNE and VNSim for various total numbers of nodes. The simulation time increase of VNSim is roughly proportional to the square of the total number of nodes because each node needs to handle messages from all of its neighbors.

With 40 nodes, VNSim runs around 158 times faster than VNE. However, for the set of simulation we did, VNSim doesn’t scale as well as VNE. This may be because of the more accurate simulation of the link layer, especially message collisions. With 80 nodes and 120 nodes in the network, VNSim runs about 82 times and 60 times faster than VNE, respectively.

Table 8-2 Simulation Speed of VNE and VNSim

	40 nodes	80 nodes	120 nodes
VNE simulation time	6.32 hours	13.07 hours	22.09 hours
VNSim simulation time	2.4 minutes	9.53 minutes	22.23 minutes

8.1.3 VNLayer message overhead

The first question we want to answer through the simulations is whether a VNLayer based system is practical. The main concern here is whether there will be excessive

control overhead, including the leader election overhead and state synchronization overhead. Because SYN-ACK messages can be big[71], large number of synchronizations can cause heavy traffic. In addition, during state synchronizations, Backup Server nodes that are out of sync ignore all incoming messages and stop acting as Backup Servers. This hurts the failover capability of the virtual nodes.

With the large network setting and speed mode “slow” and “fast”, we did simulations with various renewal message forwarding methods (flooding and geographical routing) and forwarding hop limits (1 to 8). More details on the simulations can be found in the next section. In the worst case, the case with speed mode “fast” and 8 hop flooding used for RENEW messages, the virtual node layer generates about 482 messages per region per lease time. Over 75% of the messages are the Heartbeat messages sent by region leaders. The numbers of LeaderRequest and LeaderReply messages are on the order of 24 and 50 per region per lease time, respectively. There are more LeaderReply messages since multiple nodes may respond to the same LeaderRequest message. The average numbers of SYN and SYN-ACK messages are both about 20 per region per lease time. These numbers almost stay constant with different forwarding hop limits and forwarding methods. This suggests that the VNLayer message overhead is not affected much by how the RENEW messages are forwarded.

From the simulation results, it’s estimated that the packet overhead generated by the VNLayer from a single region ranges between 200 bps and 450 bps. Because a node in a

[71] Here, the SYN-ACK message carries the application state for the 30 addresses managed by each virtual node.

region can hear messages from up to 9 regions, the combined channel bandwidth overhead for any region can range between 2Kbps and 4.5Kbps. Because the 802.11 radio channel's bandwidth is now typically 54Mbps, the virtual node layer uses less than 0.1% of the bandwidth. The system is therefore practical because it won't affect the normal operation of other protocols on the mobile nodes.

8.1.4 Different Renewal Methods

The next problem is how to engineer the protocol at the application layer to get the best performance. The renewal process is critical to the performance of VNDHCP because when a renewal fails, a client has to stop using the current address and ongoing sessions may have to be disconnected. We used the number of addresses allocated to a client during the simulations to measure the effectiveness of the renewal process. The more addresses that the client has during a given period, the more times a session may be disrupted.

With the large network setting, we run simulations with different forwarding hop limits for RENEW messages and forwarding methods, under speed mode "fast" and "slow". The hop limit ranges from 1 to 8. With hop limit 1, RENEW messages are not forwarded, at all. With hop limit 8, RENEW messages can be forwarded by up to 7 regions, including the local region of the client processes.

8.1.4.1 Fast Moving Case

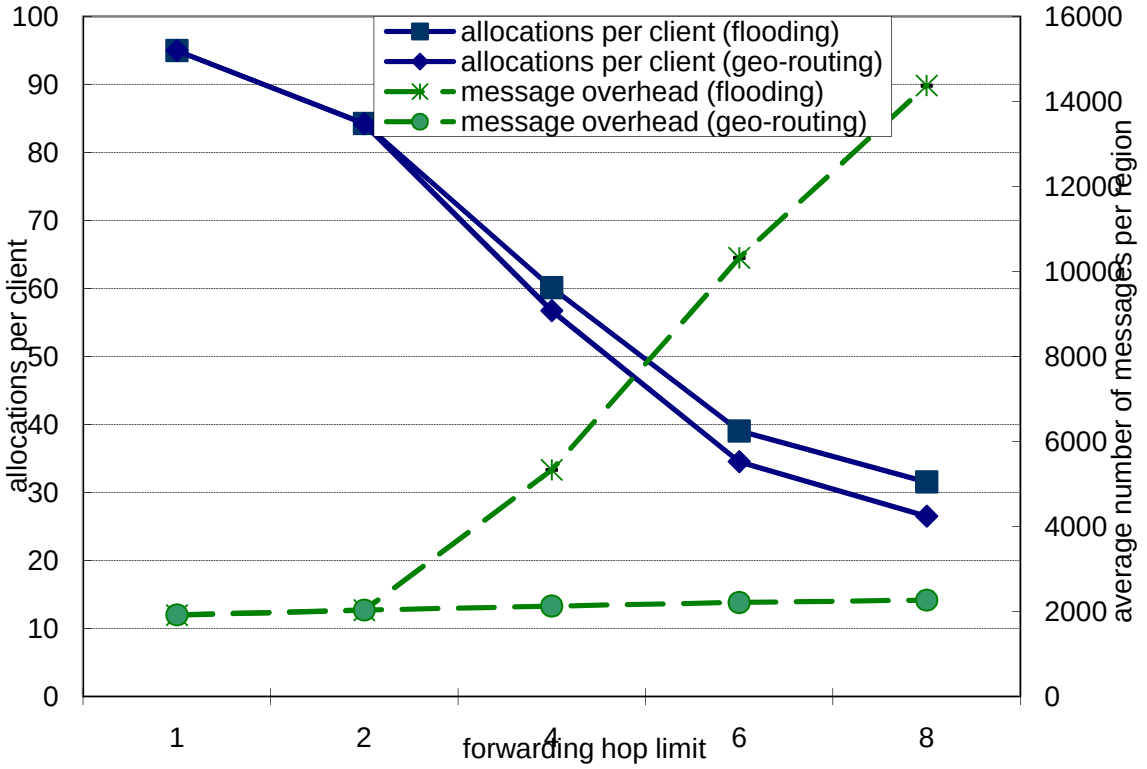

Figure 8-1 Allocation performance with different renewal methods, large network, fast moving case

With speed mode "fast", the average number of allocations per node ranges from 27 to 95 during the 40000 second simulations, as shown in Figure 8-1. Re-allocations do happen a lot in this case due to the fast node motion speed. With hop limit 1, forwarding for renewal messages are not allowed, almost every single renewal fails and the client process needs a re-allocation.

For the flooding case, with larger hop limits on how many hops a RENEW message can be forwarded, each client process needs fewer and fewer re-allocations. However, this

comes at the cost of rapid increasing message overhead. The curve for the number of allocations per client flattens as the hop limit approaches 8.

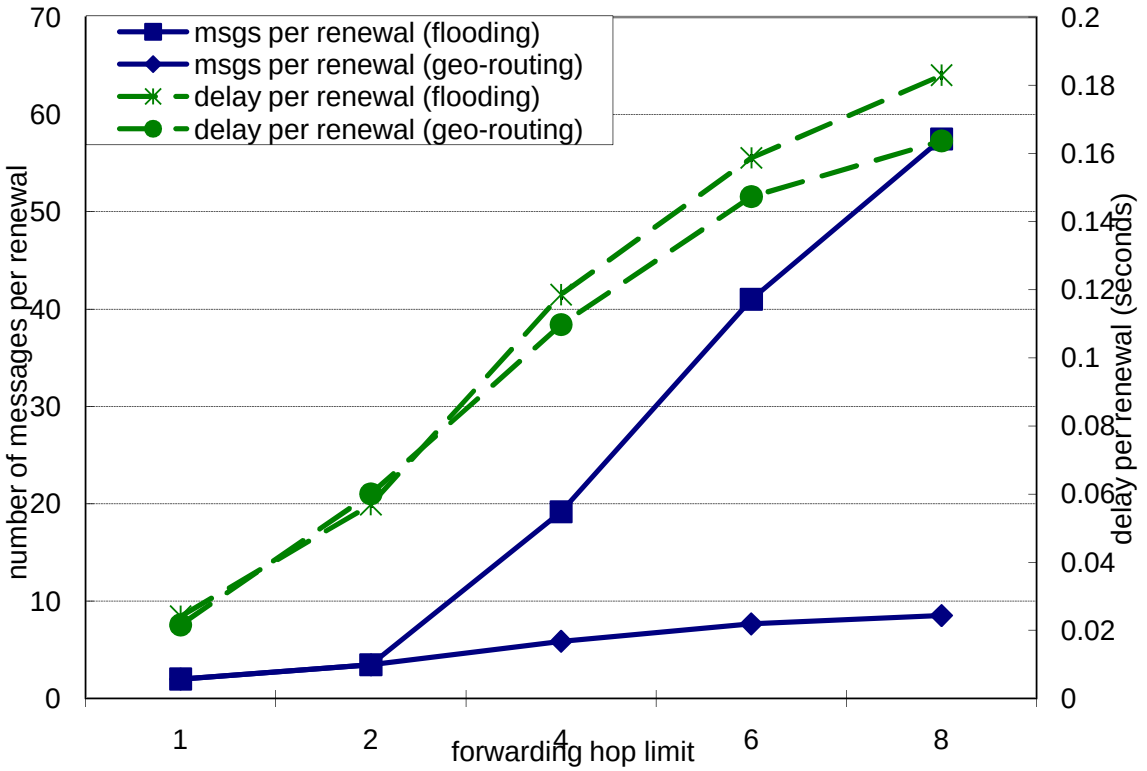

Figure 8-2 Renewal overhead with different renewal methods, large network, fast moving case

For the geographical routing case, the allocations per client and message overhead per region started the same as the flooding case when the hop limit is 1 and 2. This is because using geographical routing on the RENEW message doesn't reduce the message overhead in either case. After that, the number of allocations per client decreases faster with greater hop limits. In addition, using geographical routing generates much less message overhead because only one forwarding path is used for every RENEW message. With hop limit 8, the geographical routing case generates less than one sixth of the message overhead of the corresponding flooding case.

Figure 8-2 shows the renewal message overhead and renewal delay with different renewal methods and different hop limits. Using flooding, with hop limit 1, the renewal is limited to the local region and a successful renewal takes exactly 2 messages, one RENEW message and one RACK message. With greater hop limits, it takes each renewal more time and messages to finish. With hop limit 8, a renewal takes around 55 messages, showing that most of the regions are involved in the flooding of the RENEW messages.

With geographical routing, the renewal message overhead is the same as the flooding case with hop limit 1 and 2. But the average renewal message overhead increases much more slowly than the flooding case. Even in the case with hop limit 8, a renewal on average takes less than 10 messages.

Figure 8-3 shows the distribution of the time percentages that each client doesn't have an address for. The value at each data point is the percentage of clients that don't have an address for more than a certain percentage of the simulation time. When flooding is used for the RENEW messages, the average addressless time percentages of the clients decrease with higher hop limits. With hop limit 8, only 4 percent of the clients don't have an address for more than 0.2% of the simulation time. The geographical routing cases show same trends, we only show the hop limit 8 case here, which performs better than the 4 hop flooding case and worse than the 6 hop flooding case.

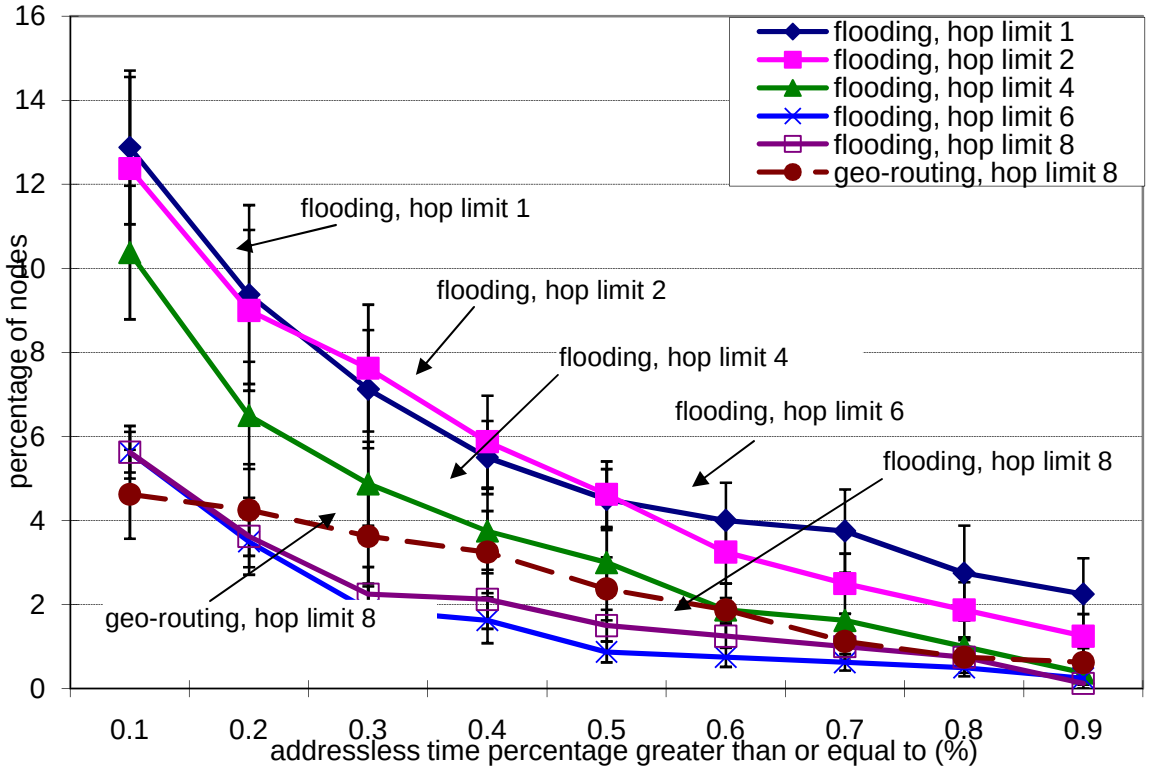

Figure 8-3 Distribution of addressless times with different renewal methods, large network, fast moving case

8.1.4.2 Slow Moving Case

With speed mode "slow", we repeated the simulations. As shown in Figure 8-4, in the flooding case, the average number of allocations per client turns flat can even go up a little with greater hop limits after hop limit 4. This means that when nodes are moving slowly, the flooding of RENEW messages by more hops, instead of helping; can hurt the allocation performance with more message collisions and congestions.

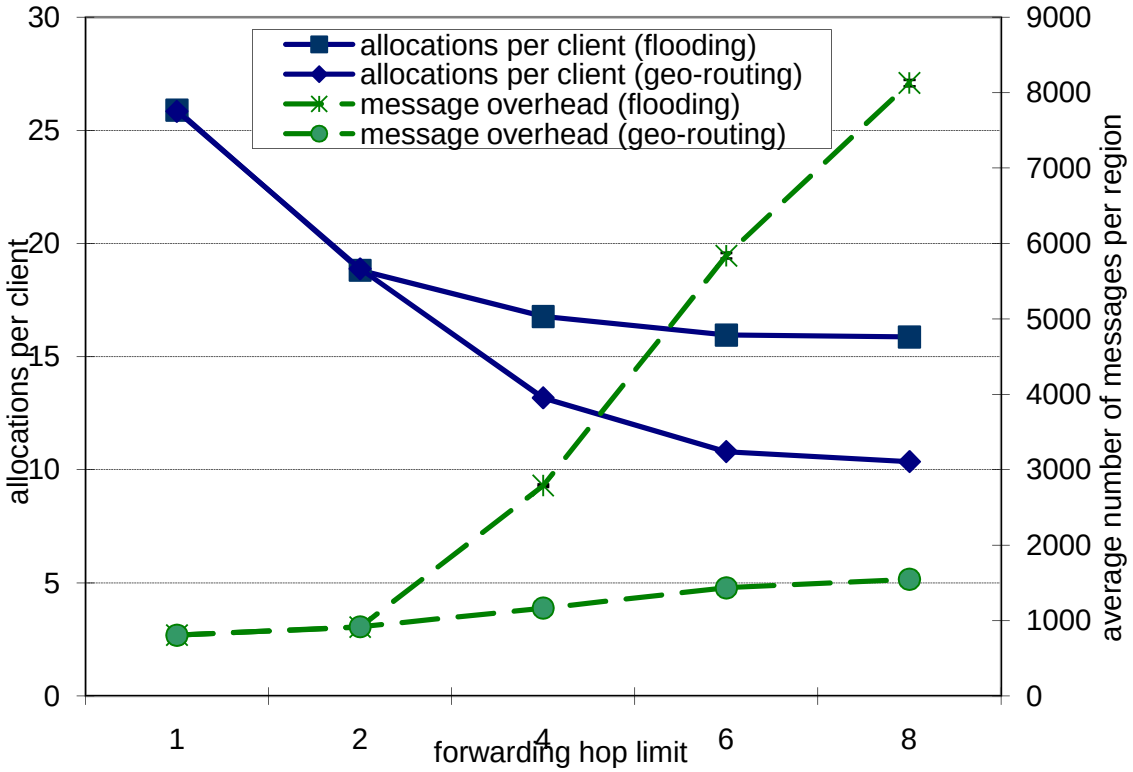

Figure 8-4 Allocation performance with different renewal methods, large network, slow moving case

In the geographical routing case, the number of re-allocations per client gets lower and lower with increasing forwarding hop limits while the message overhead increases much slower than the flooding case. Here, the geographical routing case performs much better than the flooding case.

In the slow moving case, the curves in Figure 8-5 show similar trends as those in Figure 8-2. The difference is that the average renewal message overhead only reaches 28 even in the flooding case. This is because with slower node motion speeds, a renewal message on average needs to travel through fewer hops.

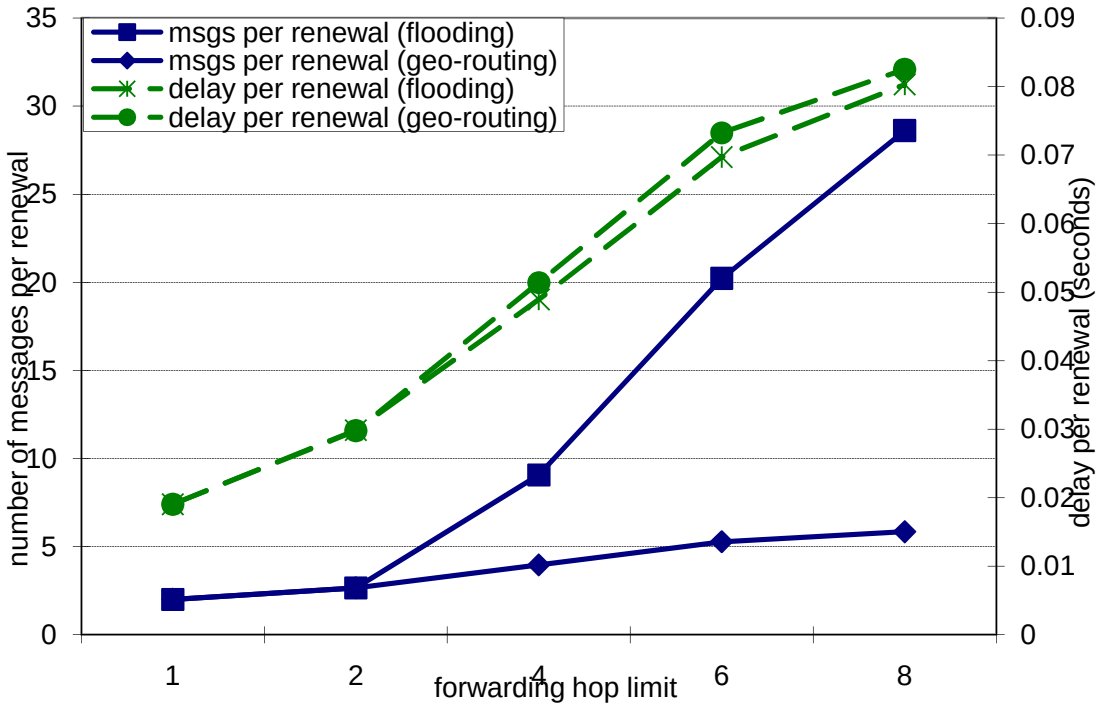

Figure 8-5 Renewal overhead with different renewal methods, large network, slow moving case

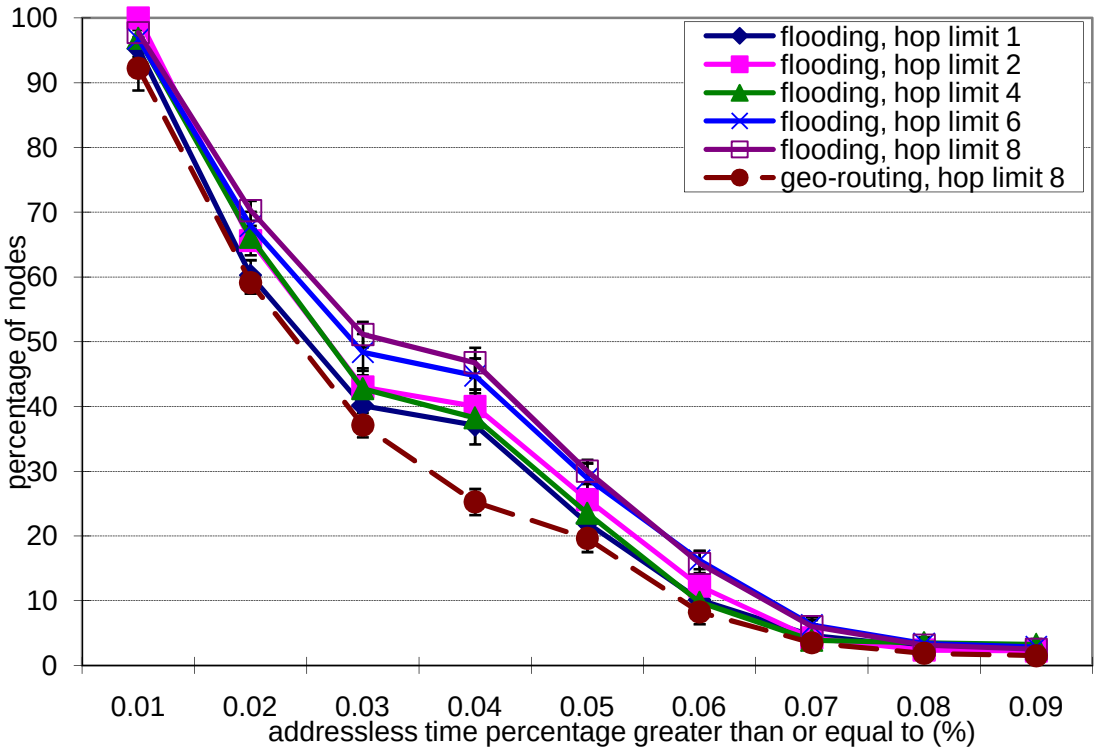

Figure 8-6 Distribution of addressless times with different renewal methods, large network size, slow moving case

Figure 8-6 shows that with slow node motion speeds, the average addressless percentages we get here are better than those we get with the fast moving case, where the nodes move about 8 times faster. In the flooding case, the average addressless time percentage gets worse with greater hop limits. This again demonstrates that when nodes move slowly, flooding the RENEW messages by more hops can hurt the allocation performance. Here, the 8 hop geographical routing case performs better than all the flooding cases.

8.1.5 Different Node Densities

Now the question is what happens with more mobile nodes in the system. Using the small network setting and speed modes from “slow” to “fast”, we run simulations with 40, 60, 80, 100 and 120 nodes, with geographical routing used for the RENEW messages and the forwarding hop limit for renewal messages set to 5.

From Figure 8-7, we can first see that with the small network setting, the allocation performance is much better than what the system gets with the large network setting, because with large networks, the renewal messages has to travel through more forwarding hops and are more susceptible to message loss and routing failure.

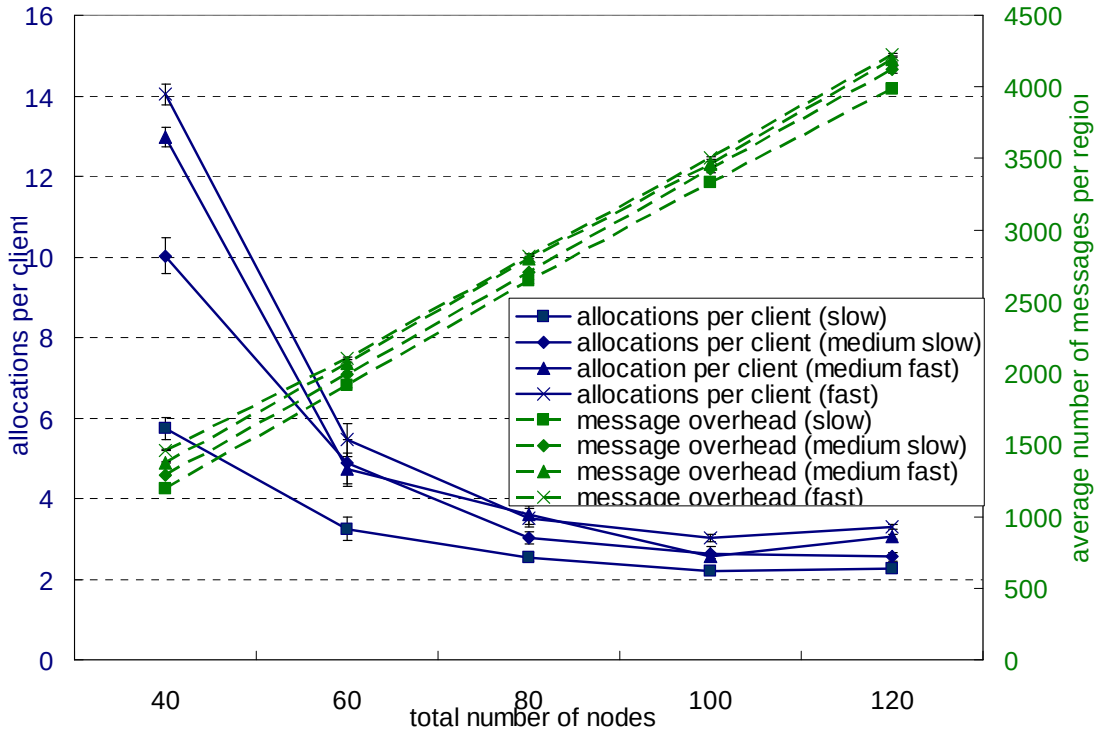

Figure 8-7 Allocation performance with different node densities and different node motion rates when geographical routing used for address renewals

More importantly, the figure shows that with higher node densities, the allocations per client decreases quickly at first and then gets stable or even increases slowly. At the beginning, with higher node densities, the probability that a region is empty and the virtual node in it is down drops very quickly and renewals are less likely to fail. Then, when the node density becomes too high, the benefit above will be offset by the increasing message overhead. This suggests that the system performs the best with a node density that is neither too low nor too high.

In addition, with increasing node densities, the curves for different motion speed modes get closer and closer to each other. This indicates that with higher node densities, the system is less and less sensitive to node motion speeds, because the virtual nodes are

more likely to function most of the time even when nodes are moving fast. This helps the clients keep their addresses longer.

In addition, with increasing node densities, the message overhead increases linearly because each node introduces the same amount of application layer burden to the system. Together with the curves for allocations per client, we can see with the network densities investigated here, we can see the performance of VNDHCP scales well with increasing network densities.

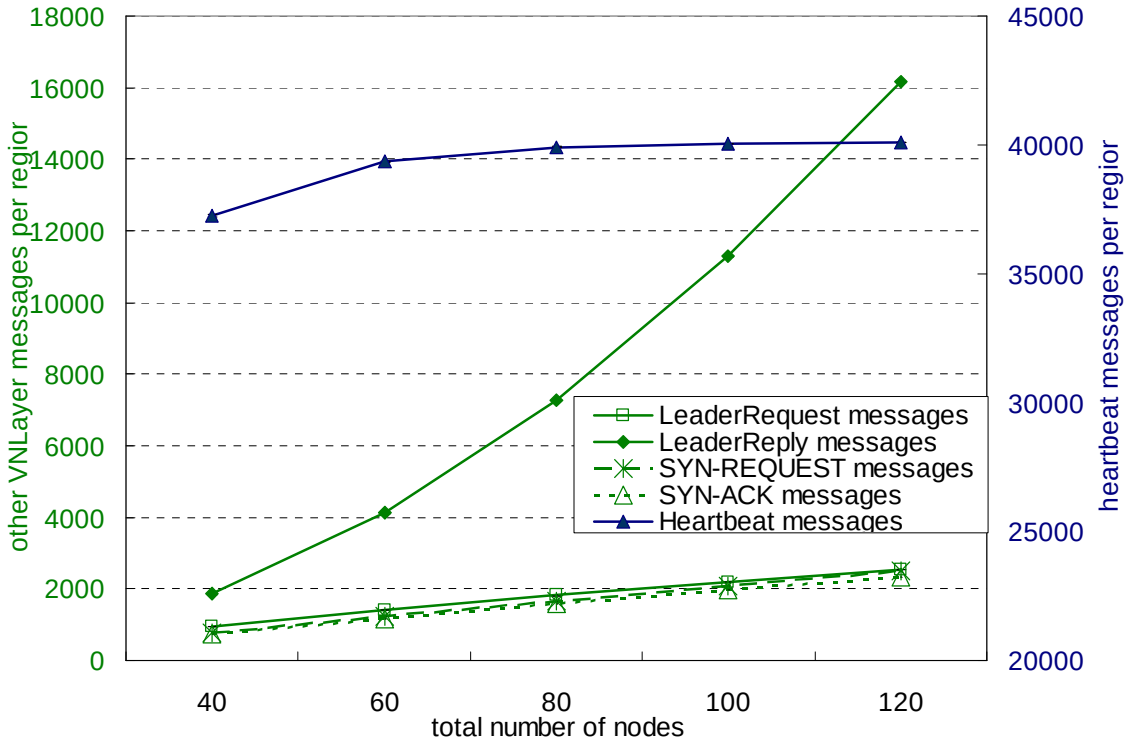

Figure 8-8 Virtual node layer message overhead with different node densities

Figure 8-8 shows the VNLayer message overhead with increasing node densities for the speed mode “fast”. The number of Heartbeat messages stays constant since the number of region leaders doesn’t vary much for the set of node densities used in the simulation. The

average number of LeaderRequest, SYN_REQ and SYN_ACK messages per region increase almost linearly because each node sends out roughly the same number of such messages. The number of LeaderReply message increases much faster because we allow non-leaders in a region to send LeaderReply messages to reject leadership requests too. With higher node densities, more and more nodes may respond to the same LeaderRequest message. To solve this problem, in our research on VNLayer based routing application, we only let the leader node send LeaderReply messages.

In these simulations, we didn't investigate the impact of the use of the LeaderLeft messages. The Heartbeat interval is set to 1 second. The periodic Heartbeat messages compose the greatest portion of VNLayer message overhead. With LeaderLeft message used, the performance VNDHCP is expected to be better because fewer Heartbeats are needed and the leadership switching can take place much faster.

8.1.6 Summary

In this section, I summarize on what is learned through the simulation studies on VNLayer based MANET address allocation.

8.1.6.1 On the Performance of VNSim

Simulation results in this case study showed that VNSim runs much faster than VNE, and is suitable for a network of up to a few hundred mobile nodes. VNSim can be used to validate any VNLayer-based application.

8.1.6.2 On the VNLayer

The simulation results on VNDHCP show that the VNLayer overhead is quite small. This proves that VNLayer based systems are practical. One reason for the low VNLayer overhead is the VNLayer implementation choice we take at the VNLayer. We choose to let Backup Servers to check only Server Messages from the Server node to look for state inconsistencies. The number of state synchronizations due to message losses (MSG-SYNCs) is reduced. However, the main reason why the VNLayer overhead is small is because low application layer overhead due to the long address lease time (400 seconds). As we are going to see in the simulation results for VNLayer based MANET routing, the use of VNLayer can cause heavy control traffic overhead.

Simulation results were obtained for a wide range of configurations, from a small 16 region network (350 meters by 350 meters) to a large 64 region network (700 meters by 700 meters) and for a variety of mobile node speeds, from a slow walk to vehicle speed. VNDHCP is proven to work well with all the simulation settings. For over 99.9% of the time; most client processes have addresses allocated. The overlay network is quite stable when the density of mobile nodes is high enough to make virtual node failure unlikely. Therefore, the main reason for the good performance is that failover capability provided by the VNLayer. Without the replicated state maintained by Backup Server in each region, each time a virtual node is down, all the client processes that got their addresses from the region would have to request for a new address.

As any cluster-based solutions, the use of the VNLayer approach makes VNDHCP scale well with greater node densities[72] because the number of entities that have to be involved in the address allocation is bounded by the number of regions in a network. As shown in section 8.1.5 , a VNLayer based application would perform the best with a MANET that is dense enough so that the virtual nodes can stay up longer.

As discussed earlier in section 5.5 , VNDHCP doesn't have to handle network partitions and mergers in a MANET due to the fixed region settings. In addition, the geographical location based region settings also allows VNDHCP to use the light weight geographical base routing to forward address renewal messages. Finally, with most programming handled in the VNLayer, the coding for the VNDHCP server and client are made easier.

All these benefits demonstrate that the VNLayer approach can be used to adapt a wireline protocol like DHCP to the MANET environment.

On the other hand, the use of the VNLayer approach does bring a few complications. First, as discussed in section 5.2.3 , special care has to be taken to avoid duplicate address allocations in VNDHCP. The loss of state due to virtual node resets is a general problem to any VNLayer based applications. This means the application layer code has to be modified. Second, in VNLayer based applications, since servers are virtual nodes emulated by multiple physical nodes, state inconsistencies can lead to complicated situations such as address duplication in VNDHCP. In a lossy channel, it is hard and

[72] With or without using VNLayer, we can expect an MANET address allocation protocol to perform worse when the geographical size of the network increases because data forwarding will be less reliable.

costly (in terms of VNLayer overhead) to keep states on emulator nodes synchronized most of the time.

8.1.6.3 On the Implementation Choices in VNDHCP

The simulation results also show that flooding hurts the scalability of protocols and should be avoided. As opposed to using flooding, the use of a simple geographical based routing to forward RENEW message greatly improved the address allocation performance.

Many address renewals still fail when empty next hop regions are picked to forward RENEW messages. To improve the success ratio of renewal attempts, a more reliable routing algorithm, rather than the simple geographical routing, shall be used. For example, VNAODV can be used to work together with VNDHCP.

8.2 Performance Evaluation on VNAODV and VNRIP

In this chapter, I present performance evaluations results on the two VNLayer based routing protocols, VNAODV and VNRIP. The performance results we got with the AODV code provided by the ns2 package is used as a benchmark in performance evaluations.

First, I present the simulation results we got with our initial implementation of VNAODV and our implementation of VNRIP. For this version of VNAODV, the major optimizations using the capabilities provided by the extended VNLayer model are not used. The application layer optimizations (for example, the local recovery option) for VNAODV are not used either. This section serves to present the base line performance of

VNAODV and VNRIP and to verify the major causes of performance issues with VNLayer based routing protocols.

In the three sections that follow, I present the effect of the three major optimizations we did to the VNAODV using the features provided by the extended VNLayer model. As we are going to see, with state synchronizations overhead reduced, forwarding path shortened and reliability of data transmission improved, VNAODV can outperform AODV due to the reduce control overhead and improved route stability.

A major strength of the VNLayer approach is that virtual node can maintain replicated state. On one hand, this make the virtual node able to maintain persistent state and be fault tolerant. On the other hand, doing so requires extra control overhead. We use one section to go to depth on how state replication affects the performance of VNLayer based routing protocols. As we are going to see, strict state synchronization is not necessary for VNLayer based MANET routing. In addition, Message Sync is not as important as Motion Sync.

In addition to the three major optimizations that greatly improved the performance of VNAODV, there are other VNLayer optimizations and application layer optimizations. We investigate the effect of these optimizations carefully with two separate sections.

To further validate the simulation results, we did more simulations with various node motion rates and a larger network size. The simulation results will be presented at the end of the chapter.

In our simulations, a 700m x 700m network is divided into 64 87.5m x 87.5m square regions. The network contains 60 to 240 mobile nodes. The radio range on each mobile node is set to 250 meters so that a message sent out by a node in a region can be heard by any other node in the same region and the immediate neighbor regions. The 802.11 channel bandwidth is set to 11Mbps, with RTS-CTS disabled.

Node mobility patterns are generated by CanuMobiSim-1.3.4 using the Random Waypoint Model. Two motion modes, slow mode and fast mode are used. With the slow motion mode, the minimum pause time is set to 100 seconds and maximum pause time is set to 200 seconds. The minimum motion rate is set to 0.73 m/s and maximum speed is set to 2.92 m/s (average speed 1.825m/s). With the fast motion mode, pause times are set the same way as in the fast motion mode. The minimum motion rate is set to 5.84 m/s and maximum speed is set to 23.36 m/s (average speed 17.52m/s).

Various number of Constant Bit Rate (CBR) sessions are created between random pairs of mobile nodes. No two sessions share either the source node or the destination node. Each session is set to transmit ten 64 byte UDP messages per second and to last throughout the simulation time. Each simulation lasts 450 seconds. The trace for the first 50 seconds in each simulation is skipped to allow the routing to stabilize before measurements are started. We repeated each simulation 10 times for each data point collected. Error bars are generated with confidence level of 95%.

AODV uses 30 for the maximum RREQ TTL in ring search. In our implementation, we set the maximum TTL to 10 in the expanding ring searches in both standard AODV and VNAODV.

We evaluated the delivery performance of the protocols simulated using the following metrics: packet delivery fraction, path length of successful end-to-end deliveries, end-to-end delivery delay and network wide traffic overhead for various types of overheads, in terms of bits per second. For most simulations, we set the network size to 240 nodes and the node mobility to fast motion mode, making it a dense and highly dynamic network.

8.2.1 VNRIP and Base Line Implementation of VNAODV

In this section, I present the simulation result we got with an initial implementation of VNAODV (for the rest of this chapter, we make modifications based on this implementation) and our implementation of VNRIP, as described in Chapter 7. Here, among the VNLayer optimizations, both VNAODV and VNRIP use selective state synchronization and selective state consistency checks in order to bring down the state synchronization overhead and get reasonable performance. The optimization that allows a client process to receive DMSGs directly from an immediate neighbor region (DR) and the optimization that allows a client process to continue receive a DMSG even if the packet is not destined for its region (ER) are also used by both VNAODV and VNRIP. As explained in section 6.3.2 , these two simple optimizations can save a few forwarding hops at the end of a forwarding path. However, optimizations such as Long Links, Directed Broadcast and Powerful Emulator are not used by these two implementations.

This version of VNAODV doesn't use any of the application layer optimizations introduced in section 6.4 , either.

8.2.1.1 Packet Delivery Fraction (PDF)

Figure 8-9 shows the PDF of successful end-to-end DMSG deliveries for AODV, VNAODV and VNRIP, with different number of CBR sessions created among the physical nodes. The plot shows that both VNAODV and VNRIP can outperform AODV when the number of CBR sessions is low. However, they don't scale as well as AODV, whose delivery ratio stays roughly the same with more sessions.

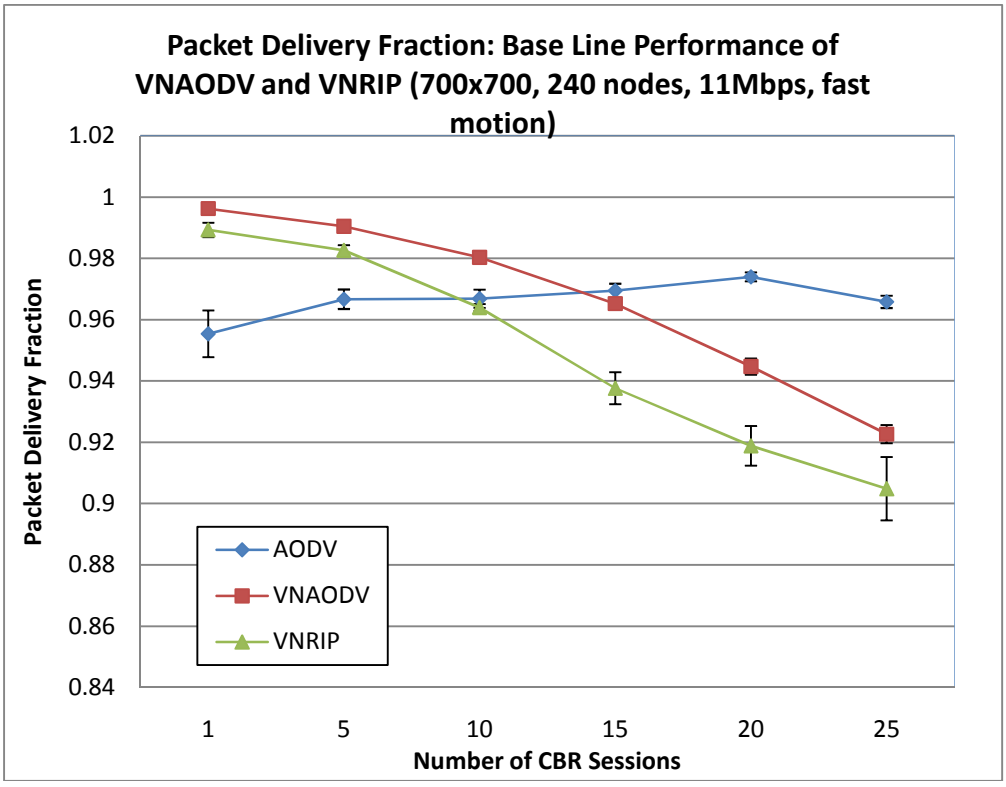

Figure 8-9 Packet Delivery Fraction of AODV, VNAODV and VNRIP

VNRIP performs worse than VNAODV. The most important reason is that VNRIP is a proactive protocol. Its routing overhead is proportional to the network size rather than the

data traffic load. With a dense network like the one we use, even with small number of sessions, its routing overhead is significant already. Another reason is that VNRIP is not equipped with some of the optimizations VNAODV has. For example, unicast based delivery of DMSGs at the last hop is not used for VNRIP. Therefore, the destination client process has to send route update messages to the last hop vrouter as explicit DMSG acknowledgements.

8.2.1.2 Forwarding Path Length and Forwarding Latency

Figure 8-10 shows the length of forwarding paths created by AODV, VNRIP and VNAODV that are used by successful DMSG deliveries, with various number of CBR sessions. We can see, VNAODV and VNRIP both create much longer forwarding paths than AODV. This is because the feature in the extended VNLayer model that allows any pair of virtual nodes and any client processes to communicate is not used. Long forwarding paths leads to long forwarding delays and more frequent delivery failures. Figure 8-11 verifies that the forwarding latencies of VNAODV and VNRIP are both much longer than AODV's. Here, we can see in terms of the length of forwarding paths and packet delivery latency, VNAODV and VNRIP performs roughly the same.

In Figure 8-11, we can also see that AODV's delivery latency actually improves with greater number of CBR sessions. One reason for this is that with more sessions, more routers launch local repair attempts that can help build reverse paths toward these routers. Some of these routers can be the destinations of other CBR sessions. Due to the saved route discoveries, the average delivery latency is shortened with larger number of CBR sessions.

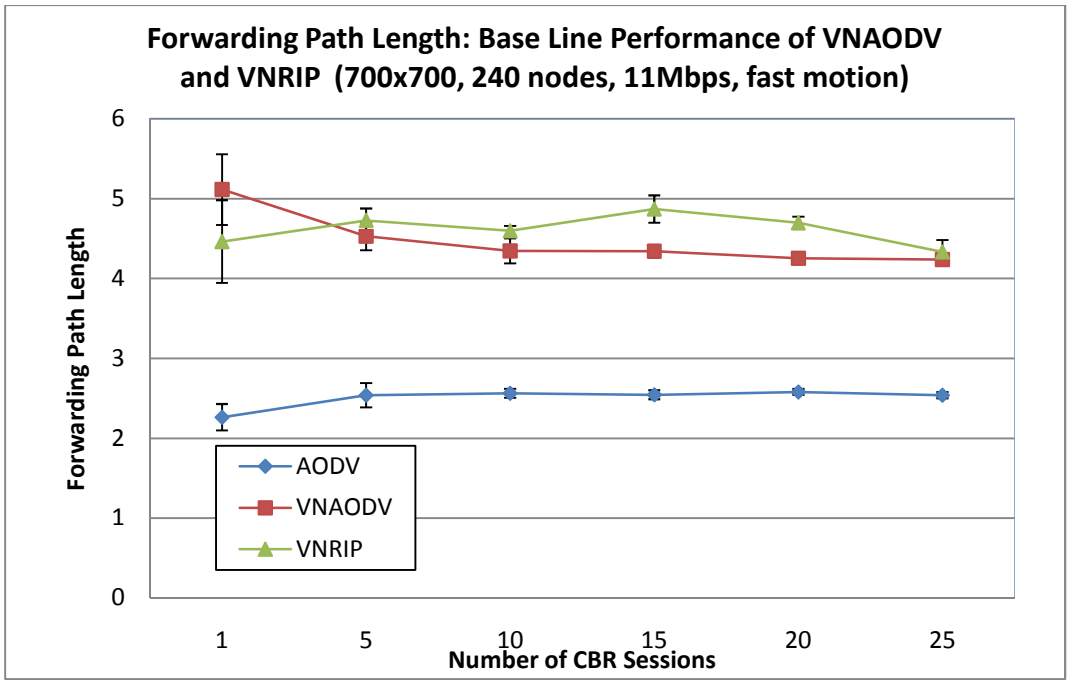

Figure 8-10 Length of Forwarding Paths Created by AODV, VNRIP and VNAODV

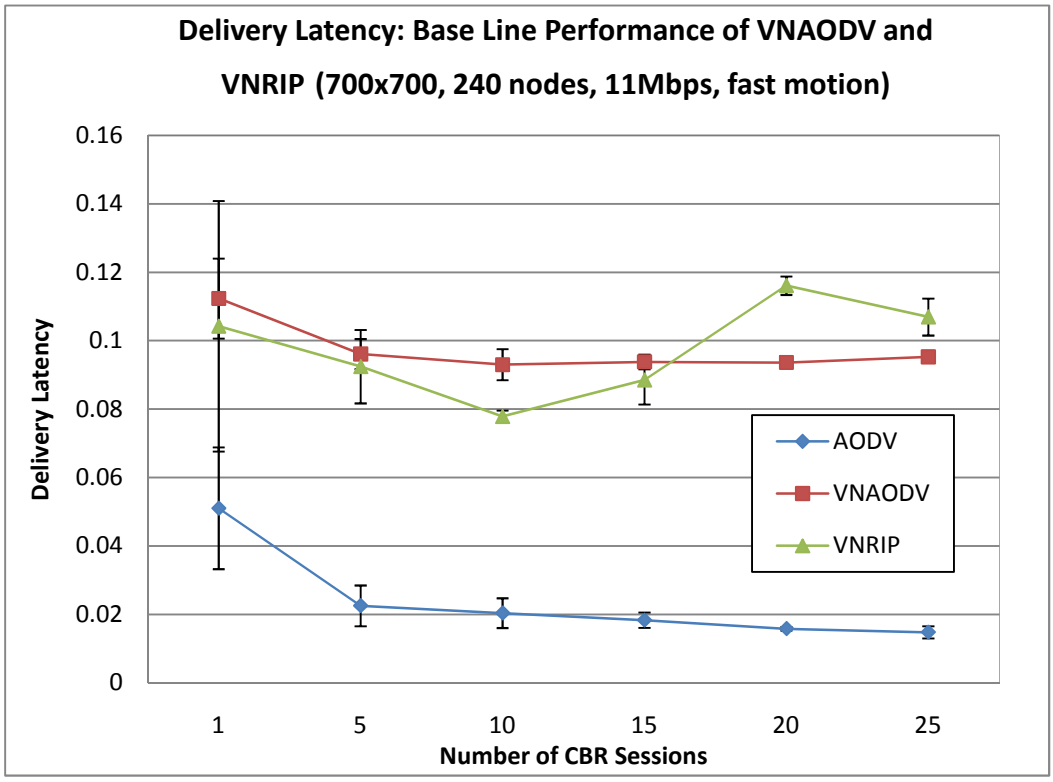

Figure 8-11 End to End DMSG Delivery Latency of AODV, VNAODV and VNRIP

In addition to the longer forwarding paths created by VNAODV, there are two other reasons that the end-to-end delivery latencies of the VNAODV are longer than AODV's. First, the VNLayer delays the incoming messages for a short period of time so that they can be sorted using their time stamps. This inserts a delay at every forwarding hop. Second, at every forwarding hop except the last one, a DMSG is broadcast to the wireless channel. To reduce collisions, every broadcast message is delayed by a random period of time before it is sent out to the channel. This "jitter sending" technique inserts another piece of delay at each hop.

8.2.1.3 Traffic Overhead

The more channel bandwidth a protocol consumes, the more message losses can happen due to collisions and congestions. It would also affect the operation of other protocols in the network. Therefore, a good MANET protocol should create as little traffic as possible to get a task done.

The AODV traffic overhead consists of two parts, the data forwarding traffic and routing traffic[73]. The routing traffic is the only control traffic in AODV. In VNAODV, there are 4 types of traffic generated: data forwarding traffic, routing traffic, state synchronization traffic and leader election traffic. The last three types of traffic compose the control traffic in VNAODV. Because the state synchronization traffic and leader election traffic are generated by the VNLayer, we consider them VNLayer traffic overhead.

[73] Here, we didn't measure the link layer overhead of AODV, which can be caused by address resolution (ARP) and MAC layer data packet acknowledgements and re-transmissions. Therefore, the total traffic here does not reflect the actual bandwidth use accurately.

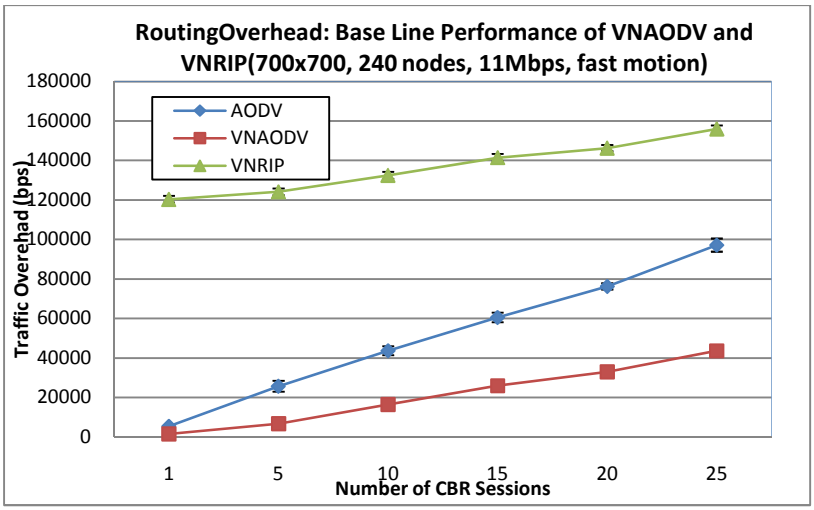

Figure 8-12 Routing Traffic Overheads of AODV, VNAODV and VNRIP

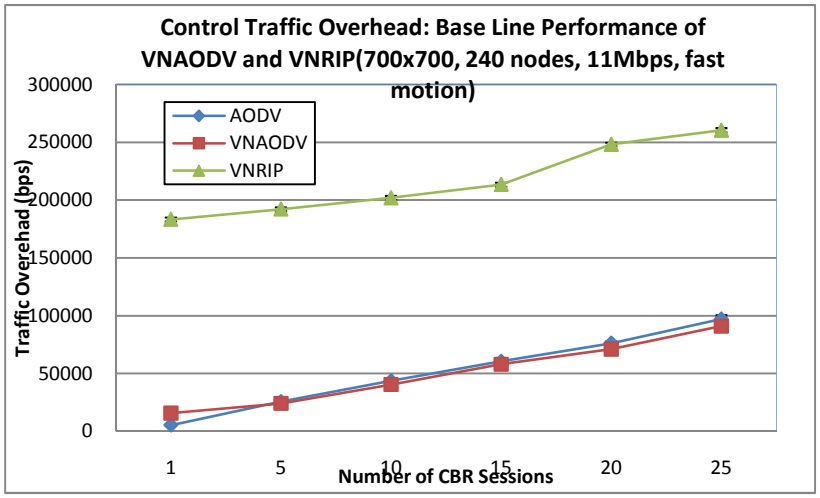

Figure 8-13 Control Traffic Overheads of AODV, VNAODV and VNRIP

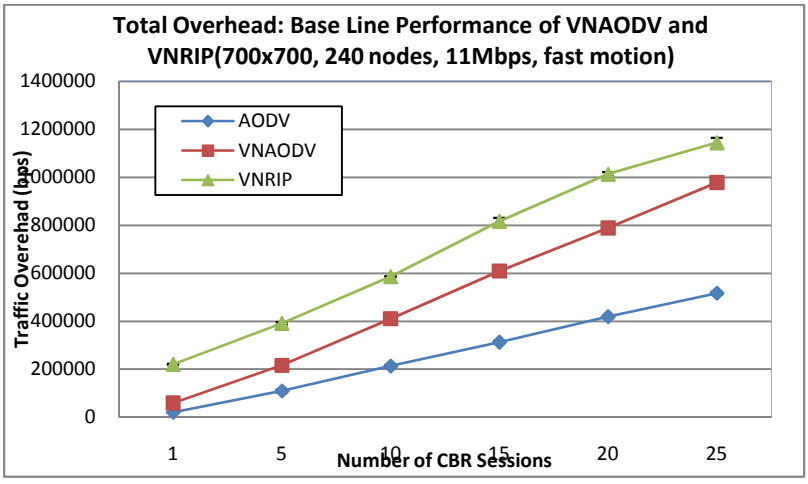

Figure 8-14 Total Traffic Overheads of AODV, VNAODV and VNRIP

Figure 8-12,

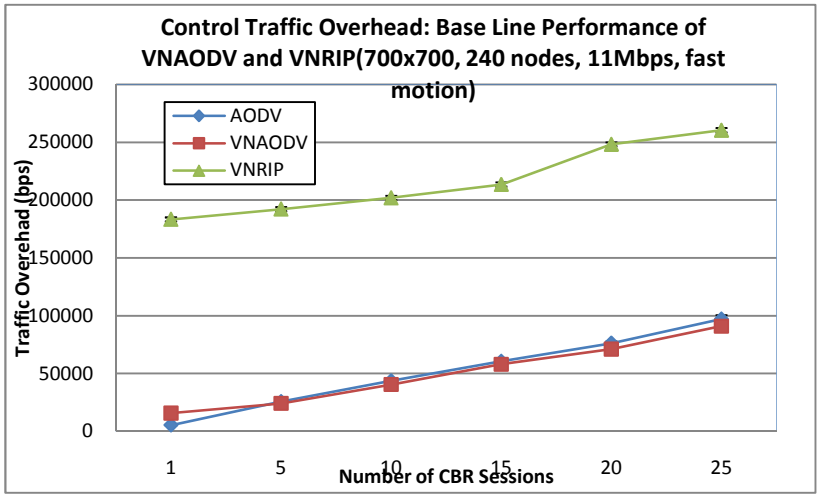

Figure 8-13 and Figure 8-14 compare the routing, control and total traffic overhead generated by AODV, VNAODV and VNRIP with various number of CBR sessions.

From Figure 8-12, we can see the routing overhead of VNRIP is the heaviest due to its proactive nature. Rather than staying constant, the routing overhead of VNRIP goes up with more CBR sessions. This is because with more CBR sessions, the number of triggered routing updates and on-demand routing updates[74] increases. Among the three protocols, VNAODV generates the least routing traffic. VNAODV outperforms VNRIP because it is an on-demand routing protocol. As expected, it also outperforms AODV due to the reduced number of entities that have to be involved in the routing operations.

The efficiency of VNAODV's routing operations doesn't come free. The VNLayer creates extra control overhead, including the leader election overhead and state synchronization overhead. Figure 8-13 shows that the total control traffic overhead of VNAODV is roughly the same as AODV's control traffic, which is equal to the routing

[74] These routing updates are generated as result of DMSG delivery failures.

overhead. This suggests that the use of VNLayer approach can reduce the total control overhead of a MANET routing protocol, if we can further reduce the leader election overhead and state synchronization overhead. Since the leader election traffic overhead is a small and constant value[75] independent of other operations of the VNLayer and the operations of specific applications, reducing the state synchronization traffic is important to further reduce the control traffic overhead.

The total traffic generated by each protocol includes the controls traffic and data forward traffic. Figure 8-14 shows that AODV generates the least total network traffic. It outperforms VNAODV because it generates less data forwarding traffic. This is first because VNAODV creates longer forwarding paths than the ones created by AODV. Another reason is that a VNLayer header is attached to every DMSG forwarded by VNAODV. This adds an extra 20 bytes of traffic per packet. Because the size of the VNLayer header is hard to be made smaller, in order to further improve the performance of VNAODV reducing the forwarding path length is important.

Since VNRIP and VNAODV generate roughly the same amount of data forwarding traffic[76], it is easy to understand that VNRIP creates the most total traffic because its control traffic is the heaviest.

[75] Around 8Kbps for the whole network, the smallest part of control traffic overhead.
[76] Not shown in the plot.

8.2.1.4 Causes of Delivery Failures

In order to improve the performance of VNAODV, we did a careful investigation on the causes of end to end DMSG delivery failures in both AODV and VNAODV. The AODV simulation trace reports three causes for DMSG drops, IFQ and NRTE, CBK.

In AODV, when a network-wide route discovery or a local route repair fails, a router drops all the DMSGs it has in its sending buffer and link layer interface queue that are destined for the unreachable destination. Such drops are reported as IFQ and NRTE failures. CBK drops happen when the link layer detects a transmission failure and the router decides not to do local route repair for it. Figure 8-15 shows the percentage of DMSG delivery failures caused by these three causes with increasing number of CBR sessions.

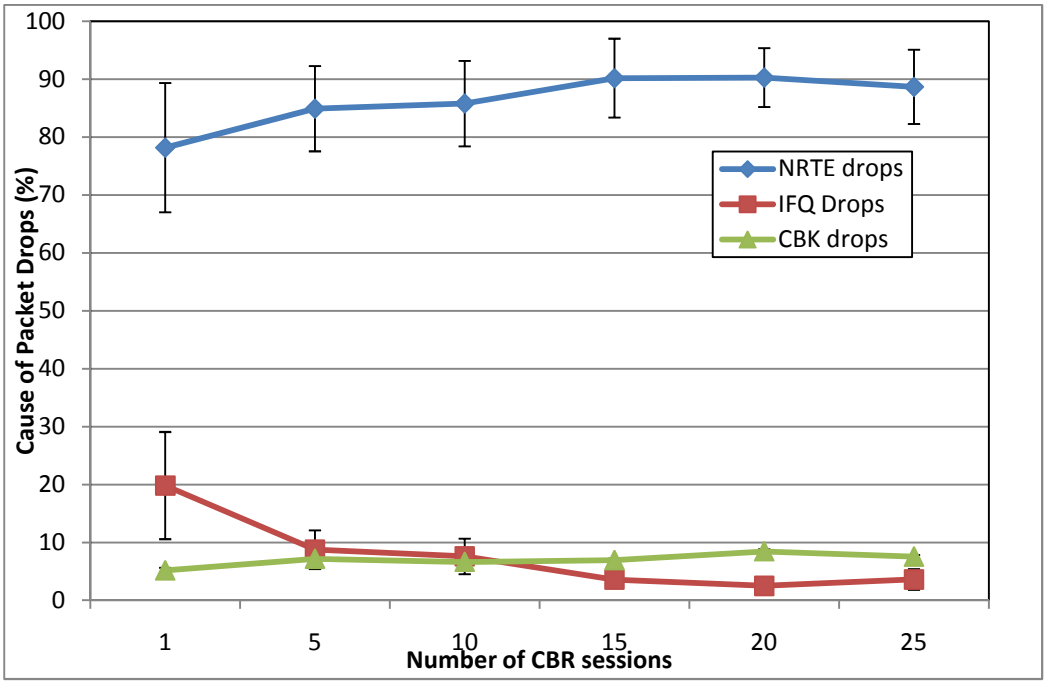

Figure 8-15 Causes of End to End Delivery Failures in AODV

From Figure 8-15 we can see less than 10 percent of the delivery failures are due to the link layer failures during DMSG transmissions. On the other hand, most of the delivery failures are due to the failures of route discoveries and local route repairs. This is because the route discovery and route repair of AODV is done by flooding RREQ messages to every physical node in a MANET. When the network is dense, these flooding based route discovery and route repair are subject to frequent failures.

Our trace analyzer can detect the following five causes of DMSG delivery failures.

1. Transmission Failure: a DMSG is sent out to the next hop vrouter. The vrouter is active but the DMSG is never received.
2. Destination Node Gone: a DMSG is delivered to its destination, but the destination client process has left its region and didn't receive the message.
3. No Route: a vrouter received a DMSG sent toward it but never forwarded the DMSG out.
4. Region Leaderless: there are physical nodes in a region. However, there is no leader present in the region. Therefore, a DMSG sent to the vrouter in the region couldn't be forwarded.
5. Region Down: a region is empty when a DMSG is forwarded to it.

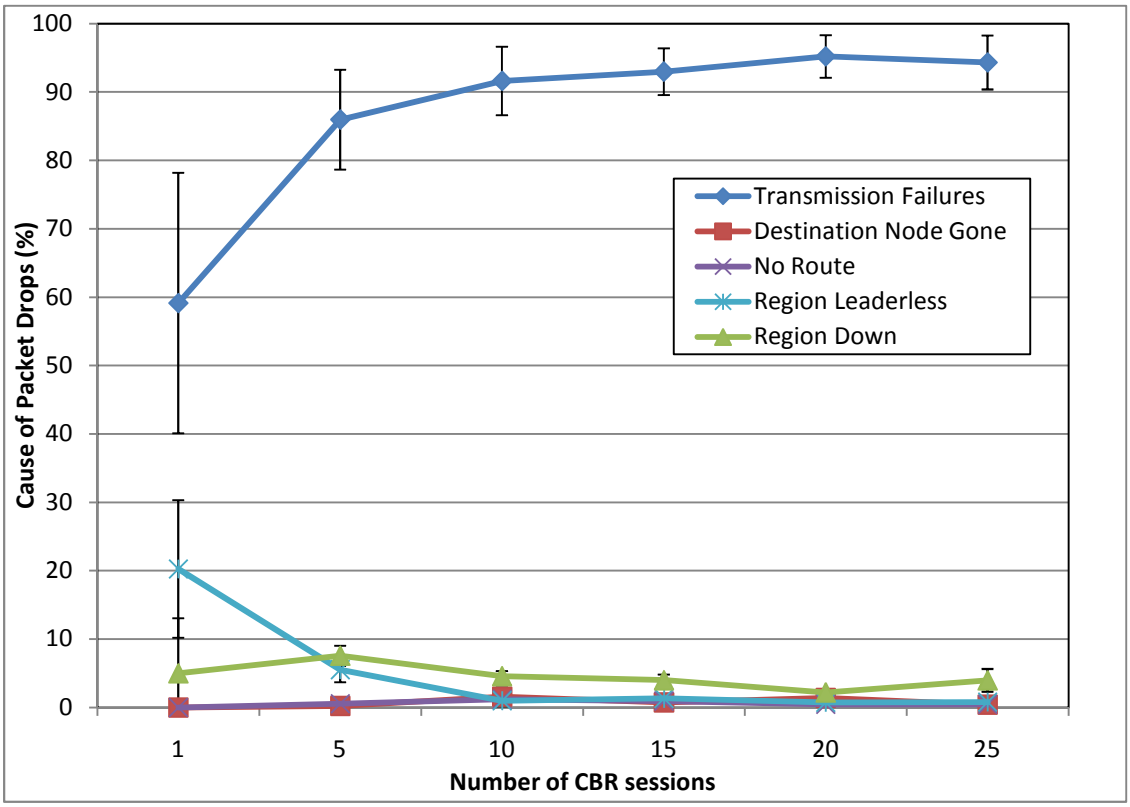

Figure 8-16 Causes of End to End DMSG Delivery Failures in VNAODV

Figure 8-16 shows the percentage of end to end DMSG delivery failures caused by the five reasons in VNAODV, with various number of CBR sessions. Unlike AODV, the leading cause of DMSG delivery failures in VNAODV routing is transmission failures. This is mainly due to the use of local broadcast in forwarding DMSGs in this VNAODV implementation. Two much smaller fractions of DMSG delivery failures are caused by leaderless or empty regions. “Destination Node Gone” causes a very small fraction of DMSG delivery failures. “No Route” also causes a very small portion of DMSG delivery failures. This suggests that the use of the VNLayer approach greatly reduced the chance that a route discovery or a route repair fails.

The simulation results here show that to improve the performance of VNAODV, reducing the length for forwarding paths, state synchronization overhead and DMSG transmission failures will be most effective.

8.2.2 The Effect of Selective State Synchronizations and Selective State Consistency Checks

The VNAODV and VNRIP implementation we evaluated in the last section already had two major VNLayer optimizations adopted. First, the two VNLayer based routing protocols don't keep the entirety of the vrouter states synchronized. Instead, only the part of a vrouter's routing table that is considered hard state are kept synchronized by the emulator nodes. In addition, a Backup Server doesn't check every incoming message from the local Server node to look for state inconsistencies. Instead, only Server messages that are generated by a vrouter are checked. These two optimizations allowed us to greatly reduce the state synchronization overhead and to greatly improve the performance of VNAODV and VNRIP.

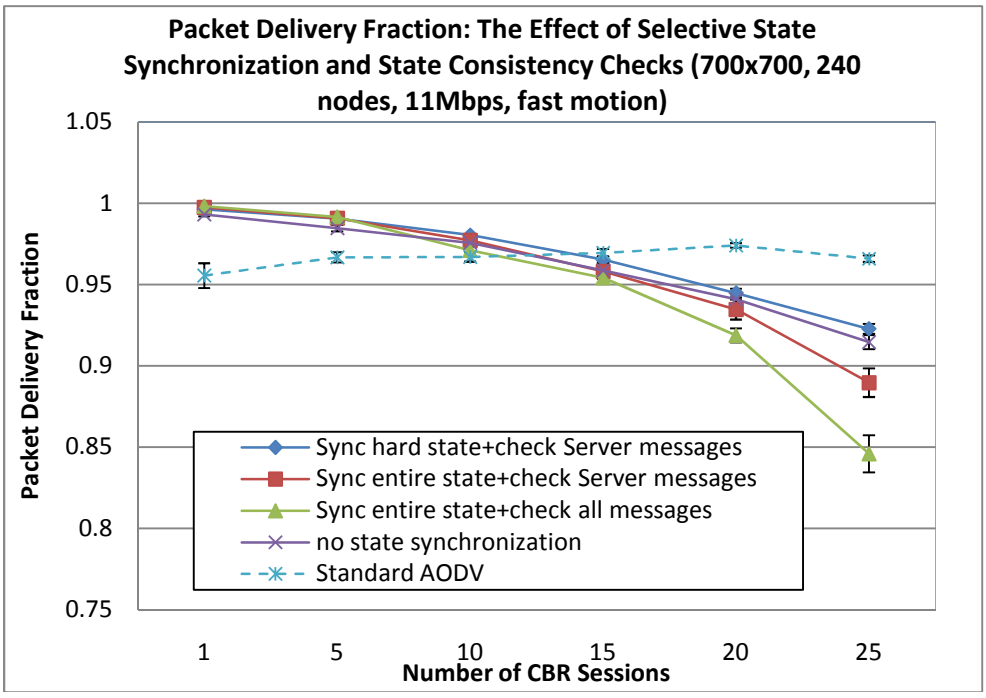

Figure 8-17 The Packet Delivery Fraction of VNAODV with selective state synchronization and selective state synchronization checks disabled

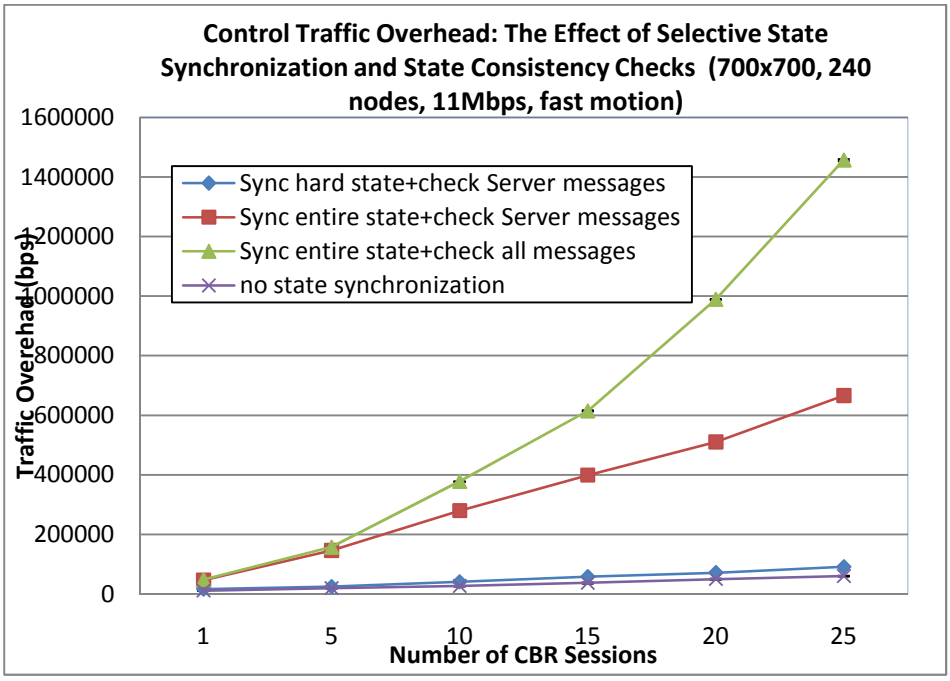

Figure 8-18 The control traffic overhead of VNAODV with selective state synchronization and selective state synchronization checks disabled

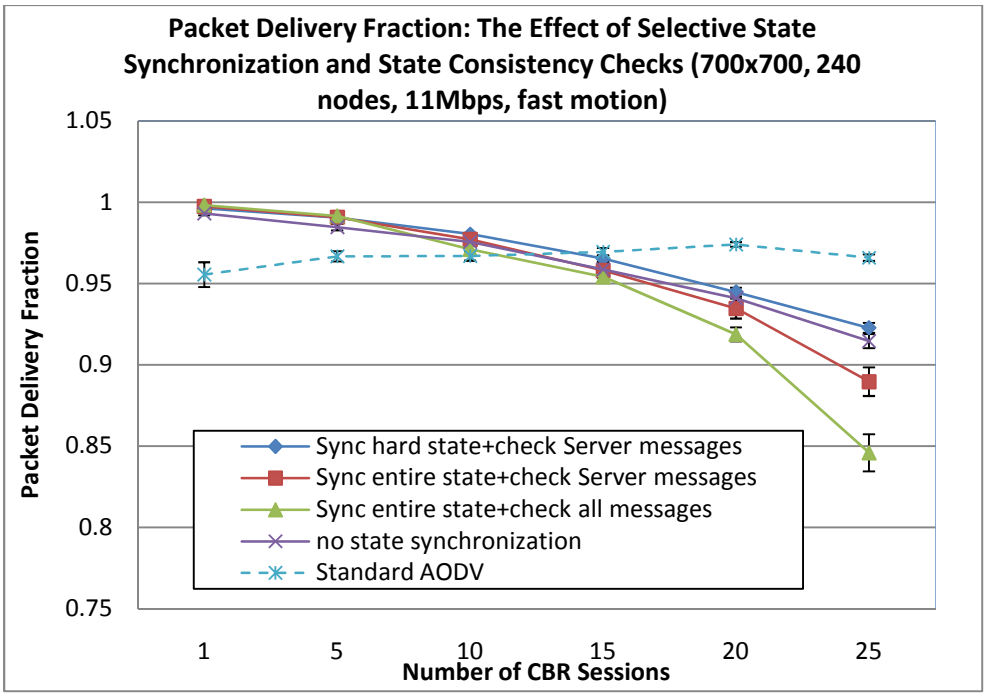

Figure 8-17 and Figure 8-18 shows what happens to the PDF and control traffic of the base line VNAODV implementation in the last section when the two optimizations are disabled. When the entire routing tables have to be synchronized and a Backup Server have to check all messages from the local Server, the control traffic becomes very heavy due to the larger state sizes and more frequent state synchronizations. We can see the PDF drops much faster with increasing number of CBR sessions.

Now, when we let a Backup Server to check only Server messages from its local Server for state inconsistencies, VNAODV's control traffic becomes much lighter and its PDF curve is much flatter. However, because state synchronizations still involve the entire application state, VNAODV in this case still generates much more control traffic than the base line implementation does.

The two figures also show what happens when state synchronization among emulator nodes is completely disabled. We can see that without state synchronization, VNAODV can actually perform better than the case when the two optimizations are not used. This interesting result suggests that for a VNLayer based MANET routing application, keeping routing table synchronized among emulator nodes is not critical to the performance. At the same time, excessive state synchronization can hurt the performance.

8.2.3 The Effect of Using Long Links

On top of the base line implementation, the next thing we did to improve the performance of VNAODV is to try to further reduce the length for the forwarding paths by using the Long Links option, which basically allows virtual nodes and client processes to communicate with each other as long as they can. Figure 8-20 shows the effect of using LL on the length of forwarding paths created by VNAODV.

Without LL, on average, the forwarding paths created by VNAODV are much longer than the forwarding paths created by AODV. With LL, the forwarding paths created by VNAODV are on average only about half a hop longer than the ones created by AODV. Considering the fact that even with LL, at the first hop, a client process still has to deliver its DMSGs to the local virtual node for routing, often resulting in an unnecessary extra hop in the forwarding path, the paths created by VNAODV with LL are already no worse than the ones created by AODV.

Shortened forwarding paths lead to less data forwarding traffic, fewer packet collisions and better PDF. Figure 8-20 verifies that with LL, the PDF of VNAODV is greatly

improved. Now, VNAODV performs no worse than AODV up to 25 CBR sessions. Figure 8-19 shows that with LL, the delivery latency of VNAODV is also greatly reduced as a result of the shortened forwarding paths. Finally, Figure 8-22 shows that with LL, the total amount of traffic generated by VNAODV is also reduced. This is because the shortened forwarding paths reduce both the data forwarding traffic and the routing traffic due to less frequent forwarding failures.

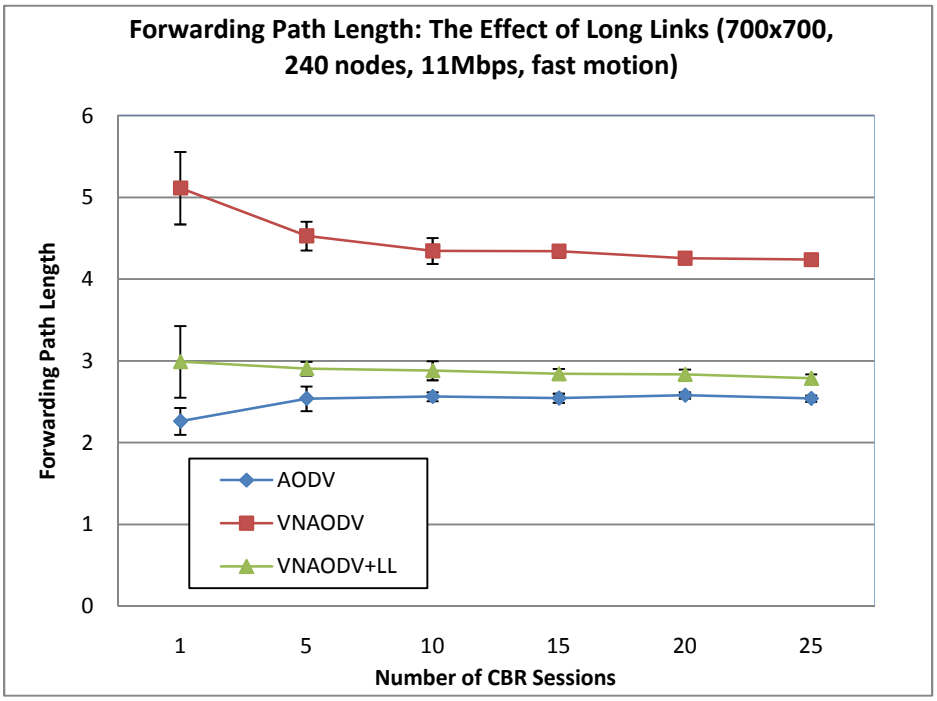

Figure 8-19 The Effect of Using Long Links on the Forwarding Path Length of VNAODV

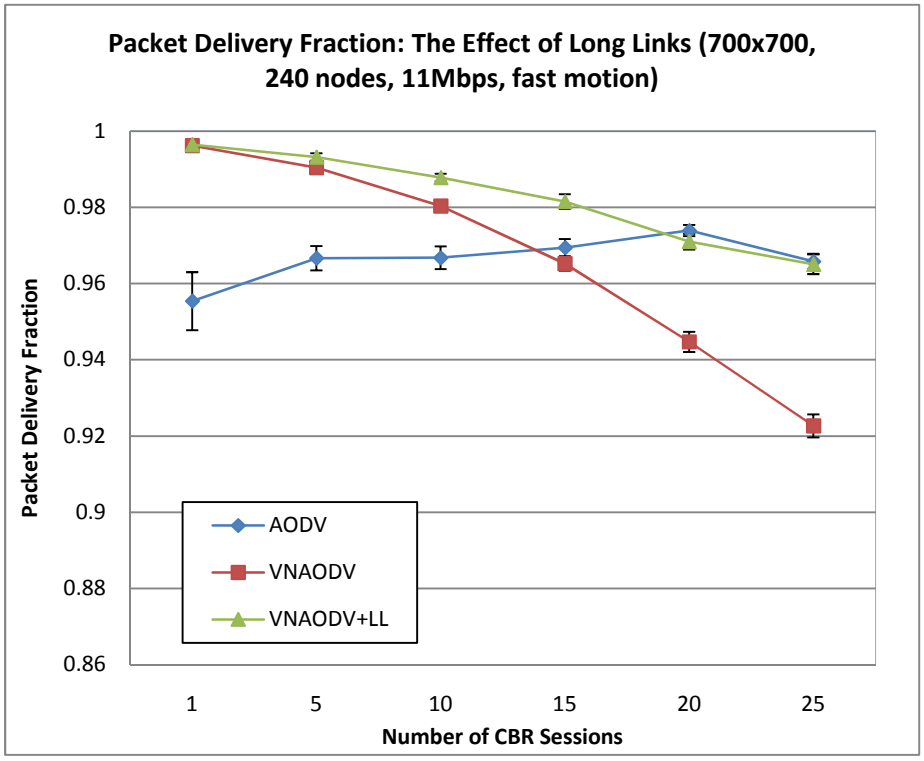

Figure 8-20 The Effect of using Long Links on VNAODV's PDF

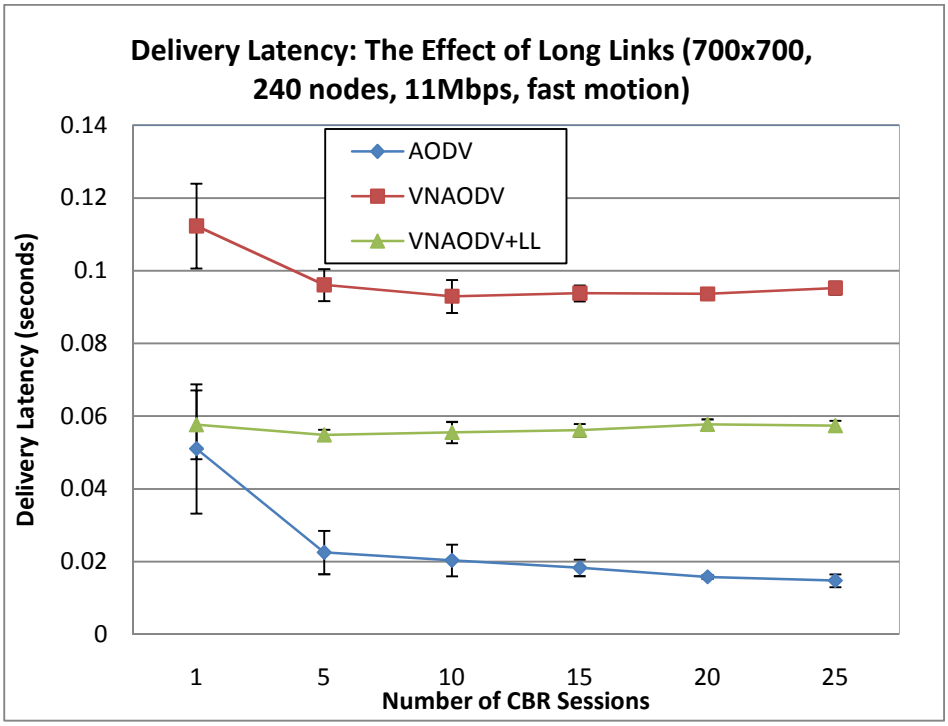

Figure 8-21 The Effect of using Long Links on the Delivery Latency of VNAODV

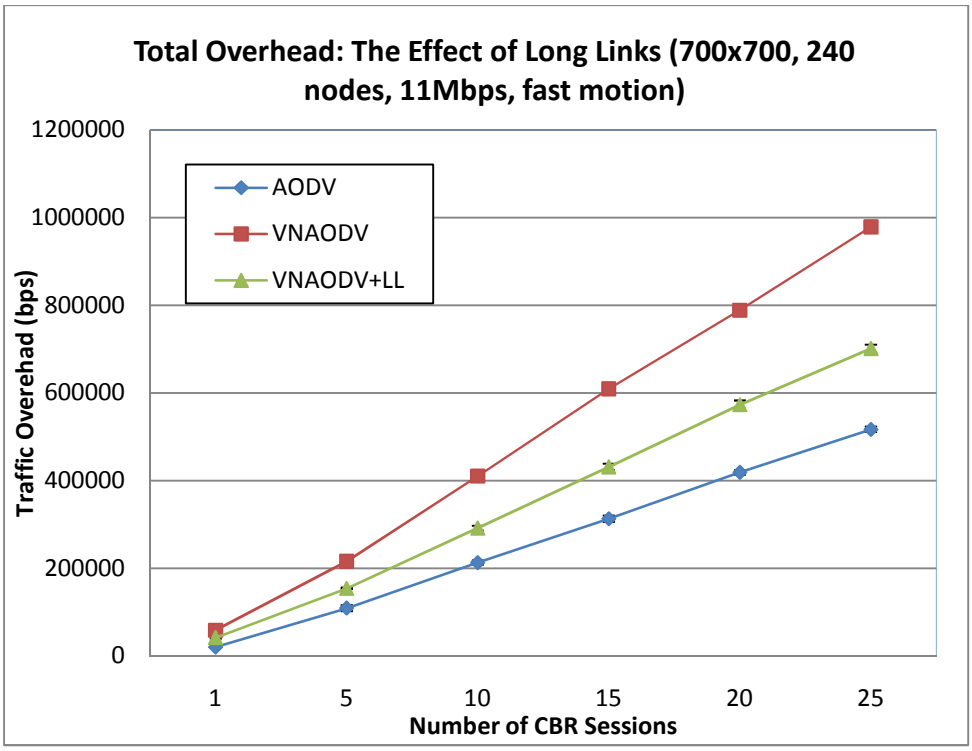

Figure 8-22 The Effect of using Long Links on the Total Traffic Overhead of VNAODV

8.2.4 The Effect of Using Directed Broadcast

Now we have improved the performance of VNAODV by reducing the state synchronization overhead and forwarding path length. The third major direction we took on to improve the performance of VNAODV is to reduce the transmission failures at the link layer. Based on the observation that the local broadcast based data transmission causes the majority of packet delivery failures the capabilities provided by the extended link layer model, we designed the optimization that uses Directed Broadcast for data transmissions. With Directed Broadcast, a unicast destination address is used to transmit a packet whenever the leader of the next hop region is known or the next hop is a client process.

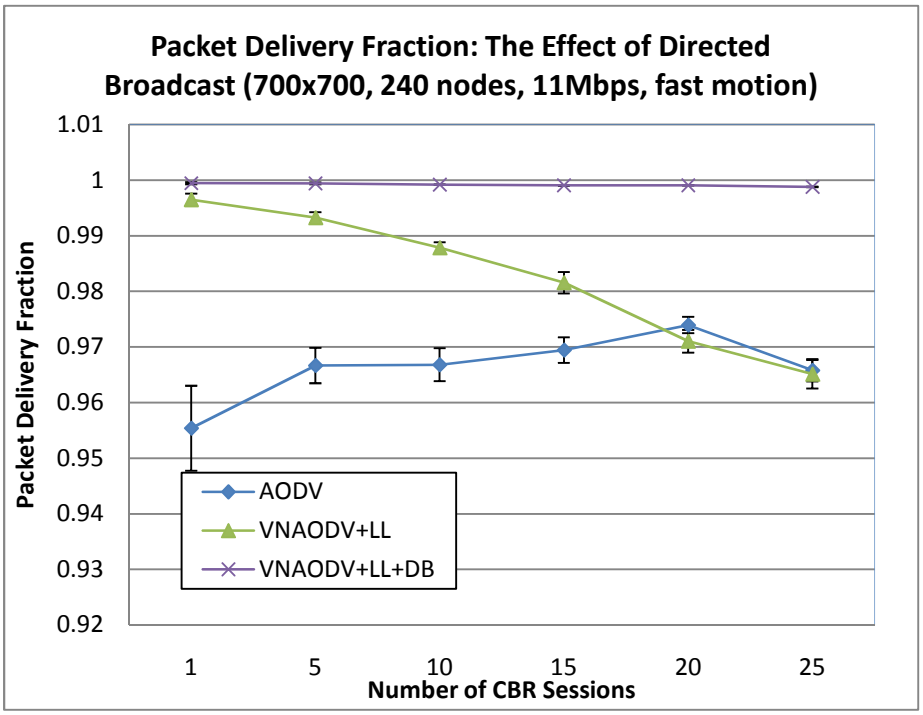

Figure 8-23 The Effect of using Directed Broadcast on the PDF of VNAODV

Figure 8-23 shows the effect of using Directed Broadcast on top of the base line VNAODV implementation optimized with the LL option. The use of Directed Broadcast" for data transmissions drastically improved the PDF of VNAODV and turned the PDF curve flat. Since the use of Directed Broadcast is just a different way of transmitting packets at the link layer, it doesn't increase or reduce the delivery latency and traffic overhead[77]. This is verified by our simulation results.

Further investigations show that with Directed Broadcast used, the percentage of DMSG delivery failures caused by data transmission failures dropped from about 84% to about 60%, on average. This indicates that although data transmission failure is still the leading

[77] Extra traffic is generated by the link layer for packet acknowledgements. Not measured in simulations.

cause of DMSG delivery failures, the number of such failures has been greatly reduced due to the use of Directed Broadcast.

Now we have a version of VNLayer based routing protocol that clearly outperforms AODV. However, this comes at the cost of requiring every physical node in a MANET to support promiscuous mode and a more complicated VNLayer implementation that can keep track of the address of region leaders.

8.2.5 Route Stability Brought by the VNLayer Approach

Since VNAODV is a clustering based routing protocol, it is easy to understand that it performs better than AODV because the number of entities that have to be involved in routing is reduced. However, on top of that, we also expected that the VNLayer approach can improve the performance of applications with a stable topology among virtual nodes (cluster heads) due to its fixed region settings. In this section, we find out whether this is the case for VNAODV.

For a routing application, a stable topology among vrouters translates into more reliable forwarding paths. In this research, we use the number of route discoveries[78] done by each protocol to infer the reliability of forwarding paths created by AODV and VNAODV. Because each route discovery could involve a large number of routers, we want the number of network-wide route discoveries to be as small as possible.

[78] This includes local route repairs but doesn't include the 1 hop Local Connectivity Checks in VNAODV.

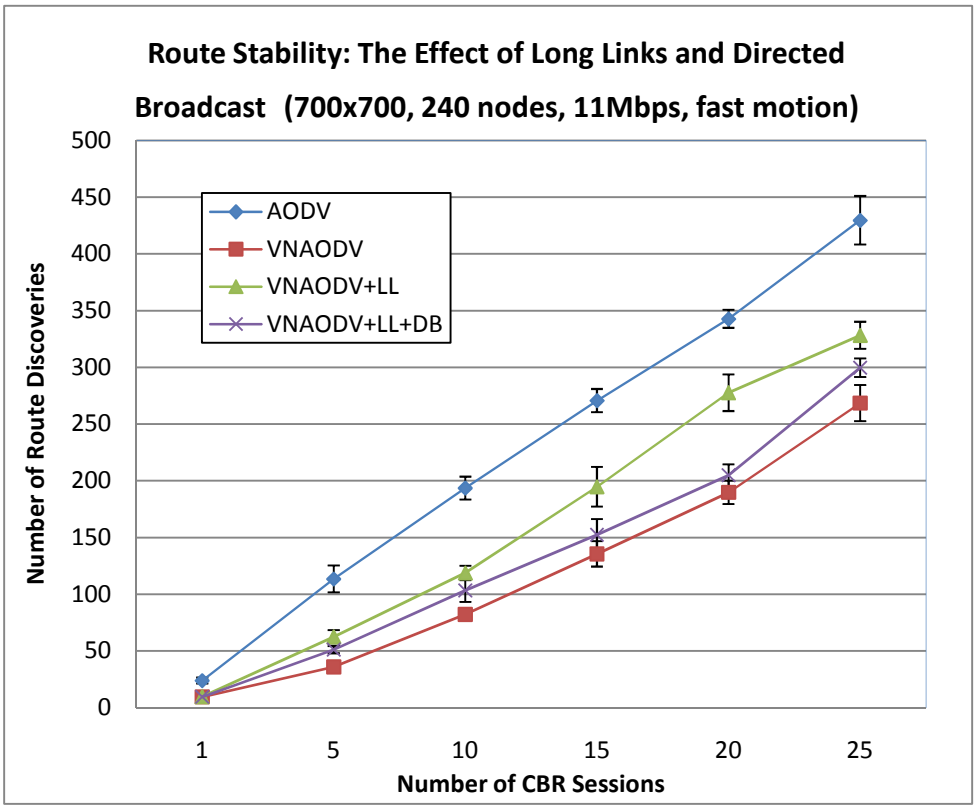

Figure 8-24 Route Discoveries/Repairs done by AODV and VNAODV

Figure 8-24 shows the route discoveries and local route repairs done by AODV and VNAODV with various number of CBR sessions. A separate curve is drawn for the three major versions of VNAODV we have seen so far. We can see that without using Long Links and Directed Broadcast, VNAODV performs much smaller number of route discoveries than AODV at every data point, suggesting that the forwarding paths created by this version of VNAODV breaks less frequently than the ones created by AODV.

As expected, the use of Long Links made the forwarding paths less stable because long links are not as reliable as the links created between immediate neighbor regions. The use of Directed Broadcast reduces the number of route discoveries. This is not because using DB improves the reliability of forwarding paths. It is because DB reduces data

transmission failures and in turn reduces the chance that a link is falsely determined as broken as a result of unacknowledged[79] data transmissions.

8.2.5.1 CBR Sessions with Static Endpoints

In the simulations we have presented so far, the endpoints in each CBR session are set to move at random[80]. Therefore, when either the source client process or the destination client process leaves one region, VNAODV may have to do a route repair or even a network wide route discovery. To further verify that the VNLayer approach can indeed improve the stability of forwarding paths, we repeated the simulations above with modified CBR sessions such that the source and destination client processes for each CBR session stays at the same geographical location throughout a simulation.

[79] Either explicitly or implicitly.
[80] Using the random waypoint model.

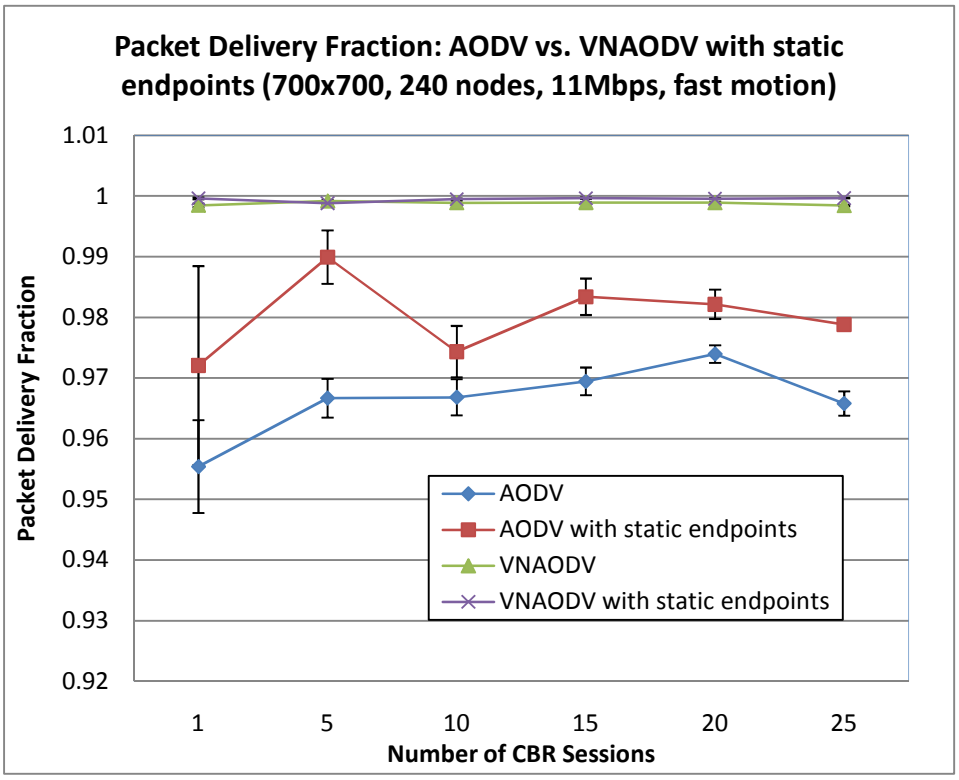

Figure 8-25 PDF of AODV and VNAODV with Static Endpoints in CBR sessions

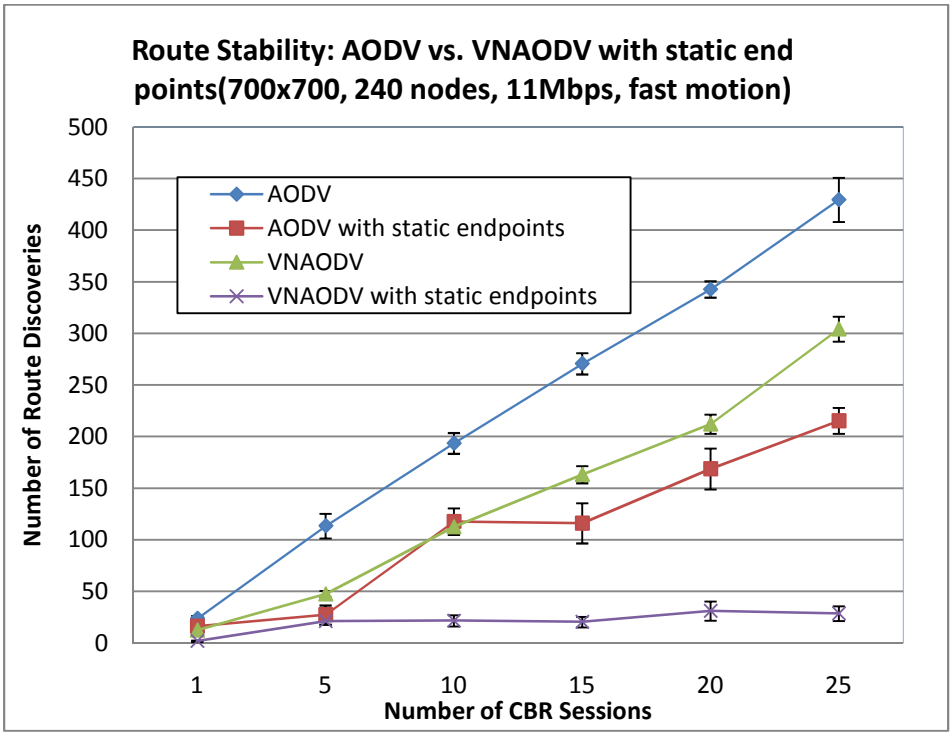

Figure 8-26 Route Stability of AODV & VNAODV with Static Endpoints in CBR sessions

In this case, the forwarding paths created by VNAODV are expected to be much more stable than the ones created by AODV. This is because a forwarding path used by VNAODV for a CBR sessions is a sequence of vrouters in between the static source and destination client processes. As long as this sequence of vrouters stays up. The same forwarding path can be used in face of mobility of the physical nodes emulating the vrouters. On the other hand, in AODV, the mobility of any router on a forwarding path can lead to a route repair or a network-wide route discovery.

Figure 8-25 shows that the PDF of AODV and VNAODV both improve when the endpoints of the CBR sessions are changed from mobile to static. Figure 8-26 shows that in the static endpoints cases, the advantage VNAODV has over AODV on route stability is much greater than the original cases with mobile endpoints. This verifies our intuition.

8.2.6 The Value of State Replication

An important capability provided by the VNLayer is state replication. As we have discussed, state replication allows a virtual node to maintain persistent application state in face of node mobility. However, this comes at the cost of extra control overhead. We have already shown in section 8.2.2 that doing no state synchronization at all in VNAODV is better than synchronizing the entire application state and check every local Server message for state inconsistencies. However, in the case without state synchronization, there are still Backup Servers in each region that handles application messages using and updating their own copy of the application state. This is still a form of state replication. Now, the question is, to what extent do we need to rely on state replication? What happens if we don’t use Backup Servers at all?

In addition, there are two subtypes of state synchronizations. A MOV-SYNC happens when a physical node move into a different region and becomes a Backup Server. A MSG-SYNCs happens when state consistency checks on incoming messages detects a state inconsistency. Intuitively, a MOV-SYNC is more important than a MSG-SYC because a Backup Server newly arrived at a region relies on the MOV-SYNC to get a copy of the application state. On the hand, MSG-SYNCs are used to patch up the state of Backup Server. The question is, can we turn off MSG-SYNC so that we can further reduce the state synchronization overhead.

Figure 8-27 and Figure 8-28 show the PDF and route stability of a VNAODV implementation[81] when state replication is completely turned off, when MSG-SYNC is turned off or when there is not state synchronization.

It is hard to tell whether turning off just MSG-SYNC hurt either the PDF or route stability, because the curves for the two cases cross each other. This is because each time a physical node moves into a new region and becomes a Backup Server, it can still receive a copy of the Server's state with a MOV-SYNC. Backup Servers still process incoming messages and update their state except that they don't check messages from the Server for state inconsistencies. In addition, with MSG-SYNCs disabled, the reduction on state synchronization traffic reduces transmission failures and helps improve the packet delivery fraction.

[81] This version of VNAODV is the base line implementation with the long link option used.

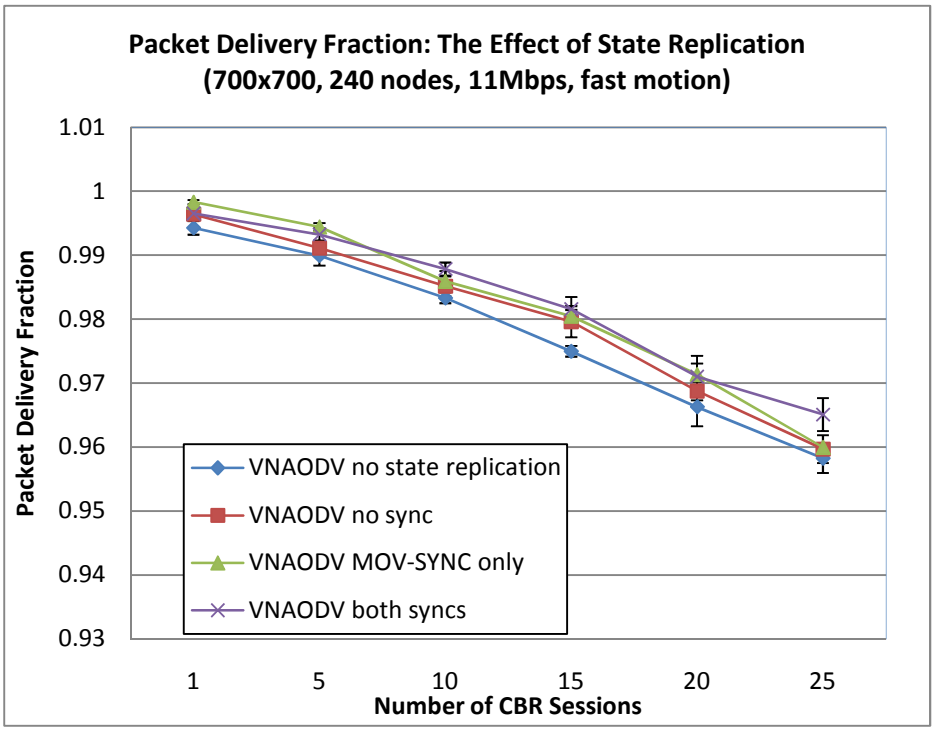

Figure 8-27 PDF of VNAODV with Different State Sync Modes

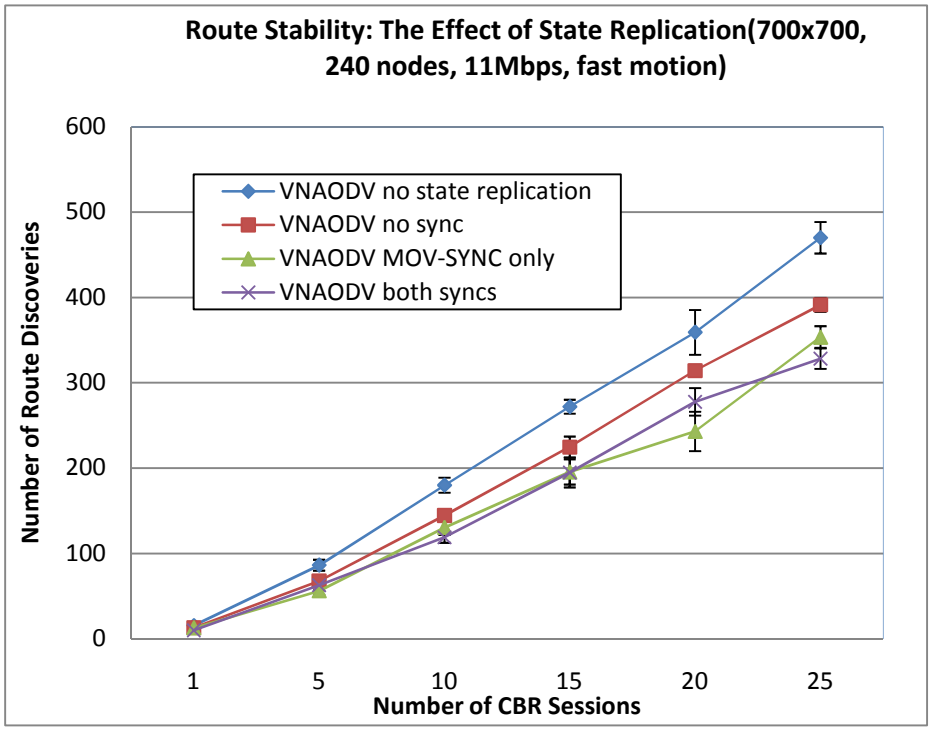

Figure 8-28 Route Stability of VNAODV with Different State Sync Modes

Turning off both MSG-SYNC and MOV-SYNC saves all state synchronization traffic. However, from the figures we can see this clearly hurts the PDF and route stability. This is because each time a physical node enters a region and become a Backup Server; it doesn't get any part of the application state from the Server. When the Backup Server becomes the Server of the region, it may not have a route for a DMSG and extra routing operations have to be done.

Turning off state replication completely hurts the PDF and route reliability the most. In this case, Backup Server nodes ignore every incoming application message. Every time the Server node of a region leaves, a session using the region as a forwarding hop has to fix its route by either local route repairs or network-wide route discoveries. Therefore, a lot more route discoveries are needed in this case.

However, even in the last case, the PDF of VNAODV is still not much worse than the first case. This shows that state replication is not critical to the performance of VNAODV, which is designed to be resilient to route changes. As we have seen, reducing transmission failures caused by heavy control traffic overhead, long forwarding paths and local broadcast matters more to the performance of VNAODV.

8.2.7 Effect of other Optimizations at the VNLayer

8.2.7.1 The Powerful Emulator Option

As explained in section 0, the Powerful Emulator option can be used to VNAODV to further shorten the forwarding paths by allowing an emulator node hosting a client process to work as a router independently.

Figure 8-29 and Figure 8-30 and Figure 8-31 show what happens to the PDF, average forwarding path length and delivery latency of VNAODV[82] when the Powerful Emulator (PC) option is used. To see the effect of shortened forwarding paths, the figures also show another curve for the VNAODV implementation with only the Directed Broadcast option used. We can see with PC used, the PDF curve becomes even flatter. This is because in this 8 by 8 region network, saving 1 forwarding hop can cut down the data forwarding traffic overhead a lot. With lower traffic overhead and fewer forwarding hops, the chance that transmission failure happens is lower.

With PC, a client process no longer has to communicate with the local server for routing service. From Figure 8-30, we can see that the average length of the paths created by VNAODV is even shorter than the paths created by AODV. This is probably due to the better stability of the forwarding paths created by VNAODV. Since the average forwarding path was only about 3 hops, with almost one extra hop saved on average, PC can save us one third of the data forwarding traffic.

[82] An implementation that is equipped with both the Long Link (LL) and Directed Broadcast(DB) options. This is also the implementation that gives us the best performance so far.

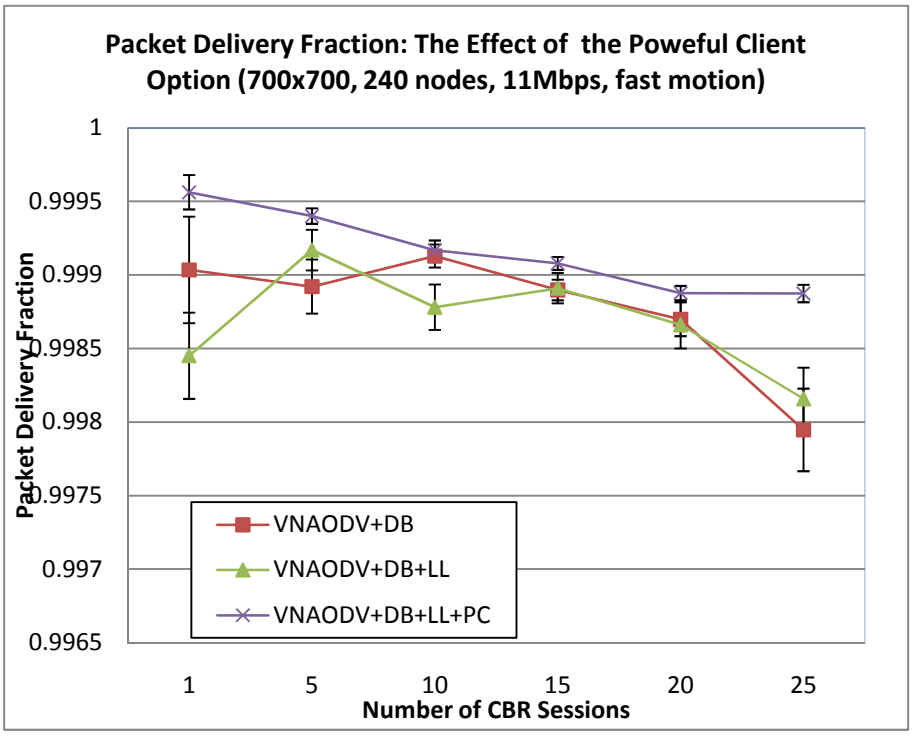

Figure 8-29 The Effect of Using the Powerful Emulator Option on the PDF of VNAODV

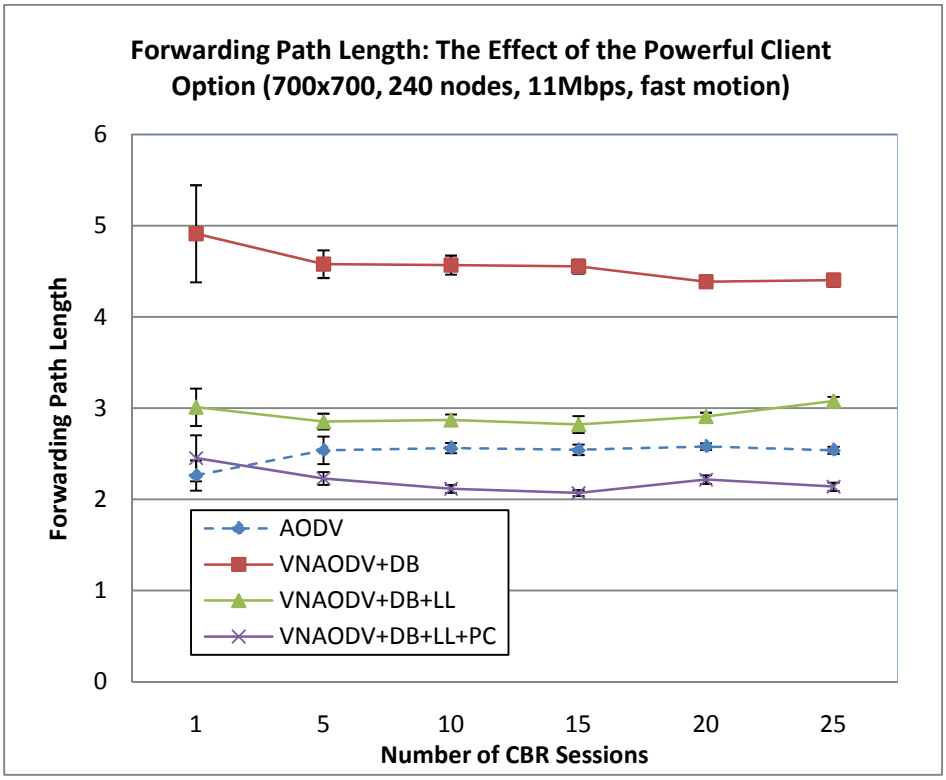

Figure 8-30 The Effect of Using the PC on the forwarding path length of VNAODV

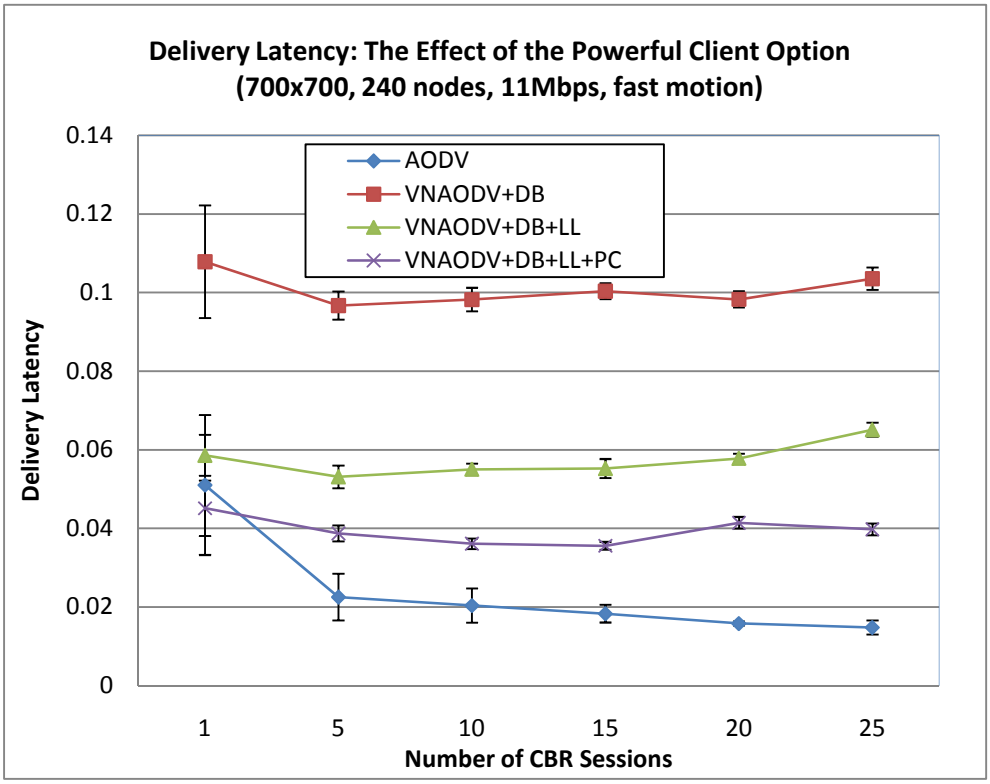

Figure 8-31 The Effect of Using the PC Option on the delivery latency of VNAODV

With forwarding paths shortened by PC, the end to end delivery latency of VNAODV is even lower. Now, with small number of CBR sessions, the delivery latency of VNAODV can be shorter than AODV. However, since VNAODV introduces the total ordering delay and can't take advantage of reverse routes learned from other route discoveries to shorten route set up times. When there are 5 CBR sessions or more, the delivery latency of VNAODV is still longer than AODV's.

Another thing we can see here is that the DB option by itself can bring most part of the improvement on the PDF VNAODV. On top of that, adding the Long Link option shortens the forwarding paths and forwarding delay, but not much on the PDF of VNAODV.

8.2.7.2 Control Over the Number of Emulator Nodes in a Region

When a MANET is dense, using every physical node as an emulator node is not only unnecessary but also costly because the more Backup Servers there is in a region, the more state synchronizations will be needed. As introduced in section 4.7.5 , our VNLayer implementation can adjust the number of Backup Servers in a region by changing a threshold value, which controls the chance a non-leader node sets itself as pure client. When the threshold is set to 1000, every non-leader node becomes a Backup Server. When the threshold is set to 0, there will be no Backup Servers. Figure 8-32 and Figure 8-33 show the effect of using different thresholds on the PDF and control overhead of VNAODV. The VNAODV implementation here is equipped with both the Directed Broadcast option and the long link option.

From the figures, we can see with threshold 750 is used, the control overhead of VNAODV is much lower than the case with threshold 1000 (all non-leaders become Backup Servers). This is due to the number of state synchronization messages is reduced when the number of Backup Servers is reduced. With the reduced control traffic, we can see the PDF of VNAODV is also better than the case with threshold 1000.

However, when we further reduce the threshold to 500, the control overhead actually goes up a little. This is because with fewer Backup Nodes, the number of sync messages actually increases a little due to the reduced state consistency among emulator nodes. In this case, the PDF of VNAODV is still good, though. When the threshold is reduced to 250, VNAODV performs worse both in terms of PDF. This is because there is not enough Backup Servers in the regions anymore and routes break more frequently.

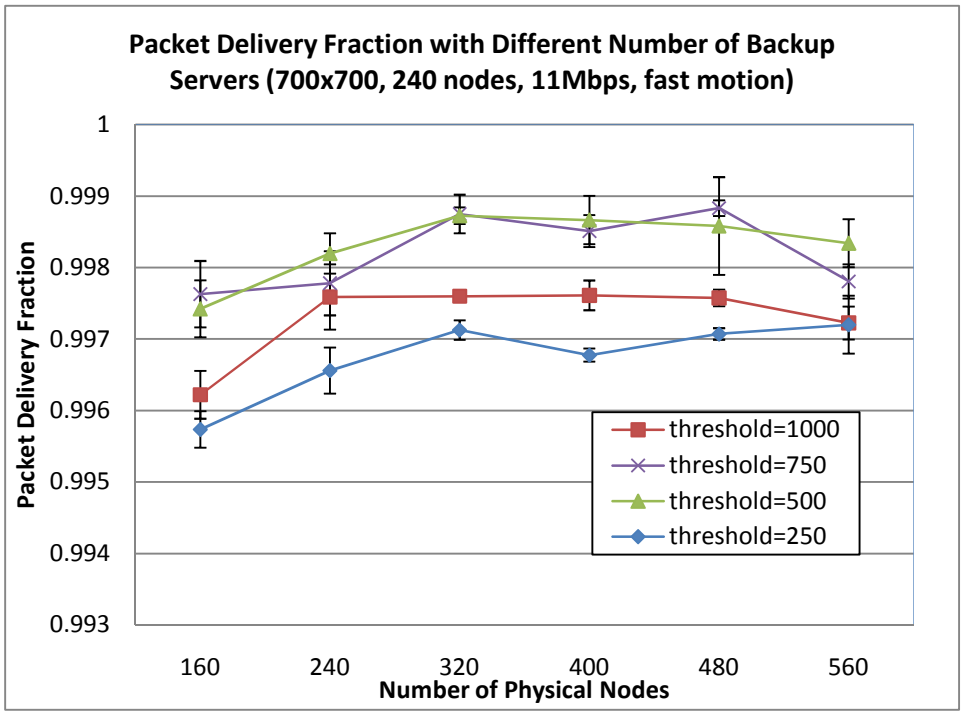

Figure 8-32 The Effect of Reducing the Number of Backup Servers in a region on the PDF of VNAODV

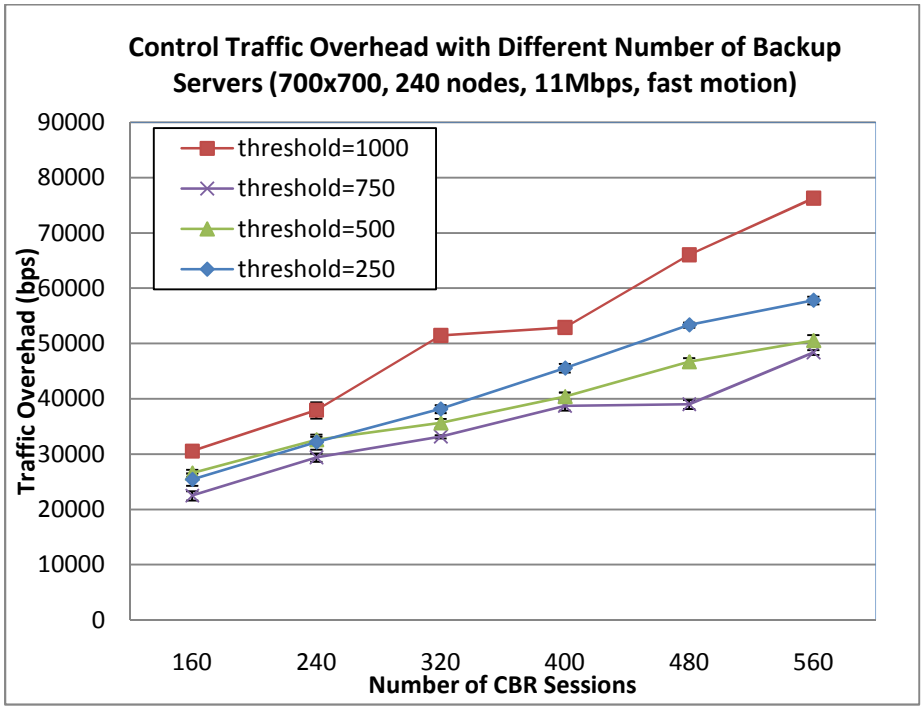

Figure 8-33 The Effect of Reducing the Number of Backup Servers in a region on the Control Overhead of VNAODV

8.2.8 The Effect of Application Layer Optimizations

8.2.8.1 Local Recovery

As we have pointed out, data transmission failures at the link layer is the lead cause of end to end DMSG delivery failures. Local Recovery is an option we designed at the application layer of VNAODV to reduce the impact of transmission failures on the performance of VNAODV. Therefore, Local Recovery and Directed Broadcast serve the same purpose. However, they are used under different situations. Directed Broadcast can be used when the address of the recipient of a packet is known and when promiscuous mode can be used on all physical nodes. When either condition can't be satisfied[83], Local Recovery can be used to recover DMSGs that are suspected of being lost.

Figure 8-34 and Figure 8-35 compares the impact of Local Recovery and Directed Broadcast on the PDF and control overhead of VNAODV. Here, the VNAODV implementation uses the Long Links option. From the plots, we can see using DB by itself does a better job than using LR itself on both parameters. This is because LR can only recover a portion of the DMSGs lost and doing so comes with unnecessary retransmissions of DMSGs that are falsely determined as being lost.

When both DB and LR are used[84], our simulation results showed that VNAODV performs at little better than the case when only DB is used. Actually, LR can also be implemented at the VNLayer for packets that can't be delivered by DB. This way, applications don't have to do Local Recovery for application messages anymore.

[83] One example is that when the address of a next hop region's leader is unknown, local broadcast has to be used. In this case, LR can be used to reduce transmission failures.
[84] LR only works on DMSGs that are sent by local broadcast.

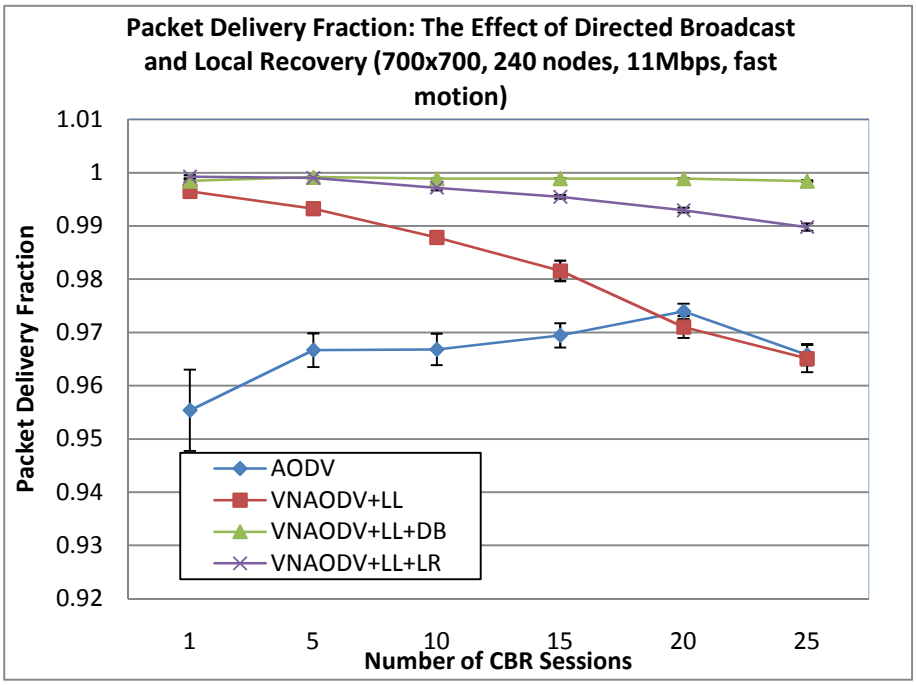

Figure 8-34 The Effect of Directed Broadcast and Local Recovery on the PDF of VNAODV

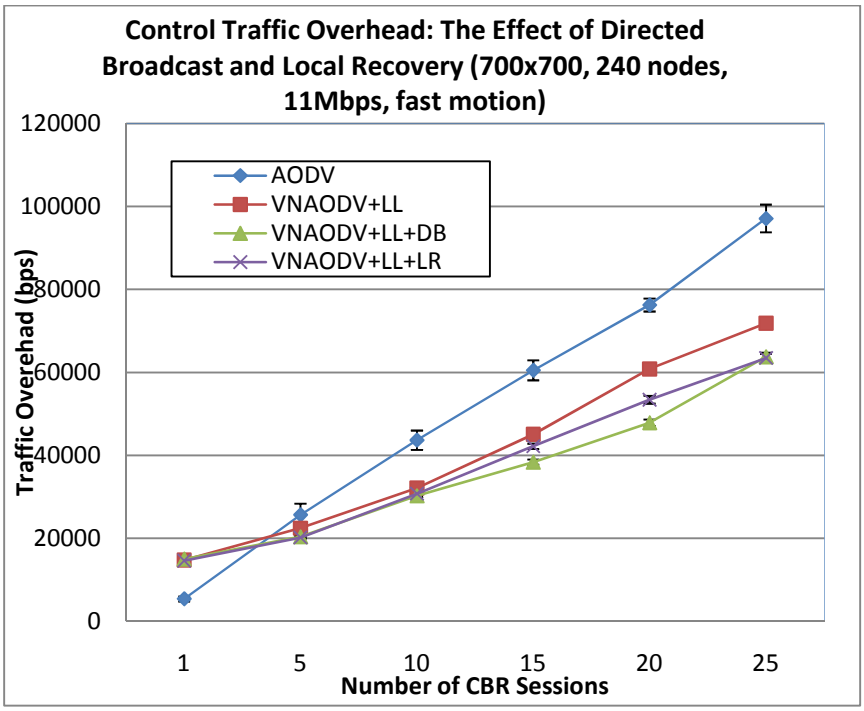

Figure 8-35 The Effect of Directed Broadcast and Local Recovery on control overhead of VNAODV

8.2.8.2 Route Correction by Destination

Route Correction (RC) is a simple optimization we did at the application layer of VNAODV so that when a client process receives a DMSG destined for it but not destined for its region, it sends out an RREP message reminding nearby vrouters about its current location. With route correction, we expect VNAODV to respond to the mobility of destination client process better. Figure 8-36, Figure 8-37 and Figure 8-38 show the effect of Route Correction on the PDF, route stability and control overhead of VNAODV. Here, the VNAODV implementation is equipped with both DB and LL. From the plots, we can see that RC doesn't improve the PDF (it is already very good), it does reduce the number of route discoveries that have to be done and it also reduces the delivery latency.

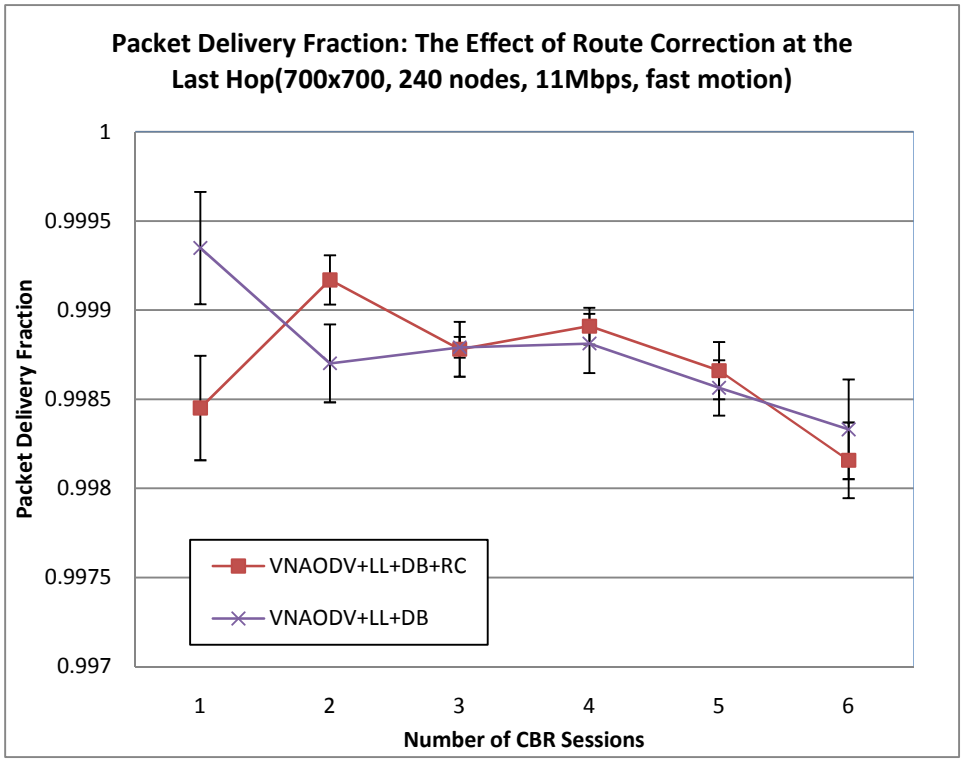

Figure 8-36 The Effect of Route Correction on the PDF of VNAODV

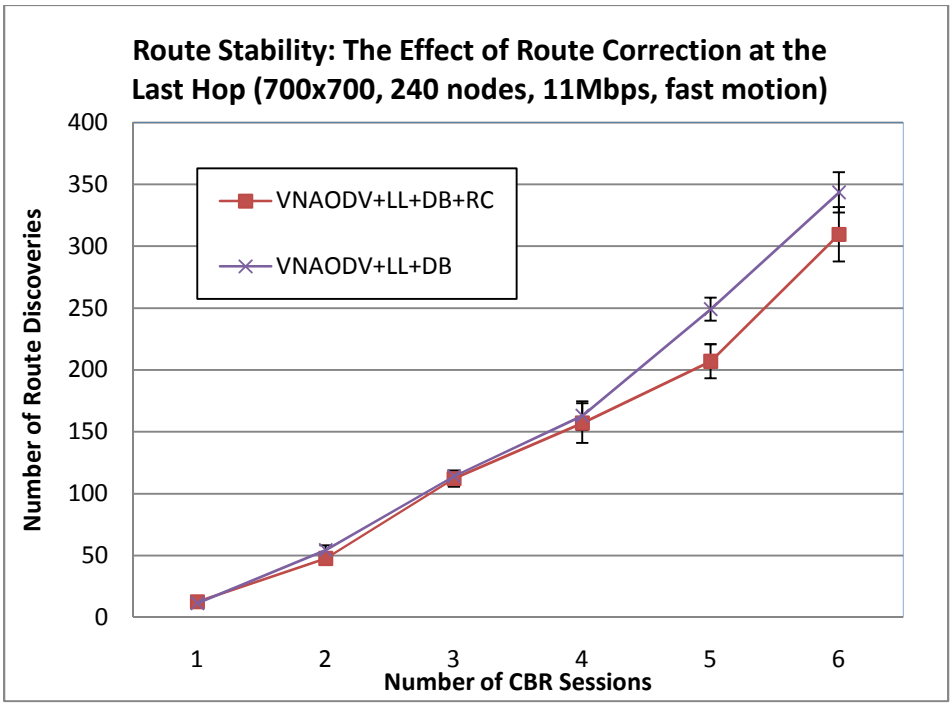

Figure 8-37 The Effect of Route Correction on the route stability of VNAODV

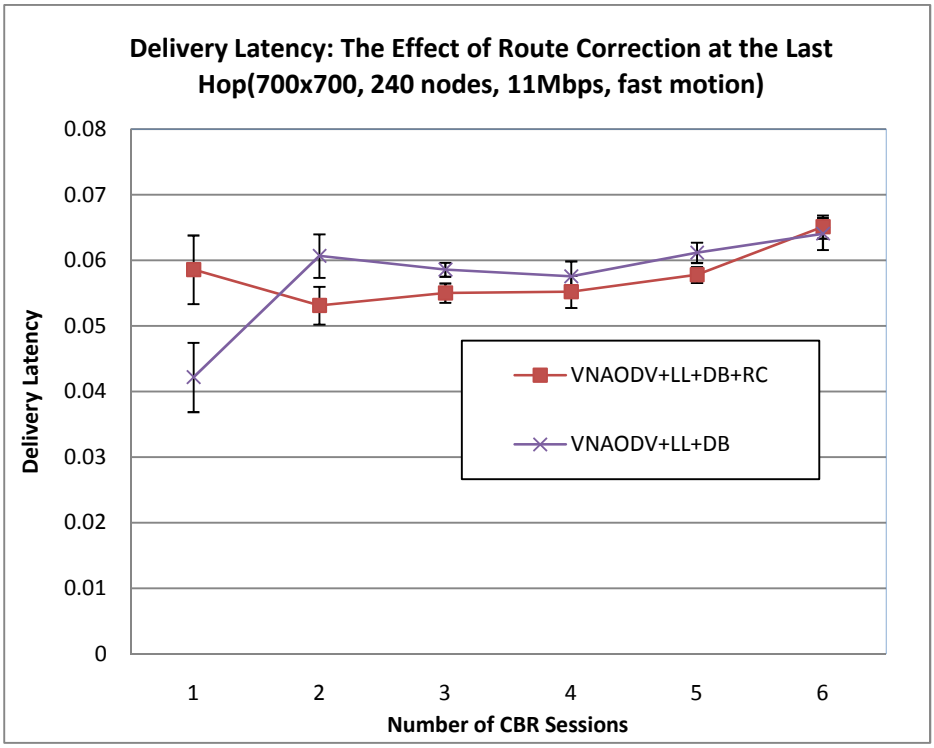

Figure 8-38 The Effect of Route Correction on the delivery latency of VNAODV

8.2.9 Different Node Motion Rates and Different Node Densities

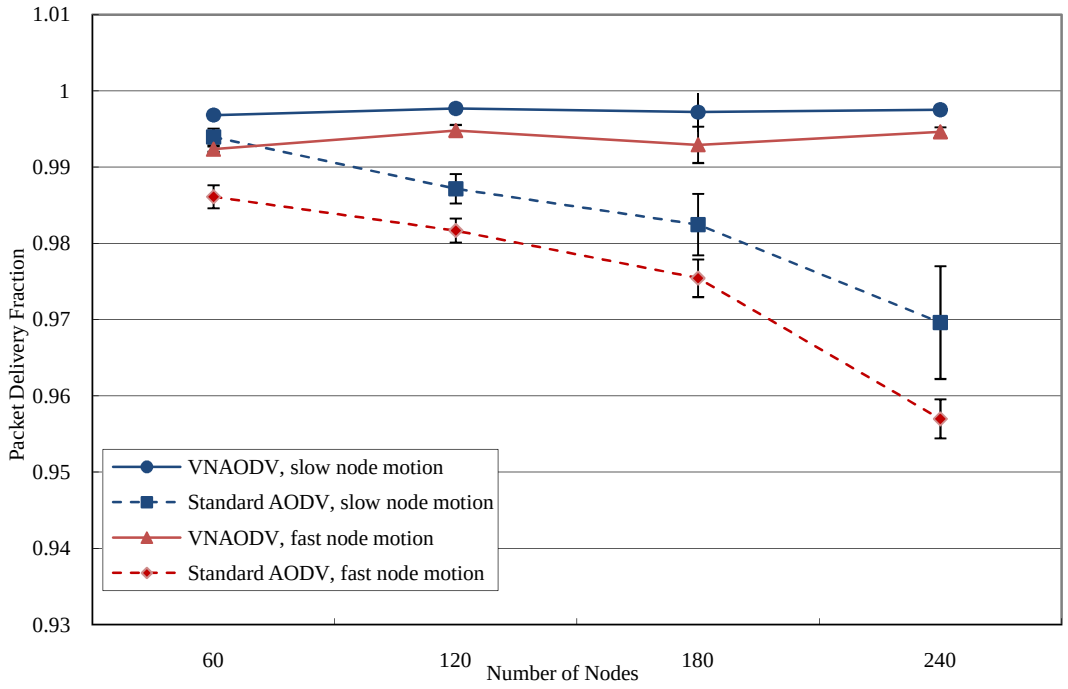

Figure 8-39 The PDF of VNAODV with different node densities and node motion rates

Figure 8-39 compares the delivery ratio of VNAODV and AODV with different node densities and node motion rates (fast mode and slow mode) with 15 CBR sessions. With increased network density, each route discovery in AODV involves heavier flooding more RREQ messages, resulting in more frequent discovery failures. Therefore, its PDF drops fast. With low network density, VNAODV doesn't much advantage over AODV. We can see it performs even a little worse due to the lack of backup servers and more frequent virtual node failures. As the network gets denser, VNAODV outperforms AODV more and more because the number of vrouters in the network is bounded by the number of regions. This verifies that VNAODV scales better than AODV.

As expected, AODV and VNAODV perform better with slower node motion rates. Figure 8 shows that VNAODV is less sensitive to mobility rate increase than AODV, due to the more stable forwarding paths created by VNAODV.

8.2.10 Different Network Sizes

The network size we used in the simulations so far is 700m x 700m, which is quite small. On average, a data packet delivery only takes a little over 2 hops. To further validate the results, we run the simulations on a large network with more mobile nodes. Now, the network covers a 1050m x 1050m area and contains 500 physical nodes. The network is split into 12 by 12 square regions. The node density of this network is roughly the same as the small 8 by 8 region network we used before. The fast motion mode is used for node mobility. Here, each simulation runs for 200 seconds. For each data point, we still repeat the simulation 5 times.

Figure 8-40 and Figure 8-41 show the PDF and route stability of AODV and VNAODV equipped with various optimizations. The trend of the curves matches with the ones we got before with the small network setting. When both Directed Broadcast and Long Links are used by VNAODV, we got the best PDF. In addition, compared with the curves we got with the small network VNAODV outperforms AODV even more. This is because AODV routing here involves even more physical nodes. On route stability, VNAODV using Directed Broadcast but not using Long Links again creates the most stable routes.

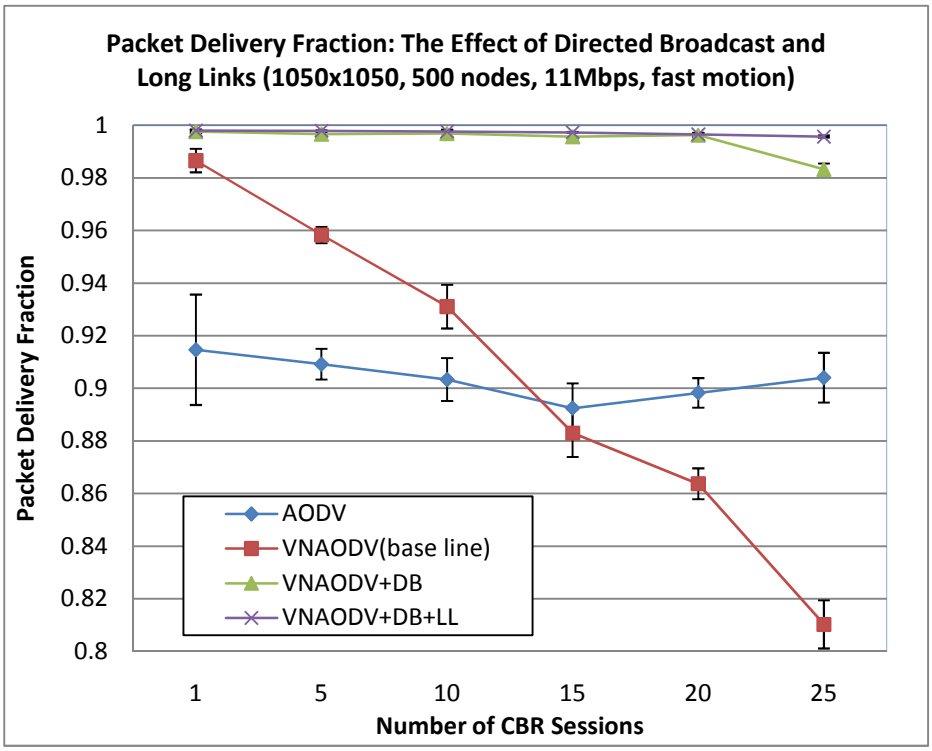

Figure 8-40 The PDF of AODV and VNAODV in a large network setting

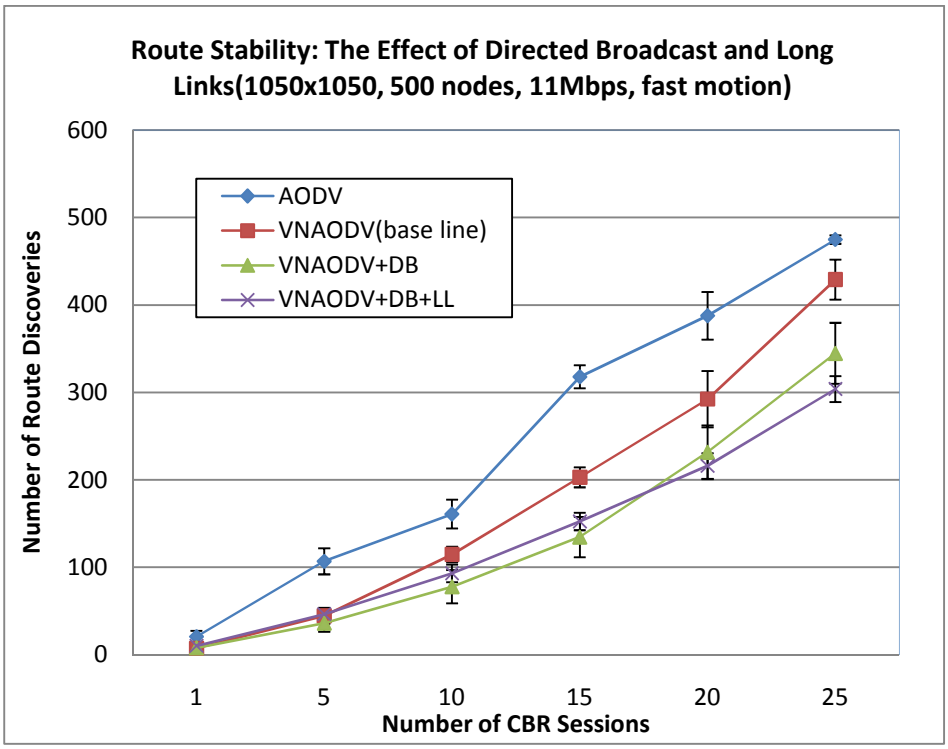

Figure 8-41 The Route Stability of AODV and VNAODV in a large network setting

Figure 8-42 and Figure 8-43 show the delivery latency and control overhead of AODV and VNAODV equipped with various optimizations. Unlike what happens with the small network setup, the delivery latency of AODV increases with more CBR sessions. This might be because with the much heavier routing traffic overhead, the route discovery time gets longer and longer with more CBR sessions. Now, we can see when both Directed Broadcast and Long Links are used, the delivery latency of VNAODV is the best and is about the same as AODV's. This verifies that with a larger network, using the Long Links options brings greater benefit on delivery latency.

Figure 8-43 shows that with DB used, the control overhead of VNAODV is the best. On top of that, with LL used, the control overhead doesn't improve although the forwarding paths are shortened. Finally, as before, the total traffic overhead generated by VNAODV is still heavier than AODV's total traffic, even with both DB and LL used.

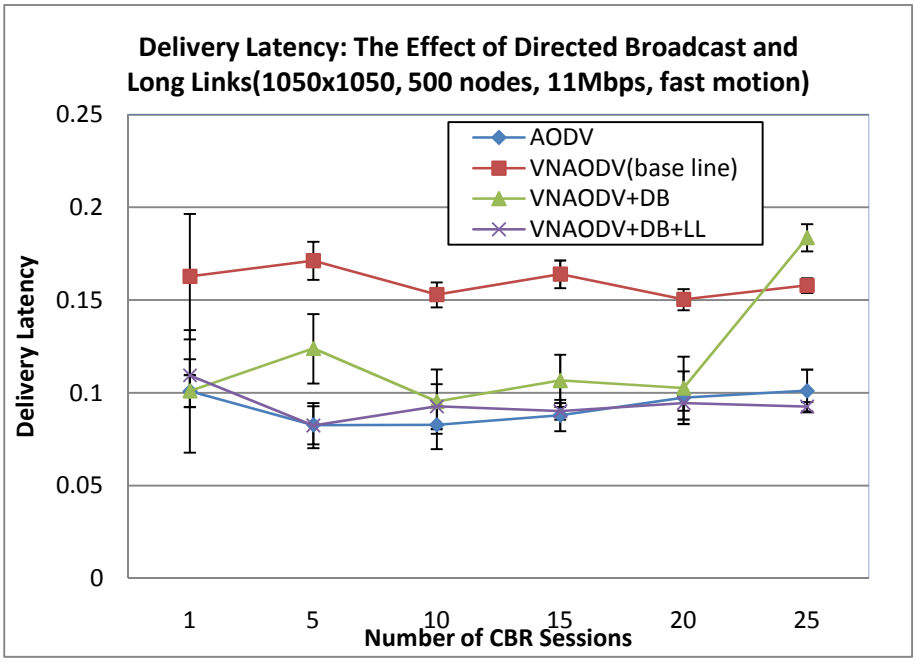

Figure 8-42 The Delivery Latency of AODV and VNAODV in a large network setting

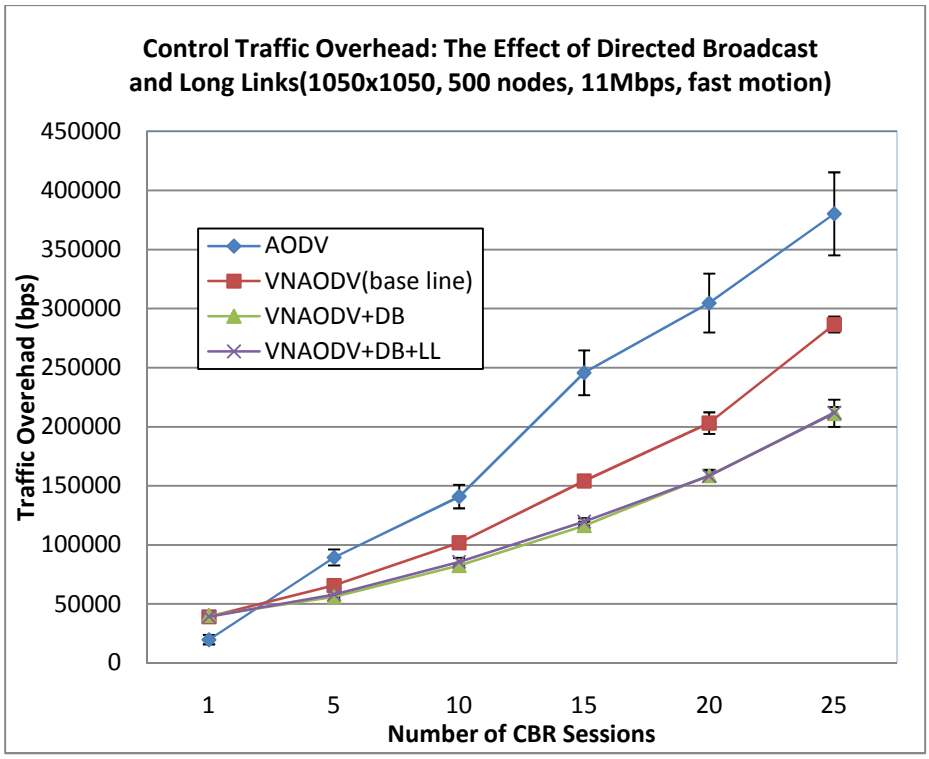

Figure 8-43 The Control Overhead of AODV and VNAODV in a large network setting

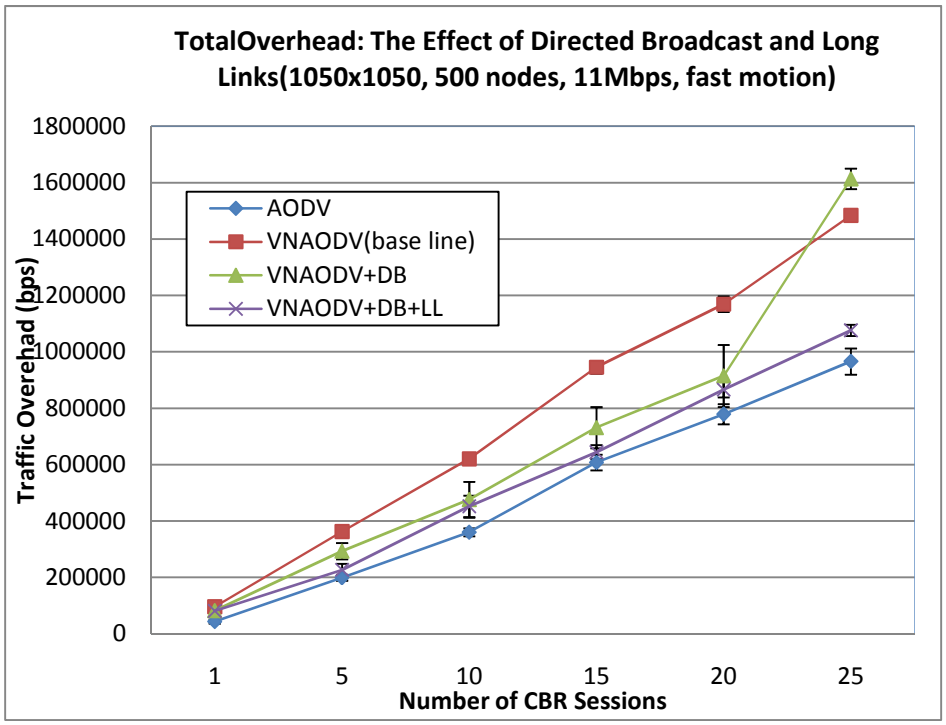

Figure 8-44 The Total Traffic Overhead of AODV and VNAODV in a large network setting

8.2.11 Different Region Setups

In the large network setting we used last section, following the definition in the VNLayer model, the network was divided into 12x12 regions. However, the performance improvement brought by the Long Links option suggests that the regions could be set larger for routing applications. To test the impact of the size of the regions on the performance of VNAODV, we repeated the simulations with the network divided into 6x6 regions and 8x8 regions. Here, this version of VNAODV is equipped with options such as Long Links, Local Recovery and Powerful Emulator.

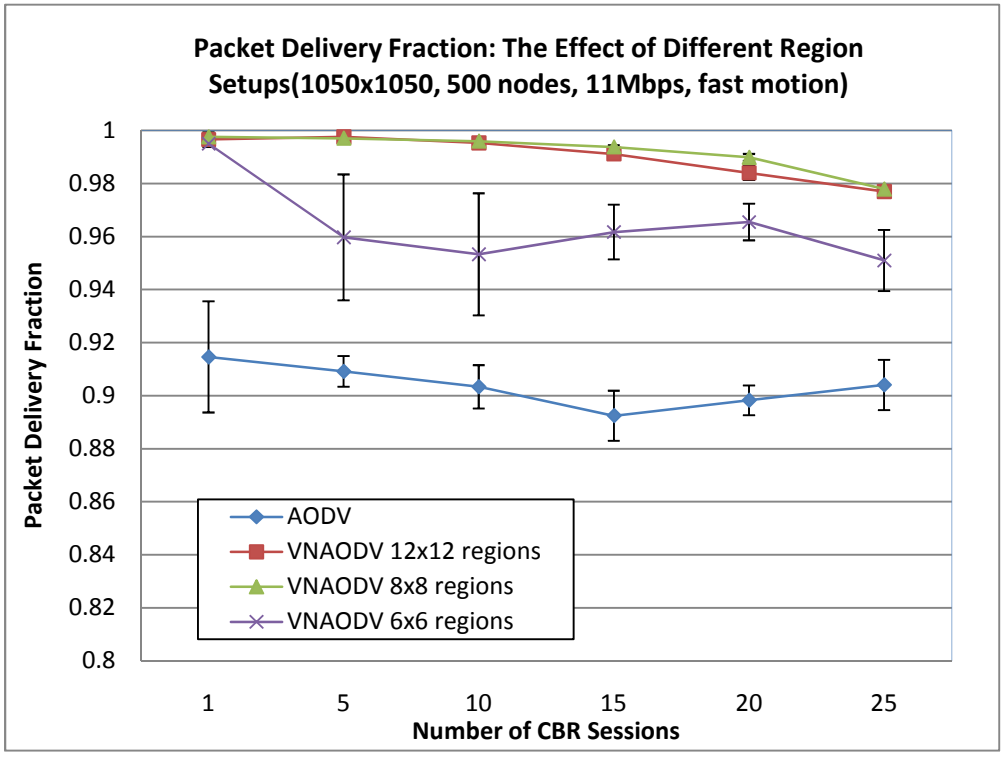

Figure 8-45 The PDF of AODV and VNAODV with different region setups

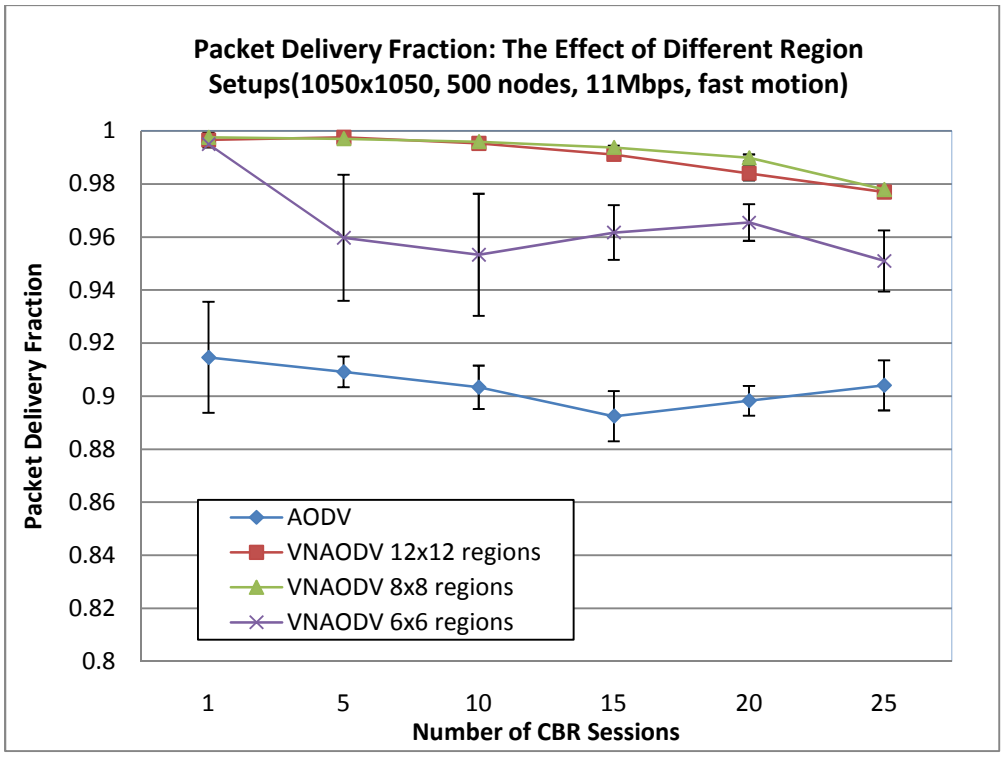

Figure 8-45 shows the data packet delivery ratio of AODV and VNAODV with different region setups. We can see VNAODV still outperforms AODV under all region setups. Reducing the region setting from 12x12 regions to 8x8 regions doesn't hurt the PDF of VNAODV. This is because reducing the total number of regions can reduce the number of vrouters in the network and reduce the number of flooded RREQ messages. However, if we further reduce the number of regions to 6x6 regions, the delivery performance drops. This is because the regions are too large and the links between vrouters are too unstable and sometimes a vrouter may even not be able to reach any vrouter around it. This result verifies that for routing applications, using larger regions won't hurt the delivery performance.

Figure 8-46 and Figure 8-47 compare the forwarding paths length and delivery latency of AODV and VNAODV. When the network is divided into 12x12 regions or 8x8 regions VNAODV creates shorter forwarding paths and causes less forwarding delay than AODV does, due to the use of the Powerful Emulator option. However, due to the use of Long Links, using large regions didn't affect the forwarding path length much. When the network is divided into 6x6 regions, VNAODV creates longer forwarding paths than AODV and causes longer forwarding delay. VNAODV no longer outperforms AODV.

From Figure 8-48, we can see using an 8x8 region setup rather than the standard 12x12 region setup produces the least total network traffic. Dividing the network into 6x6 region, causes more network traffic. The simulation results in this section suggests that when state synchronization among emulator nodes is not important to the performance of an application, larger regions can be used to improve the performance of a VNLayer based application by reducing the number of virtual nodes in a network.

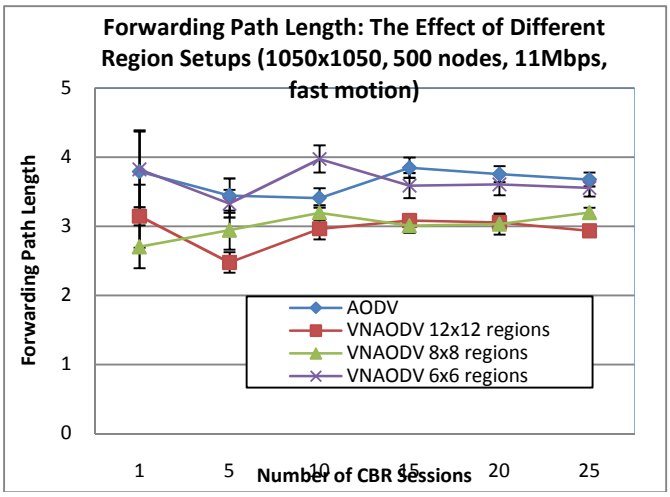

Figure 8-46 Forwarding Path Length of AODV and VNAODV with different region setups

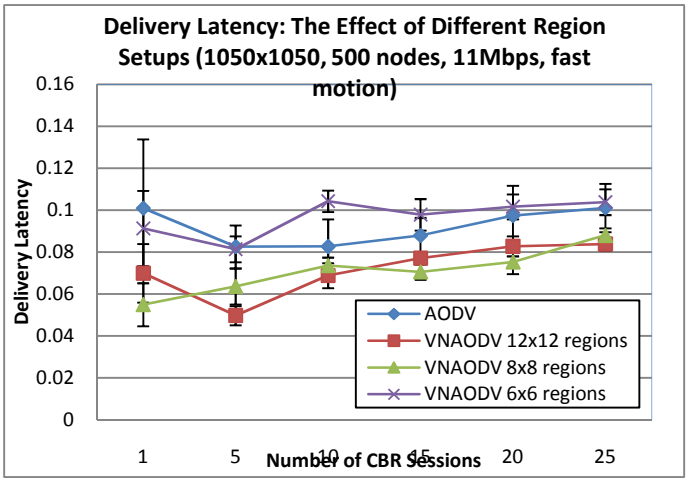

Figure 8-47 Delivery Latency of AODV and VNAODV with different region setups

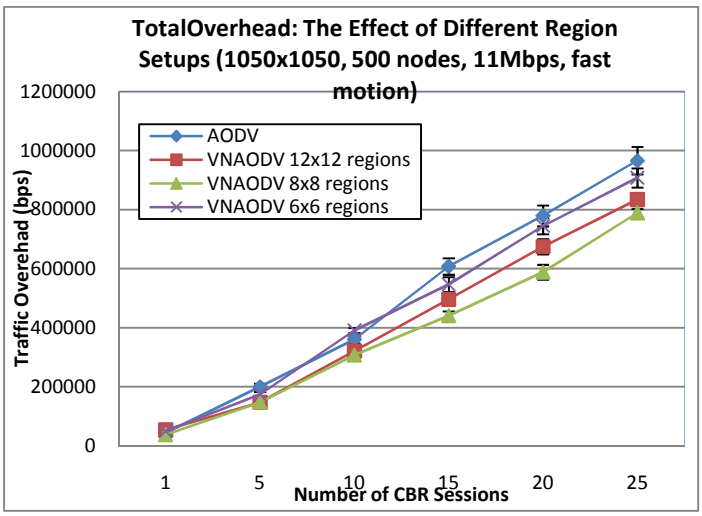

Figure 8-48 Total Traffic Overhead of AODV and VNAODV with different region setups

8.2.12 Summary

In this section, I summarize what is learned through the simulation study on VNLayer based MANET routing.

Through extensive simulations, it is shown that our VNLayer based reactive routing protocol, VNAODV, can outperform the standard AODV, due to the better routing efficiency and reliability brought by the VNLayer approach. This was achieved with optimizations using the using the extended VNLayer model. Among the optimizations, selective state synchronization and selective state consistency checks reduced the state synchronization overhead. Allowing any client processes and virtual nodes to communicate with each other reduced the forwarding path length and led to shorted delivery latency. Finally, using Directed Broadcast for data transmissions drastically reduced the frequency of transmission failures.

We verified that the VNLayer based AODV routing protocol creates more stable routes than the ones created by AODV. We also showed that state replication is not critical to the performance of VNAODV. This is possibly the reason why we could relax the limits in the basic VNLayer model.

We also investigated the effect of a few optimizations at the VNLayer and application layer on the performance of VNAODV. It is shown that the Powerful Emulator option can further reduce the length of forwarding paths create by VNAODV. It is also shown that by reducing the number of Backup Servers and cutting down unnecessary control overhead, the performance of VNAODV can be improved when a network is dense

enough. The local recovery options can be used as an alternative way to reduce delivery failures caused by link layer transmission failures. The Route Correction option can slightly improve reliability of the routes created by VNAODV.

CHAPTER 9. Conclusions and Future Works

This dissertation presents a series of simulation studies on using the VNLayer approach to implement efficient and reliable applications in MANET. The major contributions of this dissertation are:

- A discrete-event based simulator, VNSim, that runs on top of ns2 [39], which can efficiently simulate a VNLayer based network of up to a few hundred physical nodes.

- Identification of a number of performance limitations in the link layer and VNLayer models (presented in section 3.1) used in the original VNLayer simulations[36].

- Extension of the VNLayer model to address the limitations. This dissertation verified that the VNLayer approach can be used to adapt wireline protocols to MANET and to improve the reliability and efficiency of MANET protocols.

In this chapter, I first present on the simulation results. Then, I present the future works.

9.1 Simulation Results

VNSim is an efficient ns-2 based simulator for VNLayer applications, developed for this study of MANET address allocation and routing protocols. With VNSim, using a simple VNLayer based MANET address allocation protocol, VNDHCP, we validated the

intuition that the VNLayer approach can be used to adapt wireline protocols to MANET. Simulations were done for a small network of 16 regions with 40 to 120 mobile nodes and a larger network of 64 regions with 160 mobile nodes, with rate of motion for the mobile nodes varying from quite slow to quite fast. The simulations showed that VNDHCP performs well and the VNLayer layer overhead is small.

Simulation studies are also presented for VNAODV, a VNLayer based routing protocol adapted from the standard AODV. With this work, we discovered problems with the basic VNLayer model. These problems hurt the routing performance badly. To tackle the problems, we created a link layer model that reflects the properties of the MANET more accurately. We also extended the VNLayer model to relax some limits at the VNLayer.

Based on the new models, we applied three major VNLayer optimizations in our VNLayer implementation:

- By doing selective state synchronization and selective consistency checks on incoming messages, the state synchronization traffic overhead was greatly reduced. This in turn leads to fewer collisions and message losses.

- By allowing long links to be used at the VNLayer, the length of VNAODV's forwarding paths and the average delivery latency of VNAODV were greatly reduced. Shortened forwarding paths also lead to fewer message losses.

- The replacement of local broadcast with Directed Broadcast on data transmission reduces message losses. This optimization brought the biggest improvement on packet delivery fraction.

Simulation results also showed that VNAODV generates much less routing overhead due to the reduced number of entities that have to be involved in route discoveries. It is also shown the routes created by VNAODV breaks less frequently, due to the stability of the links between virtual nodes. As a result, VNAODV is able to outperform AODV in terms of packet delivery percentage, routing overhead and route reliability. However, the use of the VNLayer approach does introduce extra traffic overhead (due to extra packet header and extra forwarding at the first hop) and extra packet processing delay.

This work also validated the intuition that the VNLayer approach, as a generalized programming abstraction that hides the complexity of clustering and state replication, can be used to simplify software development and to quickly adapt wireline protocols to the MANET environment. VNRIP, a simple version of RIP built over the VNLayer, was developed very quickly and shown to perform quite well.

This work also led to other VNLayer implementation optimizations. For example, the leader election mechanism was modified to reduce multi-leadership in regions, to shorten leader switching delay, to provide more stability in the region leaders, and to reduce the chance that out-of-sync nodes become leaders. New function modules are designed in the VNLayer to keep track of the activeness of neighbor regions, the leaders of neighbor regions and the whereabouts about physical nodes in the neighborhood. Finally, our

VNLayer implementation can also adjust the number of Backup Servers in a region so that in a dense network, not every physical node has to emulate the virtual node in its region. It is shown in simulations that these optimizations can be used to improve the performance of VNLayer based applications.

9.2 Future works

9.2.1 Applying Insights gained on the Implementation of Cluster-based MANET Protocols

This research provided significant insights on how to implement cluster based MANET protocols with state replication capabilities. More work can be done to apply such insights to other cluster based MANET protocols. In this section, I summarize the important points we learned through the simulation study.

9.2.1.1 On Reducing Control Traffic Overhead

Clustering and state replication improve the efficiency and reliability of MANET protocols. However, they both come with their control overhead. In the simulation results with VNLayer based routing, the majority of DMSG (data message) delivery failures are caused by transmission failures at the link layer. Reducing message overhead always resulted in better delivery performance. To get better performance, it is important to keep the control overhead on clustering and state synchronization as low as possible.

9.2.1.1.1 Clustering Message Overhead

The clustering overhead is determined by the cluster setting and node mobility. It is not affected by the kinds of applications using the clustering scheme. When clusters are maintained through leader elections, the clustering overhead can be reduced by careful

adjustment on the periodic leadership claim message (in our implementation, the Heartbeat messages) and how many non-leaders shall participate in the rejection of a leadership request.

In our implementation, with the use of the explicit LeaderLeft message, we were able to reduce the frequency of Heartbeats from once per second to once per five seconds without affecting the performance of the applications.

In response to a leader request, if more non-leaders can send the rejection message, the chance of having duplicate leadership in a region is lower. However, the leader election overhead is also higher. This could become a problem with a dense network. In our implementation, in order to reduce the leader election overhead, we only allow the leader to reject a leader request.

9.2.1.1.2 State Synchronization Message Overhead

The state synchronization message overhead depends on the state size and the number of state synchronizations needed. Without proper control, this part of the traffic overhead can easily overwhelm the network. The following general approaches can be used to reduce the state synchronization traffic overhead when strictly synchronized state is not critical to the correctness of an application.

First, we distinguished **hard state** and **soft state** in the extended VNLayer model and allow an application to do state synchronization only on hard state. As defined in section 3.4.4 , **Hard state** is the virtual node state that is critical to the correct operation of an

application. **Soft state** is the virtual node state that is non-critical to the correct operation of an application.

It is the application developer's job to determine which part of application state is hard state and which part is soft state. For example, in VNAODV, we only synchronize the destination sequence number, next hop and metric of valid routing entries because these are considered hard state.

Second, we put an upper bound on state synchronization traffic overhead by limiting the frequency of state synchronizations done by a leader.

Third, we allow non-leaders to synchronize their state with state synchronization messages directed to other non-leaders. When state inconsistency is detected by a non-leader, it is likely that many non-leaders in the same cluster need a state synchronization too. To reduce the number of synchronization requests, non-leaders can use a random backoff mechanism when sending their requests. In addition, when a non-leader hears another synchronization request, it cancels its own request. To be able to do this, each no-leader synchronizes its state to the state sent in any synchronization response from the leader of its region.

Fourth, using the state inferencing option, non-leaders can use overheard application messages from the leader node to update their state to reduce the need for explicit state synchronization. As explained in section 3.4.9 , this is an optimization at the application layer that would break the abstraction, though.

Fifth, a non-leader can use only relevant messages from the leader to check for state inconsistency. This would reduce the number of unnecessary state synchronizations triggered by message losses. For example, in VNDHCP, we didn't do state consistency checks with Forwarded Server messages because they have nothing to do with the application state (the address allocation status).

9.2.1.1.3 Reduction on Periodic Hello messages

Many MANET applications require mobile nodes to use periodic Hello or KeepAlive messages to maintain a list of direct neighbors. This kind of message overhead is proportional to the total number of nodes in a network and is not desirable for MANET applications. To reduce the frequency of Hello messages, a clustering scheme can be implemented to provide coordinated tracking of the presence of neighbors. Our VNLayer implementation is able to track the activity each neighbor by treating every incoming message a Hello message. When a node is silent over a long period of time, the VNLayer sends out a Hello message. Each time a Hello message is received; the VNLayer sends a Hello event to the application layer. This way, the number of Hello messages needed by an application is reduced.

9.2.1.2 On Reducing Transmission Failures

Local broadcast is a simple way to send a packet to multiple physical nodes nearby. However, data transmissions by local broadcast are not reliable due to the lack of link layer support on address resolution, RTC-CTS based channel reservation, and data packet acknowledgements and retransmissions. It was the main reason why our VNLayer based routing protocols didn't scale well initially.

When promiscuous mode is available on every physical node, Directed Broadcast, as defined here, can be used for data transmission when the destination is a single physical node or when the destination is a cluster whose leader's address is known.

For packets that have to be sent by local broadcast, since the link layer doesn't provide data acknowledgement and retransmission, a custom designed local recovery mechanism can be used together with passive data acknowledgement to detect link failures quickly and reduce transmission failures. The local recovery mechanism used by VNAODV in this thesis, as described in section 6.4.1 , is shown to be able to greatly improve the packet delivery fraction.

9.2.1.3 On Shortening Forwarding Paths

Clustering may affect the optimality of forwarding paths created by an application when a physical node is allowed to communicate with its cluster head only or when inter-cluster communications are limited to clusters next to each other. Our simulation results showed that when the major goal is to get more messages delivered across the network rather than maintaining consistent state within each cluster, it is desirable to allow a physical node to communicate with any cluster head directly and allow any two clusters to communicate with each other directly.

9.2.1.4 On Better Leader Election

A clustering scheme should be engineered so that duplicate leadership happens rarely, leadership switches don't happen too frequently, and leadership switching can be done quickly. In our VNLayer implementation, optimizations are done to achieve these goals.

This section presents optimizations that can be used by any clustering scheme to improve its leader election mechanism.

9.2.1.4.1 Dealing with Multi-leadership

Multi-leadership happens when the channel is lossy. Multi-leadership can cause problems such as duplicate message forwarding, heavy traffic overhead, and disrupted state synchronization. While reducing traffic overhead can alleviate this problem, quick resolution of multi-leadership is also important. One simple solution is to let the leader that got its leadership earliest force the other leaders to give up their leadership immediately. Another solution we used is to delay new nodes in a cluster longer before it can claim its leadership, so that it chance it hears a leader Heartbeat message or a LeaderReply message is greater.

9.2.1.4.2 Quick Cluster Leadership Switching

If a region remains leaderless for a long period of time, large number of messages could be dropped. Hence, there is a need to make leadership switching as quickly as possible. The addition of the LeaderLeft message to the leader election mechanism greatly improved the performance of VNLayer based applications. In a cluster-based protocol, if possible, a cluster leader should inform its original cluster when it leaves the cluster.

9.2.1.4.3 Stabilizing Cluster Leadership

Frequently switching leaders causes more state synchronizations and more packet drops. Hence we want the leader elected for a region to last longer. In our VNLayer implementation, in a leader election, nodes moving slower are given higher precedence in

the contest for leadership so that the resulting leadership could be more stable. To achieve this, we use different leadership claim waiting times for nodes moving with different speeds. The faster a node is moving, the longer the node has to wait before it can sends out its leader election request.

9.2.1.5 Reducing the Impact of Virtual Node Resets

As we have seen, for VNDHCP, VNAODV and VNRIP, special measures have to be designed to deal with problems that can arise when a virtual node reboots and lost its state. In VNDHCP, each time a virtual node reboots, in order to prevent duplicate address allocation, we let the virtual node wait a full lease time before it can allocate addresses. In VNAODV, when the vrouter in a region reboots, in order to prevent loops, it sends out an RERR message so that no other vrouters use it as the next hop. In VNRIP, we let a rebooted vrouter send out a Request message to request routing tables from its neighbors, this message also informs the neighbors not to use it as the next hop anymore.

It would be desirable to develop a generalized way to deal with virtual node resets. For example, a VNLayer message can be sent to neighbor regions when a virtual node is restarted so that neighboring regions can take any actions required on loss of a neighbor's state. This way, no additional protocol message type needs to be designed at the application layer. What an application would need to do would be to implement an interface function that handles neighbor server failures.

9.2.2 More Works on VNRIP

VNRIP demonstrated its potential as a MANET routing protocol with reasonable performance. A reason that VNRIP didn't perform as well as VNAODV was because VNLayer optimizations such as LL and Directed Broadcast were not applied on the VNRIP simulations. Further study could be done on VNRIP to implement passive DMSG acknowledgement, local packet recovery, LL and Powerful Emulators on VNRIP. These optimizations are expected to further improve the performance of VNRIP. As a proactive routing protocol, VNRIP is not expected to perform as well as VNAODV when the number of data message sessions is small. However, it can have an advantage over VNAODV when the total number of concurrent sessions is large.

9.2.3 Better Region Setups

In this research, the way we set up the regions is clearly not optimal. The better performance brought by the use of long links and the negligible performance difference when fewer regions are used suggest that better region set up can reduce the number of regions needed to cover a network and improve the efficiency of VNLayer based routing protocols. Further investigations can be done on using different shaped regions and overlapping[85] region setups.

9.2.4 The VNLayer Shared by Multiple Applications

In this research, for each case study, the VNLayer supports only one application at a time. One important advantage of using the VNLayer approach is that multiple applications can share the services provided by the VNLayer. For example, multiple applications can share

[85] When a physical node is in the overlapping area of two regions, it identifies itself with only one region.

the same leader election mechanism so that the average control cost on each application can be reduced. More research can be done on how to use one VNLayer state machine to support multiple applications[86].

9.2.5 Geographical based MANET Routing

The VNLayer also provides a good platform to implement geographical based MANET routing. A GPSR [26] like geographical based MANET routing protocol can be implemented over the VNLayer. Geographical based MANET routing requires a distributed location service, such as GLS (grid location service) [41]. Virtual nodes can be turned into location servers. Using the location service, virtual node emulated routers can do geographical routing the same way as GPSR does. The well-known geographical locations for the virtual nodes and the grid topology among virtual node can be expected to make the geographical based routing easier.

[86] Because the only part of the VNLayer function that can be shared is the leader election algorithm, sharing the VNLayer among multiple applications may not produce much benefit while making the state machine more complicated.

Appendix A: Simulating the VNLayer with ns-2

Based on the design of the Virtual Node Emulator Layer and the Application Layer in VNE, we developed a new simulator, VNSim, for VNLayer based applications on the ns-2 platform [39]. As one of the most popular network simulators, ns-2 has a mature implementation of the 802.11 wireless link layer. The development of VNSim is easier than VNE because the most complicated part, the Mobile Node Layer in VNE is provided by the ns-2 platform. In addition, ns-2 is written with C++, which runs much faster than python. VNSim is designed as a discrete event-based simulator. No periodic state checking is used. Therefore, VNSim is more scalable than VNE. In this chapter, we introduce the structure and major design choices we made on VNSim.

A.1 The Structure of VNSim

VNSim is structured in the same way an implementation of VNLayer would be structured on a real mobile device. VNSim implemented all the function modules introduced in CHAPTER 3. Figure A.1-1 shows the architecture of a VNLayer emulator node simulated by VNSim. Built upon the ns-2 platform, the VNLayer interacts with the ns-2 platform in order to send and receive packets to the simulated wireless channel. At the top, the VNLayer interacts with the application layer, reads and writes application layer state and sending commands to the application layer through the interface functions implemented by the application layer.

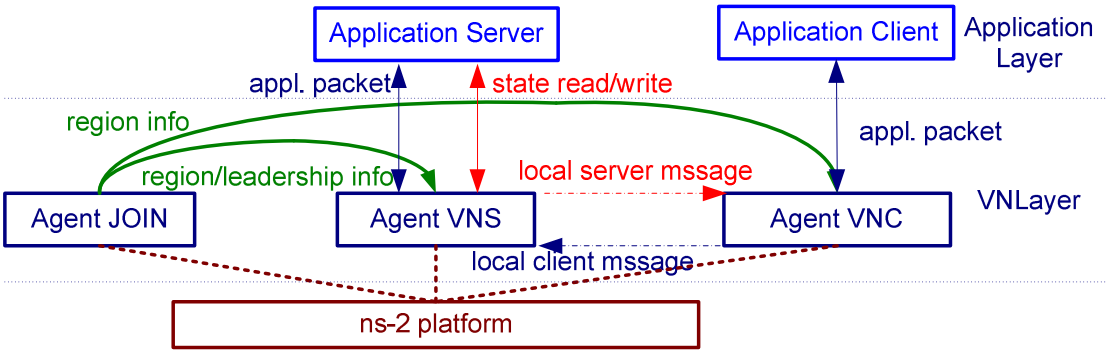

Figure A.1-1 Architecture of a VNLayer emulator node in VNSIM

In VNSim, the VNLayer of each VNLayer emulator node is implemented with 3 types of ns-2 agents, agent JOIN, agent VNS and agent VNC. When there are multiple VNLayer based applications, there will be an agent VNS and agent VNC for each application. Agent JOIN, shared by all VNLayer applications, implements the location checking module and leader election module. It communicates region changes and leader status changes to the other two types of agents, using two types of messages, REGION and LEADER. Agent VNS interfaces with the code for an application server process. On Backup Servers, it also buffers server response messages and keeps the application state synchronized with the Server node's state. Agent VNC interfaces with the code for an application client process. As inter-agent messages on the same node, the REGION and LEADER messages from agent JOIN to agent VNC are sent as loopback messages. So are the messages exchanged between agent VNC and agent VNS on the same node. The code for Agent JOIN, agent VNS and agent VNC, provides the implementation of the VNLayer abstraction. The application server built over the code of Agent VNS and the

application client processes built over the code of Agent VNC compose the Application Layer.

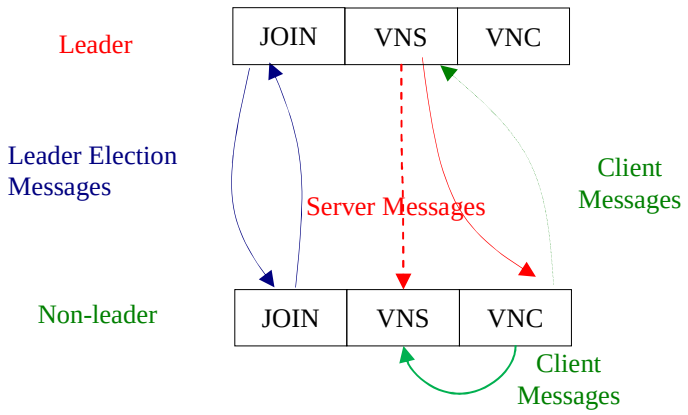

Figure A.1-2 Interaction between a Leader node and a Non-leader Node

Figure A.1-2 shows the interaction between the agents on two mobile nodes. The mobile node above acts as a Server node. The mobile node below acts as a Backup Server that also hosts a client process. The JOIN agents on the two nodes interact with each other for leader election and maintenance. The agent VNC on the Backup Server node interacts with the agent VNS on the Server node for services. The agent VNS on the Backup Server node gets a copy of all the client messages from agent VNC on the same node and prepares its response messages. It also listens to the channel and uses Server messages from the same region to detect state inconsistencies.

A.2 Agent JOIN

Agent JOIN is an agent shared by all other VNLayer agents for location checking and leader election. In ns2, each agent on a simulated mobile node is associated with a different port number. To receive the leader election result and location checking result

from agent JOIN, VNLayer agents register their port numbers with agent JOIN to receive location information and leader election results.

In VNSim, mobile nodes move according to standard ns-2 mobility traces generated using the random waypoint model [31]. Agent JOIN checks a node's current location to determine the region the node is in and to check whether it has entered a new region. The ns-2 implementation of the class MobileNode provides methods for determining a mobile node's current location, motion speed and direction. Using the region boundaries, a node's current region can be derived from its current location. On a node, if agent JOIN finds out that the node's current region is different from the one on record, a RegionChanged event is triggered. The region id of the current region is recorded. A REGION message is then sent to each agent port registered with it.

In addition, when a leader election is done, agent JOIN sends a LEADER message to each agent port registered with it.

To improve the scalability of the simulation, instead of checking the location periodically, VNSim checks the location when a node enters the network; when it starts moving; and when it crosses a region boundary. To generate region-boundary crossing events, we use a node's current location, motion rate and direction of motion to predict the time a node enters a new region. This way, we only do location checks when necessary.

A.3 VNServer, the Parent Class of VNLayer Application Servers

Agent VNS buffers and sends the application server messages, synchronizes a non-leader's state with the leader's state, and simulates the application server. In VNSim, Agent VNS is defined in a class called VNServer.

Agent VNS sorts the incoming packets with the timestamp carried in the packets and put them into a buffer, before they are passed on to the consistency manager. A bi-directional linked list is used as the sending queue on each mobile node. A timer is used to schedule the sending of the messages in the sending queue. with a short interval between consecutive packet transmissions. Since Backup Server nodes don't send response messages, they don't set the sending timer. When the Server node in the region leaves or crashes, a Backup Server node may become the Server of the region and start to set the sending timer.

Any application server class created over the VNLayer must uses VNServer as its parent class and implements a set of virtual functions declared by VNServer. Therefore, an application server is an extended agent VNS that includes the application server code. An application server agent starts running or restarts each time when it receives a REGION message from agent JOIN. Each time a REGION message is received, if the node's region is changed, agent VNS resets the application layer state and waits for the result of leader election from agent JOIN (LEADER message). Once the leader status is determined, agent VNS decides whether the node shall behave as a Server node or a Backup Server node or a Pure Client (using the Coin Tosser module). A Server node

initializes its state and starts to process application packets right away. A Backup Server node needs to get its state synchronized with the Server's state before it can process application packets.

A.4 VNClient, the Parent Class of Application Clients

Although a client process doesn't need to know anything about the VNLayer, it can't communicate directly with a virtual node before its packets are tagged with its region id. This is done by Agent VNC, defined in class VNClient, by inserting a VNLayer message header to every client message.

In VNSim, the class for a client process shall be declared as a child class of VNClient. Then, an application client in VNSim is simulated by an extended Agent VNC. An application client agent starts running when it gets a REGION message from agent JOIN.

A.5 Issues with Port Number

In ns-2, an agent can only hear messages destined to its listening port. However, for state synchronization purpose, the agent VNS on a Backup Server node needs to hear all the server messages received by the client process on the same node. However, these server messages are sent to the port used by agent VNC, rather than the port used by agent VNS. To solve this problem, in the application layer code for a client processes, when a server message is received, a copy of the message is sent to agent VNS on the same node using a loopback message.

A.6 Modified VNSim Structure for Routing Applications

The VNSim with the structure introduced above was used to simulate our VNLayer based MANET address allocation protocol and routing protocols such as VNAODV and

VNRIP. When VNSim is used to simulate VNLayer based routing applications, we modified the VNSim structure so that it accepts data traffic generated by third party traffic generators. In ns-2, built-in MANET routing protocols are implemented as individual agents working at the routing layer, which decides how to forward incoming traffic or local data traffic generated by the node itself. The built-in routing agents can process TCP or UDP data traffic generated by built in traffic generator agents. To make the performance comparison between the ns-2 built-in routing protocols and our VNLayer based routing protocol fair, we made VNSim an ns-2 compliant routing agent.

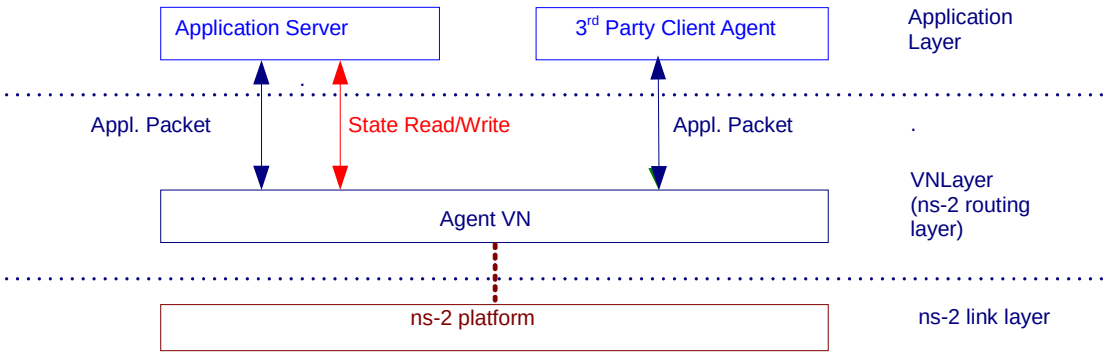

Figure A.6-1 the Architecture of the Modified VNSim

Figure A.6-1 shows the architecture of the modified VNSim. Now, the functions provided by the three types of ns2 agents are integrated into one agent called Agent VN. Agent VNC in the original structure is not needed when the client process is provided by a built-in traffic generator agent[87]. The client message handler module that was implemented in agent VNC is now provided by agent VN. Agent VN works at the ns-2 routing layer. When a client message is received from a local traffic generator agent, agent VN adds the

[87] In our simulation, the ns-2 agent, CBR is used to generate constant bit rate UDP traffic.

VNLayer packet header to the message and passes it to the application layer. The leader election module and location checking module implemented by Agent JOIN are also provided by agent VN. The integration of all the VNLayer function modules into one agent allows us to use only one ns2 port number for the simulated VNLayer. Inter-agent communications are not done by function calls within the same agent rather than loopback messages. This simplifies the simulator code. The problem with this implementation is that the leader election module can only used by a single VNLayer based application.

A.7 Interface Functions required by VNSim for VNLayer based Applications

Table A.7-1 lists out all the interface functions that have to be implemented by the application layer code. In addition to these functions, the VNLayer sending queue is accessible by the application layer. By calling a function enqueue(), the application layer can easily pass a response message down to the VNLayer.

Table A.7-1 Interface Functions Required by the VNLayer Class in VNSim

Interface Function	Purpose
equal()	A function used by the VNLayer consistency manager to check if two application layer packets are the response to the same incoming message.
getState()	Used by the VNLayer to retrieve application layer state
saveState()	Used by the VNLayer to synchronize application layer state with incoming SYN-ACK messages

getStateSize()	Used by the VNLayer to get the size of the application layer state in bytes
getStateHash()	Used by the VNLayer to retrieve a hash of the application layer state
handlePacket()	Used by the VNLayer to pass application layer packets to the application layer
handleApplMsg()	Used by the VNLayer to pass application layer packets to the application layer when a node is out of sync
handleClientPacket()	Used by the VNLayer to pass client messages to the application layer when the option Powerful Emulator is turned on
handleHello()	Used by the VNLayer to pass a Hello event to the application layer when any message with a VNLayer header is received by the VNLayer.
Server_init()	Used by the VNLayer to initialize the application layer state when a node enters a new region

Bibliography

[1] M. Brown et al., "The Virtual Node Layer: A programming Abstraction for Wireless Sensor Networks," in *Proceedings of The International Workshop On Wireless Sensor Network Architecture (WWSNA)*, Cambridge, MA, Apr. 2007.

[2] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, and T. Nolte, "Timed virtual stationary automata," in *in 9th International Conference on Principles of Distributed Systems (OPODIS 2005)*, 2005.

[3] S. Dolev et al., "Virtual mobile nodes for mobile adhoc networks," in *in Proceeding of the 18th International Conference on Distributed Computing (DISC)*.

[4] S. Dolev, S. Gilbert, E. Schiller, A. Shvartsman, and J. Welch, "Autonomous virtual mobile nodes," in *DIAL-M-POMC 2005: Third Annual ACM/SIGMOBILE International Workshop on Foundation of Mobile Computing*, Cologne, Germany, 2005, pp. 62-69.

[5] S., Gilbert, S., Schiller, E., Shvartsman, A., and Welch J. Dolev, "Autonomous Virtual Mobile Nodes," in *the 3rd Workshop on Foundatioins of Mobile Computing (DIAL-M-POMC)*, Sept. 2005.

[6] S. Dolev, L. Lahiani, N. Lynch, and T. Nolte, "Self-stabilizing mobile node location management and message routing," in *7th International Symposium on Self-Stabilizing Systems (SSS 2005)*, Barcelona, Spain, Oct. 2005.

[7] S. Gilbert, Virtual Infrastructure for Wireless Ad Hoc Networks, 2007, Ph. D. Thesis, MIT.

[8] J. Y. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, First Qtr., 2005.

[9] L., Evans, D. Hu, "Localization for Mobile Sensor Networks," in *Tenth Annual International Conference on Mobile Computing and Networking (Mobicom 2004)*, Philadelphia, 2004.

[10] Jang-Ping Sheu, Wei-Kai Hu, and Jen-Chiao Lin, "Distributed Localization Scheme for Mobile Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 4, pp. 516-526, April 2010.

[11] H. Chen, M. H. T. Martins, P. Huang, H. C. So, and K. Sezaki, "Cooperative node locali-zation for mobile sensor networks," in *Proceedings of The 5th International Conference on Embedded and Ubiquitous Computing, EUC 2008*, Shanghai, 2008, pp. 302-308.

[12] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan, "Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures," in *Infocom 2001*, pp. 1763-1772.

[13] Charles E. Perkins and Elizabeth M. Royer, "Ad hoc On-Demand Distance Vector Routing," in *the Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.

[14] Gary S. Malkin, “Routing Information Protocol V2”, RFC 2453, IETF, January, 1993.

[15] M. Al-Shurman, S.-M. Yoo, and S. Part, "A Performance Simulation for Route Maintenance in Wirelesss Ad Hoc Networks," in *Proceedings of the 42nd annual Southeast region conference*, New York, NY, 2004, pp. 25-30.

[16] Z. Bilgin, B. Khan, and A. Al-Fuqaha, "Only the Short Die Old: Route Optimization in MANETs by Dynamic Subconnection Shrinking," in *The 6th International Wireless Communications & Mobile Computing Confe-rence*, Caen, France, 2010.

[17] Ralph Droms, Dynamic Host Configuration Protocol, RFC 2131, Internet Engineering Task Force, Network Working Group, March 1997.

[18] Charles E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks", July 2000, draft-ietfmanet-autoconf-01.txt, Internet Engineering Task Force, MANET Working Group.

[19] David B. Johnson and David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Kluwer Academic Publishers, Mobile Computing*, vol. 353, 1996.

[20] Hongbo Zhou, "A survey on routing protocols in MANETs," Department of Computer Sciences, Michigan State University, Technical report MSUCSE-03-8, 2003.

[21] S. Nesargi and R. Prakash, "MANETconf: Configuration of hosts in a mobile ad hoc," in *Infocom 2002*.

[22] Baochun Li Zhihua Hu, "ZAL: Zero-Maintenance Address Allocation in Mobile Wireless," in *the Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS 2005)*, 2005, pp. pp.103-112.

[23] S. Thomson and T. Narten, IPv6 Stateless Address Autoconfiguration, RFC 2462, Internet Engineering Task Force, Zeroconf Working Group, December 1998.

[24] R. Wakikawa, Jari T. Malinen, C. E. Perkins, A. Nilsson, and J. Tuominen, “Global Connectivity for IPv6 Mobile Ad Hoc Networks”, INTERNET-DRAFT,IETF, Mobile Ad Hoc Networking Working Group , November 2001.

[25] J. Moy, OSPF Version 2, RFC 2328, IETF, April, 1998.

[26] Brad Karp and Kung H. T, "GPSR: greedy perimeter stateless routing for wireless," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 243-254.

[27] Charles E. Perkins and Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," in *ACM SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications*, 1994.

[28] Richard Bellman, "On a Routing Problem," *Quarterly of Applied Mathematics*, no. 16(1), pp. 87-90, 1958.

[29] P. Jacquet et al., "Optimized Link State Routing Protocol for Ad Hoc Networks," in *Multi Topic Conference. IEEE INMIC 2001. Technology for the 21th Century. Proceedings.*, 2001.

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[31] J. Broch, D. A. Maltz, D. B. Johnson, and Y. C. Hu, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Mobilecom*, 1998, pp. p85-97.

[32] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan, "CEDAR: a Core-Extraction Distributed Ad Hoc Routing Algorithm," in *Infocom 1999*, pp. 202-209.

[33] L. Ritchie, H. S. Yang, A. Richa, and M. Reisslein, "Cluster overlay broadcast (COB): Manet routing with complexity polynomial in source destination distance," *Mobile Computing, IEEE Transactions on Publication*, vol. 5, no. 6, June 2006.

[34] C.-C. Chiang, H.-K.Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," in *Proc. IEEE Singapore Int. Conf. on Networks*, 1997, pp. 197-211.

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

[36] Mike Spindel, Simulation and Evaluation of the Reactive Virtual Node Layer, 2007, Mater's Thesis.

[37] N. Lynch, R. Segala, and F. Vaandrager D. Kaynar, "The theory of timed i/o automata," *Synthesis Lectures on Computer Science*, 2006.

[38] M. Spindel. The Virtual Node Emulator. [Online]. https://carbide.mit.edu/trac/vne

[39] The Network Simulator, ns-2. [Online]. http://www.isi.edu/nsnam/ns

[40] C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On-demand Distance Vector Routing", RFC 3561, July 2003, IETF, Network Working Group.

[41] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris, "A scalable location service for geographic ad-hoc routing," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom 2000)*.

[42] P. Gupta and P. Kumar, "Capacity of wireless networks," University of Illinois, Urbana-Champaign, Technical report 1999.

[43] G. Pei, M. Gerla, and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility," in *Proceedings of IEEE/ACM MobiHOC 2000*, Boston, MA, Aug. 2000, pp. 11-18.

[44] Kaixin Xu, Xiaoyan Hong, and Mario Gerla, "Landmark routing in ad hoc networks with mobile backbones," *Journal of Parallel and Distributed Computing archive*, vol. 63, no. Issue 2, Special issue on Routing in mobile and wireless ad hoc networks, pp. 110-122, 2003.

[45] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," in *RFC 3626, IETF Network Working Group*, October 2003.

[46] J Broch, DA Maltz, DB Johnson, Y-C Hu, and J 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(Mobicom98)*, October, 1998.

[47] Ping Ji, Zihui Ge, J. Kurose, and D. Towsley, "A comparison of Hard-state and Soft-state Signaling Protocols," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 281-294, April 2007.

[48] D. D. Clark, "The design philosophy of the DARPA internet protocols," in *Proceeding of SIGCOMM*, Stanford, CA, August 1988.