Simulating Fixed Virtual Nodes for Adapting Wireline Protocols to MANET*

Jiang Wu [#1], Nancy Griffeth [#2], Nancy Lynch [†3], Calvin Newport [†4], Ralph Droms [††5]
[#]City University of New York [†] Massachusetts Institute of Technology [††]Cisco Systems
[1]jwu1@gc.cuny.edu [2]nancy.griffeth@lehman.cuny.edu [3]lynch@csail.mit.edu [4]cnewport@mit.edu [5]rdroms@cisco.com

***Abstract*— The Virtual Node Layer (VNLayer) is a programming abstraction for Mobile Ad Hoc Networks (MANETs). It defines simple virtual servers at fixed locations in a network, addressing a central problem for MANETs, which is the absence of fixed infrastructure. Advantages of this abstraction are that persistent state is maintained in each region, even when mobile nodes move or fail, and that simple wireline protocols can be deployed on the infrastructure, thereby taming the difficulties inherent in MANET setting. The major disadvantage is the messaging overhead for maintaining the persistent state.**

In this paper, we use simulation to determine the magnitude of the messaging overhead and the impact on the performance of the protocol. The overhead of maintaining the servers and the persistent state is small in bytes, but the number of messages required is relatively large. In spite of this, the latency of address allocation is relatively small and almost all mobile nodes have an address for 99 percent of their lifetime. Our ns-2 based simulation package (VNSim) implements the VNLayer using a leader-based state replication strategy to emulate the virtual nodes. VNSim efficiently simulates a virtual node system with up to a few hundred mobile nodes. VNSim can be used to simulate any VNLayer-based application.

*This work was supported in part by the Cisco Collaborative Research Initiative

I. Introduction

The central challenge of Mobile Ad-Hoc Networks (MANETs) is the absence of fixed infrastructure. Fixed infrastructure in wireline networks improves scalability by permitting imposition of a hierarchy on participating nodes and it improves usability of the network by providing services from fixed servers. In contrast, in a MANET, participating nodes can enter or leave the network at will and may move rapidly even while participating, making the network topology highly dynamic and the links between mobile nodes very unstable. The radio channel shared by mobile nodes is susceptible to frequent interference and message collisions; the lifetime of mobile nodes are constrained by battery power. These inherent complexities make scalable implementation of services in a MANET exceedingly difficult.

One important approach to achieving scalability in MANET protocols is clustering [1]. The Virtual Node Layer (VNLayer) [2] is a cluster-based approach that defines "virtual" servers at fixed locations in the networks – called "*Virtual Nodes*". Several different VNLayer definitions and implementation strategies have been discussed in the theoretical literature [3][4][5]. A standard technique for implementing a VNLayer is to divide the network into geographical regions at fixed locations or moving in a controlled manner. Within each region, a subset of the physical mobile nodes emulates a virtual node, using replicated state machines and elected leaders. To a physical node in a region, a virtual node works as if it is a fixed server.

VNLayer provides a generalized abstraction to the programmers, hiding most of the MANET complexities from them. The developers can deploy applications on both the mobile devices and these virtual static servers with greater ease and efficiency. In addition, state machines constantly running on mobile nodes can keep the application state consistent among participating physical nodes in a region. Therefore, although individual physical node might fail or leave a region, the virtual node of the region can maintain persistent state.

The VNLayer approach also has its disadvantages. Mobile nodes are required to have some sort of GPS-like capability so that they can know the region they are in. However, this knowledge can also be used to good effect in applications such as geographical based routing [6]. More resource and processing power are required for the mobile nodes. The operation of the VNLayer, especially the leader election and state replication, generates extra message overhead.

In this paper, we present a case study of an implementation of a DHCP-like [7] address allocation server over the VNLayer. DHCP is picked because it is a simple protocol in wireline environment and dynamic address allocation in a MANET is difficult. There are minor changes to the algorithm to take advantage of the multiple region-based servers and to account for mobility, but it is otherwise almost identical to the familiar wireline DHCP.

The major performance results are that the VNLayer overhead is small in bytes but high in total number of messages. The latency of address allocation is relatively small and mobile nodes have an address for most of their lifetime. This suggests a general approach to developing services on a MANET, in which wireline services are used directly or with minor modifications over a VNLayer architecture.

In addition, a ns-2 [8] based simulation package, VNSim, is designed to support VNLayer-based applications. Simulation results show that VNSim can simulate a virtual node system with up to a few hundred mobile nodes.

The paper is structured as follows. Section II explores the background of this research. In section III, our specific implementation of the VNLayer and the architecture of VNSim is introduced. Section IV describes our example application, a VNLayer-based address allocation algorithm. Performance

evaluations on VNSim and the example application are given in section V. Conclusions and future work are discussed in section VI.

II. Background

A. Cluster-based MANET Algorithms

The basic idea of clustering is to group mobile nodes into clusters and inter-cluster communications are handled by a set of cluster heads. A large number of clustering algorithms and cluster-based MANET protocols have been designed.

In [1], Yu gave a survey of clustering algorithms, which create a Dominating Set (DS) of the mobile nodes in a MANET as the cluster heads. Clustering is used in many MANET routing protocols. Cluster Overlay Broadcast (COB) [9] uses Least Cluster Change (LCC) algorithm [10] for clustering. It works similar to AODV[11], with route request messages and route reply messages flooded only by cluster heads. Receiving a route reply message, a cluster head marks itself as active for the specific session. However, the route created by COB can only be used once. CEDAR [12] is another cluster-based ad-hoc routing protocol. Cluster heads (core nodes) are elected using a special core extraction algorithm that creates a DS. The hop distance between any pair of mobile nodes in DS can be at most 3. Routing messages are exchanged among the core nodes using unicast.

In addition to using clusters, the VNLayer has state replication capability and maintenance of persistent state. In addition, in most dynamic clustering schemes, the restructure of a cluster can cause a neighbor cluster to restructure. This can cause the restructure of more clusters. This is called the rippling effect. The VNLayer is free of this issue because its clustering is geographical region based. The leadership or membership changes in one region won't affect other regions.

B. Address Allocation in MANET

A robust address allocation scheme is critical to successful message delivery and correct routing operation. However, it is difficult to implement dynamic address allocation in a MANET. There is no centralized entity that can provide address service because any mobile node may leave the network at any time. Mobile devices may move around quickly and the wireless link between nodes may fail any time due to message collision and channel congestion.

In [13], Perkins described a method in which a node joining a MANET picks a random address and checks to see if there is a valid route to the address. If so, it means the address has already been taken. The joining node tries more addresses until an unused one is found. The solution relies on flooding based route searches. Address duplication can happen when a route search fails to tell an address is in use. MANETConf [14] offers a more comprehensive approach, in which the address picked for a node will be confirmed and recorded by every node in the network. A node leaving the network explicitly gives up its address. However, control messages are distributed to all the nodes using flooding. In our simulation studies, we have found that flooding introduces unacceptable overhead.

III. Implementing the VNLayer

In this section, we introduce the specific implementation of VNLayer we used in our research.

A. Reactive VNLayer

In this paper, we use an implementation of the VNLayer with fixed regions. In addition, each virtual node has a simple message receive event-driven deterministic automaton built-in. The operation of the automaton is defined by a msgReceive() function. Responding to each message received, the automaton in the virtual node modifies the state and determines a response to be broadcast (if any). We call this specific implementation the Reactive VNLayer. For simplicity, we still use "VNLayer" for the rest of this paper.

B. VNLayer-based System

In a VNLayer-based system, a MANET is tiled with equal sized square ***regions***. The size of each region is small enough that a message sent from one region can be heard by all nodes in the same region and in the immediate neighboring regions. Assuming the mobile nodes have GPS-like capability, each node knows the region it is in. Selected mobile nodes in a region jointly emulate a virtual server for the region, which is called a ***virtual node***. Inter-region messages are always relayed through the virtual nodes. In each region, the nodes emulating the virtual node run the same server application code and process client messages exactly the same way. However, only one node among them, an elected ***leader***, sends out the server response messages. The other nodes, the ***non-leaders***, buffer their response messages. Overhearing the leader's response messages, the non-leaders remove the corresponding message from their buffers. The non-leaders also keep their state synchronized with the leader's state. In case the current leader leaves or crashes, a non-leader node is elected as the new leader of the region. The new leader sends the messages left in its buffer. The non-leaders hence work as backup servers for the leader.

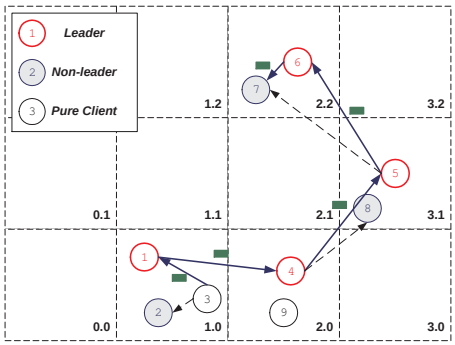

Fig. 1. Illustration of a virtual node system. A message from a pure client node 3 in region 1.0 is forwarded to a non-leader node 7 in region 2.2 through leader node 1,4,5 and 6. Non-leader node 2,7 and 8 work as backup routers.

Figure 1 illustrates a VNLayer-based system of 12 regions. Each non-empty region has one leader node. The virtual node in an empty region is ***down***; when a mobile node arrives in the

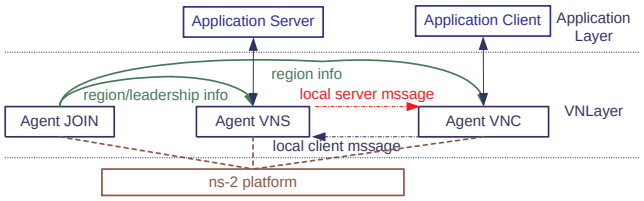

Fig. 2. The architecture of a mobile node simulated by VNSim, an ns-2 based virtual node simulator

region, it boots ***up*** and initializes the virtual node. Each node may also host a client process. The nodes hosting only client processes are called ***pure client*** nodes. In our simulations, every mobile node is both a client node and a leader emulating node.

C. Architecture of VNSim

VNSim, the VNLayer simulation, is structured in the same way an implementation would be structured. As shown in Figure 2, the VNLayer is built upon the ns-2 platform and the application layer is built over the VNLayer. The VNLayer interacts with the ns-2 platform, sends application layer and VNLayer packets to the simulated channel and receives packets from the channel for the mobile node. It buffers application layer packets before they are relayed to the application servers. The VNLayer buffers packets generated by the application layer and delivers them to the channel, depending on the role played by the mobile node. The VNLayer reads and writes application layer state variables to synchronize state between leader and non-leader nodes.

On each mobile node, the VNLayer is implemented with three ns-2 agents, **JOIN**, **VNS** and **VNC**. Agent JOIN provides the node movement tracking and leader election functions. It communicates region and leader status changes to the other two agents on the same node, using two types of loopback messages, REGION and LEADER. Agent VNS interfaces with the application server process, buffers non-leader response messages and synchronizes non-leader state with leader state. Agent VNC interfaces with the application client process. The application layer messages exchanged between the agent VNC and agent VNS on the same node are also implemented as loopback messages. On different mobile nodes, the JOIN agents interact with each other for leader election.

1) Agent JOIN: Agent JOIN keeps track of node movement and elects leaders among nodes in the same region.

Movement Tracking: Motion information about a simulated mobile node, such as the current location, motion speed and direction can be retrieved from the ns-2 platform. Using this knowledge and the region setting, agent JOIN can tell the current region a node is in. If it finds out a node is in a different region, the id of the current region is recorded and a REGION messages is sent to agent VNS and agent VNC on the same node.

To improve the scalability of the simulation, instead of checking the location continually, VNSim checks the location only 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 speed and direction to predict the next time the node crosses a region boundary.

Leader Election: A simple leader election algorithm picks the leader for a region. Each time a node enters a new region, it first resets its status to *requesting*, sends out a LeaderRequest message, and sets a request timer. Receiving the message, the current leader of the region, if there is one, sends a LeaderReply message declining the request. If another *requesting* node receives the message, it compares the sending time of its LeaderRequest message with the sending time of the incoming message. If the receiving node sent its request earlier, it sends a LeaderReply message to decline the request. Otherwise, it gives up its request and becomes a non-leader. Basically, the first node requesting leadership becomes the leader.

If the request timer on a requesting node expires and no LeaderReply message is received, the node considers itself the region leader and starts to send out HeartBeat messages periodically. Once the leader status is determined on a node, agent JOIN sends a LEADER message to agent VNS. A non-leader node sets a timer for the time that it expects the next HeartBeat message from the leader. Every time the timer expires and no HeartBeat message is heard on a non-leader, a HeartBeat miss is recorded. A non-leader starts a new leader request when two HeartBeat messages are missed.

Due to message losses, more than one leader may exist in the same region. When this happens, one leader can hear the HeartBeat messages sent by another leader. HeartBeat messages carry the time when the sender becomes a leader. Using this information, the leader that becomes the leader later gives up its leadership.

2) Agent VNS: Agent VNS buffers incoming and outgoing messages for the application layer, sends the server response messages and synchronizes a non-leader's state with the leader's state.

Packet Buffering: Incoming application messages are sent to a buffer and delayed for a short period of time, during which the messages are sorted based on their sending time. This is to reduce the number of state synchronizations caused by out of order messages.

Another buffer is used to store outgoing messages from the application layer. A timer is used to schedule the sending of the messages in the queue with a certain sending rate. Since non-leader nodes don't send response messages, they don't set the sending timer. When a non-leader turns into a leader, it starts to set the sending timer.

State Synchronization: Each VNS agent tracks the state variables to keep them synchronized between the leader and non-leaders of a region. Since the leader and non-leaders in the same region are supposed to prepare the same sequence of responses to the client messages, a non-leader should receive from the leader a copy of every message in its sending buffer. When a non-leader can't find a match in its sending buffer for a incoming messages, it triggers a state synchronization.

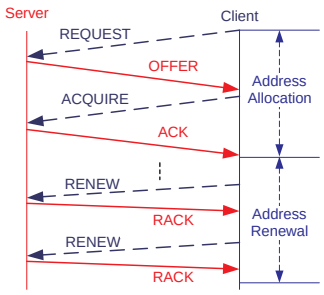

Fig. 3. Flowchart of a VNLayer-based Address Allocation Algorithm

To synchronize its state with the leader's, a non-leader clears its sending buffer, sends out a SYN-REQ message and sets a timer for the response. The leader node replies with a SYN-ACK message carrying its state and the non-leader copies the leader's state into its own state. Agent VNS on a non-leader stops relaying messages to the application server during the synchronization process and re-sends SYN-REQ messages until it receives a SYN-ACK.

Interacting with agent JOIN and the Application Server: Agent VNS starts running or restarts each time when it receives a REGION message, which is sent from agent JOIN, telling the node the region it is in. It resets its state and waits for the result of leader election from agent JOIN (LEADER message). Once the leader status is determined, agent VNS knows whether it should behave as a leader or non-leader. On a leader node, agent VNS starts relaying messages to the application server right away. On a non-leader node, agent VNS won't relay messages to the application server before it is done with synchronizing the application state with the leader node. The application server process needs to provide access to its state variables so that they can be read and written by agent VNS.

3) Agent VNC: Agent VNC also starts running or restarts when it gets a REGION message from agent JOIN. Its main function is relaying messages between the wireless channel and the application client.

IV. An Example VNLayer-based Application

In this section, we introduce our VNLayer-based application, a MANET address allocation algorithm.

A. Basic Protocol Operation

The address allocation protocol that we have implemented on VNSim, is similar to DHCP. Figure 3 shows a simple address allocation procedure. The virtual node in each region is assigned a pool of addresses for its clients. A client node that needs an address broadcasts a REQUEST message to its region. Receiving the REQUEST message, the region leader determines whether it has a ***free*** address in its address pool. If so, it sends out an OFFER message and sets the status of the address as ***pending***. Receiving the OFFER message, the client confirms that it wants the address with an ACQUIRE message. The leader answers the message with an ACK message and set the address's status as ***assigned***. Receiving the ACK message, the client node can use the address for a ***lease time***. A short time before the *lease time* expires, the client sends out a RENEW message. Once the leader receives the RENEW message, it updates the lease time for the address and sends back a RACK message. Receiving the RACK message, the client uses the address for another lease time. The RENEW-RACK cycle repeats. If the renewal process fails, the client starts a new address request by sending out a REQUEST message. On the other hand, on the leader, the lease for the address eventually times out. The leader set the status of the address back to ***free***.

The non-leaders in a region process the client messages the same way as the region leader does, except that they don't send out their response messages and they synchronize if they detect a divergence from the leader state.

B. Tweaks on the Protocol to Adapt it to MANET

Up to this point, the address allocation works almost the exactly the same way as DHCP works. However, to make the algorithm work, there are some adaptations needed.

A node in one region ignores most messages coming from other regions, with three exceptions involving the virtual servers (actually, the leaders and non-leaders implementing them).

In the first case, when the local region leader runs out of addresses, we allow the client address to be assigned by a neighbor region. Therefore, a client's REQUEST messages can be forwarded by the virtual node to the virtual nodes in neighbor regions. In addition, the subsequent OFFER, ACQUIRE and ACK messages can also be forwarded to neighbor regions.

In the second case, to avoid address duplication, when a virtual node just boots up an empty region, it needs to set all of its addresses as assigned for a whole lease time, because it doesn't know whether any of the addresses is still used by some client that gets its address from the region and has moved to another region. During this one lease time period, the clients in the region need to get address from the neighbor regions and the four types of messages mentioned above also need to be forwarded to neighbor regions.

In our protocol, when a client message (REQUEST or ACQUIRE, for example) needs to be forwarded, the local virtual node broadcast the message to the neighbor regions. At every forwarding hop, the virtual node appends its region id into a source route carried by the client message. The server messages, OFFER and ACK, for instance, are source routed back to the client using the route recorded in the corresponding client messages. Simulation shows that forwarding the four allocation messages to the immediate neighbor regions is good enough when the size of address pool at each region is large enough, since a region can have up to 8 neighbor regions.

In the third case, because a client node may leave the region where it originally got its address, clients must be able to renew leases from remote regions. To let the clients keep their addresses as long as possible, the renewal messages, RENEW

and RACK, can be forwarded by as many virtual node hops as necessary. The RENEW messages can be forwarded by flooding, or, since the location of the virtual server is known for a client, by unicast routing. The unicast routing protocol repeatedly picks the neighbor region that is the closest to the server region as the next hop. This generates much fewer messages than flooding. The RACK message is also source routed back to the client using the route carried in the corresponding RENEW messages.

C. Discussions

The VNLayer-based address allocation protocol is a distributed server system with strong fail-over capability. The burden of the address allocation service is shared by the virtual nodes covering separate regions in the network. The address allocation and inter-region message forwarding are both handled by virtual nodes, each of which is emulated by a leader and a number of non-leaders.

Although broadcasting is used in the protocol for client messages such as REQUEST and ACQUIRE, the number of hops that these messages can be forwarded is limited to 2. This generates only two times the message overhead than the case that these messages are not forwarded at all. If flooding is used, the forwarding of RENEW message can cause heavy message overhead. However, when unicast routing is used, the renewal message overhead is proportional to the hop distance between the client and the server region. The simulation results show that the service is scalable with increasing network sizes.

Because any given address can only be allocated from a single virtual node and a virtual node waits a whole lease time after it is booted up in a previously empty region, address duplication rarely happens. Due to clock skew or state inconsistency between leaders and non-leader when leadership switches, address duplication can still happen. The duplication can be resolved easily because each client has to renew its address lease. The server only acknowledges the RENEW messages from the correct owner of the address. The clients who can't get an RACK message give up the address. Address leakage will not happen in the protocol because in the absence of RENEW messages from the client using an *assigned* address, the status of the address on a server always goes back to *free*.

The problems with this VNLayer-based scheme are: First, the virtual node layer generates extra message overhead. Second, when a virtual node in a region is *down*, all the clients that got their addresses from the region have to give up their addresses when the renewal fails. Third, 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, REQUEST messages need to be forwarded to more regions.

V. Performance Evaluation

A. Simulation Settings

We ran our simulations using ns-2.31 on a Linux machine with an Intel Pentium 3.20GHz CPU and 512M bytes memory.

TABLE I
SETTINGS FOR 4 MOTION SPEED MODES

	slow	med-slow	med-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

Two network settings were used: a small network of 40 to 120 nodes that contains 16 87.5m$\times$87.5m regions in a 350m$\times$350m and a large network of 160 nodes that contains 64 87.5m$\times$87.5m regions in a 700m$\times$700m area. All the mobile nodes are set to implement the VNLayer as described in Section III. The packet receiving range is set to 250m to make sure that a message sent from a region can reach any node in the neighbor regions. 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 simulation are repeated 5 times with different mobility traces. The address pool size is set to 30 to reduce the chance that a server runs out of addresses.

We allowed the REQUEST and ACQUIRE messages to be forwarded only once because this was almost always enough. The renewal messages are allowed to be forwarded by more hops. Two forwarding methods are used for the RENEW messages, flooding and unicast routing. The flooding of the RENEW message is controlled such that a virtual node forward the same RENEW message only once. The unicast routing we used is a simple best effort routing, in which the neighbor region that is closest to the destination region of the RENEW message is always used, no matter whether the region is empty or not.

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 I.

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.

B. Simulation Speed

Table II lists the simulation time of VNSim for various total numbers of nodes, with the 16 region network setting and slow node motion. The simulation time increase is roughly proportional to the square of the total number of nodes because each node needs to handle messages from all of its neighbors.

TABLE II
SIMULATION SPEED OF VNE AND VNSIM

	40 nodes	80 nodes	120 nodes
VNSim simulation time	2.4 minutes	9.53 minutes	22.23 minutes

C. VNLayer Message Overhead

An important question is what the overhead is for state synchronizations in a VNLayer-based system. Because SYN-ACK messages can be big, large number of synchronizations can cause heavy traffic. In addition, during state synchronizations, non-leader nodes that are out of sync ignore all client messages. This can affect the failover capability of the system.

With the large network setting and speed mode "slow" and "fast", we did simulations with various renewal message forwarding methods (flooding and unicast routing) and forwarding hop limits (1 to 8). More details can be found in the next two sections. In the worst case, the case with speed mode "fast" and 8 hop flooding used for RENEW messages, the VNLayer 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-REQ and SYN-ACK messages are both about 20 per region per lease time.

From the simulation results, it's estimated that the packet overhead generated by the virtual node layer from a single region ranges between 200 bps and 450 bps. Because a node in a region can hear messages from up to 9 regions, the combined channel bandwidth overhead for any region can range between 2Kbps and 4.5Kbps, which is unlikely to affect the normal operation of other protocols on the mobile nodes. However, in terms of number of message overhead per allocation, the figure can be over 500 messages, where 90% of the overhead are the leader election messages.

D. Different Renewal Methods

The next problem is how to engineer the protocol to get the best performance. The renewal process is critical because when a renewal fails, a client has to stop using the current address and ongoing sessions may have to be disconnected. The number of addresses allocated to a client measures 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.

There are two important engineering choices for the renewal process. The first is how many hops a RENEW message is allowed to be forwarded. The second is whether flooding or unicast routing is used to forwarded the RENEW messages. Using a higher hop limit increases the chance a RENEW message reaches the server region and causes higher message overhead. Flooding generates more messages than unicast routing. A flooded RENEW message, however, is more likely to reach the server region because the single path picked by unicast routing may fail.

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. With hop limit 8, RENEW messages can be forwarded by up to 7 regions, including the client's local region.

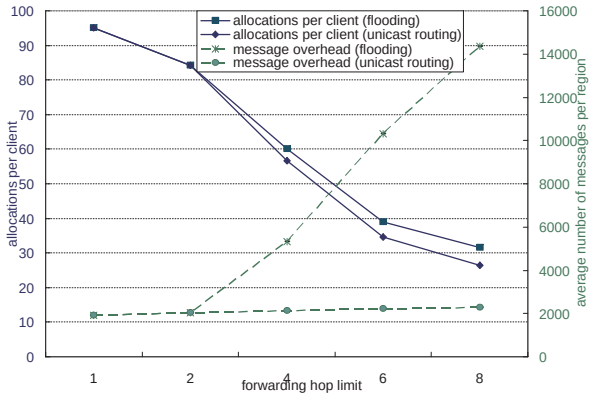

Fig. 4. Allocation performance with different renewal methods, fast moving case

1) 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 4. 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 needs a re-allocation.

For the flooding case, with larger hop limits, each client 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.

For the unicast 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. With these two hop limits, using unicast routing on the RENEW messages doesn't save anything. After that, the number of allocations per client decreases faster with higher hop limits. In addition, using unicast routing generates much less message overhead because only one forwarding path is used for every RENEW message. With hop limit 8, the unicast routing case generates less than one sixth of the message overhead of the corresponding flooding case.

Figure 5 shows the renewal message overhead and renewal delay. 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 larger hop limits, renewals take more messages and time to finish. With hop limit 8, a renewal takes around 55 messages, showing that most of the regions are involved in

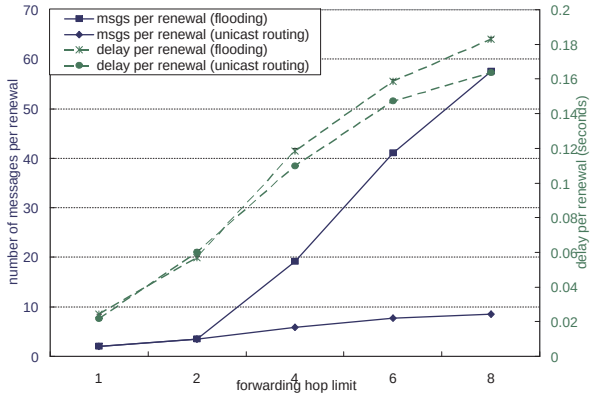

Fig. 5. Renewal overhead with different renewal methods, fast moving case

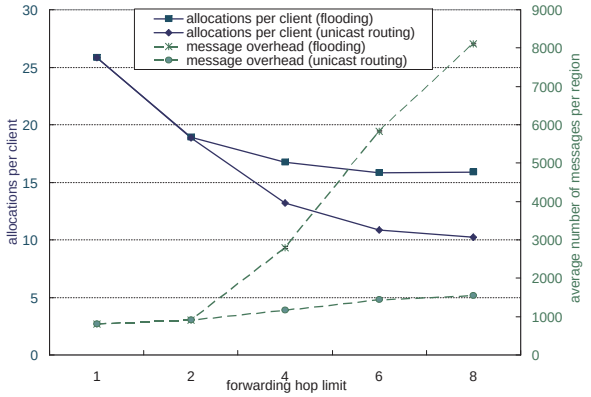

Fig. 7. Allocation performance with different renewal methods, slow moving case

the flooding of the RENEW messages.

With unicast routing, the renewal message overhead was 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.

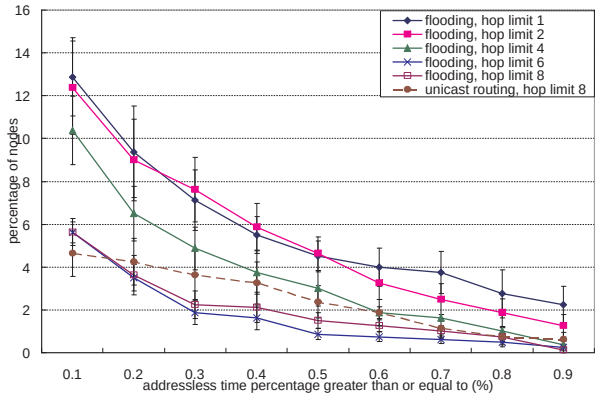

Fig. 6. Distribution of addressless time with different renewal methods, fast moving case

Figure 6 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 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, less than 4% of the clients don't have an address for more than 0.2% of the simulation time. The unicast 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.

2) Slow Moving Case: With speed mode "slow", we repeated the simulation. As shown in Figure 7, in the flooding case, the average number of allocations per client goes up with higher hop limits after hop limit 6. This means that when nodes are moving slowly, flooding of RENEW messages by many hops can hurt the allocation performance with heavy message overhead. In the unicast routing case, the number of re-allocations per client keeps getting lower with increasing forwarding hop limits while the message overhead increases much slower than the flooding case. Here, the unicast routing case performs much better than the flooding case.

The results above show that for this network setting, the addresses allocation protocol with simple unicast routing outperforms flooding under both speed modes. The virtual node based system performs the best when unicast routing is used for RENEW message and the forwarding hop limit is set to 8. With this parameter, the average allocation latency is 0.91 second and the average renewal latency is 0.16 second, with 160 fast moving nodes in the 64 region network.

E. Different Node Densities

Now we address scaling to larger number of 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 unicast routing used for the RENEW messages and the forwarding hop limit for renewal messages set to 5.

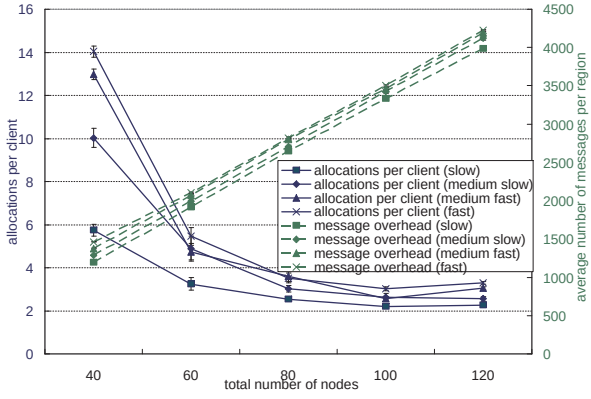

Fig. 8. Allocation performance with different node densities

From Figure 8, 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 the renewal messages now travels fewer hops.

The figure also shows that with higher node densities, the allocations per client decreases quickly at first and then gets stable or even rise up. 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 VNLayer-based system will perform the best with a node density that is neither too low nor too high.

With increasing node densities, the application layer message overhead increases linearly because each node introduces the same amount of application layer burden to the system. 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 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 to keep their addresses longer.

VI. Conclusion and Future Work

The simulation results for the example application demonstrate that the VNLayer approach can be used to adapt a wireline protocol like DHCP for use in a MANET. The message overhead introduced by the VNLayer approach is about 200 to 400 bits per second in terms of bandwidth. With most of the complexity handled by the VNLayer, the programs for the application server and the client are both simple.

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.

Our implementation of the VNLayer, VNSim is made available online as an ns-2 package. VNSim is discrete event-based, making it scalable. Simulation results showed that the C++ based simulator is suitable for a network of up to a few hundred mobile nodes. VNSim can be used to validate any VNLayer-based application.

The performance evaluation of the protocol with VNSim offered useful insights on how to engineer a protocol in a VNLayer-based system. For example, we found the use of a simple best effort geographical based unicast routing for RENEW message can outperform the case when flooding is used.

Leader election messages, especially the periodical Heartbeat message compose the largest part of the VNLayer message overhead. The leader election needs to be further engineered to reduce the message overhead and election delay. For example, knowing it has entered a different region, a leader can send a message back to its original region to ask the non-leaders to start leader election immediately. This can not only shorten the leadership switching delay, but also reduce the dependency on frequent sending of Heartbeat messages.

The state synchronization can also be engineered to reduce the message overhead because the SYN-ACK messages can be large. For example, state inconsistency tends to happen on multiple non-leader nodes in a region at the same time, a random backoff mechanism can be used by the non-leaders for their SYN-REQ messages and one SYN-ACK message can be used by every non-leader to update their states. In addition, the total number of mobile nodes emulating the virtual node can be reduced to a level such that consistent state can be maintained and the overhead of state synchronization can be minimized.

For the address allocation protocol, the routing mechanism for the RENEW messages needs to be improved to increase the success ratio of renewal attempts. A fail-over mechanism can be used to backup state variables of a virtual node to other virtual nodes so that even when the region is down, a client can still contact one of the backup virtual nodes to renew its address lease. Building on what we have learnt from this research and the VNSim package, we are continuing our research on VNLayer-based MANET routing protocols.

References

[1] J. Y. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," *in IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, First Qtr. 2005.

[2] M. Brown, S. Gilbert, N. Lynch, C. Newport, T. Nolte, and M. Spindel, "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, 2007.

[3] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, and T. Nolte, "Timed virtual stationary automata," in *9th International Conference on Principles of Distributed Systems (OPODIS 2005)*.

[4] S. Dolev, S. Gilbert, N. Lynch, E. Schiller, A. Shvartsman, and J. Welch, "Virtual mobile nodes for mobile adhoc networks," in *Proceeding of the 18th International Conference on Distributed Computing (DISC)*.

[5] 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.

[6] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *Network, IEEE*, vol. Nov/Dec 2001.

[7] R. Droms, "Dynamic host configuration protocol," *IETF Network Working Group, RFC 2131*.

[8] The network simulator - ns-2 http://www.isi.edu/nsnam/ns/.

[9] 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.

[10] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," in *proceeding of IEEE SICON'97*.

[11] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, 1999, pp. 90–100.

[12] R. Sivakumar, P. Sinha, and V. Bharghavan, "Cedar: a core-extraction distributed ad hoc routing algorithm," in *Infocom 1999*.

[13] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "Ip address autoconfiguration for ad hoc networks," *draft-ietfmanet-autoconf-01.txt, IETF, MANET Working Group*.

[14] S. Nesargi and R. Prakash, "Manetconf: Configur-ation of hosts in a mobile ad hoc network," in *Proceedings, IEEE Infocom 2002*.