Dynamic Routing Framework for OMNET++ Based Hardware-in-the-Loop (HITL) Network Simulation

Gohel Nirav and Krishna M. Sivalingam
Department of CSE, Indian Institute of Technology Madras, Chennai-600036
Email: gohel.nirav@gmail.com, skrishnam@iitm.ac.in, krishna.sivalingam@gmail.com

***Abstract*—This paper presents a dynamic routing framework for Hardware-In-The-Loop (HITL) simulation based on the OMNeT++ simulator. A Hardware-In-The-Loop (HITL) network simulation connects the real and the virtual (simulated) networks. The routing protocol studied is Optimized Link State Routing (OLSR) that is used primarily for mobile ad hoc network (MANET) routing. The OMNeT++ simulator provides a basic HITL framework to connect a real network with the simulator. The existing HITL framework of OMNeT++ supports only static routes, where routes are added manually for virtual and real nodes. In a MANET, the network nodes are mobile and hence the routing tables are dynamically updated. In order to enable HITL simulations for MANET systems, there is a need to provide dynamic routing in HITL. The routing protocol used in a simulation study should run across virtual and real networks and build routes dynamically. In this paper, we describe the architecture and implementation for achieving dynamic routing using the OLSR protocol. Different scenarios with various combinations of virtual and real nodes to validate dynamic routing in HITL have been studied, with network sizes up to 36 nodes.**

I. Introduction

Discrete-event simulation is a commonly used technique to model the behaviour of computer network protocols and systems. In typical simulation models, all the system components are modeled in the simulation environment. However, such simulations are less detailed and do not provide realistic results especially with respect to computation time and other real-time behavior. It is also often required to connect existing hardware or software based systems (e.g. a router or a switch) to a simulated environment for an integrated system study. This could be used for interoperability testing of a protocol or an application over a real network. In such cases, it is necessary for the simulation system to provide hardware-in-the-loop (HITL) or system-in-the-loop (SITL) capabilities.

For example, a real network is required for testing an application where a transmitter sends a video stream to a receiver over a network. HITL can be used for this purpose, where a video stream transmitter and a video stream receiver are deployed on real nodes and these nodes are connected via a virtual network (running inside the simulator). Similarly, HITL can also be used for scalability testing. In this case, HITL can be used to replace a large network with a virtual network that contains a large number of nodes.

In HITL network simulation, real network elements are connected to the simulated network as shown in Figure 1.

978-1-4799-2361-8/14/$31.00

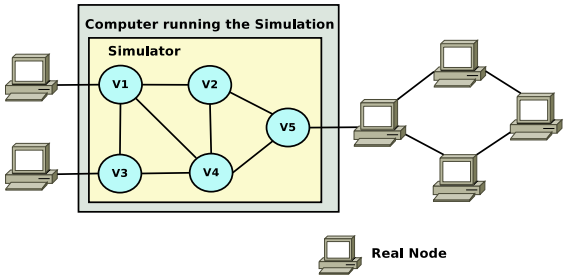

Fig. 1. Hardware-in-the-Loop (HITL) Simulation.

The simulated network consists of nodes labeled V1 through V5. Packets are passed between these two different types of networks and the entire configuration acts as a single network. Thus, some part of the network is real and some part is virtual. Hence, we can relate these results to real and virtual networks and find a transformation method to convert the simulation results of the virtual network to realistic results. As an example, *ping* application timing is studied in [1] using a HITL approach.

OMNeT++ is an open source, discrete event simulator that provides a generic framework for various kinds of simulations including computer networks [2]. INET is an open-source computer network simulation framework built on OMNeT++. It also provides HITL capabilities to connect a real network with the simulated network [3]. The commercial OPNET network simulator also provides a similar facility termed as System-In-The-Loop (SITL) [4]. Since it is not open source, it is not considered here.

In this paper, we extend the OMNET++/INET HITL simulation framework by implementing the necessary functionality for dynamic routing in mobile ad hoc networks (MANETs) that use the Open Link State Routing (OLSR) dynamic routing protocol. Routing in HITL in OMNeT++ simulator is presently done by configuring static routes in the virtual and the real nodes. In a MANET, the topology changes dynamically and hence support for dynamic routing is needed. The objective of this paper is to enable dynamic routing in the HITL framework of OMNeT++ using the Optimized Link State Routing (OLSR) protocol such that it can run across virtual and real networks and establish routes dynamically.

There are two technical challenges that arise when connecting a real network with the simulator: packet translation and

time synchronization. A packet inside the real network is in the form of a bit stream, whereas a packet inside a virtual network is an object. Whenever a packet flows from a real node to a virtual node, it needs to be converted from a bit stream to an object and vice versa.

Synchronization is required in order to give a real time response to a real network (that is connected to the simulator). If the simulated and real clocks are not synchronized, then the simulation may encounter a packet which should have been processed in the past (the time that is less than its clock time) or the simulator may process a packet earlier than the time at which it should be processed. Thus, time synchronization is an important requirement; however, it is outside the scope of this work.

Solutions have been designed to address the dynamic routing challenges and implemented in the OMNET++/INET framework. The framework is then tested for various dynamic routing scenarios where two real networks are connected via a virtual network (Live-Sim-Live) and two virtual networks are connected via a real network (Sim-Live-Sim), for networks with up to 36 nodes.

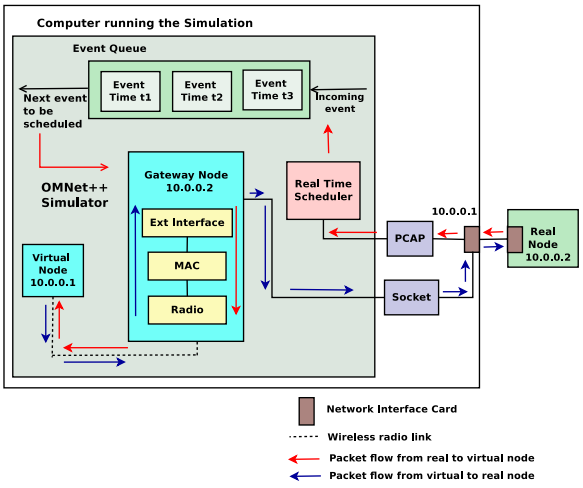

Fig. 2. Packet flow between a virtual and a real node in HITL.

II. Basics of OMNeT++-HITL

This section describes how HITL is presently implemented in the existing OMNET++ network.

OMNeT++ provides a gateway node to connect the simulator with a real network. For connecting a virtual node to a real node, the virtual node is connected to a gateway node, the gateway node is connected to the Network Interface Card (NIC) on the computer (running the simulator) and the real node is connected to this computer via its own NIC. This is shown in Fig. 2. The gateway node is a virtual node inside the virtual network that performs packet translation. When it receives a packet form the NIC, it converts the bit stream (of the packet) into an object (that is a packet in the virtual network) and passes it to the virtual node. Similarly, when it receives a packet (object) from a virtual node, it converts it into a bit stream (that is a packet in real network) and passes it to the NIC.

A. Packet flow from a real node to a virtual node

The packet flow from a real node to a virtual node is shown in Figure 2. The real node converts the packet into a bit stream and sends it via the network interface card (NIC). The bit stream (packet) arrives at the NIC of the computer (running the simulation) as shown in Figure 2. *PCAP* is a driver installed in the computer running the simulation. It listens to all the packets arriving to the NIC. The *cSocketRTScheduler* is a scheduler that is used specifically for HITL simulation and it synchronizes the simulation with the real network.

The cSocketRTScheduler registers itself to the PCAP device to get packets from a specific NIC with a specific filter and hence it gets the incoming packets. It marks the arrival time of the packet and inserts it into the event queue. The event queue is basically an implementation of priority queue. The scheduler picks up the event from the queue when its turn comes and passes it to the gateway node. The *ExtInterface* module inside the gateway node converts the packet from a bit stream (sent by the real node) to an object and passes it to a virtual node via a radio as shown in Figure 2. Hence, when the packet is received by the virtual node, it assumes that it is sent by another virtual node and processes it in the same way as it does with a virtual packet.

B. Packet flow from a virtual node to a real node

The packet flow from a virtual node to a real node is also shown in Figure 2. The virtual node passes a packet to the gateway node in the same way as it does to the other nodes inside the simulation as shown in Figure 2. The packet is received at the radio level inside the gateway node. After discarding the MAC header inside MAC layer, it is passed to ExtInterface. The ExtInterface inside the gateway node again takes care of the packet conversion. It converts each field of the packet object into a bit stream of an appropriate size. Then it merges these bit streams into an appropriate order that is defined by the protocol and writes the packet (bit stream) onto a socket. The packet reaches the NIC of the computer (running the simulation) and is then received by the real node via its NIC as shown in Figure 2. Hence, the real node assumes that the packet is sent by another real node.

C. Static Routing

Routing in HITL in OMNeT++ simulator is done by configuring static routes. Static routes are configured in both the virtual and the real nodes before running the simulation. This is done by adding the routes in a file inside the simulator and by adding the routes in the operating system routing table on the real node. These routes are hard-coded. As shown in Figure 3, the simulated network runs a routing protocol, but static routes have to be configured for connecting the real node to the virtual node. However, the network topology may change at runtime in networks as in ad hoc networks. In order

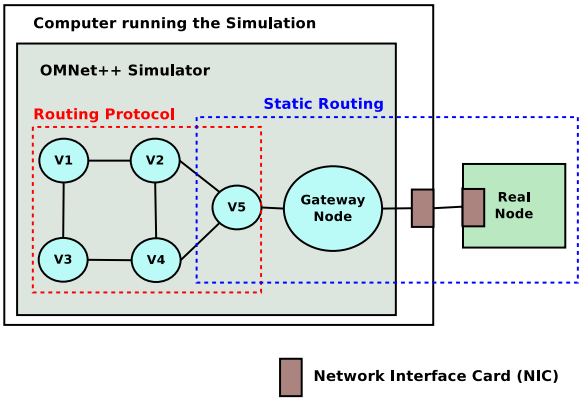

Fig. 3. Routing mechanism in HITL in OMNeT++ simulator.

to run HITL simulations for such networks, there is a need of dynamic routing in HITL in OMNeT++. This forms the motivation for this work.

III. Dynamic routing in OMNeT++-HITL

This section describes how the OLSR routing protocol for MANET networks has been implemented in the OMNET++ framework.

A. Optimized Link State Routing (OLSR)

Mobile ad hoc networks (MANETs) have been studied for nearly two decades. Here, a set of mobile nodes communicate with each other over wireless network interfaces. MANETs have several applications in both civilian and military scenarios. One of the main challenges in a MANET network is for a source node to dynamically determine the route to a destination node. There are several MANET routing protocols such as Adhoc On Demand Distance Vector (AODV) and Dynamic Source Routing (DSR). Due to space limitations, these are not described here.

Optimized Link State Routing Protocol (OLSR) is a modification of the Link State Routing protocol and designed to meet MANET specific requirements [5]. It is a proactive routing algorithm. It periodically updates routes so that routes are immediately available when needed. In case of OLSR, only a selected set of nodes, called Multi Point Relays (MPR), forward the route request messages in order to reduce the message overhead [5]. Only the MPR nodes generate the link state information thereby minimizing the overhead of control message transmissions. The OLSR protocol communicates using 3 types of messages: HELLO, TC and MID [5], as described below:

- HELLO: HELLO messages are used for link sensing, neighbor detection and MPR selection signaling. HELLO message is generated for all the interfaces of a node and it contains information about all the neighbor interfaces connected to this interface.
- TC (Topology Control): TC messages are used for topology discovery. TC message contains main addresses of all the neighbor nodes of the originator node. This information is used along with the link sensing and neighbor detection to construct routes.
- MID (Multiple Interface Declaration): MID messages are used to announce the information about running OLSR on multiple interfaces on the same node. When a node has multiple interfaces, one of its interface will be main address. Hence this relationship between the OLSR interface addresses and the main address is announced through MID messages.

B. Packet translation of OLSR packets

In order to achieve dynamic routing in HITL, we have implemented packet translation of OLSR packets inside the gateway node. The real node runs the OLSR daemon [6] (an implementation of OLSR for a real network) and the virtual node runs the OLSR protocol provided by OMNeT++. Both the nodes exchange the routing packets and subsequently build routes to each other. We have implemented the packet translation for all three types of OLSR packets (Hello, TC and MID). As mentioned earlier, the module ExtInterface inside the gateway node performs the task of packet translation.

The conversion of each field of a packet also requires byte order conversion if necessary. The nodes in a network communicate with each other in a commonly agreed order that is called “Network Byte Order”. However, each node has its own predefined byte order (to represent the multi-byte data) called Host Byte Order that may be different from the network byte order. Whenever a node receives a packet, it needs to convert each multi-byte field from network byte order to host byte order. The same task is done when sending the data to the network. No conversion is required for single byte data.

In case of the conversion from a real packet (a bit stream) to a virtual packet (an object), the *ExtInterface* breaks the bit stream into appropriate segments of various sizes according to the packet format defined by the protocol, converts each multi-byte segment into host byte order (from network byte order) and populates these values into an object in appropriate fields. This object is then sent as a virtual packet in the virtual network. In case of the conversion from a virtual packet to a real packet, *ExtInterface* extracts values from all the fields of the object, converts each multi-byte field into network byte order (from host byte order) and populates all the values into a single bit stream according to the packet format defined by the protocol.

Since OLSR communicates using UDP packets, OLSR packet conversion is done as a part of UDP packet conversion. OLSR uses port number 698 and hence all the incoming UDP packet with source and destination as 698 are considered as OLSR packets and converted to virtual OLSR packets. For serialization of outgoing OLSR packets, the transport layer protocol is checked inside the simulator, they will be serialized and will be sent as UDP packets. The OLSR packet header contains the packet length and packet sequence number. OLSR messages have a common header format. A packet can contain

more than one message of same/different types. Each type of message body has to be translated separately according to the message format. The complete translation is implemented inside the *ExtInterface*.

C. Using Virtual IP inside virtual network

The IP address of the computer (hosting the simulation) should be same as the IP address of the virtual node while connecting it to a real node in HITL. This indicates that the computer running the simulation must have a separate Network Interface Card (NIC) for each virtual node that is connecting to the real node. This problem is solved using a virtual IP address (i.e. IP aliasing) inside the computer running the simulation. A virtual IP is an IP address assigned to applications that are running on a single computer rather than being assigned to a specific NIC. A single NIC can have multiple IP address used by multiple applications. For HITL simulations, we are adding a virtual IP (on the computer running the simulation) for each virtual node that is connected to a real node and the socket that is sending data for that virtual node is bound to that particular virtual IP address. This strategy is same as IP aliasing. ExtInterface binds the socket to this IP address. As shown in Figure 4, the source address of the packets going through this virtual node will have IP address 10.0.0.1 and the real node connected to it will assume that there is a real node with IP address 10.0.0.1. Similarly, we need an NIC on the real node for each virtual node to which it connects. This problem has been solved using virtual machines having multiple Ethernet cards. Additional implementation details are available in [7].

IV. Validation and Testing

This section describes the different simulation scenarios studied and the results. The aim of the simulations is to ensure that each node (virtual and real) successfully builds routes to all the other nodes (virtual and real).

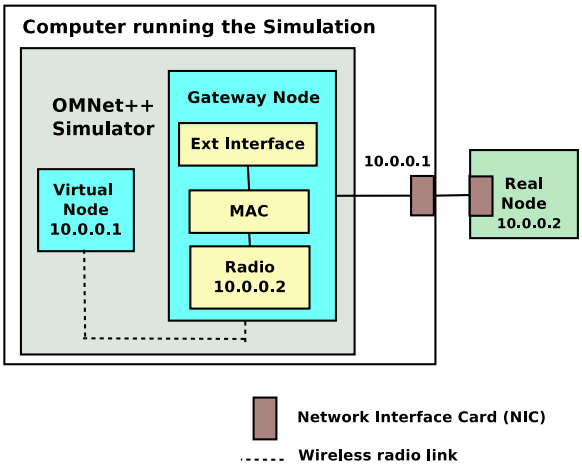

Fig. 4. Configuring a HITL network to run OLSR.

Figure 4 shows a simple example of connecting one virtual node with one real node both running OLSR. A virtual node with IP address 10.0.0.1 is connected to a real node with IP address 10.0.0.2 via a gateway node. The radio interface of the gateway node is assigned IP address same as the real node. Hence, the virtual node will then forward the packets to gateway node assuming that it is another virtual node that is running OLSR with IP 10.0.0.2. Similarly, the NIC of the computer that is running simulation is assigned the same IP address as the virtual node. Hence the real node running OLSR (10.0.0.2) assumes that there is a real node with IP address 10.0.0.1 that is sending OLSR packets. However, this indicates that separate NIC is required for connecting each virtual node to a real node. This problem is solved by using virtual IP (for virtual nodes) and virtual machines (for real nodes).

A. One Virtual node and One Real node with multiple interfaces

This scenario connects a virtual node with a real node via a gateway node as shown in Figure 5. The interface 10.0.0.1 of virtual node is connected to the interface 10.0.0.2 of the real node. However, the virtual node has an additional interface with IP address 20.0.0.1 and the real node has an additional interface with IP address 30.0.0.1. Both the nodes send OLSR packets to each other and eventually build routes. Figure 5 also shows the routes established by the virtual node to both the interfaces of the real node and the routes established by the real node to both the interfaces of the virtual node. The Hello interval time set for this scenario is 2 seconds, MID interval time is 5 seconds and TC interval time is 5 seconds.

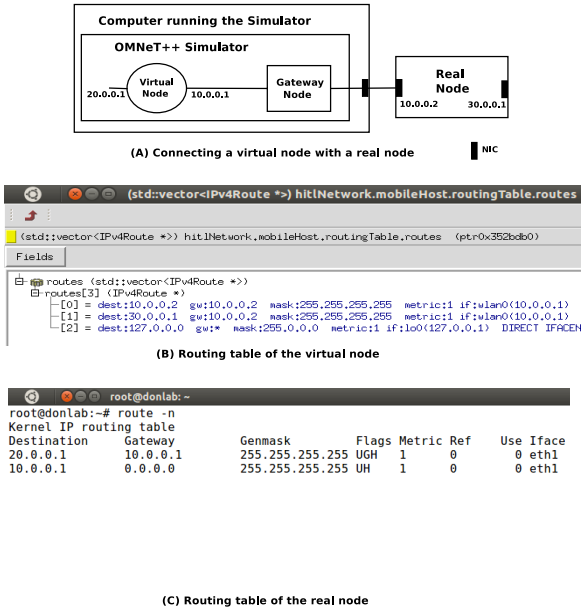

Fig. 5. Scenario 1: One virtual node and one real node with multiple interfaces.

B. 12 Virtual nodes and 4 Real nodes

This scenario aims to test the 4x4 nodes grid with 12 virtual nodes (node 0 to node 11) and 4 real nodes (R1 to R4) as

shown in Figure 6. The virtual nodes 2, 5, 8 and 11 are connected to the real nodes 1, 2, 3 and 4 respectively as shown in Figure 6. It is observed that each node (virtual and real) builds routes to all the other nodes. The Hello interval time set for this scenario is 2 seconds, MID interval time is 5 seconds and TC interval time is 5 seconds.

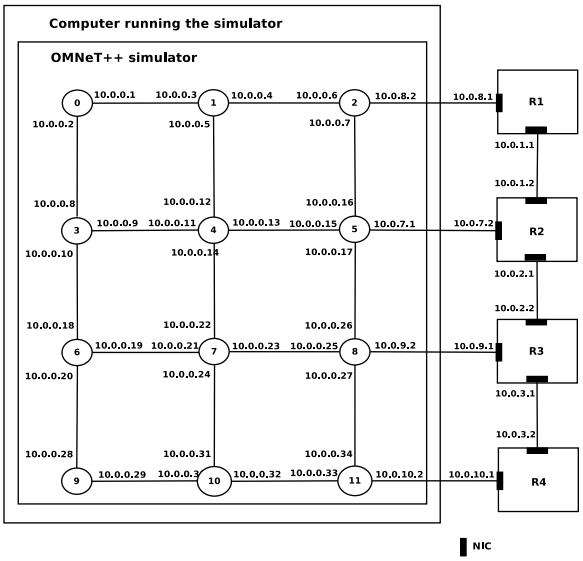

Fig. 6. Scenario 2: 12 virtual nodes and 4 real nodes

C. Live-Sim-Live Scenario of 16 nodes

This scenario aims to test 4x4 nodes grid in a Live-Sim-Live configuration where two live networks are connected via a virtual network. As shown in Figure 7, a real network with real nodes (R1 to R4) is connected to another real network of 4 real nodes (R5 to R8) via a simulated network of 8 nodes (node 0 to node 7). The virtual nodes 0, 2, 4 and 6 are connected to the real nodes R1, R2, R3 and R4 respectively as shown in Figure 7 and the virtual nodes 1, 3, 5 and 7 are connected to the real nodes R5, R6, R7 and R8. It is observed that each node (virtual and real) builds routes to all the other nodes.

It is observed that if we set the packet generation interval times same as the previous scenarios (Hello - 2 seconds, MID - 5 seconds and TC - 5 seconds), some of the nodes are not building routes to all the other nodes. For example real node 8 does not build routes to real node 1. This problem occurs due to the lack of synchronization between real and virtual networks. When a node receives an OLSR packet from another node, it expects the next OLSR packet from the same node within this time interval. However, in this scenario, a packet can not reach from one real network to another real network via a virtual network within these interval times and hence it does not build the routes. The interval times are increased to achieve synchronization between all three networks. Hence, the Hello interval time set for this scenario is 5 seconds, MID interval time is 8 seconds and TC interval time is 8 seconds.

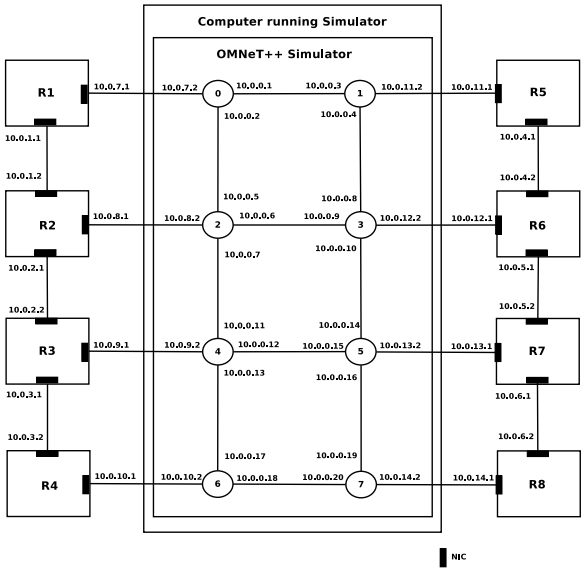

Fig. 7. Scenario 3: Live-Sim-Live Scenario.

D. Sim-Live-Sim Scenario of 16 nodes

This scenario aims to test the 4x4 nodes grid in a sim-live-sim scenario where two virtual networks are connected via a real network. As shown in Figure 8, a virtual network 1 with 4 virtual nodes (node 0 to node 3) is connected to another virtual network 2 with 4 virtual nodes (node 0 to node 3) via a real network of 4 nodes (R1 to R4). It is observed that each node (virtual and real) builds routes to all the other nodes.

It is observed that if we set the packet generation interval times same as the previous scenarios (Hello: 2 seconds, MID: 5 seconds and TC: 5 seconds), some of the nodes are not building routes to all the other nodes. This problem occurs due to the lack of synchronization between real and virtual networks. When a node receives an OLSR packet from another node, it expects the next OLSR packet from the same node within this time interval. However, in this scenario, a packet cannot traverse from one virtual network to another virtual network via a real network within these interval times and hence it does not build the routes. The interval times are increased to achieve synchronization between all three networks. Hence, the Hello interval time set for this scenario is 5 seconds, MID interval time is 8 seconds and TC interval time is 8 seconds.

E. Scalability in HITL

This scenario aims to test the scalability of HITL. We have tested a network with 36 nodes placed in a 6x6 nodes grid. As shown in Figure 9, one node is kept as a real node and the other 35 nodes are kept as virtual nodes (from node 0 to node 34). It is observed that each node (virtual and real) builds routes to all the other nodes.

It is observed that if we set the packet generation interval times same as the previous scenarios (Hello: 2 seconds, MID: 5 seconds and TC: 5 seconds), then the simulator runs far behind the real time. This problem occurs as we have

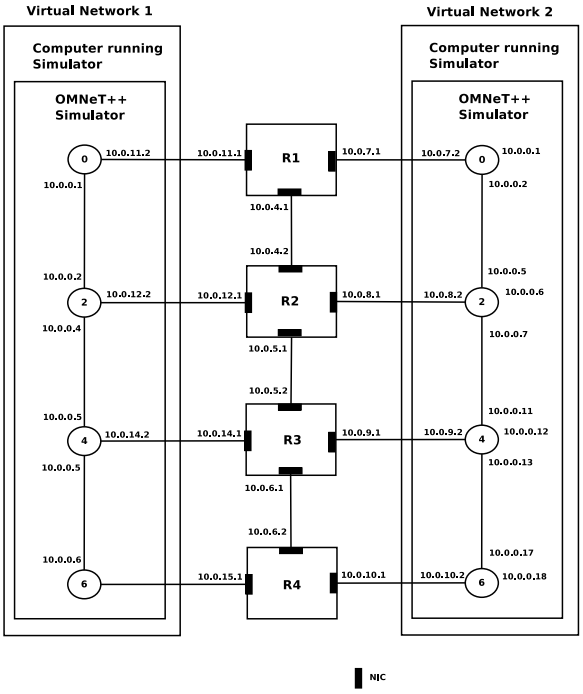

Fig. 8. Scenario 4: Sim-Live-Sim Scenario.

increased the number of nodes inside the simulator. It has been observed that the simulation runs only up to 15 minutes (of simulation clock) in 45 minutes (of real time). Hence, there is lack of synchronization between real and virtual networks. If we increase the packet generation rate, the number of events inside the simulator will be decreased and hence the simulator will run fast. We tried to increase it and achieved the synchronization between the real node and the simulator with the values: Hello interval: 10 seconds, MID interval: 15 seconds and TC interval: 15 seconds.

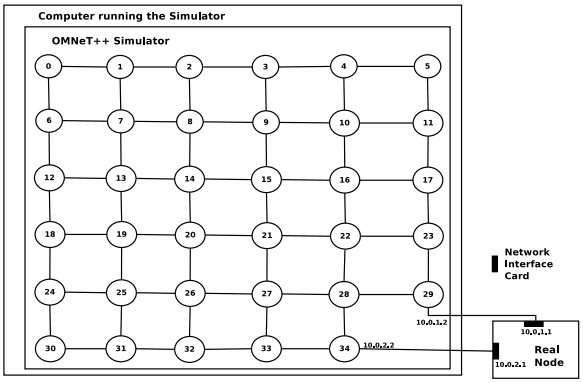

Fig. 9. Scenario 5: Scalability in HITL.

The packet generation rates to achieve route establishment in various scenarios are shown in Table I. These may be considered as representative values.

TABLE I
PACKET GENERATION INTERVAL IN VARIOUS SCENARIOS.

Scenario	Hello Interval	TC Interval	MID Interval
A	2	5	5
B	2	5	5
C	5	8	8
D	5	8	8
E	10	15	15

V. CONCLUSIONS

This paper presents a framework to enable OLSR based dynamic routing in Hardware-in-the-Loop (HITL) as part of the OMNET++ framework. The goal was to establish routes at run-time in order to conduct mobile ad hoc network studies using HITL. We tested dynamic routing in HITL in various scenarios with various combinations of real and virtual nodes. Each of the nodes (real and virtual) in these scenarios establish routes to all the other (real and virtual) nodes. We tested dynamic routing in scenarios where two real networks are connected via a virtual network (Live-Sim-Live) and two virtual networks are connected via a real network (Sim-Live-Sim). However, there is a limitation in this method that the packet generation rate cannot be decreased below a certain number. This depends of the computation power of the computer running the simulator. If the packet generation rate is decreased below a certain threshold, the simulator runs behind the real network and lacks synchronization. Hence the packet generation rate can be decreased only upto a level where the simulator runs in synchronization with the real network and hence it depends on the computer that is running the simulator. In future, the scenarios where topology changes dynamically due to node mobility can be tested. The main problem in these kinds of scenarios is to achieve time synchronization between the real and the virtual networks. Realistic results for other metrics such as throughput and delay can also be obtained.

Acknowledgments: This work forms a part of a RBIC Project jointly developed along with The Tata Power Company, Strategic Engineering Division (TPC-SED). The authors are thankful to the staff at TPC-SED for their feedback during this work.

REFERENCES

[1] S. Yafen, C. Jiaxi, Y. Jing, and H. Ning, "Reliability Analysis of System In The Loop Network Platform Based on Delays," in *Seventh International Conference on Computational Intelligence and Security*, Dec 2011, pp. 750–753.
[2] A. Varga and O. Ltd, "OMNeT++ User Manual Version 4.2.2," 2011.
[3] INET, "INET Framework for OMNeT++ Manual," 2012.
[4] OPNET, "OPNET System In The Loop Module," 2012. [Online]. Available: http://www.opnet.com/solutions/network-rd/system-in-the-loop.html
[5] T. Clausen and P. Jacquet, "RFC 3626 : Optimized Link State Routing Protocol (OLSR)," http://www.ietf.org/rfc/rfc3626, October 2003.
[6] A. Tonnesen and T. Lopatic, "OLSR daemon." [Online]. Available: http://www.olsr.org
[7] G. Nirav, "Dynamic routing framework for hardware-in-the-loop (HITL) based network simulator," Master's thesis, Indian Institute of Technology Madras, Jul. 2013.