A Simulation Framework for Performance Analysis of Multi-Interface and Multi-Channel Wireless Networks in INET/OMNET++

Heywoong Kim[1], Qijun Gu[1], Meng Yu[2], Wangyu Zang[2], Peng Liu[3]
[1] **Texas State University-San Marcos, San Marcos, TX 78666, USA**
[2] **Western Illinois University, Macomb, IL 61455, USA**
[3] **Pennsylvania State University, University Park, PA 16802, USA**

Keywords: Multi-channel network, Multi-radio network, Wireless network, Channel assignment protocol, Protocol simulation, OMNET++, INET, Simulation framework

Abstract

In wireless networks, devices can be equipped with multiple interfaces to utilize multiple channels and increase the aggregated network throughput. Various channel assignment protocols have been developed to better utilize multiple channels and interfaces. However, the research of channel assignment protocols is still lack of a good simulation tool that can content with a variety of requirements and specifications of channel assignment protocols and conduct data collection and performance analysis. In this paper, we propose MIMC-SIM, a generic simulation framework for multi-interface and multi-channel networks, built in INET/OMNET++. In MIMC-SIM, we put a new layer between the network layer and the MAC layer to provide a generic code structure and flexible extension for implementing channel assignment protocols. We also examine one of the channel assignment protocols within our simulation framework to study its performance.

1 INTRODUCTION

In a wireless network, devices can be equipped with multiple interfaces working on multiple channels. For example, IEEE 802.11b has 11 channels in the 2.4 GHz spectrum, 3 of which are orthogonal, and IEEE 802.11a has 13 orthogonal channels in the 5 GHz spectrum. Such wireless networks of multi-interface multi-channel (MIMC) devices are known as MIMC wireless networks. They are used to build mesh networks [12], vehicular ad hoc networks [17], and so on.

Because packet transmissions on these orthogonal channels do not interfere, various channel assignment (CA) protocols have been developed to better utilize the channels and the interfaces in a MIMC network. The CA protocols allow nodes to exchange their channel and traffic information, collaborate on channel assignment negotiation, and assign channels to nodes to reduce interference in transmission. The research of channel allocation in the literature has emphasized how to maximize the network capacity and channel utilization [15, 11]. The design of CA protocols has been studied in layered mesh networks [18, 14] and peer ad hoc networks [16, 19]. All the research and development efforts on MIMC networks clearly indicate the importance and necessity of developing a simulation framework to study various problems in MIMC networks and evaluate and compare performance of proposed new schemes.

Although quite a few emulation testbeds and simulation tools have been developed for studying wireless networks, they are still not sufficient yet to satisfy the needs of MIMC network research. Several deployed wireless testbeds [8, 1, 9, 4] can be used to validate some wireless protocols. However, nodes in these testbeds mostly have only one radio, even though they use multiple channels. The testbeds can only emulate a network with a limited scale. The topology of the nodes is hard to change, and node mobility can hardly be studied in these testbeds. Meanwhile, a few simulation tools have been developed [5, 7, 3, 2], which can address the problems in the wireless testbeds. They can support large-scale simulation of various protocols in wireless and mobile networks. However, to the best of our knowledge, no general simulation framework has been actually developed for MIMC networks. In the existing research of MIMC networks, the few simulations are only specific to the proposed schemes. Even though some simulation tools have partially added mechanisms for supporting multiple interfaces and multiple channels, they have not examined truly the needs of MIMC network simulation that will be discussed shortly in Section 3.2.

In this paper, we present MIMC-SIM, a generic simulation framework for MIMC networks, built in INET/OMNET++. The features of the MIMC-SIM framework include (i) providing a generic and flexible code structure, (ii) supporting a variety of CA protocols, (iii) allowing easy extension according to protocol specifications, (iv) working compatibly with protocols at the MAC and the network layers, and (v) adapting a variety of factors in simulation, such as network topology and mobility. The MIMC-SIM framework will contribute to the research and development of MIMC networks.

The rest of the paper is organized as follows. Section 2 gives the background of CA protocols and the INET/OMNET++ simulation framework. Section 3 discusses the challenges of MIMC network simulation and overviews the architecture of MIMC-SIM. Section 4 presents the details of major modules in the MIMC-SIM framework. Section 5 shows the implementation and evaluation of a CA protocol in the MIMC-SIM framework. Finally, the paper concludes in

Section 6.

2 BACKGROUND

In this section, we briefly summarize a few major channel assignment protocols that aim to improve network capacity by reducing channel interferences. Then, we give a short overview on OMNET++ and INET.

2.1 Channel assignment protocols

In a MIMC wireless network, with a dedicated interface per channel, a node can use all available channels simultaneously. However, considering cost and small size, normally the number of interfaces, m, of a node should be smaller than the number of channels, c. It is showed [15] that the network capacity is affected by the ratio of c to m, rather than the number c or m. When c/m is $O(logn)$ in a random network, network capacity will not be degraded. The work also provides the boundaries of network capacity that CA protocols can achieve when $m < c$.

To improve network capacity, several CA protocols were developed. In [16], the radios of a node are divided into two categories: fixed receiving channels and switchable channels for capacity increase. For the fixed channel assignment, a node randomly selects a channel as its initial receiving channel. Later, the node may change its fixed channel to a less used channel to reduce interference. The switchable channels are used to ensure connectivity. Obviously this fixed channel assignment takes time to converge. In addition, if the number of switchable channels is large, the switching channel delay may be large when the node needs to switch back and forth to communicate with different neighbors.

In [19], CA algorithms based on s-disjunct superimposed codes were proposed to mitigate co-channel interference of network capacity maximization. For each node, all orthogonal channels are labeled as either 1 for primary or 0 for secondary via a binary channel codeword. Then, a node, u, first searches a set of primary channels that are secondary to all interferers in two-hop communication range since these channels may not be used by the interferers. If the searching fails, u chooses the secondary channels that are not primary to any of interferers since the interferers may not use them either. If u cannot find such channel, it picks up the primary channels that are primary to the least number of interferences.

In [18, 14], CA protocols were proposed specifically for wireless mesh networks. [18] considers the channel assignment problem as two sub problems: 1) interface assignment problem where interfaces of a node are divided into two categories: UP-NICs used for communicating with its parent node, and DOWN-NICs used for communicating with its child nodes; 2) interface-channel assignment problem where the channel assignment of a node's UP-NICs is determined by its parents. A less loaded channel will be assigned to a DOWN-NIC to prevent the interference. A node periodically reevaluates its current channel usage and switches a heavily-loaded channel to a less loaded channel. The channel assignment of a node relies on its parents. The parents always have higher priority than the children. A node close to a gateway will pick a channel earlier than those further away.

In [14], a distributed CA protocol is proposed for dual-radio mesh network. Each gateway in the mesh network associates a channel sequence with each of its radios. The channel sequence is propagated along with routing information in periodic route announcement messages. A node obtains its two channels based on the channel sequence and the distance (hops) to the gateway. The nodes on the same hops from a gateway share one channel in common, then all paths to the gateway can operate on distinct channels to eliminate intra-path interference. Compared with [18], the CA approach does not rely on the parent. However, if a gateway changes its channel sequence, the nodes connected to the gateway need to change channels accordingly.

2.2 OMNET++ and INET

OMNET++ [6] is an open source discrete event simulation environment. It is not a simulator of any particular system, but rather provides a generic and flexible architecture for writing simulation tools. It has been used to model and simulate communication networks, operating systems, hardware architectures, distributed systems, and so on.

The fundamental ingredients of OMNET++, that distinguish it from other simulators, are its object-oriented component architecture and message passing mechanism. Components (modules) are programmed in C++, then assembled into larger components using a high-level language (NED). Modules are connected via gates. Modules do not directly call other modules' functions. Instead, they interact by passing messages through the gates. Messages may carry arbitrary data structures, including any packets for network communication. Such architecture allows developers to encapsulate a set of functions in a component and also resembles the actual interaction among individual entities in distributed systems.

The INET framework [3] is an open-source communication network simulation package built in OMNeT++. The INET framework contains models for various networking protocols, such as UDP, TCP, IP and IEEE 802.11. Protocols are represented as modules. The modules are then combined to construct hosts and network devices including router, switch, access point, etc.

Inside a host, the modules for protocols are connected according to their layers in the networking model. For example, Figure 1 shows the internal structure of a wireless node. The modules at the top layer represents applications. The modules in the middle implement protocols at the transport and the network layers. The module at the bottom layer is a com-

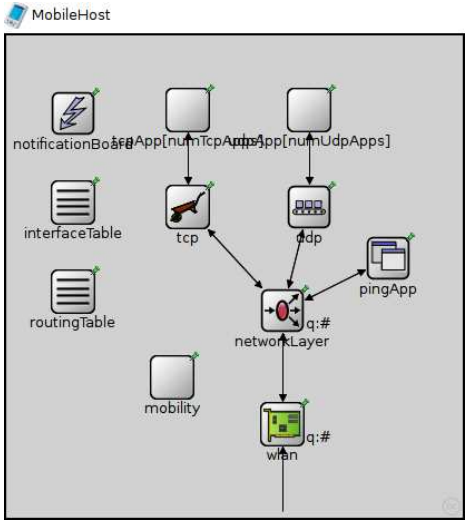


Figure 1. Mobile Host Structure in INET

pound module, resembling a network interface card (NIC) in the host and implementing protocols at the link and the physical layers. Nevertheless, not all modules implement protocols though. Some modules hold data (for example *routingTable*), facilitate communication of modules (*notificationBoard*), and move a mobile node around (*mobility*).

Various extensions have been added into INET. INETMANET is a project to model mobile ad hoc network protocols in the INET framework, and OverSim is a project to model overlay and P2P network protocols. Our MIMC-SIM framework is also an extension in INET to model MIMC network protocols.

3 OVERVIEW OF MIMC-SIM

In this section, we discuss the assumptions used by MIMC-SIM, the main issues and challenges in designing MIMC-SIM, and the overall architecture of MIMC-SIM.

3.1 Assumptions

The MIMC-SIM framework assumes a MIMC network utilizes multiple orthogonal channels. In the current implementation of INET, this assumption is well supported by the signal propagation model adapted in the radio module. A signal delivered in one channel will not contribute anything to another orthogonal channel. In the future, the radio module can be modified to adapt a better signal model to capture the major characteristics of signals in overlapping channels.

The MIMC-SIM framework assumes all NICs are using the same communication protocol, or compatible protocols in the same protocol family. For example, in a mesh network, a node can be equipped with two NICs. One NIC may work on IEEE802.11b and the other may work on IEEE802.11g. The assumption implies that a packet transmitted in a channel could be delivered to all NICs in that channel. If different communication protocols with overlapping channels were used, a packet of one protocol being transmitted in a channel will become a noise signal to other protocols using the same channel. The current radio module in INET does not support concurrent multiple communication protocols.

MIMC-SIM also assumes the number of channels is greater than the number of NICs in each node. Researchers [14, 13, 10] have showed that multiple NICs of a node should be separated by at least 18 inches so that their radio transmission will not interfere each other even though they use orthogonal channels. Hence, given the limit size of most mobile devices, a node will only have a few NICs (two or three). Whereas, wireless networks often have more orthogonal channels. For example, IEEE802.11b/g has 3 orthogonal channels, IEEE802.11a has 13, and IEEE 802.15.4 has 16.

3.2 Challenges and Issues

Although INET can support partially multiple interfaces and multiple channels in network simulation, quite a few challenging issues remain unaddressed for MIMC network simulation due to two major reasons. One is that the wireless framework in INET was designed for simulating wireless communication in one channel. Even though it allows NICs to use multiple channels, it assumes that all NICs of the same host will use the same channel and work on the same mechanism in simulation. The other reason is that INET handles multiple NICs in wireless communication directly based on the model of wired network, which simply makes the NICs forward packets over separated communication links. In a MIMC network, such a model ignores the collaboration among the NICs and thus cannot be used to support MIMC simulation. The MIMC-SIM framework is designed to address the following major issues.

First, CA protocols assign channels to nodes in various ways and assign various roles to NICs accordingly. The MIMC-SIM framework is designed to support two major categories of CA protocols. One category is node-based channel allocation [16, 19] that assigns a set of channels to each node, which then assigns a channel to each NIC. The other category is link-based channel allocation [18, 14] that assigns channels to links. In these protocols, NICs in a node are grouped as unlink NICs and down link NICs according to the routing topology of the network. The framework should support nodes to manage their NICs with different channels and working mechanisms.

Second, because NICs of the same node are using different channels, the MIMC-SIM framework needs to handle issues including the mapping between MAC address and assigned channel, the mapping between IP address and multiple NICs, and data broadcast in multiple channels, because a packet needs to be delivered by a particular NIC in a particular channel. In a MIMC network, a NIC is always uniquely identified

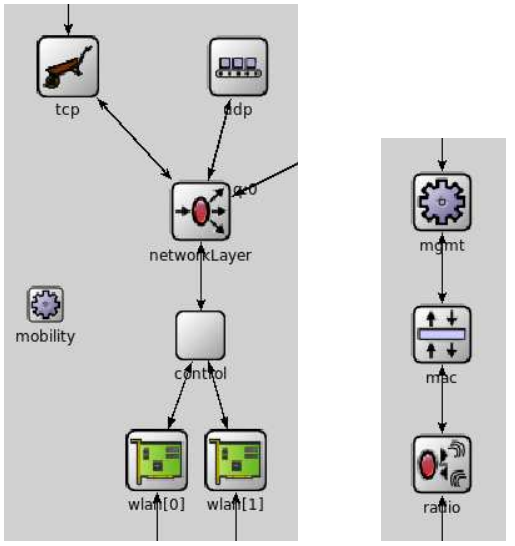
by its MAC address, while a node could be identified by a single IP or multiple IPs. When a node sends or forwards a data packet, the data packet normally only carries the IP addresses of the destination and the next hop. The sending node needs to resolve the MAC address of the next hop NIC and the channel information with the CA protocol and the ARP protocol. As NICs could switch on different channels, the CA protocol needs to help nodes maintain channel information associated with their next hop NICs. Hence, the MIMC-SIM framework needs to properly maintain MAC addresses, IP addresses and channels of NICs and nodes to support CA protocols.

Third, CA protocols need NICs to interact with other protocols, beyond simply making NICs forward packets. The MIMC-SIM framework is desired to support CA protocols working with various MAC protocols, ARP protocols, routing protocols and IP protocols. To achieve this, the framework needs to identify the components in CA protocols that are independent of MAC and network protocols. Meanwhile, the framework should provide mechanisms for these protocols to interact so that a CA protocol can work with a specific MAC protocol or network protocol.

Finally, a variety of CA protocols have been proposed in the past. The MIMC-SIM framework shall provide a coding structure that accommodates common features shared among these protocols and allows flexible extension to implement specific protocol behaviors as well. Many CA protocols can be modeled by an *operation plane* and an *algorithm plane*. The operation plane specifies the operations to collect channel information by coordinate multiple NICs on scanning, beaconing, and exchanging channel information. The algorithm plane computes the channel allocation based on the channel information collected by the operation plane. Although a CA protocol always differs from other CA protocols in many details, they share quite some common procedures of executing operations and algorithms. Hence, the MIMC-SIM framework utilizes these observations to structure its code.

3.3 Architecture of MIMC-SIM

To address the aforementioned issues in MIMC network simulation, the MIMC-SIM framework adds a new layer between the network layer and the MAC layer to implement CA protocols. Correspondingly, MIMC-SIM defines a new host structure as shown in Figures 2(a) and 2(b). The new host structure allows CA protocols to easily work with various MAC protocols in the lower layer and IP protocols in the upper layer. Compared with a typical host in INET as shown in Figure 1, MIMC-SIM adds (i) a new *control* module in the new host structure to manage multiple *wlans* so that they can collaborate on channel management and data transmission, and (ii) a new *mgmt* module in the *wlan* module to implement some channel management operations. The *wlan* module represents a wireless NIC in the host for sending and receiving



(a) Host Structure

(b) Structure of Wlan Module

Figure 2. Architecture of MIMC-SIM

packets via wireless communication. A host could have multiple *wlans*, but the original host structure in INET does not coordinate them. The other modules execute corresponding protocols as in INET.

The *control* module implements the operation plane and the algorithm plane of CA protocols. It composes channel management packets, coordinates the operations of *wlans*, analyzes channel information collected from neighboring nodes, and selects proper channels. The *control* module maintains the mapping between MAC addresses and assigned channels of its own NICs and its neighbors' NICs and groups NICs according to their roles in CA protocols. It ensures an outgoing packet will be transmitted via a correct *wlan* to the right next hop NIC. Hence, a new CA protocol can be adopted into the simulation framework by implementing a new *control* module without changing other modules. Researchers and developers can easily plug in their own CA protocols using the *control* module.

The *wlan* module in the MIMC-SIM framework is also designed to support CA protocols. Figure 2(b) shows its internal structure. It has three sub modules as in the original structure of a wireless NIC in INET. The *radio* module works at the physical layer, processes radio signal, and modulates and demodulates transmitted packets. The *radio* module always works at a specified channel under the command of the *mgmt* module. The *mac* module implements MAC protocols and is not affected by any upper-layer CA protocols. The *mgmt* module in the MIMC-SIM framework replaces the original *mgmt* module in INET to support CA protocols. It is not designed according to any particular CA protocol. Rather,

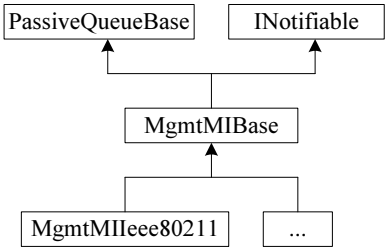


Figure 3. Class Hierarchy of Mgmt Module

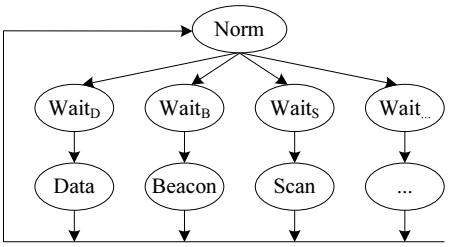


Figure 4. State Transition Diagram in Mgmt Module

it only directs the lower two modules to conduct common operations for all CA protocols and ensures that all packets are delivered on the correct channels. Such design ensures changing CA protocols will not affect the *mgmt* module.

4 MAIN MODULES OF MIMC-SIM

In this section, we present the design details of the modules and the messages of the MIMC-SIM framework. We also discuss the necessary modification in other modules of INET to add MIMC support.

4.1 Per-Interface Management Module

The *mgmt* module is built upon a base class *MgmtMIBase*, which specifies the common operations for all CA protocols. Since CA protocols need nodes to exchange channel management packets, the base class is extended to child classes for various MAC protocols. These child classes add MAC headers to the channel management messages for transmission. Figure 3 illustrates the implementation of the class hierarchy of the *mgmt* module. Currently, only one child class is built for the IEEE802.11 MAC protocol to compose channel management messages in IEEE802.11 data frames.

The *MgmtMIBase* class mainly handles the data transmission operation and certain channel management operations common to all CA protocols. It maintains a *data packet queue* that buffers data packets received from upper layer modules and a *management packet queue* that buffers channel management packets. The channel management packets have a higher priority than the data packets. Whenever a channel management packet arrives, the *mgmt* module will try to sent it out first, even if the packet arrives later than a data packet.

Figure 4 shows the state transition diagrams of the operations. The data transmission operation consists of three steps, starting from the *Norm* state upon receiving a data packet from the upper layer. The *mgmt* module first retrieves the channel for sending the data packet to the next hop NIC, and sends a command to the *radio* module to change the channel. Then, the *mgmt* module gets into the $Wait_D$ state, waiting for the *radio* module to tune the channel. When the *radio* module changes to the channel, it will send a notification to the *mgmt* module. Then, the *mgmt* module gets into the *Data* state to send out the data packet. After the data packet is sent out, the *mgmt* module goes back to the *Norm* state.

The other channel management operations follow the same procedure as the data transmission operation. These operations shall be used by all CA protocols. For example, many CA protocols need NICs to scan traffic and broadcast beacon packets in a channel periodically. The *mgmt* module will set the channel first, and then take the action. Hence, each operation goes through three steps. In addition, these operations are atomic in that when the *mgmt* module conducts one operation, it shall not conduct another one, as a NIC can only work on one channel at one time.

However, the *mgmt* module does not decide when to conduct an operation. Instead, it only carries an operation upon receiving a command from the *control* module. It is the responsibility of the *control* module to send proper commands in a proper manner according to a CA protocol. The *mgmt* module in fact provides these operations as tool kits that various CA protocols can utilize. Developers only need to decide how to use these tool kits in CA protocols, instead of mixing these operations within CA protocols. The *mgmt* module can also be easily extended to support more channel management operations, as long as they are atomic and go through the three steps.

4.2 Control Module

The *control* module is the core module implementing CA protocols. It is built upon the *MIControlBase* class, which declares a set of abstract functions and implements only a few basic INET functions that initialize the module and pass messages to proper functions. A child class shall be derived and implemented from the base class for each CA protocol. For example, in the class hierarchy of Figure 5, the *MIControlSIC* class implements the detail of the superimpose code based CA protocol [19].

In order to coordinate multiple NICs within a node, the *control* module maintains a *NicMgmtTable* that includes the channel and address information for all NICs. The *control* module also maintains a *NeighborTable* that records the channel and address information of the NICs of neighboring nodes. The *control* module performs the following major operations for a CA protocol: (a) broadcasting data packets, (b) computing channels, (c) scheduling channel scans, (d)

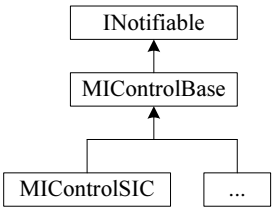


Figure 5. Class Hierarchy of Control Module

scheduling and broadcasting beacon packets, (e) handling the state diagram of the CA protocol.

In a multi-channel network, operation (a) is quite complicated as it requires the sender to replicate the broadcast in all available channels. The *control* module clones a broadcast data packet on each channel and puts the to-send channel in each cloned packet. Then, it distributes the cloned packets to the sending NICs so that the broadcasting from the NICs will not duplicate. Broadcasting a data packet is different from unicasting a data packet in the MIMC-SIM framework. Unicasting a data packet is directly handled by the *mgmt* module, and the *control* module simply sends a unicast data packet to the *mgmt* module.

Operation (b) executes the core channel assignment algorithm of a CA protocol. When a node collects sufficient information of traffic and assigned channels from its neighbors, the node computes a proper channel according to some CA algorithms. In the *control* module, the *assignChannel* function is declared and must be implemented to perform this operation.

In order to get channel and traffic information, CA protocols usually require nodes to scan and listen local traffic periodically. Operation (c) is defined for this purpose. But, the *control* module does not directly control the NIC to scan. Instead, the *control* module only schedules the time points of scanning and sends commands at the scheduled time points to the *mgmt* module to perform this operation.

When a node is assigned a channel, the node shall let other nodes know its channel and related channel information. CA protocols achieve this by asking nodes to periodically broadcast beacon packets. A beacon normally includes the node identification information and channel information. When other nodes scan, they can receive the beacons and thus obtain the channel information of the beaconing nodes. The *control* module carries operation (d) by generating beacon packets in all channels and scheduling the broadcasting time points. Then, the *control* module sends the beacon packets at the scheduled time points to NICs to broadcast.

As any other networking protocols, a CA protocol can be described by a state transition diagram as shown in Figure 6. In MIMC-SIM, the state transition diagram is implemented as a finite state machine (FSM) in the *control* module. As illustrated, many CA protocols share three common states that perform the first four operations, and have other protocol-specific states to conduct protocol-specific operations.

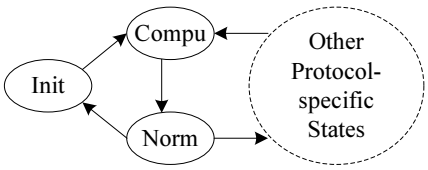


Figure 6. State Transition Diagram in Control Module

The *Init* state is when a node joins a network or resets itself. In this state, the *control* module performs operation (c) to collect channel information of its neighbors. The *Compu* state is when a node collects sufficient channel information and performs operation (b) to select a channel. The *Norm* state is when a node is assigned a channel and can fulfill its networking functions of routing and forwarding packets. In this state, the node also performs operations (a), (c) and (d).

In the *Norm* state, a node could receive updated channel information from its neighbors. Hence, the node could go into another state to process the updated information, such as verifying the information or relaying the information to other nodes. These operations are protocol-specific. The mechanism of implementing a FSM in the INET framework can easily support extension and adoption of a state transition diagram defined according to a CA protocol.

4.3 Channel Management Packets

The MIMC-SIM framework groups all packets above the MAC layer into two categories. One is data packets that are generated at or above the network layer. Processing a data packet is similar to the processing in INET, except that a channel must be associated with each data packet to transmit.

The other category is channel management packets specified by CA protocols. In a MIMC network, nodes exchanges channel management packets to negotiate channels and notify neighbors. The two common packets implemented in the MIMC-SIM framework are beacon packets *MICTRL_BEACON* and notification packets *MICTRL_NOTICE*. The notification packets are unicast packets. When a node updates its channel information, it uses this packet to notify its neighbors. A notification packet differs from a beacon packet in that the beacon packet is broadcast while the notification packet only targets the neighbors known to the sender.

Because the MIMC-SIM framework is built atop the MAC layer, both data packets and channel management packets are considered as data frame at the MAC layer. For example, if the underlying MAC protocol is IEEE802.11, the two types of packets will be formatted as IEEE802.11 Data Frame.

4.4 Modifications in INET

To support MIMC network simulation, we also inspected the existing modules in INET and made necessary and minimum changes in some modules. The major changes and the corresponding classes are as follows.

• *AbstractRadio* class. The class implements the *radio* module for communication at the PHY layer. When the *radio* module changes to a new channel while it is receiving, it should clear its states so that new receiving and transmitting procedures can be started. This was not correctly implemented in INET.
• *ChannelControl* class. The original class assigns and maintains one channel per host. In a MIMC network, multiple channels are assigned to NICs of a single host. The class is modified to give each host a list of radios and assign a channel to each radio.
• *ChannelAccess* class. The original class was designed to support packet delivery among hosts only. But, in a MIMC network, a packet should reach NICs on the same channel of the sending NIC. The class is modified to make packets delivery happen among NICs.
• *Ieee80211Mac* class. The class implements the IEEE802.11 MAC protocol. In the original class, it is possible that the radio is asked to change channel while the MAC is still in some waiting states for transmitting a packet. Apparently, the radio shall only change channel when no packet is waiting for transmission in MAC, so that a waiting packet can be transmitted in its expected channel. The class is modified accordingly.

5 IMPLEMENTATION AND EVALUATIONS

In this section, we discuss our current implementation of the MIMC-SIM framework and show the results of experiments with an implemented CA protocol.

5.1 Implementation

The MIMC-SIM framework is implemented in INET snapshot 2009-03-12 with OMNET++ 4.0. We implemented a CA protocol that uses node-based channel assignment as [16] and computes channels based on superimposed codes according to the CA algorithm proposed in [19]. The implemented CA protocol works as follows. When nodes start, they do not beacon but rather scan and listen on all channels. After two complete rounds of scanning, they choose a temporary channel to beacon and also keep scaning on channels periodically. Over time, they will get codes assigned to their neighbors and thus compute channels. Whenever they obtain channels, they will notify their known neighbors. They keep changing to better channels when they learn more codes of their neighbors. After they get all neighbor codes, they will find their final channels to stay on.

The implementation is tested in networks of nodes equipped with dual radios. 13 orthogonal channels are used. Each node is assigned a 13-bit superimposed code. The node computes a channel based on its own superimposed code and the codes of its neighbors. The channel is then assigned to one of the dual radios of the node. Hence, the node always has one channel and one NIC to receive incoming packets. The other NIC is used to switch on different channels for sending packets to different next hops.

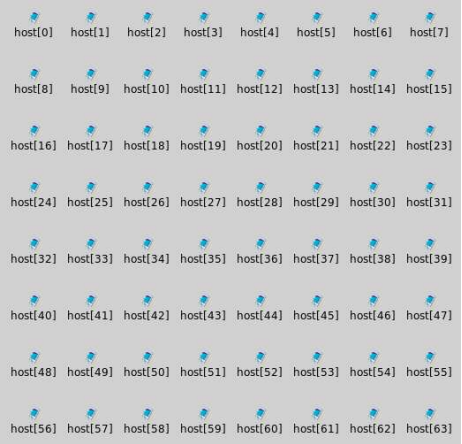


Figure 7. An Example 8x8 MIMC Network in Experiments

The *mgmt* module implements three atomic operations for the CA protocol: data transmission, beacon, and scan. Two channel management packets are implemented: beacon and notification. The CA protocol makes each node beacon on its assigned channel once every 0.01 second. Each node also scans one channel for 0.02 second every 1 second.

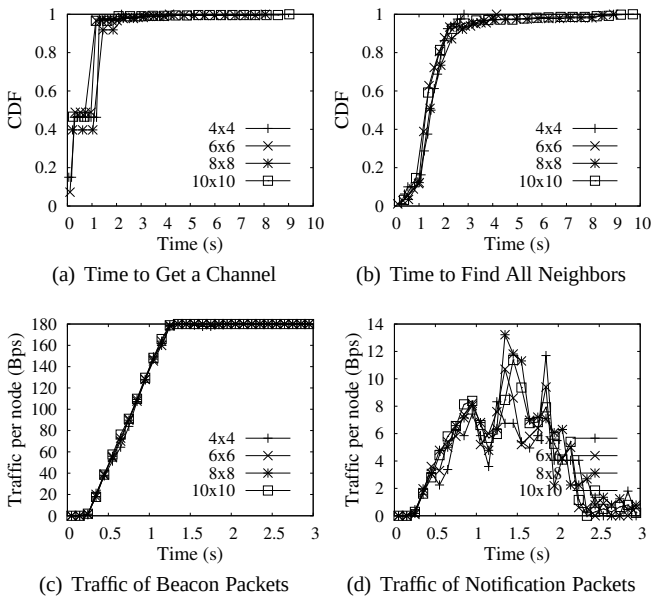
5.2 Experiments and Evaluation

A few experiments were conducted to test the MIMC-SIM framework and the implemented CA protocol. We set a network with nodes evenly distributed on 4x4, 6x6, 8x8 and 10x10 grids in the network. Nodes are one-hop neighbors if they are in adjacent grids. Routes are set statistically so that any node can communicate with any other node. An example network is depicted in Figure 7. We let all nodes in the network start at random time points in the first second. Then, we traced them to find how they obtain their channels.

We collected data of the following metrics to study the performance of the implemented CA protocol: time to get a channel, time to discover all neighbors, traffic volume of beacon packets, and traffic volume of notification packets. These metrics show the convergence and the overhead of channel allocation. We verified the framework and the protocol by comparing the measured metrics. If these metrics show consistent results, the implementation is correct.

Figures 8(a) and 8(b) show the cumulative distribution function (CDF) of the time that all nodes spend to find their neighbors and obtain their channels. The two figures illustrate that nodes need fewer than 3 seconds to get their final channels, given the parameters used in the simulation. After nodes get their final channels, they will stay on their channels, as the network does not change.

The same results can be observed from Figures 8(c) and



(a) Time to Get a Channel (b) Time to Find All Neighbors

(c) Traffic of Beacon Packets (d) Traffic of Notification Packets

Figure 8. Evaluation of Channel Assignment

8(d) that show the traffic volume of beacon and notification packets. Because nodes do not beacon in the early phase when they start, beacon traffic volume is low. Then, the traffic volume increases as more and more nodes choose temporary channels and beacon. When all nodes obtain temporary channels and start beaconing (at about 1.5 seconds in Figure 8(c)), the beacon traffic volume stabilizes. However, nodes continue to get more codes of their neighbors and thus they continue to change their channels to better utilize channels in their vicinities. Hence, the notification traffic volume grows to the peak at about 1.5 seconds when all nods start beaconing. Then, the traffic volume decreases to zero at about 3 seconds, when nodes find all neighbors and obtain the final channels. After that, nodes do not change channels and thus do not send notification packets.

6 CONCLUSION

In this paper, we designed and developed a generic simulation framework for MIMC networks in INET/OMNET++. The framework provides a extensible and flexible code structure that effectively supports various CA protocols. In implementation, a new layer is added between the network layer and the MAC layer. Two new modules are constructed: *control* module and *mgmt* module. The *control* module handles the operations according to the specifications of CA protocols. New CA protocols can also be implemented in the *control* module by extending its base class. The *mgmt* module handles channel assignment packets and ensures that packets are transferred on correct channels. It also provides common operations for CA protocols. The implementation and experiments showed that MIMC-SIM can be used for research and development of CA protocols.

REFERENCES

[1] EmuLab. http://www.emulab.net/.

[2] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[3] INET. http://inet.omnetpp.org/.

[4] MAP. https://engineering.purdue.edu/MESH.

[5] NS2. http://www.isi.edu/nsnam/ns/.

[6] OMNET++. http://www.omnetpp.org/.

[7] OPNET. http://www.opnet.com/.

[8] UCR-Testbed. http://networks.cs.ucr.edu/testbed/.

[9] WINLAB. http://www.winlab.rutgers.edu/.

[10] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks. In *Proc. of IEEE BROADNETS*, pages 344–354, 2004.

[11] V. Bhandari and N. H. Vaidya. Capacity of multi-channel wireless networks with random (c, f) assignment. In *Proc. of ACM Mobihoc*, pages 229–238, 2007.

[12] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. of ACM Mobicom*, pages 31–42, 2005.

[13] C.-M. Cheng, P.-H. Hsiao, H. Kung, and D. Vlah. Adjacent Channel Interference in Dual-radio 802.11a Nodes and Its Impact on Multi-hop Networking. In *Proc. of IEEE GLOBECOM*, pages 1–6, 2006.

[14] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian. Practical, distributed channel assignment and routing in dual-radio mesh networks. In *Proc. of ACM SIGCOMM*, volume 39, pages 99–110, 2009.

[15] P. Kyasanur and N. H. Vaidya. Capacity of multi-channel wireless networks: impact of number of channels and interfaces. In *Proc. of ACM Mobicom*, pages 43–57, 2005.

[16] P. Kyasanur and N. H. Vaidya. Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(1):31–43, 2006.

[17] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve. Data aggregation and roadside unit placement for a vanet traffic information system. In *Proc. of ACM VANET*, pages 58–65, 2008.

[18] A. Raniwala and T.-c. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proc. of IEEE Infocom*, volume 3, pages 2223–2234, 2005.

[19] K. Xing, X. Cheng, L. Ma, and Q. Liang. Superimposed code based channel assignment in multi-radio multi-channel wireless mesh networks. In *Proc. of ACM Mobicom*, pages 15–26, 2007.