Chapter 4 A Framework Simulation in OPNET Modeler

4.1 Introduction

In this chapter will present about a designed framework which simulating on OPNET Modeler software. The research is simulating those techniques and designed processes on OPNET Modeler software which is a licensing at University of Salford. OPNET Modeler software is a network simulation software and solution. This software provides for application and network management issues.

4.2 Network simulation

For doing research in the network field, network simulation software is a very useful and important tool. As researchers or protocol designers have to design and testing the system in simulation software before using it in a real network. There are many network simulations that widely used in networking research such as OMNET++ (Objective Modular Network Testbed in C++), NS-2 (Network Simulator version 2) and QualNet [69].

OPNET Modeler is generally used by researchers, developing protocol designers and so on. The OPNET software was funded in 1986 by Alain Cohen. OPNET stands for Optimizing Network Engineering Tools [70]. OPNET Modeler provides a comprehensive development environment which is powerful for instance simulation, data analysis, model design and etc. also it can support lot of technologies including local area network (LAN), mobile network, sensor network, wireless network and so on.

4.2.1 Basic Structure within OPNET Modeler

This is the workflow for OPNET Modeler. Normally, the researcher use these steps to build a network model, create the traffic, choose statistics and then run simulations.

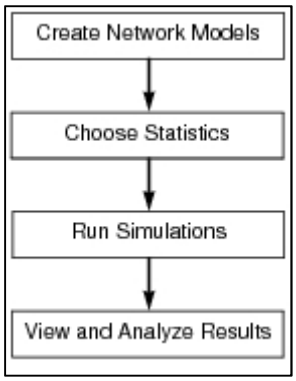


Figure 4-1 Basic step for creating network simulation

These 4 steps in Figure 4-1 consist of creating network environments which is including network devices and traffic, and then choose statistics that we want to study. Next step is run simulations. Finally, view and analyze the results. To complete these 4 steps, OPNET Modeler provides variety kinds of editor to support users as show below.

- ***The Project Editor***

This is a main area of OPNET simulation. We use this area to create network topology, generate traffic within network and view the results via this editor. Moreover, this area still covers about choosing statistics and running simulations.

- ***The Node Editor***

The user can define the behavior of each network object via *"Node Editor"*. In Node Editor of each model, the behavior is defined using different modules for example data storage, data creation, etc. A network object in OPNET Modeler is typically building up from multiple modules which define that object. The user can add their modules into the network object via Node Editor.

- ***A Network Model in the Project Editor***

The OPNET Modeler let user to design and create any elements of network as they wish. For instance, user can create node, link model, process models and build packet formats. Also, the user can create filters and parameters that they want to analyze.

➢ ***Node Model***

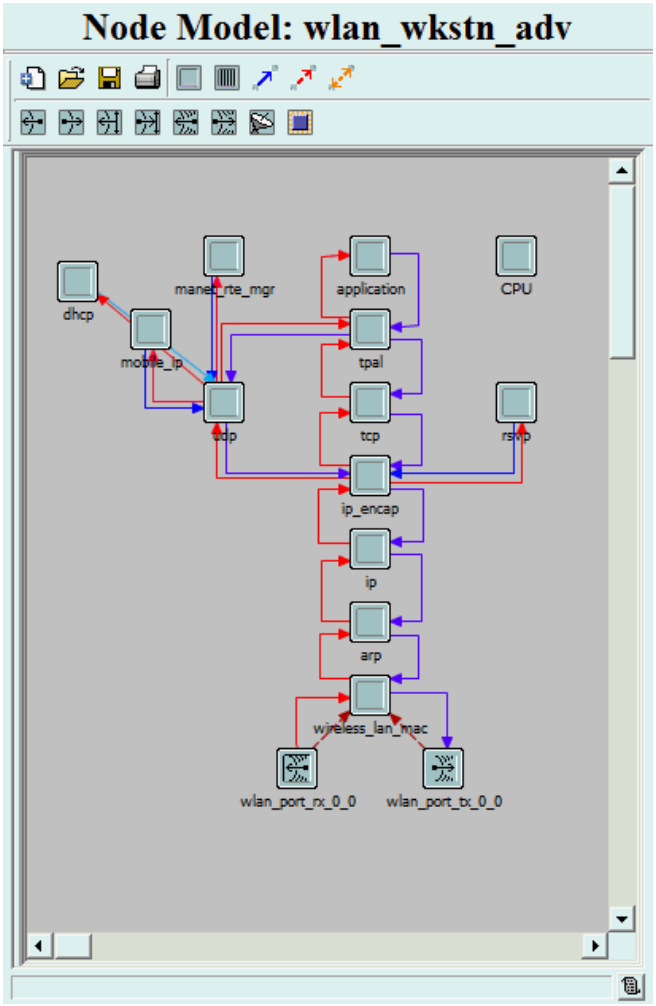


Figure 4-2 Node Mode example

➢ **The Process Model Editor**

The OPNET Modeler let user design and creates their process models via the *"Process Editor"*. The user can start from create node model in Node Editor and then they can build process model, which control the functionality of that node mode.

➢ **Process Model**

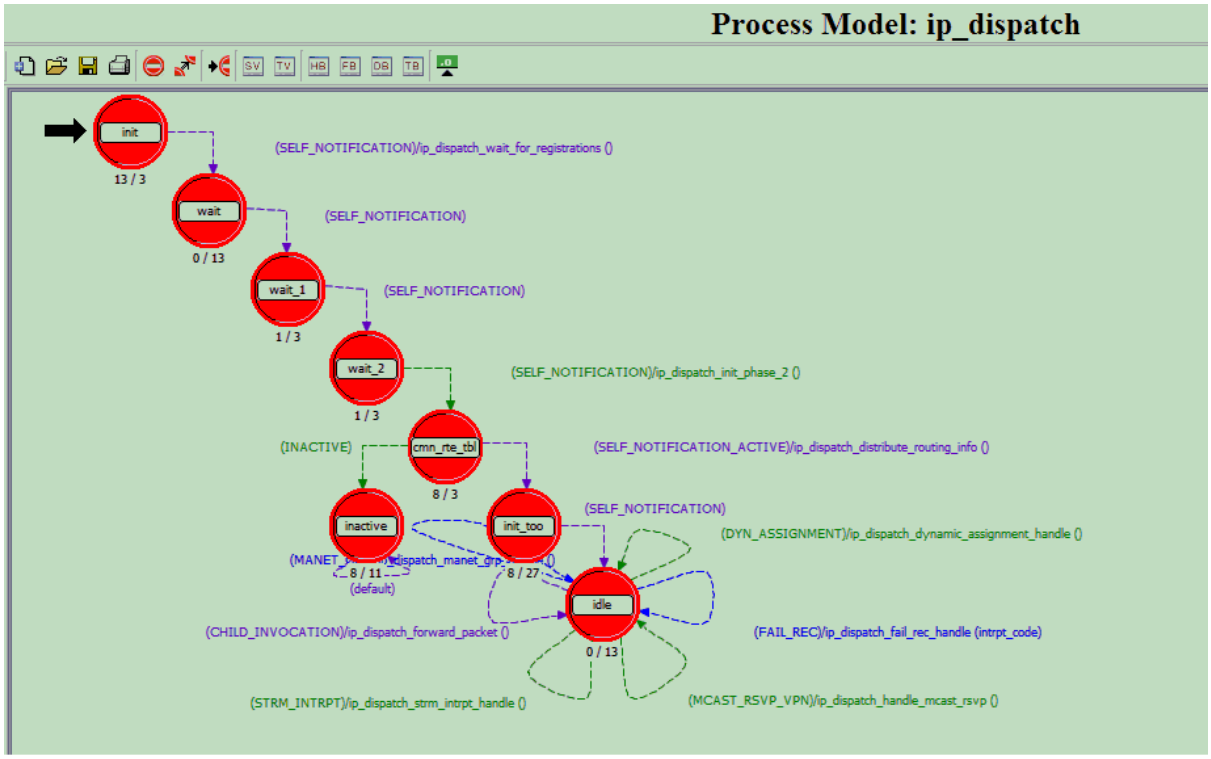


Figure 4-3 Process Mode example

4.3 Implementation of the Proposed Framework in OPNET Modeler

Due to the implementation of this research has been simulation environments and testing the performance of designing on OPNET Modeler software version 16.0 which is not supported multicast communication over IPv6 WiFi environment. Hence, the implementation and development of this thesis has been modifying based on IPv4 environment. However, the concept and designed of this framework can adapt to WiFi network both on IPv4 and IPv6 Networks.

4.3.1 Network Architecture

Normally, the first thing that has to start network simulation is OPNET Modeler is to create network architecture. The common start network topology that is used in this research is shown in Figure 4-4.

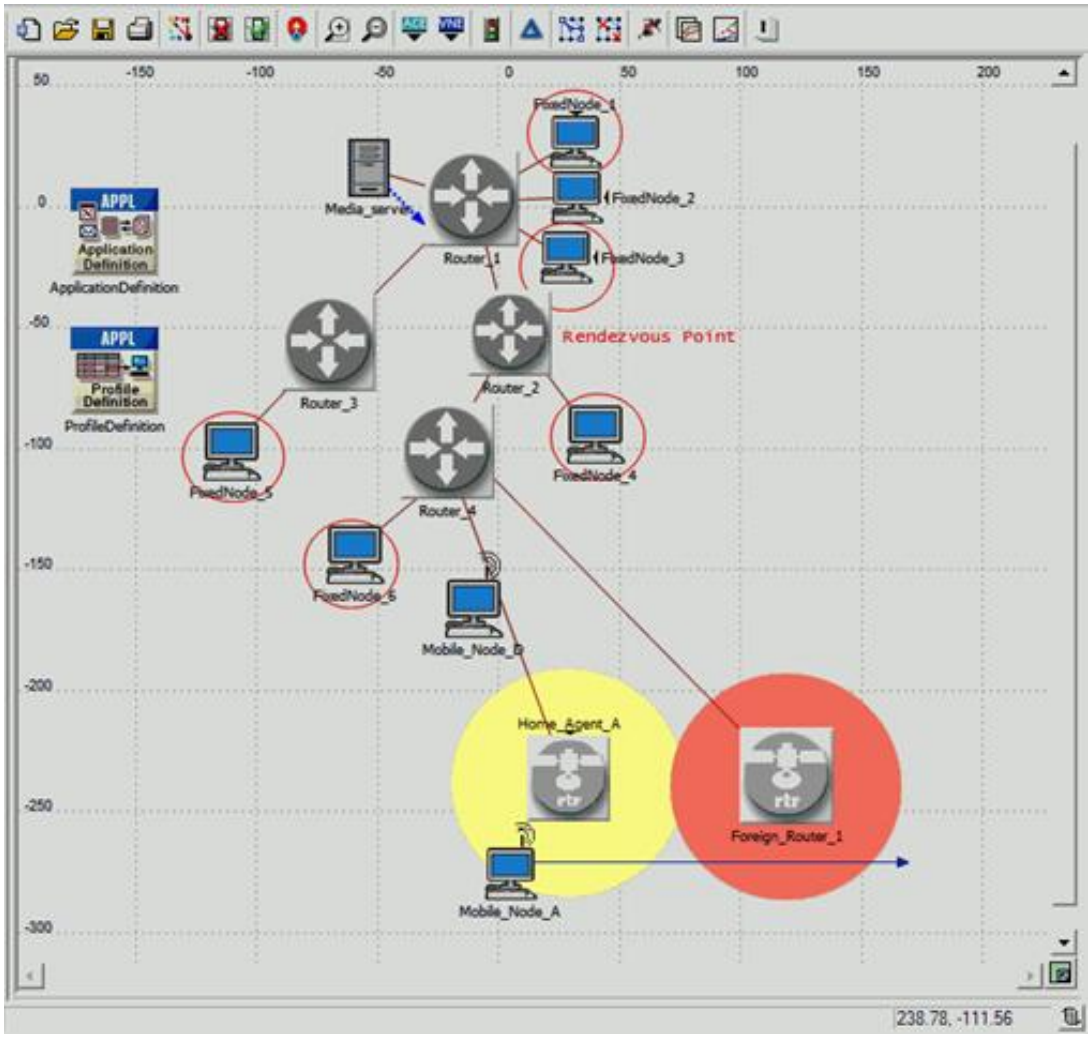


Figure 4-4 the common network topology that is used in this research.

4.3.2 Process Model

Therefore, we have to create a network environment and add some processes into standard protocol. So, we have to deal with process model many times.

4.3.2.1 Asking CoA in Advance process

In the designed framework, a mobile node will send an Agent Solicitation Message to foreign agent to ask for a CoA in advance. This process happens via the Mobile IP protocol. However in OPNET Modeler, this method will happen by creating an extra state and adding into the process model of mobile_ip_mn.

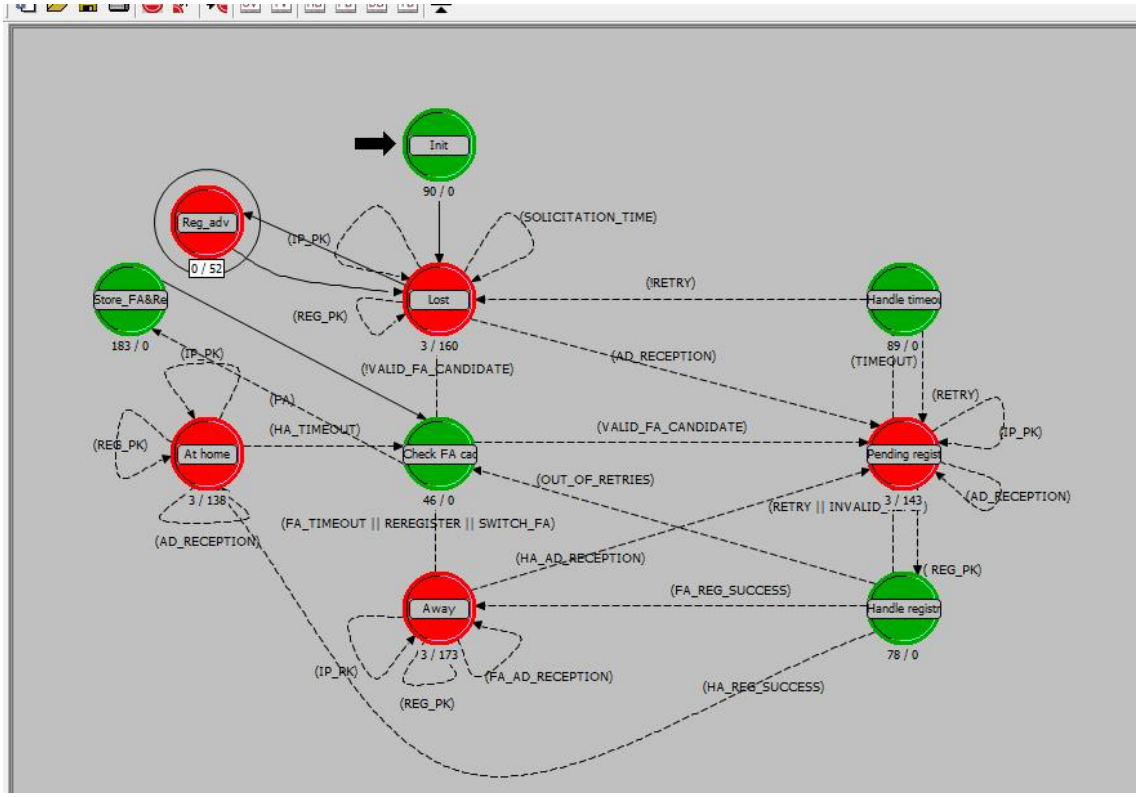


Figure 4-5 Process Model: mobile_ip_mn

Figure 4-5 is shown the Process Model of Mobile IP protocol within mobile node. To achieve the process of asking CoA in advance, state name “Reg_adv” have to be added. The coding of this state is shown in the Figure 4-6.

```
static void
mip_mn_agent_solicit_pk_send_adv (void)
    {
    Packet*     solicit_pkptr_adv;
    double      solicit_interval;

    /** PURPOSE: Send the ICMP agent solicitation packet.**/
    /** REQUIRES: none. **/
    /** EFFECTS: Packet will be given to IP to handle.**/
    FIN (mip_mn_agent_solicit_pk_send (void));

    /* Time to send out the solicitation. */

    usleep(10000000); // 10 secs
    solicit_pkptr_adv = op_pk_create_fmt ("mobile_ip_irdp_solicit");

    /* Send the packet out. */
    module_data->ip_ptc_mem.child_pkptr = mip_sup_irdp_pkt_encapsulate
        (solicit_pkptr_adv, home_address, subnet_bcast_addr, IcmpC_Type_IRDP_Sol);

    /* Record some stats. */
    op_stat_write (irdp_sent_pkts_sh, 1.0);
    op_stat_write (irdp_sent_bits_sh, op_pk_total_size_get (module_data->ip_ptc_mem.child_pkptr));
    op_stat_write (g_irdp_sent_bits_sh, op_pk_total_size_get (module_data->ip_ptc_mem.child_pkptr));
    op_stat_write (g_irdp_sent_bits_sh, 0.0);

    /* Invoke IP to handle the packet. */
    op_pro_invoke (proc_info_struct_ptr->ip_phndl, OPC_NIL);

    /* Schedule the next transmission. */
    if (++solicit_count > 3)
        {
        solicit_interval = MipC_MN_Solicit_Min_Interval * pow (2.0, (double) (solicit_count - 3));

        if (solicit_interval > MipC_MN_Solicit_Max_Interval)
            {
            solicit_interval = MipC_MN_Solicit_Max_Interval;
            }
        }
    else
        {
        solicit_interval = MipC_MN_Solicit_Min_Interval;
        }

    solicit_timer_ehndl = op_intrpt_schedule_self (op_sim_time () + solicit_interval, MipC_MN_Timer_Solicit);

    FOUT;
    }
```

Figure 4-6 The coding of "Reg_adv" state

4.3.2.2 Joining Multicast using CoA Address

This process model has been called after the mobile node received CoA in advance and then tries to build another route to multicast tree by using CoA address. This stage has been extended from IGMP process model. To achieve this, we have to create a process state adding into IGMP process model name "JOIN_ADV" as is shown in Figure 4-7. The coding of process state is presented in Figure 4-8.

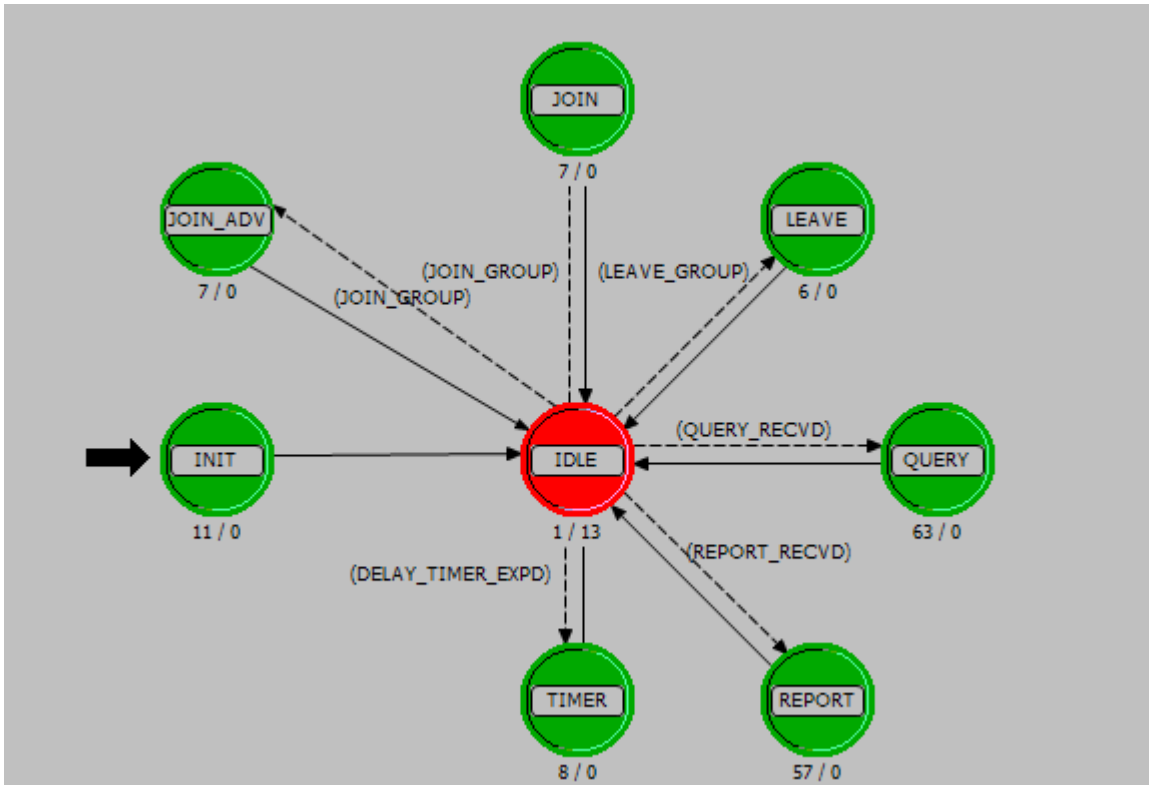


Figure 4-7 Process Model: IGMP host

```
/** Received a CoA in Advance, and then use these CoA to join multicast tree for  **/
/** creating a second route. Send an Unsolicited Membership Report message and start**/
/** start the delay timer for this group to send the next Unsolicited Report message **/

ip_igmp_host_join_grp (ip_ptc_mem_ptr->ip_mcast_ptc_info.ip_grp_addr,
                       ip_ptc_mem_ptr->ip_mcast_ptc_info.major_port);
```

Figure 4-8 The coding of "JOIN_ADV" state

4.3.2.3 Re-join Multicast

When the mobile node has been realized that, now it is in a foreign agent. The process re-join will call multicast joining state in process model "ip_igmp_rte_grp" to send a message to join multicast group again.

```
/** time. Notify the router and start the group membership timer    **/

/* Generate trace messages  */
if (LTRACE_IGMP)
    {
    ip_address_print (ip_addr_str, ip_grp_addr);
    sprintf (msg0, "IP Group Address        :       %s", ip_addr_str);
    sprintf (msg1, "IP Interface            :       %d", ip_interface);
    op_prg_odb_print_major ("Received an IGMP Membership Report message for: ", msg0, msg1, OP
    op_prg_odb_print_major ("A local host has joined the above group. Notifying PIM-SM router
    }

/* Notify the router */
ip_igmp_rte_grp_notify_router (IpC_Igmp_Rte_Pim_Sm_Notify_Plus);

/* Start the group membership timer only if simulation efficiency is disabled */
if (igmp_attrs_ptr->igmp_sim_efficiency == OPC_FALSE)
    {
    /* Generate trace messages  */
    if (LTRACE_IGMP)
        {
        sprintf (msg2, "Starting the Group Membership timer with value, %d for the above group
        op_prg_odb_print_major (msg2, OPC_NIL);
        }
    ip_igmp_rte_grp_start_timer (IPC_IGMP_RTE_GRP_TIMER, igmp_attrs_ptr->grp_member_interval);
    }

/* Tag this connection process for debugging purposes.  */
if (op_sim_debug () == OPC_TRUE)
    {
    ip_address_print (ip_addr_str, ip_grp_addr);

    sprintf (msg0, "Group address: %s\tInterface index: %d", ip_addr_str, ip_interface);

    op_pro_tag_set (op_pro_self (), msg0);
    }
```

Figure 4-9 the coding for re-join multicast

4.3.2.4 Keeping Multicast Route

After the mobile node created other multicast routes, the mobile node have to keep other routes become Standby mode, only one route at a time being Active mode. To achieve this goal, we modified Join/Prune message in PIM protocol to keep these multicast routes alive. The coding of this process state is shown in Figure 4-11.

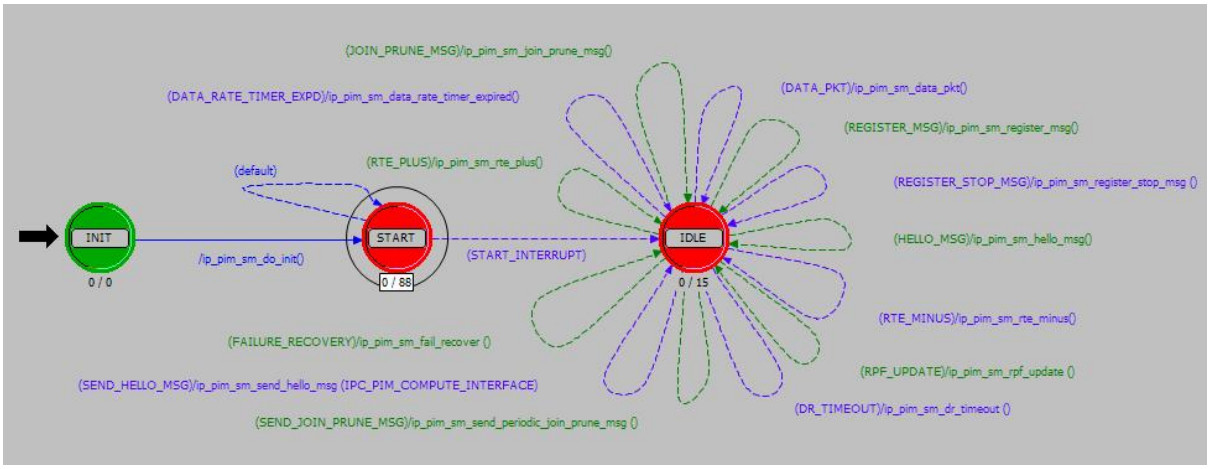


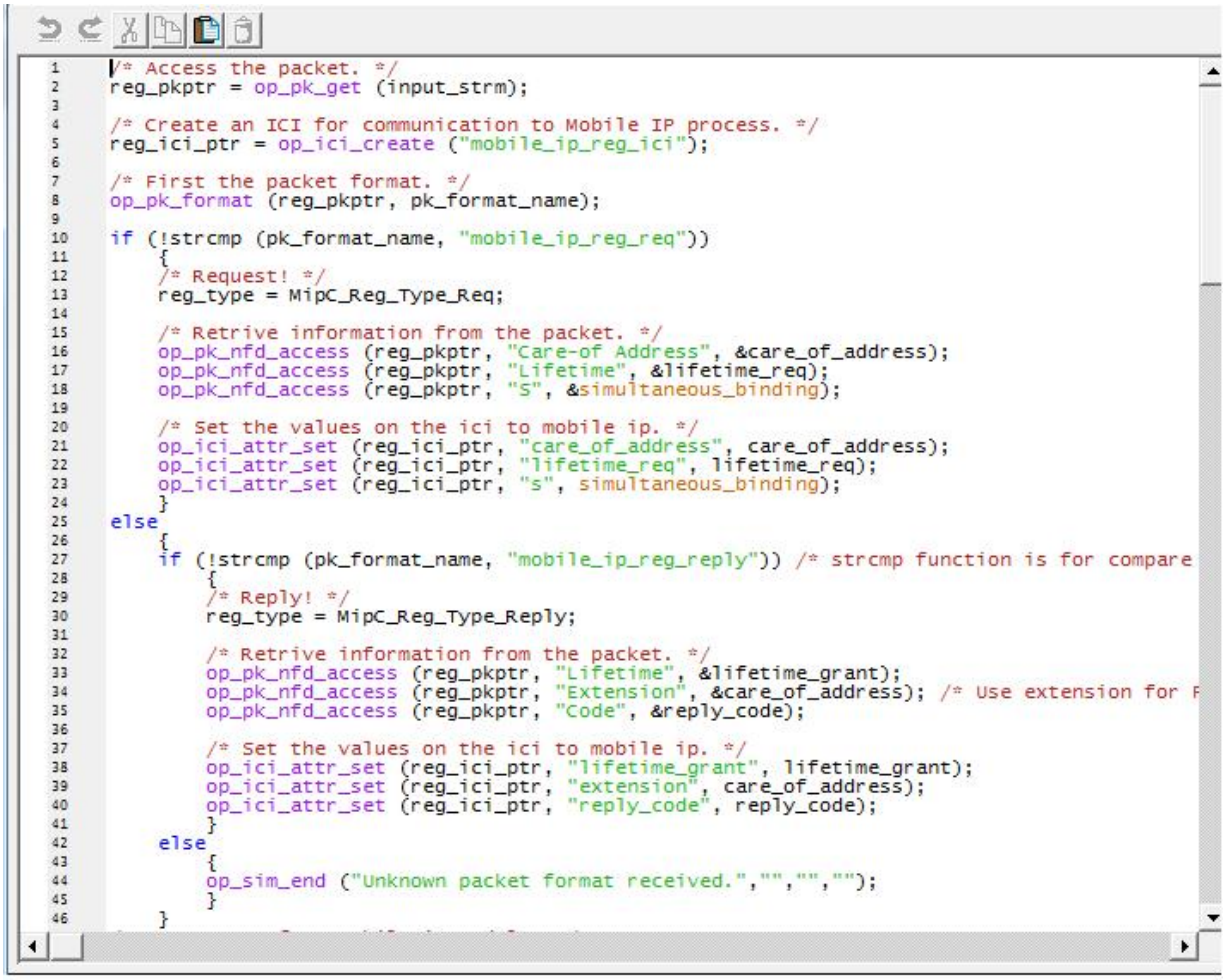
Figure 4-10 Process Model: PIM-SM protocol

```
/* Send the hello messages.                          */
ip_pim_sm_send_hello_msg (IPC_PIM_ALL_INTERFACES);

/* Join/Standby message to keep multicast route.     */
/* Bootstrap candidate RP information. */
/* static and auto-rp configuration.                 */
if ((boostrap_support == OPC_TRUE) && (bootstrap_rp_lptr != OPC_NIL))
    {
    ip_pim_rp_lists_merge (bootstrap_rp_lptr);
    }
else if ((boostrap_support == OPC_FALSE) && (bootstrap_rp_lptr != OPC_NIL))
    {
    /* There were some candidate RPs configured for bootstrap    */
    /* However, there is no preferred BSR. Log a message.        */
    ipnl_protwarn_mcast_bootstrap_not_supp ();
    }

if (log_call_scheduled == OPC_FALSE)
    {
    /* Keep Standby the route.  */

    op_intrpt_schedule_call (OPC_INTRPT_SCHED_CALL_ENDSIM, 0,
        ipnl_protwarn_mcast_endsim_log_write, OPC_NIL);

    log_call_scheduled = OPC_TRUE;
    }

/* Check for the list of Active/Standby mode     */
ip_pim_sm_rp_table_to_ot_export_schedule ();

/* Check if Group mapping has been configured to export to OT reports    */
ip_pim_sm_group_table_to_ot_export (OPC_NIL, 0);
}
```

Figure 4-11 The coding for keeping multicast route

4.3.2.5 Store CoA addresses

When the mobile node moves, the mobile node starts to receive CoA address along the path. However, in the large network which consists of many routers in different zones, the mobile node also receive multiple CoA address from foreign router. Hence, the mobile node needs to have a process of store and process multiple CoA addresses. To solve this issue, we had modified Mobile IP protocol to have the process of store and retrieve CoA address on the mobile node. We have created the state named “Store_FA&Re” state which is shown in Figure 4-5 to solve this issue.



```
/* Access the packet. */
reg_pkptr = op_pk_get (input_strm);

/* Create an ICI for communication to Mobile IP process. */
reg_ici_ptr = op_ici_create ("mobile_ip_reg_ici");

/* First the packet format. */
op_pk_format (reg_pkptr, pk_format_name);

if (!strcmp (pk_format_name, "mobile_ip_reg_req"))
    {
    /* Request! */
    reg_type = MipC_Reg_Type_Req;

    /* Retrive information from the packet. */
    op_pk_nfd_access (reg_pkptr, "Care-of Address", &care_of_address);
    op_pk_nfd_access (reg_pkptr, "Lifetime", &lifetime_req);
    op_pk_nfd_access (reg_pkptr, "S", &simultaneous_binding);

    /* Set the values on the ici to mobile ip. */
    op_ici_attr_set (reg_ici_ptr, "care_of_address", care_of_address);
    op_ici_attr_set (reg_ici_ptr, "lifetime_req", lifetime_req);
    op_ici_attr_set (reg_ici_ptr, "s", simultaneous_binding);
    }
else
    {
    if (!strcmp (pk_format_name, "mobile_ip_reg_reply")) /* strcmp function is for compare
        {
        /* Reply! */
        reg_type = MipC_Reg_Type_Reply;

        /* Retrive information from the packet. */
        op_pk_nfd_access (reg_pkptr, "Lifetime", &lifetime_grant);
        op_pk_nfd_access (reg_pkptr, "Extension", &care_of_address); /* Use extension for F
        op_pk_nfd_access (reg_pkptr, "Code", &reply_code);

        /* Set the values on the ici to mobile ip. */
        op_ici_attr_set (reg_ici_ptr, "lifetime_grant", lifetime_grant);
        op_ici_attr_set (reg_ici_ptr, "extension", care_of_address);
        op_ici_attr_set (reg_ici_ptr, "reply_code", reply_code);
        }
    else
        {
        op_sim_end ("Unknown packet format received.","","","");
        }
    }
```

Figure 4-12 the coding of “store_FA&Re” state

State “store_FA&Re” has two processes of designed framework in there. That is keeping CoA from foreign router, and another process is retrieving CoA address. Some part of coding state is shown in Figure 4-12.

```
/* Structure of each CoA that storage in Mobile node. */
reg_ici_ptr = op_intrpt_ici ();
op_ici_attr_get (reg_ici_ptr, "reg_type", &reg_type);
switch (reg_type)
    {
    case MipC_Reg_Type_Req:
        {
        /* get additional attributes from ici. */
        op_ici_attr_get (reg_ici_ptr, "care_of_address", &care_of_address);
        op_ici_attr_get (reg_ici_ptr, "lifetime_req", &lifetime_req);
        op_ici_attr_get (reg_ici_ptr, "S", &simultaneous_binding);


        reg_pkptr = op_pk_create_fmt ("mobile_ip_reg_req");
        op_pk_nfd_set (reg_pkptr, "Care-of Address", care_of_address);
        op_pk_nfd_set (reg_pkptr, "Lifetime", lifetime_req);
        op_pk_nfd_set (reg_pkptr, "S", simultaneous_binding);

        break;
        }

    case MipC_Reg_Type_Reply:
        {
        /* get additional attribute from ici. */
        op_ici_attr_get (reg_ici_ptr, "lifetime_grant", &lifetime_grant);
        op_ici_attr_get (reg_ici_ptr, "dest_address", &dest_address);
        op_ici_attr_get (reg_ici_ptr, "reply_code", &reply_code);

        reg_pkptr = op_pk_create_fmt ("mobile_ip_reg_reply");
        op_pk_nfd_set (reg_pkptr, "Lifetime", lifetime_grant);
        op_pk_nfd_set (reg_pkptr, "Extension", dest_address);
        op_pk_nfd_set (reg_pkptr, "Code", reply_code);

        break;
        }
```

Figure 4-13 the coding of structure of CoA address list

Figure 4-13 presents the coding of the process in “Store_FA&Re” state. This part is shown the structure of each CoA information that has been stored in the mobile node responsible by Mobile IP message.

```
/* Each CoA address do Building update to Home Agent*/
/* Also take care of identification field as well. */
op_ici_attr_get (reg_ici_ptr, "identification", &identification);
op_pk_nfd_set (reg_pkptr, "Identification", identification);

/* Record received stats. */
op_stat_write (reg_sent_bits_sh, op_pk_total_size_get (reg_pkptr));
op_stat_write (reg_sent_pkts_sh, 1.0);

/* Access the necessary common attributes. */
op_ici_attr_get (reg_ici_ptr, "home_address", &home_address);
op_ici_attr_get (reg_ici_ptr, "home_agent", &ha_address);
op_ici_attr_get (reg_ici_ptr, "dest_address", &dest_address);

/* Set the values on the packet. */
op_pk_nfd_set (reg_pkptr, "Home Address", home_address);
op_pk_nfd_set (reg_pkptr, "Home Agent", ha_address);

/* Send it on to UDP module for delivery. */
op_ici_attr_set (command_ici_ptr, "local_port", MipC_Reg_UDP_Port_Num);
op_ici_attr_set (command_ici_ptr, "rem_port", MipC_Reg_UDP_Port_Num);
op_ici_attr_set (command_ici_ptr, "rem_addr", dest_address);

/* Install the ici and send.                */
op_ici_install (command_ici_ptr);
op_pk_send_forced (reg_pkptr, output_strm);

/* Get the status indication from the ici */
op_ici_attr_get (command_ici_ptr, "status", &ind);

/* Clean up. */
op_ici_destroy (reg_ici_ptr);
```

Figure 4-14 the coding of Binding Update multiple CoA address to home agent

Normally, the mobile node has to report every CoA address that received from foreign agent to home agent. Hence, in the framework the mobile node has to do the same but in advance. So, this process is part of reducing handover latency because the binding update process had been doing in advance before handover occur. The coding of this process is shown in Figure 4-14.

4.4 Summary

This chapter presents the fundamentals of OPNET Modeler software, the implementation that has been made to achieve our designed framework, the state diagram, some coding of the process. The full coding of this research has been attached in Appendix B.

However, some process of the designed framework, we do not need coding program. We can handle it by setting a value of attribute of the protocol such as IP parameter and wireless parameter. Also, the designed process setting a second multicast route to Standby/Active mode can be done by changing the parameter flag at gateway router.