**SCSA2409 – IoT and Cloud Security Lab**

# Experiment 2 – Implement the formation of virtual modes using OMNeT++.

**Aim:** To Implement the formation of virtual modes using OMNeT++.

**Software Required:** OMNeT++ 6.0.1

**Operating System Required:** Windows OS

**Algorithm:**

To build a "network" that consists of two nodes, where one of the nodes will create a packet and the two nodes will keep passing the same packet back and forth.

Let’s call the nodes tic and toc.

**STEP 1:** Start the OMNeT++ IDE by typing omnetpp in your terminal.

In the IDE, choose *New -> OMNeT++ Project* from the menu.

**STEP 2:** A wizard dialog will appear. Enter ‘tictoc’ as project name, choose *Empty project* when asked about the initial content of the project, then click *Finish*. An empty project will be created.

The project will hold all files that belong to our simulation

**STEP 3:** Adding the NED file

To add the file to the project,

**STEP 3.1:** Right-click the project directory in the *Project Explorer* panel on the left, and choose *New -> Network Description File (NED)* from the menu. Enter tictoc1.ned when prompted for the file name.

**STEP 3.2:** The OMNeT++ IDE's NED editor has two modes, *Design* and *Source*; one can switch between them using the tabs at the bottom of the editor. Changes done in one mode will be immediately reflected in the other.

Switch into *Source* mode, and enter the following:

|  |  |
| --- | --- |
|  | **simple** Txc1 |
|  | { |
|  | **gates**: |
|  | **input** in; |
|  | **output** out; |
|  | } |
|  |  |
|  | // |
|  | // Two instances (tic and toc) of Txc1 connected both ways. |
|  | // Tic and toc will pass messages to one another. |
|  | // |
|  | **network** Tictoc1 |
|  | { |
|  | **submodules**: |
|  | tic: Txc1; |
|  | toc: Txc1; |
|  | **connections**: |
|  | tic.out --> { delay = 100ms; } --> toc.in; |
|  | tic.in <-- { delay = 100ms; } <-- toc.out; |
|  | } |

The first block in the file declares Txc1 as a simple module type

The declaration also says that Txc1 has an input gate named in, and an output gate named out.

The second block declares Tictoc1 as a network. Tictoc1 is assembled from two submodules, tic and toc, both instances of the module type Txc1. tic's output gate is connected to toc's input gate, and vice versa.

**STEP 4:** Adding the NED file

To implement the functionality of the Txc1 simple module in C++.

**STEP 4.1:** Create a file named txc1.cc by choosing New -> Source File from the project's context menu and enter the following content:

|  |  |
| --- | --- |
|  | #include <string.h> |
|  | #include <omnetpp.h> |
|  |  |
|  | using namespace omnetpp; |
|  |  |
|  | /\*\* |
|  | \* Derive the Txc1 class from cSimpleModule. In the Tictoc1 network, |
|  | \* both the `tic' and `toc' modules are Txc1 objects, created by OMNeT++ |
|  | \* at the beginning of the simulation. |
|  | \*/ |
|  | **class** Txc1 : **public** cSimpleModule |
|  | { |
|  | **protected**: |
|  | // The following redefined virtual function holds the algorithm. |
|  | virtual **void** initialize() override; |
|  | virtual **void** handleMessage(cMessage **\***msg) override; |
|  | }; |
|  |  |
|  | // The module class needs to be registered with OMNeT++ |
|  | Define\_Module(Txc1); |
|  |  |
|  | **void** Txc1::initialize() |
|  | { |
|  | // Initialize is called at the beginning of the simulation. |
|  | // To bootstrap the tic-toc-tic-toc process, one of the modules needs |
|  | // to send the first message. Let this be `tic'. |
|  |  |
|  | // Am I Tic or Toc? |
|  | **if** (strcmp("tic", getName()) **==** 0) { |
|  | // create and send first message on gate "out". "tictocMsg" is an |
|  | // arbitrary string which will be the name of the message object. |
|  | cMessage **\***msg **=** **new** cMessage("tictocMsg"); |
|  | send(msg, "out"); |
|  | } |
|  | } |
|  |  |
|  | **void** Txc1::handleMessage(cMessage **\***msg) |
|  | { |
|  | // The handleMessage() method is called whenever a message arrives |
|  | // at the module. Here, we just send it to the other module, through |
|  | // gate `out'. Because both `tic' and `toc' does the same, the message |
|  | // will bounce between the two. |
|  | send(msg, "out"); // send out the message |
|  | } |

The Txc1 simple module type is represented by the C++ class Txc1. The Txc1 class needs to subclass from OMNeT++'s cSimpleModule class, and needs to be registered in OMNeT++ with the Define\_Module() macro.

**STEP 4.2:** Redefine two methods from cSimpleModule: initialize() and handleMessage().

In initialize() we create a message object (cMessage), and send it out on gate out. Since this gate is connected to the other module's input gate, the simulation kernel will deliver this message to the other module in the argument to handleMessage()

after a 100ms propagation delay assigned to the link in the NED file. The other module just sends it back (another 100ms delay), so it will result in a continuous ping-pong.

## STEP 5: Adding the omnetpp.ini file:

To be able to run the simulation, we need to create an omnetpp.ini file. omnetpp.ini tells the simulation program which network you want to simulate.

**STEP 5.1:** Create an omnetpp.ini file using the File -> New -> Initialization file (INI) menu item. The new file will open in an Inifile Editor. As the NED Editor, the Inifile Editor also has two modes, Form and Source, which edit the same content.

**STEP 5.2:** Switch to *Source* mode and enter the following:

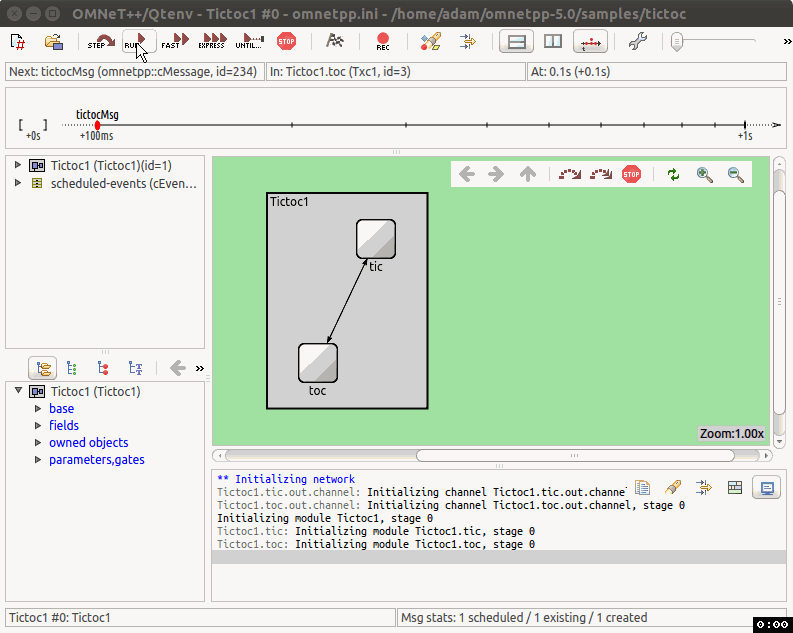
[General]

network = Tictoc1

You can verify the result in *Form* mode

We are now done with creating the model, and ready to compile and run it.

**Output:**



**Result:**

Thus the formation of virtual modes using OMNeT++ is implemented successfully.