

Network Simulator, Technique of implementing the network on the computer

Shikha Saxena¹, Suraiya Husain²

¹Mtech student, Jamia Hamdard, New Delhi

²Asst. Professor, Jamia Hamdard, New Delhi

ABSTRACT- In the network research area, establishing of network in a real time scenario is very difficult. A single test takes a large amount of time and cost. So implementation of a whole network in real world is not easily possible and very costly too. The simulator helps the network developer to check whether the network is able to work in the real time. Thus both the time and cost of testing the functionality of network have been reduced and implementations are made easy. This paper introduce the main features of different simulator. Further details of Omnet++ simulator, appropriate network simulators for research. Further in this paper proposes a language for the description of model topologies in discrete event simulators. The language contains an efficient way to create parametrized, flexible topologies. The language has been implemented as part of the OMNeT++ simulator. Omnet++ simulator (1) provide explicit support,(2) not only fixed topologies are supported, (3) flexible topologies require little bit programming. OMNeT++ uses a description language with a powerful combination of simple constructs (multiple connections, conditional connections etc.) to allow parameterized description of regular structures. This paper presents comparative study of different types of simulators and few examples topology templates as a tool for reusing interconnection structure, presents three general patterns of using the tools of the language, and describes the issues of creating complex, parameterized structures in a simulation program at run-time on omnet++.

KEYWORDS-Topology, Description Language, OPNET, NetSim, JSIM,QUAL NET,OMNET,GloMosim,NED,INI

I. INTRODUCTION

Simulation is one of the important technologies in modern time. The simulation in computer can model hypothetical and real-life objects on a computer so that it can be studied. The network is also simulated on the computer. A network simulator is a technique of implementing the network on the computer. Through this the behavior of the network is calculated either by network entities interconnection using mathematical formulas, or by capturing and playing back observations from a production network. "The Network Simulator provides an integrated, versatile, easy-to-use GUI-based network designer tool to design and simulate a network with SNMP, TL1, TFTP, FTP, Telnet and Cisco IOS device." [3]. Network simulator allows the researchers to test the scenarios that are difficult or expensive to simulate in real world. It particularly useful to test new networking protocols or to changes the existing protocols in a controlled and reproducible environment. One can design different network

topologies using various types of nodes (nodes, hubs, bridges, routers and mobile units etc.). The network simulators are of different types which can be compared on the basis of: range (from the very simple to the very complex), specifying the nodes and the links between those nodes and the traffic between the nodes, specify everything about the protocols used to handle traffic in a network, graphical applications (allow users to easily visualize the workings of their simulated environment.), text-based applications (permit more advanced forms of customization) and programming-oriented tools (providing a programming framework that customizes to create an application that simulates the networking environment to be tested.) [2]. There are different network simulators with different features. Some of the network simulator are OPNET, NS2, NS3, NetSim, OMNeT++, REAL, J-Sim and QualNet. In this paper present working on omnet++ simulator.

II. COMPARITIVE STUDY OF DIFFERENT TYPES OF SIMULATORS:

NS2: Network simulator 2 has been developed under the VINT (Virtual Inter Network Testbed) project; in 1995 it is a joint effort by people from University of California at Berkeley, University of Southern California's Information Sciences Institute, Lawrence Berkeley National Laboratory and Xerox Palo Alto Research Center. The main sponsors are the Defense Advanced Research Projects Agency and the National Science Foundation. It is a discrete event simulator that provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks.

NS3: The ns-3 simulator is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project developing ns-3. Ns-3 is free software, licensed under the GNU GPLv2 license.

OPNET: This simulator is developed by OPNET technologies; Inc. OPNET had been originally developed at the Massachusetts Institute of Technology (MIT) and since 1987 has become commercial software. It provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations.

NETSIM: NetSim is a discrete event simulator developed by Tetcos in 1997, in association with Indian Institute of Science. NetSim has also been featured with Computer Networks and Internets V edition by Dr. Douglas Comer, published by Prentice Hall. It has an object-oriented system modeling and simulation (M&S) environment to support simulation and

analysis of voice and data communication scenarios for High Frequency Global Communication Systems (HFGCS).

OMNET++: It is a component-based, modular and open architecture discrete event simulator framework. The most common use of OMNeT++ is for simulation of computer networks, but it is also used for queuing network simulations and other areas as well. It is licensed under the its own Academic Public License, which allows GNU Public License-like freedom but only in noncommercial settings. It provides component architecture for models.

JSIM: JSim has been developed by a team at the Distributed Real-time Computing Laboratory (DRCL). The project has been sponsored by the National Science Foundation (NSF), DARPA's Information Technology Office, Air Force Office of Scientific Research's Multidisciplinary University

Research Initiative, the Ohio State University and the University of Illinois at Urbana-Champaign. J-Sim is free and available with source code.

QualNet: It is a commercial network simulator from Scalable Network Technologies, Inc in 2000-2001. It is ultra highfidelity network simulation software that predicts wireless, wired and mixed-platform network and networking device performance. A simulator for large, heterogeneous networks and the distributed applications that execute on such networks.

REAL: It is in Computer Science Department Technical Report 88/472, UC Berkeley, 1988. REAL is a simulator for studying the dynamic behavior of flow and congestion control schemes in packet switch data networks. It provides users with a way of specifying such networks and to observe their behavior.

Attribute→ Simulator↙	Devel oped in	Licensed -by	Availability (web site)	Language	Use (free/commercial)
NS-2	1995	VINT (Virtual Inter Network Testbed) project	http://www.isi.edu/nsnam/ns/ns-build.html	C++, Octl	Free
NS-3	2006	GNU GPLv2 license	http://www.nsnam.org/ns-3-13/download/	C++, Python	Free
OP-NET	1987	Massachusetts Institute of Technology	http://www.opnet.com/university_program/itguru_academic_editi on/	C(C++)	Commercial
NETSIM	1997	Tetcos+ Indian Institute of Science	http://www.ssfnet.org/download/li cense.html	Java	Commercial
OMNET++	2003	Own Academic Public License	http://www.omnetpp.org/component/docman/cat_view/17-downloads/1-omnet-releases	C++	Free
JSIM	2000-01	Distributed Real-time Computing Laboratory (DRCL)	http://www.cs.cornell.edu/skeshav/real/overview.html	Java, Tcl	Free
QUALNET	1980	GloMosim	https://sites.google.com/site/jsimofficial/downloads	Visual C++	Commercial
REAL	1988	Computer Science Department Technical Report 88/472, UC Berkeley	http://www.it.iitb.ac.in/~qualnet/	C	Free
JSIM	2000-01	Distributed Real-time Computing Laboratory (DRCL)	http://www.cs.cornell.edu/skeshav/real/overview.html	Java, Tcl	Free

Table1. Comparative study of different type of Simulator

III. WHY OMNET++

By comparing with other simulators the network simulation scene has changed a lot in the past ten years, simulation tools

coming and going. This section presents an overview of various commercial and noncommercial network simulation tools in wide use today, and compares them to OMNeT++.

Specialized network simulator. Also, the discussion only covers the features and services of the simulation environments themselves, but not the availability or NS-2 [6] is currently the most widely used network simulator in academic and research circles. NS-2 does not follow the same clear separation of simulation kernel and models as OMNeT++: the NS-2 distribution contains the models together with their supporting infrastructure, as one inseparable unit. This is a key difference: the NS-2 project goal is to build a *network simulator*, while OMNeT++ intends to provide a *simulation platform*, on which various research groups can build their own simulation frameworks. The latter approach is what called the abundance of OMNeT++ based simulation models and model frameworks into existence, and turned OMNeT++ into a kind of an “ecosystem”. NS-2 does not follow the same clear separation of simulation kernel and models as OMNeT++: the NS-2 distribution contains the models together with their supporting infrastructure, as one inseparable unit. NS-2 lacks many tools and infrastructure components that OMNeT++ provides: simulator port for hierarchical models, a graphical editor, GUI-based execution environment (except for *nam*), separation of models from experiments, graphical analysis tools, simulation library features such as multiple RNG streams with arbitrary mapping and result collection, seamlessly integrated parallel simulation support, etc. This is because the NS-2 project concentrates on developing the simulation models, and much less on simulation infrastructure. This architecture makes it practically impossible to create graphical editors. NS-3 is an ongoing effort to consolidate all patches and recently developed models into a new version of NS. Although work includes refactoring of the simulation core as well, the concepts 2 In fact, OTcl, which is an object-oriented extension to Tcl. 3 Generating a Tcl script from a graphical representation is of course possible, but not the other way round: no graphical editor will ever be able to understand an arbitrary NS-2 script, and let the user edit it graphically. The NS-3 project goals [7] include some features (e.g. parallel simulation, use of real-life protocol implementations as simulation models) that have already proven to be useful with OMNeT++. J-Sim [8][9] (formerly known as JavaSim) is a componentbased, compositional simulation environment, implemented in Java. J-Sim is similar to OMNeT++ in that simulation models are hierarchical and built from self-contained components, but the approach of assembling components into models is more like NS-2: J-Sim has the same drawback as with NS-2: it makes implementing graphical editors impossible. In fact, J-Sim does provide a graphical editor (*gEditor*), but its native format is XML. Although *gEditor* can export Tcl scripts, developers recommend that XML files are directly loaded into the simulator, bypassing Tcl. This way, XML becomes the equivalent of OMNeT++ NED. However, the problem with XML as native file format is that it is hard to read and write by humans. Simulation models are provided in the *Inet* package, which contains IPv4, TCP, MPLS and other protocol models. The fact that J-Sim is Java-based has some implications. On one hand, model development and debugging can be significantly faster than C++, due to existence of excellent Java development tools. However, simulation performance is

characteristics of specific simulation models like IPv6 or QoS (the reason being that they do not form part of the OMNeT++ simulation package.)

significantly weaker than with C++, and it is also not possible to reuse existing real-life

protocol implementations written in C as simulation models. (The feasibility and usefulness of the latter has been demonstrated with OMNeT++, where simulation models include port of the Quagga Linux routing daemon, the TCP stack from the FreeBSD kernel, the port of the UU-AODV routing package, etc. The NS-3 team has similar plans as well.) OPNET Modeler is the flagship product of OPNET Technologies Inc. [10]. OPNET Modeler is a commercial product.

OPNET and OMNeT++ provide rich simulation libraries of roughly comparable functionalities. The OPNET simulation library is based on C, while the one in OMNeT++ is a C++ class library. OPNET's architecture is similar to OMNeT++ as it allows hierarchical models with arbitrarily deep nesting (like OMNeT++), but with some restrictions (namely, the “node” level cannot be hierarchical). A significant difference from OMNeT++ is that OPNET models are always of fixed topology, while OMNeT++'s NED and its graphical editor allow parametric topologies. In OPNET, the preferred way of defining network topology is by using the graphical editor. The editor stores models in a proprietary binary file format, which means in practice that OPNET models are usually difficult to generate by program (it requires writing a C program that uses an OPNET API, while OMNeT++ models are simple text files which can be generated e.g. with Perl) Both OPNET and OMNeT++ provide a graphical debugger and some form of automatic animation which is essential for easy model development. OPNET does not provide source code to the simulation kernel (although it ships with the sources of the protocol models). Qualnet [41] is a commercial simulation environment mainly for wireless networks, which has a significant client base in the military. Qualnet has evolved from the Parsec parallel simulation “language”⁴ [11] developed at the UCLA Parallel Computing Laboratory (PCL), and the GloMoSim (Global Mobile system Simulation) model written on top of Parsec. The Parsec language divides the simulation model into *entities*, and provides a minimalistic simulation API (timers, etc) for them. Entities are implemented with coroutines. Because coroutine CPU stacks require relatively large amounts of memory (the manual recommends reserving 200KByte each), it is rarely feasible to map the natural units of the simulation (say, hosts and routers, or protocols) one-to-one onto entities. What GloMoSim and Qualnet models do is implement the equivalent of the OMNeT++ model structure in model space, above the Parsec runtime. The Parsec kernel is only used to provide event scheduling and parallel simulation services. Parsec provides a very efficient parallel simulation infrastructure, and models (GloMoSim and Qualnet simulation models) have been written with parallel execution, resulting in an excellent parallel performance for wireless network simulations.

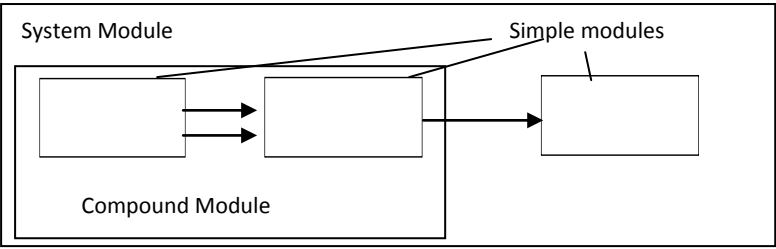
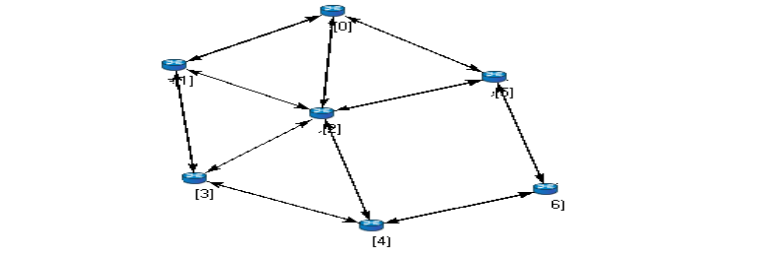
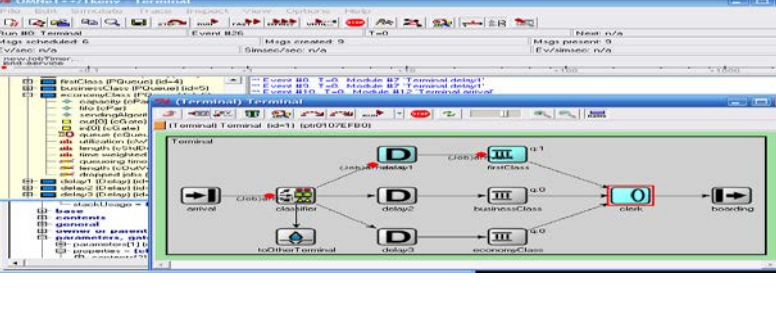

In this section we have examined the simulation packages most relevant for analysis of telecommunication networks, and compared them to OMNeT++. NS-2 is still the most widely used network simulator in the Academia, but it lacks much of



the infrastructure provided by OMNeT++. The other three open-source network simulation packages examined (J-Sim, SSFNet and JiST/SWANS), have failed to gain significant acceptance, and their project web pages indicate near inactivity since 2004. We have examined two commercial

products as well. Qualnet emphasizes wireless simulations. OPNET has similar foundations as OMNeT++, but ships with an extensive model library and provides several additional programs and GUI tools.

IV. MODEL STRUCTURE IN OMNET++

1.		<p>An OMNeT++ model is built from components (modules) which communicate by exchanging messages. Modules can be nested, that is, several modules can be grouped together to form a compound module. When creating the model, you need to map your system into a hierarchy of communicating modules.</p>
2.		<p>Define the model structure in the NED language. You can edit NED in a text editor or in the graphical editor of the Eclipse-based OMNeT++ Simulation IDE.</p>
3.	<pre>class star : public cSimpleModule { protected: // state variables // ... virtual void processMsgFromHigherLayer(cMessage *packet); virtual void initialize(); virtual void handleMessage(cMessage *msg); };</pre>	<p>The active components of the model (<i>simple modules</i>) have to be programmed in C++, using the simulation kernel and class library.</p>
4.	<pre>[General] network = FifoNet sim-time-limit = 100h cpu-time-limit = 300s #debug-on-errors = true #record-eventlog = true [Config Fifo1] description = "low job arrival rate" **.gen.sendInterval = exponential(0.3s) **.gen.msgLength = 100b **.fifo.bitsPerSec = 1000bps</pre>	<p>Provide a suitable omnetpp.ini to hold OMNeT++ configuration and parameters to your model. A config file can describe several simulation runs with different parameters.</p>
5.		<p>Build the simulation program and run it. You'll link the code with the OMNeT++ simulation kernel and one of the user interfaces OMNeT++ provides. There are command line (batch) and interactive, graphical user interfaces.</p>
6.		<p>Simulation results are written into output vector and output scalar files. You can use the Analysis Tool in the Simulation IDE to visualize them. Result files are text-based, so you can also process them with R, Matlab or other tools.</p>

V. CREATING A FIRST NETWORK

In OMNeT++, networks and nodes inside the network are described using the NED (NETwork Description) language.

A- Objective

In this first step, we will build a two-node wired network. At simulation startup, the first node will send a message to the second node. Then, upon receiving, each node will re-send the message back to the sender.

B- Network topology

First, create a project directory called check as a subdirectory of Development In project directory, we will now describe what will be the basic topology of our simulated network. Therefore, create a file called Net.ned and open it for edition. Then insert the following code:

Create NED file

Net.ned

```
network net
{
  @display("bgb=178,112");
  submodules:
    computer1: computer {
      @display("p=28,44;b=30,43");
    }
    computer2: computer {
      @display("p=138,44;b=20,49");
    }
  connections:
    computer1.out --> computer2.in;
    computer2.out --> computer1.in;
}
```

Create package file and ini file

Package.ned

```
@license(omnetpp);
```

Omnnetpp.ini

[General]

```
network = net
```

Create C++ program

computer.cc

```
#include <string.h>
#include <omnetpp.h>

class computer : public cSimpleModule
{
protected:
  virtual void initialize();
  virtual void handleMessage(cMessage *msg);
};
Define_Module(computer);
```

```
void computer::initialize()
{
  if (strcmp("computer1", getName()) == 0)
  {
    cMessage *msg = new cMessage("checkMsg");
    send(msg, "out");
  }
}

void computer::handleMessage(cMessage *msg)
{
  send(msg, "out");
}
```

C- Simulation configuration, build and run

OMNeT++ graphical environment right click on check project then build network to simulate and press "Run". We can see the two modules exchanging the message called cMessage check message.

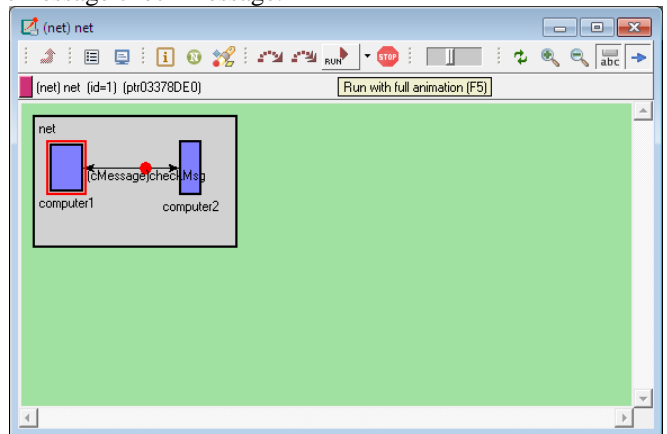


Fig1: two nodes communication

Generating ring topology

To generate ring topology with random number of nodes, modify the Sim module definition as follows:

Define ned file

```
//
// A generated network with ring topology.
//

Net.ned
network Net
{
  submodules:
    node0: Node {
      @display("p=200,50");
    }
    node1: Node {
      @display("p=306,94");
    }
    node2: Node {
      @display("p=350,201");
    }
}
```

```

}
node3: Node {
  @display("p=306,307");
}
node4: Node {
  @display("p=199,350");
}
node5: Node {
  @display("p=93,307");
}
node6: Node {
  @display("p=50,200");
}
node7: Node {
  @display("p=93,94");
}
}
connections:
node0.next <--> DelayChannel <--> node1.prev;
node1.next <--> DelayChannel <--> node2.prev;
node2.next <--> DelayChannel <--> node3.prev;
node3.next <--> DelayChannel <--> node4.prev;
node4.next <--> DelayChannel <--> node5.prev;
node5.next <--> DelayChannel <--> node6.prev;
node6.next <--> DelayChannel <--> node7.prev;
node7.next <--> DelayChannel <--> node0.prev;
}
  
```

This description will generate a seven nodes connected to each other.

Performing simple routing

The SimpleMessage class allows to store the source and destination address of each message. This will be useful for routing operation. Therefore we propose to modify the nodet model by creating the following functions : generateMessage() . The node with index 0 will generate a SimpleMessage at initialization; forwardMessage() that will randomly select a next hop amongst neighbors (i.e. a random out gate in the gate vector) to forward the received message. Each time a message is received, a node will forward it if it is not the final destination. Otherwise, it will delete it and generate a new message.

Create header file

node.h

```

#ifndef NODE_H_
#define NODE_H_
#include <iostream.h>
#include <string.h>
#include <omnetpp.h>
#include "SimpleMessage_m.h"

class Node : public cSimpleModule
  
```

```

{
private:
  int counter;
  int limit;

protected:
// The following redefined virtual function holds the
algorithm.
  virtual void initialize();
  virtual void handleMessage(cMessage *msg);
  virtual SimpleMessage *generateMessage();
  virtual void forwardMessage(SimpleMessage
*msg);
};
#endif /* NODE_H_ */
  
```

Write source code (C++ code)

```

Node.cc
#include "Node.h"
// The module class needs to be registered with OMNeT++
Define_Module(Node);
void Node::initialize()
{
  Variables initialization here.
}

if (...)
{
  ev << "I have index 0, sending initial message" << endl;
  SimpleMessage *msg = generateMessage();
  forwardMessage(msg);
}

void Node::handleMessage(cMessage *msg)
{
  SimpleMessage *smsg = check_and_cast<SimpleMessage
*>(msg);
  // I am final destination
  if (...)
  {
    ev << "Message " << smsg << " arrived after " << smsg
>getHopCount() << " hops.\n";
    delete smsg;
    // Generate another one.
    ev << "Generating another message: ";
    SimpleMessage *newmsg = generateMessage();
    forwardMessage(newmsg);
  }
  else
  // I am not final destination
  {
    forwardMessage(smsg);
  }
}

SimpleMessage* Node::generateMessage()
// Produce source and random destination addresses.
  
```

```
// Create message object and set source and destination
// field.
SimpleMessage *msg = new
SimpleMessage(msgname);
msg->setSource(src);
msg->setDestination(dest);
return msg;
}
void Node::forwardMessage(SimpleMessage *msg)
{
// Increase hop count
...
// Randomly select a next hop k
...
ev << "Forwarding message " << msg << " on port out[" <<
k << "]\n";
send(msg, "out", k);
}
```

The handleMessage() function has been defined with a pointer to an object of generic class cMessage as argument. Here we use objects of class SimpleMessage instead. So as to have access to specific fields of the SimpleMessage class, it is required to cast the pointer to this more precise object type. The check_and_cast template method allows to do this safely, i.e. when trying to cast to an improper destination type, the simulation will crash and stop with an error message reporting the wrong casting attempt.

Simulation configuration, build and run

Change the omnetpp.ini file and package file then built project and run it.

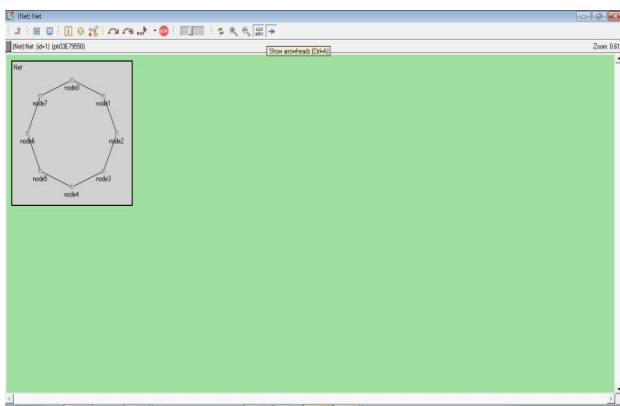


Fig2: Ring Topology

Generating star Topology: Topology can be generated by random number of nodes that can be seen in the following ned file.

Create ned file

```
import ned.DelayChannel;
module Node {
```

```
parameters:
  @display("i=misc/node_vs");
gates:
  inout hub;
connections allowunconnected:
}
module Hub {
parameters:
  @display("i=misc/node_vs");
gates:
  inout spoke[];
connections allowunconnected:
}
//
// A generated network with star topology.
//
network Net
{
parameters:
  int n = default(6);
submodules:
  hub: Hub {
    gates: spoke[n];
  }
  node[n]: Node {
    @display("p=,,ring,");
  }
}
connections:
  for i=0..n-1 {
    hub.spoke[i] <--> DelayChannel <--> node[i].hub;
  }
}
```

write source code and build and run, the topology will be shown like figure.

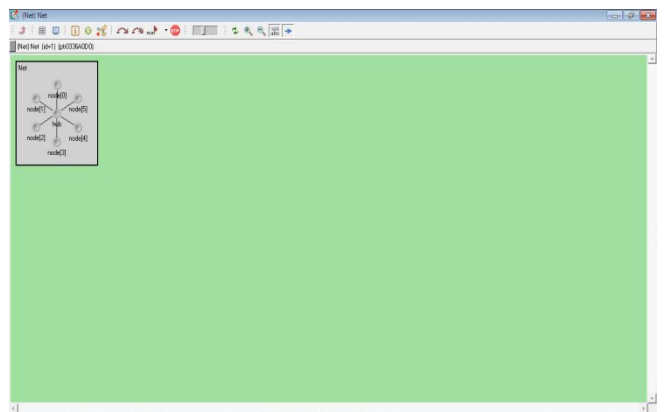


Fig3: Star Topology

Like these basic topologies we can create tree ,hybrid and grid structure of network.

5. CONCLUSIONS

This paper presented an overview of the OMNeT++ discrete event simulation platform, designed to support the simulation of telecommunication networks and other parallel and distributed systems. The OMNeT++ approach significantly differs from that of NS-2, the most widely used network simulator in academic and research circles: while the NS-2 (and NS-3) project goal is to build a network simulator, OMNeT++ aims at providing a rich simulation platform, and leaves creating simulation models to independent research groups. The last ten years have shown that the OMNeT++ approach is viable, and several OMNeT++-based open-source simulation models and model frameworks have been published by various research groups and individuals.

6. REFERENCES

- [1] Jianli Pan, Prof. Raj Jain, Project, "A Survey of Network Simulation Tools: Current Status and Future Developments", report.
- [2] András Varga, Rudolf Hornig "AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT", SIMUTools, March 03 – 07, 2008, Marseille, France.
- [3] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku-Badgujar, "Network Simulation Tools Survey", *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 1, Issue 4, June 2012
- [4] OMNeT++ Home Page. <http://www.omnetpp.org> [accessed on September, 2007]
- [5] Varga, A. 2001. The OMNeT++ Discrete Event Simulation System. In the Proceedings of the European Simulation Multiconference (ESM2001. June 6-9, 2001. Prague, Czech Republic).
- [6] Bajaj, S., L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu and D. Zappala. 2000. Improving simulation for network research. *IEEE Computer*. (to appear, a preliminary draft is currently available as USC technical report 99-702)
- [7] T. R. Henderson, S. Roy, S. Floyd, G. F. Riley. ns3 Project Goals. WNS2 ns-2: The IP Network Simulator, Pisa, Italy-Oct.10,2006. <http://www.nsnam.org/docs/meetings/wns2/wns2-ns3.pdf>
- [8] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, J-Sim: a simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications Magazine*, Vol. 13, No. 4, pp. 104--119, August 2006.
- [9] J-SIM home page: <http://www.j-sim.org> [accessed on September, 2007]
- [10] OPNET Technologies, Inc. *OPNET Modeler*. <http://www.opnet.com> [accessed on September, 2007]
- [11] Bagrodia, R, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, H. Song. 1998. Parsec: A Parallel Simulation Environment for Complex Systems. *Computer*, Vol. 31(10), October, pp. 77-85.
- [12] Kaage, U., V. Kahmann, F. Jondral. 2001. An OMNeT++TCP Model. To appear in *Proceedings of the European Simulation Multiconference (ESM 2001)*, June 7-9, Prague.
- [13] Wehrle, K, J. Reber, V. Kahmann. 2001. "A Simulation Suite for Internet Nodes with the Ability to Integrate Arbitrary Quality of Service Behavior". In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference 2001*, Phoenix (AZ), USA, January 7-11.
- [14] MiXiM home page. <http://sourceforge.net/projects/mixim/> [accessed on September, 2007]
- [15] JiST home page. <http://jist.ece.cornell.edu> [accessed on September, 2007]
- [16] Varga, A. and Gy. Pongor. 1997. Flexible Topology Description Language for Simulation Programs. In *Proceedings of the 9th European Simulation Symposium (ESS'97)*, pp.225-229, Passau, Germany, October 19-22.
- [17] Varga, A and B. Fakhamzadeh. 1997. The K-Split Algorithm for the PDF Approximation of Multi-Dimensional Empirical Distributions without Storing Observations. In *Proc. of the 9th European Simulation Symposium (ESS'97)*, pp.94-98. October 19-22, Passau, Germany.
- [18] Varga, A. 1998. Parameterized Topologies for Simulation Programs. In *Proceedings of the Western Multiconference on Simulation (WMC'98), Communication Networks and Distributed Systems (CNDS'98)*. San Diego, CA, January 11-14.
- [19] Consensus homepage. <http://www.consensus.tudelft.nl/software.html> [accessed on September, 2007]
- [20] Field Bus homepage. <http://developer.berlios.de/projects/fieldbus> [accessed on September, 2007]
- [21] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl, A Mobility Framework for OMNeT++. 2003. 3rd International OMNeT++ Workshop (Budapest University of Technology and Economics, Department of Telecommunications Budapest, Hungary, January 2003). <http://www.tkn.tu-berlin.de/~koepke/>
- [22] OverSim: The Overlay Simulation Framework <http://www.oversim.org> [accessed on September, 2007]
- [23] Ingmar Baumgart and Bernhard Heep and Stephan Krause. 2007. OverSim: A Flexible Overlay Network Simulation Framework. *Proceedings of 10th IEEE Global Internet Symposium* (May, 2007). p.79-84.
- [24] Ingmar Baumgart and Bernhard Heep and Stephan Krause. 2007. A P2PSIP Demonstrator Powered by OverSim. *Proceedings of 7th IEEE International Conference on Peer-to-Peer Computing (P2P2007, Galway, Ireland. Sep, 2007)*. pp. 243-244,
- [25] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, A. Durresi, and S. Sastry. 2005. Simulating Wireless Sensor Networks with OMNeT++ , submitted to *IEEE Computer*, 2005 http://csc.lsu.edu/sensor_web/publications.html

- [26] Sensor Simulator. http://csc.lsu.edu/sensor_web [accessed on September, 2007]
- [27] S. Valentin. 2006. ChSim - A wireless channel simulator for OMNeT++, (TKN TU Berlin Simulation workshop, Sep. 2006) <http://www.cs.uni-paderborn.de/en/researchgroup/research-group-computernetworks/projects/chsim.html>
- [28] I. Dietrich, C. Sommer, F. Dressler. Simulating DYMO in OMNeT++. Erlangen-Nürnberg: Friedrich-Alexander- Universität. 2007 Internal report.
- [29] Isabel Dietrich, Volker Schmitt, Falko Dressler and Reinhard German, 2007. "SYNTONY: Network Protocol Simulation based on Standard-conform UML 2 Models," Proceedings of 1st ACM International Workshop on Network Simulation Tools (NSTools 2007), Nantes, France, October 2007.
- [30] Feng Chen, Nan Wang, Reinhard German and Falko Dressler, 2008. "Performance Evaluation of IEEE 802.15.4 LR-WPAN for Industrial Applications," Proceedings of 5th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (IEEE/IFIP WONS 2008), Garmisch-Partenkirchen, Germany, January 2008.