Network Simulation Tools Survey

Mrs. Saba Siraj¹, Mr. Ajay Kumar Gupta², Mrs Rinku-Badgujar³
Department of Computer Science and Engineering, PGMCOE, Wagholi, Pune¹,²,³

Abstract—
In the network research area, establishing of network in a real time scenario is very difficult. A single test bed takes a large amount of time and cost. So implementation of a whole network in real world is not easily possible and very costly to. 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. In this paper, we introduce the main features of different simulator and consider their advantages and disadvantages. We hope this survey prove to be a good reference source for those people who feel difficult to select the appropriate network simulators for their research.

Keywords: Network Simulator, NS², NS³, OPNET, NetSim, OMNeT++, REAL, J-Sim and QualNet

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, TLI, 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 (hosts, 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 we are working on some of the simulator.

NS² (Network Simulator version2): NS² is a discrete event simulator targeted at networking research. It provides support for simulation of TCP, routing, and multicast protocols over all networks (wired and wireless).

NS³ (Network Simulator version3): NS³ is also an open sourced discrete-event network simulator which targets primarily for research and educational use. NS³ is licensed under the GNU GPLv2 license, and is available for research and development.

OPNET (Optimized Network Engineering Tools): It is extensive and powerful simulation software with wide variety of possibilities to simulate entire heterogeneous networks with various protocols

NETSIM (Network Based Environment for Modelling and Simulation): It is an application that simulates Cisco Systems networking hardware and software and is designed to aid the user in learning the Cisco IOS command structure.

OMNET++ (Optical Micro-Networks Plus Plus): It is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators.

JSIM (Java-based simulation): It is a Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data.
JSim is an application development environment based on the component-based software architecture.

QUALNET: It is a commercial version of GloMoSim used by Scalable Network Technologies for their defense projects.

REAL (REalistic And Large): It is a network simulator originally for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks. It provides users with a way of specifying such networks and to simulate their behavior.

II. INTRODUCTION OF NETWORK SIMULATORS

NS3: 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.

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 2000-2001. It is ultra high-fidelity 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.

III. SOFTWARE AVAILABILITY

There are different types of network simulator. And these simulators are available free or commercially over the network. In this section we are divide these simulator on their easy availability. The Table1 shows the availability of software and the url adders where the setup of software is found.
TABLE I
SOFTWARE AVAILABILITY

<table>
<thead>
<tr>
<th>Name of Network Simulator</th>
<th>Availability(site)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS2</td>
<td>Free for use</td>
</tr>
<tr>
<td>NS3</td>
<td>Free for use</td>
</tr>
<tr>
<td>OPNET</td>
<td>Commercial network simulator</td>
</tr>
<tr>
<td>NetSim</td>
<td>Commercial network simulator</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>Fee for academic and non-profit use</td>
</tr>
<tr>
<td>REAL</td>
<td>Free for use</td>
</tr>
<tr>
<td>J-Sim</td>
<td>Free for use</td>
</tr>
<tr>
<td>QualNet</td>
<td>Commercial simulator</td>
</tr>
</tbody>
</table>

IV. LANGUAGE USED AND COMPONENT DIAGRAM OF SIMULATORS

In this section we are defining the languages that are used by different network simulator. We also define why these languages are used in the simulator.

**NS2:**

C++: C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. [17]

**OTcl:** OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. Ns provides glue to make objects and variables appear on both languages. The component diagram of ns2 is given in Figure 1.

**NS3:**

C++: implementation of simulation and core model.Ns-3 is built as a library which may be statically or dynamically linked to a C++ main program. These libraries define the start of simulation and simulation topology.

**Python:** C++ wrapped by Python. Python programs to import an “ns3” module. The component diagram of ns3 is given in Figure 2.

**OPNET:**

C (C++): The main programming language in OPNET is C (recent releases support C++ development). The initial configuration (topology setup, parameter setting) is usually achieved using Graphical User Interface (GUI), a set of XML files or through C library calls. Simulation scenarios (e.g., parameter change after some time, topology update, etc.) usually require writing C or C++ code; although in simpler cases one can use special “scenario” parameters (e.g., link fail/restore time) [13]. The component diagram of OPNET is given in Figure 3.

---

**Copyright to IJARCCE**

www.ijarcce.com

203
OMNET++:

C++: A C++ class library which consists of the simulation kernel and utility classes (for random number generation, statistics collection, topology discovery etc) -- this one you will use to create simulation components (simple modules and channels); infrastructure to assemble simulations from these components and configure them (NED language, ini files); runtime user interfaces or environments for simulations (Tkenv, Cmdenv); an Eclipse-based simulation IDE for designing, running and evaluating simulations; extension interfaces for real-time simulation, emulation, MRIP, parallel distributed simulation, database connectivity and so on. [15]. The component diagram of OMNET++ is given in Figure 5.

JSIM:

Java: Java is easy to learn and easy to use. In case of any problems, source texts provided with J-Sim can be used to generate new code, compiled in the target environment, thus 100-percent compatible with JVM used. Java provides a class called Thread whose instances run parallel with other such instances. Java is a fully object-oriented language, providing the concepts of classes, instances, encapsulation, inheritance and polymorphism. J-Sim provides basic classes for simulation, process and queue. These classes can be either directly used or extended according to specific user's requirements.

Tcl: Scripting is an essential part of J-Sim, use it to "glue" all the components and define how the system operates. It makes it possible to manipulate Java objects in the Tcl environment, such as creating an object from a Java class, invoking a method of a Java object, or accessing a field variable of a Java object. The component diagram of JSIM is given in Figure 6.

QualNet:

C++: For implementing new protocols, Qualnet uses C/C++ and follows a procedural paradigm. Uses the parallel simulation environment for complex systems (PARSEC) for basic operations, hence can run on distributed machines [15]. The component diagram of QUALNET is given in Figure 7.

REAL:

C: The actions are C code that creates the graph structure. The combination of a workload and flow control protocol is implemented by a single C function. Each such function is executed in parallel by the underlying thread-based simulation package, and can be thought of as being an independently scheduled and non-preemptable entity. The component diagram of REAL is given in Figure 8. The summary of this section is given in Table II that tells about languages used by different simulators.

<table>
<thead>
<tr>
<th>Name of Network Simulator</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS2</td>
<td>C++, Otcl</td>
</tr>
<tr>
<td>NS3</td>
<td>C++, Python</td>
</tr>
<tr>
<td>OPNET</td>
<td>C (C++):</td>
</tr>
<tr>
<td>NetSim</td>
<td>Java</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>C++</td>
</tr>
<tr>
<td>REAL</td>
<td>C</td>
</tr>
<tr>
<td>J-Sim</td>
<td>Java, Tcl</td>
</tr>
</tbody>
</table>

Copyright to IJARCE www.ijarce.com 204
V. EXAMPLE OF SIMULATORS

**NS2:** The code of ns2 should be written in tcl. And the output file is shown in nwm window. Here is an example:

```
set node(s)_1 [Ns node]
set node(s)_2 [Ns node]
set node(s)_1 [Ns node]
set node(s)_2 [Ns node]
set node(s)_3 [Ns node]

$s$ duplex-link $node(s)_1$ $node(s)_1$ 10Mb 2ms DropTail
$s$ duplex-link $node(s)_2$ $node(s)_2$ 10Mb 3ms DropTail
$s$ queue-limit $node(s)_1$ $node(s)_2$ 25
$s$ queue-limit $node(s)_2$ $node(s)_1$ 25
$s$ queue-limit $node(s)_3$ $node(s)_2$ 10Mb 4ms DropTail
$s$ duplex-link-op $node(s)_1$ $node(s)_1$ right-down
$s$ duplex-link-op $node(s)_2$ $node(s)_2$ right-upright
$s$ duplex-link-op $node(s)_1$ $node(s)_2$ right-right
$s$ duplex-link-op $node(s)_1$ $node(s)_2$ queuePos 0
$s$ duplex-link-op $node(s)_2$ $node(s)_1$ queuePos 0
$s$ duplex-link-op $node(s)_3$ $node(s)_2$ left-left
$s$ duplex-link-op $node(s)_4$ $node(s)_2$ left-down

set tcp1 [Ns create-connection TCP/Reno $node(s)_1$ TCPSink $node(s)_3]
set tcp2 [Ns create-connection TCP/Reno $node(s)_2$ TCPSink $node(s)_3]

$tcp$ set window_ 15
set tcp2 [Ns create-connection TCP/Reno $node(s)_2$ TCPSink $node(s)_3]

$ftp$ set at 0.0 "$ftp$ start"
set tcp1 [Ns create-connection TCP/Reno $node(s)_1$ TCPSink $node(s)_3]

set tcp1 $tcp1$ get window
set tcp2 $tcp2$ get window
```

**NS3:** The ns3 code is written in .cc file. Example of ns3 is given below [29]

```c++
int main (int argc, char *argv[])
{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    RandomVariable::UseGlobalSeed (1, 2, 3, 5, 8);
    NodeContainer nodes; nodes.Create (2);
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue("2ms"));
    NetDeviceContainer devices;
    devices = pointToPoint.Install (nodes);
    InternetStackHelper stack; stack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);
    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
    serverApps.Start (Seconds (1.0));
    serverApps.Stop (Seconds (10.0));
    UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.1)));
    echoClient.SetAttribute ("PacketSize", UintegerValue (10242));
    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0)); clientApps.Stop (Seconds (10.0));
    Simulator::Run (); Simulator::Destroy (); return 0;
}
```

**OPNET:** The opnet simulator provides a graphical interface for writing the code. Step for writing the code [30]

**Project Editor:**

![Figure 9: Example ns2](image)

![Figure 10: Example ns3](image)
**Initial topology:**

Figure 11a: Example opnet

**Network scale:**

Figure 11b: Example opnet

**Object Palette:**

Figure 11c: Example opnet

**Available configurations:**

Figure 11d: Example opnet

**Choose Results:**

Figure 11e: Example opnet

**NETSIM:** It is graphical software so a graphical platform is provided of the simulation. Example:

Start navigation and add objects:

Add Device, and select 3640 from the list of available routers.

Figure 11f: Example opnet

Figure 11g: Example opnet

Figure 11h: Example netsim

Copyright to IJARCE www.ijarcce.com 206
Add new connection

In the New Connection dialog box for Router1, select the Serial1/0 interface in the Local Interface dropdown list.

Move the devices in your topology around until it looks something like the graphic below:

OMNET++: here is an example of omnet++
// sink1.cpp code
#include <omnetpp.h>
class Sink: public cSimpleModule
{ Module_Class_Members(Sink, cSimpleModule, 8192);
  virtual void activity(); 
};
Define_Module(Sink);
void Sink::activity()
{ while(1)
  { cMessage *msg = receive(); int pkt_type = msg -> kind();
    if (pkt_type == 1) ev << "Received data packet\n";
      else ev << "Received voice packet\n";
      delete msg; } }

JSIM: here is a tcl code with the java library file and the output
# echoer.tcl
cd [mkdir -q drcl.comp.Component /test] 
puts "create topology..."
set link_ [java::new drcl.inet.Link] 
$link_ setPropDelay 0.3; # 300ms 
set adjMatrix_ [java::new [int[][]] 3 {[1] [0 2] [1]}]
java::call drcl.inet.InetUtil createTopology ![.] $AdjMatrix_
$link_ 
puts "create builders..."
set nb [mkdir drcl.inet.NodeBuilder .nodeBuilder]
$nb setBandwidth 1.0e7; puts "build..."
$nb build ![n?] 
$nb build ![h?] {
  Udp drcl.inet.transport.UDP 
  echo 101/udp new_echoer }
! h?/udp setTTL 3
! n1 setBandwidth 1 1.0e4;! n1 setBufferSize 1 6000;
puts "setup static routes..."
java::call drcl.inet.InetUtil setupRoutes ![h0] ![h2]
"bidirection"
puts "set up simulator..."
set sim [attach_simulator .] 
puts "Done!"
QualNet: Here is the step of qualnet simulator work


REAL:

```c
#include "../kernel/real.h"
ecn_dummy() {
    PKT_PTR pkt; int node, num_pkts_sent = 0;
    ident destn, sender, sink; long key;
    longv now;
    int seq_no = 0; node = get_node_id();
    sink = assigned_sink[node];
    abs_advance(node_start_time[node]); now = runtime();
    if(node is 1) {
        make_pkt(pkt);
        pkt->dest = sink;
        pkt->data[0] = num_pkts[node];
        sendm(sink, 0, pkt);
        printf("Node 1 sent request packet\n");
    }
    for (ever) {
        sender = recvm(&destn, &key, &pkt);
        now = runtime();
        switch (pkt->type) {
            case ACK:
                free(pkt);
                break;
            case INT:
                free(pkt);
                break;
            case DATA:
                pkt->type = ACK;
                pkt->dest = pkt->source;
                pkt->source= node;
                sendm(pkt->dest, 0, pkt);
                break;
        }
    }
}
```

VI. MERITS AND DEMERITS OF SIMULATORS

**NS2**: NS-2 provides emulation functionalities. NS-2 can be used for parallel and distributed simulation: PDNS.

**NS3**: Users of ns-3 can construct simulations of computer networks using models of traffic generators, protocols such as TCP/IP, and devices and channels such as Wi-Fi, and analyze or visualize the results.


**NETSIM**: NetSim was used to create a successful national cyber exercise and considers. NetSim has been used to create a fast, functional simulator. NetSim is intended for use within several different defense systems. It will support computer-based collaborative work, such as shared work areas and means of communication.
OMNET++: OMNeT++ is a C++-based discrete event simulator for modeling communication networks, multiprocessors and other distributed or parallel systems.

JSIM: JSIM, a Java-based simulation and animation environment supporting Web-Based Simulation, a rapidly emerging area of simulation research and development. [25]

QualNet: QualNet is designed to simulate large-scale wired and wireless networks with thousands of mobile nodes, each of which may different communication capabilities via multi-hop ground, aircraft and satellite media. [24]

REAL: REAL is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data network? REAL can be modified to analyze modifications to these protocols or alternate protocols.

REFERENCES

[23] András Varga,Rudolf Hornig, AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT, SIMUTools, March 03 – 07, 2008, Marseille, France
[33] Punit Rathod, QualNet Tutorial, 10th Sep 2005

BIography

Ms. Saba Siraj is working as Assistant Professor in Department of Computer Science at PGMCOE, Pune, MH, INDIA. Her Research activities are based on Software Engineering. She had done her M.Tech in Computer Science and Engineering from Aligarh Muslim University.

Mr. Ajay Kumar Gupta is working as Assistant Professor in Department of Computer Science at PGMCOE, Pune, MH, INDIA. He Research activities are based on Software Engineering. He had done her M.Tech in Computer Science and Engineering.
Ms. Rinku Badgujar is working as Assistant Professor in Department of Computer Science at PGMCOE, Pune, MH, INDIA. Her research activities are based on Software Engineering. She is pursuing her M.Tech in Computer Science and Engineering from JNTU, Hyderabad, India.