International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || PP.50-56

Congestion control in Packet Networks and Simulation Tools for Congestion Control Algorithms

Joy Karan Singh[1], Guneet Kaur[2]
[1](Department of Electronics and communication engineering, CTITR, Jalandhar)
[2](Department of Computer Science and engineering, Lovely Professional University, Jalandhar)

Abstract: Congestion is a catastrophic event which depends upon factors such as characteristics of underlying network, the mechanism of transmission protocol and functionality of switching devices, which are routers and gateways. It is important to control the queue of routers for the effective transmission of packets, for which many congestion control algorithms are available. The most basic Active Queue Management (AQM) is Random Early Detection (RED). Later, many enhancements of RED have been proposed. The implementation of these algorithms can be carried out in network simulation that provides a virtual environment in which algorithms can be implemented at relatively lesser cost and time than required to experiment with real implementations. This paper enlightens on popular simulators namely OPNET, QualNET, OMNET++, NS-2 and NS-3. Detailed features of NS-3 have been discussed and its capabilities that makes it worth using simulation tool.

Keywords: Network Simulation, RED, OPNET, QualNET, OMNET++, NS-2, NS-3.

I. Introduction

In a packet switching network, congestion is a state in which performance degrades due to saturation of networking resources such as memory buffers, communication link bandwidth and processor cycles. It is known that a network consists of collection of nodes, communication channel (coaxial wires, ethernet cable, fiber optic etc) and switching devices, which are called as routers and gateways. Any communication channel has a limit to the amount of traffic it can accommodate. Whenever the traffic load exceeds this limit, congestion is said to have occurred.

Switching devices like routers also play an important role in controlling the congestion. A router is responsible for collecting the incoming information, determining the destination from that information and forwarding those packets. Each router has finite amount of buffer space, in which it stores the information before forwarding it to next node. This space is necessary so as to prevent loss of information, as the rate at which router receives packets is greater than that it transmits them. Hence, there is a need for effective and intelligent management of this buffer. A variety of buffer management algorithms have been devised which are also called as Active Queue Management (AQM) algorithms.

AQM algorithms are designed to be implemented at network routers. Routers are best switching devices to control congestion as routers can effectively distinguish between propagation delay and persistent queuing delay. Only routers posses unified views of the queuing behavior over time. The prime motive of these algorithms is to provide a mechanism to detect network congestion early and to start dropping or marking the packets before it affects network bandwidth performance. There is a wide variety of AQM algorithms available, the most common of them is Random Early Detection (RED).

II. Background of Red and Drop Tail

This congestion control algorithm calculates average queue size and notifies the sender either by dropping the packets or marking them. Average queue size is calculated using a low pass filter with an exponential weighted moving average. This algorithm defines two thresholds, minimum threshold and a maximum threshold. When the average queue size is less than the minimum threshold, then no packet is marked. If the queue size lies in between minimum and maximum threshold, then it drops or marks the packet with a probability p_a, which is a function of average queue size avg. This probability is roughly proportional to the connection's share of bandwidth at the router [1]. The general RED algorithm is given below.

```
for each packet arrival
   calculate the average queue size avg
   if min_th ≤ avg < max_th
      calculate probability p_a
      with probability p_a:
         mark the arriving packet
   else if max_th ≤ avg
      mark the arriving packet
```

Fig.1 General Algorithm for RED

Though RED was an improvement over Droptail mechanism in which congestion is detected only by packet loss due to buffer overflow at First In First Out(FIFO) router. Hashem[2] points out that Droptail routers cause global synchronization, a phenomenon in which window size of all the connections is decreased at the same time. This results in a sharp decrease in link utilization and throughput.

In a contradiction to Droptail, RED doesn't make use of global synchronization. But RED suffers from many drawbacks like its sensitivity towards parameter settings, number of competing sources/flows and many others. To minimize these problems, many enhancements of RED were introduced.

Some of those schemes include Stabilized Random Early Drop (SRED), Effective Random Early Drop (ERED), and Gentle Random Early Drop (GRED) etc. Most of the proposed queue management schemes are RED oriented. The only difference in these schemes is mainly in the parameter-adjusting method. All these schemes posses additional parameters and techniques. This also adds extra complexity to parameter setting task. To carry out the implementation of these AQM algorithms, many simulation tools are available that are discussed as below. But before that it is important to understand the concept of network simulation.

III. Network Simulation

Simulation is the imitation of some real thing, state of affairs, or process [3]. This dominant tool is widely used for the improvement and development of new communication architectures and various inter-networking protocols. The escalating requirements with growth and advancements in the networking field, Simulation become one of the important technologies in modern times. It provides a virtual environment which is useful to test new networking protocols, conceptual models, topologies, algorithms or bring any change in the existing protocols/algorithms in a cost effective manner. Any networking scenario which is difficult to implement and involves a handsome cost, can be easily tested on a network simulator.

Most of the network simulation toolkits are based on Discrete Event Simulation (DES) paradigm [3]. For an instance when a packet is sent from one node to the other, then an event is triggered by network nodes. An Event queue is maintained by the simulator which is sorted as per the scheduled event execution time. The simulation is performed by the successive processing of these events present in the queue.

A Network Simulator usually comes with a set of predefined modules and user-friendly GUI. It also allows its user to define various topologies (bus, star, mesh etc) using wide variety of nodes (hosts, hubs, routers, bridges and mobile units etc). They provide a high degree of flexibility in implementation of various network entities in a simulated environment. In this paper, a search is on for the simulator that is best suitable for implementing these schemes for controlling the problem of congestion. Some of the popular simulators have been presented as below:

3.1 Optimized Network Engineering Tool Modeler(OPNET)

This commercial DES was developed by MIT of Technology in 1987 using C++. It is helpful in analysis of realistic simulated networks so as to compare the impact of different technological designs on end to end behavior. OPNET claims to be the fastest simulation engine amongst all leading simulators. It posses many wired/wireless protocols and also allows Object-Oriented modeling of components [13]. It is flexible in nature allowing integration with other libraries and simulators. It provides an excellent support for Graphical User Interface (GUI). But a tool namely OPNET Guru Academic Edition Software provided by above simulator is available free of cost for academic limited use. This software was created for introductory level networking courses, and designed and tested to be used with popular classroom lab manuals. It is based on IT Guru commercial version 9.1 and has a renewable license for 6 months [5].

But there are some analogies in this software, some of which are mentioned below:

- limits maximum number of intermediate nodes by 20
- Doesn’t support other product modules.
- Limited import and export capabilities.

3.2 QualNET

A commercial version of Global Mobile Information System Simulator (GloMoSim) which was first released in 2000 by SNT. It provides a comprehensive environment for implementing various protocols, creating and animating network scenarios and evaluating their performance. It is based on C++. It is available for Linux systems (both 32 and 64 bit versions) and also for Microsoft Windows operating systems [6]. It provides large scalability and has an excellent support for GUI. This simulator also provides standard classes which includes elements like weather impact on communication and terrain which means the physical terrain over which communication has to take place [6].

3.3 OMNET++

Objective Modular Network Testbed was developed by Andre Varge of Technical University of Budapest and was publicly introduced in 1997. It is more of a simulation framework than to be called as network simulation software but its explicit and hardwired support for computer networks provides an infrastructure for writing simulations [7]. It is licensed free of cost only for educational purpose. In OMNET++, the basic entity is called as a module/simple module, which is atomic in nature like a specific protocol. They communicate with each other by sending and receiving messages through gates which are linked via connection. Multiple simple modules can be combined together to form a compound module. A special type of language called as Network Description Language (NED) is required while integrating simple modules into compound module [12]. NED allows specification of variable parameters in the network description. It also poses an output analyzer which is capable of displaying statistical data in graphical format.

3.4 NS-2

It is an Open source Network Simulator introduced in 1996 which gained a lot of popularity in academia field. It is based on two languages [3]:

- C++ code used to model the behavior of simulated nodes, and
- oTcl an object oriented extension of Tcl which serves as an interpreter to execute user command scripts. These scripts are used to further define network scenario say network topology.

The main drawback of this was that there was a need to recompile C++ programs frequently which was a time consuming process.

3.5NS-3

It is a successor of NS-2 which has been introduced in 2006. It is free software, licensed under GNU and GPLv2. The best way to explain this software is by comparing it with its predecessor and the differences are as follows:

3.5.1 Language support

NS-3 is purely C++ based DES which means it allows user to write the simulation code solely in C++ language. It also allows users an option to use Python as a language for writing simulations. Use of Python in place of Otcl provides a scripting language with a larger user base, more active development and friendlier syntax. Moreover the compilation time is reduced when a simulation script is written entirely in C++.

3.5.2 Source code documentation

The documentation is provided by Doxygen, an open source multi-language documentation generator which is capable of parsing and extracting NS-3 documentation directly from the source code. It also allows the documentation to be generated in various different formats like XML, pdf, HTML etc. Whereas NS-2 just provides a web page consisting of several section, tutorials and third party manuals which may suffer from problems such as obsolescence and lack of maintenance.

3.5.3 Packet format

NS-3 packet consists of a single buffer (stream of bits to be sent over the network) of bytes and a collection of small tags containing meta-data which is optional. In a contradiction to it, NS-2 packet consists of 2 different regions which include header and the payload data [8]. It never frees up the memory used to store the packets until the simulation ends.

3.5.4 Simulation portability

NS-3 provides a common library called as*libns3.so* which must be same for all the NS-3 instances belonging to the same release. Any new model or simulation is often placed in *scratch* folder by the developers. As a result of which they will obtain standard binaries that need *libns3.so* shared library so as to use NS-3 features, and they will include custom models as a part of their own binary code[9].NS-2 simulations are portable only if they simply consist of Otcl scripts. However, if in case the script makes use of custom models

developed by third-parties that aren't official parts of NS-2 project, then a custom *ns* executable generated from standard NS-2 source code and the new model code will be required.

3.5.5 Building the source code

NS-3 makes use of *waf* build automation tool, having Python as its only single external dependency [9]. Waf tool is portable in nature. Unlike NS-3, the building process of NS-2 is based on the *make* building tool. A file called as project's main Makefile, has to be edited whenever a new module to NS-2 has to be added or any current configuration has to be changed, which again requires rebuilding the whole source code.
Example for NS-3 build system:

To configure : ./waf -d[optimized|debug] configure
To run : ./waf --run <path/name of script file> (file extension not required)

3.5.6 Performance

NS-3 performs better than NS-2 because of removal of the overhead associated with oTcl interpreter and overhead associated with interfacing oTcl with C++. NS-3 also succeeds its predecessor in terms of managing the memory.

3.5.7 Distribution

NS-3 uses Mercurial, an emerging distributed revision control tool and the project maintainers have packaged the whole set of dependencies in one single installation package, also called as all-in-one. This package contains NS-3 source code which is ready to compile.
In case of any missing software dependency, use command as below:

$ sudo apt-get <dependency/package name>

In contrast, NS-2 relies on CVS which is a classical revision control system for managing its source code. Some of its software dependencies can be obtained through software repositories of main GNU/Linux distributions whereas the other dependencies have to be manually installed because of their absence in mainstream software repositories.

IV. Salient Features of NS-3

4.1 Fundamental objects

NS-3 consists of various key abstractions, important ones are as follows [3,10]:

4.1.1 Node

It is an abstraction for any computing device that connects to a network. This class also provides various methods for managing the representation of computing device present in a network.

4.1.2 Application

This basic abstraction represents any kind of activity generated by the user program. The Application class provides various methods for managing the representation of user-level applications in simulation. The most commonly used classes are UdpEchoClientApplication and UdpEchoServerApplication which consist of a client/server application set to generate and echo network packets.

4.1.3 Channel

It is an abstraction for communication channel which is the media over which data flows over a network. The most commonly used versions of Channel class includes CsmaChannel,PointToPointChannel and WifiChannel.

4.1.4 NetDevice

It is an abstraction for network device drivers used to control network devices like devices managed by device drivers. A NetDevice has to be first installed on a Node so that it can communicate with other Nodes via Channels. The NetDevice class is responsible for providing methods for managing connections to Node and Channel objects. Some specialized versions of this class includes CsmaNetDevice, PointToPointNetDevice and WifiNetDevice.

4.2 Support for emulation

NS-3 has been designed in such a way that it can be integrated into testbed and virtual machine environments. Emulation support is provided by provision of 2 kinds of NetDevice which are as follows:

- Emu NetDevice: it enables NS-3 simulations to send data on a real network.
- Tap NetDevice: it enables a real host to participate in NS-3 simulation as if it were one of the simulated nodes.

4.3 Visualization tool

NS-3 doesn't possess any default animator. It provides software called as "NetAnim" that processes the xml files for graphical output which are generated by NS-3 based on Qt4 Toolkit. This software package can be downloaded from URL http://code.nsnam.org/jabraham3/netanim-3.104

However, Ubuntu users can get it directly by using "apt -get" and installing the required prerequisites. After the successful installation of the required dependencies, a directory named netanim-3.104 will appear in NS-3 all-in-one directory. This animation software allows to add many attributes of networking objects like IP address, Node size, MAC address, Node number etc as shown in Fig. 2

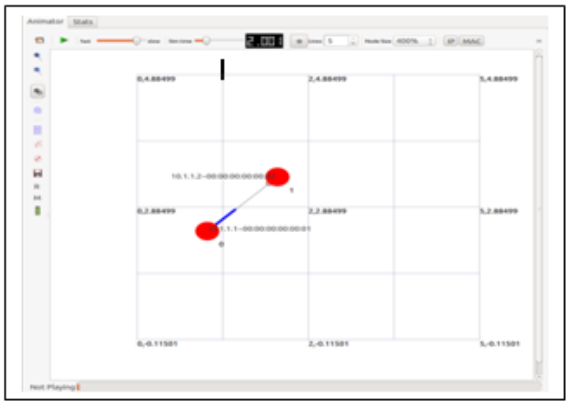

Fig.2- Nodes linking through point to point link in NetAnim

In the script, just add the module [11]:

#include "ns3/netanim-module.h"
and a piece of code which is compulsory[11],
AnimationInterface anim("<filename>.xml");
Other functions (optional to include in code) available for AnimationInterface[11]:
anim.SetMobilityPollInterval(Seconds(1));
anim.SetConstantPosition(Ptr<Node> n,double x, double y);
anim.EnablePacketMetadata(true);
anim.SetConstantPosition(nodes.Get(0),int x, int y);
On the terminal, write $*./NetAnim* and then a window would open and select the desired xml file.

4.4 Output Analysis

A mechanism called as tracing is used to get simulation output. Tracing is a general purpose mechanism which means to get data out of models which should have been preferred for simulation output. This framework provides a set of per-configured trace sources and sinks. Users are allowed to add their own trace sources and sinks. Trace sources can be defined as entities that can signal events that happen in a simulation and provide access to the underlying data. For example, a trace source might indicate when a packet is received by a particular net device (installed on a node) and provide access to the packet contents for interested trace sinks (entities consuming trace information).

Tracing can be enabled by adding following code just before Simulator::Run ()

AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream(filename.tr));

After running the program, trace files would be created at the top level directory of repository by default (because of the way waf tool works). Each line in this file corresponds to a **trace event**. There are two tools available as explained below:

4.4.1 Wireshark

It is a cross-platform, open network analyzer. Originally called as "Ethereal", it was renamed in 2006 due to some trademark issues. It provides graphical interface to view trace files and analyze network results.

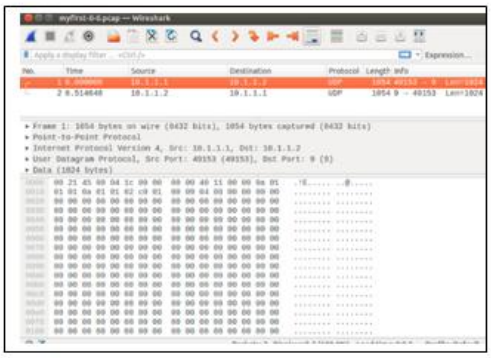

Fig.3- Trace file in Wireshark

It enables to open trace files and display its contents as if you had captured the packets using a packet sniffer. Another method to view trace file contents is through following command which uses tcpdump tool:

$ tcpdump -nn -tt -r nameOfTraceFile.pcap

Where pcap stands for packet capture which is another extension format for trace files. A number of stock topology objects should be predefined. Stock objects include trees, meshes, bus, stars and other topologies of any arbitrary size. NS-3 enables building for both wired as well as wireless network topologies.

For example, to build a bus topology, NS-3 provides a net device and channel which is called as Carrier Sense Multiple Access (CSMA). For construction of any number of nodes in the network, NodeContainer class fulfills the purpose. CsmaHelper class and NetDevice class provides the facility to set attributes of CSMA channel and install net device on each node.

To enable internetworking routing, a concept called as global routing is provided by NS-3. Global routing is a concept that makes the entire internetwork accessible in the simulation and each router behaves as an Open Shortest Path First (OSPF) router. Each node generates link advertisement and communicates them directly to a global route manager which uses this global information to construct routing tables for each node [10]. Setting up this form of routing is a one-liner:

IPV4GLOBALROUTINGHELPER::PopulateRoutingTables();

V. Conclusions

It is hard to judge a simulator in general whether it is good or bad, because each one of them has advantages and disadvantages as well. It all depends on the type of programming language one is interested into, time and memory constraints, complexity level of the simulator, whether one wants to use an open source platform or a commercial tool and many other factors. Though NS-3 is maturing, still it beats its competitor simulators.

NS-3 is an active open-source development model which means one is free to develop new modules, debug them and share results.NS-3 possesses new capabilities such as handling multiple interfaces on nodes [3], use of IP Addressing, support for tracing facility etc. Although it doesn't have all the models that NS-2 posses but still it is way better than its predecessor, as can be concluded from afore mentioned points. The performance of OMNET++ is also inferior as compared to that of NS-3 and moreover OMNET++ is preferably used for modeling distributed and parallel systems.

References

[1] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance". *IEEE/ACM Trans Networking*,vol. 1,pp.397-413, Aug. 1993.

[2] Hashem, "Analysis of random drop for gateway congestion control", *Report LCS TR-465*,Laboratory for Computer Science, MIT, Cambridge, MA ,pp. 103,1989.

[3] R. Chaudhary, S. Sheweta, R. Keshari and S. Goel, "A Study of Comparison of Network Simulator-3 and Network Simulator-2",*International Journal of Computer Science and Information Technologies(IJCSIT)*,vol. 3,pp. 3085-3092,2012.

[4] A.Kumar et al, "Simulators for Wireless Networks: A Comparative Study", in *2012 Int. Conf. on Computing Sciences*,2012, © IEEE. Doi: 10.1109/ICCS.2012.65.

[5] http://www.opnet.com/university_program/itguru_academic_edition/

[6] R. Kokoska, J. Handrikova and J. Valiska,"Software Network Simulators for IPTV Quality of Services",*Acta Electrotechnica et Informatica*, vol. 14,pp. 18-22,2014.
[7] https://www.omnetpp.org/intro
[8] http://wrc-ejust.org/crn/images/Tutorials/ns2vsns3.pdf
[9] J. L. Font, P. Inigo , M. Dominguez , J. L. Sevillano and C. Amaya, "Analysis of source code metrics from ns-2 and ns-3 network simulators", *SIMUL MODEL PRACT TH.*,vol. 19, pp. 1330-1346, 2011.
[10] https://www.nsnam.org/docs/release/3.18/tutorial/ns-3-tutorial.pdf
[11] https://www.nsnam.org/wiki/NetAnim_3.105
[12] E. Weingaertner et al, "A performance comparison of recent network simulators",in *IEEE ICC 2009 proc*,2009,©IEEE.
[13] Sunil Kumar, Jyotsna Sengupta , "AODV and OLSR Routing protocols for Wireless Mesh Network an Adhoc Networks" *IEEE ICCCT* Sept, 2010 at MNNIT Allahabad.