ISSN No. 0976-5697

Volume 7, No. 1, January-February 2016

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Introducing Quagga Test Bed Simulator with QoS Feature

Jaya S. Pujar
Department of Computer Science
Krishnamurthy Institute of Technology and Engineering
Hyderabad, India

Satish Hakkali
Department of Computer Science
Krishnamurthy Institute of Technology and Engineering
Hyderabad, India

Akshata B. A
Department of Computer Science Gogte Institute of Technology,
Belagavi, India

Karuna C.G.
Department of Computer Science and Engineering, K.L.E. Institute of Technology,
Hubballi, India

Abstract: Quagga is a UNIX based network routing protocol that runs as a daemon. It is a fork of popular Zebra framework. Quagga is mainly a text based network suite which relies on text commands for configuration. As the suite is relatively new, it is essential to build a simulation model that can utilize the features of Quagga and provide the administrator with design option that can help administrator understand how Quagga actually works. There is no existing system to best of our knowledge that provides a simulation test bed for Quagga. Therefore in this work we develop a Quagga simulation framework. The objective has been to incorporate two major functions of Quagga: Routing and QoS provisioning. We also provided a framework that helps comparing the Quagga suite with non QoS provisioning flat routing scheme. The proposed simulator incorporates realistic simulation environment by inheriting and extending the classes of Quagga through JNS (java network simulator). Hence the results obtained are at par with real networks. The proposed simulator also provides logging service whereby the result of simulation is available as trace files. These traces can be aggregated and comparative graphs can be designed. Results show that the performance of Quagga is multifold better than non-Quagga based flat routing under overloaded condition.

Keywords: JNS (Java Network Simulator), Latency, Protocol, PDR (Packet Delivery Ratio), Throughput.

I. INTRODUCTION

Quagga is an open source routing stack. The major difference with conventional routing stacks to that of Quagga is it mainly takes care of packet forwarding and with certain accepted degree of Quality of Service. Most of the commercial routing stacks runs on specific routers and need specialized router hardware. Quagga is a routing software package that provides TCP/IP based routing services with routing protocols. Quagga is a routing protocol suite which provides TCP/IP based routing services with routing protocol support such as RIPv1,RIPv2,RIPng,OSPFv2,OSPFv3,IS-IS,BGP-4 and BGP-4+. Quagga also supports special BGP Route Reflector and Route Server behavior. Quagga supports both IPv4 and IPv6 routing protocols. Quagga has a advanced software architecture which provides a high quality, multi server routing engine. Quagga uses an interactive user interface for each routing protocol and also supports common client commands. Due to this design we can easily design or add new protocol daemons to Quagga.

Quagga utilizes UNIX core (Linux to be more specific) to forward the packet. It is particularly well suited for more dynamic network where topology changes frequently due to state of art routing table update services. However adaptation of any protocol or networking suite needs proof of concept. A system administrator must know the probable performance being offered by software before the deployment. Most of the protocols and protocol suits come with their simulator that helps the network designer to test the network on a virtual environment for it's robustness, resilience to overloading and congestion. Quagga offers no simulators to test it's features. Therefore it is quite difficult to quantitatively analyze it's performance for a network with varying nodes and load.

Several past studies have addressed the core issues in Quagga and tried to provide viable extension to the stack itself. However no past works have focused towards the simple basic requirement of extending Quagga's features through a simulator.

To overcome this, our work addresses the mentioned problem and proposes a comprehensive Quagga based simulator where features of Quagga can be tested before deployment. Problem statement can be summarized as to design and develop a Quagga simulator and justify Quagga over normal TCP/IP stack driven routing through simulation results.

Our objective is to build a Quagga simulator that can extend the features of Quagga into a virtual environment. By extending real time routing services over a virtual environment, thorough performance analysis can be made available for the network designer. The objective here is to import the functionality over an existing network simulation model and then extending the simulation model with visualization service to utilize the services of the framework. JNS being a well adopted text simulator that extends NS2 classes is good candidate for base Quagga simulation model. The objective here was to build a Graphical Quagga simulator that provides Quagga specific simulation visualization, the core animation service from network animation trace needed to be bypassed and had to be incorporated into proposed design.

Thus the objective can be summarized as to build a visual simulator based on graphical user interface that can simulate

Quagga features with utmost efficiency and help network designer to design and test virtual network running Quagga services.

The rest of the paper is organized as follows. First in Section 2, we discuss related work. In Section 3, we describe Methodology and User defined Algorithm of work carried out. Expected experimental results are discussed in Section 4. In Section 5 Performance analysis of Algorithm is done. Finally in Section 6, we discuss the conclusion and future work.

II. LITERATURE SURVEY

Quagga is routing software suite which implements OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP for unix, linux, Free BSD, Solaris and Net BSD. Quagga is a fork of GNU Zebra which was developed by Kunihiro Ishiguro. All these software share the same architecture for managing the various protocol daemons. Traditional routing software (such as GateD [1]) is made as one process program that provides all of the routing protocol functionalities as a whole.

Quagga consists of core daemon Zebra which acts as a abstraction layer to the underlying Unix kernel and implements Zserv API to Unix or TCP stream to Quagga clients. Quagga has rich development of library to implement different protocols and a client daemon provides effective configuration and administration. The internal implementation of quagga is provided in www.Quagga.net[2].

Software implementations of routers based on standard PC hardware have been recently made available in the "open software" and "free software" world [3]. Here are some open source routers like Quagga: Modular Router: a software architecture based on Linux, and developed at the MIT, well documented, and freely distributed.

Quagga flow is a transparent combination of Quagga routing protocol suite and openflow enabled hardware where moving completely from legacy protocol stack to logically centralized controllers using openflow protocol as the only communication channel with routers as the forwarding engines.

A propose of design of a bug-tolerant router that significantly reduces the likelihood of a software error affecting the network. Internally, bug-tolerant router consists of several virtual routers running in parallel. Each of these virtual-router instances is made different from the others, by modifying their execution environment (e.g., by reordering the routing updates they receive, by changing their configuration, or by modifying their layout in memory) or by modifying their internal structure (e.g., by running router code implemented by different programmers)[6].

In virtual routers as a service the author speaks about RouteFlow which is innovated by providing interfaces to multi-vendor networking hardware which are commercially available and open source software development. RouteFlow, is an architecture following the software-defined networking (SDN) [5] standard based on a programmatic approach topologically centralize the network control, unify state information, and decouple forwarding logic and configuration from the hardware elements.

In the paper “can software router scale?”[7] speaks that instead of sticking to special purpose hardware routers, switch to software routers which provides the environment of general purpose routers.

To provide the better performance to Quagga 9.8 the OSPF (open shortest path first) code is altered . earlier in Quagga 9.8 in OSPF they were using Dijskstra’s algorithm to find the shortest path when there is a topology change in router. The solution for this is provided by replacing the Dijskstra’s algorithm by binary heap data structure. This optimization leads to better performance than conventional OSPF routers[8].

In paper DROP: An open source project towards distributed software router architecture [9]. This paper explores the developing and giving a new distributed model for IP router control and management. DROP is partially based on the main guidelines of the IETF ForCES standard (open-source realization), and it allows building logical network nodes through the aggregation of multiple software routers, which can be used to forward the packets or to control different services provided by the router.

Software routers are gaining attention in the mind of technology developers as they can be developed on the standard PC with less cost [7]. But router based on single PC are suffering from limited bus and Central Processing Unit (CPU), bandwidth, high memory access latency, limited scalability in terms of number of NICs and lack of flexibility mechanisms. Solution for this is to consider Multi-stage architectures created by interconnecting several PC’s which results in i) increase the performance of single software routers, ii) scale router size, iii) distribute packet-forwarding and control functionalities, iv) recover from single-component failures, and v) incrementally upgrade router performance[10].

In the paper “Control and Management Plane in a Multi-stage Software Router Architecture” describes a control protocol for multistage architecture based on PC interconnection. The protocol allows information exchange among internal PC’s which supports configuration of the interconnected architecture, packet forwarding routing table distribution.

III. METHODOLOGY

The proposed system is depicted in Figure 1. The procedure of proposed system is a follows:

```
Source -> Source : Transmit Environment
alt Router Selection
Source -> Router : Select
else failed
Source  -> Source: Wait
end
Source -> Router : Transmit Data
Router -> Router : Aggregate
alt RouteAvailableToDestination
Router -> Destination: BufferedData
else FindRoute
Router -> Destination: Bandwidth Efficient Route ( DSDV)
end
Router -> Destination : Transmit Buffered Data Through Ro
alt EnoughBandwidth
Source -> Router : Message
Router -> Destination: Message
else Wait Schedule
Router -> Source: Decrease Packet Rate
Source -> Router : Packet Rate Drop

End
```

Fig. 1 shows the basic architecture of the network. Firstly group of nodes, distributed over a geographic area select a Router pertaining to that area. Once selected, this node acts as local data gathering and buffering node. It schedules the entire

domain member's reception and is responsible for channel allocation. It buffers the packets and forwards the packets to the sink node using IP routing. If any node is found to flood the network with too many packets, then the QoS framework of detects the node and bypasses the node for the current transmission cycle.

The overall framework first compiles the Quagga library which is present as a C++ source. Basically Quagga framework is a daemon that runs Zebra, a unix based open source routing source. Using Java's native class support, the C++ source files are imported into java simulation using java network simulator package which in itself is an extension of NS2.

ALGORITHM

```
Step. 1     Make Network with N nodes over grid Topology
            of WIDTH x HEIGHT
Step. 2     Decide Number of Regions(domains).
Step. 3     Make the node with highest degree of neighborhood
            as Router
Step. 4     Other Nodes Join Router
Step. 5     Transmission
For( node=1:N)
        Transmit data
      If(!node(i) is Router?)
                        Transmit to Router
Node(i) ->BW=Node(i)-> BW distance  between node(i) , Router
Else // Router
        Gather data from other nodes.
        Bufferize Data.
        Transmit data to Destination
Node(i) ->BW=Node(i)-> BW distance  between node(i) , Router
if(Node(i)->BW<Threshold)
        [OFN]=DetectOverfloodingNode();
For each ofn in OFN
        WithdrawLink(ofn)
End
End
End
Step. 6     At the end of simulation, calculate pdr,
        packet par node and bandwidth
```

This java package provides Quagga framework as four major components : IP addressing, IP protocol (DSDV and OSPF), QoS provisioning and packet structure. The simulator inherits Quagga's node structure into graphical nodes, thereby providing the graphics abilities to these nodes alongside the conventional networking being inherited from Quagga through JNS.

Nodes forwards packets which are IP packets encapsulating TCP frames. These packets are routed through the router which maintains a link to all the domain members as well as the routers of other domain. Therefore packets are routed using routing module extension through routers to the sink node. Sink is assumed to be gateway node that connects entire network with internet. Therefore the proposed simulation provides a realistic simulation of local network running Quagga where objective is to connect every node to the internet through their routers and the gateway node.

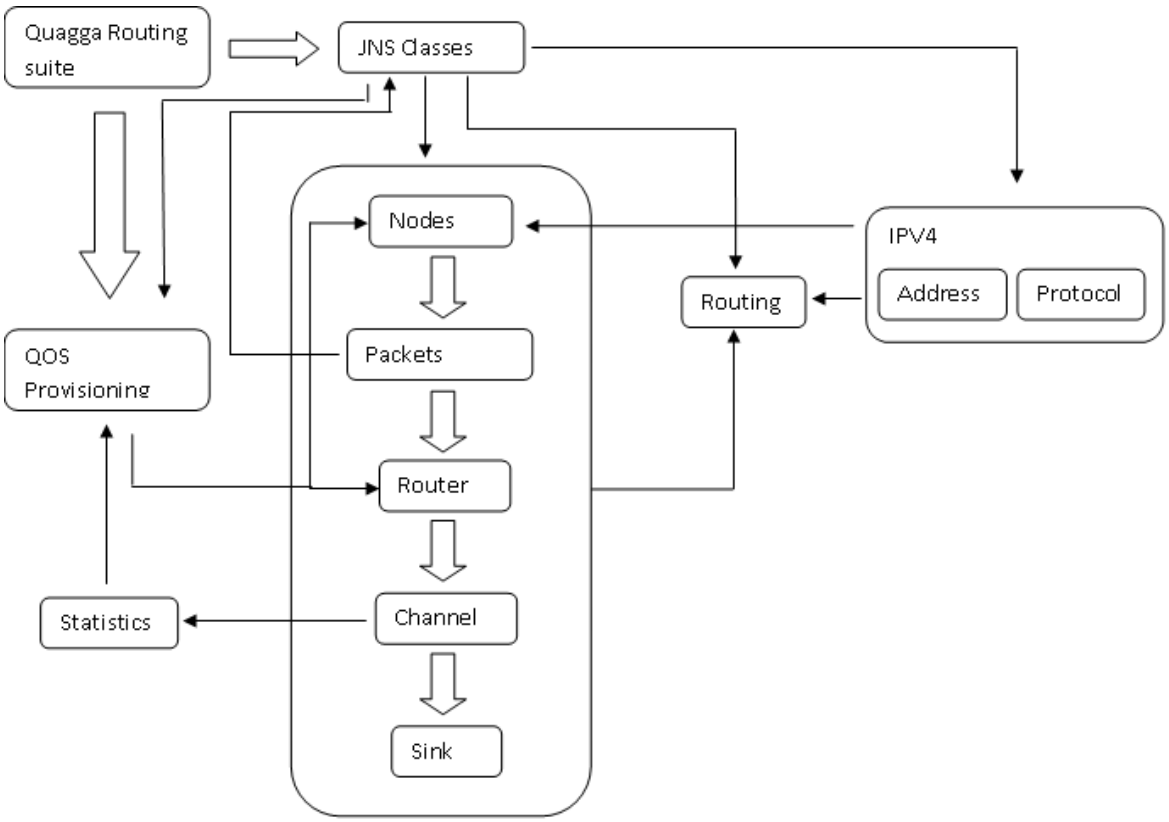

Figure 1. Basic Architecture of proposed work

IV. EXPECTED RESULTS

The expected output is shown in the following figures. First - create a dynamic network with N nodes which needs to be given as input. Fig.2. depicts the same.

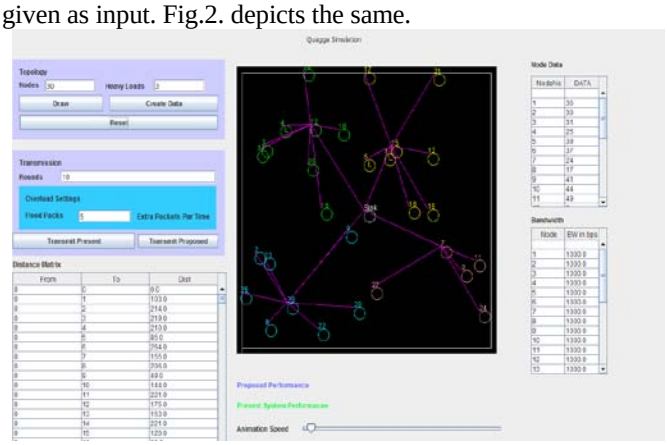

Figure 2: Dynamic network with 30 nodes specified by user

After giving N nodes as input create data for the nodes to transmit at maximum bandwidth capacity of 1000 bps (Fig.2. Left side). Create button Connectivity shows initially about the picture of nodes communicating with each other.

Generate random number of packets from every node to be sent to sink. We can observe three heavily loaded nodes in the snap shot because we have mentioned in the heavy load text box as 3 (Fig.2) which consumes more bandwidth.

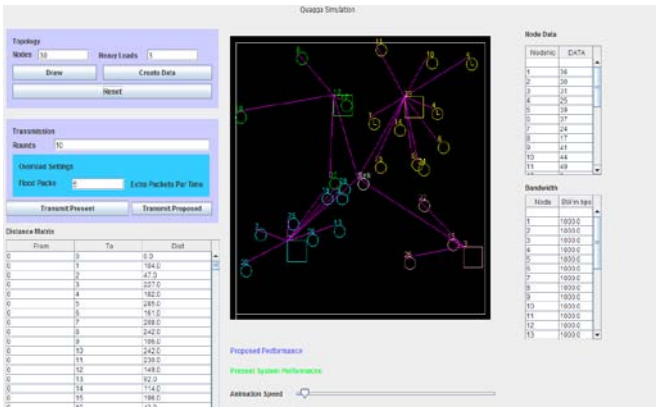

Figure 3: Entire network is divided into four domains.

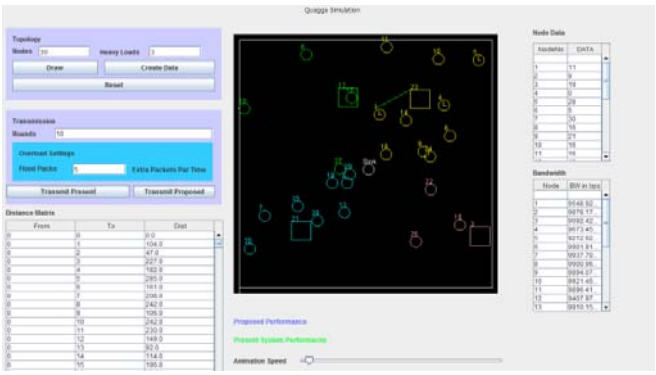

Figure 4: Clicking the transmit present button the packet starts to transmit from nodes to destination ie.,to internet sink

In Fig.3 Nodes in the domain/clusters are displayed with same color. Place the routers in a way such that most of the nodes are

reachable through router. This can be done by clicking Reset button.

At this stage we can see network configuration changed such that:

a) All routers are connected
b) Routers are connected to sink
c) Nodes in a domain is connected to router
d) No two nodes can communicate without router.

Fig 5. shows the performance of the present system with estimation. After clicking the proposed system button we can observe LB nodes which are black listed nodes means these node consumes more bandwidth than other nodes. These nodes are blocked for a session and the performance is calculated and displayed are black listed nodes means these node consumes more bandwidth than other nodes. These nodes are blocked for a session and the performance is calculated and displayed. We can compare both present system and proposed systems performance ratio.

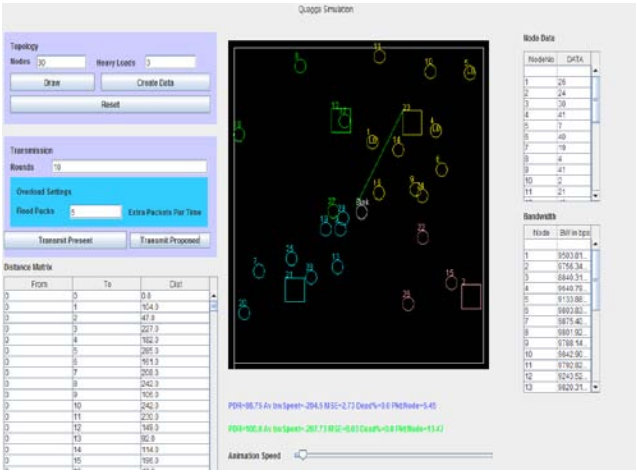

Figure 5: Performance of present system

V. PERFORMANCE ANALYSIS

In this section we are discussing regarding the performance of the present and proposed systems. We mean by present system, the Quagga routing protocol stack is just a control plane, whereas in proposed system control plane, forwarding plane and quality of services are embedded. We are considering the attributes like number of Nodes, Packet delivery ratio, Mean square error, dead packet and average bandwidth. For both present and proposed systems the resulting values are compared and graph is plotted. After the analysis for all the features considered the proposed system is showing better result. The following subsections present the analysis.

Following performance metrics are observed for each simulation:

Packet Delivery Ratio(PDR) = Ratio of packets received to packets sent (in %)

Throughput (TP) = Average transform rate or Bandwidth of the route

= (packets received / simulation time) * packet size (in Mbps)

Nodes = number of Mobiles including Router nodes

Simulation Time = total simulation time of network (in seconds)

App rate = Application rate (number of packets per second)

Latency = Time spent to deliver each data packet (in seconds)

Bandwidth: Average Bandwidth remaining is calculated by taking the average of remaining bandwidth of all the nodes in the network. Node looses bandwidth in transmission and reception.

Packet Delivery Ratio (PDR) : PDR is calculated as the summation of number of packets received by all destinations is devided by summation of number of packets sent by all the sources.

$$\text{i.e. PDR} = \frac{\sum \text{number of packets received by all destinations}}{\sum \text{number of packets sent by all sources}}$$

Table 1 Data of Present and Proposed MSE

Mean Square Error		
No. nodes	**Present**	**Proposed**
10	61.92	15.89
15	17.1	12.83
20	21.38	10.83
25	1.26	3.33
30	3.57	1.59
35	4.21	2.27
40	33.97	1.98
45	0.22	0.62
50	0.5	1.26
100	0.08	0.13

The obtained PDR values from simulation are plotted across number of nodes (Fig 6 & 7). X and Y-axis represents number of nodes and PDR ratio respectively. By observing the graphs PDR is better in the proposed system.

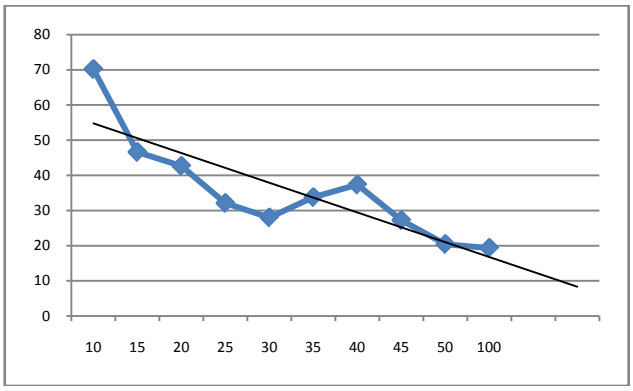

Figure 6: Present PDR

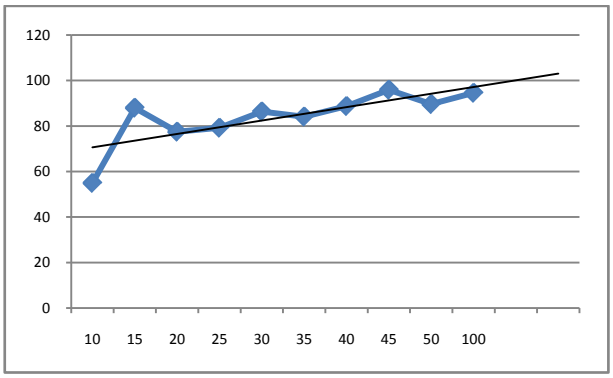

Figure 7: Proposed PDR

Mean Square Error (MSE)

MSE is a comparison of aggregated sink data with total network data. Actual data from all nodes is calculated and is expected that it should reach the sink. But the actual data that reaches sink is not the expected value. That is being compared in this section. The table (Table.1) shows comparison of data of present and proposed system and graph is plotted (fig 8 & 9.) X and Y-axis represents number of nodes and MSE respectively. We can conclude that the MSE of proposed system is reduced.

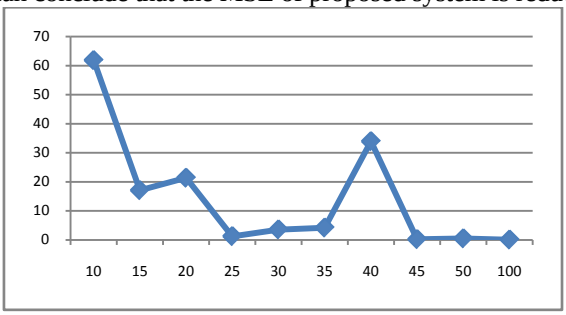

Figure 8: Present MSE

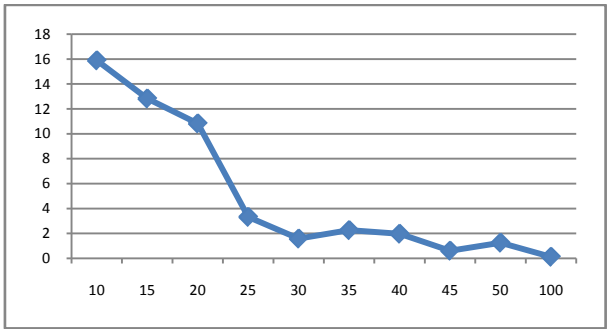

Figure 9: Proposed MSE

Packets Per Node (Pkt/Node) and Average Bandwidth utilization

In the network each node is sending/receiving the data packets. Packets per node are calculated based on the number of packets received at the sink to the number of packets generated by all the nodes. Present and proposed system and graph is plotted (Fig 10. and 11) and x and y-axis represents number of nodes and Pkt/Node respectively. It is clearly seen that in the proposed system bandwidth utilization is efficient and hence the packets per node is also high. But in the present system the router aggregates the data and the router losess the bandwidth for transmission. The bandwidth utilization is shown in the graph (Fig 12).

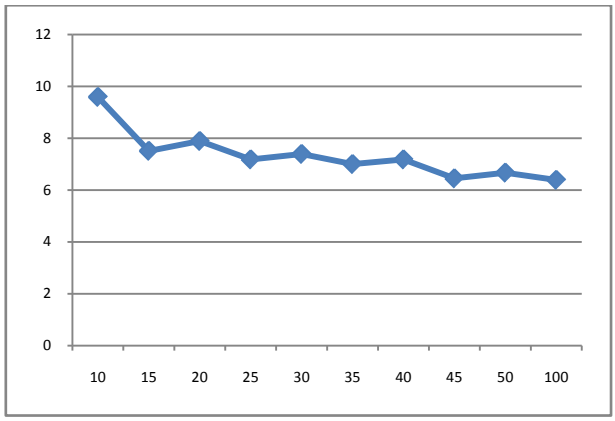

Figure 10: Present

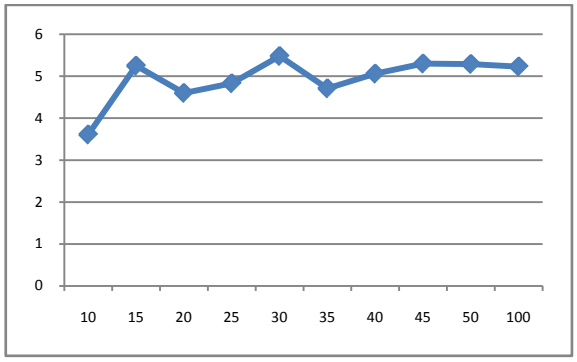

Figure 11: Proposed Pkt/Node

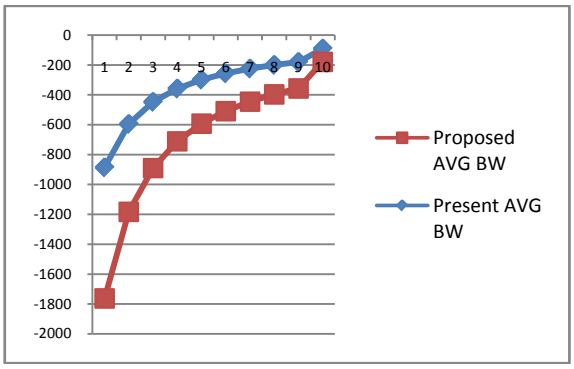

Figure 12: Average Bandwidth of Present & Proposed

VI. Conclusion And Future Work

Designing a Quagga simulator helps the network administrator to analyze and debug the problem if persist. The major difference with conventional routing stacks to that of Quagga is it mainly takes care of packet forwarding and with certain accepted degree of Quality of Service. Quagga utilizes Unix core (Linux to be more specific) to forward the packet and is particularly well suited for more dynamic network where topology changes frequently due to state of art routing table update services. The advantage of Quagga is that it extends the conventional TCP/IP stack and therefore nodes can be configured in Quagganet seamlessly.

Most of the protocols and protocol suits come with their simulator that helps the network designer to test the network on a virtual environment for it's robustness, resilience to overloading and congestion. Quagga offers no simulators to test it's features. Therefore it is quite difficult to quantitatively analyze it's performance for a network with varying nodes and load. Its hereby concluded that this system proposes a comprehensive Quagga based simulator where features of Quagga can be tested before deployment.

VII. References

[1] A Study of the interaction of BGP/OSPF in Zebra/ZebOS/Quagga Avinash Ramanath.

[2] 10. Quagga Routing Software Suite, http://www.quagga.net.

[3] Application of Free and Open source software and its Impact on society, Ambar Kundu et al. / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 1 (4) , 2010, 226-229.

[4] QuagFlow: Partnering Quagga with OpenFlow, Marcelo Ribeiro Nascimento,∗ Christian Esteve Rothenberg,∗† Marcos Rogerio Salvador∗ and Maurício Ferreira Magalhães.
[5] Building Bug-Tolerant Routers with Virtualization Matthew Caesar and Jennifer Rexford Princeton University.
[6] When Open Source Meets Network Control Planes et al. / Christian Esteve Rothenberg(*).
[7] Can Software Routers Scale? et al. Katerina Argyraki.
[8] Routing Reconfiguration in IP Networks by Paolo Narviez, June 2000.
[9] DROP: An Open-Source Project towards Distributed SW Router Architectures, et al/ Raffaele Bolla
[10] High Performance Switching and Routing, 2008. HSPR 2008. International Conference on,et al. Bianco, A. 15-17 May 2008,IEEE