PERFORMANCE OPTIMIZATION OF INTRA DOMAIN ROUTING PROTOCOLS USING QUAGGA

Ashish Kuamr Mishra, P. Selvaraj

Abstract—Optimization and high performance of routing protocols are needed in this interconnected world. As the intra-domain routing protocols (OSPF) is widely used in ip network. In ospf implementations the processing delays impact the time needed to re-convergence after a topology change for both intra-domain and inter-domain routing. The performance index used to characterize the re-convergence capability is also referring as switching time. To measure the switching time on pc based router and open source routing software we built a test-bed. Moreover we describe a set of changes made on Quagga code in order to optimize some processes, whose algorithms were not efficient. After obtained result we show that, if the routing software is optimized, the pc-based routers perform better than commercial router in terms of switching time. The realized implementation allows the shortest path computation time to be reduced of about the 97%.

Index Terms— network, Ospf, Pc-based router, quagga, switching time.

1. INTRODUCTION

THE diffusion of open software implementing routing protocols, together with the big computing power of normal PCs, have been raising a big interest towards the possibility of developing a complete routing system based on open source software and standard low-cost hardware [1]. Software routers are achieving a great interest in the last years because they represent an even more realistic alternative to commercial routers. In my work I have always used Linux operating system because it is becoming an interesting competitor for Windows, with a number of users constantly growing. Software implementations of routers based on standard PC hardware have been recently made available in the "open software" and "free software" world. Quagga and Xorp represent the most common open-source routing software and in my work I have studied and tested quagga.

The interest in the Software Router employment has an economic motivation and a research motivation. In fact PCs hardware is available at low cost, their architectures are well documented and their performance evolution is guaranteed. Another important aspect of Software Routers is that software is free and documented while in the case of commercial devices software is not available. Of course it is important to evaluate Software Router performance; in this way such a device has to be conformed with protocols it implements, it has to communicate with different Software Routers and with commercial devices, and it should have performance at least comparable with ones of a commercial device. So the performance evaluation of Software Routers represents the first activity of my work.

To examine the performance of a router running the OSPF protocol [2] we built up a set of tests. A Black Box measurement [4] of switching time [5, 6] is conducted in a PC running Linux and Open Source [18] routing software. Switching time is defined as the time needed by a router to re-converge its routing tables and redirect data traffic to the best route. Switching time depends on the Shortest Path First (SPF) computation time, that is, the time needed to execute Dijskstra's algorithm when a topology change is received by the router. Our measurement results compare switching time in a PC router and in a Cisco commercial router [9].

From our evaluation of switching time we uncover inefficiency in the implementation of Dijsktra's algorithm in Quagga [17] routing software. Our analysis of the code found that the data structures used to implement the Candidate List [7] during the SPF calculation were suboptimal. We modified the code to use a binary heap data structure [8] and have evaluated the switching time of the new version of Quagga.

The organization of the paper is as follows. The test-bed for the SPF time evaluation is illustrated in Section II. The OSPF performance of Quagga is shown and compared to the one of the Cisco router in Section III. The optimization of Quagga is described in Section IV. Our main conclusions are discussed in Section V.

a. Ospf Performance Indexes

The performance characterization of an OSPF router is based on the detection of its main functions and on the definition of procedures to measure them using OSPF control packets. This work has been completed by Benchmarking Methodology Working Group (BMWG) of IETF which has defined OSPF performance indexes and methodologies for tests execution, described in [10,11]. The

three performance indexes defined by BMWG, each one related to a specific function, are reported, with a brief description, in the following list:

• LSA processing time.

When an OSPF router receives an LSA it has to control integrity, age, if LSA is a new one or a duplicated one (already present in its LSA Database), eventually it has to insert the new LSA in its Database and sends an Acknowledge LSA. Of course LSA processing time is influenced by LSA type (new or duplicate), LSA links number and Database extension.

• LSA flooding time.

When an OSPF router receives a new LSA it has to perform flooding and so it has to transmit the new LSA through all its interfaces, except the one it has received the LSA from. Also in this case LSA type, links number and Database extension influence this performance index.

• Shortest Path First (SPF) computation time.

The execution of Dijkstra algorithm to compute all shortest paths is the most onerous operation for an OSPF router. The time needed to perform this function depends on network topology complexity and so, as will be better described in next Section, on LSA Database extension. Besides these three performance indexes proposed by BMWG I have also defined a further one, which considers the interaction between control plane (OSPF) and data plane:

• Switching time.

It represents the time for an OSPF router to update its routing table after a topological event and so to switch traffic from an old path to a new one. In following Sections I will analyze in depth the last two indexes because they give a full characterization of impact of OSPF router functions on network performance and because results obtained represent the starting point for Software Router modification. In fact, as a consequence of SPF computation time results, I'll show how Software Routers need to be optimized to be compared with commercial ones.

2. TEST-BED FOR THE EVALUATION OF THE SWITCHING TIME

In this section I'll first describe test methodology to measure SPF computation time and then I'll analyze results obtained on Quagga, and Cisco routers. The aim of this test is to determine how mch time it takes for a Device Under Test (DUT) to finish the SPF Computation. The test configuration used is illustrated in Fig. 1. When a best route to a destination is found it determines switching time, that is the time for an OSPF router to re-converge the routing table and redirect data traffic.

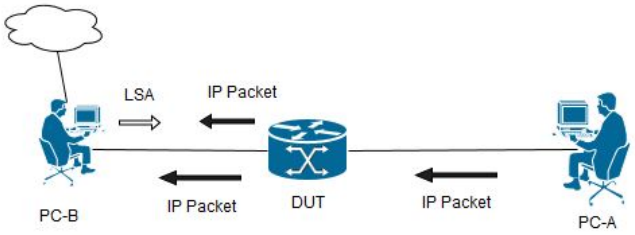

Fig.1. Test-bed for SPF computation. To send data packets PC-A, using RUDE traffic generator. PC-B emulates a network topology and decides which path will be followed by the data packets.

PC-A function is to generate data traffic that DUT will switch using the best path. RUDE traffic [15] is used to generate data traffic. PC-B emulates the topology reported in Fig. 2.This topology is composed of a variable number of simulated routers and networks. The DUT has to find the shortest paths to every network and router. The software BRITE [12, 16] is used to generate the emulated network topology. PC-B uses the LSA generator software SPOOF [14] to send to the DUT appropriate LSAs describing the emulated network topology.

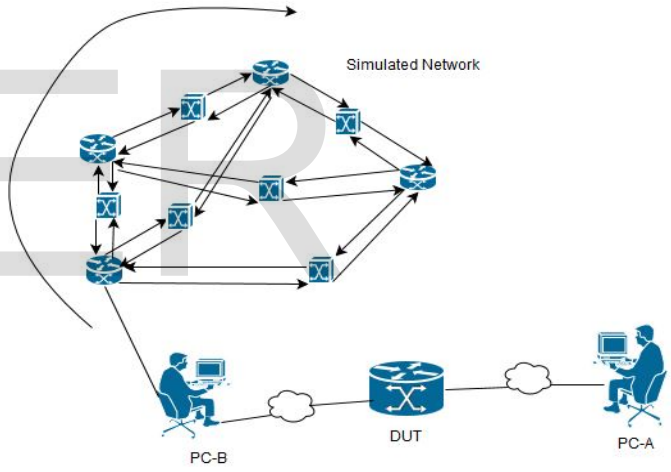

Fig. 2 A network topology is emulated in PC-B by sending appropriated LSAs to the DUT.

The measure methodology for the SPF computation [10] has defined by the IETF. So, when an Update LSA arrives, with a fixed delay the SPF computation start time is scheduled, a timer is set and after the timer expires the SPF calculation starts. OSPF routers use to schedule the instant (timer) in which the SPF computation starts to avoid to perform the calculation too many times when receiving Update LSAs [2]. Moreover, another timer enforces a lag between two consecutive SPF computations. In particular the following two timers are defined in [2]:

• spf_delay: time between receiving an Update LSA and starting the SPF computation;

• spf_hold_time: time between two consecutive SPF computations.

The test can be performed varying the position of the destination network. In particular we consider the case of highest switching time and this happens when destination network is the last inserted in DUT Routing Table. In this case the components of the switching time, reported in Fig. 3, are:

- RTT, the Round Trip Time between the PC-B and the DUT;
- Tproc_LSA, the update LSA processing time;
- TSPF , the Shortest Path First computation time, the time needed to perform Dijkstra algorithm finding shortest paths from the DUT to all destinations;
- Tupd_RT, the time needed to update the routing table of the DUT

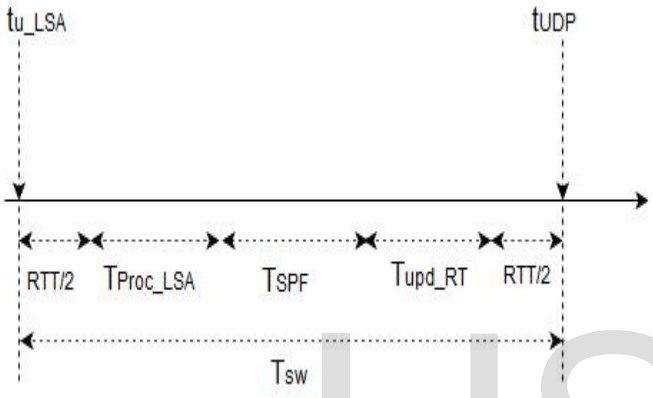

Fig.3. Components of the switching time Tsw.

The switching time can be calculated as the time difference between the sending of the update LSA and the receiving of the first UDP packet on the new path including networks N2 and N 4 and router B2:

$T_{sw}=t_{UDP} - t_{u\_LSA}$ (1.1)

and can be expressed as follows:

$T_{sw}=RTT + T_{proc\_LSA} + T_{SPF} + T_{upd\_RT}$ (1.2)

Let us analyze how each component influences the switching time. RTT can be ignored because PC-B and DUT are connected through a Fast Ethernet directed link so its value is smaller than 1 µs. Tproc_LSA and TSPF can be measured according to the test methodologies described in [13]. From results reported in [10, 11] it is possible to verify that Tproc_LSA is much smaller than both TSPF and Tupd_RT. In conclusion switching time mainly depends on two operation: SPF computation time TSPF and routing table updating time Tupd_RT.

3. OSPF PERFORMANCE ON QUAGGA

All performed tests are based on fully meshed network topologies, with each router connected to each other through a different transit network. Fig. 3 shows an example of fully meshed topology. The obtained results are compared to the ones taken on the Cisco.

The input parameters based on we are doing experiment:

i) Fp, the constant rate at which the packets are transmitted by PC-A;
ii) Lp, the length of the packets sent from PC -A;
iii) N and M, the number of vertexes and edges of the directed graph representing the emulated network topology respectively.

It is important to remark that in representing the emulated network as a directed weighted graph [3], each router and each transit network becomes a vertex of the graph, and each network-router link becomes an edge. Each edge is labeled with a cost representing the interface cost of the link connecting a router to a network [3]. One of the components of switching time is the SPF computation time whose evaluation depends on both N (vertex) and M (edge). In particular, the SPF computation starts when the DUT receives the update LSA.

If we consider an emulated network topology composed by R routers, we have that:

$N=R(R-1)/2 + R$ (2.1)

$M =2R(R -1) +2$ (2.2)

In Fig. 4 we have Experimental values taken on a PC based router and on a Cisco router. We report switching time as a function of N for packet length Lp=100 bytes and rate Fp=500 pps. By a more accurate analysis we argued that the high switching time in Quagga is due to the high time needed to update the routing table once the Dijkstra's algorithm has been executed and the best paths have been evaluated. This can be revealed by Fig. 5 where we report the SPF computation time as a function of N for Cisco, Quagga.

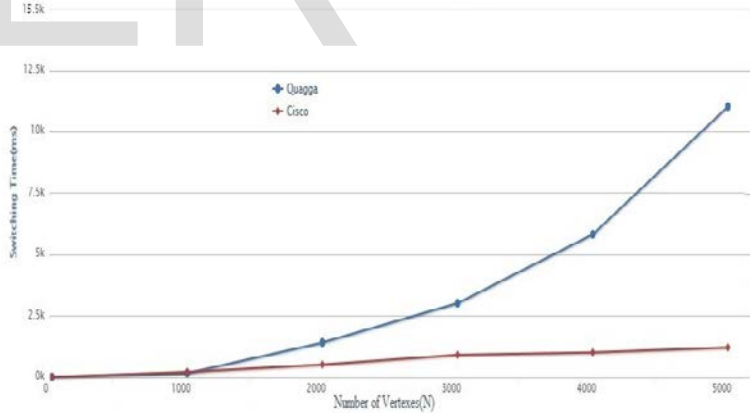

Fig.4. Switching time as a function of the number of vertexes (N) in the emulated network topology

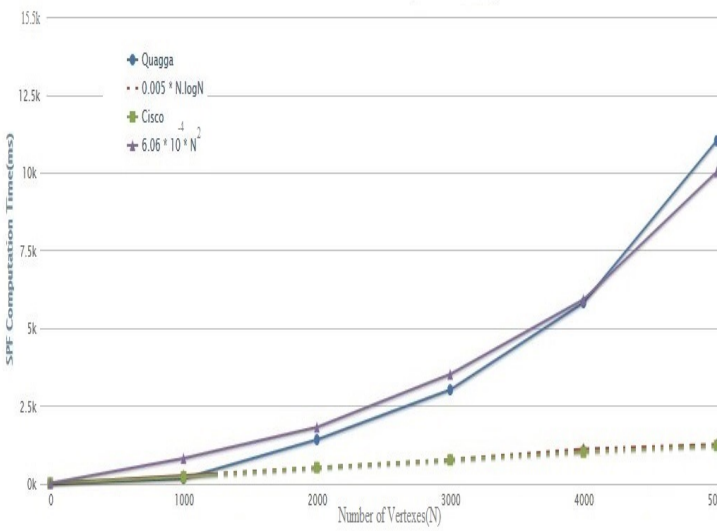

Fig.5. SPF computation time as a function of the number of vertexes (N) in the emulated network topology (Fp=500 pps and Lp=100 bytes)

Comparing the two curves, we conclude that Quagga performs better than the Cisco only when the number of routers R in the emulated topology is smaller than 45 corresponding to 1035 vertexes. For large network cisco is better. In fact it is possible to notice from Fig. 5 that the experimental measurements obtained for the Cisco router closely fit the curves 0.005×NlogN. This result is expected from the complexity of Dijkstra's algorithm [7,8]. On the basis of these considerations we have optimized Dijkstra's algorithm to obtain performances comparable to commercial routers. Section IV therefore will be dedicated to the analysis of the Dijkstra's algorithm complexity and to its optimization in Quagga.

4. Optimize SPF Computation Time In Quagga

The SPF computation is based on the Dijkstra's algorithm [2]: the algorithm examines the directed weighted graph, in order to find the shortest paths from a root vertex to each vertex in the graph. In Quagga, the directed graph is represented by the LSA set, stored in the LSA database. The Dijkstra's algorithm finishes when all of the vertexes have been inserted in the Spanning Tree and that occurs when the Candidate List becomes empty. During this procedure the Candidate List is the most stressed structure, and its implementation is critical for the resulting global performances. The Candidate List performs four different functions:

Extract-Min: This extracts the node with the minimum key from the Candidate List;

Insert: This inserts a node into the Candidate List.

Decrease-Key: This updates the total cost associated with a particular node;

Lookup: which find a node whether it is stored in the Candidate List;

In Quagga, the Candidate List is implemented with a linked list, whose elements are stored in key increasing order. In this case the Dijkstra's algorithm complexity is $O(N^2+NM)$. If we want to obtain an $O((N+M) \log N)$ total cost, we need to reduce the cost of the Insert, Decrease-Key and Lookup functions down to $O(\log N)$. This result can be achieved only changing the data structure adopted to implement the Candidate List. In Section 4.1, a binary heap data structure Implementing the Candidate List will be proposed and its complexity will be evaluated. One difficulty with this proposal is that binary heaps do not provide efficient Lookup function. In Section 4.2 we will illustrate how this operation can be eliminated by modifying the LSA database data structure. In Section 4.3, the modified Quagga routing software will be evaluated.

a. A binary heap data structure to implement the Candidate List

In particular in the new Quagga version we have chosen a binary heap to implement the Candidate List. A binary heap is a complete and balanced binary tree with a local sorting [7, 8]. Leaves are always inserted starting from the left. Thus the heap depth is always less than logN, where N is the number of nodes. Each node of the heap has a key, and the whole heap is locally ordered on these keys, so that each node has a key lower than the ones of its two children. This particular sorting ensures that the node with the minimum key is the root of the heap. The management of the tree is based on two internal functions: the sift-up and the sift -down functions. The sift-up function illustrated in Fig.6 . The sift -down function illustrated in Fig.7. These two functions perform the three main functions supported by the heap structure: Insert, Extract-Min and Decrease-Key.

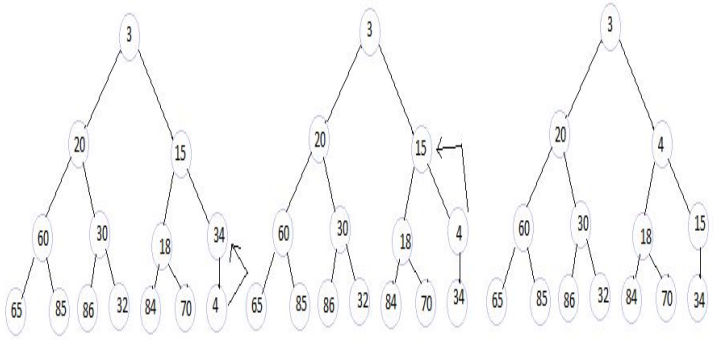

Fig.6. An example illustrating Insert function for a node with key 4.

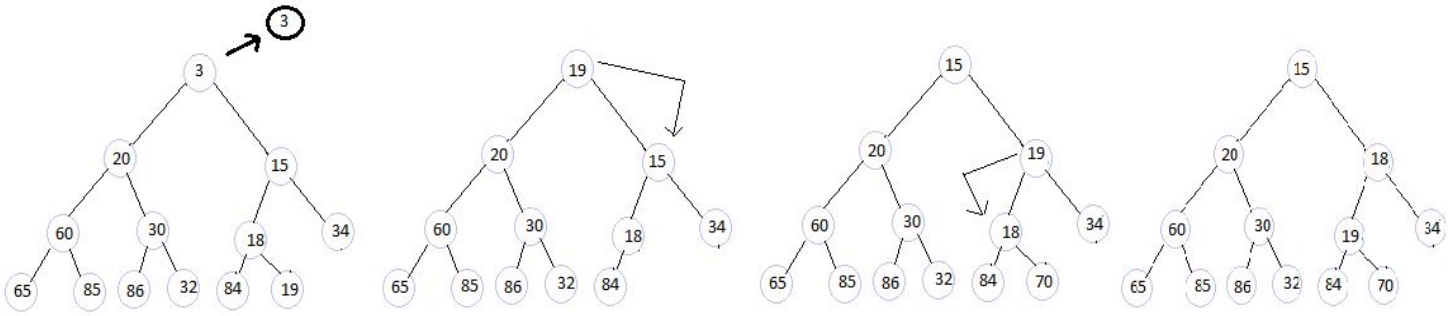

Fig.7. An example illustrating Extract-Min function for the node with lowest key 3.

The Decrease-Key function changes the key of a particular node to a lower value. Once the key value has been decreased, it executes the sift-up procedure on the node, and takes it to its new position. An example is shown in Fig. 10 where the key of a node is decreased from 18 to 5.

Notice that to implement the Decrease-Key function, a Number of sift-up operations at most equal to the maximum depth of the heap is required. For this reason the Deacrese-Key function cost is O(logN).

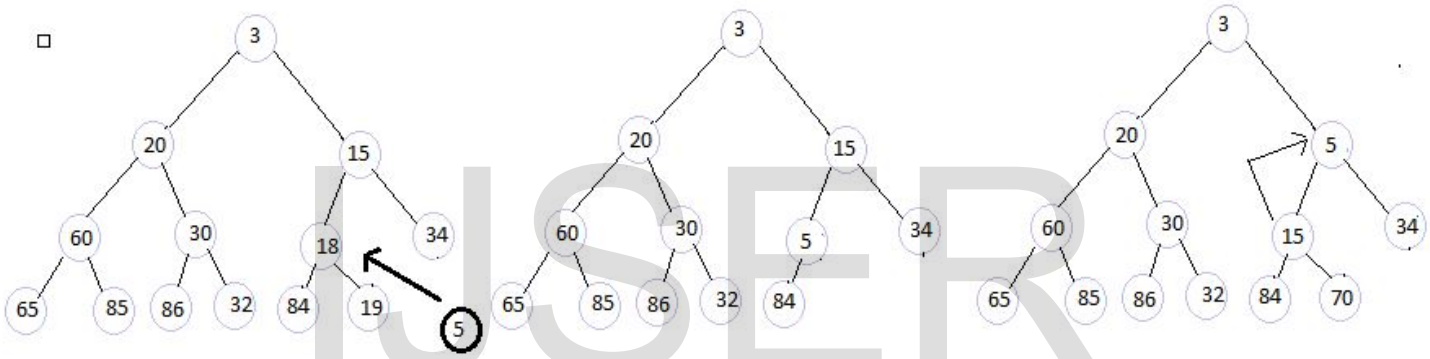

Fig.9 . An example illustrating Decrease-Key function for the node decreasing its key from 18 to 5.

So the amortized costs are O(N*logN) for the Insert function, O(N*logN) for the Extract-Min function and O(MlogN) for the Decrease-Key function. Finally, because in Section 5.2 we will show that by modifying the
LSA database data structure the Lookup function is no more needed, the total cost of the new implementation of the Dijkstra's algorithm in modified Quagga becomes as expected O((M+N)logN). In particular when M=O(N) the amortized cost reduces to O(N*logN).

4.3 New solution for lookup operation

Instead of finding a way to implement the Lookup function, with an O(logN) cost, we have modified the LSA database data structure so that the Lookup function becomes no longer needed. In particular for each LSA, stored
in the database, we have added information denoting whether the LSA is in the Candidate List. In the positive case, the information also denotes the position in the Candidate List where the vertex associated to the LSA is stored. That allows a vertex associated to an LSA to be immediately accessed during the execution of the Dijkstra's algorithm. Further, because the sift-up and the shift-down operations may change the position of a vertex in the Candidate List, a pointer to the information of the associated LSA is added to each vertex, so the LSA can be updated simultaneously.

4.4 Numerical Results for Modified Quagga Routing Software

The measured values of the switching time on the modified Quagga version is presented in Fig. 10 varying the number N of vertexes in the graph. They are compared with the same measure taken on the Cisco and on the original Quagga version.

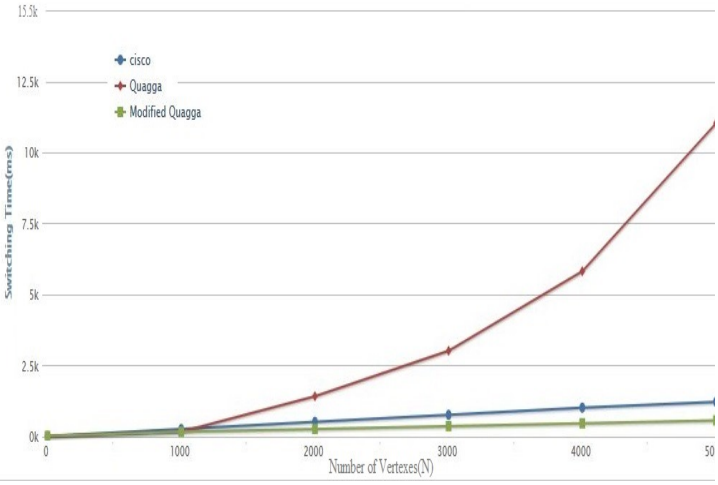

Fig. 10. Performance comparison among Cisco, Quagga , modified Quagga (Fp=500 pps and Lp=100 bytes)

From the results shown in Fig. 10 we notice that switching time on the modified Quagga version is always less than the time needed on the original version, proving the success of our optimization process.

EXPERIMENT SETUP

We have used a system having Ubuntu 14.04 operating system with minimum 2 GB RAM, 350 GB Hard disk, 3.10 GHz processor. To create Test-Bed (explained in fig.1) we used VMware Workstation. Create 3 Virtual Machine having Ubuntu 14.04 on VMware Workstation. One virtual machine work as PC-A using RUDE traffic generator for generating packet. Another virtual machine work as PC-B using BRITE software for network topology. And third virtual machine work as DUT. Install quagga on DUT. First check performance using the source code available on quagga the experiment result is showing in fig. 5.

5. CONCLUSIONS

This experiment is setup to evaluate the OSPF performance of software router. In particular I have reported the performance indexes to characterize an OSPF router. So I have directed my attention to two performance indexes: SPF computation time, which represents the time for a router to compute Dijkstra algorithm, and Switching time. So to have a performance comparison between Software Routers and a commercial router I have realized a test-bed to evaluate these indexes on both devices. After a results analysis I have concluded that the Software Routers (quagga), have performance worse than a Cisco device. After our changes to Quagga code, improved performance. The results obtained for switching time with our modified version of Quagga are better than the ones obtained with a high-end commercial router. In this way I have proved that a Software Router is a realistic competitor for a commercial device.

REFERENCES

[1] Raffaele Bolla, Roberto Bruschi "PC-based Software Routers: High Performance and Application Service Support" of the ACM workshop on Programmable routers ,2008, University of Genoa, Italy
[2] J Moy, "OSPF Version 2" , Request for Comments 2328, April 1998
[3] Afek Y., Bremler-Barr A., Har -Peled S., "Routing with a clue," IEEE Trans. On Networking, Vol.9, n.6, 2001, pp.693 -705
[4] A Shaikh and A Greenberg, "Experience in Black-box OSPF Measurements," in Proc ACM SIGCOMM Internet Measurement Workshop (IMW) 2001, pp. 113-125, November 2001
[5] V. Eramo, M. Listanti and A. Cianfrani, " Switching Time Measurement and Optimisation Issues in GNU Quagga Routing Software," IEEE Globecom 2005, St. Louis (USA), December 2005
[6] S.Bhosale ,R. Joshi "Conformance testing of OSPF protocol," IEEE , Beijing, Jan. 2008
[7] S. Saunders, "A Comparison of Data Structures for Dijkstra's Single Source Shortest Path Algorithm,"
citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.6508.
[8] A. V. Goldberg and R. E. Tarjan, " Expected performance of Dijkstra's Shortest Path algorithm," Technical Report 96 -062, NEC Research Institute, Princeton, NJ, June 1996
[9] www.cisco.com
[10] V. Manral, R. White, A. Shaikh. Benchmarking Basic OSPF Single Router Control Plane Convergence, Request for Comments 4061, April 2005.
[11] V. Manral, R. White, A. Shaikh. Consideration When Using Basic OSPF Convergence Benchmarks, Request for Comments 4063, April 2005.
[12] A. Medina, A. Lakhina, I. Matt a, and J. Byers, "BRITE: An Approach to Universal Topology Generation," in Proc. Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01), IEEE Computer Society, Cincinnati, August 15 -18 2001
[13] V. Eramo, M. Listanti, N. Caione, I. Russo, G. Gasparro, "Optimization in the Shortest Path First Computation for the Software Routing GNU Zebra," IEICE Transactions Communications, vol. E88-B, no. 6, June 2005, pp. 2644-2649.
[14] LSA Generator Software SPOOF, http://www.cs.ucsb.edu/~rsg/Routing/download.html
[15] RUDE/CRUDE Traffic Generator, http://rude.sourceforge.net/
[16] Network Emulation Software Brite, http://www.cs.bu.edu/brite/download.html
[17] GNU "Quagga.". http://www.nongnu.org/quagga/
[18] The Open Source Initiative, www.opensource.org.

———————————————————————

- *Ashish Kumar Mishra is currently pursuing master's degree program in Information technology in SRM University, Chennai , India, PH-09710031071. E-mail: akmishra1881@gmail.com*
- *P.Selvaraj is currently an assistant professor of Information Technology engineering in SRM University, Chennai, Indi. E-mail: selvaraj.P@ktr.srmuniv.ac.in*