QoS Provisioning for Real Time Services on DiffServ aware IPv6 Optical Network using IXP-2400 Intel Network

Kashif Saleem, Norshiela Bt. Fisal and Muhammad Mun'im B Ahmad Zabidi.

***Abstract*—Providing Quality of Service (QoS) and Traffic Engineering (TE) capabilities in the internet is essential, especially in supporting the requirement of real- time traffic, as well as mission critical applications. Differentiated Services (DiffServ) is an emerging technology that provides QoS and traffic engineering features in Internet Protocol (IP) Network by programming the IXP2400 Intel Network Processor. Since the introduction of the network processors, there have been a number of network services developed based on the special-built, packet processing optimized programmable microprocessors. This paper is mainly concerns on how to deploy DiffServ in order to assess priority functionalities. A scheduling mechanism was developed on the IXP2400 network processor to provide QoS by maintaining priority of incoming packets based on criteria i.e. class of packets and traffic. The queuing mechanism improves the QoS of the traffic at the expense of some performance degradation. To reduce the performance degradation, a cache unit has been added into the operation of the QoS mechanism to cut down SRAM access during the lookup operation. A performance study was then carried out to evaluate the performance of the QoS mechanism after adding the cache unit. The overall speed is enhanced and delay is minimized. At the same time, the mechanism can be implemented on the ENP-2611 Evaluation Board to verify the functionality of the application on hardware.**

***Index Terms*— DiffServ, Microengine, QoS provisioning, WRED.**

I. Introduction

At present, the Internet utilizes best effort delivery based on IPv4 which is not reliable for real time data. There is no guarantee of real time data transfer. A significant percentage of packets are dropped and the delay jitter is high. This jitter, also causes unpredictable variabiliy of delay caused by congestion. Latency in terms of end-to-end delay must be emited in a network throughout the communication process. In the next generation Optical Internet, one must address, among other issues, how to support quality of service (QoS) at the Wavelength Division Multiplexing (WDM) layer [1] [2]. This is because current IP provides only best effort service, but mission-critical and real-time applications require a high QoS (e.g., low delay, jitter, and loss probability) [3]. Supporting basic QoS at the WDM layer will facilitate as well as complement a enhanced version of IP (such as IPv6) [4]. DiffServ architecture can be considered as a refinement to this model, since we more clearly specify the role and importance of boundary nodes and traffic conditioners, and since our per-hop behavior model permits more general forwarding behaviors than relative delay or discard priority [5]. In addition, various service disciplines and scheduling algorithms, which govern per-hop behavior, have been proposed in the literature [6]. However, to date, there are no QoS schemes that take into account the unique properties. Real-time data transfer (voice, video, etc) require QoS guarantee. The following approaches provide QoS guarantees:

- Differentiated Services (DiffServ) - Categorizes traffic into different classes, also called class of service (CoS), applies QoS parameters to those classes [2].
- Integrated Services (InterServ) - Application that requires some kind of guarantees has to make an individual reservation [3].
- Multiprotocol Label Switching (MPLS) - Tagging each packet to determine priority [4].

In this paper we propose a mechanism which can handle priority queuing for IPv4 and IPv6 network traffic by enabling DiffServ. Simulation studies have already shown that there is 0% of packet lose and have proved that delay is minimized as shown in figure 5 and described in [7]. By programming ENP-2611 card with the given mechanism, the IXP2400 will provide better and efficient throughput to the real-time traffic on optical network. The remainder of the paper is organized as follows: in Section II will elaborate the internal process of a network processor. Section III shows the implementation of QoS on the ENP-2611. Section IV describes the study on simulation. Section V illustrates the test case settings and

Manuscript received February 1, 2008. This work was supported in part by the Faculty of Electrical Engineering, University Technology Malaysia.

Kashif Saleem is PhD research student in the Faculty of Electrical Engineering, University Technology Malaysia, Skudai, 81310, Johor, Malaysia (phone: 607-553-5428; e-mail: kashnet@hotmail.com).

Dr. Norshiela Bt. Fisal is the Head of Telematic Research Group and Professor in the Faculty of Electrical Engineering, University Technology Malaysia, 81310-Skudai, Johor, Malaysia. (e-mail: sheila@fke.utm.my).

Muhammad Mun'im B Ahmad Zabidi is Associate Professor in Faculty of Electrical Engineering, University Technology Malaysia, 81310-Skudai, Johor, Malaysia (e-mail: raden@fke.utm.my).

978-1-4244-1980-7/08/$25.00 ©2008 IEEE.

verification results. Traffic delay measurement and figures are given in section VI. Section VII contains conclusion of this research and future work suggestions.

II. Network Model & Internal Process

The processors inside the Intel Network Processor IXP2400 is programmed to route the data by maintainning quality described as in figure 1, when the packets are generated from the user end based as of different categories, some of which having data like e-mail, messaging, file, voice, video etc. According to this real time application packets (surgery, video conference, HDTV, Internet telephony, and digital audio) which can tolerate loss but require dealy eligent.

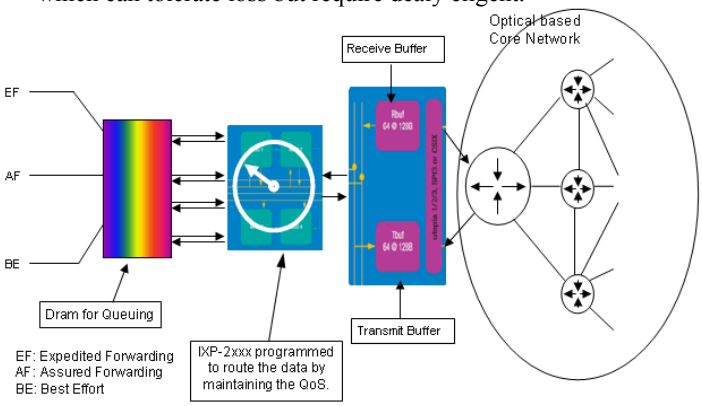

Figure 1: Network Model

Non-real time traffic like e-mail, file transfer, etc can tolerate delay but loss is not affordable. By keeping all of these issues in mind, the priority and sheduling is maintained by programing a IXP2400, to work with buffering, located in physical memory or DDR-RAM of Intel Network Processor [8] [9] as shown in figure 1. When voice and video packets need to pass through router (IXP2400), they will be treated as a real time traffic and processed at the highest priority level.

The first priority will be for the Expedited Forwarding (EF) which handles real time traffic. When EF traffic got some break then second priority for Assured Forwarding (AF) traffic will be forwarded. In this process both incoming traffic and stored packets in the memory will be moved if there is any. The third priority packets are buffered accordingly, when there is no packets or traffic for both EF or AF then the packets for third priority means Best Effort (BE) will be forwarded. Eight processors will work in parallel way to process the packets and maintain the quality of service [7].

III. Implementation Of QoS On The ENP-2611

Although the final stage, implementing the QoS on the ENP-2611, had been called off, some researches had been done on this stage. After coming to Stage Three, it had been learnt that the ENP SDK 3.5 only provides sample application for Static Forwarding. There is no support for the IPv6 Forwarder in ENP SDK 3.5. If the QoS Application is to be implemented on hardware using ENP SDK 3.5, it would involve a great deal of changes to be made on the microblocks structure of the Static Forwarding sample application since the QoS is built on the IPv6 Forwarder application.

As a solution, a newer version of the ENP SDK i.e. version 4.2 was suggested. This version provides support for the IPv6 Forwarder sample application besides the Static Forwarding sample application. In order to apply ENP SDK 4.2, the software mentioned on Stage Two needed to be updated to newer version as well. The software utilize are Ubuntu version 6.06, MontaVista Preview Kit 3.1, IXA SDK 4.2 and ENP SDK 4.2. The installation and configuration of the software is of not much difference from the previous combination of software [7].

IV. Simulation Study

The Workbench provides packet simulation of media bus devices as well as simulation of network traffic. There is a feature under the Packet Simulation called Packet Simulation Status that enables the observation of the number of packets being received into the media interface and number of packets being transmitted from the fabric switch interface. Besides, there is another feature called Stop Control that enables the user to specify a condition to stop or to continue the packet reception (simulation). This feature can be access by select Simulation > Packet Simulation Options > Stop Control in the menu bar of the Workbench. The Packet Simulation Status is used to verify the packet filtering function of the QoS. This is necessary to make sure that the added cache unit does not affect the service ability of the application.

A. Network interfaces

The physical layer or the machine access code (MAC) can be configured by this interface. This is done by characterizing the protocol, data rate, and any parameters which is applicable to the protocol [10].

B. Data Structures

Most of the tasks as access data structures are maintained, and therefore data structures are typically defined prior to Task drawings. Data structures are created and managed in the Data Store pane of the project editor [10].

C. Thread Executables

Thread executables allow the tasks to be grouped and to be allocated for the various threads on the MEs. To allocate the threads, the AT Thread Allocation view may be used [10] Weighted Random Early Detection (WRED) and DiffServ process is added in tasks which are then defined in threads to be processed accordingly.

D. Pipeline Editor

The pipeline editor displays data streams and thread executables on the same canvas. The entries shown within a thread executable are the execution trace entries that refer to code paths contained in the tasks assigned to the thread executable [10].

V. Test Case Settings And Verification Results

To verify the service mechanism, we run 1000000 numbers of cycles which generates the traffic through simulator as shown in figure 2. Under the Stop Control command, the simulation set to be stopped after the desired number cycles.

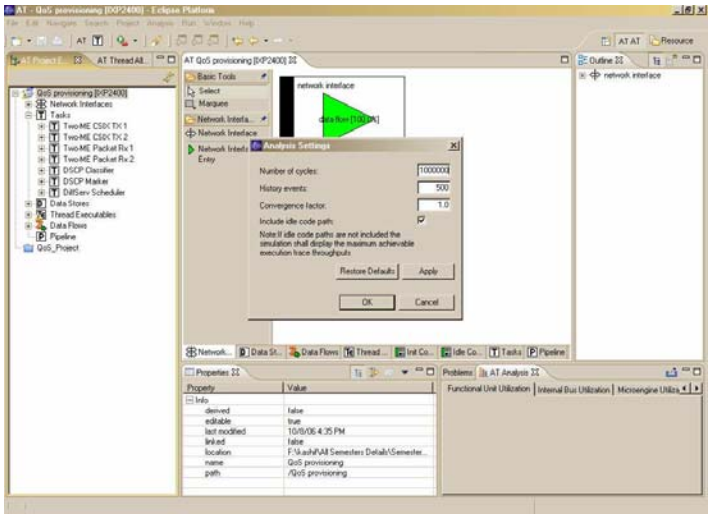

Figure 2: Settings of Rules Configuration for the Test Case

When the analysis is complete, the results are displayed in the AT Analysis view. The AT Analysis view can be maximized to view the results for better inquiry. The analysis results are shown separately through sections, which can be accessed by tabs within the window as shown in figure 3:

- Functional Unit Utilization. (Figure 3)
- Internal bus utilization.
- Microengine Utilization. (Figure 6)
- Execution Trace Throughput. (Figure 4 &Figure 5)
- Thread History.

Figure 3 displays the utilization of each functional unit in the system through bar chart, which includes both static and dynamic utilizations. sram0, sram1, dram, RX & TX shows some utilization because of the I/O reference in the task's code path [7]. The scratch shows utilization because of the idle code path.

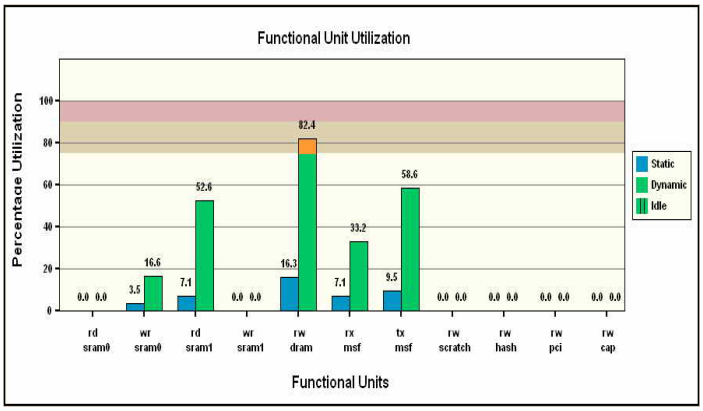

Figure 3: Functional Unit Utilization

Figure 4 & figure 5 shows the packet throughput result of each execution trace. If the throughput for an execution trace is less than the rate injected, then this indicates it in failing to keeping up the line rate.

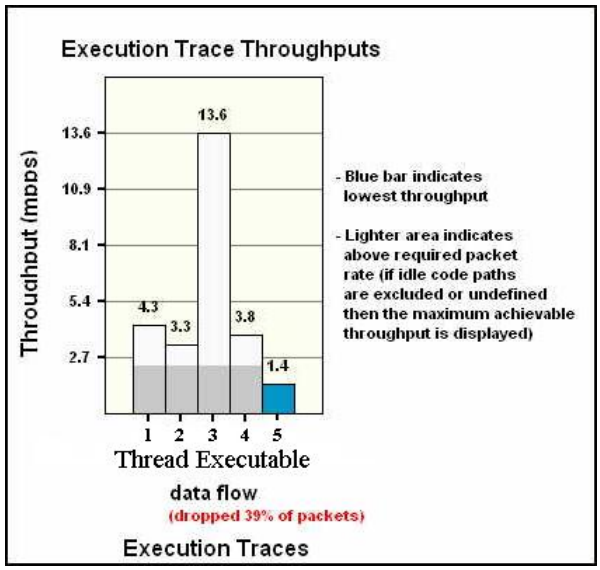

Figure 4: Uneven ME utilization and Without DiffServ

In figure 4 the simulation indicates that some of the packets were being dropped and thus the number of packet being transmitted would never reach the destination. This method provides a quick verification on the correctness of the function.

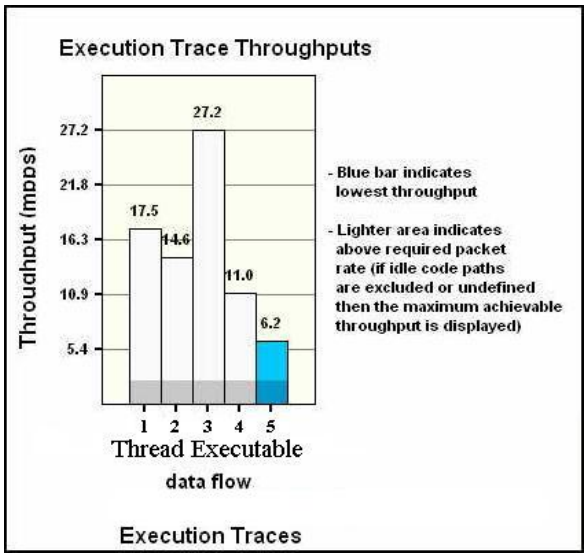

Figure 5: ME configured properly & with DiffServ

Figure 5 shows that all the traffic is transmitted without any drop of packet. This is due to the proper utilization of the Microengines by dividing the tasks accurately in between

these microengines. Figure 5 shows the screenshot of the Packet Simulation Status. This also shows the throughput of all traffic which is received and transmitted without packet loss.

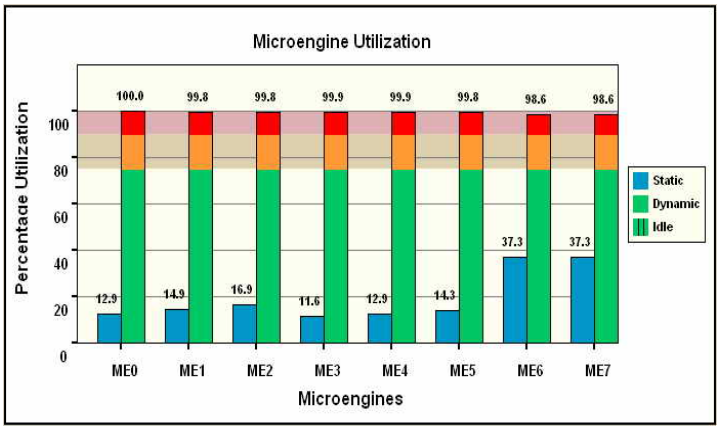

Figure 6: Microengine Utilization

A. Figure 6 shows the compute utilization result of the microengines in the system. The microengine utilization is at a maximum the process is functioning at its full performance.

VI. Traffic Delay

Table 1: Delay in different incoming traffic

Usage in %	Expedited Forwarding (EF)	Assured Forwarding (AF1y)	Assured Forwarding (AF2y)	Assured Forwarding (AF3y)	Best-Effort (BE)
0	0.001	0.001	0.001	0.001	0.082
70	0.001	0.001	0.001	0.001	0.09
90	0.001	0.001	0.001	0.001	0.1
100	0.001	0.001	0.001	0.001	0.103
105	0.001	0.001	0.001	0.001	0.106
110	0.001	0.001	0.001	0.001	0.109
115	0.001	0.001	0.001	0.005	0.112
120	0.001	0.001	0.001	0.01	0.115
125	0.001	0.001	0.001	0.015	0.118
130	0.001	0.0015	0.001	0.02	0.121
135	0.001	0.001	0.001	0.025	0.124
140	0.001	0.001	0.001	0.03	0.127
145	0.001	0.001	0.002	0.035	0.13
150	0.001	0.0017	0.001	0.038	0.133
155	0.001	0.001	0.0015	0.045	0.134
160	0.001	0.001	0.001	0.05	0.1345
165	0.001	0.001	0.0017	0.055	0.135
170	0.001	0.0019	0.0015	0.057	0.1355
175	0.001	0.0013	0.002	0.06	0.136
180	0.001	0.001	0.001	0.0635	0.1365
185	0.001	0.001	0.0014	0.0667	0.137
190	0.001	0.0016	0.0015	0.0699	0.1375
195	0.001	0.001	0.0019	0.0731	0.138
200	0.001	0.001	0.0018	0.0763	0.1385

The deployment of the usage in percentage on x axis and packet delay on y axis and ploting the values as mentioned in table 1, we get the graph as shown in figure 7 (a) and figure 7 (b). The graph shows upward or increasing trend for some traffic. In figure 7 (a) and figure 7 (b) five classes of traffic are shown. Describe through figure 7 (a); BE traffic is having the maximum delay. The delay start rising for AF class 3 packets when traffic utilization reach at 110 %. AF class 2 packets having less delay but slightly graeter then AF class 1 packets. The minimum delay we have get is for the EF traffic which have the real-time data transfer as mentioned in figure 7 (a) and figure 7 (b).

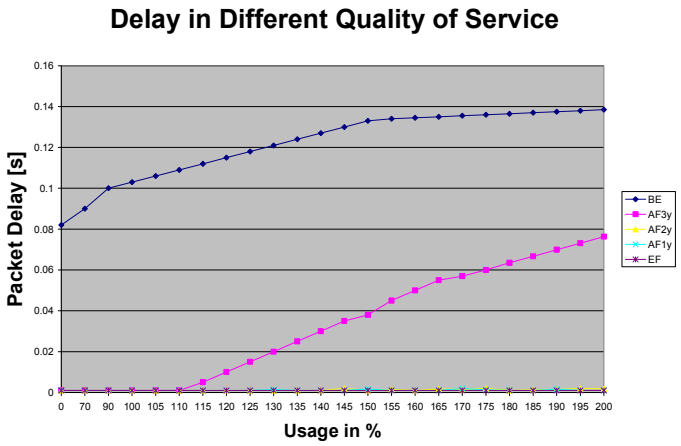

Figure 7 (a): Delay in different incoming traffic (including all five kinds)

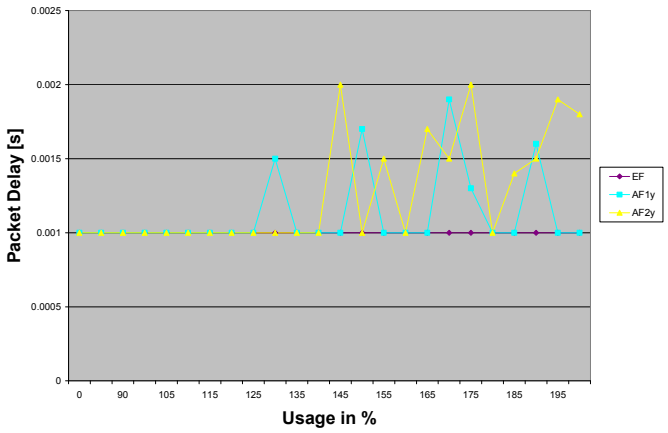

Figure 7 (b): Delay in different incoming traffic (EF, AF1y, AF2y are included).

After the cache unit was verified to be able to function as expected, it was evaluated in terms of performance. Through out the performance study, the original design of the QoS application was being compared with the new design of the application in which a cache unit was added. As the average one way delay without adding cache unit is 165msec, while transferring packet having payload size of 500 bytes, which is minimized to 110msec when enhanced with cache unit.

In this project, performance study were conducted at three levels, namely macro-level, function-level and system-level. The performance was evaluated in term of latency which is represented by the number of microengine clock cycles.

VII. Conclusion

Utilizing Intel® IXP2400 network processor to enhance DiffServ aware IPv6 network for real-time services with minimized packet loss, delay (approx. 5500 micro sec) and managed the data throughput at 7.14 Gbps (2.38 Gbps x 3 ports). Maintain priority scheduling of the packet's in the ENP-2611 and getting the maximized reliability by IXP2400 network processor.

This research aims to provide QoS in IPv6 network through DiffServ service. Packets will be generated from the user end with the help of simulator to analyse the program code and to generate result. IXP2400 is programmed to provide the scheduling based on WRED. The implementation of WRED enables the congestion avoidance and minimization of packet delay in DiffServ aware IPv4 & IPv6 Network. Manage data-dependent operations among multiple parallel processing stages with low latency.

The IXP2400 provide the QoS by maintaining the priority of different kind of packets and having different characteristics. This minimize the packet delay for the real-time data packets in IPv6 DiffServ Network to maximize the throughput of the data like voice, video etc in which delay is not tolerable. Manage data-dependent operations is optimized to reduce latency among multiple parallel processing stages.

The cache unit has been successfully designed and implemented for QoS provisioning. The cache unit comprises of CAM and Local Memory which was designed to reduce the SRAM access during the lookup operation. A performance study was carried out on the application to compare the performance of the scheduling with and without the cache unit applied. The results show in figure 5, that the cache unit helps in improving the performance of application [7].

ENP-2611 is flashed by the code, which is analyzed and generates the better result in this project [7]. To check the network processor properly; deployment of card ENP-2611 in the test bed is essential before it utilize for some real time project.

The hardware implementation could be continued after the corresponding switch is available. Implementing the QoS on one ENP-2611 as mentioned. With these, the expected performance as indicated in simulation could be achieved. As mentioned previously, the original QoS used a software hash to retrieve the rule entries from the SRAM whereas this project introduced a cache unit in the operation. In fact, there is still another suggestion on implementing the lookup operation i.e. using the real hardware-support hash unit. The hardware-support hash unit can support up to 128 bits of hash key.

That is, each of the packet header field is compared one-by-one using the arithmetic instructions. For the case of the hardware-support hash unit, since it can support up to 128 bit of hash key, it is recommended that all of the related incoming packet header fields are used as the hash key to retrieve accordingly. This is expected to be able to further improve the performance operation.

Acknowledgment

I wish to express my sincere appreciation, sincerest gratitude to University Technology Malaysia for their support and special thanks to researchers in Telematic Research Group.

References

[1] C. Qiao and M.Yoo, "Optical Burst Switching (OBS) - A new Paradigm for an optical internet," J High Speed Nets., vol. 8, no. 1, Jan.1999, pp.69-84

[2] Yijun Xiong, Marc Vandenhoute, and Hakki C. Cankaya, "Control Architecture in Optical Burst-Switched WDM Networks," IEEE JSAC, vol.18, no.10, oct.2000, pp. 1838–51.

[3] M. Yoo, C. Qiao, and S.Dixit, "QoS Performance of Optical Burst Switching in IP-Over- WDM Networks," IEEE JSAC, vol. 18, no. 10, Oct. 2000, pp. 2062-71.

[4] Ch. Bouras, A. Gkamas, D. Primpas, and K. Stamos, Performance Evaluation of the Impact of QoS Mechanisms in an IPv6 Network for IPv6-Capable Real-Time Applications, Journal of Network and Systems Management, Vol. 12, No. 4, December 2004.

[5] S. Blake et al., "An architecture for differentiated services," RFC 2475, Dec. 1998.

[6] Mark Allman and Aaron Falk, "On the Effective Evaluation of TCP", ACM SIGCOMM Computer Communication Review, vol. 29, no. 5, pp. 59-70, 1999.

[7] Kashif Saleem, "QoS Provisioning for Real Time Services in MPLS Diffserv aware IPv6 Network using IXP-2xxx (Intel® Network Processor)," Master's thesis, Faculty of Electrical Engineering, University Technology Malaysia, 2006.

[8] Intel Corp. (2003b). Intel® IXP2400 Network Processor Hardware Reference Manual. US: Manual.

[9] *Intel® IXP2400/IXP2800 Network Processor Programmer's Reference Manual*, US: Manual. Intel Corp. (2003c).

[10] *Intel® Internet Exchange Architecture Software Building Blocks Developer's Manual*, US: Manual, Intel Corp. (2003d).