Parallel IP Packet Forwarding for Tomorrow's IP Routers [1]

Jun Wang, Klara Nahrstedt [2]
Department of Computer Science
University of Illinois at Urbana-Champaign
{junwang3, klara}@cs.uiuc.edu

Abstract

The invention and evolution of the Dense Wavelength Division Multiplexing (DWDM) technology has brought a breakthrough to high-speed networks, and it has put a lot of pressure on research in the area of IP routers to catch up. Besides, with up-coming Quality of Service (QoS) requirements raised by a wide range of communication-intensive, real-time multimedia applications, the next-generation IP routers should be QoS-capable. Limited by the Moore's Law, one possible solution is to introduce parallelism as well as the Differentiated Service (DiffServ) scheme [5, 11] into the router architecture to provide QoS provision at a high speed and a low cost. In this paper, we propose a novel architecture called the High-Performance QoS-capable IP Router (HPQR). We address one key design issue in our architecture - the distribution of IP packets to multiple independent routing agents so that the workload at routing agents is balanced and the packet ordering is preserved. We introduce the Enhanced Hash-based Distributing Scheme (EHDS) as the solution. Simulations are carried out to study the effectiveness of EHDS. The results show that EHDS does meet our design goals very well.

I. Introduction

The traffic in the Internet is exploding as it is doubling every year [10]. Emerging multimedia applications, especially the two-way HDTV (High Definition TV) video applications, will make the explosion even faster in the future. With the invention and evolution of fiber-optic technologies, such as the DWDM [8] in the backbone networks, the carrying capacity of fiber links is doubling every 12 months and provides high speed network fabric to meet the future bandwidth demands. However, according to the Moore's Law, the computing power will only double every 18 months, hence the access IP routers will inevitably become bottleneck in the Internet. Moreover, multimedia and realtime applications require timing and other Quality of Service (QoS) guarantees, besides bandwidth, which puts even more burden on the routers. Therefore, the future growth of the Internet requires design and development of high-speed IP edge routers that forward exponentially increasing volume of traffic and provide QoS guarantees at the same time.

In order to bridge the gap between the increase of computing power and the explosion of bandwidth demand, parallelism has been introduced into the routers design. From the viewpoint of the degree of the parallelism, the routers have evolved into the fourth generation [12, 15]. The first generation routers used centralized routing unit and its internal data path was a bus. The line cards and the routing unit communicated with each other through the bus. Obviously the routing unit and the bus were bottlenecks. The second generation routers used distributed routing units instead of the centralized one. Each line card had its own routing unit. But the bus was still the bottleneck. The third generation routers used a switch fabric instead of the bus. In order to meet the future requirements, the fourth generation routers are expected to be the next generation of high-performance QoS-capable routers. By using new architectures, based on parallel and scalable switch fabrics, higher degree of parallelism and scalability will be brought into the new system.

Extensive research has been done on the next generation high-speed routers. Tzi-cker Chiueh and Prashant Pradhan proposed a cluster-based scalable packet router prototype, called Suez [4]. Suez is built on a hardware platform consisting of a cluster of commodity PCs connected by a Myrinet switch with a 10-Gbps backplane. Nick McKeown's group at Stanford University did intensive research on high-speed switching [9, 14, 16]. E. Basturk proposed a design and implementation of a QoS capable switch router based on RSVP protocol [2]. Many other papers and proposals are dealing with some key issues in high-speed routing [10], such as high-speed routing table lookups [7, 21], real-time packet scheduling [13], fine grain QoS control [17], high-speed switches for the data path [3, 14], and so on.

Some companies have focused on this topic too. BBN's MGR project [18] proposed a high-speed router architecture that achieves packet forwarding rate at tens of millions of packets per second, and the router has a backplane speed of 50 Gbps. Avici's TSR and Pluris's Teraplex20 high speed switch routers have been pushed into today's market [1, 19].

Our work has been influenced by these existing systems. However, comparing to these architectures, our prototype focuses on the control plane and QoS issues. This paper presents a novel, highly-scalable architecture, the HPQR (High-Performance QoS-capable IP Router) architecture. Based on this architecture, a specific key design issue - distribution of IP packets to multiple independent routing agents in an efficient manner - is addressed, using Enhanced Hash-based Distribution Algorithm (EHDA) [3]. Efficient EHDA means that (1) packets are distributed in a load-balanced fashion, and (2) EHDA preserves packet ordering for individual TCP flows.

The rest of this paper is organized as follows. Section II gives an overview of our HPQR architecture. Based on the architecture, the packet distribution issue and the EHDA are described in detail. Section III presents the simulation results. Finally, the concluding remarks are given in Section IV.

II. High-performance, Scalable, QoS-capable Control Framework for Next-Generation IP Routers

The basic idea of an IP router is to route IP packets. However, a QoS-capable IP router should have the following basic functionalities:

[1]This work was supported by NSF Grant under contract number NSF CCR 9988199 and NSF CISE Grant under contract number NSF EIA 99-72884.

[2]Please address all correspondences to Jun Wang and Klara Nahrstedt at Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, phone: (217) 244-5841, fax: (217) 244-6869.

[3]In our context, the "distribution" means dissemination of IP packets from a line card to multiple independent routing agents.

(1) Receiving IP packets from incoming links; (2) IP header analysis (such as packet classification, QoS analysis); (3) Routing table lookup; (4) QoS-aware packet scheduling; (5) Transmitting packets to outgoing links. All these functionalities must be parallelized in order to make this router a high-performance and scalable next-generation IP router, which is a non-trivial task.

A. *HPQR Architecture*

Our HPQR architecture belongs to the fourth generation of switch routers.

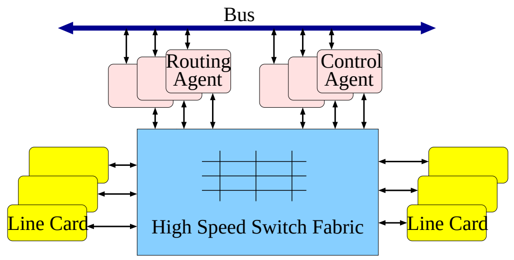


Figure 1: Architecture of HPQR

The architecture of HPQR is shown in Figure 1. It consists of the following components

- *Line Cards (LC)* are responsible for receiving packets from the incoming links and sending packets to the outgoing links. The link cards should be very high-speed to keep up with the high-speed links.
- *Routing Agents (RA)* perform routing table lookups in a parallel manner.
- *Control Agents (CA)* handle routing table computation and QoS control tasks.
- *High Speed Switch Fabric* is the underlying data path which connects LCs, RAs and CAs together. The switch fabric itself can be parallelized.
- *Bus* is dedicated for broadcast, for example, routing table updates, between RAs and CAs.

In our architecture, we separate the IP header analysis (IP packets classification and filtering in case of DiffServ) and the routing table lookups, which usually reside in the same line card in a conventional router. The separation of two functionalities is necessary because:

- *With the separation, the routing table lookup (which is the bottleneck due to memory accesses) and QoS control will move out of the line cards, hence the line cards become more light-weighted.* A light-weighted line card can achieve higher forwarding speed and therefore accommodate higher-speed link interfaces.
- *With the separation, we achieve better load-balancing and therefore higher performance.* The reason is that: (a) we relieve the routing table lookup bottleneck by adding more routing agents into the system without affecting the number of line cards, (b) we distribute the incoming traffic from the line cards to the routing agents in a load-balanced manner since the routing agents are shared by all line cards.

B. *Design of Line Cards and Routing Agents*

Figure 2 shows the design of line cards and routing agents.

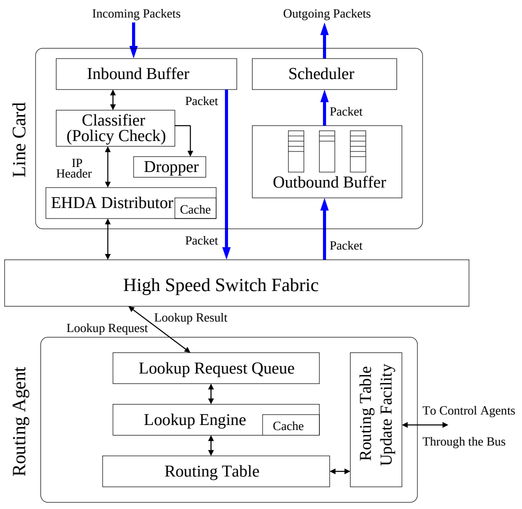


Figure 2: Structure of Line Cards and Routing Agents

After receiving a packet from the incoming link, the packet is put into the *Inbound Buffer*. Its IP header is removed and forwarded to the *Classifier* where DiffServ classification and other policy check are performed. If the packet is a “bad” one (e.g., it is out-of-profile in the DiffServ case, or it does not conform to the access policy, or its TTL expires), it is dropped directly (or might be remarked in the DiffServ case). If the packet passes through, its header goes into the *Distributor*, where it is determined which RA it will go to, by using our EHDA. As soon as the RA is determined, the IP header is put into a *Lookup Request* message. The request goes through the *High Speed Switch Fabric* and is put into the RA’s *Lookup Request Queue*. The *Lookup Engine* gets the request from the buffer and performs high-speed routing table lookup. After that, the IP header is modified accordingly and sent back to its original LC through the switch fabric, together with the appropriate output port number. After receiving the output port number, the *Distributor* picks up the original packet from the *Inbound Buffer*, attaches the new IP header, and forwards the whole packet to the output LC through the switch fabric.

For the outgoing path, outgoing packets are queued into the *Outbound Buffer* before being scheduled. There might be multiple queues in the *Outbound Buffer* with different priorities. QoS-aware scheduling schemes are used to enforce the differentiation between different service classes (e.g., PS-queue and RIO-queue in the DiffServ model).

C. *IP Packet Distribution Approach*

We present an IP packet distribution approach, performed by LCs and RAs, satisfying two requirements:

- the RAs should be working in a load-balanced manner, which is the main reason to separate the RAs from the LCs;
- the packets from the same flow should be transmitted in the order they arrive, especially packets belonging to TCP connections. This is because out-of-order packet transmissions may result in TCP retransmits or even timeouts which deteriorates the overall performance of those TCP connections.

Both requirements are necessary because otherwise, one could optimize the load-balancing at the expense of in-order packet transmission (for example, the packet-based round-robin scheme), and vise versa. Therefore, the packet distribution approach should be carefully designed to tradeoff between these two requirements. One direct solution is the Flow-based Distribution Algorithm (FDA), which distributes packets on a per-flow basis. But FDA is too coarse-grained because the time-scale at which load balancing is done (the time-scale of flow inter-arrivals) may be much larger than the time-scale of packet inter-arrivals. Therefore, the FDA has the following disadvantages: (1) it may result in unbalanced traffic distribution with respect to the individual packets; (2) it requires the packet distributor on each LC to keep track of individual flows so that high overhead may be introduced in case of large number of flows.

As we know, the Internet traffic includes TCP/IP flows and UDP/IP flows. The TCP flows are much more sensitive to packet ordering than UDP flows are. Our novel packet distribution approach, called Enhanced Hash-based Distribution Algorithm (EHDA), will take advantage of this feature to achieve both load-balancing and in-order packet transmission simultaneously. In our approach, packets belonging to TCP flows are assigned to the RAs in a way that the packet ordering is the first consideration, while packets from UDP flows are distributed among all available RAs dynamically to achieve a high measure of the load balancing.

1) Enhanced Hash-based IP Packet Distribution Algorithm

To identify each entity, we assume that each RA, LC and CA has a unique name in the system, i.e., a unique internal identification number (ID#). A simple way of using hash value to distribute packets, called the basic Hash-based packet Distribution Algorithm (HDA), is to generate a hash value for each IP packet, based on its destination IP address, and then use this hash value directly as the ID# of the RA it is forwarded to [17]. Since packets with the same destination address are forwarded to the same RA (the hash value of the same IP address remains the same), packet ordering is naturally preserved. But there are two disadvantages to this method: (1) Although the hash values are pseudo random numbers and therefore supposed to be uniformly distributed, there is no guarantee for that, especially when the destination addresses are not uniformly distributed; (2) The ID#s are bound to destination addresses and there is no way to adapt them, therefore the workload at each RA cannot be tuned dynamically even when the workload is not balanced.

Instead of using the RA ID#s as hash values directly, EHDA uses indirect hashing - the hash values are used as indices to a hash table which contains ID#s of those RAs. The contents of the hash table (RA ID#s) can be dynamically changed according to the instantaneous workload of each RA. The basic idea is, if one RA that is currently being used, say RA-x, is known to be too busy, another idle one, say RA-y, will be found to replace it. The corresponding value in the hash table (RA-x) will be updated to RA-y. Figure 3 shows the structure of the hash table and the brief workflow of the EHDA, implemented at the LC site. Notice that we use different entries for TCP packets from UDP packets. It means the TCP packets and UDP packets may use different RAs for routing table lookup, even they have the same destination addresses.

At each RA, the workload is defined as the average length of its Lookup Request Queue (Figure 2). The current average queue length can be quantized into a 2-bit or 3-bit value (see Equation 1, where 3-bit values are used) and sent back to LCs as piggybacks of Lookup Result messages.

$$Quantized\_workload\_value = \frac{Average\_queue\_length * 8}{Maximum\_queue\_length} \quad (1)$$

In order to keep track of the workload at each RA, there is a "Usage" table at each LC (Figure 3), in which the latest quantized workload value for each RA is stored. For example, in Figure 3, the first entry of the "Usage" table is 3, meaning that the current quantized workload value at RA-1 is 3 (out of 8). Once a LC receives a Lookup Result message, it extracts the quantized workload value as well as the RA ID# from the piggyback. The Usage table can then be updated accordingly.

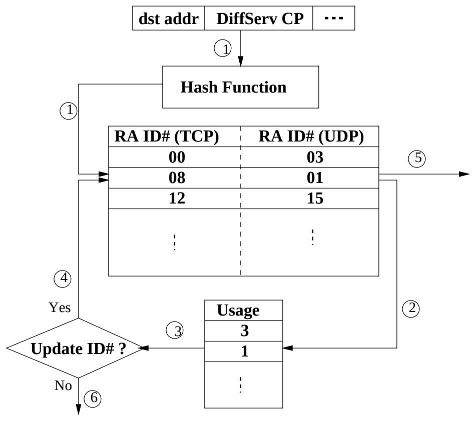


Figure 3: Enhanced Hash-based Distribution Algorithm

In order to differentiate the distribution of TCP and UDP requests, different hash table update strategies are used. Assume that 3-bit quantized values (0 to 7) are used to measure the workload at RAs. One example of the update probability curves is shown in Figure 4. As we can see in this example, when the quantized workload value at an RA reaches 5, all LCs will update their corresponding UDP entries with probability of 0.5 and the probability will increase to 1 when the workload value reaches 6. On the other hand, the update probability of TCP entries remains 0 until the workload value is 6. Even when the queue is full, the TCP entries will be updated only with 0.5 probability so that TCP flows remain more stable compared to UDP flows. Therefore, the packet ordering is preserved as well as high load-balancing is expected. Notice that the actual probability values and workload thresholds can be tuned in a real system.

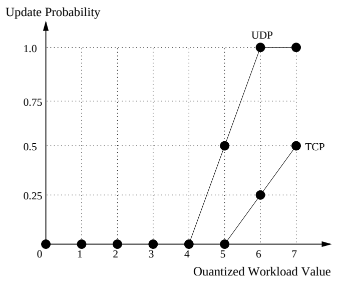


Figure 4: Hash Table Update Probability w.r.t. Workload

Another major reason, why we introduce probability into the hash table update, is to avoid the dramatic workload fluctuations at RAs. For example, when the workload at RA-1 reaches the threshold, if all LCs update their hash table and switch to another RA, the workload at RA-1 will suddenly drop to zero. And later on it may be overwhelmed again suddenly because all LCs find out its workload is zero and change back to it immediately. This is certainly an undesirable effect.

Figure 3 also illustrates the workflow of EHDA.

Step 1: Get the hash value. The destination address is hashed by using a hash function. Other fields in a IP header, such as the DiffServ CodePoint (DSCP) [6] if the DiffServ model is adopted, may also be used to generate the hash value. A typical hash function uses the division method. If the number (r) of RAs is a power of 2, for example, $r = 2^q$, where $q > 1$, then we can: 1) use the CRC code (the generator polynomial $G(x) = x^q - x - 1$) [20] or 2) just randomly select q bits from the destination address to form the hash value. Otherwise, we can use the result of destination address modulo r as the hash value. Theoretically, we can prove that the hash values are uniformly distributed, given that the destination addresses are uniformly distributed, and the hash function can be implemented at high speed.

Step 2: Get the RA ID#. Using the hash value from *Step 1* as the index to the hash table, a RA ID# can be obtained. Notice that we differentiate TCP packets from UDP packets - they use separate parts of the table entry to store the ID# since different update strategies will be used for them.

Step 3: Check the workload and make a decision if the hash table is updated or not. After having the RA ID#, its current workload is checked by looking up the "Usage" table. By using the quantized workload value got from the "Usage" table, a decision is made whether the hash value needs to be updated in the hash table or not, based on the hash table update probability curves the LC is currently using. As described above in Figure 4, different curves are used for TCP and UDP flows. If a positive decision is made, go to *Step 4*, otherwise go to *Step 6*.

Step 4: Update the hash table. The RA ID# with the lowest workload value is chosen from the Usage table. The original entry in the hash table is updated with the new RA ID#.

Step 5: After updating the hash table, the new RA ID# is used as the final result.

Step 6: The hash table is not updated, so the original RA ID# is returned as the final result.

In a real implementation, both the hash table and the usage table can be put into a cache memory at the LC site to achieve high speed since they are small.

III. Simulations and Results

In order to examine the effectiveness of our design, simulations are conducted and the results are analyzed. We have developed a basic event-driven simulator and some intensive simulations have been carried out on it. The simulation results prove that our design goals are met very well.

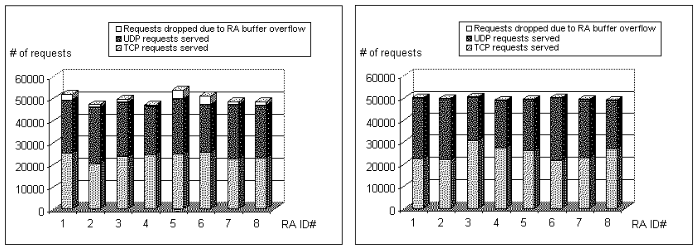
A. Simulation Model

We model the input traffic as TCP and UDP flows. Each flow has its own starting time t_{start}, ending time t_{end} and sending rate r. Flows arrivals follow Poisson distribution and the duration of each flow is exponentially distributed. The sending rates are randomly picked within a certain range. In most of our simulations, we fix the number of LCs (N_{lc}) to be 4 and the number of RAs (N_{ra}) to be 8, without losing generality. The basic HDA is used as the comparison to our EHDA.

B. Simulation Results and Analysis

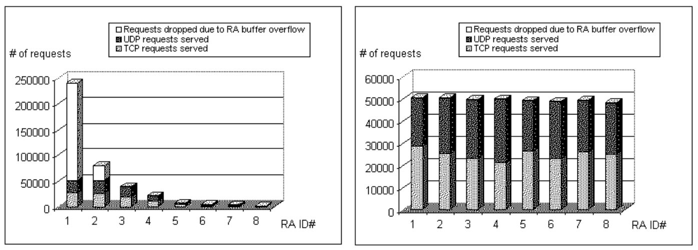
The first set of simulations is conducted to investigate the effectiveness of EHDA. The curves in Figure 4 are used as the hash table update strategy. Figure 5 illustrates the results under balanced input traffic and Figure 6 gives the results under unbalanced input traffic [4].

From Figure 5 we can learn that the loss rates due to the RA buffer overflow [5] are 3.91% for the basic HDA and almost 0% for our EHDA, which proves that even in the case of balanced input traffic, the EHDA



(a) Basic HDA (b) EHDA

Figure 5: Distribution of lookup requests under balanced input



(a) Basic HDA (b) EHDA

Figure 6: Distribution of lookup requests under unbalanced input

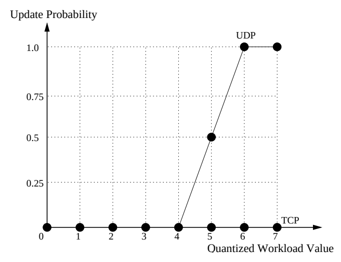
outperforms the basic HDA. In the case of unbalanced input traffic, which is shown in Figure 6, the EHDA is superior to the basic HDA because it makes the workload at RAs balanced so that the loss rate due to buffer overflow is reduced significantly from 54.75% to 0%.

In order to study the expense of achieving more balanced workloads and lower loss rates at the RAs, the second set of simulations is carried out. In this set of simulations, we evaluate the number of the hash table changes or updates. The hash table updates imply the changes of the assigned RA of some flows, so that the packets from such flows might be transmitted out of order. Therefore, the smaller the number of the hash table updates, the better ordering of the packet transmission. Since the packet ordering is much more important to TCP flows than to UDP ones, TCP flows should have higher priority in the sense of the packet ordering. For example, in the case of Figure 5, the total number of hash table updates for TCP and UDP flows is 1079 and 5561 respectively. And in the case of Figure 6, the number is 1178 and 5460 respectively. From these numbers we can conclude that the EHDA gives TCP flows higher priority over UDP flows in the sense of the packet ordering, since TCP flows have less number of hash table updates than UDP flows. By conducting more simulations we also know that the number of hash table updates is independent of the input traffic pattern (balanced or unbalanced). But it does depend on the hash table update strategies (for example, the one was used in the past two sets of simulations is shown in Figure 4). In fact, we can even tune the update strategies (or curves) to achieve different degrees of ordering guarantees for TCP flows.

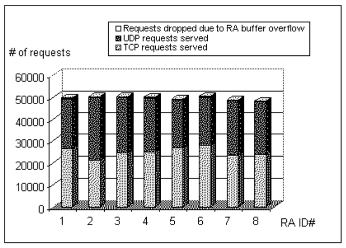
Figure 7 shows an example of a hard guarantee for TCP flows, in which no hash table update is allowed to occur for TCP flows except during intervals when no TCP flow is active. For UDP flows, the number

[4] "Unbalanced" here means the destination addresses of input traffic are NOT uniformly distributed. For example, in Figure 6, nearly 80% input traffic is directed to RA-1 and RA-2, thereby causing large loss rate at RA-1 and RA-2, while the other RAs are relatively idle.

[5] The dropped requests include both TCP and UDP requests.



(a) Hash table update strategy

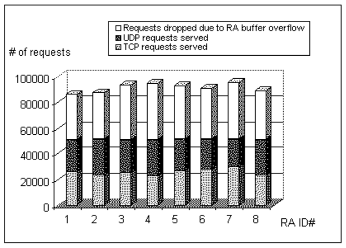


(b) EHDA results

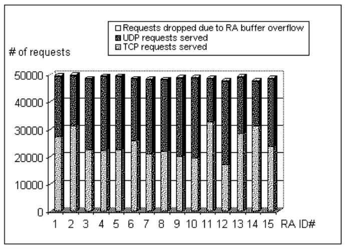
Figure 7: A hard guarantee for TCP packet ordering

of updates surprisingly decreases from 5460 to 4105 [6]. Meanwhile, the loss rate increases slightly from 0% to 0.16%.

The last set of simulations is conducted to study the scalability of our architecture. Because of the separation of the routing table lookups and the distributed structure of RAs, our architecture is supposed to be scalable to accommodate higher volume of input traffic, which means when the volume of input traffic increases and the existing RAs can no longer handle it, the problem can be solved by adding more RAs into the system. The simulation results suggest that this goal has been achieved.



(a) EHDA with 8 RAs



(b) EHDA with 15 RAs

Figure 8: Upgrade the system from 8 RAs to 15 RAs

Figure 8(a) shows the result with 8 RAs and doubled volume of input traffic. It is clear that the computing power of RAs are no long sufficient to handle the high volume of input traffic (the loss rate due to RA buffer overflow is as high as 44.14%). After adding 7 more RAs into the system, with the result shown in Figure 8(b), the upgraded system with totally 15 RAs is capable to handle the input traffic again (the loss rate is back to 0%).

Therefore, from the three sets of simulations, our results show clearly that:

- The workload at the RAs is balanced, no matter what the input traffic pattern is (for example, the original input traffic at the LCs might be unbalanced and the destination addresses of IP packets may not be uniformly distributed).
- The packet ordering for TCP flows is preserved and the degree of guarantee can be tuned.
- High scalability is provided. In the case that the workload is too heavy so that the computing power at RAs becomes insufficient, the system can be upgraded by adding more RAs.

[6]In this example, balanced input traffic is used. It is the same as the input traffic used in Figure 5.

IV. Conclusion

This paper presents a novel, parallelized and highly-scalable control framework for next-generation IP routers. The HPQR architecture is introduced and justified. Based on the architecture, the IP packet distribution algorithm is addressed. As a key element of the HPQR architecture, the EHDA algorithm is studied in detail. Intensive simulations are carried out and the simulation results and analysis are introduced. The results prove that our original design goals for HPQR have been met very well.

V. References

[1] Avici Systems, http://www.avici.com/whitepapers.html. *Technical White Papers*, 2000.

[2] E. Basturk and et al. Design and Implementation of a QoS Capable Switch-Router. *Computer Networks*, 31(1-2):17–30, January 1999.

[3] Cheng-Shang Chang and et al. On Service Guarantees for Input Buffered Crossbar Switches: A Capacity Decomposition Approach by Birkhoff and von Neumann. In *Proceedings of IWQoS'99, London, England*, pages 79–86, May 1999.

[4] Tzi-cker Chiueh and Prashant Pradhan. Suez: A Cluster-Based Scalable Real-Time Packet Router. In *Proceedings of 20th International Conference on Distributed Computing Systems, Taipei, Taiwan*, pages 136–144, April 2000.

[5] S.Blake et. al. An Architecture for Differentiated Services. *RFC 2475*, December 1998.

[6] F.Baker, D.Black, S.Blake, and K.Nichols. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *RFC 2474*, December 1998.

[7] Pankaj Gupta, Steven Lin, and Nick McKeown. Routing Lookups in Hardware at Memory Access Speeds. In *IEEE Infocom 1998, San Francisco*, volume 3, pages 1240–1247, April 1998.

[8] Jeff Hecht. Wavelength Division Multiplexing. http://www.techreview.com/articles/ma99/hecht.htm, March/April 1999.

[9] Sundar Iyer, Amr A. Awadallah, and Nick McKeown. Analysis of a Packet Switch with Memories Running Slower than the Line Rate. In *IEEE Infocom 2000, Tel-Aviv, Israel*, March 2000.

[10] S. Keshav and R. Sharma. Issues and Trends in Router Design. *IEEE Communications Magazine*, pages 144–151, May 1998.

[11] K.Nichols, V.Jacobson, and L.Zhang. A Two-bit Differentiated Services Architecture for the Internet. *RFC 2638*, July 1999.

[12] V.P. Kumar, T.V. Lakshman, and D. Stiliadis. Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet. *IEEE Communications Magazine*, pages 152–164, May 1998.

[13] D. Love, S. Yalamanchili, and J. Duato. Switch Scheduling in the Multimedia Router (MMR). In *Proceedings of IPDPS'00, Cancun, Mexico*, pages 5–11, May 2000.

[14] Nick McKeown. A Fast Switched Backplane for a Gigabit Switched Router. *Business Communications Review*, http://www.bcr.com/bcrmag/12/mckeown.htm, March/April 1997.

[15] Nick McKeown. An Introduction to Packet Switching. *EE370 ISL Seminar Presentation, Stanford,* http://tiny-tera.stanford.edu/∼nickm/talks/EE370_Jan00.ppt, Jan. 2000.

[16] Nick McKeown and et al. The Tiny Tera: A Packet Switch Core. In *IEEE Micro 1997*, pages 26–33, Jan./Feb. 1997.

[17] Masataka Ohta and et al. Hash Parallel and Label Parallel Routing for High Performance Multicast Router with Fine Grain QoS Control. In *Proceedings of IWS'99, Suita, Japan*, pages 13–16, Feb. 1999.

[18] C. Partridge and et al. A 50-Gb/s IP Router. *IEEE/ACM Transactions on Networking*, 6(3):237–248, June 1998.

[19] Pluris, Inc., http://www.pluris.com/pdfs/Pluris-Avoid-Bottlenecks.pdf. *Avoiding Future Internet Backbone Bottlenecks*, 2000.

[20] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ 07458, January 1996.

[21] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable High Speed IP Routing Lookups. In *Proceedings of ACM SIGCOMM'97, Cannes, France*, September 1997.