IntServ6: An Approach to Support QoS over IPv6 Networks

Jhon Padilla
Pontificia Bolivariana University, Colombia
jpadilla@upbbga.edu.co

Mónica Huerta
Simón Bolívar University, Venezuela
mhuerta@usb.ve

Josep Paradells
Technical University of Catalonia, Spain
teljpa@mat.upc.es

Xavier Hesselbach
Technical University of Catalonia, Spain
xavier.hesselbach@entel.upc.es

Abstract

This paper proposes an approach to support End-to-End Quality of Service over IPv6 networks. In our solution, IPv6 flow labels are used to give better performance into the process of packet classification on network routers. In order to evaluate our proposal, a router model is obtained and it is compared with other technologies as IntServ and MPLS. As result, we obtain a solution with benefits of QoS support and label switching in IPv6 routers.[*]

1. Introduction

In the last decade, integration between IPv6 and *Label Switching Technologies* was studied several times. In 1996, an Internet draft was published by Baker and Rekhter [1]. At that time, several discussions were done in IETF's IPng Working Group, but that approach was discarded. However, in 1997 Rekhter published the 2105 RFC [2], after which, a hard competition was created to establish a *Label Switching* solution. Some of them are *Tag Switching* by CISCO, *Aggregate Route-based IP Switching* (ARIS) by IBM, *IP Navigator*, *IP Switching* by IPsilon, *Cell Switch Router* (CSR), and the proposal that was adopted as an IETF Standard: *MPLS* (Multiprotocol Label Switching).

Another aspect of great importance today is the Quality of Service (QoS); current trends are the use of MPLS technology to support traffic engineering [3, 4] and Diffserv [5] in internet backbone. Differv is used in this environment due to its capacity to support great amounts of flows and to group them in classes. On the other hand, IntServ [6] is used to support QoS in access networks due to its fine granularity and its adaptation to user characteristics of Quality of Service. In addition, a lot of efforts are focus in using MPLS to do routing with QoS in fixed and mobile networks [7-9]. Under this panorama, our proposal finds its greater utility in the access networks, because those use IntServ to support QoS.

Our proposal, known as IntServ6, uses IPv6 flow label [10] in order to make a process similar to label switching (tagging) and at the same time, it use IntServ to support QoS. Therefore, our approach presents a hybrid performance similar to MPLS in some aspects and similar to IntServ in others.

It is necessary to clarify that MPLS needs special routers to do label switching [11], so that to make fast IP packages routing, it needs MPLS transport network to send packages from the source IP network to the destination IP network. An advantage of this solution is a high speed for packet transport, and at the same time has the capability of works with IPv4 and IPv6 packages. MPLS allows to do routing with QoS restrictions, using signalling protocols like Constraint based Routing over the Label Distribution Protocol (CR-LDP) or Reservation Protocol (RSVP) to establish the path adapted to QoS's restrictions [7, 12].

On the other hand, our proposal offers an intermediate solution that brings QoS with an improvement in routers processing time respect to IntServ. In addition, routers with different technology from IPv6 are not necessary to transport label switched packets.

The content of this article is distributed in the following form: section 2 has the explanation of our proposal (IntServ6); in sections 3 and 4, the evaluation

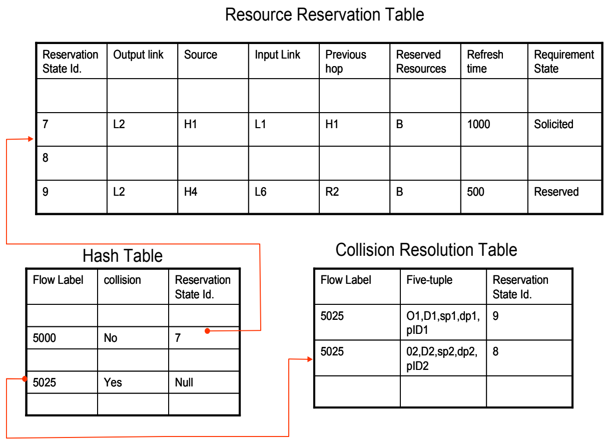
This paper presents some results from RUBI Project. It is sponsored by CICYT from Spain (code: TIC2003-01748).

of the new proposal is done by means of a mathematical model for routers in the process of packet classification and the simulation of the obtained equations with its respective analysis; finally conclusions are considered in section 5.

2. Our proposal: IntServ6

Our proposal is based on IntServ architecture [6] for QoS support, and it improves the current IntServ routers performance. In order to achieve this objective, IntServ6 uses IPv6 flow label to speed up the process of packet classification on routers.

The packet classification process of an IntServ router, uses five fields of received IP packets (Source IP address, Destination IP address, Source port, Destination port and Protocol Identification), commonly named *Five-tuple*, to obtain a *Hash number*. This number is used to search a pointer to the *Resource Reservation Table* in a *Hash Table;* where *Resource Reservation Table* is used to hold scheduling information. Each time a packet arrives, a packet classification process is done.



Resource Reservation Table

Reservation State Id.	Output link	Source	Input Link	Previous hop	Reserved Resources	Refresh time	Requirement State
7	L2	H1	L1	H1	B	1000	Solicited
8							
9	L2	H4	L6	R2	B	500	Reserved

Hash Table

Flow Label	collision	Reservation State Id.
5000	No	7
5025	Yes	Null

Collision Resolution Table

Flow Label	Five-tuple	Reservation State Id.
5025	O1,D1,sp1,dp1, pID1	9
5025	02,D2,sp2,dp2, pID2	8

Figure 1. Reservation table management in IntServ6

On the other hand in our solution, named *IntServ6*, packet classification process in routers is improved doing the *Hash Number* calculation at Source Host instead at each router. This is feasible and it is not a problem because the resulting *Hash Number* is the same independently of calculation site due to both, the used parameters (*Five-tuple*) and calculation algorithm are the same. In our proposal, the *Hash Number* is used to identify packet flows and this number is carried into the *Flow Label* field of IPv6 packets.

Another important aspect in our proposal is that while reservation setup is done, the *Hash Number* is carried to each router on to the flow path using PATH and RESV messages of RSVP protocol [13]; this is necessary to create a new entry to the Hash Table. Thus, during normal data packet forwarding (see figure 1), each IntServ6 router extracts the *Flow Label* field from IPv6 packets and uses it as a search index to find Reservation parameters and the next forwarding link in *Resource Reservation Table*. This information was previously held in Resource Reservation setup. The use of *Flow label* field was described in [10, 16, 17] and our proposal was created according to them.
There is an additional problem due to the use of the *Hash number*: a *collision* may occur. A *collision* occurs when, during the *Reservation* setup, a *Hash number* originally calculated at the Host is already assigned at the Router for another flow. To solve this problem, it is necessary to use a *Collision Resolution Table* (see figure 1), which holds information of collided flows, such as *flow label* field, *five-tuple*, and a Resource Reservation entry pointer. When a *collision* happened, the router searches an entry for a *flow label* and *five-tuple* matching in the *Collision Resolution Table*, and extracts the pointer to *Resource Reservation Table*.

3. IntServ6 evaluation

To evaluate our proposal, we have based our study on other studies performed by B. Hardekopf en [18] and J. Kaur in [19]. In these documents, a performance router model was developed, where the router was build with IXP1200 Intel processor. This router model has three different tasks: packet classification, route selection and packet scheduling. Each task was modeled with two different disciplines, and since different combinations are used, different technologies as IP (best-effort), IntServ and MPLS could be constructed. To model packet classification, *tagging* and *matching disciplines* were studied and modeled; *tagging* is commonly known as *Label Switching* and is used on MPLS routers, and *matching* is known as *Hashing* and is used on IntServ routers. However, the Hardekopf and Kaur models were created without the hashing collision effect and therefore, in our model we have improved this weakness. In our case, a main modification was done in packet classification, and thus, *tagging* and *matching* disciplines were studied to build the model. In the following, we describe the Hardekopf´s model for packet classification and then, we explain the implemented modifications to add collisions effect.

3.1. Original Model

This model was developed in [18, 19], where the model was described by equations (1) and (2) respectively.

$$BW_m=(8k+2)*32*(C/M) \quad (1)$$

$$H=((k+1)*L_{sram} + L_{sdram}) * (C/M) \quad (2)$$

The modeling parameters in these expressions are:

Memory bandwidth (BW_m): Their units are bits per second (bps). BW_m is the number of bits that are taken from memory within a processor task.

Threads number (H): Is the number of processor threads necessary to perform packets forwarding with the required input link speed.

Equation (1) is composed of three items, the first is (8k+2) which is the number of memory references performed; the second item is the processor word size (32), and the third part is the number of packets per second (C/M). A detailed meaning of each part is explained as follows:

- Number 8 is the row size of *Hash Table*. Its units are *words*.
- *k* is the maximum number of readings from *Hash Table*. This parameter is explained below.
- Number 2 is the row size of *Resource Reservation Table* (in words).
- Number 32 is the processor word size (given in bits/word).
- C is the speed of input link (given in bits per second).
- M is the packet size (given in bits).

For a better understanding of equation (1), the modeled situation should be described (see figure 2). It has a *Hash Table* with pointers to Resource reservation parameters (*Reservation State* and *flow descriptor*) as entries. These entries are named *Hash Slots*. When a packet is processed by a thread to obtain the flow identification (*hash number*), the table is read to get the *Reservation State* and to modify the *flow descriptor* (modified parameters are *maximum delay* or *deadline*, and *number of packets per flow*). These entries are named *Hash Slots*. When a packet is processed by a thread to obtain the flow identification (*hash number*), a read from this table should be done to read the *Reservation State* and to modify the *flow descriptor* (modified parameters are *maximum delay* or *deadline*, and *number of packets per flow*). To modify the *flow descriptor*, a resource locking should be done during a time interval to avoid another threads do modifications on this parameter during the prescribed time. This is the moment in which a collision may occur because two threads could try to access the same *hash slot*. In the worst case, k threads have the same *hash number* and at this moment, a collision should be resolved. The last packet will wait (k-1) times in order to access the *Hash Table*, therefore, the last packet will access the *hash table* at the *k-th* attempt, i.e. *k* readings to *hash table* are performed in the worst case. In addition, one more access to memory is necessary to modify the *flow descriptor* and thus, the total number of memory accesses will be (*k*+1).

Under the above conditions, the number of collisions is the same as the number of threads, and then if k increases, the worst case packet processing time will increase too. The value of *k* is 1 for the tagging discipline because there is no calculation of hash number in this case; thus, the *Reservation State* of the flow is analyzed and the *flow descriptor* is modified without any other previous step.

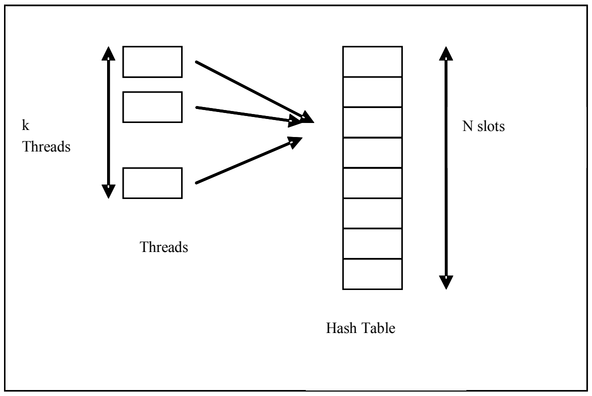


Figure 2. Original situation

On the other hand, the equation (2) was developed taking care of operations for each packet as follows:

- One reading from SDRAM will be performed to obtain a packet.
- In the worst case, one reading from *hash table* (allocated in SRAM) will be executed in *k-th* attempt to find appropriate entry on the table.
- After a string was found, one writing in SRAM will be done to modify the flow descriptor.

Therefore, the classification occupies one thread for an average of (*k*+1) references to SRAM per packet and one reference to SDRAM per packet. L_{sram} and L_{sdram} correspond to the access times to SRAM and SDRAM respectively.

To summarize, equations (1) and (2) are generic models that can be applied to *tagging* (k=1) as well as to *matching* (k>1).

3.2. Modifications to Hardekopf's Model

Hardekopf's model does not have the effect of hashing collisions, i.e. doesn't take care of two flows with the same hash number. We have introduced this aspect in our approach, and since it should be compared with MPLS and IntServ models, modifications to these models were implemented in order to achieve the same conditions.

The collision effect is introduced in equations (1) and (2) through a parameter which represents the process of reading the collision table. Thus, equation (1) will be:

$$BW_m=(8k+2+6mC_h)*32*(C/M) \quad (3)$$

It is important to note that item $6mC_h$ was added. This item is composed of mC_h item (Collision resolution table size) and number 6. It means that, in the worst case the flow will be searched and found at the end of the table, and then mCh readings are performed. The parameter m is the number of flows allocated on *Hash Table*, and C_h is the *hash collision rate* for these m flows. Each time this hash table should be read, 6 words are accessed because a row has the following fields: hash number (32 bits), five-tuple (104 bits for IPv4 and Reservation state pointer (32 bits).

For equation (2), the modified model introduces mCh readings to Collision resolution table, and the new equation becomes:

$$H = ((k + 1 + mC_h) * L_{sram}+ L_{sdram}) * (C/M) \quad (4)$$

3.3. IntServ6 Model

The IntServ6 model for H (Number of Threads) corresponds to equation (4). However, the model for the Memory Bandwidth takes into account the hashing collisions and is similar to equation (4) with some modifications. This model is observed in equation (5).

$$BW_m=(1k+2+11mC_h)*32*(C/M) \quad (5)$$

The modifications are implemented on the hash table size because in our proposal, the hash table is composed by: collision field (1 bit) and Reservation State field (20 bits). Flow label field is used as index and is not a part of hash table. However, in figure 3, flow label appears as field because it is a simple way to explain our proposal operation. Therefore, we need 1 word to read one entry of hash table (this aspect appears in equation (5) as 1k).

Another modification consists in the use of readings of 11 words to extract one entry of the Collision Resolution table. This is because one row of this table has the fields: flow label (20 bits), five-tuple (296 bits for IPv6) and Reservation state identifier (20 bits).

In equation (5), k is 1 since the search process in the hash table is the same as tagging. Finally, the collision rate value is the same as the number of collisions of a perfect hash function with uniform distribution of hash numbers [20]. This expression is observed in equation (6):

$$C_h = \frac{N*\left(1-\left(\frac{N-1}{N}\right)^m\right)}{m} \quad (6)$$

In this equation, N is the size of the hash table and m is the number of flows allocated in the hash table. To obtain the complete expressions, Ch should be replaced in equations (3) to (5).

Table 1. Characteristics of the architectures in the general model

Architecture	Number of readings in the hash table (k)	Size of collision resolution table
IntServ	>1	mC_h
MPLS	1	0
IntServ6	1	mC_h

4. Results

On this work, three simulations are done considering the characteristics of each one of the architectures. In table 1, we can see that the IntServ architecture has k readings in the hash table at the worse case, since it uses matching as classification method. Also we saw that MPLS and IntServ6 architectures only take one reading because they use the value of the label as index search in the routing table (Tagging). On the other hand, MPLS has Ch = 0 because it doesn't present collisions, while IntServ and IntServ6 present collisions; the reason is that they use a hash number to identify the flows. The simulations made and their results will be explained in the next section.

4.1. Relation between memory bandwidth (BW_m) and hash table readings (k)

The memory bandwidth based on k was simulated, maintaining the number of flows (m) constant. The bit rate of the input link was 400Mbps and package size was 512 bits. Figure 3 shows variation of normalized memory bandwidth. The variation of these parameters affect IntServ, the increase of readings in the hash table is consequence of the increase of threads. As result of this, memory bandwidth in IntServ grows proportionally with the number of threads. This doesn't affect MPLS or IntServ6, the number of readings in table is always 1 (k=1) due to the use of the Tagging, for this reason memory bandwidth for these two architectures stays constant.

The amount of hash collisions, in other words, the allocation of the same hash number to two different flows stays constant for that case. In our proposal (IntServ6) hash table size is constant with value 2^{20} (*flow label* has 20 bits of length) and flows number (*m*) is constant in 6400.

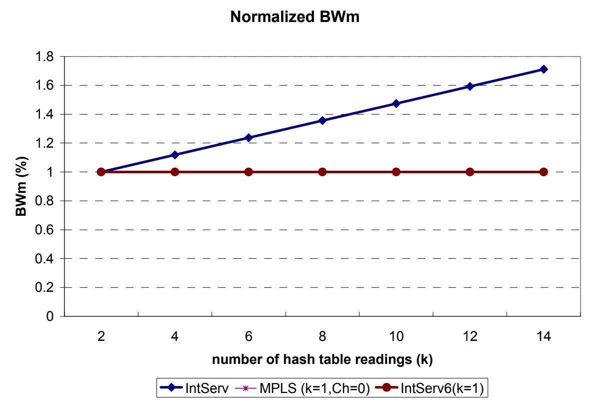


Figure 3. Normalized memory bandwidth vs. hash table readings

4.2. Relation between memory bandwidth (BWm) and flows number

These results are observed in figures 4 and 5. We simulated memory bandwidth under the following conditions:

- Hash table size of 2^{20}
- Variable amount of flows (m)
- Capacity of input link 400Mbps (Maxine capacity of the IXP1200)
- Package size was 512 bits
- k had a value of 4 (number of threads of the IXP1200)

In this case we observed that the memory bandwidth remains constant for MPLS, whereas IntServ and IntServ6 present an increase on number of flows. Memory bandwidth in IntServ6 has a better behaviour than IntServ for fewer flows, whereas for 4000 flows or more is equal to IntServ. Growth is exponentially. The Figure 4 shows that for small number of flows (smaller or equal to 1000) IntServ6 improved the bandwidth of MPLS. In addition, the increase of memory bandwidth in IntServ6 is consequence of the use of 128 bits for IPv6 addressing. This aspect is surmountable, when the memories storage capacity in semiconductors increases and the word size in processor increases too.

4.3. Relation between Throughput and number of threads of the Router

The Throughput of the process of flow classification was simulated, that is, how many bits per second can process a router for each one of the technologies when the number of threads increases. The results are observed in fig 6. If considers that this Router can works with connections until 400Mbps, the three technologies can works at that speed without problems. Nevertheless, when increasing the number of threads the MPLS behaviour is the best followed by IntServ6, whereas IntServ presents the worse of the behaviours. This aspect is an advantage since when being increased the benefits of the processor our proposal will improve its performance with respect to the IntServ architecture.

5. Conclusions

This article presents an adaptation of the IntServ standard to be supported in IPv6 networks. This proposal, called IntServ6, uses the facilities of IPv6 to support Quality of Service by means of the flow label field. An advantage on IntServ6 is to have a great adaptability to the conditions of the end users, because uses reserve of resources by flows, which makes IntServ ideal to use in access networks based on IPv6. In addition, due to use of labels for flow identification in our proposal, it is easily mapped in the Edge Routers of a MPLS transport network.

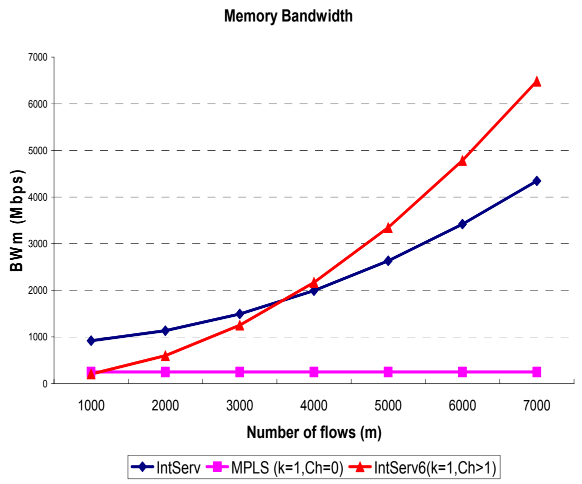


Figure 4. Memory bandwidth vs. number of flows. (BW_m scale:1000).

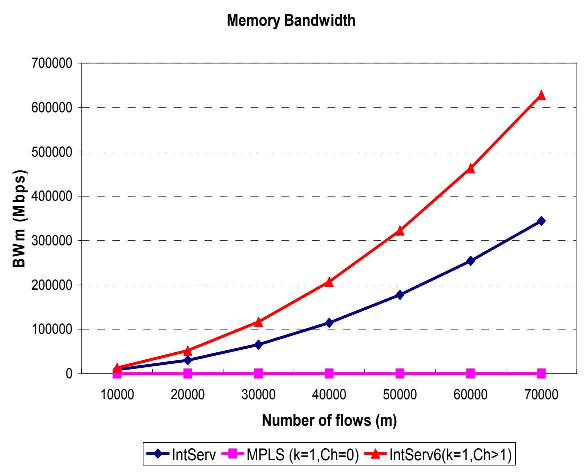


Figure 5. Memory bandwidth vs. number of flows.(BWm scale: 10000).

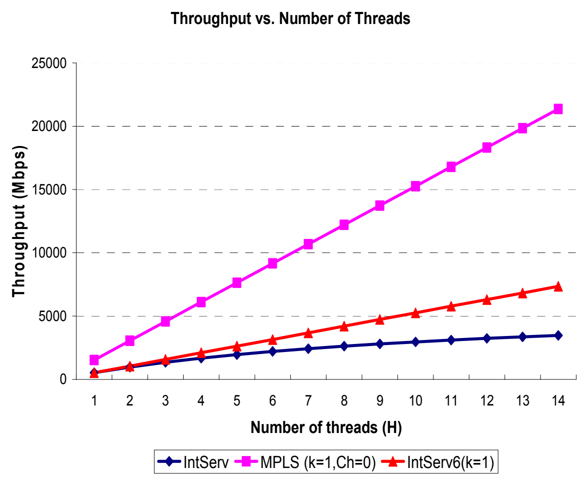


Figure 6. Throughput of packet classification process.

On the other hand, although MPLS continues being the technology that better benefits presents for the flow classification process, is a solution that is applied better on backbone transport networks, but it is not an end to end solution. However, in spite of giving benefits of label switching for IntServ6 are smaller than MPLS, it is a solution that allows end to end solutions of Quality of Service in an Internet based on IPv6; additionally presents remarkable improvements in the benefits related to IntServ and label switching.

6. References

[1] F. Baker and Y. Rekhter, "The use of Flow label for Tag Switching", IETF, Internet draft, October 1996.

[2] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, "Cisco Systems' Tag Switching Architecture Overview", IETF, RFC 2105 February 1997.

[3] J.-M. Chung, "Analysis of MPLS traffic engineering", presented at Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on, 2000.

[4] D. O. Awduche, "MPLS and traffic engineering in IP networks", *Communications Magazine, IEEE*, vol. 37, pp. 42-47, 1999.

[5] S. Blake, D. Black, and M. Carlson, "An Architecture for Differentiated Services", IETF, RFC 2475 December 1998.

[6] R. Braden, "Integrated Services in the Internet Architecture: an Overview", IETF, RFC 1633 June 1994.

[7] O. Aboul-Magd and B. Jamoussi, "QoS and service interworking using constraint route label distribution protocol (CR-LDP)", *Communications Magazine, IEEE*, vol. 39, pp. 134-139, 2001.

[8] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjali, "Adding QoS protection in order to enhance MPLS QoS routing", presented at Communications, 2003. ICC '03. IEEE International Conference on, 2003.

[9] M. M. Shahsavari and A. A. Al-Tunsi, "MPLS performance modeling using traffic engineering to improve QoS routing on IP networks", presented at SoutheastCon, 2002. Proceedings IEEE, 2002.

[10] S. Deering, "Internet Protocol version 6 (IPV6)", IETF, RFC 2460 December 1998.

[11] B. Jamoussi, "Multiprotocolo Label Switching Arquitecture", IETF, RFC 3031 January 1999.

[12] P. Pan, G. Sawallow, and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", 2004.

[13] R. Braden and L. Zhang, "Resource Reservation Protocol (RSVP)", IETF, RFC 2205 September 1997.

[14] J. Ash, "Applicability Statement for CR-LDP", IETF, RFC 3037 January 2002.

[15] A. Farrel, D. Papadimitriou, J.-P. Vasseur, and A. Ayyangar, "Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) Establishment Using RSVP-TE", IETF, draft-ietf-mpls-rsvpte-attributes-04.txt July 2004.

[16] C. Partridge, "Using the Flow Label Field in IPv6", IETF, RFC 1809 June 1995.

[17] A. Conta and S. Deering, "IPv6 Flow Label Specification", draft-ietf-ipv6-flow-label-03.txt September 2002.

[18] B. Hardekopf, T. Riché, J. Mudigonda, M. Dahlin, and H. Vin, "Impact of Network Protocols on Programmable Router Architectures", Laboratory of Advanced systems Research, Department of Computer Science, University of Texas at Austin, Austin, Texas April 2003.

[19] J. Kaur, "Scalable Network architectures for providing per-flow service guarantees", in *Faculty of the Graduate School*. Austin, Texas: Dissertation. The University of Texas at Austin, 2002, pp. 153.

[20] Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*, 1st edition ed: Morgan Kaufmann, March 15, 2001.