Design of a High-Performance Active queue management

Hattab **Guesmi*,** Ridha **Djemal***, Kholdoun **Torki**** & Rached **Tourki***
**Laboratoire d'Electronique et de Micro-Electronique, Faculté des sciences de Monastir, Tunisia*
***Laboratoire TIMA-CMP, France*
e-mail : Hattab.guesmi@fsm.rnu.tn

Abstract. *In this paper we present a scalable architecture for Active queue management used in high-performance switching with support for fine-grained quality of service (QoS) guarantees. Quality of services guarantees in terms of delay, throughput and loss rate can be provided by using mechanism's support like scheduling and buffer management in switching architecture of packet switching networks. Our architecture consists in a new data structure memory management based on a priority circular linked list. It uses the pipelined sorted circular linked list of priorities for the active queues management. In addition, to be very fast, the architecture also scales very well to support a large number of priority levels and a large queue size. We give a detailed description of this new data structure related to our proposed algorithms and its corresponding implementation.*

Keywords. Internet protocol networks, IP switch, QoS, design, AQM, DiffServ

1. Introduction

The traffic in the internet is exploding as it is doubling every few months and the speed of technology is doubled every two years. Emerging multimedia applications will make the explosion even faster. The growth of the Internet requires design and development of high-speed IP switch that forward exponentially increasing volume of traffic and provide QoS guarantees at the same time. Optimal throughput and delay performance is obtained by using output buffered switches. Moreover, since upon arrival, the packets are immediately placed in the output buffers, it is possible to better control the latency of the packet. This helps in providing QoS guarantees [5]. While this architecture appears to be especially convenient for providing QoS guarantees [1, 2, 3]. When packets arrive at each input and all packets are directed to the same output, this means that the bandwidth towards each output must be equal to the sum of the bandwidths available on all input lines. At each output, packets are stored in different queues for different classes of services. In order to respect quality of service for every class of flow, queues are designated as one sorted priority queue for every class of service.

QoS means a series of service requirements that network should satisfy to delivering data. It can be represented by many parameters such as: delay, delay jitter, loss rate, bandwidth, etc [4]. QoS control is to provide consistent, predictable and controllable data delivery service, and to satisfy different application requirements. In fact, QoS has to guarantee different classes of packets at different levels of services. There're many mechanisms to support QoS, such as the resource reservation (RSVP), admission control in Integrated Services (IntServ) and traffic shaping/marking in Differentiated Services (DiffServ). It allows the IP traffic to be classified into a finite number of service classes that receive different treatments. Switches at the network edges classify packets into predefined service classes based on the demand requirements and characteristics of the associated application. Core switches forward each packet according to its class. By this way, the model provides service differentiation on each node (Per-Hop behaviours) for large aggregates of network traffic. DiffServ achieves scalability and manageability by providing quality per traffic aggregate and not per application flow. Our proposed architecture of the AQM which provides QoS guaranties is decomposed in three blocs presented by the figure 1.

All recently proposed packet-scheduling algorithms for output-buffered switches that support quality of services (QoS) transmit packets in some priority order, e, g, according to dead-lines, virtual finishing times, eligibility times, or other time stamps that are associated with a packet [15, 16, 17]. Since maintaining a sorted priority queue introduces significant overhead, much emphasis on QoS scheduler design is put on method to simplify the task of maintaining a priority queue. In our architecture we use a fast and a scalable pipelined priority queue architecture for use in high-performance switches with support for fine-grained quality of services guarantees. This paper presents a novel, highly-scalable architecture for an AQM architecture is organized as following: Section two introduces the active queue management and its data structure which is the priority circular linked list. Then, the implementation of the architecture of the AQM that supports the QoS management is presented in Section three. Then at section four design results are presented. At last the paper is concluded.

2. The active queue management

The QoS parameters are optimized by the mechanism of scheduling and memories management. Therefore the creation, access and release of a received flow are driven in a dynamic way according to the number of declared flows and memory capacity [9, 10, 11]. These latters are adjustable according to the dynamic parameters of the algorithm of queue management. The number of flows per class is not static, but it is dynamic according to the allocated memory capacity. To meet these needs, the suitable data structure of these architectural choices is implemented in the dynamic queues management (figure 2). This architecture is composed of a label assignement unit, a queue controler unit and a P-CLL management unit. The assignement unit assigns the incoming packets with a certain priority value which is calculated according to the implemented scheduling algorithm. In our case the label is the virtual finish time calculated according to the CBWFQ (Class Based Weighted Fair Queueing) algorithm. The unit of P-CLL manager contains P-CLL priority queues. Each element in the P-CLL holds a priority value, which interconnect packets having the same priority level. The queue controller maintains a lookup table with entries corresponding to each piority level. Each entry consists of a pointer to a packets list has the same. We refer to this list as a priority list. This structure is the shape of a sorted circular linked list, it is an implementation per-flow queueing (per-priority queueing). So it is more general to also handle various different priorities in the same way priority for same flow. At every moment the P-CLL contains only the values of active priorities e-g of the lists of priority which are not empty. Figure 1 presents the architecture of active queue

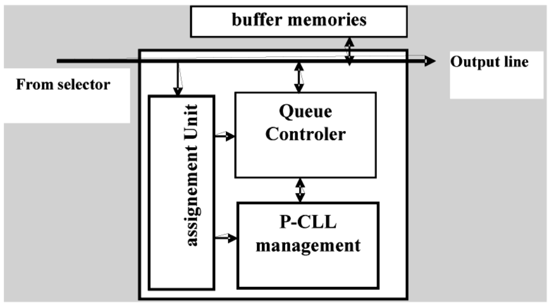

Figure 1: the output queue manager

When a packet needs to be inserted into the queue, the priority assignment unit stamps the packet with a suitable priority value. The queue controller unit determines whether a priority list already exists for the stamped priority value. If it does, it simply adds the new packet to the corresponding priority list. However, if the list does not exist, the queue controller unit creates a new priority list. It also signals the P-CLL manager unit to perform an enqueue operation, which inserts the new priority value into the P-CLL in a sorted manner. This is done to make sure that the highest priority stays at the top of the P-CLL

When a packet needs to be removed from the queue, the P-CLL manager unit determines the non-empty priority list by looking at the topmost element of the P-CLL and sends this priority value to the queue controller unit. The queue controller accesses the corresponding priority list and removes a single packet from it. If this causes the priority list to become empty, the P-CLL manager unit initiates a dequeue operation which removes the topmost element from the P-CLL while making sure that the P-CLL remains sorted.

2.1. Data Structure of the active queue management

This module implements the selected data structure which is the priority circular linked list P-CLL. This list consists of a list of the service classes descriptors, these descriptors inter-connects a priority circular linked list of priority levels. Each cell of these descriptors of the priority levels has an active priority level which gathers all packets having the same priority level in a circular linked list which is called priority list (figure 2).

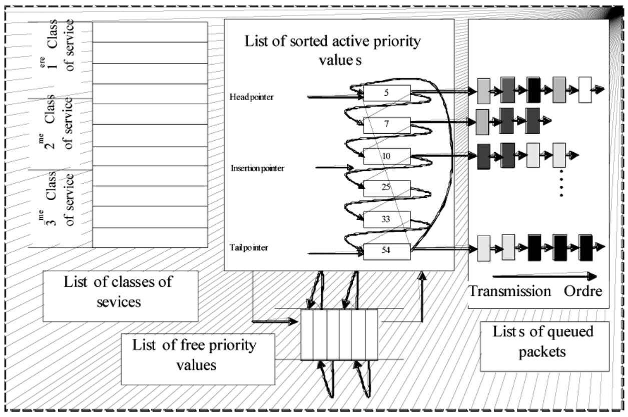

Figure 2: data structure of the output queue manager

The P-CLL data structure maintains various priority values sorted in the circular linked list. Each enqueue operation requires N times to be serviced, where N is the size of the P-CLL. However, the emulation of the circular linked list which implements the priority queuing mechanism is easy to make and it provides a big save in term of time constraints. Furthermore, this structure allows a pipelined implementation compared with the conventional circular linked list and provides a constant time operations. In order to improve the performance of our P-CLL, we implement the method of insertion based on a content structured memory, where each priority queuing operation is performed in only one time. The priority value represents their address in the memory

2.2. Priority queue operation on the P-CLL

The motivation behind defining a new data structure is to be able to pipeline the enqueue and the dequeue operations on it. The conventional circular linked list operations, though very simple to implement, execute in O(n) steps, where n is the number of elements in the circular linked list, and they cannot be easily pipelined. On the other hand, architectures like the systolic array and binary heap can be pipelined, but have extremely high hardware requirements. Our purpose is to design a modified circular linked list data structure and associates algorithms with it, which while being simple and easy to implement, can be easily pipelined to provide constant time priority queueing operation at a low hardware costs [7, 8].

2.2.1. The enqueue operation

To enqueue a new value of priority in the circular linked list of the P-CLL, we need to find the insertion position of the cell in a sorted manner. This operation is achieved by exploring the valid data path from the table T (queue controller unit) values from the value of insertion cell which allows us to determine the inserted cell address or the address pointer. If the pointer of the cell was determined, we modify the flag in table T with one (priority list non-empty), which indicates the state of the priority list. Then, we add this cell to the circular linked list P-CLL in a sorted manner and we carry out the list updated. The insertion position is identified by determining the first least weak priority level of non-empty priority list and by reading the table T from the inserted value address. If we determine the pointer of the cell which at the least weak priority we will determine the insertion position of the priority level. Thus, the operation of addition consists on reading and writing the bond pointers of the two cells and updating the class descriptor service list. The operation of research requires N time where N is the non-empty priority levels number. However, if we use a structure addressable memory, the operation of research needs only the conversion time of the priority level into address.

2.2.2 The dequeue operation

The dequeue operation extracts the cell pointed by the header pointer from the P-CLL since it has the highest priority value making the cell inactive and hence decreasing the file capacity.

3. Implementation of the AQM

3.1. Priority assignment Module (PA)

This module stamps the packets by a label calculated according to the implemented scheduling algorithm. This label determines the transmission order of the packets to the output line and needs to know the three following factors:

- The weight allotted to the concerned class of service: the concept of weight determines the band-width percentage that the class is seen allotting.
- The length of the packet of this service class;
- The interaction with the other concurrent active classes.

In order to stamps packets by labels, this unit calculates the virtual finish time and the virtual start time according to the scheduling algorithm. The calculated values related to each assigned packet level represent the transmission order. According to these values, packets are stored in suitable queue or class of service. Since our architecture implements N_{cs} classes of services, the calculation of priority values needs a fluid model which is implemented in a data base in order to describe the state of each service class and its interaction with the others. This data base is constructed using a cash memory to be consulted and updated after any reception operation in order to provide the correct priority values [14, 16].

3.2. P-CLL manager module (PM):

Our design of the P-CLL manager unit uses two separate memories: one contains the class of service descriptors and the other contains the P-CLL list. The first memory implements the control parameters of each class of services which is organized as Ncs blocs (Ncs: number of service classes), each bloc has Npc fields (Npc: number of control parameters of each service class). The second memory which implements the free priority values linked list is organized as Npv * 4 fields (Npv: number of priority values; one field for the priority value, one for the pointer and two fields for the head and tail pointer). The variation of the memory size depends only of the number of service classes for the first memory and the number of priority values for the second one. The size of these memories varies in a linear way according to the increase of the number of levels of priority managed by this architecture and the number of service classes which increase the architecture performances [17, 18]..

We are used two separate memories in our design for tow reasons: Firstly, the architecture will be scalable by a simple adding space memory to increase the number of priority values. Secondly, the dynamic adjustment of the bandwidth between flows.

3.3. Queue controller module (QC):

This module is used to control all priority levels related to the active queue management: which can be switched between the active and the inactive state. It is represented by a table which stores all the priority values with their associated parameters. When an assigned packet is received, this module seeks in the table if the priority value is active. If yes, so this module launches an addition operation requested by this packet with its corresponding priority list. The addition of the packet is the determination of the available address of the priority list by reading the field address pointer in the table and the update of the sorted packet list. In the other case, this module inserts (enqueue) this priority value in its suitable place in the sorted circular linked list. During the enqueueing operation, QC issues an available address to the priority value. This available address is appended to a linked list for the class list. During the emission, this module outputs pointers of the packet to be transmitted. If this packet is the last in the list, therefore it launches a dequeue operation to make the level of priority inactive in the table. The dequeue operation is achieved by the QC which outputs the pointer to the cell buffer at the head of the queue of the highest priority level found (figure 3).

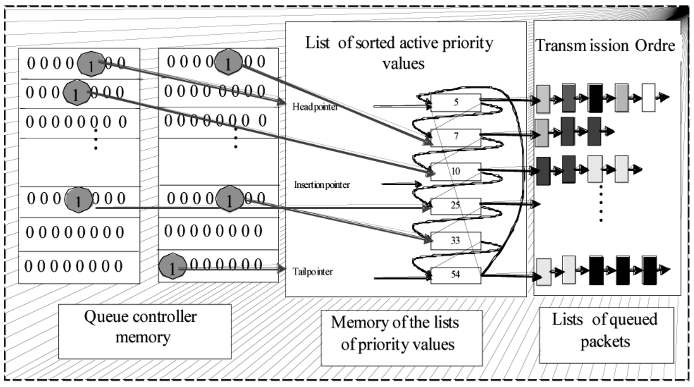

Figure 3: structure of the queue controller module

The design of the queue controller module uses parallel memories organized as an NxN table where all priority levels are represented by one bit which indicate their state (empty or full). This table is organized that all priority values are indicated by their line and column number in the table which facilitate the access to any priority level in one time slot. The memories size of this component is equal to Npv bits (Npv: number of priority levels) which increases linearly with the increase of the number of priority levels [6, 13].

4. Design results of the proposed AQM

In general, the process of designing a system will proceed from a behavioral to a physical representation, gaining implementation details along the way. High level synthesis converts a behavioral specification of a digital system into an equivalent RTL design that meets a set of stated performance constraint [4, 6]. The designer describes his system with a high level specification at one of abstraction levels. This description with a HDL (Hardware Description Language) is synthesized using existent synthesis tool allowing passage to the next abstraction level until reaching either integration in ASIC or implementation in FPGA. After design verification, a design compiler is used to perform logic synthesis. The result of synthesis is a gate level description of the system. The logic synthesis tool starts with two kinds of information: an RTL specification given in VHDL and a functional unit library that can include complex functional units.

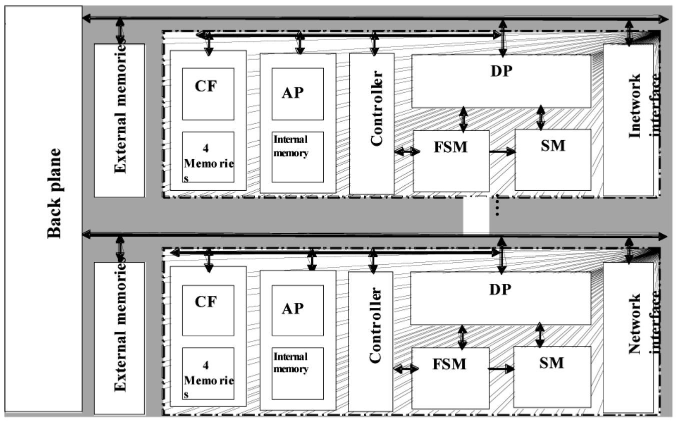

Figure 4: architecture of the OPC switch IP

Our proposed architecture of the AQM consists of a set of interfaces and control components. This architecture includes different parts as depicted in Figure 4 which is described with the VHDL language at the RTL level [17, 18]:

The memory part Our proposed architecture is based on the QoS dynamic management. This brings us to define the suitable data structure and to choose the appropriate technique. Memories are used to implement selected techniques for QoS optimization.

The finite state machine (FSM): it receives the primitive of the system and it defines the primitive execution order.

The sequencer (SM) it allows the generation of all necessary commands and control signals for the data path operation.

The data path (DP): In order to perform all operations required by the entire architecture, this module contains an interface on one hand to registers and on other hand to dedicated and standard operators (counters addition, etc…).

The controller: To reconfigure architecture according to the need for the QoS management, and to administer these memories a general controller is designed to assure the dynamic QoS management.

Assignment priority (AP): implements the dynamic model of the priority calculus.

Queue controller (CF): implements the queue controller module.

The network interface: This unit is responsible for the reception/emission of data from the network interface/buffers.

The design is simulated and synthesized using Design Analyser of Synopsys. The resulting netlist is used as input to SoC Encounter (Cadence) in order to perform mapping and routing with a 0.35 µm CMOS technology. The final ASIC its area is about 13.7 mm2 and it consists of 215837 gates. This circuit operates with clock frequency of 250 MHz and allows the cells transfer on 2 Gbit/s. It can process 65536 priority levels and requires two types of cache memories: the first one has 128x32Bits size used by the AP module, the second one 265x64 bits size, the CF module use four parallel memories to construct a 256x256 table which holds 65536 priority levels.

5. Conclusion

The growth of the Internet led to a set of reflexions on the manner of optimizing the use of this network so that the applications can take advantage from suitable services instead of undergoing overall a policy known as "BEST-EFFORT". Qualities of Service became impossible to circumvent and required the installation of various mechanisms of QoS management at various levels of network architecture. Through this study, we presented the concepts of quality of service as well as the components of optimization of the QoS parameters (loss, delay, bandwidth and gigue). This study guided us to propose the architecture of an AQM optimized for the differentiation of services. This architecture is made up primarily by three blocks which are: assignment priority unit, P-CLL manager unit and queue controller unit. At this stage the packet will be treated and stored in its proper class of service where various operations are executed to respect the QoS desired parameters for this class. These operations are queueing, buffer management and scheduling to transmit the packet to its destination with respect of their parameters (loss, delay, jitter and bandwidth). This architecture has been modeled in VHDL language at RTL level of the various blocks in order to design this switch and to validate it at a low level.

References

[1] Shang-Tse Chuang, , A. Goel; N. McKeown, B. Prabhakar, Matching output queueing with a combined input/output-queued switch, IEEE Journal on Selected Areas in Communications, 17(6), 1030-1039(1999).

[2] Pankaj Gupta, Nick McKeown, Algorithms for packet classification, IEEE Network, 15(2), 24-32 (2001)

[3] O. N-M Varvaja, étude des algorithmes d'attribution de priorités dans un internet à différentiation de service, thesis de l'université de Renne1, France, (2001)

[4] Paolo. Giaccone, Queueing and scheduling algorithms for performance switches, thesis de l'université de polytechnique de Torino, Itali, (2002

[5] Sally Floyd, Van Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking, Volume 1(4), 397-413 (1993).

[6] J. Feng, G. rubino, J-M Bonnin, a QoS routing algorithm to support classes of service, conference, proceeding of ICIIEM 2001, Pekin, octobre (2001).

[7] N. Ni, L-N Bhuyan, fair scheduling and buffer management in internet router, conference, conference, INFOCOM2002, vol3, New York, (2002).

[8] A. C-K. Kam, efficient scheduling algorithms for quality of services guarantees in the internet, thesis de l'institut de technologie de Massachust, (2000).

[9] Pankaj. Gupta, S. Lin, N. Mckeown, Routing lookups in hardware at memory acces speeds, conference, in *Proc*. IEEE INFOCOMM'98, San Francisco, 1240-1247, (1998).

[10] R. Bhagwan, B. Lin, Design of a high-speed packet switch for fine-grained quality of service guarantees, conference, IEEE International conference on communication (ICC'00), vol3, New Orleans, 1430-1434, (2000).

[11] Henry C. B. Chan, Hussein M. Alnuweiri, Victor C. M. Leung, A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers, IEEE Journal on Selected Areas in Communications, 17(6), 1013-1029 (1999).

[12] Donpaul C. Stephens, Jon C. R. Bennett, Hui Zhang, Implementing Scheduling Algorithms in High-Speed Networks, IEEE Journal on Selected Areas in Communications, 17(6), 1145-1158, (1999)

[13] R. Mahajan, S Floyd, D. Wetherall, controlling high-bandwidth flows at the congested router, conference, 9th ICNP, (2001).

[14] Seshasayi Pillalamarri, S. Ghosh, "high-speed networks: definition and fundamental attributes", article, computer communications 28 (2005) 956-966.

[15] Zoubir Mammeri, "framework for parameter mapping to provide end-to-end QoS guarantees in IntServ/DiffServ architectures", article, computer communications 28 (2005) 1074-1092

[16] Fengyuan Ren, C. Lin, B. Wei, "a robust active queue management algorithm in large delay networks", article, computer communications 28 (2005) 485-493.

[17] Hattab. Guesmi, R. Djemal, B. Bouallegue, J-P. Diguet, R. Tourki, "high performance architecture of integrated protocols for encoded video application", article, computer standards and interface 26 (2004) 301-315.

[18] H. Guesmi, R. Djemal, B. Bouallegue, H. Youssef et R. Tourki, "Architecture d'un routeur IP de haute performance optimisée pour la différentiation de qualité de service", conference SCS'04, 18-21 Mars 2004, Monastir – Tunisia.