Research on Distributed Packet Fair Queueing with Feedback Mechanism(DF^2Q)

Xiaoxia SUN, Jianping WU, Yong JIANG
Department of Computer Science and Technology
Tsinghua University, Beijing, 100084, China

Abstract- **As to now, the router is required to provide not only high forwarding performance, but also advanced quality of service. Packet scheduling in buffered queues is required in all algorithms to support QoS, such as Fair Queueing. Output queueing systems can achieve desired quality of service, but lack of essential scalability. Input buffered systems, while scalable, lack the necessary quality of service features. In this paper we design and implement DF^2Q (Distributed Feedback Fair Queueing), on the basis of CIOQ(Combined Input Output Queueing) architecture. The most important feature of this algorithm is the feedback mechanism. It can avoid internal congestion effectively and improve the efficiency of resource utilizing. We discuss the performance of DF^2Q in different cases. And at last research directions and open problems in this area are proposed.**

I. INTRODUCTION

There's much research on designing scheduling algorithms to support QoS. Most of them assume output queuing. When a packet arrives at an output-queued switch, it is immediately put into a queue that is dedicated to its outgoing line, where it waits until departing from the switch. That approach can be used by a switch or a router to control the packet's latency, and hence provide QoS guarantees. But output queuing has not met the needs as the emergence of high-performance router with a large number of ports and with port rate of 2.5Gigabits(OC-48) or more. Such as a high-performance router with N ports, its internal switch fabric and memory must work at N times line speed, that is, speedup of N. At the same time its control logic in output ports must run at a very high speed, so the expansibility of output-queuing systems is poor. To solve those problems, most high-performance routers (research[14], commercial[15]) choose architecture with input-queuing. By adding buffers in input points, internal switch and memory visit in routers can work on a lower speed. But it lacks of the necessary QoS features.

So how to combine the input-queuing and output-queuing is to be considered. One way is to use CIOQ switches. It is shown in [1] that speedup of 2 in CIOQ(Combined Input and Output Queuing) method is sufficient to resemble the output pattern of any output queuing switch. So we can employ proper packet scheduling algorithms in input and output ports to provide QoS guarantee for different applications. In this paper we proposed a scheduling algorithm DF^2Q (Distributed Feedback Fair Queuing) under a distributed CIOQ architecture. This algorithm introduces a feedback mechanism from output port to input ports to avoid congestion and HOL in input ports. Our packet scheduling algorithm can provide guarantee of QoS similarly to output queuing. Further more in that CIOQ distributed architecture there is no need of global synchronization, but implements high speed of pipelining. So forwarding performance can be guaranteed. The feedback mechanism detects congestion in output ports, then send congestion information to input ports according to some forecasting rules when congestion is about to happen. Input ports slow down the speed transmitting to that output port on the basis of the feedback, so congestion can be alleviated or avoided.

The organization of paper is as follows. First we review issues in packet scheduling research, then we introduce distributed architecture of our high-performance router and the requirement in packet forwarding. In Section 4, we discuss the scheduling algorithm we employed, especially the feedback mechanism. Performance test and analyse are presented in Section 5. Further discussion is in Section 6, and in the end research directions and open problems are proposed.

II. BACKGROUND

A. Packet Scheduling

Many packet fair queuing algorithms based on per-session are proposed in order to provide QoS [4],[6],[11],[12]. Most of them are output queuing, and as discussed above they lack of expansibility. So input queuing is introduced. But it increases competition points. There are competitions both in input and output ports. No competition exist in input ports if FIFO is used for scheduling, but HOL will appear instead[10]: if the packet on top of a queue is blocked, the packets in the same input queue but directed to other block-free output ports are blocked too. How to settle the competitions in the router with input queuing is what we should consider most. But most of the research nowadays put emphasis on improving throughput in switch fabric [16],[17]. Research on how to provide QoS in input buffered and medium speedup environment is still less.

B. Packet Fair Queuing

The proposed packet scheduling algorithms can be classified into three categories: work-conserving, non-work-conserving and hierarchical algorithms. Summarization about packet scheduling algorithms can be seen in [3]. The original target of packet queuing is to avoid congestion, but proper packet scheduling algorithm can also provide guarantee for QoS. At present scheduling algorithms that commercial routers support most is packet fair queuing algorithms (PFQs).

Packet Fair Queuing algorithms approximate the fluid GPS (General Processor Sharing) system. In a GPS system with N sessions, session i is characterized a positive real number

0-7803-7097-1/01/$10.00 ©2001 IEEE

F_i. During any time interval the server serves the packets at the head of the non-empty queues simultaneously, in proportion to their service shares. The service that session i got in time interval [t1,t2], denoted by $W_i(t_1, t_2)$, can be described as follows:

$$W_i(t_1,t_2) \geq \frac{\Phi_i}{\sum_{j \in B(t1,t2)} \Phi_j} r(t_1 - t_2)$$

B(t1,t2) is the sessions that are backlogged in time interval [t1,t2].

Each packet fair algorithm should maintain a system virtual timer V(t), and associate with each session i a virtual start time $S_i(t)$ and a virtual finish time $F_i(t)$. Intuitively, V(t) represents the normalized fair amount of service time that each session has received by time t, $S_i(t)$ represents the normalized amount of service time that session i has received by time t The target of all PFQ algorithms is then to minimize the discrepancies among $S_i(t)$ and V(t). The calculation and update of $S_i(t)$ and $F_i(t)$ follow the following (1),(2)

$$S_i(t) = \begin{cases} \max(V(t), S_i(t)), & \text{i becomes active} \\ S_i(t) + \frac{L_i^k}{r_i}, & p_f^k \text{ finishes service} \end{cases} \quad (1)$$

$$F_i(t) = S_i(t) + \frac{L_i^k}{r_i} \quad (2)$$

All PFQ algorithms differ in two aspects: the computation of the system virtual time and the packet selection policy. The choice of different system virtual time functions and packet selection policies will affect the complexity and fairness properties of the resulting PFQ algorithm.

The packet scheduling algorithm, DF^2Q, as will be discussed later, is used in our high-performance router. It is a kind of PFQ algorithms. There are multiple fair queues in input and output ports, and they are connected by the central switch fabric.

III. Distributed Router Architecture

A. Hardware Architecture

Our high-performance router employs distributed architecture with multiple processors. There's one master processor, several slave processors, as figure 1. Master processor process routing protocol packets, maintain global routing table, keep synchronization between local and global routing table, configure and manage of the router, while slave processors is responsible for forwarding IP packets by using local routing table, filtering IP packets using security database, supporting multiple network interface protocols, supporting the different interface units of the processor

Using such distributed architecture is to support high-speed data forwarding so as to meet the requirements of high-performance routers.

B. Logical Structure of Forwarding and Scheduling Mechanism

The logical structure of forwarding and scheduling mechanism is shown in Figure 2. We employ CIOQ architecture. Queues based on per-session exist in all input and output ports. The queues in input ports are organized as multiple virtual output queues (VOQs) according to their output directions. Thus HOL problem can be resolved effectively.

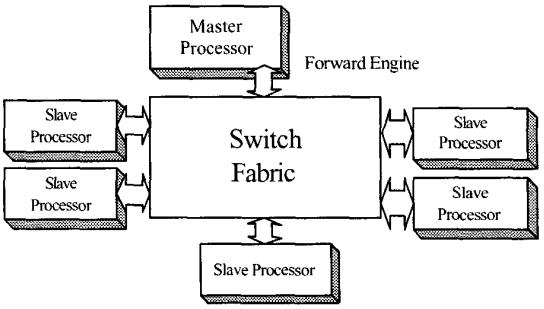

Figure 1 Architecture of high-performance and security router

In a system with N ports, there are N schedulers in every input port, corresponding to a VOQ respectively. It's responsibility to schedule and forward the several sessions in the VOQ to proper output ports. Every scheduler uses the same scheduling algorithm to guarantee the fairness of service. Switch fabric sorts and selects the scheduling results of the N*N schedulers. It searches for proper matching of N inputs to N outputs, and puts them into the per-session queues on the output ports. Scheduling in output ports is the general output-queuing scheduling. There is a scheduler for each output port to schedule packets out. Our schedulers employ a kind of PFQ. The N schedulers in input port i , denoted as PFQ(i,j), 1=i=N, 1=j=N. The scheduler in output port j is denoted as PFQj, 1=j=N.

IV. Packet Scheduling Algorithm and Feedback Mechanism

A. Packet Scheduling Algorithm

In the original admission control, the real rate allocated for flow f r_f equals to R_f, the rate that flow f required to allocate. But r_f can change as the change of the feedback received. Detailed discussion will be shown later. Each flow has a start and a finish time tag, and every scheduler schedules packets according to their tags. The algorithm we implemented is WF^2Q+ [6].

For PFQ(i,j), the system virtual time $V_{i,j}(t)$ is computed as (3), $V_{i,j}(0)=0$, $F_{i,j,f}(0)=0$, and the start and the finish time tag of every session is calculated respectively according to (4)(5). Originally $V_{i,j}(t)=0$, and so are the start and the finish time tags of every session. Updating of virtual time and start time tag of every session is only computed under the following two situations:

i) The queue of session f is empty when its packets come, then compute new $S_{i,j,f}(t)$ according to (4-1), here $V_{i,j,f}(t)$ used is the new value computed based on (3), the new finish time tag can be gotten too.

ii) After selecting a packet p_f^k of session f, compute the new start time tag according to (4-2) and (5). If there are still

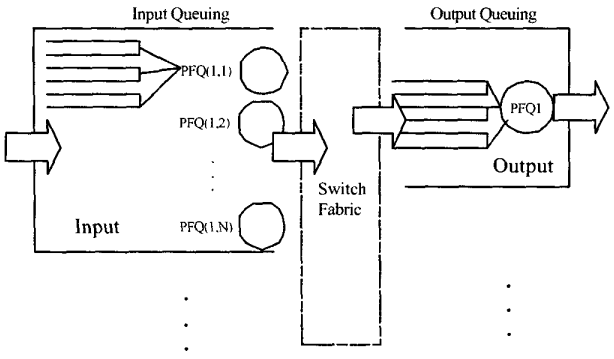

Figure 2 Logical Structure of Forwarding and Scheduling Mechanism

packets in the queue, new finish time tag should be computed again.

Then the packet with the minimum finish time flag is selected. The same algorithm is used in the scheduler PFQj in the output ports.

There is a lot of research on the algorithms used in switch fabrics. [7] proposed a scheduling algorithm called as Joined Preferred Matching (JPM). It can cooperate with the schedulers in the input ports and simulate performance of output queuing effectively. In [8] a kind of heuristic scheduling algorithm which tracks and simulates fluid policies is proposed. It provides guarantee of QoS, such as time delay, by detecting Critical Node and matching between the nodes. These algorithms can be taken into our switch fabric according to the practical requirements, and cooperate with the schedulers fore-and-aft. Considered the factors such as our crossbar-scheduling chip, we used JPM.

B. Feedback Mechanism

We designed a kind of congestion feedback mechanism based on per-session detection periodically, to avoid the occurrence of congestion effectively. The principle of the feedback mechanism is as follows. For output port i, when the input rate of flow f to that port (denoted as r_{fin_i}) is greater than the output rate (denoted as r_{fout_i}), the number of the items in the queue will always increase, and congestion may occur. According to current input and output rate to that queue, the time, $t_{f\_full\_i}$, when the queue will be full can be estimated. If the value is less than some predetermined threshold values, the probability of the congestion occurrence will be very large, and it is needed to send the forecasting information of the congestion to the scheduler in the input port. Schedulers of input ports can adjust the real allocated rate r_f properly to slow down the input rate to that output port and avoid congestion.

Then we will discuss the implementation of the feedback mechanism. Let T is the detected time interval, t_{max} and t_{min} are the threshold values of congestion.

Step 1: Estimat r_{fin_i}, the input rate of session f to output port i. In fact, it is the average rate in N time intervals. The output queue of every flow records the total length of the packets arrived in the foregone N time intervals, which is updated as the enqueuing of every packet and the coming of every time interval.

Step 2: Estimate r_{fout_i}, the output rate of flow f in ouput port i. Similarly to the above input rate, it is also the average rate in N time intervals and is updated as the dequeueing of every packet and the coming of every time interval.

Step 3: Forecast congestion and send feedback information. When $r_{fin\_i} > r_{fout\_i}$, congestion may happen. Estimate the time value $t_{f\_full\_i}$ when the queue will be full, using the following formula:

$$t_{f\_full\_i} = \frac{n}{r_{fin\_i} - r_{fout\_i}}$$

here n is the number of free items in the queue

When $t_{f\_full\_i} > t_{max}$, feedback needn't be sent for the probability of congestion occurrence is very small.

When $t_{min} < t_{f\_full\_i} = t_{max}$, general congestion is forecasted, and usual feedback will be sent.

When $t_{full\_i} = t_{min}$, congestion is about to happen, and emergent feedback should be transmitted.

The feedback is only sent to the input ports where the input queue of flow f exists.

Step 4: Process of feedback in the input ports.

To usual feedback, real allocated rate r_f of flow f is slowed down, r_f? ar_f. The Decreasing Factor $a=r_{fout\_i} / r_{fin\_i}$ until received the usual feedback from the same output port more then m times.

To emergent feedback, r_f is decreased more heavily. r_f? br_f, Decreasing Factor $b = r_{fout\_i} / cr_{fin\_i}$, $c>1$.

When no feedback is received during k time intervals, r_f should increase gradually, r_f? dr_f, increasing factor $d>1$, until reaching the requested allocated rate R_f.

Note that in this mechanism, there are several constants. time interval T, threshold value t_{max}, t_{min}, the c in the decreasing factor b, the m using for judging if the feedback is valid, the k using for judging if the r_f should be increased and the increasing factor d.

All the constants have great influence on the performance. We determined them by first using experiential value and then verified through experiments.

V. PERFORMANCE TESTING AND ANALYZING

A. Introduction of Experimental Environment

Our experimental environment is shown as figure 3. In the central place is our high-performance router archetypal system, which is based on Mortorola MPC750 with multiple 100Mbps and two 1000Mbps Ethernet ports. In addition, four PCs are used as input and output nodes, connecting to the central router by 100Mbps Ethernet network.

B. Experiments and Performance Analyzing

In the experiment, we constructed several flows requiring forwarded to the other three nodes. We observed the situation in node 1 mainly. Table 1 is the parameters of the several

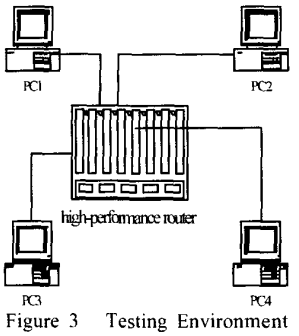

Figure 3 Testing Environment

groups of flows orientating node 1.

We can see that the real transmitting rate of flows 100~103, 120~123 is greater than their reserved rate. If the rate is holding, congestion in the output ports will occur inevitably. The results prove that too.

For limiting of paper length, we only presented partial results, as figure 4. And the figure is plot only a period of the whole test procedure. The whole curve is periodical. In the figure, (a)(b) is in the situation of no feedback.. In (a) output queue is smaller, only 100Kbytes, while in (b) the output queue is larger, 1Mbytes. (c) is the situation with feedback.

From figure 4, we can see our DF^{2}Q can simulate QoS requirements of output queuing, and has little influence on bandwidth allocation. Packet forwarding delay in the feedback mechanism will not increase much when system resources conflicts occur, while congestion will happen inevitably in the same situation if feedback doesn't exist, and that will influence forwarding performance greatly: packet loss increasing, network performance decreasing. Increasing system resources (packet queue length) can avoid congestion partially, but resource increasing isn't unlimited. It can also affect system expansibility. However feedback mechanism is a kind of method for congestion avoidance, it decreases the danger of congestion occurrence, then avoids additional overhead for processing congestion, and decreases packet delay too. So the performance of DF^{2}Q is better than the common PFQs. And we can see that the additional overhead of feedback is small enough not to affect the whole system performance.

VI. Further Discussion

In the distributed architecture, packet scheduling can be divided into three steps: (1)In input ports, N schedulers work parallelly, using the same type of PFQs to calculate and select packets; (2)The input ports of switch fabric sort the output packets of the schedulers first, find the best matching of input and output ports, and send them to the queues in the output ports; (3) Schedulers in output ports are output-queuing schedulers. They select proper packets to transmit out.

If feedback mechanism works, packet scheduler in the input port that received feedback information slows down the real allocated rate r_f of flow f whose congestion may occur ($r_f=a\,r_f=(r_{fout\_i} / r_{fin\_i})r_f$, $a=(r_{fout\_i} / r_{fin\_i})<1$), then calculate new finish time tag $F_{i,j,f}(t)$ according to its starting time tag $S_{i,j,f}(t)$ and formula(4), and re-sort the finish time tags of the flows. From formula (4) and (5) we can see that if r_f is slowed down, the finish time of flow f $F_{i,j,f}(t)$ and the new starting time $S_{i,j,f}(t)$ when a packet of that flow is scheduled will both increase. Packet scheduler PFQ(i,j) selects the packet with the minimum finish time, thus the probability of selecting packets of flow f will descend. And in fact the rate of flow f transmitting to that output port is slowed down, then the probability of the congestion occurrence of flow f is decreased.

Table 1 Flow Parameters

Flow ID	Flow Type	Resv. Rate	Arriving rate	Packet length	Delay bound	Inputting node
100~103	CBR	10	20	1200	N/A	3
105~108	On /Off	10	10	1200	1	4
110~113	CBR	1	1	1200	10	3
120~123	Poisson	2	6	520	N/A	2
130~133	CBR	100	0.1	520	8	2

We designed two kind of feedback information to adapt to different practical situation. For the common congestion situation detected, its probability can be decreased using the method discussed above. For urgent congestion detected, larger adjustment to the real allocated rate of that flow is needed to avoid the occurring congestion effectively.

Using feedback can optimize resource utilization, and avoid resource wasting when congestion occurs. But the problem of expansibility exists for feedback mechanism needs detecting mechanism and communicating between the nodes in system. Extra cost should be little enough to keep the expansibility advantage of distributed architecture. Our feedback is based on per-flow detection. The output port only sent feedback to the input ports that flow f exists when it detected congestion of flow f may occur. Schedulers in input ports only decrease the real allocated rate of flow f. Thus the feedback information of output ports to input ports is minimized, the extra overhead can be accepted completely. So the good expansibility of distributed architecture is kept well, the interference between distinct flows is isolated effectively, avoiding the bad influence of bad-action flows to good-action flows, and the fairness of fair queue scheduling algorithms can be supported better.

VII. Summarization

Data forwarding efficiency embodies the performance of router mostly, but congestion will descend forwarding performance greatly. In this paper we discussed a kind of packet scheduling algorithm in distributed router architecture. The main contribute is the presentation of a kind of congestion controlling algorithm based on per-flow detection

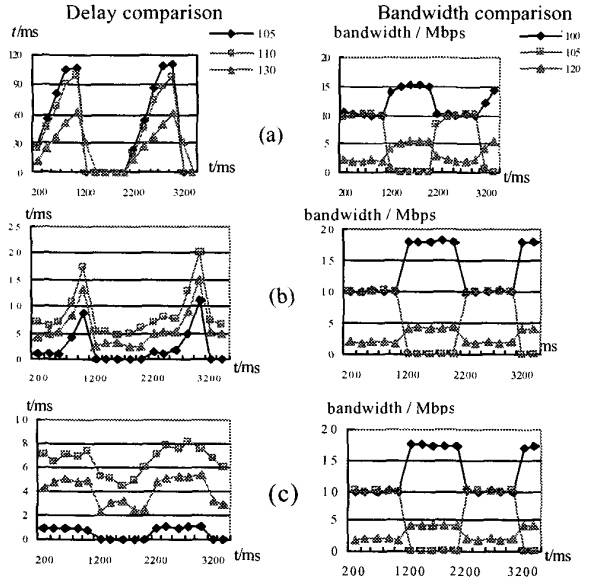

Figure 4 results of performance tesing

and feedback. It forecasts the probability of congestion of each flow on the basis of periodical detecting, then according to the results send feedback to the input ports of that flow. The input ports slow down the real allocated rate of that flow by some methods, decrease the rate to the output port where congestion will happen probably, and avoid congestion effectively.

In current research area of differential services, there is much research on feedback mechanism ([9]). Those mechanisms are mostly focus on the feedback between nodes in the network. Internal nodes in the network send feedback to border nodes, and border nodes control the traffic into the area. Compared to our feedback mechanism, they are more macroscopical, and should be ultimate resolvents of congestion. But it is clearly that in that situation cooperating between multiple nodes is needed, and the control between nodes is also very complicated, the communicating traffic is larger, and so its support to expansibility is very limited.

For the research of scheduling algorithms itself, we consider there are many points worthy of studying. For example:

(1) Scheduling algorithms showed nowadays only support unicast transferring mode. Multicast services must be the next target. In integrated services networks how to support QoS in the case of multicast by using effective scheduling algorithms is a valuable research direction .

(2) Interference between items of the scheduling algorithms and the implementing model of the integrated services. Today, there's research on combining scheduling algorithms and hierarchical link sharing. But to other parts, such as QoS routing and admitting control based on measuring, they are still seldom. Resolving the problems on that research direction may be more significative for the ultimate integrated services networks.

REFERENCES

[1] S.Chuang, A.Goel, N.McKeown, B.Prabhakar Matching output queueing with a combined input output queued switch. *Proceedings of INFOCOM '99*, p.1169-1178, IEEE, April 1999

[2] Donpaul C.Stephens, Hui Zhang, Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture. *Proceedings of INFOCOM '98*, p.282-290, IEEE, March 1998

[3] Hui Zhang, Service disciplines for guaranteed performance service in packet-switching *networks. Proceedings of the IEEE*, 1995. 83(10): 1374-1399

[4] P.Goyal, H.M.Vin, H.Chen. Start-time Fair Queuing: A scheduling algorithm for integrated services. *Proceedings of the ACM-SIGCOMM '96*, p. 157-168, Palo Alto, CA, August 1996

[5] S.Golestani, A self-clocked fair queuing scheme for broadband applications. Proceedings *of IEEE INFOCOM '94*, p.636-646, June 1994

[6] J.C.R. Bennett and H.Zhang, WF^2Q: *Worst*-case fair weighted fair queuing, *Proceedings of ACM-SIGCOMM '96*, p.143-156, Palo Alto, CA, August 1996

[7] Ion Stoica, Hui Zhang, Exact *Emulation* of an Output Queuing Switch by a Combined Input Output Queuing Switch, *IWQoS '98*

[8] V. Tabatabaee, L.Georgiadis, L.Tassiulas, Qos Provisioning and Tracking Fluid Policies in Input *Queuing* Switches, *Proceedings of INFOCOM '2000*, IEEE, March 2000

[9] H. Chow,and A. Leon-Garcia. A *Feedback* Control Extension to Differentiated Services. *IETF Internet Draft* <draft-chow-diffserv-fbctrl-00.txt>, March 1999.

[10] M.J.Karol, M.G.Hluchyj, and S.P.Morgan, Input *versus* output queuing on a space division packet switch, *IEEE Transactions on Communications*, vol.35, p.1347-56, December 1987

[11] A.Demers, S.Keshav, and S.Shenker, Analysis and simulation of a fair queue algorithm, Internetworking *Research and Experience*, vol.1, No.1, p.3-26, 1990

[12] L.Zhang, VirtualClock: a new traffic control algorithm for packet switching networks, Proceedings *of the IEEE,* vol.83, p.1374-96, May 1991

[13] A.K.Parekh and R.G.Gallager, A generalized processor sharing approach to flow control- the single *node* case, *Proceedings of INFOCOM '92*, vol.2, p.915-924, May 1992

[14] N. McKeown, M.Izzard, A.Mekkittikul, W.*Ellersick*, and M.Horowitz, The Tiny *Tera*: A Packet Switch Core, *IEEE Micro,* p.26-33, January 1997

[15] Digital Equipment Corporation, GIGAswitch, www.networks.digital.com

[16] N.McKeown; V. Anantharam; and J. Walrand, Achieving 100% Throughput in an input-queued switch *INFOCOM '96*

[17] T.Anderson, S.Owicki, J.Saxe, and C.Thacker. High speed switch scheduling for local area *networks. Proceedings of the* International Conference *on* Architectural *Support for Programming languages and Operating* Systems, Boston, MA, October 1992