Fair Scheduling in Input-Queued Switches

Hongwei Ma, Jingguo Ge, Hualin Qian

Abstract— Input-queued switch architecture has become attractive in the context of high performance networking because the switching fabric and the buffer need not run at a multiple (speedup) of the external link’s rate. With the expanding of network applications, the networks should provide QoS guarantees and QoS control is becoming a major issue in the design of high-performance switches. The scheduling in input-queued switches should give attention to the efficiency and fairness of resource allocation. In this paper, we put forward an iterative matching algorithm (P-FQ) and evaluate its performance by simulations. The simulation results show that P-FQ supports fair bandwidth distribution among the flows and can achieve asymptotically 100 percent throughput.

Index Terms-- fair scheduling, input-queueing, QoS, resource reservation.

I. INTRODUCTION

The rapid growth of the Internet is creating an insatiable demand for bandwidth growth and the large scale deployment of high-speed technologies. The deployment of Dense Wavelength Division Multiplexing (DWDM) technologies has expanded fiber transmission capacities to an extent that electronic routers remain as the main bandwidth bottlenecks [1]. At the same time, the emergence of high-speed networks provides new opportunities for new applications on the Internet, such as video conferencing, visualized computing and medical imaging. The Internet has been becoming a universal infrastructure of information exchanging.

Various applications mean diversified requirements for quality of service (QoS). QoS refers to the capability of a network to provide better service to selected network traffic over various technologies [13]. QoS is often described with some parameters such as throughput, end-to-end delay, delay jitter and packet loss ratio. At present, the Internet has only one service model--- best-effort, so it can not provide differentiated services to diverse applications and guarantee the QoS. The key factors to provide QoS control in the Internet are traffic differentiating and allocating resources fairly. None but divides traffic into various classes and distinctively, fair schedules network resources the QoS can not be guaranteed.

QoS control is becoming a major issue in the design of high-performance switching/routing devices [2, 3, and 4]. Fairness in resource allocation is very important for the network to support QoS control. Fair queuing algorithms have been developed to schedule packets at an output link of a switch and [5] is an excellent survey. However, little research has been done to address fair scheduling issues in the context of input-queued switching. Although the main function of the switch is forwarding packets from input side to output side, some measures should be adopted in order to implement fairness scheduling. The main purpose of this paper is to address the fair scheduling in input-queued switches.

The term scheduling algorithm for switching architecture is used in the literature for two different types of schedulers: switching matrix schedulers and flow-level schedulers [6]. Switching matrix schedulers decide which input interface is enabled to transmit in an input-queued switch; they avoid blocking and solve contentions within the switching fabric. Flow level schedulers decide which data flows must be served in accordance to QoS requirements. In this paper the term scheduling algorithm is only used to refer to the first class of algorithms.

High throughput and fairness are contradictory goals in scheduling algorithms design. In the context of input-queued switches, only a single cell can be transmitted from each input of the crossbar in a given slot, cells forwarded based on a maximum set of input-output match may not coincide with those satisfying fairness. On the other hand, passing cells based on fair resource allocation may not produce high throughput and may give rise to underutilization of the crossbar switch [7]. As a result, some form of tradeoff between throughput and fairness should be exploited.

The aim of this paper is to find scheduling techniques that are both fair and highly efficient for link utilization. We propose a fair scheduling algorithm (P-FQ) which provides high throughput, low latency, as well as fair bandwidth distribution among the contesting flows. The basic idea of P-FQ is picking a cell from each input in a round-robin fashion and the number of time that an input identifier appears in the circular list is proportional to reservation. In this way, we are able to allocate output link bandwidth to various flows in proportion to their reservation and prevent misbehaving flows from taking advantage at the expense of others.

The rest of the paper is organized as follows: Section 2 gives the overview of related works on switch architecture and scheduling algorithms. In Section 3, we propose the fair scheduling algorithm, namely P-FQ. Extensive simulation results are presented in Section 4. Finally, Section 5 concludes this paper.

The authors are with Computer Networks Information Center, Chinese Academy of Sciences, Beijing 100080, China(telephone: +86 010-62537785, e-mail: mhw, g.jg@cestnet.net.cn)

II. RELATED WORKS

Our goal is to develop a fair scheduling algorithm for input-queued switches that are both high performance and fairness in resource allocation. In this section, we briefly introduce some aspects that are related to switching scheduling such as switch architecture and switching algorithm appeared in the literatures.

A. Switch Architecture

The purpose of installing buffers in the switch is to temporarily store the cells that can not be transferred from output link due to resource contesting. Because of the unscheduled nature of traffic arrivals, multiple cells may simultaneously arrive on different inputs destined for the same output and they contend not only output link but also switching fabric. Only one cell can be forwarded at any time and the rest cells should be queued for later transmissions instead of being discarded.

Switches can be divided into input-queued switches (IQ), output-queued switches (OQ) and combined input output queued switches (CIOQ) according to the buffer position in the switch. FIFO queues are subject to head-of-line (HOL) blocking which severely limits switches performance. Karol et al. proved with an asymptotic analysis that under source- and destination-uniform traffic, HOL blocking limits the maximum throughput of IQ switches with an arbitrarily large number of input/output lines to 58.6%\cite{8}. Consequently, traditional switches usually adopt output-queueing scheme. By right of the speed of buffer and switching fabric, all cells arrived from different inputs can pass through the switching fabric in one cell slot, even though they are destined for the same output. Due to the limits of output link transmission rate, the cells that can not be transmitted immediately are stored in the output queues. Intuitually, the output link can not be idle unless there are no cell arriving and the throughput of switch can reach to 100%. At the same time, queueing in the output sides suits flow-level algorithms’ convenience of guaranteeing QoS for flows. However, the output-queued switches suffer from poor scalability. The reason is that the switching fabric and output port including queues have to operate N time faster than the input in order to accommodate requests on all possible inputs, where N is the number of input/output ports of the switch. In the context of high-speed networking, output-queued switches are very difficult to be implemented.

In order to keep up with the advances of transmission technologies, more and more attentions are paid to the input-queueing. In an IO switch, the input/output ports and the switching fabric can run at the same rate as that of the input link. By means of virtual output queuing (VOQ) \cite{9,10,11}: in each interface card, input buffers are organized into a set of queues, each queue storing cells directed toward a specific output, HOL blocking can be eliminated from IO switches. Actually, no need of speedup is at the expense of installing more queues in input cards.

B. Scheduling Algorithms

Assume an IQ switch has N input ports and N output ports. There are L(\(i,j\)) cells in the queue for output \(j\) in input \(i\), denoted with \(Q_{i,j}\). The task of scheduling algorithm is to determine which inputs will transmit a cell to which outputs in a given cell slot. The switch scheduling is simply an application of bipartite graph matching--each output must be paired with at most one input that has a cell destined for that output. Most existing works on scheduling for input queued switch attempt to achieve high throughput by looking for maximum bipartite matching between inputs and outputs. Schemes, such as PIM \cite{9} and iSLIP\cite{12} repeatedly search for matches at each time of scheduling. An iterative scheduling algorithm essentially consists of three major steps in each iteration, as illustrated in Fig. 1.

![Schedule_Algs](image)

Fig.1 Three-Step Matching Procedure

1) **Request**: Each unmatched input sends requests to the outputs for which it has cells for.

2) **Grant**: Each unmatched output chooses one from received requests and sends a grant signal to the corresponding input.

3) **Accept**: Each unmatched input chooses one from received grants and send a accept signal to the output which offers the grant.

The main differences between various iterative matching algorithms are the policies used in selecting one from received grant/accept signals by inputs/outputs. PIM is the first iterative matching algorithm. At the grant stage, an output selects one from received requests randomly and sends the grant signal to the corresponding input. At accept stage, an input uses the same policy to select an output and sends the accept signal. Using randomness comes with its problems \cite{7,12}, however. It is difficult and expensive to implement at high speed because each arbiter must make a random selection among the members of a time-varying set. Moreover, when the switch is oversubscribed, PIM can lead to unfairness between connections or flows.

The Basic Round-Robin Matching (RRM)\cite{12} uses rotating priority arbitration to schedule active inputs and outputs in turn. At the grant stage, the output chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented (modulo N)
to one location beyond the granted input. At the accept stage, the input uses similar methods to choose an output and update the highest priority pointer. The major advantages of RRM are that it is simple to implement. However, at the grant stage, the highest priority pointer of output is updated before the accept stage. As a consequence, whether or not the grant is accepted, the high priority pointer will point to another input and this can lead to pointers synchronization. The synchronization phenomenon leads to a maximum throughput of just 50% [12].

The iSLIP improves the output highest priority pointer updating policy, namely; only when the grant is accept by an input, the corresponding pointer points to the next input. When an output sends a grant signal that is not accepted by the input, its highest pointer does not change. Such a simple improvement of pointer updating policy leads to throughput of 100% [12].

However, the issue of fair resource allocation is not considered in the above-mentioned approaches. At the grant stage of RRM and iSLIP, the output arbiters treat each input equally. Consequently, bandwidth of output link is distributed equally among the active inputs. What should be mentioned is that equality does not mean fairness. As mentioned above, QoS is the capability of a network to provide better service to selected network traffic [13].

Statistical Matching (SM) [9] is a scheduling algorithm that does address bandwidth fair allocation. In SM, each iteration is initiated by the outputs. Each output sends the grant signal to a randomly selected input based on the reserved proportion of bandwidth. Due to lacking of knowledge of the queues occupancy, output may pick an input whose queue is empty and the switch is poorly utilized. WPIM [14] is based upon PIM. Based on the reservation, every input is assigned a quota that can be used in a frame of a constant number of cell slots. During each frame, inputs that have not reached their quotas secure an equal share of bandwidth by random selection, as is done in PIM. To accomplish this, an additional masking stage is added to the 3-step procedure to exclude those inputs that have consumed their quotas in the current frame. WPIM can satisfy the well-behaving inputs’ bandwidth requirements by limiting the number of cells that each input can send in a frame. However, the bandwidth of output link is distributed equally among the inputs that do not consume their quotas and it does not accord with the fairness definition in [15]. Simulation results of [7] prove that.

In iFS [7], a fair queueing engine is maintained for each output link, which assigns a virtual time to every incoming cell based on bandwidth reservation of the flow. In each iteration, each unmatched output independently sends a grant signal to one of the inputs whose cell is with the minimal virtual time and each unmatched input selects one of received grants that the corresponding cell is with the oldest age. That is, in the accept stage, an input resolves contention on a first-come-first-serve (FCFS) basis. It can be seen that iFS uses for reference to the idea of fair queueing used in flow-level scheduling and iFS can accomplish fair bandwidth allocating. The disadvantage of iFS is that a fair queueing engine must be maintained which leads to heavy costs.

III. THE FAIR SCHEDULING ALGORITHM--P-FQ

Admission control and bandwidth control. When a switch receives a new connecting request, it evaluates the utilization of resources, such as buffers, bandwidth. If the switch has enough resources, the new request is accepted, otherwise the request is rejected. For the purpose of centralizing our discussion of fair scheduling algorithm, we assume that all traffics in the switch are admissible.

P-FQ is an iterative matching algorithm, like iSLIP and iFS. In each iteration, output arbiters employ rotation priority arbitration to schedule the active inputs at the grant stage; each output resolves the contention on the FCFS basis. Each output arbiter maintains a circular list (also called as priority wheel in [12]) which has LC elements. LC is larger than N and usually is multiple times of N. Each element in the list is identifier of the input (e.g., port number). There is a priority pointer which always points to the highest priority list element in each output arbiter.

Output arbiter of P-FQ differs from that of iSLIP in the number of elements in the circular list. The input i’s identifier appears \( P_{i,j} \) times in the list of output j’s arbiter, where \( P_{i,j} \) is proportional to the reserved bandwidth on output link j. If an identifier appears more than once time in the list, the identifier should be symmetrically distributed in the list in order that the output arbiter can not grant the same input continuously at grant stage under heavy traffic. Let \( f(i,j) \) denote the jth flow from input i which destined for the output j and it reserves a bandwidth of \( R_{i,j} \). Assume the capacity of output link j is \( C_j \). We can calculate \( P_{i,j} \) as follows:

\[
P_{i,j} = \left \lfloor \frac{R_{i,j} \times LC}{\sum_{j} R_{i,j}} \right \rfloor
\]

The elements number in a list, namely, list length, is in inverse proportional to the allocation unit of output link bandwidth. In P-FQ, if flow (i, j) has reserved some bandwidth on the output link j, then, the input i’s identifier appears at least once in the list of output arbiter j. So the allocation unit of output link bandwidth is \( C_j/LC_j \). The bigger the \( LC_j \) is, the smaller the allocation unit can be achieved. On the one hand, with a small allocation unit, the output link bandwidth can be utilized more efficiently; on the other hand, more memory spaces should be used to store the circular list and management of the circular list is more complicated. Tradeoff in the list length and allocation unit of bandwidth should be made.

In summary, the P-FQ can be formalized as the following:

- **Initialization**
  Each output arbiter computes the times that each input identifier appears in its circular list according to the reservation and distributes the identifiers symmetrically in
its list. The highest priority pointer is initialized with a randomly selected integer between 0 and LC-1, where \( j \) is number of the output arbiter. All inputs and outputs are considered as unmatched.

- In each iteration:
  
  
  1. **Request:** Each unmatched input sends a request to every output for which it has a queued cell.
  
  2. **Grant:** If an unmatched output receives any requests, the arbiter chooses the one that appears next in a fixed, round-robin schedule starting from the elements pointed by the highest priority pointer. The output arbiter notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented (modulo \( LJC \)) to one location beyond the granted input if and only if the grant is accepted in Step 3.
  
  3. **Accept:** If an input receives any grants, it chooses one output to send an accept signal on the FCFS basis. The input and output are then considered as matched.

- At the end of each scheduling cycle, cells are transferred from the matched input sides to their corresponding output sides.

IV. SIMULATIONS AND RESULTS

We evaluate the P-FQ algorithm by simulations in two aspects: efficiency and fairness. Simulation results are presented to show that the P-FQ can achieve average cell delay and overall throughput close to the existing schemes, like iSLIP and iFS, at the same time, support fair bandwidth allocation.

A. Efficiency of the P-FQ

We evaluate the throughput and the average cell delay of P-FQ under i.i.d. Bernoulli traffic, where at any given slot a cell arrives with the probability determined by the offered workload. The simulation is performed on a switch with 16 inputs and 16 outputs, all running at the same speed. The switch operates on 64-octet long data units. There are 16 flows in each input, each destined for a different output.

In the uniform scenario, each input sends cells with destinations uniformly distributed among all the output. Fig. 2 shows the delay versus the offered load for iSLIP, iFS and P-FQ with the number of iterations equal to four. As shown in Fig. 2, the average cell delay for P-FQ is almost identical to iSLIP and iFS. The iSLIP is considered as the most efficient scheduling scheme for input-queued switches. Hence, P-FQ also has the potential of achieving high throughput under uniform traffic.

The traffic pattern used in the nonuniform traffic scenario is similar to that used in [4, 7]. Four of the switch ports are connected to servers and the remaining 12 switch ports to clients. Each client sends 40 percent of generated traffic to the servers and the traffic is uniformly distributed among all the servers; the remainder traffic generated by a client is uniformly distributed among the other clients. Each server sends 96 percent of its traffic to the clients and the traffic is uniformly distributed among all the clients and the remaining 4 percent to the other servers.

Fig. 3 shows the average cell delay versus offered load under nonuniform traffic. The three curves are almost overlapped together and this indicates that the performance of P-FQ in terms of average cell delay is very close to iSLIP and iFS.

B. Fair Bandwidth Allocation

In order to plot the result charitably, the simulation is performed a switch with 4 inputs and 4 outputs. Assume that each input has four flows that each goes to a different output and flows, \( f(0,0), f(1,0), f(2,0) \) and \( f(3,0) \), have reserved 10 percent, 20 percent, 30 percent and 40 percent of the bandwidth on the output link 0, respectively. Also assume that the four flows always maintain the same actual arrival rate. Others are background flows with a load of 5 percent each.

Fig. 4 shows the actual service rate of the flows that destined for output 0 with P-FQ. When the offered load is less than 27.78 percent, the service rate of each flow destined for output 0 is equal to its actual arrival rate. However, when the offered workload goes beyond 27.78 percent, the output link 0 is overloaded, since the aggregate arrival rate of flows destined for output 0 is \( 4 \times 0.85 \). The result shown in figure indicates that P-FQ can differentiate the flows according to the promised
share when the load beyond 27.78 percent and isolate the ill-behaved flows from well-behaved ones. When the offered load is between 27.78 percent and 47.1 percent, the output link 0 is undersubscribed and one or more flows’ actual arrival rate to output 0 is less than its reservation. The free bandwidth is also distributed fairly among the active flows. For example, when the offered load is 40 percent, the actual arrival rate of each flow destined for output 0 is 34 percent. Then, the actual arrival rate of flow \( f(3,0) \) is less than its reservation. The unused part is distributed to flow \( f(0,0), f(1,0) \) and \( f(2,0) \) so that they acquire a larger fraction of bandwidth than their reservations. Flow \( f(0,0) \), receives 11 percent, \( f(1,0) \) receives 22 percent and \( f(2,0) \) receives 33 percent. Such behavior also conforms to the fairness requirements in [15], which states that the unused portion should be assigned fairly to other active flows. When the load is greater than 47.1 percent, the arrival rate of each flow for output 0 all goes beyond its reservation, the service rate of each flow is equal to its reservation and the bandwidth is allocated fairly among the flows.

![Service Rate of flows with P-FQ](image)

**V. CONCLUSIONS**

We discuss scheduling algorithms for input queued switches to support fair bandwidth allocation in this paper. The switches should support QoS control mechanisms in order that the networks can satisfy different QoS requirements of different applications. Admission control and fair scheduling are essential mechanisms for network supporting QoS control. We proposed an iterative fair scheduling algorithm, namely P-FQ, capable of scheduling cells so that each flow receives bandwidth proportional to its reservation under heavy traffic. At the same time, P-FQ can achieve high performance in terms of throughput and average cell latency. Compared with IFS, P-FQ is simple to implement in the context of high performance networking and need not maintain complex fair queuing engines and GPS server.

**VI. PUBLICATION POLICY**

The submitting author is responsible for obtaining agreement of all coauthors and any consent required from sponsors before submitting a paper.

**REFERENCES**


