A QoS-aware scheduling algorithm for combined-input-crosspoint-queued switch

XU Ning 1, WANG Xue-shun

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, P. R. China
Wuhan Research Institute of Posts and Telecoms, Wuhan 430074, P. R. China
State Key Laboratory for New Optical Communication Technologies and Networks, Wuhan 430074, P. R. China

Received 4 July 2011; received in revised form 7 May 2012

Abstract: In this paper, we study the quality of service requirement and guaranteed-rate scheduling problem for combined-input-crosspoint-queued (CICQ) switches. Using crosspoint buffers between input queues and output ports, the input and output scheduling can be decoupled in CICQ switches. Here we propose a new guaranteed-rate scheduling algorithm which is divided separately into input and output scheduling. Simulation results show that our algorithm can perform better than existing scheduling schemes in both packet loss and jitter control.

Keywords: combined-input-crosspoint-queued switch; guaranteed-rate scheduling; low jitter; quality of service

CLC number: TP31  Document code: A

1 Introduction

More and more multimedia services being offered over the Internet include voice over IP (VoIP) and television over IP (IPTV). Those services are very sensitive to delay variation, and require tight quality of service (QoS) guarantees in IP routers and switches. It is known that traffic can be delivered in a network of output-queued (OQ) switches with 100% throughput and delay guarantees, using the GPS/WFQ theory developed by Parekh and Gallager [1]. However, OQ switches require an internal speedup of $O(N)$ which renders them impractical for high capacity routers; herein $N$ is for number of ports on a switch. An input queued (IQ) crossbar switch with unity speedup places the buffers at the input side, in a set of virtual output queues (VOQs). The VOQs can eliminate the head-of-line (HOL) blocking inherent in IQ switches. A scheduling algorithm is needed to determine cells in which VOQ could be transmitted through the crossbar. Packet switches with combined input and output queuing (CIOQ) [2] and with internal cross-point queues [3] have also been proposed.

The scheduling algorithms used by an IQ switch can be classified into two categories: (a) dynamic scheduling which computes new bipartite matching in every scheduling cycle dynamically without any pre-knowledge of the long-term guaranteed rate demands on the switch, and (b) guaranteed-rate (GR) reservation-based schedulers which periodically compute a sequence of $F$ matching called a transmission schedule to be used in $F$ consecutive time-slots to form a scheduling frame; herein $F$ is a constant positive integer. There are several guaranteed-rate scheduling algorithms that have been proposed for an IQ switch [4-8]. The first one is the Birkoff von-Neuman (BVN) scheme [4], in which a doubly stochastic traffic rate matrix is decomposed into a sequence of permutation matrices and associated weights. Then the...
matrices are scheduled using the GPS or WFQ algorithm. BVN decomposition results in a service lag bound of $O(N^2)$, where $N$ is for number of ports on switch, and has a time complexity of $O(N^{d+5})$ which is obviously too complex for use in applicable packet-switched IP routers. Another stochastic matrix decomposition algorithm is proposed in Ref. [6]. A doubly stochastic traffic rate matrix is quantized to have integer values and is then decomposed into a series of permutation matrices and associated weights. With a speedup $\delta$ between 1 and 2, the maximum service lag over all IO pairs is bounded by $O((N/4)(\delta/(\delta-1)))$. For a speedup $= 2$, this algorithm reduces the service lag bound to $O(N)$. Another stochastic matrix decomposition algorithm is greedy low-jitter decomposition (GLJD) [3]. The low-jitter GR traffic is constrained to be a relatively small fraction of the total traffic. The delay and jitter minimization problem is first formulated as an integer programming problem which is NP-hard. They formulate a greedy low-jitter decomposition with complexity $O(N^3)$ time. The scheduling algorithm requires a worst-case speedup of $O(\log(N))$ and it can achieve 80% throughput, with a low jitter $O(1)$ time. Most recently a new low-jitter rate guarantee scheduling scheme based upon recursive fair stochastic matrix decomposition (RFSMD) is proposed in Ref. [8]. This scheme forms a recursive matrix decomposition and maps the decomposition to a re-arrangeable switch network and finds a route through it with complexity $O(\max(N, NF))$ time.

Nowadays, with the advance of very large scale integration (VLSI) technology, a internal cross-point queues switch has become a very attractive solution for its decoupling of input and output scheduling. Although there are many scheduling algorithms for the cross-point queued switch [9-11], only a few guaranteed-rate scheduling schemes are for this architecture [12-13]. So, in this paper, we propose a new guaranteed-rate scheduling algorithm which focuses on the combined input and crosspoint buffered switch and provides better jitter performance.

2 Guaranteed-rate definitions

An $N\times N$ input and crosspoint buffered switch (CICQ) has $N$ input and $N$ output ports. Each input port has virtual output queues (VOQ) and each crosspoint has a buffer. As most research papers on crossbar switches do, we assume that all packets have a fixed size.

The time is divided into slots such that the length of each slot is equal to the transmission time of a packet. Without loss of generality, we assume that consecutive slots constitute a frame. The switching fabric is no speedup. In each time slot, the input schedulers select packets from its VOQs and send them to the crosspoint buffer, while output schedulers select packets from corresponding crosspoint buffer and send them to the output port. Each input port or output port can transfer or receive at most one packet in one time slot. Slots in each frame are numbered from 0 to $F-1$. The slot covers time interval. Every packet is assumed to arrive or depart at the beginning of a time slot. There is an associated traffic rate matrix with the switch. The GR traffic requirements for an $N \times N$ packet switch can be specified in a doubly sub-stochastic or stochastic traffic rate matrix as follows.

$$A = \begin{bmatrix}
\lambda_{0,0} & \lambda_{0,1} & \cdots & \lambda_{0,N-1} \\
\lambda_{1,0} & \lambda_{1,1} & \cdots & \lambda_{1,N-1} \\
\vdots & \vdots & \ddots & \vdots \\
\lambda_{N-1,0} & \lambda_{N-1,1} & \cdots & \lambda_{N-1,N-1}
\end{bmatrix}, \quad (1)
$$

where $\sum_{i=1}^{N-1} \lambda_{i,j} \leq 1$ and $\sum_{j=1}^{N-1} \lambda_{i,j} \leq 1$, and each traffic rate $\lambda_{i,j}$ represents the traffic reserved by the flow from input $i$ to output $j$. Based on the traffic matrix, a new quantized traffic rate matrix $R$ can be defined where each traffic rate is expressed as an integer number times the minimum quota of reserved bandwidth:

$$R = \begin{bmatrix}
R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\
R_{1,0} & R_{1,1} & \cdots & R_{1,N-1} \\
\vdots & \vdots & \ddots & \vdots \\
R_{N-1,0} & R_{N-1,1} & \cdots & R_{N-1,N-1}
\end{bmatrix}, \quad (2)
$$

where $\sum_{i=1}^{N-1} R_{i,j} \leq F$ and $\sum_{j=1}^{N-1} R_{i,j} \leq 1$.

Here, we use the definition from Ref. [12] to better illustrate the problem of GR scheduling.

Definition: A frame schedule of length $F$ is a sequence of partial or full permutation matrices (or vectors) which define the crossbar switch configurations for $F$ time slots within a frame.

Definition: The ideal inter-departure time denoted by $\text{IDT}(i,j)$ of cells in a GR flow between IO pair $(i,j)$ with quantized rate $R_{i,j}$ time-slot reservations in a frame...
of length \( F \), given a line-rate \( L \) in bytes per second and fixed sized cells of \( C \) in bytes, is given by \( \text{IIDD}(i,j) = \frac{F}{R_j} \) time-slots, each of duration \((CIL)\) in seconds.

**Definition:** The received service of a flow with a quantized guaranteed rate \( R_j \) at time slot \( t \) within a frame of length \( F \), denoted by \( S_y(t) \), equals the number of permutation matrices in time slot \( t \), where \( 1 < t < F \), and input port \( i \) is matched to output port \( j \).

**Definition:** The service lag of a flow between input port \( i \) and output port \( j \), at time-slot \( t \) within a frame of length \( F \), denoted by \( L_y(t) \), equals the difference between the requested quantized GR prorated by \( tf \), and the received service at time slot \( t \), i.e. \( L_y(t) = S_y(t) - (tf)R_y(t) \). A positive service lag denotes the case where the received service is less than the requested service, i.e. a cell arrives later than its ideal service time. A negative service lag is a service lead, where the received service exceeds the requested service, i.e. a cell arrives sooner than its ideal service time.

In many multimedia applications, delay and jitter are most important parameters to get better QoS. Higher jitter may need more playback buffer at the termination and degrade quality of voice and movie. To provide a better performance, we need a smoother scheduling scheme for the CICQ switch. Here, “smooth” means that a packet serviced for each flow is distributed evenly in the whole frame. This requires the inter-packet gap of the flow to be equal to the ideal inter-departure time \( \text{IIDD}(i,j) \) and all flows should be served evenly so that the absolute value of service lag, \( |L_y(t)| \) is at a minimum.

**3 Low jitter and guaranteed-rate scheduling algorithm**

Different from the traditional input queued crossbar switch, the CICQ switch has \( N^2 \) buffers located at each crosspoint of the crossbar as shown in Fig. 1. Every input scheduler only needs to choose one from \( N \) corresponding crosspoint buffers and every output scheduler does the same thing instead of the global IQ scheduling algorithms which select \( N \) connection from \( N^2 \) input-output pairs. Both input and output scheduling can make decisions independently, and their contention should be smoothed by the crosspoint buffers, e.g. if an input scheduler selects an output port to send a packet and is not selected by this output port, the packet sent by input scheduler will be stored in a crosspoint buffer and wait for output scheduling. This is called the decoupling of the input and output scheduling. Since the input and output scheduler can work independently with low complexity, to take advantage of the CICQ structure and achieve better jitter performance, there should be two scheduling algorithms designed for the input and output schedulers separately, unlike the former global low jitter scheduling algorithms for IQ switches with at least \( N^2 \) complexity. The jitter of output packets is decided mainly by the output scheduler, the algorithm in output ports should focus on the low jitter scheduling, while another algorithm running at input ports is used to send from VOQs of input ports to the crosspoint buffers with better service lag. This scheme can make output schedulers get better jitter performance easier, with the cooperation since the input schedulers have sent packets which are close to their ideal service time and apt to adjust delay variation.

In the input port scheduling, we create the two parameters for each packet in the VOQs. one is first available time for the cell and the other is scheduling deadline.

**Definition:** The scheduling deadline is time slot in a schedule frame. The packet should not be scheduled later than this slot. For the flow from input \( i \) to output \( j \), it’s value is given by: \( \text{SD}(i,j) = (k+1/2)\times\text{IIDD}(i,j) \).

**Definition:** The first available time is the time slot in a schedule frame after which the packet can be send to buffered crossbar. For the \( k \)-th packet of the flow from input \( i \) to output \( j \), it’s value is given by \( \text{FAT}(i,j,k) = (k-1/2)\times\text{IIDD}(i,j)+1 \). From the definition of scheduling deadline, we can find that the \( k+1 \)th packet of same flow can be calculated as \( \text{FAT}(i,j,k+1) = \text{SD}(i,j,k)+1 \).

The scheduling algorithm at the input port is...
described below.

1) When a packet arrives at the input port \( i \) with destination port \( j \), calculate its SD and FAT based on the last queued packet and the IIDT of this flow.

2) At each time slot of the scheduling frame, select the packets where FAT is equal or smaller than this slot number and mark them as eligible. Among these eligible packets, choose the one with smallest SD and send it to the crosspoint buffer. If there are multiple packets with the same smallest SD, choose them arbitrarily.

3) If it is needed by output scheduling, calculate the deviation of \( k \)-th packet from IIDT by

\[ \text{Dev}_k = |\text{Allocated slot} - k \times \text{IIDT}(i,j)| \]

and attach it to the packet, so it can be used by the output scheduler.

Here we analyze the feasibility of the input scheduling algorithm.

**Theorem 1.** In the input scheduler, a frame which has a length \( F \) where the \( k \)-th packet of flow can be successfully scheduled to satisfy the FAT and SD defined above.

**Proof.** In this algorithm, the scheduling algorithm is based on the deadline of packet in the flow. Define the sum of each row of rate matrix as \( R_i = \sum_{j=0}^{N-1} R_{i,j} \) and sum of each column as \( R_j = \sum_{i=0}^{N-1} R_{i,j} \). The frame length is chosen as \( F = \max\{\max(R_i), \max(R_j)\} \).

Because each packet requires only one time slot, the total requirement for the input scheduler \( R \) will be no more than \( \max\{\max(R_i), \max(R_j)\} \). So according to Ref. [14], for some jobs \( J_i = \{e_i, d_i\} \), where \( J_i \) requires service after its eligible \( e_i \) and before the deadline \( d_i \) during a time interval \( [1, F] \). If the total requirement of service for the flows with an eligible time more than 1 and earlier than F is no more than the interval itself, then there is a feasible EDF (earliest deadline first) scheduling scheme for such a job set. Packets in the flow can find a feasible schedule to meet the FAT and SD.

Now consider the output scheduling. During input scheduling, we achieve that each packet from input port \( i \) to output port \( j \) could be arranged between the FAT and SD of the packet itself when it is sent to the crosspoint buffer. So in output scheduling, we focus on improving the jitter performance.

Here we propose a scheduling algorithm for output ports. It is an optimal algorithm which aims at alleviating jitter.

Definition: Due time is the time slot at which the \( k \)-th packet from input \( i \) to output \( j \) should be scheduled in the smooth scheduling, and it is given by \( \text{DT}(i,j,k) = k \times \text{IIDT}(i,j) \).

In output scheduling, each output scheduler chooses a packet from the crosspoint buffers at every time slot, according to their due time. The packet with smallest due time should be sent first.

If there are conflicts, the packet with more Dev should be selected, which means choosing the smallest accumulated deviation from IIDT.

4 Experimental results

In this section, we study the performance of the proposed scheduling algorithm via simulations. All our results are for a 8x8 CICQ switch with or without internal speedup, running in a C++ base switch model simulation platform and each simulation case comprises \( 10^6 \) cells, which is long enough to get stable results. We choose the BMD [7] and GLJD [5] for comparison. The rate requirement matrix is generated arbitrarily with a constraint that the minimum quota of reserved bandwidth is a time slot and total traffic is admissible, i.e.

\[ \max\{\max(\sum_{i=0}^{N-1} \lambda_{i,j}), \max(\sum_{j=0}^{N-1} \lambda_{i,j})\} \leq 1. \]

Fig. 2 shows that our scheduling using second output scheduling scheme can achieve a very low packet loss when the traffic load approaches 100%. It is much better than GLJD, which has a distinct increase when the traffic approaches 80%. Most packet loss in our scheme is caused by outflow of crosspoint buffer which is set as F/8 in our simulation.

![Fig. 2 Results of packet loss, where GLJD is greedy low-jitter decomposition, and BMD is Boolean matrix decomposition](image)
Fig. 3 shows that the average jitter performance with the increasing of the switch load. In BMD and GLJD, the allocations of some port pairs may be concentrated in the front of the schedule table, thus producing large jitter. This result is especially remarkable when the traffic load is low, while our algorithm using the second output scheduling scheme has obviously better performance.

Fig. 3 Results of average jitter performance, where GLJD is greedy low-jitter decomposition, and BMD is Boolean matrix decomposition

Fig. 4 shows the distribution of deviation from IIDT under the proposed output scheduling scheme with the maximum admissible traffic compared with output scheduling adopted by Ref. [12]. Our optimal scheme (left picture) has distinctly better jitter distribution while the other scheme gets poor performance, for the input schedulers only guarantee the deadline.

5 Conclusion

With the fast growth of multimedia applications on Internet, there has been a lot of research work on the guaranteed-rate scheduling and CICQ switch. But there has been little research using both. We present a guaranteed rate and low jitter packet scheduling algorithm for the CICQ switch architecture. Simulation shows that our scheme has better throughput and jitter performance than the comparing schemes. More work should be done to get a smaller crosspoint buffer size and achieve a balance of packet loss and jitter performance.

References

[6] Koksal C, Gallager RG, Rohrs CE. Rate quantization and
service quality over single crossbar switches [C]. In: INFOCOM 2004: Twenty-third Annual Joint


[8] Szymanski TH. A low-jitter guaranteed-rate scheduling
algorithm for packet-switched IP routers [J]. IEEE
Transaction on Communications [ISSN 0090-6778], 2009, 57(11): 3446-3459.


scheduling algorithm for a buffered crossbar switch

combined-input-crosspoint-queued (CICQ) switches [C]. In: INFOCOM 2004: Twenty-third Annual Joint

[12] Szymanski TH. A low-jitter guaranteed-rate scheduling

[13] He SM, Sun ST, Qiang Z. On guaranteed smooth

[14] Stankovic JA, Spuri M. Deadline scheduling for real-
time systems: EDF and related algorithms [M]. New

Edited by LUO Min