# Fair Airport Scheduling Algorithms *

**Pawan Goyal** and **Harrick M. Vin**

Distributed Multimedia Computing Laboratory
Department of Computer Sciences, University of Texas at Austin
Taylor Hall 2.124, Austin, Texas 78712-1188, USA

E-mail: {pawang,vin}@cs.utexas.edu, Telephone: (512) 471-9732, Fax: (512) 471-8885
URL: http://www.cs.utexas.edu/users/dmcl

### Abstract

We design a class of Fair Airport algorithms that combines Start-time Fair Queuing with a non-work conserving algorithm in Rate Controlled Service Discipline (RCSD) class. We derive fairness and deadline guarantees for FA servers and demonstrate that by appropriately choosing an algorithm from RCSD class, algorithms that either allocate only rate or achieve separation of rate and delay allocation and are fair over Fluctuation Constrained variable rate servers can be designed. This method leads to the design of the first fair algorithm that achieves separation of rate and delay allocation. We also show that the FA class contains algorithms with different implementation complexity and performance guarantees and thus enables tradeoffs between the two. Furthermore, since FA contains algorithms that are fair over variable rate servers, they can be employed for achieving hierarchical link sharing. Finally, we demonstrate that the FA algorithms can be generalized to achieve other objectives such as prioritized fair allocation of residual bandwidth.

## 1 Introduction

Integrated services networks are required to support a variety of applications (e.g., audio and video conferencing, multimedia information retrieval, ftp, telnet, WWW, etc.) with a wide range of Quality of Service (QoS) requirements. Whereas continuous media applications such as audio and video conferencing require a network to provide QoS guarantees with respect to bandwidth, packet delay, and loss; applications such as telnet and WWW require low packet delay and loss. Throughput intensive applications like ftp, on the other hand, require network resources to be allocated such that the throughput is maximized. A network meets these requirements primarily by appropriately *scheduling* its resources.

An examination of application and network requirements in [9] revealed that a suitable packet scheduling algorithm for integrated services network should: (1) achieve low average as well as maximum delay for low throughput applications (e.g., interactive audio, telnet, etc.); (2) be fair over variable rate servers; and (3) be computationally efficient. To meet these requirements, Start-time Fair Queuing (SFQ) was developed in [9]. Though SFQ meets these requirements, the maximum delay of packets in SFQ depends on the maximum packet length of all the flows at a server. Consequently, if SFQ is employed to provide a priori bounds on packet delay of flows, the maximum packet length and the number of flows at a server would have to be estimated. This estimate may be large and hence the bounds on packet delay may be large. Hence, in networking environments where one of the main objective of a scheduling algorithm is to provide pre-specified bounds on packet delay, it may be desirable to employ:

- Scheduling algorithms that allocate only rate and have delay guarantee similar to Weighted Fair Queuing (WFQ): Several fair algorithms that have delay guarantee similar to WFQ are known (for example, WFQ [6, 13], FFQ [15], SPFQ [14], $WF^2Q$ [3], $WF^2Q+$ [2], Leap Forward Virtual Clock [16], etc.). Most of these algorithms are either unfair over variable rate servers or require the number of bits transmitted to be counted to achieve fairness over variable rate servers [10]. Since counting the number of bits transmitted may be expensive, algorithms that do not have this

*This research was supported in part by IBM Graduate Fellowship, IBM Faculty Development Award, Intel, the National Science Foundation (Research Initiation Award CCR-9409666 and CAREER award CCR-9624757), NASA, Mitsubishi Electric Research Laboratories (MERL), and Sun Microsystems Inc.

requirement for achieving fairness over variable rate servers are desirable.

- Scheduling algorithms that achieve separation of rate and delay allocation: The algorithms that have delay guarantee similar to WFQ provide low delay to low throughput flows by reserving higher rate; this may result into low utilization of network resources. Hence, algorithms that achieve separation of rate and delay allocation and provide lower maximum delay to packets of a flow without increasing the rate reservation are desirable. Though unfair algorithms that achieve separation of rate and delay allocation are known, fair algorithms that achieve the same objective are not.

Thus, efficient algorithms that either allocate only rate or achieve separation of rate and delay allocation and are fair over variable rate servers are not known. To address this limitation, we design a class of Fair Airport (FA) algorithms. An algorithm in FA class combines SFQ with a non-work conserving algorithm in Rate Controlled Service Discipline (RCSD) class [7, 18]. We derive fairness and deadline guarantees for FA servers and demonstrate that by appropriately choosing an algorithm from RCSD class, algorithms that either allocate only rate or achieve separation of rate and delay allocation and are fair over Fluctuation Constrained variable rate servers can be designed. This method leads to the design of the *first* fair algorithm that achieves separation of rate and delay allocation. We also show that the FA class contains algorithms with different implementation complexity and performance guarantees and thus enables tradeoffs between the two. Furthermore, since FA contains algorithms that are fair over variable rate servers, they can be employed for achieving hierarchical link sharing [9]. Finally, the FA algorithms can be generalized to achieve other objectives such as prioritized fair allocation of residual bandwidth.

The rest of the paper is structured as follows. We present the design of FA algorithms in Section 2 and summarize their properties in Section 3. Two instantiations of FA algorithms are presented in Section 4 and salient features of the FA algorithms are summarized in Section 5. Finally, Section 6 summarizes the results of the paper.

## 2 Design of Fair Airport Scheduling Algorithms

The class of Fair Airport (FA) scheduling algorithms is inspired by and derives its name from the class of Airport scheduling algorithms proposed in [5]. In an Airport scheduling algorithm, every packet of a flow on arrival joins a rate regulator for the flow and an Auxiliary Service Queue (ASQ) (see Figure 1). Once a packet passes through the rate regulator, it joins a Guaranteed Service Queue (GSQ) if it has not been serviced by ASQ by then. The server is work conserving and services packets from either GSQ or ASQ but gives priority to GSQ. The scheduling algorithms for GSQ and ASQ may be different.

In the class of FA scheduling algorithm we employ SFQ for ASQ and leave the GSQ scheduling algorithm unspecified. However, GSQ scheduler is required to guarantee a bound on the departure time of a packet based on its expected arrival time. Let $r_f$ be the the rate assigned to flow $f$, $p_f^j$ the $j^{th}$ packet of flow $f$, $l_f^j$ the length of $p_f^j$ and $A(p_f^j)$ the arrival time of $p_f^j$. Then the expected arrival time $p_f^j$, denoted by $EAT(p_f^j, r_f)$, is defined as:

$$EAT(p_f^j, r_f) = \max\left\{A(p_f^j), EAT(p_f^{j-1}, r_f) + \frac{l_f^{j-1}}{r_f}\right\} \tag{1}$$

where $EAT(p_f^0, r_f) + \frac{l_f^0}{r_f} = 0$. The class of FA scheduling algorithms is defined as follows:

1. On arrival, a packet of a flow joins the rate regulator for the flow and ASQ.

2. The departure time of packet $p_f^j$ from the rate controller, $L_{RC}(p_f^j)$, is given as:

$$EAT^{RC}(p_f^j, r_f) - \epsilon \leq L_{RC}(p_f^j) \leq EAT^{RC}(p_f^j, r_f) \tag{2}$$

   where $EAT^{RC}(p_f^j, r_f)$ is the expected arrival time of packet $p_f^j$ computed at the rate controller and $\epsilon$ is a positive constant that facilitates efficient implementation of the rate regulators. $EAT^{RC}(p_f^j, r_f)$ is computed using only the subsequence of packets of flow $f$ that were serviced through GSQ prior to $p_f^j$. Hence, if $p_f^{\theta(j)}$ is the packet prior to $p_f^j$ that was served via GSQ, then using (1) we get:

$$EAT^{RC}(p_f^j, r_f) = \max\{A(p_f^j), \\ EAT^{RC}(p_f^{\theta(j)}, r_f) + \frac{l_f^{\theta(j)}}{r_f}\} \tag{3}$$

3. The ASQ scheduler is SFQ which is defined as follows:

   (a) On arrival, a packet $p_f^j$ is stamped with start tag $S(p_f^j)$, computed as:

$$S(p_f^j) = \max\{v(A(p_f^j)), F(p_f^{j-1})\} \tag{4}$$

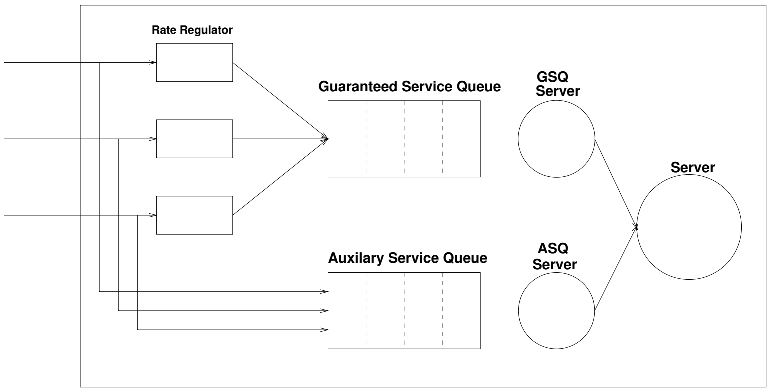


**Figure 1**: Fair Airport Scheduler

where $v(t)$ is the server virtual time and $F(p_f^j)$ is the finish tag of packet $p_f^j$. $F(p_f^j)$ is defined as:

$$F(p_f^j) = S(p_f^j) + \frac{l_f^j}{r_f} \tag{5}$$

where $F(p_f^0) = 0$ and $r_f$ is the rate of flow $f$.

(b) Initially the server virtual time is 0. During a busy period, the server virtual time at time $t$, $v(t)$, is defined to be equal to the start tag of the packet in service at time $t$. At the end of a busy period, $v(t)$ is set to the maximum of finish tag assigned to any packets that have been serviced by then.

(c) Packets are serviced in the increasing order of the start tags; ties are broken arbitrarily.

4. The GSQ scheduler is not specified. Any scheduling algorithm can be employed as long as it guarantees that the departure time of packet $p_f^j$ in GSQ, denoted by $L_{GSQ}(p_f^j)$, is:

$$L_{GSQ}(p_f^j, r_f) \leq EAT^{GSQ}(p_f^j, r_f) + \beta_f^j \tag{6}$$

where $EAT^{GSQ}(p_f^j, r_f)$ is the expected arrival time of $p_f^j$ computed at the GSQ.

5. A packet is removed from the rate regulator when it starts service in the ASQ.

6. A packet that joins GSQ is dequeued from ASQ only after it has been serviced by the GSQ server. On removal of such a packet, the start tag of the next packet of that flow in the ASQ (if such a packet exists) is set to the start tag of the packet being removed.

7. The server gives priority to GSQ over ASQ. However, it does not preempt the transmission of a packet from ASQ.

In the next section, we determine the implementation complexity and present the fairness and deadline guarantee of the FA algorithms.

## 3 Properties of Fair Airport

The properties of the class of Fair Airport algorithms are as follows:

- **Implementation complexity**: To determine the implementation complexity of a Fair Airport algorithm, consider the complexity of rate regulator and the scheduling algorithms for GSQ and ASQ:

  – Rate regulator: If a packet is to be made eligible in GSQ by time $t$, then as per (2), it can be made eligible in the interval $[t - \epsilon, t]$. Since $\epsilon$ is non-zero, the rate regulators for all the flows can be implemented by a single calendar queue with the clock tick value of $\epsilon$. The per packet complexity of calendar queue, and hence rate regulator, is $O(1)$. Observe that if $\epsilon$ was $0$, then implementing rate regulator for all the flows would require a priority queue of eligible times of the first packets of the flows which would increase the rate regulator complexity to $O(logQ)$.

  – GSQ and ASQ scheduling algorithms: SFQ requires only the packets at the head of each flow queue to be sorted. Hence, it can be implemented by using a priority queue consisting only of the packets that are at the head of each flow queue. Consequently, the per packet computational complexity is $O(logQ)$ for the ASQ scheduling algorithm. The complexity of the GSQ scheduler, on the other hand, depends on the the choice of the algorithm. If the GSQ scheduler is a static priority scheduler, then its complexity would be $O(1)$ per packet and hence the aggregate complexity would be $O(logQ)$. If the GSQ scheduler is a dynamic priority scheduler that employs a priority queue consisting only of the packets that are at the head of each flow queue, then its complexity would be $O(logQ)$ and the aggregate complexity would continue to be $O(logQ)$. An important point to observe is that if GSQ scheduler employs a priority queue, then even though there are two priority queues, every packet is not inserted into the priority queue of both SFQ and GSQ servers. If $n$ packets of a flow arrive in the flows busy period, a total of at most $n+1$ insertions occur in either of the priority queues. This is because: (1) every packet that is inserted into the priority queue of the GSQ server is served by it, and (2) due to rule 6 of the algorithm, if $k$ packets of a flow are inserted into SFQ priority queue during the flows busy period, then at least $k-1$ packets are served by SFQ server. Consequently, at most $n+1$ insertions occur.

Hence, if the GSQ scheduler per packet complexity is either $O(1)$ or $O(logQ)$, the per packet complexity of Fair Airport algorithms is $O(logQ)$.

Observe that it is possible that the expected arrival time (at the rate regulator) of all the packets that are at the head of each flow queue may be same. Consequently, $Q$ packets may join the GSQ during a single packet transmission time. Hence, in the worst-case, $O(Q)$ computation may have to be performed in a single packet transmission time. This worst-case complexity can be reduced to $O(logQ)$ by implementing the rate regulators and the GSQ and ASQ scheduling algorithms using a single augmented red-black tree data structure [1, 4, 12].

- **Fairness Guarantee**: Theorem 1 presented in Appendix A demonstrates that if the GSQ scheduler ensures (6), then in any interval $[t_1, t_2]$ in which both flows $f$ and $m$ are continuously backlogged:

$$\left| \frac{W_f(t_1,t_2)}{r_f} - \frac{W_m(t_1,t_2)}{r_m} \right| \leq 2\left(\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}\right) + (\beta_f^{max} + \beta_n^{max}) + 2\epsilon \quad (7)$$

where $W_f(t_1,t_2)$ denotes the service received by flow $f$ in $[t_1, t_2]$ and $\beta_f^{max} = \max\{\beta_f^j\}$. This fairness guarantee will hold over variable rate servers also if the GSQ scheduler ensures that (6) holds when the server rate varies. The proof of the fairness guarantee depends on the property of SFQ that it does not employ the length of a packet for determining its priority. Hence, SFQ is central to the design of Fair Airport algorithms.

- **Deadline Guarantee**: A FA server provides two types of deadline guarantees:
  - Delay guarantee: Theorem 2 presented in Appendix A demonstrates that the delay guarantee of a FA server is given as:

$$L_{FA}(p_f^j) \leq EAT(p_f^j, r_f) + \beta_f^j \quad (8)$$

    This guarantee is employed to determine bounds on various QoS guarantees for a flow [8].
  - Delay-cum-throughput guarantee: Theorem 3 presented in Appendix A demonstrates that the delay-cum-throughput guarantee of a FA server is given as:

$$L_{FA}(p_f^j) \leq \max\{L_{FA}(p_f^{j-1}), A(p_f^j)\} + \frac{l_f^{max}}{r_f} + \beta_f^j + \epsilon \quad (9)$$

    where $l_f^{max}$ is the maximum packet length of flow $f$. This guarantee is employed to improve upon the performance bounds determined from delay guarantee when the actual service received by a flow is better than that guaranteed by the server [10].

The end-to-end delay and delay-cum-throughput guarantee of a network of servers are derived in [8] and [10], respectively.

# 4 Two Instantiations of Fair Airport

In this section, we instantiate the GSQ scheduler to Virtual Clock and Delay EDD to demonstrate that FA algorithms can achieve the delay guarantee of WFQ as well as separation of rate and delay allocation, respectively.

To demonstrate that Virtual Clock and Delay EDD as GSQ schedulers satisfy (6), we need to characterize the behaviour of servers. To ensure that the fairness as well as delay guarantee of FA servers hold over variable rate servers, we characterize the server as a Fluctuation Constrained (FC) server. A FC server has two parameters; average rate $C$ (bits/s) and burstiness $\delta(C)$ (s). Intuitively, in a FC server the time taken to serve packets of aggregate length $w$ in a busy period, can exceed the time taken in an equivalent constant rate server by at most $\delta(C)$. Formally,

**Definition 1** A server is a Fluctuation Constrained (FC) server with parameters $(C, \delta(C))$, if the time taken to serve packets of aggregate length $w$ in a busy period, denoted by $T(w)$, satisfies:

$$T(w) \leq \frac{w}{C} + \delta(C) \tag{10}$$

Virtual Clock and Delay EDD and their delay guarantee when the servers are FC are as follows:

- Virtual Clock: It stamps packet $p_f^j$ on its arrival with its virtual clock value, denoted by $VC(p_f^j, r_f)$, which is defined as:

$$VC(p_f^j, r_f) = EAT(p_f^j, r_f) + \frac{l_f^j}{r_f} \tag{11}$$

It schedules packets in increasing order of virtual clock values. Theorem 4 presented in Appendix A demonstrates that if Q is the set of flows served by a FC Virtual Clock server with parameters $(C, \delta(C))$ and $\sum_{n \in Q} r_n \leq C$, then the departure time of packet $p_f^j$ at the server, denoted by $L_{VC}(p_f^j)$, is given as:

$$L_{VC}(p_f^j) \leq EAT^{GSQ}(p_f^j, r_f) + \frac{l_f^j}{r_f} + \frac{l_{max}}{C} + \delta(C) \tag{12}$$

Thus, Virtual Clock satisfies (6) with $\beta_f^j = \frac{l_f^j}{r_f} + \frac{l_{max}}{C} + \delta(C)$. Hence, using (8) we conclude that the departure time of packet in the FA server is given as:

$$L_{FA}(p_f^j) \leq EAT(p_f^j, r_f) + \frac{l_f^j}{r_f} + \frac{l_{max}}{C} + \delta(C) \tag{13}$$

This is the same delay guarantee as WFQ when $\delta(C) = 0$. Since the admission control equation requires only checking that aggregate bandwidth does not exceed the capacity, we conclude that a FA algorithm with Virtual Clock as the GSQ scheduler achieves the delay guarantee of WFQ with $O(1)$ complexity admission control algorithm.

- Delay EDD: It stamps packet $p_f^j$ on its arrival with its deadline, denoted by $D(p_f^j)$, which is defined as:

$$D(p_f^j) = EAT(p_f^j) + d_f \tag{14}$$

where $d_f$ is the deadline of flow $f$ and $l_f^j = l_f$. It schedules packets in increasing order of deadlines. Theorem 12 in [10] demonstrates that if $Q$ is the set of flows serviced by the server and

$$\forall t > 0: \quad \sum_{n \in Q} \max\left\{0, \left\lceil \frac{(t - d_n) r_n}{l_n} \right\rceil \frac{l_n}{C}\right\} \leq t \tag{15}$$

and the server is a $(C, \delta(C))$ Fluctuation Constrained Delay EDD server, then the departure time of packet $p_f^j$, denoted by $L_{EDD}(p_f^j)$, is:

$$L_{EDD}(p_f^j) \leq EAT^{GSQ}(p_f^j) + d_f + \frac{l_{max}}{C} + \delta(C) \tag{16}$$

Thus, Delay EDD satisfies (6) with $\beta_f^j = d_f + \frac{l_{max}}{C} + \delta(C)$. Hence, using (8) we conclude that the departure time of packet in the FA server is given as:

$$L_{FA}(p_f^j) \leq EAT(p_f^j, r_f) + d_f + \frac{l_{max}}{C} + \delta(C) \tag{17}$$

Since $d_f$ can be chosen to be less than $\frac{l_f}{r_f}$, equal to $\frac{l_f}{r_f}$, or more than $\frac{l_f}{r_f}$, separation of rate and delay allocation in FA with Delay EDD as the GSQ scheduler is achieved.

Due to high computational complexity, it may not be feasible to employ (15) as the schedulability test. Hence, conditions stronger than (15) that have lower computational complexity have been developed in [19]. The delay guarantee holds under the stronger conditions as well. The computational complexity of the stronger conditions is $O(Q)$.

# 5 Salient Features of Fair Airport

Some of the salient features of the class of Fair Airport algorithms are:

- As demonstrated in Section 4, it contains algorithms that have the delay guarantee of WFQ as well as those that achieve separation of rate and delay allocation.

- As demonstrated in Section 4, Virtual Clock and Delay EDD ensure (6) when the server is Fluctuation Constrained. Hence, FA contains scheduling algorithms that are fair over variable rate servers.

- Scheduling algorithms in FA that achieve fairness over variable rate servers meet a key requirement of hierarchical fair schedulers and can be employed to support hierarchical link sharing [10]. Furthermore, as Lemma 1 demonstrates, the throughput guaranteed to a flow by a FC FA server can be modeled as a FC server. Hence, analysis in [10] can be used to provide guarantees to flows when a link bandwidth is hierarchically shared using a FA algorithm.

- It contains algorithms with different implementation complexity and performance guarantees. For example, a static priority GSQ scheduler is more efficient than a Delay EDD scheduler. However, whereas the

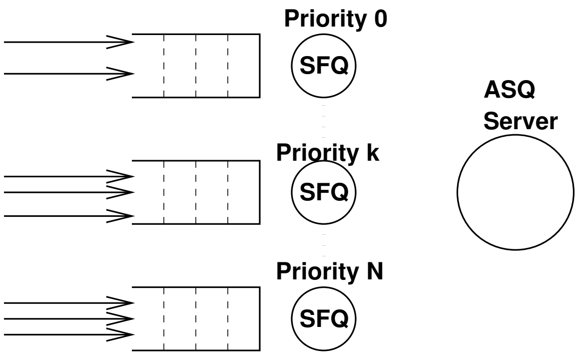


**Figure 2**: Generalization of Fair Airport ASQ scheduler to achieve prioritized fair allocation of residual bandwidth

former can support only predetermined set of delay vectors, the latter can support any [17].

- The concept of Fair Airport algorithm is general. The rate controller can be generalized as in [7] to enforce different traffic constraints. It can be shown that the generalized FA class continues to remain fair.

- The FA algorithms can be generalized to achieve prioritized fair allocation of residual bandwidth which may be desirable to support high throughput flows in max-min fair networks [11]. To achieve this objective, let the set of flows served by a server be partitioned into various priority classes. Then, FA is generalized by changing the ASQ scheduler to an algorithm that provides non-preemptive static priority to the flows of higher priority classes and schedules packets belonging to the same priority class using SFQ (see Figure 2). It can be observed from the proof of the fairness guarantee of FA algorithms that the fairness guarantee continues to hold for any two flows that belong to the same priority class. Furthermore, the deadline guarantees do not change. Thus, the generalized FA algorithm enables prioritized fair allocation of residual bandwidth without affecting the deadline guarantees.

## 6 Concluding Remarks

We designed a class of Fair Airport algorithms that combines SFQ with a non-work conserving algorithm in Rate Controlled Service Discipline class. We derived fairness and deadline guarantees of FA servers and demonstrated that by appropriately choosing an algorithm from RCSD class, algorithms that either allocate only rate or achieve separation of rate and delay allocation and are fair over FC variable rate servers can be designed. This method lead to the design of the *first* fair algorithm that achieves separation of rate and delay allocation. We also demonstrated that the FA class contains algorithms with different implementation complexity and performance guarantees and thus enables tradeoffs between the two. Furthermore, they can be employed for achieving hierarchical link sharing. Finally, we demonstrated that the FA algorithms can be generalized to achieve other objectives such as prioritized fair allocation of residual bandwidth.

## References


[1] S. K. Baruah, J. E. Gehrke, and C. G. Plaxton. Fair On-Line Scheduling of a Dynamic Set of Tasks on a SingleResource. Technical Report TR-96-03, Department of Computer Sciences, The University of Texas at Austin, February 1996.

[2] J.C.R. Bennett and H. Zhang. Hierarchical Packet Fair Queuing Algorithms. In *Proceedings of SIGCOMM'96*, pages 143–156, August 1996.

[3] J.C.R. Bennett and H. Zhang. $WF^2Q$: Worst-case Fair Weighted Fair Queuing. In *Proceedings of INFOCOM'96*, pages 120–127, March 1996.

[4] T. H. Cormen and C. E. Leiserson. *Introduction to Algorithms*. MIT Press, Cambridge MA, 1990.

[5] R.L. Cruz. Service Burstiness and Dynamic Burstiness Measures: A Framework. *Journal of High Speed Networks*, 2:105–127, 1992.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of ACM SIGCOMM*, pages 1–12, September 1989.

[7] L. Georgiadis, R. Guerin, V. Peris, and K.N. Sivarajan. Efficient Network QoS Provisioning Based on per Node Traffic Shaping. In *Proceedings of INFOCOM'96*, pages 102–110, March 1996.

[8] P. Goyal and H. M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. In *IEEE/ACM Transactions on Networking (to appear)*. Also available as technical report TR95-30, Department of Computer Sciences, The University of Texas at Austin.

[9] P. Goyal, H. M. Vin, and H. Cheng. Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. In *Proceedings of ACM SIGCOMM'96*, pages 157–168, August 1996.

[10] P. Goyal, H. M. Vin, and H. Cheng. Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. Technical Report TR-96-02, Department of Computer Sciences, The University of Texas at Austin, January 1996. Available via URL http://www.cs.utexas.edu/users/dmcl.

[11] Q. Ma, P. Steenkiste, and H. Zhang. Routing High-bandwidth Traffic in Max-min Fair Networks. In *Proceedings of SIGCOMM'96*, pages 206–217, August 1996.

[12] E. M. McCreight. Priority Search Trees. *SIAM Journal on Computing*, 14:257–276, 1985.

[13] A.K. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1992.

[14] D. Stiliadis. *Traffic Scheduling in Packet-Switched Networks: Analysis, Design and Implementation*. PhD thesis, Department of Computer Science and Engineering, University of California at Santa Cruz, 1996.

[15] D. Stiliadis and A. Varma. Design and Analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet Switched Networks. In *Proceedings of SIGMETRICS'96*, May 1996.

[16] S. Suri, G. Varghese, and G. Chandramenon. Leap Forward Virtual Clock: A New Fair Queuing Scheme with Guaranteed Delays and Throughput fairness. In *Proceedings of INFOCOM'97*, April 1997.

[17] H. Zhang and D. Ferrari. Rate Controlled Static Priority Queueing. In *Proceedings of INFOCOM'93*, volume 2, pages 227–236, 1993.

[18] H. Zhang and D. Ferrari. Rate-controlled Service Disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[19] Q. Zheng and K. Shin. On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-switching Networks. *IEEE Transactions on Communications*, 42(3):1096–1105, March 1994.

# A Proofs of Properties of Fair Airport Scheduling Algorithm

In the following sections we establish the fairness and deadline guarantee of Fair Airport algorithm.

## A.1 Fairness Guarantee

To prove fairness guarantee of Fair Airport algorithm, we first derive lower and upper bounds on the service received by a flow through GSQ in Lemmas 1 and 2, respectively. We then prove the fairness guarantee in Theorem 1. Let $W_f^{GSQ}(t_1,t_2)$ and $W_f^{ASQ}(t_1,t_2)$ be the service received by flow during $[t_1,t_2]$ from GSQ and ASQ, respectively. Let $\beta_f^{max} = \max\{\beta_f^j\}$.

**Lemma 1** If the GSQ scheduler ensures (6), then for all intervals $[t_1,t_2]$ in which the flow is backlogged throughout the interval:

$$r_f(t_2-t_1)-r_f(\beta_f^{max}+\epsilon)-l_f^{max} \leq W_f^{GSQ}(t_1,t_2) \quad (18)$$

**Proof:** Let packet $p_f^j$ refer to the $j^{th}$ packet of flow $f$ serviced by GSQ. Since flow $f$ is backlogged during $[t_1,t_2]$ and the rate regulator lets a packet into the GSQ at most $\epsilon$ before its expected arrival time, the maximum value of the expected arrival time of a packet that is served before time $t$ is $t+\epsilon$. Hence, the maximum value of the expected arrival time of the first packet that is served via GSQ in the interval $[t_1,t_2]$, denoted by $p_f^m$, is:

$$EAT^{GSQ}(p_f^m, r_f) \leq t_1 + \epsilon + \frac{l_f^{max}}{r_f} \quad (19)$$

Let packet $p_f^n$ be such that:

$$EAT^{GSQ}(p_f^n, r_f) + \beta_f^n \leq t_2 \quad (20)$$

$$t_2 \leq EAT^{GSQ}(p_f^n, r_f) + \frac{l_f^n}{r_f} + \beta_f^{n+1} \quad (21)$$

Since flow $f$ is backlogged, packet $p_f^n$ exists. Furthermore, since GSQ scheduler guarantees that packet $p_f^n$ will depart by $EAT^{GSQ}(p_f^n, r_f) + \beta_f^n$, from (20) we conclude that packet $p_f^n$ is guaranteed to be served by time $t_2$. Hence, $W_f^{GSQ}(t_1,t_2) \geq \sum_{k=0}^{k=m-n} l_f^{m+k}$. Since flow is continuously backlogged in $[t_1,t_2]$,

$$EAT^{GSQ}(p_f^n, r_f) + \frac{l_f^n}{r_f} = EAT^{GSQ}(p_f^m, r_f) + \frac{\sum_{k=0}^{k=m-n-1} l_f^{m+k}}{r_f} + \frac{l_f^n}{r_f} \quad (22)$$

Hence, from (21) we get:

$$t_2 \leq EAT^{GSQ}(p_f^m, r_f) + \frac{\sum_{k=0}^{k=m-n} l_f^{m+k}}{r_f} + \beta_f^{n+1} \quad (23)$$

$$t_2 - \beta_f^{n+1} - EAT^{GSQ}(p_f^m, r_f) \leq W_f^{GSQ}(t_1,t_2) \quad (24)$$

Using (19) and $l_f^{n+1} \leq l_f^{max}$, we get

$$(t_2 - t_1) - (\beta_f^{max} + \epsilon) - \frac{l_f^{max}}{r_f} \leq W_f^{GSQ}(t_1, t_2) \quad (25)$$

Hence the lemma follows. ■

**Lemma 2** If the GSQ scheduler ensures (6), then for any interval $[t_1, t_2]$ in which the flow is backlogged throughout the interval:

$$W_f^{GSQ}(t_1, t_2) \leq r_f(t_2 - t_1) + (\beta_f^{max} + \epsilon) + l_f^{max} \quad (26)$$

**Proof:** Let packet $p_f^j$ refer to the $j^{th}$ packet of flow $f$ serviced by GSQ. Since GSQ scheduler ensures that packet $p_f^j$ departs by $EAT^{GSQ}(p_f^j, r_f) + \beta_f^j$. Hence, the first packet to be serviced within $[t_1, t_2]$, $p_f^m$, must be such that:

$$EAT^{GSQ}(p_f^m, r_f) + \beta_f^m \geq t_1 \quad (27)$$

Since the rate regulator ensures that at all time $t$, the expected arrival time of the packets in GSQ is at most $t + \epsilon$, we know that last packet that is served in $[t_1, t_2]$, denoted by $p_f^n$, is such that:

$$EAT^{GSQ}(p_f^n, r_f) < t_2 + \epsilon \quad (28)$$

From the definition of expected arrival time, we know:

$$EAT^{GSQ}(p_f^n, r_f) - EAT^{GSQ}(p_f^m, r_f) \geq \sum_{k=0}^{k=m-n-1} \frac{l_f^{m+k}}{r_f} \quad (29)$$

$$\Rightarrow EAT^{GSQ}(p_f^n, r_f) - EAT^{GSQ}(p_f^m, r_f) + \frac{l_f^n}{r_f} \geq \sum_{k=0}^{k=m-n} \frac{l_f^{m+k}}{r_f} \quad (30)$$

Using the bounds on $EAT^{GSQ}(p_f^n, r_f)$ and $EAT^{GSQ}(p_f^m, r_f)$, we get:

$$(t_2 - t_1) + \epsilon + \beta_f^m + \frac{l_f^n}{r_f} \geq \sum_{k=0}^{k=m-n} \frac{l_f^{m+k}}{r_f} \quad (31)$$

Since $\sum_{k=0}^{k=m-n} l_f^{m+k} \geq W_f^{GSQ}(t_1, t_2)$,

$$r_f(t_2 - t_1) + r_f(\beta_f^{max} + \epsilon) + l_f^{max} \geq W_f^{GSQ}(t_1, t_2) \quad (32)$$

■

**Theorem 1** If GSQ scheduler ensure (6), then for any interval $[t_1, t_2]$ such that flows $f$ and $m$ are backlogged during the entire interval, the difference in the service received by two flows is given as:

$$\left| \frac{W_f(t_1, t_2)}{r_f} - \frac{W_m(t_1, t_2)}{r_m} \right| \leq 2\left(\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}\right) + (\beta_f^{max} + \beta_n^{max}) + 2\epsilon \quad (33)$$

**Proof:** To derive a bound on $W_f^{ASQ}(t_1, t_2)$ observe that

- Since a packet is removed from the ASQ only when it has been service either through ASQ or GSQ, whenever a flow is backlogged in the system, it is also backlogged in the ASQ.
- Whenever a packet is serviced from the GSQ, the start tag of the next packet in the ASQ is set to the start tag of the packet. Hence Lemma 1 and 2 in [9] continue to hold.

Since the flow is backlogged in in the interval $[t_1, t_2]$, from Lemmas 1 and 2 of [9], we conclude:

$$r_f(v_2 - t_1) - l_f^{max} \leq W_f^{ASQ}(t_1, t_2) \leq r_f(v_2 - v_1) + l_f^{max} \quad (34)$$

where $v_2 = v(t_2)$ and $v_1 = v(t_1)$. The theorem follows from (34) and Lemmas 1 and 2. ■

## A.2 Deadline Guarantee

Theorems 2 and 3 establish the delay and delay-cum-throughput guarantee of a FA server.

**Theorem 2** The departure time of packet $p_f^j$ in a FA server, denoted by $L_{FA}(p_f^j)$, is given as:

$$L_{FA}(p_f^j) \leq EAT(p_f^j, r_f) + +\beta_f^j \quad (35)$$

**Proof:** Consider packet $p_f^j$. It may be served through ASQ or GSQ. Let us consider the two cases:

- Packet $p_f^j$ is serviced through GSQ: Since GSQ ensures (6):

  $$EAT^{GSQ}(p_f^j, r_f) + +\beta_f^j \quad (36)$$

  It has been shown in [8] that:

  $$EAT^{GSQ}(p_f^j, r_f) = EAT^{RC}(p_f^j, r_f) \quad (37)$$

  when the rate controller satisfies (2). Since $EAT^{RC}(p_f^j, r_f) \leq EAT(p_f^j, r_f)$, from (36) and (37) we conclude that (35) holds for packet $p_f^j$.

- Packet $p_f^j$ is serviced through ASQ: Packet $p_f^j$ starts service from ASQ at time $t$ only if $t < EAT^{RC}(p_f^j, r_f)$. Hence, packet $p_f^j$ departs by $t + \beta_f^j$. Since $EAT^{RC}(p_f^j, r_f) \leq EAT(p_f^j, r_f)$, we conclude that (35) holds for packet $p_f^j$.

■

**Theorem 3** In a FA server the departure time of packet $p_f^j$ is given as:

$$L_{FA}(p_f^j) \leq \max\{L_{FA}(p_f^{j-1}), A(p_f^j)\} + \frac{l_f^{max}}{r_f} + \beta_f^j + \epsilon \tag{38}$$

where $L_{FA}(p_f^0) = 0$.

**Proof:** Let $t$ be the time at which packet $p_f^j$ becomes the first flow $f$ packet to be scheduled, i.e.,

$$t = \max\{L_{FA}(p_f^{j-1}), A(p_f^j)\} \tag{39}$$

Then, since the last packet to be scheduled through GSQ could have a maximum packet length of $l_f^{max}$ and expected time at the rate controller of at most $t + \epsilon$, using (2) we conclude:

$$EAT^{RC}(p_f^j) \leq t + \epsilon + \frac{l_f^{max}}{r_f} \tag{40}$$

Packet $p_f^j$ may be served through the GSQ or ASQ. Let the packet be served through GSQ. Using (37) we get:

$$EAT^{GSQ}(p_f^j) \leq t + \epsilon + \frac{l_f^{max}}{r_f} \tag{41}$$

Since GSQ server guarantees that $p_f^j$ will depart by $EAT^{GSQ}(p_f^j) + \beta_f^j$:

$$L_{FA}(p_f^j) \leq EAT^{GSQ}(p_f^j) + \beta_f^j \tag{42}$$

Using (41) and then (39) to substitute for $t$ yields (38). Since a packet can be served from ASQ only earlier than it is guaranteed to depart from GSQ, the theorem follows. ■

## B Delay Guarantee of FC Virtual Clock Servers

The Virtual Clock scheduling algorithm stamps packet $p_f^j$ on its arrival with its virtual clock value, denoted by $VC(p_f^j, r_f)$, which is defined as:

$$VC(p_f^j, r_f) = EAT(p_f^j, r_f) + \frac{l_f^j}{r_f} \tag{43}$$

Define rate function of flow $f$, denoted by $R_f(t)$, as:

$$R_f(t) = \begin{cases} r_f & \text{if } \exists j \ni \Big(A(p_f^j) \leq t\Big) \wedge \\ & \Big(VC(p_f^{j-1}, r_f) < t \leq VC(p_f^j, r_f)\Big) \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

Theorem 4 proves the delay guarantee of a FC Virtual Clock server.

**Theorem 4** If $Q$ is the set of flows served by a FC VC server with parameters $(C, \delta(C))$ and $\sum_{n \in Q} R_n(t) \leq C$ for all $t$, then the departure time of packet $p_f^j$ at the server, denoted by $L_{VC}(p_f^j)$, is given as:

$$L_{VC}(p_f^j) \leq VC(p_f^j, r_f) + \frac{l_{max}}{C} + \delta(C) \tag{45}$$

**Proof:** It is easily observed from (43), i.e., the definition of virtual clock value and (44) that the cumulative length of all flow $n$ packets that arrive in interval $[t_1, t_2]$ and have virtual clock value no greater than $t_2$, denoted by $AP_n(t_1, t_2)$, is given as:

$$AP_n(t_1, t_2) \leq \int_{t_1}^{t_2} R_n(t)dt$$

Consider packet $p_f^j$ served in the busy period beginning at time $t_0$. Let $t_2 = VC(p_f^j, r_f)$ and let $t_1$ be the largest time less than $t_2$ at which a packet that has virtual clock value greater than $t_2$ is scheduled in the busy period. If such a packet does not exist, set $t_1 = t_0^-$. Then, since packets are served in the increasing order of virtual clock values, all the packets scheduled in the interval $(t_1, L_{VC}(p_f^j)]$ arrive in the interval $(t_1, t_2]$ and have virtual clock value at most $t_2$. Hence, the aggregate length of packets scheduled in the interval $(t_1, L_{VC}(p_f^j)]$ is $\sum_{n \in Q} AP_n(t_1^+, t_2)$. Consequently, the aggregate length of packets served in the interval $[t_1, L_{VC}(p_f^j)]$ is at most $l_{max} + \sum_{n \in Q} AP_n(t_1^+, t_2)$. Since the server is FC, we get:

$$\begin{aligned} L_{VC}(p_f^j) &\leq t_1 + \frac{l_{max} + \sum_{n \in Q} AP_n(t_1^+, t_2)}{C} + \delta(C) \\ &\leq t_1 + \frac{\int_{t_1^+}^{t_2} \sum_{n \in Q} R_n(t)dt}{C} + \frac{l_{max}}{C} + \delta(C) \\ &\leq t_1 + \frac{C(t_2 - t_1)}{C} + \frac{l_{max}}{C} + \delta(C) \\ &\leq t_2 + \frac{l_{max}}{C} + \delta(C) \end{aligned}$$

Since $t_2 = VC(p_f^j, r_f)$, the theorem follows. ■

We observe that this proof method can also be employed to derive the delay guarantee of generalized Virtual Clock scheduling algorithm, presented in [8], for FC servers.