Implementing the Dual-rate Grouping Scheme in Cell-based Schedulers †

Dong Wei, Jie Yang, Nirwan Ansari, and Symeon Papavassiliou
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA

Abstract— The use of fluid Generalized Processor Sharing (GPS) algorithm for integrated services networks has received a lot of attention since early 1990's because of its desirable properties in terms of delay bound and service fairness. Many Packet Fair Queuing (PFQ) algorithms have been developed to approximate GPS. However, owing to their implementation complexity, it is difficult to support a large number of sessions with diverse service rates while maintaining the GPS properties. The grouping architecture has been proposed to dramatically reduce the implementation complexity. However, it can only support a fixed number of service rates, thus causing the problem of granularity. In this paper, we present a viable implementation of our recently proposed dual-rate grouping architecture, and demonstrate that, as compared with the original grouping architecture, our proposed scheme possesses better performance in terms of approximating per session-based PFQ algorithms without increasing the implementation complexity.

I. INTRODUCTION

High-speed, service-integrated packet switches are required to support a large number of sessions with diverse service rate requirements. When multiplexed at the same output of a scheduler, different sessions interact with each other, and therefore scheduling algorithms are used to control the interactions among them.

Based on an idealized fluid model, A.K. Parekh [1] proposed the Generalized Processor Sharing (GPS) algorithm, which has been proven to have three desirable properties: 1) it can guarantee the latency bound to any leaky-bucket-constrained session; 2) it can ensure fair allocation of bandwidth among all backlogged sessions; 3) it has a certain capability of immunity, i.e., it can isolate well-behaving sessions from disadvantageous effects of other misbehaving sessions. However, GPS is an idealized model and cannot be implemented in real world. Some service disciplines generally called Packet Fair Queuing (PFQ) algorithms, which differ in tradeoffs between implementation complexity and performance in terms of latency bound and service fairness, have been proposed to approximate GPS. In reality, due to the complexity, it is difficult to implement these disciplines in a scheduler to support a large number of sessions with diverse service rate requirements while maintaining all desirable GPS properties.

Implementation complexity of PFQ algorithms is determined by the following factors [2]: 1) the calculation of the system virtual time; 2) sorting the service order of all sessions; 3) the management of another priority queue to regulate packets (only if those algorithms with "smallest eligible virtual finish time first," such as WF^2Q [8] or WF^2Q+ [6], are adopted). PGPS [1] and Weighted Fair Queuing [3] use the virtual system time defined by the GPS model. Both need to track all backlogged sessions, and hence the worst case complexity is $O(N)$ where N is the number of sessions. Some other PFQ algorithms, whose virtual system time complexity are $O(1)$ [4] and $O(\log N)$ [5][6], have been developed. The sorting complexity of most algorithms is $O(\log N)$. S. Suri, *et al.*, [7] proposed to use the van Emde Boas data structure, which has the complexity of $O(\log\log N)$. H. Zhang, *et al.*, [6][8] proposed a selection policy by selecting packets among all eligible sessions. This selection policy can improve the worst-case delay for clearing the backlog of a session's queue, but it requires extra management of another priority queue.

A novel grouping architecture, which can dramatically reduce the overall complexity, has been proposed in [2]. All sessions with the same service rate stay in the same group when they are active. However, this architecture has a restriction that only a fixed number of service rates can be supported. This restriction leads to the problem of service rate granularity. Such a problem may degrade the fairness of bandwidth allocation among different sessions. We observe that if bandwidth is not allocated fairly, even though the scheduling disciplines have integrated traffic regulation capability [6][8], their immunity capability to protect any session from other sessions' negative impact will be degraded. Based on the fact that, in a packet-based scheduler, the service rate is essentially the amount of service received in a time interval, in order to achieve fair bandwidth allocation in the time interval, our principle is to provide different service rates to a session alternately [9]. With this approach, the number of service rate groups in the scheduler is not increased, and thus the implementation complexity remains the same. In this paper, we present a viable implementation of this scheme. The corresponding numerical results demonstrate that the proposed implementation improves not only the rate granularity but also the fairness of bandwidth allocation and the immunity capability.

The rest of the paper is organized as follows. In Section II, we review PFQ algorithms and the grouping architecture. In Section III, we describe the principle and implementation of the dual-rate grouping scheme. Experimental results presented in Section IV show that performance is

† This work has been supported in part by the New Jersey Commission on Science and Technology via the NJ Center for Wireless Communication, and the New Jersey Commission on Higher Education via the NJ I-TOWER project.

0-7803-7632-3/02/$17.00 ©2002 IEEE

significantly improved, with our proposed approach, in terms of approximating per session-based PFQ algorithms. Finally, concluding remarks are included in Section V.

II. BACKGROUND

PFQ algorithms have a global variable –virtual system time V(.), which is defined differently for different PFQ algorithms. They also maintain a virtual start time and a virtual finish time for each session. When the k^{th} packet of session i arrives, the virtual start time $S_i(.)$ and virtual finish time $F_i(.)$ of this packet are given as follows:

$$S_i(t) = \begin{cases} \max(V(t), F_i(t-)) & \textit{session } i \textit{ becomes active} \\ F_i(t-) & p_i^{k-1} \textit{ finished service} \end{cases} \quad (1)$$

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

where L_i^k is the size of the k^{th} packet of session i, and r_i is the required service rate of session i.

The virtual system time is updated when a packet starts to receive service [4] or new sessions become active. All PFQ algorithms have similar sorted-queue architecture; they differ in two aspects: 1) the virtual system time function; 2) packet selection policy.

WF^{2}Q+ [6] is the optimal PFQ algorithm in terms of approximating GPS and it has a desirable implementation complexity. Therefore, WF^{2}Q+ is adopted to conduct the performance analysis and simulations in this paper.

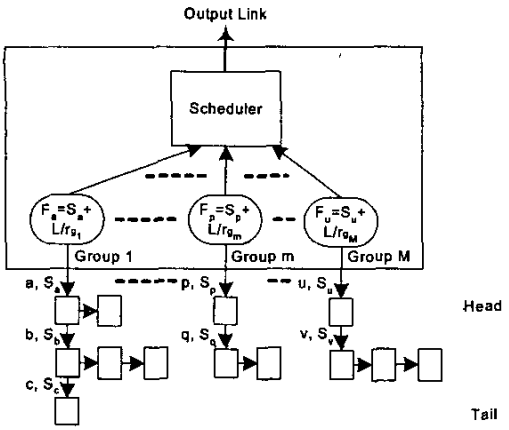

Figure 1 A cell-based scheduler with the grouping architecture.

A grouping architecture for cell-based schedulers [2], as shown in Figure 1, was presented to efficiently implement PFQ algorithms in high-speed cell-based switches. By employing the Locally Bounded Timestamp (LBT) property [2], the priority relationship among sessions in the same service group can be maintained without sorting. Let g_m denote the m^{th} service group and r_{g_m} denote the service rate of g_m. All sessions with the same service rate requirements are placed in the same group. The operation of the system can be summarized as follows:

1. When a new session, q, is set up, it is assigned to a service rate group according to its rate requirement. The service rate of the group must be no less than the requirement of session q. At this moment, the first packet of session q is placed at the tail of its service group.
2. The scheduler selects the packet with the smallest virtual finish time to transmit among all sessions in the heads of service rate groups.
3. After a session receives service, if it is still backlogged, it is placed at the tail of the service rate group; if the session is temporarily idle or finished, it is taken out of the service rate group. The next session in the same group is placed at the head.
4. When a session becomes active again, it can be treated as a new session and placed at the tail of the corresponding group.

In each group, each backlogged session is shifted one by one to the head of the group, and thus the session with the smallest virtual start time in each group is always at the head of the group. Scheduling is performed only among sessions at the head of each group. Therefore, with this grouping architecture, the worst-case algorithm complexity of scheduling and updating of the virtual system time is reduced. When WF^{2}Q+ is employed, the implementation complexity is reduced from *O(logN)* to *O(logM)*, where M is the number of service rate groups. This scheduler can be considered as a discrete version of the Rate Proportional Processor Sharing (RPPS) model [1], in which the service rates of all sessions are set according to the session group service rate proportionally.

III. THE DUAL-RATE GROUPING ALGORITHM

Consider session i with the required service rate r_i, such that $r_{g_{m-1}} < r_i \le r_{g_m}$, and assume that, in per session-based PFQ algorithm, totally K packets in session i receive service in time interval (t_1, t_2), and all packets have the same size 1. By pumping some packets into service group g_m and the rest to g_{m-1}, we try to achieve the same amount of service in time interval (t_1, t_2). If we denote by α the portion of packets allocated to service group g_m and by $(1-\alpha)$ the portion of packets allocated to service group g_{m-1}, the following relationship must hold:

$$r_i = \frac{K}{t_2 - t_1} = \frac{K}{\underbrace{\frac{\alpha K}{r_{g_m}}}_{} + \underbrace{\frac{(1-\alpha)K}{r_{g_{m-1}}}}_{}} = \frac{1}{\underbrace{\frac{\alpha}{r_{g_m}}}_{} + \underbrace{\frac{1-\alpha}{r_{g_{m-1}}}}_{}} \quad (3)$$

Thus, α can be computed from (3). Therefore, the scheduler can place packets into different service rate groups accordingly to achieve a desirable bandwidth.

Two components are introduced to implement the dual-rate grouping scheme: 1) a counter is used to record how many packets are transmitted in a periodic manner; 2) a

marker is used to indicate which service group the session should be inserted into.

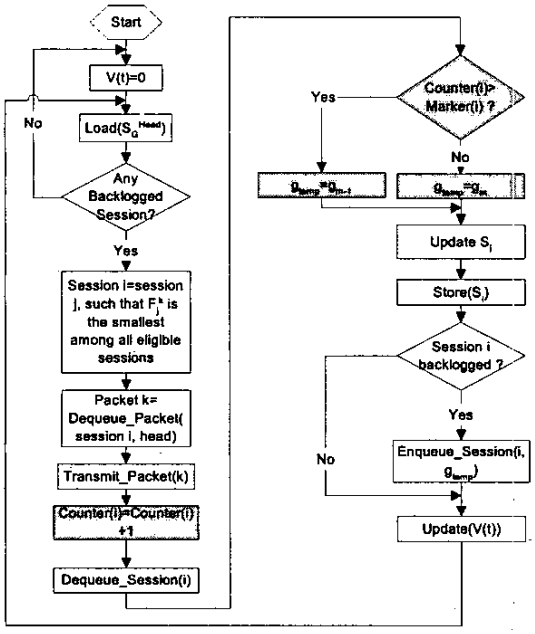

Figure 2: Scheduling algorithm in dual-rate grouping.

Figure 2 shows the flow chart of the scheduling algorithm, while Figure 3 presents the operation on new packet arrival with the dual-rate grouping. The shaded decision and processing blocks are additional operations compared with the original grouping architecture. Note that these extra operations do not increase the complexity of the PFQ algorithm and memory access in implementation. $V(t)$ can be updated as $V(t+\tau) = \max(V(t)+\tau, \min_{i \in B(t+\tau)} S_i(t+\tau))$ if WF^{2}Q+ is adopted.

The process of Enqueue_Session(i, g_{temp}) (in Figure 2) places session i at the tail of the service group g_{temp}, and Dequeue_Session(i) takes session i out of the head of its current service group.

In order to avoid sorting when a session is inserted into a service rate group, the following cases need to be considered, while updating the virtual start time of each session, to ensure the LBT property [2]:

1. Session i is backlogged, and there is no need to change to another service rate group. Then, the virtual start time $S_i(t) = F_i(t-)$.
2. Session i is active again from the idle status,an d there is no need to change to another service rate group. Then, the virtual start time $S_i(t) = \max\{S_{g_{temp}}^{Tail}(t), V(t)\}$, where $S_{g_{temp}}^{Tail}(t)$ is the virtual start time of the session at the tail of the same service rate group.
3. Session i is backlogged and needs to change to another service rate group. Then, the virtual start time $S_i(t) = \max\left\{S_{g_{temp}}^{Tail}(t), \min\{S_{g_{temp}}^{Head}(t) + \frac{L}{r_{g_{temp}}}, F_i(t-)\}\right\}$, where g_{temp} is the service rate group in which the session is inserted and $S_{g_{temp}}^{Head}(t)$ is the virtual start time of the session at the head of this service rate group.
4. Session i is active again from the idle status and needs to change to another service rate group. Then, the virtual start time is updated as $S_i(t) = \max\left\{S_{g_{temp}}^{Tail}(t), \min\{S_{g_{temp}}^{Head}(t) + \frac{L}{r_{g_{temp}}}, V(t)\}\right\}$

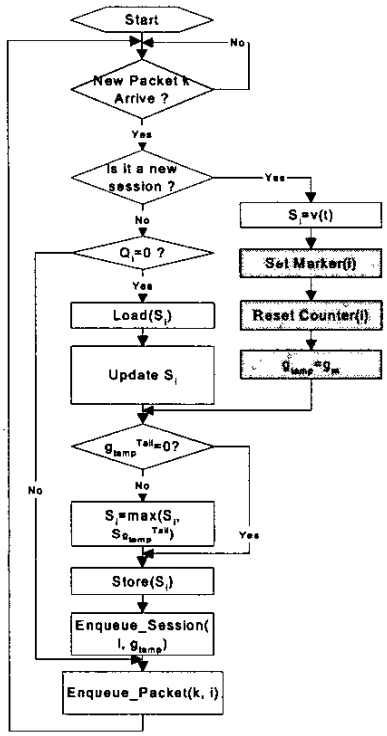

Figure 3: Operation on packet arrival in dual-rate grouping.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we provide some simulation results to evaluate the performance of our proposed scheme and compare it with the corresponding performance of the grouping architecture. To achieve this, two sets of experiments are performed. In Experiment 1, we demonstrate that, using the proposed approach, fairer bandwidth allocation can be achieved and the performance in terms of latency approximates per session-based PFQ better than the grouping architecture. In Experiment 2, we demonstrate that, if some sessions are not well shaped, the integrated regulation function of the scheduler can be degraded, and the proposed approach can alleviate the negative impact of misbehaving sessions.

We have implemented WF^{2}Q+ with per session-based queuing, WF^{2}Q+ with the grouping architecture, and our proposed scheme by OPNET. Let us assume that the output link capacity is $R = 8000$ cells/s, and there are four service rate group with r_{g_i} =1000, 2000, 4000, 8000 cells/s, where i=1, 2, 3, 4. Three sessions are in service, and the size of the token buffer is 1024 cells.

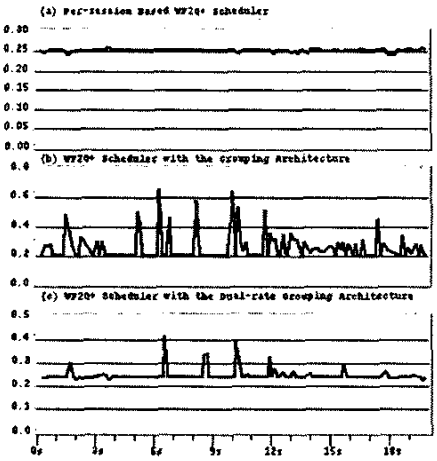

Figure 4: Bandwidth for session 1 in Experiment 1.

Experiment 1

Let r_1=2000 cells/s, $r_2 = r_3$= 3000 cells/s, and assume that each session is shaped by a leaky-bucket, and the session's token rate is the same as its required service rate.

When all sessions are continuously backlogged (for example in time interval (5s, 8s)), the normalized bandwidths allocated to each session should be 0.25, 0.375, and 0.375, respectively, as shown in Figure 4 (a) and 5 (a) for session 1 and 2, because the bandwidth is allocated proportionally to their required service rate, respectively. With the grouping architecture, as shown in Figure 4 (b) and 5 (b), the received bandwidths should be 0.20 and 0.40, respectively, because they are proportional to their rate of service group. Note that the arrival rates of session 1 and 2 are 2000 cells/s and 3000 cells/s, respectively, and the allocated bandwidths are 1600 cells/s and 3200 cells/s, respectively; in other words, session 2 is over-provisioned, while session 1 is under-provisioned. Thus session 2 is emptied frequently, and session 1 can receive more bandwidth when the queue of session 2 is emptied. Therefore, the allocated bandwidth to each session oscillates. With dual-rate grouping, bandwidths allocated to each session (Figure 4 (c) and 5 (c)) approximate the ideal case better than the grouping architecture since session 2 and 3 are placed into two different service groups alternately. The oscillations as well as their amplitude have been reduced. The allocated bandwidths for the three sessions are closely aligned with the ideal case, i.e., 0.24, 0.38 and 0.38, respectively, in the dual-rate grouping.

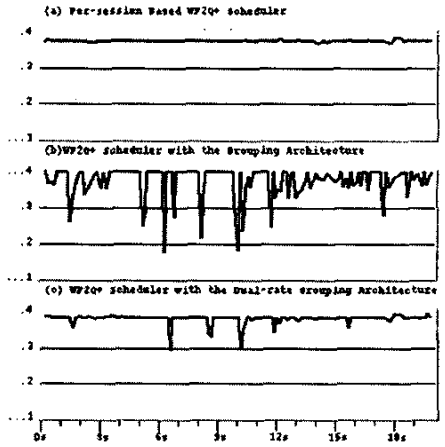

Figure 5: Bandwidth for session 2 in Experiment 1.

As shown in Figures 6, 7, and Table 1, the dual-rate grouping also approximates per session-based WF^{2}Q+ better than the grouping architecture in terms of delay, especially when all sessions are backlogged (in the interval (5s, 8s)). This is attributed to the fact that when all sessions are backlogged, the dual-rate grouping scheme can allocate more accurate bandwidth to all sessions than the grouping architecture. Note that the unit of delay is time slot.

Average Delay (in time slot)			
	Per session-based	The grouping architecture	The dual-rate grouping
Session 1	841	1362	1101
Session 2	433	262	350

Table 1 Average delay of sessions with different implementations

Experiment 2

In reality, per session-based leaky-bucket shaping may not be implemented in high-speed schedulers due to implementation complexity. In this experiment, we assume that sessions 2 and 3 are misbehaving, both with arrival rate 4000 cells/s, although both require only 3000 cells/s.

As shown in Figure 8(a) and 9(a), session 1 and 2 receive bandwidth of 0.25 and 0.375, respectively, because of the integrated regulation function of WF^{2}Q+. As shown in Figure 8 (b) and 9 (b), with the grouping architecture, the bandwidth allocated to session 1 is adversely affected by the misbehavior of session 2; session 2 can take advantage of the grouping architecture to gain more bandwidth (0.40) than they should receive. With the dual-rate grouping, as shown in Figure 8 (c) and 9 (c), the undesirable effect from the misbehaving sessions is alleviated; session 1 and 2 receive 0.24 and 0.38 of bandwidth, respectively, which is better in terms of approximating WF^{2}Q+ with per session based queuing. In this experiment, the performance in terms of delay can also be improved with our scheme. (The corresponding results are not shown here due to the space limitation).

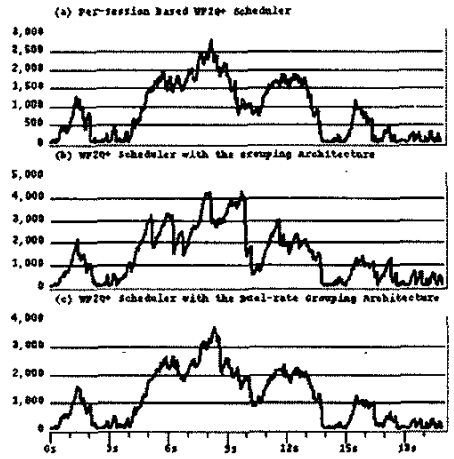

Figure 6: Delay of session 1 in Experiment 1.

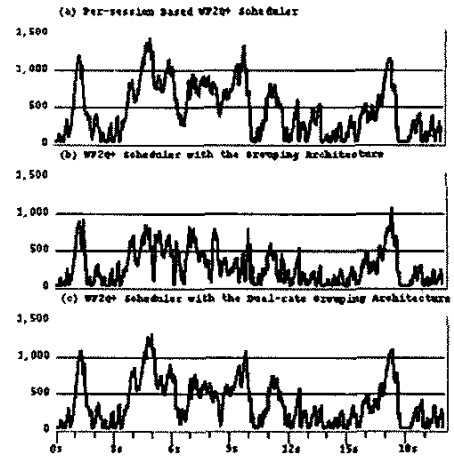

Figure 7: Delay of session 2 in Experiment 1.

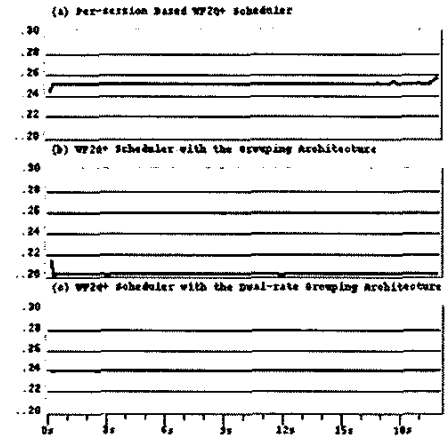

Figure 8: Bandwidth for session 1 in Experiment 2.

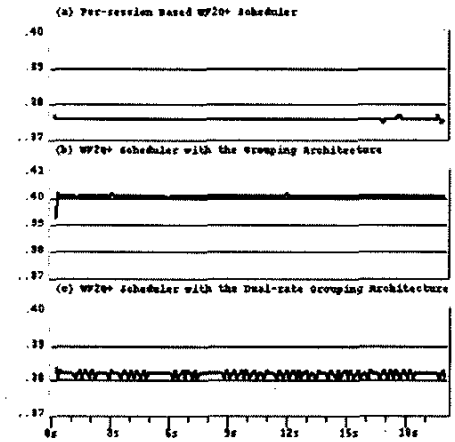

Figure 9: Bandwidth for session 2 in Experiment 2.

V. Conclusions

In this paper, we have presented a viable implementation of dual-rate grouping scheme in order to alleviate the problems of granularity associated with the original grouping architecture. The performance evaluation study has demonstrated that the proposed scheme can approximate the PFQs better than the original grouping architecture in terms of fairness of bandwidth allocation, immunity capability, and delay. One of the most important advantages of our proposal is that the implementation complexity remains the same as the grouping architecture.

References

[1] A.K. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," Ph.D thesis, MIT, 1992.

[2] D.C. Stephens, J.C.R. Bennett and H. Zhang, "Implementing scheduling algorithms in high-speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1145 –1158, June 1999.

[3] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queuing algorithm," *Internetworking: Research and Experience*, vol. 1, no.1, pp.3-26, 1990.

[4] S.J. Golestani, "A self-clocked fair queuing scheme for broadband applications," *Proceedings of IEEE INFOCOM'94*, pp. 636-646, April 1994.

[5] D. Stiliadis and V. Varma, "Design and analysis of frame-based fair queuing: a new traffic scheduling algorithm for packet-switched networks," *Proceedings of the ACM SIGMETRICS conference on Measurement & modeling of computer systems*, pp. 104 – 115, 1996.

[6] J.C.R. Bennett and H. Zhang, "Hierarchical packet fair queuing algorithms," *Proceedings of ACM SIGCOMM'96*, pp. 143-156, August 1996.

[7] S. Suri, G. Varghese and G. Chandranmenon, "Leap forward virtual clock," *Proceedings of INFOCOM'97*, vol.2, pp. 557 –565, April 1997.

[8] J.C.R. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queuing," *Proceedings of IEEE INFOCOM'96*, pp. 120-128, March 1996.

[9] J. Yang, D. Wei, S. Papavassiliou, and N. Ansari, "Improving service rate granularity by dual-rate session grouping in cell-based schedulers," *Proceedings of IEEE Globecom 2001*, pp. 2425-2429, November 2001.