Improving Service Rate Granularity By Dual-rate Session Grouping In Cell-based Schedulers[1]

Jie Yang, Dong Wei, Symeon Papavassiliou and Nirwan Ansari

New Jersey Institute of Technology,
Electrical and Computer Engineering Department,
University Heights, Newark, NJ, 07102

Abstract— In this paper we propose a scheme referred to as *dual-rate session grouping* to improve the service rate granularity for cell-based schedulers. In this scheme a session is split into two subsessions to provide the average service rate that a user requires. By applying dual-rate session grouping, the utilization of bandwidth and fairness among users can be improved, while the complexity of the scheduling algorithm remains the same as the conventional scheme. The overall computational complexity of the dual-rate session grouping does not increase with the rate granularity that is only limited by the available memory space. Several implementation issues are also presented in this paper.

I. Introduction

The ATM service models and the recent Integrated and Differentiated Service models in Internet [9] require that current and future packet switches support various service classes and provide connections to large amount of sessions over a single resource-shared physical infrastructure. Therefore, performance of the packet scheduler is critical to the QoS that a packet switch is able to provide. For this reason, packet scheduling algorithms have been intensively studied in recent years.

It is well known that Generalized Processor Sharing (GPS) algorithm and its packet-based version, *packet-by-packet* Generalized Processor Sharing (PGPS) [6] is an idealized algorithm which is able to: a) guarantee end-to-end delay to leaky-bucket-constrained sessions; b) provide real-time fair allocation of bandwidth among backlogged sessions regardless of the behavior of the sessions in the switch. However, GPS and PGPS are difficult to be implemented in packet schedulers due to their complexities. In the literature, a lot of scheduling algorithms for packet switching that approximate GPS and PGPS have been proposed such as WFQ [2], EFQ [4], H-PFQ [5] *etc*, which generally are referred to as Packet Fair Queueing (PFQ) algorithms [7]. The objective of these algorithms is to find a balance between the implementation complexity and the performance approximation to the idealized GPS.

In the literature many PFQ algorithms have two important properties: Locally Bounded Timestamp (LBT) and Globally Bounded Timestamp (GBT) [7]. To implement PFQ algorithms with LBT and GBT properties, Stephens *et al* [3], [7] proposed a scheduler architecture which is also applied in a cell-based scheduler in [8]. In this architecture, sessions with same rate are grouped together and the scheduler only supports a fixed number of rates. The complexity of the scheduling algorithm is then reduced from the number of sessions to the number of rates supported by the scheduler, which solves the scalability problem of the PFQ algorithm family. However, in such a scheme, the rate granularity of a scheduler limits the fairness among the sessions. For example, in a scheduler which supports exponentially distributed rates with 1Mbps, 2Mbps, 4Mbps,...., *etc.* when a session only requires a rate of 1.25Mbps, the conventional solution is to provide it with the rate 2Mbps. This solution not only makes the session user take advantage of the service provider but also is unfair to other sessions because such a session will be able to consume more resources than it is supposed to. Coarse rate granularity will also degrade link utilization [1].

In this paper we propose a scheme in which when a session requires a rate between two rates supported by the scheduler, it is split into two subsessions which are enqueued into the corresponding rate groups respectively. The ratio of the entire session in each of the subsession queues will ensure the average serving rate is equal to the rate that the session requires. We refer to this scheme as *dual-rate session grouping*. With dual-rate session grouping, we do not need to change the current scheduler architecture and we can improve the rate granularity without increasing the number of rate groups, i.e., without increasing the complexity of the PFQ algorithm applied in the scheduler. As long as we have sufficient memory to represent the ratio of a session in a subsession queue, we can provide any rate a user requires. We will study this scheme under the context of ATM-like networks in which the length of each packet is fixed, referred to as a *cell*. We believe that a cell-based scheduler with its scheduling algorithm and architecture can be widely applied, not only in ATM switches but also in packet-based switches due to the integration of IP and ATM switching technologies [10]. Moreover, the PFQ algorithms that can be applied in our scheduler have both the LBT and GBT properties.

The rest of this paper is organized as follows. In section II we discuss the principle of the scheme and its performance properties. In section III we explain the issues of implementation, which include the impact of session length on the scheme performance, the rate error that will be induced during implementation to make this scheme easily realized, and mechanisms to maintain cell sequence of a session while avoiding sorting in a rate group. We conclude our paper in section IV.

II. Dual-rate Session Grouping

Similar to [7], the architecture of a cell-based scheduler is shown in Fig.1 in which sessions with same rate are grouped together into one *rate group queue* and each session in the rate

[1] This work is supported in part by the New Jersey Center for Multimedia Research, and in part by the New Jersey Commission on Higher Education via the NJI-Tower Project.

group queue consists of cells of that session, which is therefore referred to as a *session queue*. The scheduler will only support a limited number of rate groups and their session queues. (Part of the notations are listed in Table I.) Any type of PFQ algorithms with LBT and GBT properties can be applied in this scheduler. In each rate group, only the cell with the smallest virtual start time is placed into the scheduler processor which in turn decides which cell should be served according to the packet selection policy of the PFQ algorithm. Without loss of generality, we assume in Fig.1, the following condition holds: $r^{(1)} < r^{(2)} \cdots < r^{(M)}$. When a session S_i requires a rate r_i that is between rate $r^{(K)}$ and $r^{(K+1)}$, i.e., $r^{(K)} < r_i \leq r^{(K+1)}$, where $r^{(K)}$ and $r^{(K+1)}$ are two consecutive rates supported by the scheduler, the conventional scheme adds S_i to rate group $(K+1)$ so that $\bar{r}_i$, the service rate of S_i, is $r^{(K+1)}$. We refer to such a scheme as *one-rate session grouping*.

TABLE I
NOTATIONS

M	the number of rate groups supported by a scheduler
l	the length of a cell
L_i	the number of cells in S_i
$r^{(K)}$	the rate of rate group K
S_i	session i
$S_i^{(K)}$	the subsession of S_i entering rate group K
s_i	virtual start time of S_i
f_i	virtual finish time of S_i
s_i^k	virtual start time of cell k, S_i
f_i^k	virtual finish time of cell k, S_i
r_i	the rate that S_i requires
r_i^k	the service rate that cell k in S_i receives
$\bar{r}_i$	the average service rate that S_i receives
e_i	the relative error between the service rates that S_i requires and receives
D_i	the average delay of S_i
V_i	the variance of delay of S_i induced by our scheme

Definition 1: For session S_i, the relative error between the required service rate r_i and the received service rate $\bar{r}_i$, e_i, is defined as

$$e_i = \mid \frac{r_i - \bar{r}_i}{r_i} \mid \qquad (1)$$

It is easy to follow that with one-rate session grouping scheme e_i can be as large as $(\gamma - 1)$ when $r^{(K+1)} = \gamma r^{(K)}$ $(\gamma > 1)$ and a rate r_i in the range $(r^{(K)}, r^{(K+1)}]$ is requested.

Based on such a cell-based, rate-grouping scheduler architecture, our proposed *dual-rate session grouping* scheme works as follows: when S_i requests such a rate r_i that $r^{(K)} < r_i < r^{(K+1)}$, we split S_i into two subsessions, $S_i^{(K)}$ and $S_i^{(K+1)}$, which associate with rate group K and $K+1$ respectively. According to properties of PFQ algorithms, cells in $S_i^{(K)}$ will be served in average rate $r^{(K)}$ and cells in $S_i^{(K+1)}$ will be served

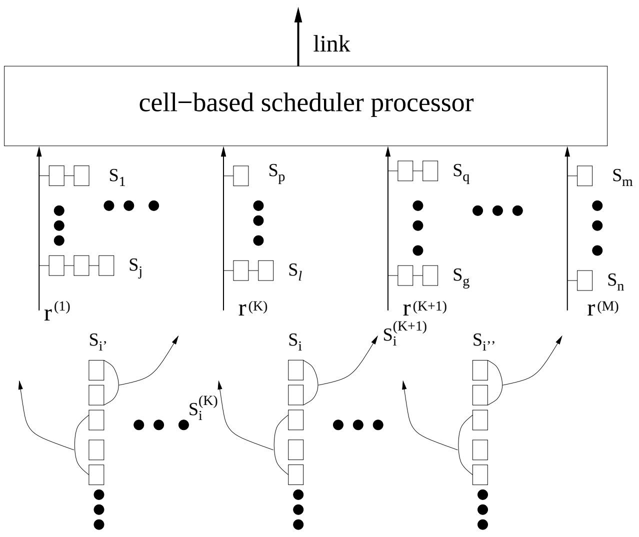

Fig. 1. The cell-based scheduler architecture

in average rate $r^{(K+1)}$ while we require that the average service rate of the entire session S_i is r_i. The ratio of cells in $S_i^{(K)}$ is decided by the following theorem.

Theorem 1: The ratio $x_i^{(K)}$ of cells in $S_i^{(K)}$ to the entire session S_i is given by

$$x_i^{(K)} = \frac{r^{(K)}(r^{(K+1)} - r_i)}{r_i(r^{(K+1)} - r^{(K)})} \qquad (2)$$

where $r^{(1)} \leq r^{(K)} < r^{(K+1)} \leq r^{(M)}$.

Correspondingly,

$$x_i^{(K+1)} = 1 - x_i^{(K)} \qquad (3)$$

Proof: Suppose the length of session S_i is L_i. The average service time for cells in $S_i^{(K)}$ is given by $\frac{lL_i x_i^{(K)}}{r^{(K)}}$. Correspondingly, the average service time for cells in $S_i^{(K+1)}$ is $\frac{lL_i x_i^{(K+1)}}{r^{(K+1)}}$. Therefore, the average service rate of S_i, $\bar{r}_i$, is given by

$$\bar{r}_i = \frac{lL_i}{\frac{lL_i x_i^{(K)}}{r^{(K)}} + \frac{lL_i x_i^{(K+1)}}{r^{(K+1)}}} = \frac{1}{\frac{x_i^{(K)}}{r^{(K)}} + \frac{(1 - x_i^{(K)})}{r^{(K+1)}}} \qquad (4)$$

We require that $\bar{r}_i = r_i$. Then (2) and (3) can be achieved. ■

From (2) and theorem 1, we can directly get the following corollary.

Corollary 1: $x_i^{(K)}$ has the following properties:

i) $0 \leq x_i^{(K)} < 1$;

ii) $x_i^{(K)}$ can be represented by

$$x_i^{(K)} = \frac{n_i}{d_i} \qquad (5)$$

where $n_i \geq 0$,$d_i > 0$ and they are integers.

It is easy to follow that with dual-rate session grouping, the session's average cell delay D_i will be $l/\bar{r}_i$. In the ideal case in which the ratio $x_i^{(K)}$ can be achieved, D_i is equal to the delay required by S_i and there is no relative error between the required service rate r_i and the received service rate $\bar{r}_i$.

There is additional cell delay jitter induced by this scheme, which is represented by the following definition.

Definition 2: The additional cell delay jitter induced by dual-rate session grouping is defined as the additional delay variance induced by the scheme and is given by

$$V_i = x_i^{(K)}(\frac{l}{r^K} - \frac{l}{\bar{r}_i})^2 + (1 - x_i^{(K)})(\frac{l}{\bar{r}_i} - \frac{l}{r^{(K+1)}})^2 \quad (6)$$

The dual-rate session grouping has the following property.

Corollary 2: The difference between the average cell delay up to a cell k and the session's average cell delay D_i, is bounded by

$$\mid D_i(k) - D_i \mid \leq \max(l(\frac{1}{r^{(K)}} - \frac{1}{\bar{r}_i}), l(\frac{1}{\bar{r}_i} - \frac{1}{r^{(K+1)}})) \quad (7)$$

where $D_i(k)$ is the session's average cell delay up to cell k.

III. Realization of Dual-rate Session Grouping

When we implement the scheduler, we have to set up a processing table for each session which stores such information as the session's current virtual start time s_i, the required service rate r_i, *etc*. To implement dual-rate session grouping, additional information has to be provided in the processing table.

First, equation (5), the representation of $x_i^{(K)}$, has to be implemented in the processing table. This can be achieved through a label for n_i and a counter c_i. Second, we need a flag to indicate which rate group the next cell should enter. Third, besides s_i for session S_i, the virtual start time for cell k, the latest cell of S_i entering the system, s_i^k is to be stored to avoid potential sequence error induced by dual-rate session grouping, which is explained later.

Dual-rate session grouping can be implemented in a weighted round robin fashion. Initially c_i is set to 0 and the flag indicates that arriving cells enter rate group $K+1$. Each time a new cell of S_i comes, c_i is incremented by 1. When $d_i - n_i$ cells have entered rate group $K+1$, the flag is set to indicate that future cells enter group K. When d_i cells have been received, c_i is reset to 0 while the flag is reset to indicate that future cells enter rate group $K+1$.

Compared to one-rate session grouping, the computational complexity upon cell arrival is slightly higher and more space is needed for the processing table. However, these are only minor trade-offs in the implementation because the two major bottlenecks in a scheduler is [7]: 1) the number of memory accesses needed during the processing; 2) the number of rate groups that the scheduler processor can support. Since the computation above will only involve two memory accesses upon cell arrival: one to load processing table, one to store it after update, the number of memory accesses is same as that of one-rate session grouping, which also involves two memory accesses for the same purposes. Note that since the PFQ algorithm performed in the scheduler processor is not affected, the cell-departure procedure will remain unchanged under the dual-rate session grouping as under the one-rate session grouping. For this reason, the number of rate groups that the scheduler processor can support is same under the two schemes. Therefore, dual-rate session grouping is applicable without increasing the complexity of the PFQ algorithms performed in the scheduler processor.

A. Effect of Session Length on Performance

Note that the scheduler is based on cells. When we implement the dual-rate session grouping by splitting the incoming cells in a weighted round-robin fashion, the length of a session will affect the performance of our scheme. For example, when the ratio $x_i^{(K)}$ is implemented by putting first $d_i - n_i$ cells into rate group $K+1$ for every d_i incoming cells, if the length of the session is smaller than $d_i - n_i$, the session actually receives a service rate $r^{(K+1)}$. The performance converges to the average value given by(4) and (6) while $D_i(k)$ converges to D_i when $k \to \infty$. Therefore, the larger d_i is, the longer session we need to get our desired performance by dual-rate session grouping. This effect is shown by the following example. Suppose a user requires a rate 3Mbps, while the scheduler only has rate groups of 2Mbps and 4Mbps. By applying dual-rate session grouping, $x_i^{(K)} = 1/3$ and $x_i^{(K+1)} = 2/3$. To implement this scheme, for every 3 incoming cells, we put first 2 cells to rate group of 4Mbps, and the last one cell to rate group 2Mbps. If the session is only 2-cell long, it receives a service rate of 4Mbps, higher than it requires. However, the service rate will converge to 3Mbps when the number of cells in the session increases, as shown in Fig.2. Fig.3 demonstrates the relationship between the average cell delay and the number of cells in the session. It can be seen that with increasing of session length, the average cell delay converges to the value that the user requires.

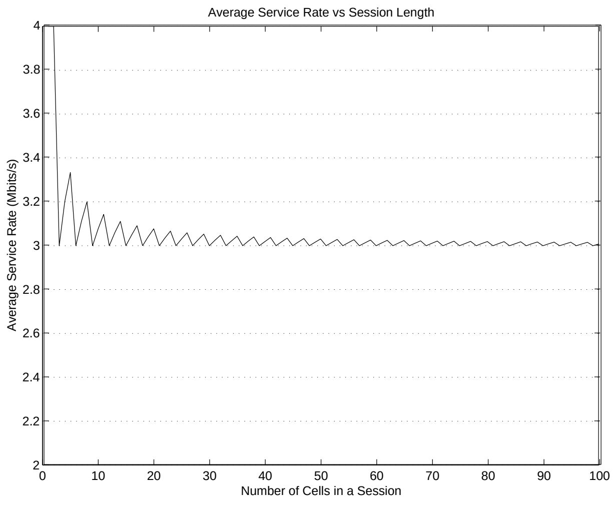

Fig. 2. The relationship between average service rate and session length

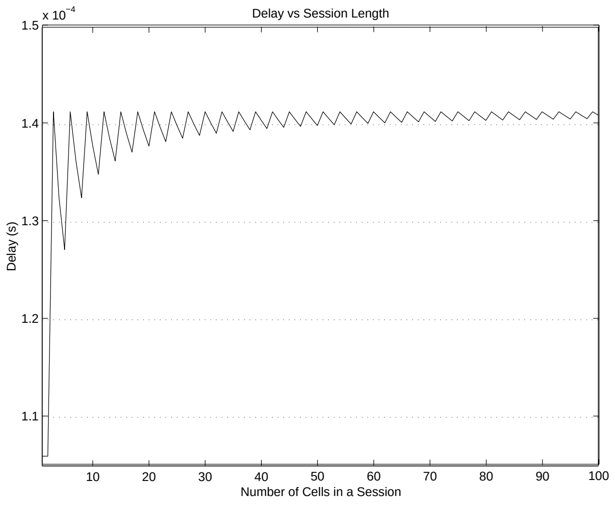

Fig. 3. The relationship between average delay and session length

B. Implementation of Dual-rate Session Grouping

For different $r^{(K)}$, r_i and $r^{(K+1)}$, different value of $x_i^{(K)}$ is required. For example, when $r^{(K)}$=1Mbps, $r^{(K+1)}$=2Mbps, if $r_i = 1.6$Mbps, $x_i^{(K)} = \frac{1}{4}$, which can be implemented by three bits: one bit for n_i and two bits for c_i. In general, we need $\lceil \log_2 n_i \rceil$ bits for n_i, and $\lceil \log_2 d_i \rceil$ bits for c_i. From Corollary 1, we have $\lceil \log_2 n_i \rceil < \lceil \log_2 d_i \rceil$.

Since r_i can not be known *a priori*, when we implement n_i and c_i, we allocate in the processing table a fixed number of bits, b, for representation of n_i and c_i, respectively. To simplify the operation of dual-rate session grouping and save space for the processing table, we will not implement the representation of d_i in the processing table. c_i will only be reset when it overflows and rolls over back to 0. Therefore, the possible values of $x_i^{(K)}$ are: 0, $\frac{1}{2^b}$, $\frac{2}{2^b}$, $\cdots$, $\frac{2^b-1}{2^b}$. From equation (4), we can see that each $x_i^{(K)}$ corresponds to a specific $\bar{r}_i$. For this reason, we can improve rate granularity by adding $2^b - 1$ additional rates between the rate $r^{(K)}$ and $r^{(K+1)}$.

C. Implementation error

For a user who requests a rate r_i that requires a $x_i^{(K)}$ not supported in the processing table , i.e. $\frac{n_i}{d_i} \neq \frac{j}{2^b}$, where $j = 0, 1, 2, \cdots, 2^b - 1$, we provide the user with a rate $\bar{r}_i > r_i$, which is also the closest rate to r_i and supported by the processing table. In this case, there is a relative error e_i between the service rate requested and received. The upper bound of e_i is given by the following theorem.

Theorem 2: Suppose $r^{(K+1)} = \gamma r^{(K)}$, $\gamma > 1$. If in the processing table $x_i^{(K)}$ can only be represented by the discrete values: $x_i^{(K)} = \frac{j}{2^b}$, where $j = 0, 1, 2, \cdots, 2^b - 1$, then e_i is bounded by

$$e_i \leq \frac{\gamma - 1}{2^b} \tag{8}$$

Proof: Suppose $\bar{r}_1$ and $\bar{r}_2$ are two consecutive rates supported in the processing table due to dual-rate session grouping and $r^{(K)} < \bar{r}_1 < \bar{r}_2 \leq r^{(K+1)}$. If $\bar{r}_1$ requires a ratio $x_1^{(K)}$ and $\bar{r}_2$ requires a ratio $x_2^{(K)}$, from (2) and (4), we have $x_1^{(K)} > x_2^{(K)}$, since (4) demonstrates that $\bar{r}_i$ is a monotonic decreasing function of $x_i^{(K)}$. Let $x_1^{(K)} = \frac{n_1}{d_1}$ and $x_2^{(K)} = \frac{n_2}{d_2}$. From our implementation, $d_1 = d_2 = 2^b$, and $n_1 = n_2 + 1$. The e_i is maximized when a user requires a rate r_i slightly larger than $\bar{r}_1$ and is then provided with a rate $\bar{r}_2$. Therefore, from (1) and (4) we can have

$$e_i \leq \frac{\bar{r}_2}{\bar{r}_1} - 1 = \frac{(\gamma-1)x_1^{(K)}+1}{(\gamma-1)x_2^{(K)}+1} - 1 = \frac{\gamma-1}{(\gamma-1)n_2+d_2} \leq \frac{\gamma-1}{d_2} = \frac{\gamma-1}{2^b}$$

■

Although from relation (8) we can see that e_i can be significantly reduced by using a larger b, it requires longer session for the average rate to converge to the required value, which can be seen in subsection III-A. Therefore, we can get benefit from large b only when the session is long enough.

D. Maintaining Cell Sequence

With one-rate session grouping, a session queue is maintained in First In First Out (FIFO) manner. In the processing table, only the virtual start time at the head of the session queue needs to be stored. Each time a cell is fetched into the scheduler processor or a new session comes in, the session will be moved or appended to the tail of the rate group queue and a new virtual start time will be stored for that session. It is proved in [7] that with this implementation, in a rate group queue, a session with the smallest virtual start time will be served first by the processor without performing the costly sorting. Moreover, the sequence of cells in each session will be maintained, which is important to virtual connections.

In dual-rate session grouping, in a rate group the sequence between different sessions and subsessions can be maintained in the same way as in the one-rate session grouping. Moreover, when a session is split into two subsessions, the relative sequence of cells in the same subsession can be still maintained in FIFO manner. However, a potential problem is how to maintain the relative sequence of two cells in the same session but in two different subsessions, i.e., in two rate groups.

A property of the PFQ algorithm is that the virtual clock $v(t)$, the virtual start time $s_i(t)$ and the virtual finish time $f_i(t)$ are monotonic increasing functions of time t. Moreover, when two cells k and $k+1$ arrive sequentially, the virtual start time has the following relationship [6]:

$$s_i^{k+1} = \max(v(t), s_i^k + \frac{l}{r_i^k}) \tag{9}$$

Therefore, we have

$$s_i^{k+1} \geq s_i^k + \frac{l}{r_i^k} \tag{10}$$

where r_i^k is the service rate cell k will receive. The above relation (10) also implies that $s_i^k < s_i^{k+1}$. Since $f_i^k = s_i^k + l/r_i^k$ and $f_i^{k+1} = s_i^{k+1} + l/r_i^{k+1}$, we have

$$f_i^{k+1} \geq f_i^k + \frac{l}{r_i^{k+1}} > f_i^k \tag{11}$$

Therefore, when (10) holds, even if cell k and $k+1$ stay in different rate group, we can guarantee that (11) will hold. When both (10) and (11) hold, the packet selection policies of PFQ algorithms, such as "smallest virtual finish time first" (SFF), "smallest virtual start time first"(SSF) *etc* [7] will guarantee that the service sequence of cells in a session is same as their arrival sequence.

However, even in the one-rate session grouping, there are two complications in implementation. First, since the number of bits to represent the timestamp is finite, when the timestamp of $v(t)$ is rolled over, the computation of (9) may give wrong result. For this reason, in one-rate session grouping, when S_i is not backlogged as cell $k+1$ arrives, s_i^{k+1} will be assigned a value in the range $[v(t), v(t) + (l/r_i^k)]$ based on the GBT property. The second complication is in each rate group, sorting among different sessions should be avoided. Therefore, a newly backlogged session will be assigned a value at least as large as that of the tail of the backlogged sessions in the rate group [3] but still in the range $[v(t), v(t) + (l/r_i^k)]$.

In dual-rate session grouping, there are three additional complications regarding the overflow of the timestamp and sorting in the rate group. The first complication is when cell $k+1$ arrives, cell k is not backlogged and cell $k+1$ will enter a rate group different from that of cell k. The desired value of s_i^{k+1} is given by $\max(s_i^k + (l/r_i^k), v(t))$. However, to guarantee LBT property of the rate group that cell $k+1$ will enter, s_i^{k+1} should be in the range $[s_T, s_H + (l/r_i^{k+1})]$, where s_H and s_T are the virtual start time of the head and tail of the backlogged sessions in the rate group that cell $k+1$ will enter. Therefore s_i^{k+1} will be assigned a value given by $\max\{s_T, \min[s_H + (l/r_i^{k+1}), \max(s_i^k + (l/r_i^k), v(t))]\}$. This value is in the range $[v(t), v(t) + (l/r_i^{k+1})]$, which also satisfies the GBT property.

The second complication is that when cell $k+1$ arrives, cell k is backlogged, and cell k and $k+1$ are in different rate groups. From (9), s_i^{k+1} should be given by $s_i^k + (l/r_i^k)$. But to avoid sorting and satisfy LBT property in the new rate group, s_i^{k+1} should also be in the range $[s_T, s_H + (l/r_i^{k+1})]$, therefore, is actually given by $\max\{\min[s_i^k + (l/r_i^k), s_H + (l/r_i^{k+1})], s_T\}$.

The third complication in maintaining cell sequence of a session is when cell k is backlogged in one rate group, cell $k+1$ in the other rate group is not necessarily backlogged. If the bit number of a timestamp were infinite, (10) and (11) would guarantee that cell $k+1$ would not leave the scheduler earlier than cell k. However, since the bit number of a timestamp is finite, the comparison among the timestamps of sessions from different rate groups might give error results. In one-rate session grouping, this will only cause additional delay and delay jitter to a session, but in dual-rate session grouping, it will cause cell sequence error. To avoid such an error, when cell k and $k+1$ are in different rate groups, in the processing table we can use a flag for cell $k+1$ to indicate whether cell k is backlogged or not. If cell k is backlogged, cell $k+1$ will be first appended to the tail of cell k. Only when cell k is served, will cell $k+1$ calculate its new virtual start time and be appended to the new rate group that it should enter.

IV. Conclusion

In this paper, we propose a dual-rate session grouping scheme to improve the service rate granularity of the cell-based scheduler architecture. We first discussed its principle and performance properties which indicate that we can provide any service rate that a user requires with some additional bounded delay jitter. The complexity of applied PFQ algorithms and the architecture of the scheduler are not affected by our scheme. When we implement the scheme, the rate granularity will be limited by the memory space used for it, although in the scheme the overall computational complexity will not increase with finer rate granularity. Finally we discuss how to maintain cell sequence in a session and avoid sorting in a rate group, which guarantee the correctness of the dual-rate session grouping scheme.

References

[1] A.Charny, K.K.Ramakrishnan, and A.G.Lauck. "Scalability Issues for Distributed Explicit Rate Allocation in ATM Networks". *IEEE INFOCOM'96*, pages 1198–1205, 1996.

[2] A.Demers, S. Keshav, and S. Shenker. "Analysis and simulation of a fair queueing algorithm". *J. Internetworking Res. Experience*, 1:3–26, 1990.

[3] J.C.R. Bennett, Donpaul C. Stephens, and Hui Zhang. "High Speed, Scalable, and Accurate Implementation of Packet Fair Queueing Algorithms in ATM Networks". *IEEE ICNP'97*, pages 7–14, 1997.

[4] C.Wang, K. Long, X.Gong, and S. Cheng. "Effective fairness queueing algorithms". *ICON'2000*, pages 294–301, 2000.

[5] J.C.R.Bennett, D.C.Stephens, and H.Zhang. "Hierarchical packet fair queueing algorithms". *ACM-SIGCOMM'96*, pages 143–156, 1996.

[6] Abhay K. Parekh and Robert G. Gallager. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case". *IEEE/ACM Transactions On Networking*, 1(3):344–357, 6 1993.

[7] Donpaul C. Stephens, J.C.R. Bennett, and Hui Zhang. "Implementing Scheduling Algorithms in High-Speed Networks". *IEEE Journal on Selected Areas in Communications*, 17(6):1145–1158, 6 1999.

[8] Dong Wei, Jianguo Chen, and Nirwan Ansari. "An Efficient Expression of Timestamp and Period in Packet-based and Cell-based Schedulers". *Proc.of ICC'01*, 2001.

[9] Xipeng Xiao and Lionel M.Ni. "Internet QoS: A Big Picture". *IEEE network*, pages 8–18, March/April 1999.

[10] Jie Yang, Necdet Uzun, and Symeon Papavassiliou. " The Architecture Design for a Terabit IP Switch Router". *Proc.of IEEE HPSR'01 Workshop*, 2001.