An Efficient Expression of Timestamp and Period in Packet-based and Cell-based Schedulers *

Dong Wei†, Jianguo Chen‡, and Nirwan Ansari†

† Advanced Networking Lab
New Jersey Institute of Technology
dxw3077@njit.edu, nirwan.ansari@njit.edu

‡ Bell Labs
Lucent Technologies
jianguo11@lucent.com

Abstract – **Scheduling algorithms are implemented in hardware in high-speed switches to provision Quality-of-Service guarantees in both cell-based and packet-based networks. Being able to guarantee end-to-end delay and fairness, timestamp-based fair queuing algorithms have received much attention in the past few years. In timestamp-based fair queuing algorithms, the size of timestamp and period determines the supportable rates in terms of the range and accuracy. Furthermore, it also determines the scheduler's memory in terms of off-chip bandwidth and storage space. An efficient expression can reduce the size of the timestamp and period without compromising the accuracy. In this paper, we propose a new expression of the timestamp and period, which can be implemented in hardware for both high-speed packet-based and cell-based switches. As compared to fixed-point and floating-point number expression, when the size is fixed, the proposed expression has a better accuracy.**

I. INTRODUCTION

The current high-speed, service-integrated and packet-switched networks support many kinds of services at the same time. Packet switches are required to support a large number of sessions with diverse bandwidth requirements; for example, the supportable rate can go as low as 4 Kbits/s and as high as 2.4Gbits/s (OC48) and 10Gbits/s (OC192). Packet switches are also required to support a wide range of packet sizes, from 40 bytes to 64 Kbytes (such as IP). Three important issues should be considered in the design of a scheduler: 1) end-to-end delay, 2) fairness, and 3) implementation complexity.

Based on the architecture of the schedulers, packet switches are classified into two types [1]: 1) frame-based, and 2) sorted priority. Recently, sorted priority algorithms, also known as packet fair queuing (PFQ), have received much attention because they can approximate the idealized generalized processor-sharing (GPS) algorithm, which has desirable properties in terms of end-to-end delay and fairness [2].

In a PFQ algorithm, there is a global variable called virtual time, associated with outgoing sessions being scheduled. The virtual time is updated when a packet receives service. Each packet has its own timestamp in the system. All packets are sorted by their timestamps. Timestamp sorted algorithms [1]-[2] include weighted fair queuing (WFQ), self-clocked fair queuing (SCFQ), and worst-case weighted fair queuing such as WF^2Q and WF^2Q+. The virtual start time and the virtual finish time are the typical timestamps used in these algorithms. Service interval is a function of the packet size and the required session bandwidth. Given a virtual start time, the service interval is used to calculate the virtual finish time, and vice versa.

The size of the timestamp and period determines the supportable rate in terms of range and accuracy. In this sense, it seems to be tempting to use a larger size to represent them. However, the size of the timestamp and period determines the system memory in terms of bandwidth required to access and space to store. In this regard, the smaller size the better. To resolve this trade-off, we need to find the optimal representation of the timestamp and period that can meet the required accuracy in the smallest size. Note that it is very difficult to use normal fixed-point or floating-point to represent the large range of both the packet size and service rate efficiently. To simplify the implementation and obtain a satisfactory accuracy, we propose an alternative expression. By shifting the decimal point to accommodate different ranges of service rates, we can have a better representation in terms of accuracy and size of timestamp and period. This representation is generically applicable to any timestamp-based scheduling algorithm.

The rest of the paper is organized as follows. Section II presents the background of the expression of timestamp and period of PFQ. Our proposed expression for packed-based schedulers is discussed in Section III. The proposed expression is extended for cell-based (ATM) schedulers in Section IV.

II. BACKGROUND

PFQ algorithms are used to approximate the idealized generalized processor-sharing (GPS) algorithm. All PFQ algorithms have similar sorted-queue architecture. They differ in two aspects [2]: virtual time function and packet-selection policy.

* This work was supported in part by the New Jersey Commission on Science and Technology via the NJ Center for Wireless Telecommunications, and the New Jersey Commission on Higher Education via the NJI-TOWER project.

0-7803-7097-1/01/$10.00 ©2001 IEEE

A. Notation

$Z(.)$ – the number of bits
$\bar{P}$ – the idealized period with infinite bit expression
P – actual period representation with finite bit expression
$\bar{r}$ – the idealized service rate with infinite bit expression
r – actual service rate with finite bit expression
r_{max} – maximum supportable service rate
r_{min} – minimum supportable service rate
r_i – required service rate for session i
r_{LC} – link capacity of the scheduling system
$\varepsilon(\alpha)$ – the relative error of α
T – timestamp
I_α – integer part of α
F_α – fractional part of α
M_α – mantissa part of α
E_α – exponent part of α
L – the packet size
N- total number of bits of timestamp and period

B. PFQ Algorithms

PFQ algorithms have a global variable – system virtual 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:

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

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

The worst-case fair index (WFI) [3] was introduced to characterize the fairness performance of PFQ algorithms. It was shown that PFQ algorithms with two tags (the virtual start time and the virtual finish time) can achieve better fairness performance than those with only a single tag. From (1) and (2), the virtual finish time can be derived from the virtual start time and the packet service interval.

C. Expression of Timestamp and Period

In a packet scheduling system, owing to the size limit of the outgoing buffer, each packet is split into fixed-size fragments, and the size of each fragment normally equals to the minimum supportable packet size L_{min}. The concept of time slot is introduced to normalize the time interval. One time slot τ equals to the time interval required to transmit a fragment at link rate r_{LC}: $\tau = L_{min} / r_{LC}$. Usually, the maximum supportable service rate is the same as the link rate of the scheduling system: $r_{max} = r_{LC}$. The period of session i is defined as follows: $P_i = r_{max}/r_i$, where r_i is the required service rate of session i.

Denote Φ_i^k as the service interval for the k^{th} packet of session i, and $\underline{\Phi_i^k}$ as the normalized value of Φ_i^k. That is,

$$\Phi_i^k = \frac{L_i^k}{r_i} \text{ and } \underline{\Phi_i^k} = \frac{\Phi_i^k}{\tau} = \frac{L_i^k / r_i}{L_{\min} / r_{\max}},$$

where L_i^k is the size of the k^{th} packet of session i.

The service interval of one packet is the time in seconds required to transmit this packet. The period of a session is the ratio of the system maximum supportable rate to the bandwidth requirement of this session, and it is a unitless parameter. A session with $P=3$ means that the session needs to receive service of one time slot in every three time slots. The service interval and period are related as follows:

$$\Phi^k \quad \frac{L^k}{r} = \frac{L_i}{L} \frac{L_{\min}}{r_{\max}} \frac{r}{r_i} = \frac{L^k}{L_{\min}} \tau P \quad (3)$$

The period is stored in a processing table for each session and the timestamp is assigned for each packet. Using the virtual start time as the timestamp, the service interval can be calculated from (3), and thus we can derive the virtual finish time from (2); using the virtual finish time as the timestamp, the virtual start time can be similarly computed.

Modular comparison is used to select 1) packets eligible to enter the scheduler; 2) fragments which should get service. By employing modular comparison, two binary numbers represented by $n+1$ bits can be compared without ambiguity if the difference between them is less than 2^n. Using the notation X[i:j] to represent the binary number extracted from the i^{th} through j^{th} bits of X, with the convention that the LSB bit is the 0^{th} bit. A modular arithmetic comparison X>Y can be computed by the following pseudo code:

```
Boolean Modular_Comparison (X, Y)
1   if X[n-1:0] > Y[n-1:0]
2     then result = TRUE
3     else result = FALSE
4   if X[n]=Y[n]
5     then return result
6     else return NOT result
```

X[n-1:0] represents the binary number, and X[n] (the n^{th} bit of X) is used to discern wraparound ambiguity [2],[5]. The following condition must be satisfied:

$$\left| X - Y \right| < 2^n.$$

With this property, Reference [4] suggests that the size of timestamps has to be at least one bit larger than the largest normalized service interval $\underline{\Phi_{\max}}$.

$$\underline{\Phi_{\max}} = \frac{L_{\max} / r_{\min}}{L_{\min} / r_{\max}}$$

Therefore:

$$Z(I_T) \geq \left\lceil \log_2 \frac{r_{\max}}{r_{\min}} + \log_2 \frac{L_{\max}}{L_{\min}} \right\rceil + 1 \quad (4)$$

The largest period is r_{max}/r_{min}. Therefore, the number of bits to represent the integer part of the period must satisfy the following inequality:

$$Z(I_P) \geq \left\lceil \log_2 \frac{r_{max}}{r_{min}} \right\rceil \qquad (5)$$

In a cell-based scheduling system, it is tempting to use an integer representation of the timestamp, so that the system virtual time is simply increased by one each time a cell is transmitted. However, this would adversely affect the provisioning of those sessions with high bandwidth requirement. In a scheduling system with integer representation of virtual time, the period of 1,2,3 ... represents service rate of 1, 1/2, and 1/3... times the link capacity. As a result, session rates between 1/2 and 1/3 of the link capacity cannot be represented. Therefore, we need more bits after the decimal point to represent the timestamp and period of high-rate sessions. The relative error of the service rate can be expressed in terms of the period representation with finite bits as follows: $\varepsilon(r) \approx \varepsilon(P)$, with the assumption that we have enough bits to represent the period, then $|P - \overline{P}|$ is far less than $\overline{P}$.

The timestamp and period are stored and used together for each packet in the system. The accuracy of both of them determines the accuracy of the service rate. Usually, the number of bits of the timestamp is determined by Φ_{max}, and the accuracy of the period is determined by both of the number of bits of the period and the accuracy of the timestamp.

1) The Fixed-Point Expression:

Using the fixed-point expression, the minimal number of bits to represent (from (4) and (5)) the timestamp and period are:

$$\begin{cases} Z(I_T) = \left\lceil \log_2 \frac{L_{max}}{L_{min}} + \log_2 \frac{r_{max}}{r_{min}} \right\rceil + 1 & (6) \\ Z(I_P) = \left\lceil \log_2 \frac{r_{max}}{r_{min}} \right\rceil & (7) \end{cases}$$

Since scheduling involves arithmetic operations on both timestamp and period, they should maintain the same accuracy. That is, they should have the same number of fractional bits.

$$Z(F_P) = Z(F_T) \quad , \qquad (8)$$

in which case we will not waste any bit in the expression. Thus, the generalized accuracy of the fixed-point expression is:

$$\varepsilon(P) = \begin{cases} \dfrac{2^{-(Z(F_P)+1)}}{P}, & if \quad Z(F_P) \leq Z(F_T) \\ \dfrac{2^{-(Z(F_T)+1)}}{P}, & if \quad Z(F_P) \geq Z(F_T) \end{cases} \qquad (9)$$

The total number of bits to represent the timestamp and period is:

$$N = Z(I_T) + Z(F_T) + Z(I_P) + Z(F_P) \qquad (10)$$

Consider a packet scheduling system ***S*** which is required to support service rates from 4 Kbits/s to 622 Mbits/s, and the packet size ranged from 40 bytes to 64 Kbytes. Then, $Z(I_T)$ = 29 and $Z(I_P)$ = 19. With $Z(F_T) = Z(F_P) = 0$, i.e. $N = 48$, the maximum relative error of the service rate is shown in Fig. 1.

2) The Floating-Point Expression

Similarly, in order to achieve unambiguous modular comparison, the following inequality must be satisfied for the floating-point expression:

$$Z(M_T) \geq \left\lceil \log_2 \frac{L_{max}}{L_{min}} + \log_2 \frac{r_{max}}{r_{min}} \right\rceil + 1 \qquad (12)$$

Let δ be the number of additional bits:

$$Z(M_T) = \left\lceil \log_2 \frac{L_{max}}{L_{min}} + \log_2 \frac{r_{max}}{r_{min}} \right\rceil + 1 + \delta \qquad (13)$$

$$Z(M_P) = \left\lceil \log_2 \frac{r_{max}}{r_{min}} \right\rceil + \delta \qquad (14)$$

To maintain the same degree of accuracy of the timestamp,

$$Z(E_T) = \lceil \log_2 Z(M_T) \rceil \qquad (15)$$

$$Z(E_P) = \lceil \log_2 Z(M_P) \rceil \qquad (16)$$

Thus,

$$N = Z(M_T) + Z(E_T) + Z(M_P) + Z(E_P) \qquad (17)$$

Denote $\lambda(P)=Z(M_P)-E_P$ and $\lambda(T)=Z(M_T)-E_T$. $\lambda(P)$ and $\lambda(T)$ are the number of bits after the decimal point in the period and timestamp, respectively. Thus, the maximum relative error can be expressed as:

$$\varepsilon(P) = \begin{cases} \dfrac{2^{-1}}{\sum_{i=0}^{Z(M_P)-1} B_i(M_P) \cdot 2^i + 2^{Z(M_P)}} & if \quad \lambda(P) \leq \lambda(T) \\ \dfrac{2^{\lambda(P)-\lambda(T)-1}}{\sum_{i=\lambda(P)-\lambda(T)}^{Z(M_P)-1} B_i(M_P) \cdot 2^i + 2^{Z(M_P)}} & else \end{cases} \qquad (18)$$

Intuitively, with $\lambda(T) < \lambda(P)$, if the timestamp has more bits after the decimal point, the accuracy of the period is higher. The value of the timestamp ranges from 0 to Φ_{max}, implying that $\lambda(T)$ ranges from δ to $Z(M_T)$. The worst case, corresponding to $\lambda(T) = \lambda_{min}(T) = Z(M_T)-\delta$, yields the following maximum relative error:

$$\varepsilon(P)=\begin{cases}\dfrac{2^{-1}}{\sum_{i=0}^{Z(M_P)-1}B_i(M_P)\cdot 2^i+2^{Z(M_P)}} & \text{if } \lambda(P)\le\lambda_{\min}(T)\\ \dfrac{2^{\lambda(P)-\lambda(T)-1}}{\sum_{i=\lambda(P)-\lambda_{\min}(T)}^{Z(M_P)-1}B_i(M_P)\cdot 2^i+2^{Z(M_P)}} & \text{else}\end{cases}$$

Consider the same packet scheduling system ***S*** as before. Using (15) and (16), we have

$Z(E_T)=Z(E_P)=5$.

With ($Z(M_T)$ = 31, $Z(M_P)$ = 7), i.e., N= 48, δ=2, the maximum relative error of the service rate using the floating-point expression is shown in Fig. 1.

III. The Proposed Expression for Packet Schedulers

We propose a new expression by allocating bits to a range number, say, C. Each range number C determines the decimal point, and thus the range of the period. Denote C_i as the range number for session ***i***. For example, when C_i=4, then $2^4 \le P_i < 2^5$.

Equations (6), (8) and (9) are still held for this expression, but the total number of bits becomes:

$$N = Z(I_T)+Z(F_T)+Z(I_P)+Z(F_P)+Z(C) \qquad (19)$$

In a scheduling system, $Z(I_T)$ and $Z(C)$ are constants. Other three parameters are determined by the range of the period, but the total number of bits N remains the same. We shall illustrate this representation by means of an example.

A. Example

Consider the same packet scheduling system ***S*** as before. To compare the performance with the fixed-point and floating-point representations, the number of bits for all representations should be kept the same, say, N=48. With $Z(I_T)$ = 29 and $Z(C)$ = 4, Table I tabulates $Z(I_P)$, $Z(F_P)$, $Z(F_T)$ and different ranges of the period.

Fig. 1 shows the maximum relative error of the service rate using different expressions. Normally, 1% relative error of the service rate induced by the expression is acceptable in a scheduling system. Note that the proposed scheme has a higher accuracy.

B. General Formula to Calculate the Number of Bits by Using the Proposed Representation

In this section, we shall derive the formula for computing the number of bits required to meet the maximum relative error using our proposed expression. Denote $\varepsilon_{max}(r)$ as the maximum relative error of the service rate. Let

$$\beta=\left\lceil \log_2\frac{r_{\max}}{r_{\min}}\right\rceil \qquad (20)$$

To maintain the same accuracy between the timestamp and the period, the number of bits to represent the fractional parts should be the same. Thus,

$Z(F_P)=Z(F_T)$

TABLE I
RANGE OF THE PROPOSED EXPRESSION

Range #	P_{min}	P_{max}	$Z(I_P)$	$Z(F_P)$	$Z(F_T)$
0000	1	2	1	7	7
0001	2	4	1	7	7
0010	4	8	3	6	6
0011	8	16	3	6	6
0100	16	32	5	5	5
0101	32	64	5	5	5
0110	64	128	7	4	4
0111	128	256	7	4	4
1000	256	512	9	3	3
1001	512	1024	9	3	3
1010	1024	2048	11	2	2
1011	2048	4096	11	2	2
1100	4096	8192	13	1	1
1101	8192	16384	15	0	0
1110	16384	32768	15	0	0
1111	32768	155500	15	0	0

To perform modular comparison without ambiguity, (4) must be met, and the minimum number of bits is

$$Z(I_T)=\left\lceil \log_2\frac{L_{\max}}{L_{\min}}+\log_2\frac{r_{\max}}{r_{\min}}\right\rceil+1$$

Range number C determines the range of the period as follows:

$$2^{C_i}\le P_i<2^{C_i+1}$$

Then, the number of bits to represent C is

$$Z(C)=\lceil\log_2\beta\rceil-1 \qquad (21)$$

It can be shown that the minimum number of bits to represent I_P, F_P and F_T such that the relative error is within the maximum relative error $\varepsilon_{max}(r)$ is

$$Z(I_P)+Z(F_P)+Z(F_T)=M \qquad (22)$$

where

$$M=\max\{2\cdot\lceil-\log_2\varepsilon_{\max}\rceil,\lceil\log_2\beta\rceil\}$$

The range number of P_i can be computed as follows:

$$C_i=\min\{\lceil\log_2 P_i\rceil-1,\ 2^{Z(C)}-1\} \qquad (23)$$

Then, the number of bits of the fractional parts is

$$Z(F_{P_i})=Z(F_{T_i})=\begin{cases}\left\lceil\dfrac{M-\lceil\log_2(P_i-2^{C_i})\rceil}{2}-1\right\rceil & \text{if } P_i<2^{C_i+1}\\ 0 & \text{if } P_i\ge 2^{C_i+1}\end{cases} \qquad (24)$$

Thus, the number of bits of I_P is

$$Z(I_{P_i})=M-2\cdot Z(F_{P_i}) \qquad (25)$$

Again, consider the previous packet scheduling system ***S***, in which the relative error of the service rate is required to lie within 1%. Using the fixed-point expression, it takes at least 60 bits to meet the requirement of the system (i.e. $\varepsilon_{max}(r)$, L_{max}, L_{min}, r_{max} and r_{min}); alternatively, using the floating-point expression requires at least 52 bits. However, it only requires

47 bits to satisfy the same system requirement by employing our proposed representation. Obviously, the proposed expression is more efficient.

C. *Implementation Issues*

We have proved that the proposed expression uses fewer bits than the fixed-point and floating-point expression to represent the timestamp and period with the same accuracy of the service rate, thus saving memory in terms of off-chip bandwidth and storage space.

With the fixed-point representation, it is very easy to perform operations such as addition and comparison. With the floating-point expression, we need to first shift mantissa of both timestamp and period, and then perform addition or comparison. The computational complexity of the proposed expression lies between them. The proposed expression uses additional operations to adjust the decimal point by range number as compared to the fixed-point expression. This can be readily realized by hardware with some extra logic operations.

IV. The Proposed Shifting Expression for Cell-Based (ATM) Schedulers

The cell-based scheduler can be considered as a special case of the packet-based scheduler, with $L_{max}=L_{min}$= constant. Equations derived for the fixed-point, floating-point and our proposed shifting fixed-point representations still hold. Note that in this case $log_2(L_{max}/L_{min}) = 0$ and $\underline{\Phi_i} = P_i$.

Consider an ATM scheduling system that is required to support service rates from 4 Kbits/s to 622 Mbits/s. With cell size of 53 bytes, $Z(I_T)$ = 19. With N=38, comparable results similar to the ones shown in Fig. 1 are obtained for each respective expression, implying that the proposed expression has a higher accuracy for cell-based schedulers too.

V. Conclusion

In this paper, we have developed a new expression, which can be implemented in hardware for high-speed switches, to represent timestamp and period in packet-based and cell-based scheduling system. It is applicable to any timestamp-based scheduler.

In comparison with the normal fixed-point and floating-point representation, the proposed expression can achieve better performance in terms of the number of bits and accuracy. If the number of bits for all expressions is kept the same, the proposed representation has a smaller relative error than those of the other two. In other words, if the relative error is kept the same for all expressions, our representation uses fewer bits than others, thus saving system memory and indirectly reducing latency. We have also derived the formula to calculate the minimum number of bits to represent the timestamp and period that meets the system requirement (i.e. $\varepsilon_{max}(r)$, L_{max}, L_{min}, r_{max} and r_{min}).

References

[1] A. Varma and D. Stiliadis, "Hardware Implementation of Fair Queuing Algorithms for Asynchronous Transfer Mode Networks" *IEEE Communication Magazine*, December 1997, pp. 54-68

[2] D.C. Stephens, J.C.R. Bennett and Hui Zhang, "Implementation Scheduling Algorithms in High-Speed Networks" *IEEE Journal on Selected Areas in Communications*, Vol.17, No.6, June 1999, pp. 1145-1158

[3] J.C.R. Bennett and Hui Zhang,"WF^{2}Q: Worst-case fair weighted queuing", *Proc. IEEE INFOCOM'96*, San Francisco, CA, pp. 120-128

[4] J.L. Rexford, A.G. Greenberg, and F.G. Bonomi, "Hardware-efficient fair queuing architecture for high-speed networks", *IEEE INFOCOM'96*, San Francisco, CA, pp. 638-646

[5] G.R. Wright and W.R. Stevens, *TCP/IP Illustrated Volume 2: The Implementation*, Reading, MA: Addison-Wesley, 1995, pp. 807-812

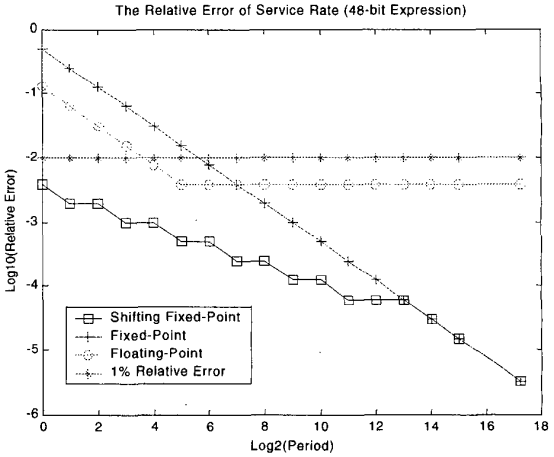

Fig. 1 The comparison of various expressions with 48 bits in terms of the maximum relative error