ABSTRACT

AN EFFICIENT EXPRESSION OF THE TIMESTAMP AND PERIOD IN PACKET- AND CELL-BASED SCHEDULERS

by
Dong Wei

Scheduling algorithms are implemented in hardware in high-speed switches to provide 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, which include SCFQ, WFQ, WF2Q and WF2Q+, 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 thesis, we propose a new expression for 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 expressions, when the size is fixed, the proposed expression has a better accuracy.

AN EFFICIENT EXPRESSION OF THE TIMESTAMP AND PERIOD IN PACKET- AND CELL-BASED SCHEDULERS

by
Dong Wei

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 2001

APPROVAL PAGE

AN EFFICIENT EXPRESSION OF THE TIMESTAMP AND PERIOD IN PACKET-BASED AND CELL-BASED SCHEDULER

Dong Wei

Dr. Nirwan Ansari, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Jianguo Chen, Thesis Co-Advisor — Date
Technical Staff of Bell Labs, Lucent Technologies

Dr. John Carpinelli, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Dong Wei

Degree: Master of Science in Electrical Engineering

Date: January 2001

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2001

- Bachelor of Engineering in Electrical Engineering,
  Tsinghua University, Beijing, PRC, 1991

Major: Electrical Engineering

This work is dedicated to

my beloved family

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Prof. Nirwan Ansari, who not only served as my research supervisor, providing valuable resources, insight and intuition, but also constantly gave me technical support and encouragement. I would also like to thank Dr. Jianguo Chen for his capital ideas and great support. Special thanks are given to Prof. John Carpinelli for participating in my committee.

In addition, I also wish to thank my wife for her encouragement and assistance over the years.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

PREFACE

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 Kbps and as high as 2.4Gbps (OC48) and 10Gbps (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 service interval determines the supportable rate in terms of range and accuracy. In this sense, it seems to be tempting to use larger size to represent them. However, the size of the timestamp and service interval 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 service interval 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 service interval. If the size of representation of timestamp and period is the bottleneck of the granularity problem, the proposed expression can also improve the granularity of the scheduler. This representation is generically applicable to any timestamp-based scheduling algorithm.

The rest of the thesis is organized as follows. Chapter 1 presents the background of the expression of timestamp and period of PFQ. Our proposed expression for packed-based schedulers is discussed in Chapter 2. The proposed expression is extended for cell-based (ATM) schedulers in Chapter 3. Chapter 4 presents the performance analysis of delay and delay jitter induced by the proposed expression. The implementation issue is discussed in Chapter 5.

CHAPTER 1

BACKGROUND: THE EXPRESSIONS OF TIMESTAMP AND SERVICE INTERVAL IN TIMESTAMP-BASED SCHEDULERS

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.

1.1 Notation

$Z(.)$ – the number of bits

$\overline{P}$ – the idealized period with infinite bit expression

P – actual period representation with finite bit expression

$\overline{r}$ – 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

$B_i(\alpha)$ – i^{th} bit of α (0 or 1)

L – the packet size

N – total number of bits of timestamp and period

n – total number of sessions of the scheduler

1.2 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:

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

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

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 Equations (1) and (2), the virtual finish time can be derived from the virtual start time and the packet service interval.

1.3 The Ideal Model of the Timestamp-based Scheduler

The ideal model of the timestamp-based scheduler is shown in Figure 1.

The switch creates a priority queue (FIFO) for each session. The timestamp of each packet is updated according to Equation (1) and (2). Before the packet enters the scheduler, its timestamp is the virtual start time. Only when the virtual system time is larger than the virtual start time, the packet is allowed to enter the scheduler. When it enters the scheduler, the corresponding virtual finish time is assigned as the timestamp. The scheduler services the packet with the smallest virtual finish time.

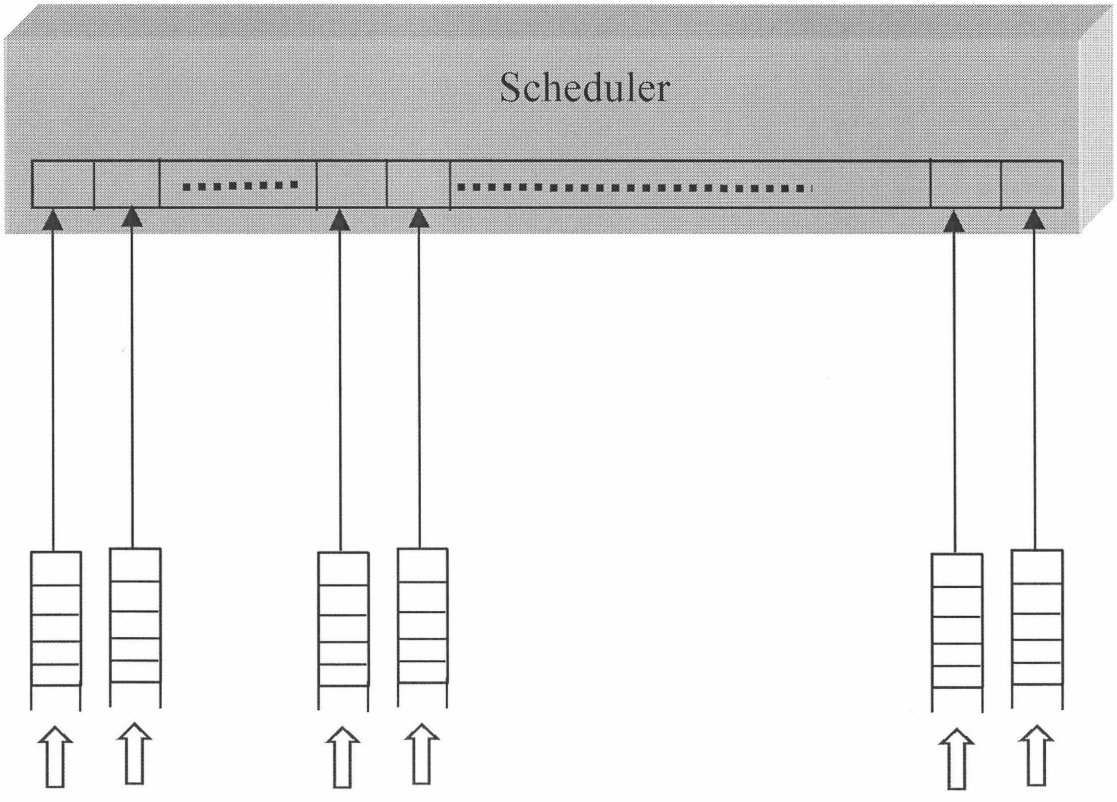

Figure 1 The ideal model for the timestamp-based scheduler

However, in this architecture, the algorithm complexity increases with the total number of session, i.e., O(n). In order to reduce the complexity, two grouping models are developed to replace the ideal model.

1.4 Generalized Grouping Architecture for Cell- and Packet-based Schedulers

The grouping architecture is employed to reduce the overall complexity of priority-queue management from the number of sessions to the number of groups. However, the downside is that this architecture has a granularity problem, which means that the scheduler can support only fixed service rate or fixed service interval service.

1.4.1 The Grouping Architecture for Cell-based Scheduler

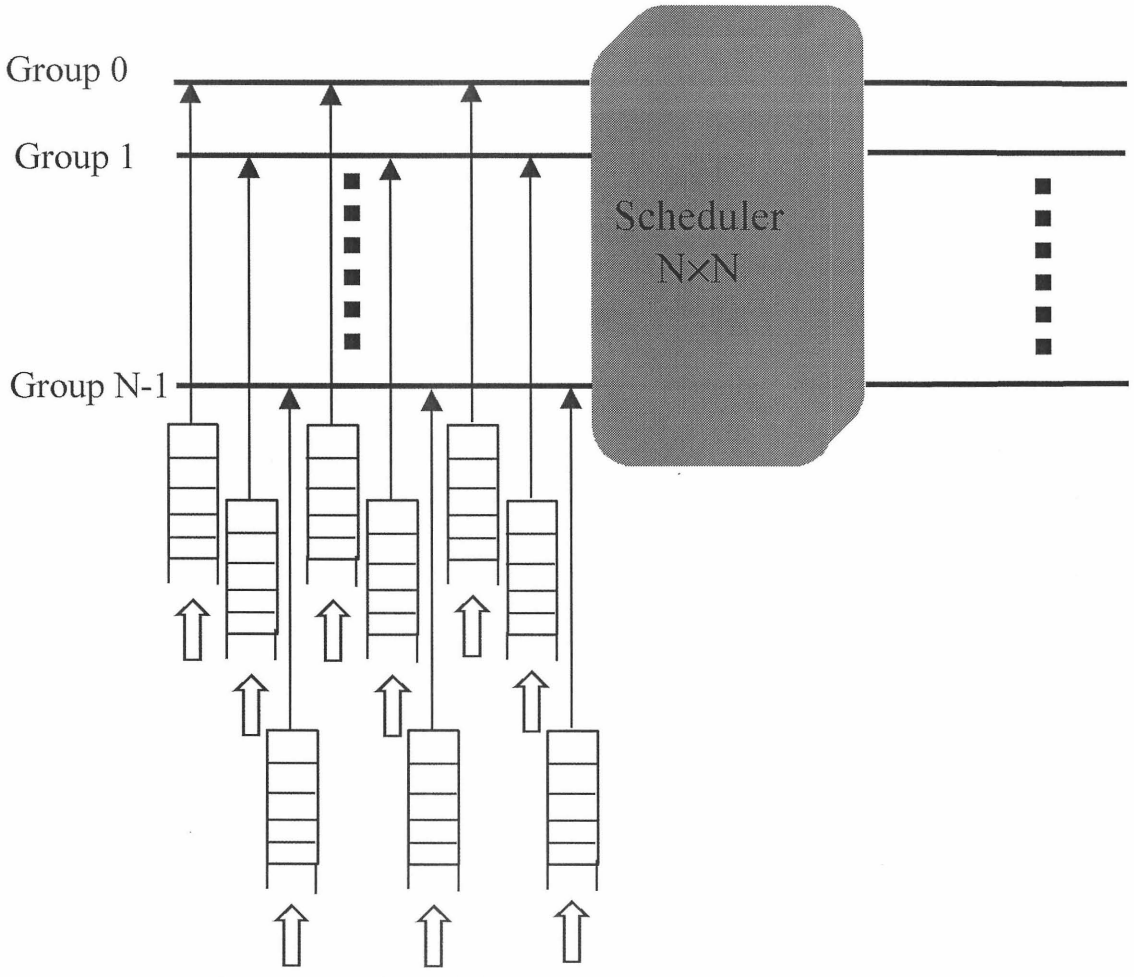

Figure 2 The grouping architecture for cell-based schedulers

The most difficult thing of implementing PFQ algorithms is that the complexity of maintaining the priority queue and computing the virtual time function grows as a function of the number of sessions sharing the link. To reduce the complexity of from the

number of sessions, the following restriction is introduced: at any time, only a fixed number of service rates are supported by the scheduler. All sessions with the same service rate are stored in one group. In each group, the session with the smallest virtual start time is placed in the scheduler.

1.4.2 The Grouping Architecture of Hierarchical Calendar Queue for Packet-based Scheduler

For a packet-based scheduler, a hierarchical calendar queue is introduced. First, when a packet comes, it is located in a group according to its service interval. Then it is put in a priority queue according to its service rate. At any time, only a fixed number of service intervals and service rates are supported by the scheduler.

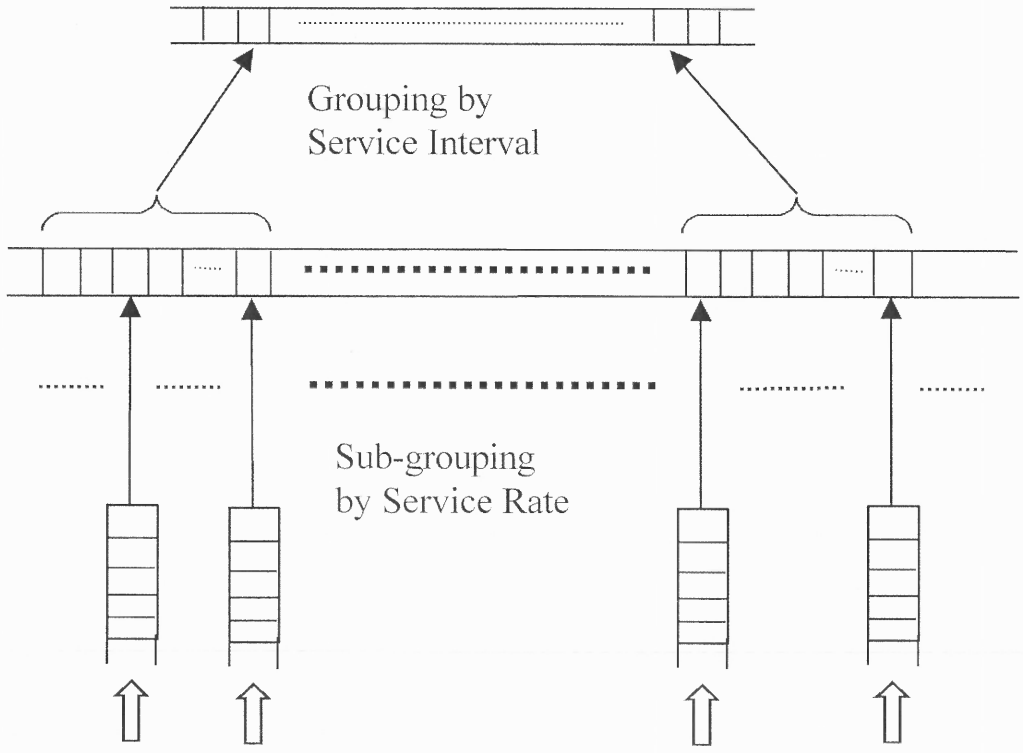

Figure 3 The grouping architecture of hierarchical calendar queue for packet-based schedulers

Note that by employing the grouping architecture, we reduce the algorithm complexity from the number of sessions ***O(n)*** to the number of groups ***O(1)***, where the number of groups is fixed for each scheduler at any time. However, the compromise is that the supportable service intervals and service rates are constrained.

1.5 Expression of the Timestamp and Period

1.5.1 The Period of Session

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}:

$$\tau = \frac{L_{\min}}{r_{LC}}.$$

Usually, the maximum supportable service rate is the same as the link rate of the scheduling system:

$$r_{max} = r_{LC}.$$

The system virtual time $V(t)$ is defined differently from one scheduling algorithm to another. $V(t)$ is expressed in the unit of time slot; for example, $V(t)$ is nothing but a counter in the virtual clock scheduling scheme, and it is incremented by one when a single fragment is sent out; in WFQ scheme, $V(t)$ is a piecewise linear function of the real time, and its slope of each time interval depends on the number of sessions receiving service and their service rates [1]. Modular comparison is used to select 1) packets eligible to enter the scheduler; 2) fragments which should get service. The period of session ***i*** is defined as follows:

$P_i = Period \quad of \quad Session \quad i = \frac{r_{\max}}{r_i}$, where r_i is the required service rate of session ***i***.

Let Φ_i^k denote 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,

$\Phi_i^k = \frac{L_i^k}{r_i}$, 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:

$$\Phi_i^k = \frac{L_i^k}{r_i} = \frac{L_i^k}{L_{\min}} \frac{L_{\min}}{r_{\max}} \frac{r_{\max}}{r_i} = \frac{L_i^k}{L_{\min}} \tau P_i$$

The normalized service interval is used in the system, which is:

$$\underline{\Phi_i^k} = \frac{\Phi_i^k}{\tau} = \frac{L_i^k}{L_{\min}} P_i \qquad (3)$$

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 normalized service interval can be calculated from Equation (3), and thus we can derive the virtual finish time from Equation (2); using the virtual finish time as the timestamp, the virtual start time can be similarly computed. Cells are allowed to enter the sorted-queue of the scheduler by comparing the system virtual time and their virtual start time. Cells are

scheduled for transmission based on the virtual finish time (the smallest virtual finish time first).

1.5.2 Modular Comparison

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. For example, given X=1011101, X[5:2]=0111. 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] is used to discern wraparound ambiguity [2][5]. The following condition must be satisfied:

$$|X - Y| < 2^n .$$

For example, when *X*=110 and *Y*=001, which represent 6 and 1, respectively. Since the above condition must be met, *Y*=*X*+011 rather than *X*=*Y*+101; in other words, *Y*>*X*. Thus,

the result of Modular_Comparison(X, Y) is *FALSE*, implying that X is not greater than Y because of wraparound.

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:

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

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 and 1/2, 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.

Theorem 1. The approximation of the relative error of the period equals the relative error of the service rate.

$$\hat{\varepsilon}(P_i) = \varepsilon(r_i)$$

Where $\hat{\varepsilon}(P_i) = \left| \frac{P_i - \overline{P_i}}{P_i} \right|$.

The proof may be found in Appendix A.

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 $\Phi_{\max}$, and the accuracy of the period is determined by both of the number of bits of the period and the accuracy of the timestamp.

1.6 The Fixed-Point Expression

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

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

Integer Part						Fractional Part				
1	1	1	1	0	0.	1	1	0		Timestamp T
			1	0	1.	0	0	1	1	Period P_i
		1	1	0	0.	1				Period P_j

$B_5(I_T)\, B_4(I_T)\, B_3(I_T)\, B_2(I_T)\, B_1(I_T)\, B_0(I_T) = 111100$
$B_2(F_T)\, B_1(F_T)\, B_0(F_T) = 110$
$T = 60.75$

$B_2(I_{Pi})\, B_1(I_{Pi})\, B_0(I_{Pi}) = 101$
$B_3(F_{Pi})\, B_2(F_{Pi})\, B_1(F_{Pi})\, B_0(F_{Pi}) = 0011$
$P_i = 5.1875$

$B_3(I_{Pj})\, B_2(I_{Pj})\, B_1(I_{Pj})\, B_0(I_{Pj}) = 1100$
$B_0(F_{Pj}) = 1$
$P_j = 12.5$

Figure 4 The fixed-point expression

Figure 4 illustrates the fixed-point expression of an example. The accuracy of the period is determined by both of the number of bits used to represent the period and the accuracy of the timestamp. P_i has more fractional bits than the timestamp, while P_j has less fractional bits than the timestamp. In this example, the relative errors of P_i and P_j are:

$$\hat{\varepsilon}(P_i) = \frac{2^{-4}}{Pi} ,$$

$$\hat{\varepsilon}(P_j) = \frac{2^{-2}}{P_j} .$$

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:

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

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

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

With the fixed-point expression, the representation of P includes the integer part and fractional part. The value of P can be obtained as follows:

$$P = \sum_{i=0}^{Z(I_P)} B_i(P) \cdot 2^i + \frac{\sum_{j=0}^{Z(F_P)-1} B_j(P) \cdot 2^j}{2^{Z(F_P)}} \qquad (11)$$

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,

$Z(I_T) = 29$

$Z(I_P) = 18$

With $Z(F_T) = Z(F_P) = 0, 2, 4,6,8$, respectively (i.e., $N = 47, 51, 55, 59$ and 63), the relative error of the service rate is shown in Figure 5.

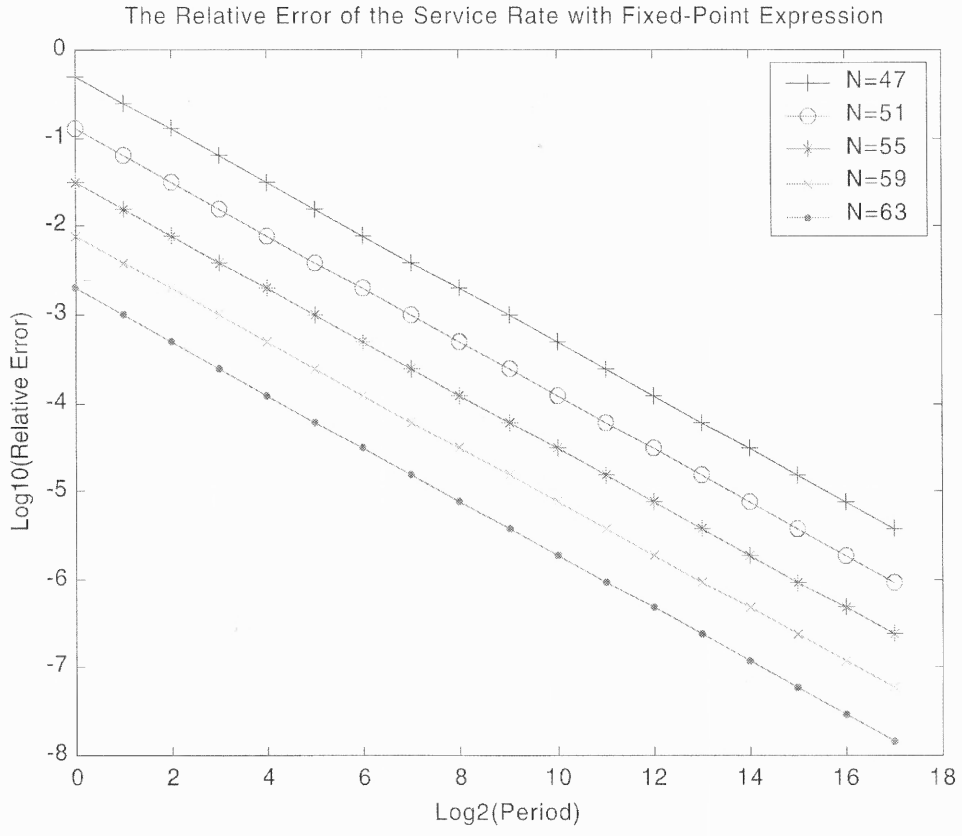

Figure 5 The relative error using the fixed-point expression

1.7 The Floating-Point Expression

Similarly, in order to achieve unambiguous modular comparison, the following inequality must be satisfied:

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

Let δ be the number of additional bits:

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

To maintain the same degree of accuracy of the timestamp,

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

Thus,

$$Z(E_T) = \log_2 Z(M_T) \ , \qquad (15)$$

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

The exponent part of the timestamp and period of an example is illustrated in Figure 6, where

$Z(M_T) = 10$,

$Z(M_P) = 7$, and

$Z(E_T) = Z(E_P) = 3$.

Mantissa (M_T & M_P)										Exponent (E_T & E_P)			
1	0	0	0	0	0	0	0	1	1	1	1	0	Timestamp
			1	0	0	0	0	0	1	1	0	1	Period P_i
			0	0	1	0	0	0	1	0	0	1	Period P_j

Figure 6 The floating-point expression

Including the hidden '1' (default of the IEEE expression), the binary equivalent of the expression is shown in Figure 7:

Integer Part							Fractional Part						
1	1	0	0	0	0	0.	0	0	1	1			Timestamp
	1	1	0	0	0	0.	0	1					Period P_i
					1	0.	0	1	0	0	0	1	Period P_j

Figure 7 The binary equivalent of the floating-point expression

With reference to the accuracy of the timestamp, P_i is 2-bit short in acquiring the same accuracy of the timestamp, while P_j has two extra redundant bits.

Again, the total number of bits to represent the timestamp and period is

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

$\overbrace{xxxxxxxx}^{1+E_T}\ .\ \overbrace{xxxxxx}^{Z(M_T)-E_T}$ *Timestamp*

$\overbrace{xxxxxx}^{1+E_P}\ .\ \overbrace{xx}^{Z(M_P)-E_P}$ P_i

$\overbrace{xx}^{1+E_P}\ .\ \overbrace{xxxxxx}^{Z(M_P)-E_P}$ P_j

Figure 8 The binary equivalent of the floating-point expression for the timestamp and period

Figure 8 illustrates how to fix the position of the decimal point in the binary equivalent of the timestamp and period. The size of the binary equivalent of the timestamp and period are $Z(M_T)+1$ and $Z(M_P)+1$, respectively. The values of E_T and E_P determine how many bits the decimal point should shift from the position just after the most significant bit. Let $\lambda(P)=Z(M_P)-E_P$ 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 relative error can be expressed as:

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

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:

$$\hat{\varepsilon}_{\max}(P)=\begin{cases}\dfrac{2^{-1}}{2^{Z(M_P)}} & if \quad \lambda(P)\le\lambda_{\min}(T)\\ \dfrac{2^{\lambda(P)-\lambda(T)-1}}{2^{Z(M_P)}} & if \quad \lambda(P)>\lambda_{\min}(T)\end{cases} \tag{19}$$

Consider the same packet scheduling system ***S*** as before.

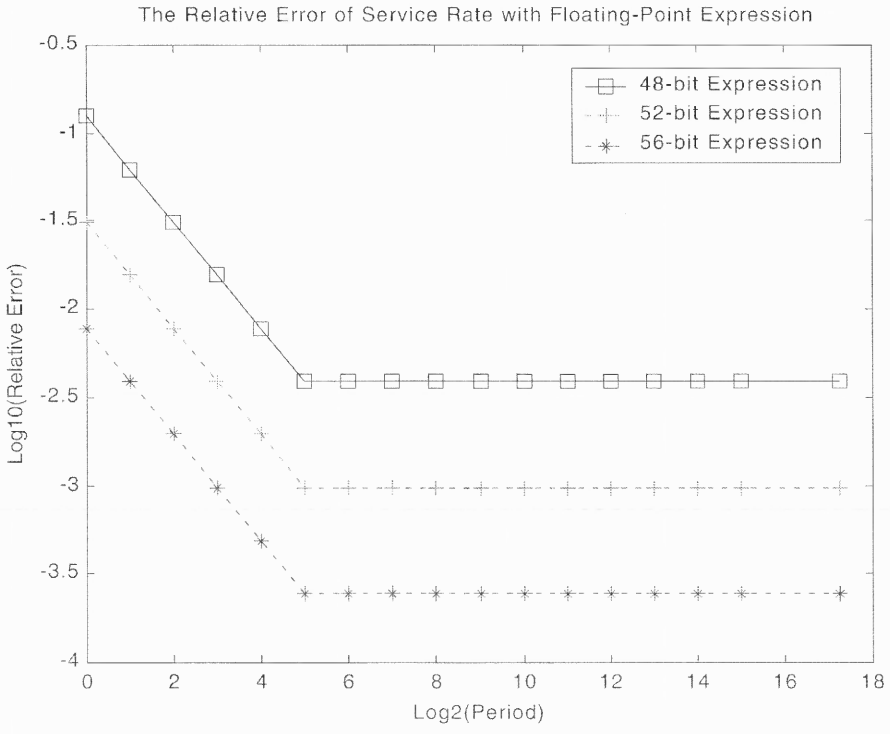

Figure 9 The maximum relative error of the floating-point expression A

Using Equations (15) and (16), we have

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

With ($Z(M_T)$ = 31, $Z(M_P)$ = 7), ($Z(M_T)$ = 33, $Z(M_P)$ = 9) and ($Z(M_T)$ = 35, $Z(M_P)$ = 11) respectively (i.e., N = 48, δ=2; N = 52, δ=4; N = 46, δ=6) , the upper bound of the relative error of the service rate using the floating-point expression is shown in Figure 9.

Note that in Figure 9, when Log_2(Period) 5, the maximum relative error is a horizontal line, implying that the relative error is independent with the period, which is desirable in a scheduler. Now we reallocate the number of bits, with N=39, δ=0; N=43, δ=2; N=47, δ=4; N=51, δ=6; and N=55, δ=8, the maximum relative error of the service rate is shown in Figure 10.

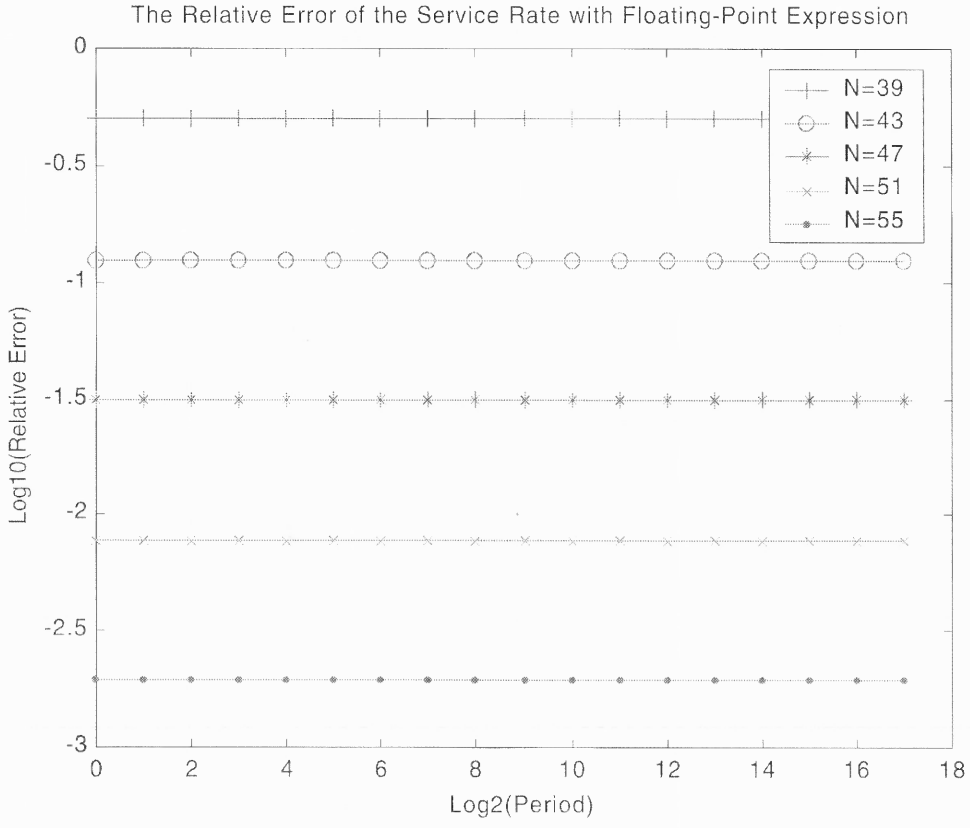

Figure 10 The maximum relative error of the floating-point expression B

CHAPTER 2

THE PROPOSED EXPRESSION FOR PACKET SCHEDULERS

As shown in Figure 10, using the floating-point expression, the maximum relative error is consistent in the whole range of period, i.e., it is independent of the service rate. This is a desirable property for a scheduling system. However, the total number to represent the period and timestamp ***N*** and the implementation complexity of floating-point operation are still not good enough. By employing the modular comparison, we propose a new expression by locating the decimal point according to the range number C of a session. Each range number C determines the decimal point, and thus the range of the period. Let $\underline{C_i}$ denote the category number for session ***i***.

2.1 The Compressed Timestamp Expression

For session ***i***, the timestamp-based algorithms have the globally bounded timestamp (GBT) [2]:

$$S_i(t)+L_i^k/r_i \geq V(t) \geq S_i(t)-L_i^k/r_i\ ,\ \textit{if queue of session } \boldsymbol{i} \textit{ is not } \textbf{empty} \qquad (20)$$

$\Phi_i^k = \frac{L_i^k}{r_i}$, thus (20) can be re-written as:

$$V(t) + \Phi_i^k \geq S_i(t) \geq V(t)- \Phi_i^k,\ \textit{if queue of session } \boldsymbol{i} \textit{ is not } \textbf{empty} \qquad (21)$$

For example, consider session ***i*** which requires period $\boldsymbol{P_i}$=12.5. If $L_i^k = 2L_{min}$, then, $\underline{\Phi_i^k}$= $\boldsymbol{2P_i}$=25. Assume at the moment of V(t)= 200.75, this packet is allowed to enter the queue. Then $S_i(t)=200.75$ *and* $F_i(t)=225.75$.

Note that only those bits marked 'x' (bit 4 to bit –1) in $S_i(t)$ *and* $F_i(t)$ are different from those in $V(t)$. The number of 'x' depends on the number of significant bits of $\boldsymbol{P_i}$ and packet size L_i^k as well. Intuitively, we only need to save those bits marked 'x' as the

timestamp, and compare this timestamp with those corresponding bits of the virtual system time.

0	0	1	1	0	0	1	0	0	0	.	1	1	0	0	$V(t)$
0	0	0	0	0	0	1	1	0	0	.	1	0	0	0	P_i
0	0	1	1	0	0	1	0	0	0	.	1	1	0	0	$S_i(t)$
0	0	1	1	0	1	0	1	0	1	.	0	1	0	0	$F_i(t)$
9	8	7	6	5	4	3	2	1	0	.	-1	-2	-3	-4	
					x	x	x	x	x	.	x				

Figure 11 The relationship of V(t), $S_i(t)$, $F_i(t)$ and P_i

The ***compressed timestamp expression*** is defined as follows:

If the period of session ***i*** is P_i and it is expressed with bits from bit ***m*** to ***n***, the timestamp of packet ***k*** of this session can be expressed with bits from bit ***m+l+1*** to ***n+l***, where $l_i^k = \left\lfloor \log_2 \frac{L_i^k}{L_{\min}} \right\rfloor$. One extra bit in the timestamp is use to discern the wraparound ambiguity.

By using the compressed timestamp expression, the size of the timestamp can be reduced dramatically, especially for those sessions requiring higher service rates. This is shown in the following example.

2.2 The Compressed Period Expression with Fixed-size

From Figure 10 in Section 1.5, we see that a consistent maximum relative error of the service rate could be achieved. If all bits of the period can determine the accuracy, implying that the timestamp maintains the same degree of the accuracy, then the maximum relative error of this expression is

$$\varepsilon_{\max}(r) = \hat{\varepsilon}_{\max}(P) = 2^{-(Z(P)+1)}. \qquad (22)$$

Let $\varepsilon^*(r)$ denote the acceptable relative error of the service rate. Select the number of bits of the period Z(P) such that the maximum relative error is less than $\varepsilon^*(r)$, then the relationship between Z(P) and $\varepsilon^*(r)$ is as follows:

$$Z(P) = \lfloor -\log_2 \varepsilon^*(r) \rfloor \qquad (23)$$

To reduce the relative error of the expression, we generate P by rounding off $\overline{P}$.

$$\begin{array}{ll} \overline{P} & \underbrace{1xxxxx}_{Z(P)}xxxxxxx.xxxxx \\ + & \underbrace{0000001}_{Z(P)}000000.00000 \\ \hline P & 1xxxxx0000000.00000 \end{array}$$

In a scheduler, period P must be larger than 1. Therefore, by using the hidden '1', which is used in IEEE floating-point expression, one more bit can be saved. P' denotes the compressed period expression. Then Equation (23) can be rewritten as:

$$Z(P') = \lfloor -\log_2 \varepsilon^*(r) \rfloor - 1 \qquad (24)$$

For example, if $\varepsilon^*(r)$=1%, 5 bits are needed to represent the period. The relative error with 6 bits (including the hidden '1') expression < 2^{-7} = 0.78%. Note that in order to discern the wraparound ambiguity, Z(T) should be one bit larger than Z(P). To take the hidden '1' into consideration, the number of bits to represent the timestamp can also be calculated as

$$Z(T')=Z(P')+2 \ . \qquad (25)$$

2.3 Range Number

Note that if the timestamp and the period maintain the same degree of accuracy, thus the maximum relative error can be consistent over the whole range of service rates. In order to generate the compressed timestamp, a range number is also needed to locate the decimal point of the timestamp and period. For example, when C_i=4, then $2^4 \le P_i < 2^5$. Thus the maximum range number can be computed as

$$C_{max} = \lceil \log_2(r_{max}/r_{min}) \rceil. \quad (26)$$

Thus, the number of bits to represent the range number

$$Z(C) = \lceil \log_2 C_{max} \rceil. \quad (27)$$

P_i can be obtained from P_i' by right shifting the decimal point C_i bits. The timestamp can be similarly obtained with its compressed form and packet size. With these properties, 1) we calculate the period for each session and save it in the compressed form, 2) calculate the timestamp and save it in the compressed expression, 3) reconstruct the timestamp to the complete form from the compressed expression, and 4) the scheduler sorts packets by comparing their complete timestamps.

2.4 Generating and Reconstructing the Expression

2.4.1 Truncating Operation

The truncating operation is used to generate the proposed expression of the timestamp and period. The decimal point is not saved.

1. Integer A= **Truncating (W, x+y, x)**

For example, $W = 100111000.001101$

A= **Truncating (W, 6, 1)** = $(011101)_2$

B= **Truncating (W, 4, -2)** = $(1100000)_2$

2. Integer A= **Truncating (W, x)**

For example, $W = 100111000.001101$

A= **Truncating (W, 6)** = $(100111)_2$

B= **Truncating (W, 12)** = $(100111000001)_2$

2.4.2 The Size of the Virtual System Time

$$V(t) \qquad \overbrace{\underbrace{xxxxxxxx}_{Z(I_V)}.\underbrace{xxxxx}_{Z(F_V)}}^{Z(V)} \qquad x \in \{0,1\}$$

In order to discern the wraparound ambiguity, Inequality (4) still must hold. Let

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

To maintain the same degree of accuracy as the when the service rate is high, we need to add Z(P)-1 bits after the decimal point, i.e.,

$$Z(F_V) = Z(P) - 1$$

$$\therefore \qquad Z(V) = \left\lceil \log_2 \frac{r_{\max}}{r_{\min}} + \log_2 \frac{L_{\max}}{L_{\min}} \right\rceil + Z(P) \qquad (28)$$

2.4.3 Generating the Range Number

C_i determines the position of the decimal point. Define C_i as the number of bits between the decimal point and '1'.

$$P_i = 1\underbrace{xxxxxxxx}_{C_i}.\underbrace{xxxxx} \qquad x \in \{0,1\}$$

$$\therefore \quad 2^{C_i} \le P_i \le 2^{C_i+1}$$

Thus, $C_i = \lfloor \log_2 P_i \rfloor$ (29)

2.4.4 Generating the Compressed Expression and Reconstructing the Complete Expression of the Period

To round off the period, let $A = Truncating(\overline{P_i}, Z(P)+1)+1$

$P_i = Truncating(A, Z(P))$ (30)

The compressed period expression is saved as an integer and the range number determines the position of the decimal point, just like the mantissa part and exponent part in IEEE floating-point expression.

$P_i' = (P_i - 2^{C_i}) \times 2^{Z(P')-C_i}$ (31)

$P_i = 2^{C_i} + P_i' \times 2^{C_i - Z(P')}$ (32)

For example, let $Z(P') = 5$ and session ***i*** $P_i = 1101.10$, then $C_i = 3$ and $P_i' = 10110$; for session ***j***, $C_j = 9$ and $P_j' = 01101$, then $P_j = 1011010000.000$.

2.4.5 Generating the Compressed Expression and Reconstructing the Complete Expression of the Timestamp

Without loss of generality, define the virtual finish time as the timestamp and let

$T_i^k = F_i^k(t) = V(t) + \underline{\Phi_i^k}$, and $\beta = \left\lfloor \frac{L_i^k}{2L_{\min}} \right\rfloor + C_i$.

Thus, the compressed virtual finish time can be calculated as

$T_i^k' = Truncating(T_i^k, \beta + Z(T') - 1, \beta) + 1$. (33)

The compressed timestamp is assigned to each packet and stored in the system. When packets are selected to enter the scheduler, we need to compare the timestamp with

the corresponding part of the virtual system time. In order to avoid some problems that could be caused by truncating the low-order bits, we add '1' to the compressed timestamp.

$$V(t) \qquad xxxxxx\underbrace{xxxxxx}_{Z(T')}xxxx.xxxxxx \qquad x \in \{0,1\}$$

$$T_i^{k\prime} \qquad \underbrace{xxxxx}_{Z(T')}$$

When packets are in the scheduler, the scheduler compares their timestamp, and the packet with the smallest timestamp will receive service. In order to compare the timestamp among packets with different service rate, we need to reconstruct the timestamp – from the compressed one to the complete form. We fill '1' in the low-order bits and copy high-order bits from the virtual system.

$$T_i^k = V[Z(V)-1 : Z(T')+\beta]\times 2^{Z(T')+C_i} + T_i^{k\prime}\times 2^{C_i} + \underbrace{1111....111}_{\beta-1}. \qquad (34)$$

We shall explain this method in detail in the following example.

Note:

1. To select an eligible packet to enter the scheduler, its timestamp needs to be compared with the virtual system time. By employing modular comparison, we only need to compare the compressed timestamp with the corresponding parts in the virtual system time.
2. To schedule the packet with the smallest timestamp, all timestamps of those packets in the scheduler need to be compared. The compressed timestamps need to be reconstructed before this comparison.

2.5 Example

For example, suppose that the service rate of the packet-based scheduling system S ranges from 4kbps to 622Mbps and the packet size is in the range of 40 bytes to 64 Kbytes. 1% relative error of service rate is considered acceptable.

With Equation (23) and (25), we can get Z(P') = 5 and Z(T') = 7.

$$Z(V) = \left\lceil \log_2 \frac{r_{\max}}{r_{\min}} + \log_2 \frac{L_{\max}}{L_{\min}} \right\rceil + Z(P) = \left\lceil \log_2 \frac{622\times10^6}{4\times10^3} + \log_2 \frac{64\times10^3}{40} \right\rceil + 6$$

$= 34.$

The form of *V(t)* is as follows:

$$V(t) \qquad \underbrace{xxxxxxxxxxxxxxxxxxxxxxxxxxxxx}_{29}.\underbrace{xxxxx}_{5} \qquad x \in \{0,1\}$$

The number of bits to represent range number C can be calculated as

$C_{max} = \lceil \log_2(r_{max}/r_{min}) \rceil = 18$

$Z(C) = \lceil \log_2 C_{max} \rceil = 5$

The total number of bits to represent period and timestamp is

$N = Z(C) + Z(P') + Z(T')$ (35)

$= 5+5+7 = 17$

The relationship between the range number and corresponding range of the period and the maximum relative error are shown in Figure 12.

For example, session ***i*** requires the service rate of 100Mbps. Thus the period

$\overline{P_i} = 622/100 = (6.22)_{10} = (110.00111....)_2.$

With Equation (29), the range number $C_i = (00010)_2 = (2)_{10}$.

With Equation (30) and (31), we can have

$P_i = (110.010)_2$ and $P_i' = (10010)_2$

Range # C	P_{min}	P_{max}	Z(P)	Z(T)	N	ε_{max}
00000	1	2	5	7	17	2^{-7}
00001	2	4	5	7	17	2^{-7}
00010	4	8	5	7	17	2^{-7}
00011	8	16	5	7	17	2^{-7}
00100	16	32	5	7	17	2^{-7}
00101	32	64	5	7	17	2^{-7}
00110	64	128	5	7	17	2^{-7}
00111	128	256	5	7	17	2^{-7}
01000	256	512	5	7	17	2^{-7}
01001	512	1024	5	7	17	2^{-7}
01010	1024	2048	5	7	17	2^{-7}
01011	2048	4096	5	7	17	2^{-7}
01100	4096	8192	5	7	17	2^{-7}
01101	8192	16384	5	7	17	2^{-7}
01110	16384	32768	5	7	17	2^{-7}
01111	32768	65356	5	7	17	2^{-7}
10000	65356	130712	5	7	17	2^{-7}
10001	130712	155500	5	7	17	2^{-7}

Figure 12 Ranges of the proposed expression for packet-based scheduler S

Suppose the k^{th} packet of session i, whose size is 1 kbytes, enters the scheduler when V(t)= $(1000000000100001101110101101 0.10011)_2$. The ideal normalized service interval of this packet is

$$\underline{\Phi_i^k} = \frac{L_i^k}{L_{\min}} \overline{P_i} = \frac{1000}{40} 6.22 = (155.5)_{10} = (10011011.1)_2$$

Thus the complete virtual finish time of this packet is

$$T_i^k = V(t) + \underline{\Phi_i^k} = (1000000000\,10000110111111110110.00011)_2 .$$

With 5-bit period and 7-bit timestamp compressed expression

$$\beta = \left\lfloor \frac{L_i^k}{2L_{\min}} \right\rfloor + C_i = \left\lfloor \frac{1000}{2\times 40} \right\rfloor + 2 = 6$$

$$\underline{\Phi_i^k} = \frac{L_i^k}{L_{\min}} P_i = (\frac{1000}{40})_{10} (110.010)_2 = (10011100.000)_2 = (156)_{10}$$

$$\underline{\Phi_t^k} = \underbrace{10011100}_{6}.010$$

$$V(t) = 1000000000100001101\underbrace{1010110}_{7}10.10011$$

$$\therefore \quad \underline{\Phi_t^k} = 100111$$

With Equation (33), the compressed timestamp $T_t^k{}' = (1111110)_2$.

Before its can be compared with other timestamps, it has to be reconstructed to the complete form.

$$T_t^k = 1000000000100001101\underbrace{1111110}_{7}1111111$$

After positioning the decimal point with C_t, the complete timestamp is

$$1000000000100001101\underbrace{1111110}_{7}11.11111$$

Compared with the ideal timestamp $1000000000\,1000011011\,111110110.00011$, the reconstructed is always a bit larger, implying an extra latency induced by the compressed expression. To do the modular comparison with the system virtual, we only need to compare the marked 7 bits with the corresponding bits in the system virtual time.

With Z(C) =5, (Z(T')=3, Z(P')=1); (Z(T')=5, Z(P')=3); (Z(T')=7, Z(P')=5); and (Z(T')=9, Z(P')=7) respectively (i.e. N=9; N=13; N=17 and N=21), The maximum relative error of the service rate with the compressed expression is shown in Figure 13.

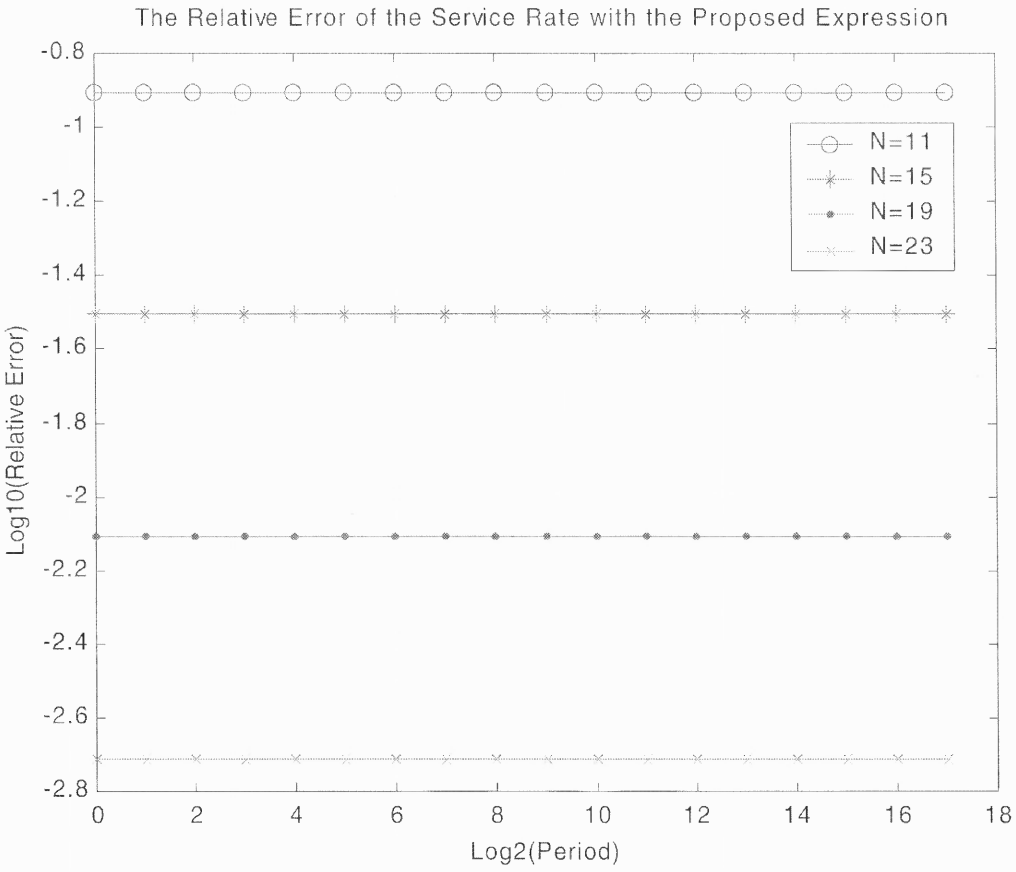

Figure 13 The maximum relative error of the service rate with the proposed expression

CHAPTER 3

THE PROPOSED EXPRESSION FOR CELL-BASED 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

$\log_2 \frac{L_{\max}}{L_{\min}} = 0$ and $\underline{\Phi_i} = P_i$.

Therefore, Equation (28) can be rewritten as

$$\therefore \qquad Z(V) = \left\lceil \log_2 \frac{r_{\max}}{r_{\min}} \right\rceil + Z(P) \qquad (36)$$

Equation (23)-(27) and (29)-(35) still hold, and $\beta = C_i$.

Suppose that the service rate of the cell-based scheduling system S' ranges from 4kbps to 622Mbps and the cell size is 40 bytes. Again, 1% relative error of service rate is considered acceptable.

With Equation (23) and (25), we can get Z(P') = 5 and Z(T') = 7.

$Z_{min}(V(t)) = \lceil \log_2(r_{max}/r_{min}) \rceil + 1$

$= \lceil -\log_2(622\times10^6/4\times10^6) \rceil + 1 = 19$

To maintain the same degree of the accuracy as the period when P is small, say $P=(1.10101)_2$, 5 bits have to be added after the decimal point. Thus $Z(V(t)) = 24$. The form of $V(t)$ is as follows:

$$V(t) \qquad \underbrace{xxxxxxxxxxxxxxxxxxx}_{19}.\underbrace{xxxxx}_{5} \qquad x \in \{0,1\}$$

The number of bits to represent range number C can be calculated as

$C_{max}=\lceil \log_2(r_{max}/r_{min}) \rceil=18$

$Z(C)=\lceil \log_2 C_{max} \rceil=5$

The total number of bits to represent period and timestamp is

$N= Z(C) + Z(P') + Z(T') =5+5+7=17$

The relationship between the range number and the corresponding range of the period and maximum relative error are shown in Figure 14.

Range # C	P_{min}	P_{max}	Z(P)	Z(T)	N	ε_{max}
00000	1	2	5	7	17	2^{-7}
00001	2	4	5	7	17	2^{-7}
00010	4	8	5	7	17	2^{-7}
00011	8	16	5	7	17	2^{-7}
00100	16	32	5	7	17	2^{-7}
00101	32	64	5	7	17	2^{-7}
00110	64	128	5	7	17	2^{-7}
00111	128	256	5	7	17	2^{-7}
01000	256	512	5	7	17	2^{-7}
01001	512	1024	5	7	17	2^{-7}
01010	1024	2048	5	7	17	2^{-7}
01011	2048	4096	5	7	17	2^{-7}
01100	4096	8192	5	7	17	2^{-7}
01101	8192	16384	5	7	17	2^{-7}
01110	16384	32768	5	7	17	2^{-7}
01111	32768	65356	5	7	17	2^{-7}
10000	65356	130712	5	7	17	2^{-7}
10001	130712	155500	5	7	17	2^{-7}

Figure 14 Ranges of the proposed expression for cell-based scheduler S'

For example, session ***i*** requires the service rate of 100Mbps. Thus the period

$\overline{P_i} = 622/100 = (6.22)_{10}=(110.00111....)_2$

With Equation (29), the range number $C_i=(00010)_2=(2)_{10}$.

With Equation (30) and (31), we can have

$P_i = (110.010)_2$ and $P_i' = (10010)_2$

Suppose the k^{th} packet of session ***i***, enters the scheduler when V(t)= $(1000011011101011010.10011)_2$. The ideal normalized service interval of this packet is

$$\underline{\Phi_i^k} = \overline{P_i} = (6.22)_2 = (110.00111....)_2$$

Thus the complete virtual finish time of this packet is

$$T_i^k = V(t) + \underline{\Phi_i^k} = (1000011011\ 101100000.11010)_2 . \quad (*)$$

With 5-bit period and 7-bit timestamp compressed expression

$$\underline{\Phi_i^k} = P_i = (110.010)_2$$

$$V(t) = 10000110111010\underbrace{11010.1001}_{7}1$$

With Equation (33), the compressed timestamp $T_i^k{}' = (0000110)_2$.

With Equation (34), the complete form of the timestamp can be reconstructed

$$T_i^k = = 10000110111010\underbrace{1000011}_{7}011$$

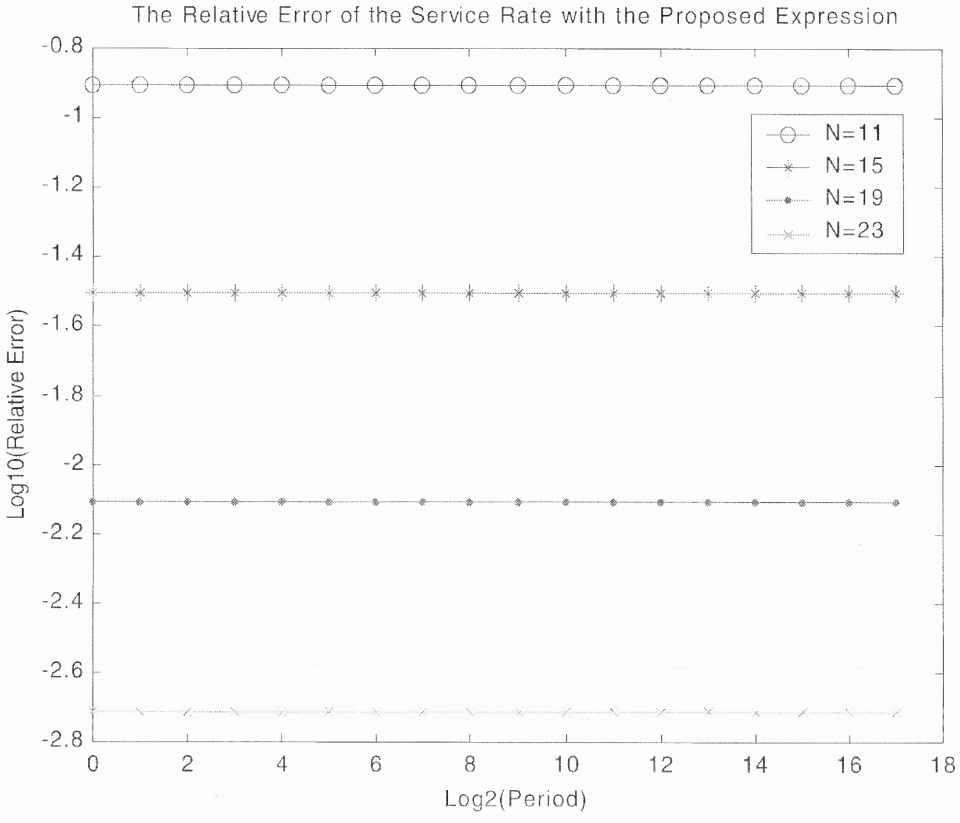

Figure 15 The maximum relative error of the service rate with the proposed expression for cell-based scheduler S'

After positioning the decimal point with C_l, the complete timestamp is

$$1000011011101010\underbrace{0000.110}_{7}11.$$

By modular comparison, the reconstructed 0000.110 is always a bit larger than the corresponding 7 bits in the ideal timestamp $100001101\,1111110110.00011$, implying an extra latency induced by the compressed expression.

With Z(C) =5, (Z(T')=3, Z(P')=1); (Z(T')=5, Z(P')=3); (Z(T')=7, Z(P')=5); and (Z(T')=9, Z(P')=7) respectively (i.e. N=9; N=13; N=17 and N=21), The maximum relative error of the service rate in compressed expression is shown in Figure 15.

CHAPTER 4

PERFORMANCE ANALYSIS

The proposed expression of the timestamp and period is explained in Chapter 2 and 3. It is shown that two kinds of latency could be caused by the proposed expression:

1) When a packet gets virtual system time as a timestamp, the compressed timestamp is added by '1' before it is saved. This will delay its entrance into the scheduler. $D_i^k(1)$ denotes the maximum value of this kind of latency for packet k of session i. The normalized value is defined as $\underline{D_i^k(1)} = \frac{D_i^k(1)}{\Phi_i^k}$.

2) When a packet enters the scheduler, in order to compare its timestamp with other packets, the compressed timestamp needs to be reconstructed by filling '1' in the low-order bits. This could cause packet transmission delay. $D_i^k(2)$ denotes the maximum value of this kind of delay. Similarly, the normalized value is defined as $\underline{D_i^k(2)} = \frac{D_i^k(2)}{\Phi_i^k}$.

Thus, the overall latency caused by the proposed expression is

$$D_i^k = D_i^k(1) + D_i^k(2), \qquad (37)$$

and

$$\underline{D_i^k} = \underline{D_i^k(1)} + \underline{D_i^k(2)}, \qquad (38)$$

For ***m***-bit expression of the period, the number of significant bits of the normalized service interval is also ***m*** bits, implying that $\underline{D_i^k(1)} = 2^{-(m-1)}$. Therefore, it can be computed as

$$\underline{D_i^k(1)} = 2^{-Z(P_i')} \qquad (39)$$

Similarly, $\underline{D_i^k(2)}$ can be computed as

$$\underline{D_i^k(2)} = 2^{-Z(P_i')} \qquad (40)$$

Therefore, the overall latency is

$$\underline{D_i^k} = 2^{-Z(P_i')+1} \qquad (41)$$

Define delay-jitter as the difference between the maximum delay and the minimum delay, without loss of generality, $\Delta D_i^k = D_i^k$ and $\underline{\Delta D_i^k} = \underline{D_i^k}$. Thus, from Equation (41), we have

$$\underline{\Delta D_i^k} = 2^{-Z(P_i')+1} \qquad (42)$$

Note that the delay and delay-jitter over the service interval are constant, implying that the delay and delay-jitter are proportional to the service interval. It seems reasonable.

$$\Delta D_i^k = D_i^k = \underline{D_i^k} \times \Phi_i^k = 2^{-Z(P_i')+1} \times \Phi_i^k \qquad (43)$$

For the example in section 2.5, $L_i^k = 1$ *kbytes* and required service rate is 100Mbps, i.e., P_i= 6.22. Thus, the service interval Φ_i^k is 8 ms. The maximum delay and delay-jitter can be calculated by Equation (43)

$$\Delta D_i^k = D_i^k = 2^{-4} \times \Phi_i^k = 0.5ms$$

With Z(C) =5, (Z(T')=3, Z(P')=1); (Z(T')=5, Z(P')=3); (Z(T')=7, Z(P')=5); and (Z(T')=9, Z(P')=7) respectively (i.e. N=9; N=13; N=17 and N=21), The normalized

delay and delay-jitter in the compressed expression are shown in Figure 16 and 17 respectively.

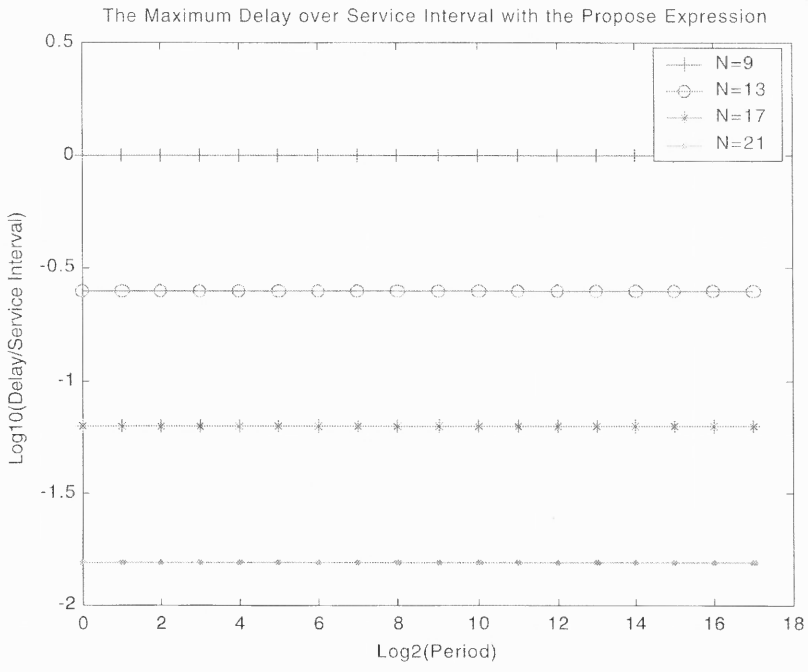

Figure 16 The normalized delay bound in the proposed expression for packet-based scheduler S

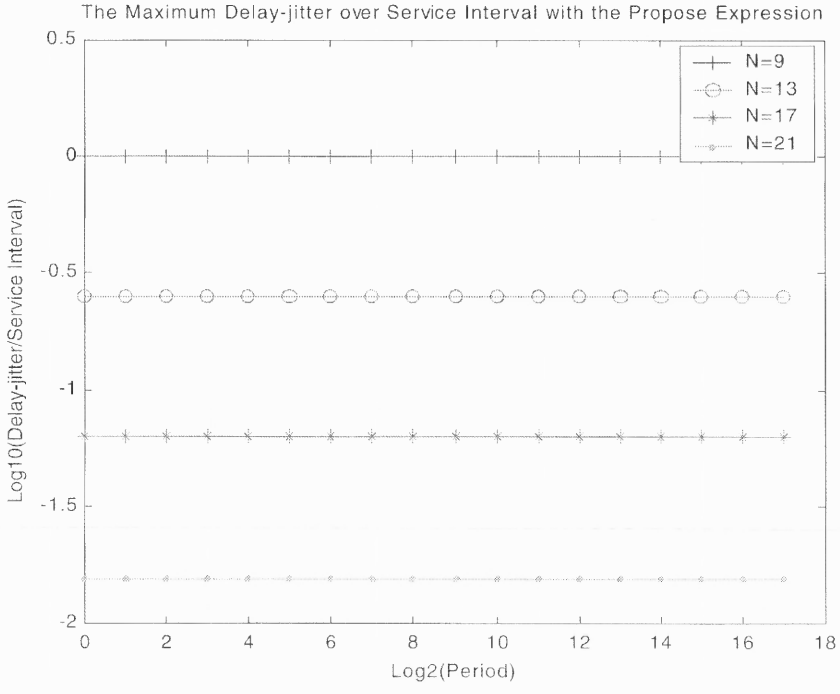

Figure 17 The normalized delay-jitter bound in the proposed expression for packet-based scheduler S'

For a cell-based scheduler, since each packet has the same size, then $\Phi_i^k = P_i \times \tau$. Equation (37)-(42) still hold, Equation (43) can be rewritten as

$$\Delta D_i^k = D_i^k = \underline{D_i^k} \times P_i \times \tau = 2^{-Z(P_i')+1} \times P_i \times \tau \qquad (44)$$

For the example in Chapter 3, session ***i*** requires service rate 100Mbps. Then P_i= 6.22 and $\tau = 0.5145\mu s$. Thus the maximum delay and delay-jitter can be calculated by Equation (44)

$$\Delta D_i^k = D_i^k = \underline{D_i^k} \times P_i \times \tau = 2^{-4} \times 6.22 \times 0.5145 = 0.2\mu s$$

Similarly, with Z(C) =5, (Z(T')=3, Z(P')=1); (Z(T')=5, Z(P')=3); (Z(T')=7, Z(P')=5); and (Z(T')=9, Z(P')=7) respectively (i.e. N=9; N=13; N=17 and N=21), The normalized delay and delay-jitter in the compressed expression are shown in Figure 18 and 19 respectively.

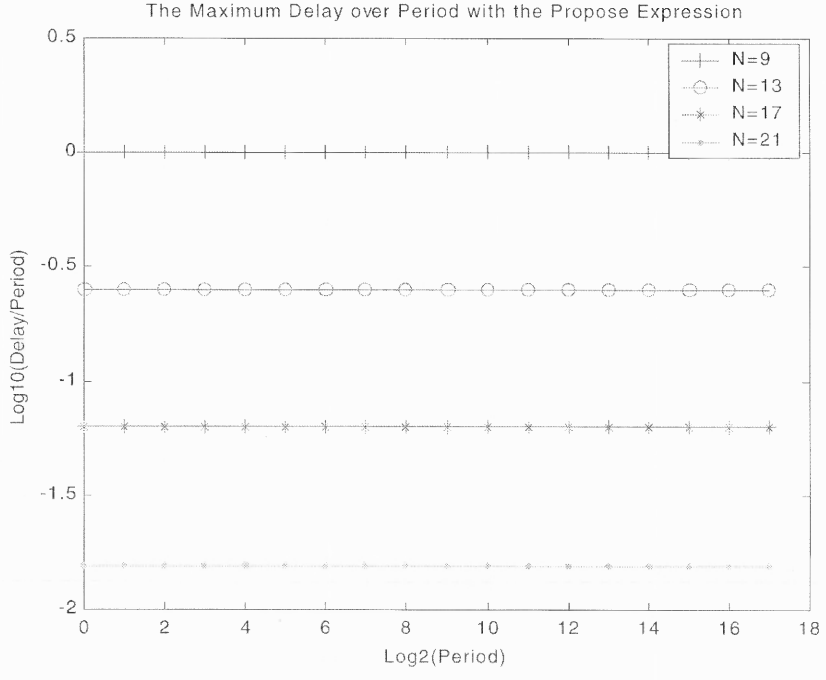

Figure 18 The normalized delay bound in the proposed expression for cell-based scheduler S'

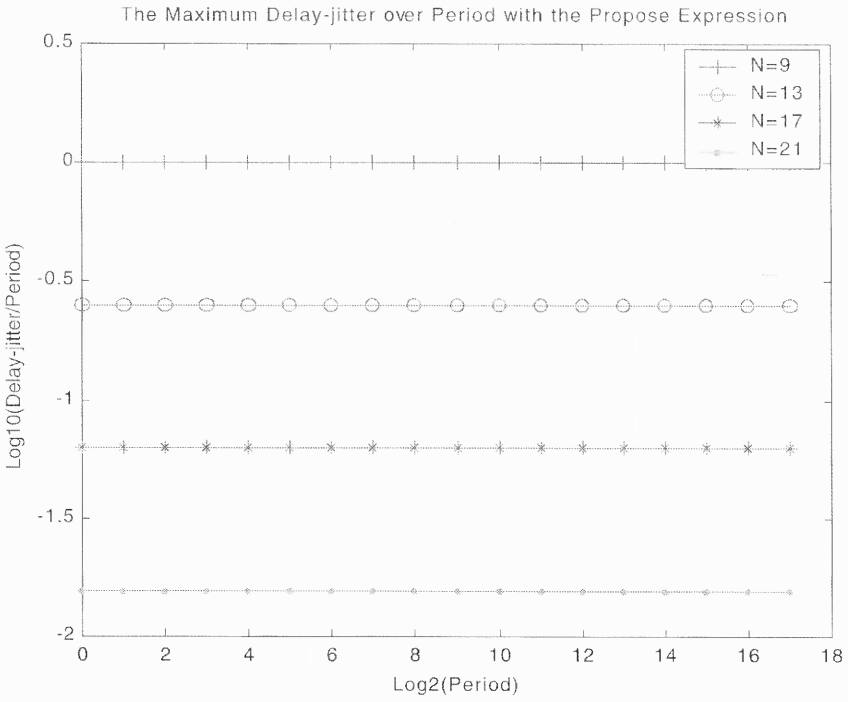

Figure 19 The normalized delay-jitter bound in the proposed expression for cell-based scheduler S'

CHAPTER 5

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 saves memory in terms of off-chip bandwidth and storage space.

In timestamp-based schedulers, there are always three operations: 1) addition; 2) modular comparison; and 3) multiplication and division. We introduce two more operation in the proposed method: 1) generating the compressed timestamp; and 2) reconstructing the timestamp. 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 two additional operations to adjust the decimal point by category number as compared to the fixed-point expression. This can be readily realized by hardware with some extra logic operations.

CHAPTER 6

CONCLUSION

In this thesis, 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 size 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. ε_{max} (r), L_{max}, L_{min}, r_{max} and r_{min}). Furthermore, if the size of representation of timestamp and period is the bottleneck of the granularity problem, the proposed expression can also improve the granularity of the scheduler.

APPENDIX

PROOF OF THEOREM 1

Proof:

The relative error of the service rate can be expressed in terms of the period representation with finite bits as follows:

$$\varepsilon(r_i) = \left|1 - \frac{r_i}{\overline{r_i}}\right| = \left|\frac{\frac{1}{r_i} - \frac{1}{\overline{r_i}}}{\frac{1}{\overline{r_i}}}\right| = \left|\frac{\frac{r_{\max}}{r_i} - \frac{r_{\max}}{\overline{r_i}}}{\frac{r_{\max}}{\overline{r_i}}}\right| = \left|\frac{P_i - \overline{P_i}}{P_i}\right|$$

where

$\overline{P_i}$ – ideal period of session ***i*** with infinite bit expression

P_i – actual period representation of session ***i*** with finite bit expression

$\overline{r_i}$ – ideal service rate of session ***i*** with infinite bit expression

r_i – service rate of session ***i*** with finite bit expression

r_{max} – maximum supportable service rate of the scheduling system

Define $\hat{\varepsilon}(P_i)$ as the approximation of relative error of Pi

$$\hat{\varepsilon}(P_i) = \left|\frac{P_i - \overline{P_i}}{P_i}\right|$$

Note that the relative error of the service rate is independent of the packet size. Therefore, the following equation always holds for both packet-based and cell-based scheduling system.

$$\hat{\varepsilon}(P_i) = \varepsilon(r_i) \qquad \text{(A.1)}$$

REFERENCES

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

[2] D.C. Stephens, J.C.R. Bennett and Hui Zhang, "Implementing 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^2Q: 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.