Considerations for Scheduling and Servicing Events Using a Calendar Structure

Wladek Olesinski, Rob Robotham, Mustapha Aissaoui, Jordan Lu
Alcatel Networks Corporation
600 March Road
Ottawa, Ontario, Canada K2K 2E6
e-mail: {wladyslaw.olesinski,rob.robotham,mustapha.aissaoui,jordan.lu}@alcatel.com

Abstract— **Work-conserving disciplines may be implemented with the use of scheduling calendars. Since calendars have finite resolutions, theoretical emission times that determine the positions of frames within the calendar slots have to be rounded to match the boundaries of the slots. The selection of a rounding method affects the performance of a work-conserving discipline. In this paper, we present an efficient calendar structure and examine several rounding methods.**

I. Introduction

Networks are supposed to support multiple service categories that include real-time services and best-effort services. Streams belonging to different service categories require different treatment at the switches. Real-time applications rely on timely delivery of frames and are sensitive to delay variations (jitter). Timing constraints imposed on applications using best-effort services are more relaxed. Such applications benefit from the available, excess bandwidth.

In both types of applications, it is important to differentiate between requirements of different streams, and allow for an appropriate distribution of resources among them. Work-conserving service disciplines, modeled on Generalized Processor Sharing (GPS) algorithm, have been designed for this purpose. With such disciplines, a server is never idle when there is a frame to send. Some notable examples of work-conserving service disciplines [8] are: delay earliest-due-date (delay-EDD) [4], virtual clock [9], weighted fair queueing (WFQ) [3] also called packetized generalized processor sharing (PGPS) [7], self-clocked fair queueing (SCFQ) [5], and worst-case fair weighted fair queueing (WF^2Q) [1].

In weighted work conserving-service disciplines, the amount of bandwidth allocated to a stream depends on its weight. A queue calendar [2] is a practical way of a discipline implementation. The proposed calendar structure used in this study is described in more detail in the next section.

With our approach, a work conserving-service discipline calculates, so called, *theoretical emission time* (*TET*). The *TET* is calculated for every frame that arrives to the system, and its value depends on the weight of the stream this frame belongs to (or, in short, a frame's weight). Depending on *TET*, the frame is placed in one of the slots into which a calendar is divided.

If the system processes only one stream, then obviously the frame will be processed immediately – the system is never idle when there is a frame to send. If the system shares its resources with other streams, on the other hand, the order in which the frames are processed depends on their relative weights. These weights determine the values of *TET*, and different values of *TET* may place frames in different slots.

A calendar resolution is finite. When a frame is scheduled, its *TET* value is likely to be between the boundaries of a slot, rather than exactly at one of its boundaries. In that case, the *TET* is rounded up (or down) to the nearest slot boundary.

In this paper, we present a calendar structure especially suited for scheduling, as well as several methods of rounding the *TET*. We study their impact on implementation complexity, and service differentiation between streams of different weights. We also look at how the proposed methods assure that the low-weight traffic does not interfere with high-weight traffic.

In the next section, we describe a structure of a calendar and its processing. In Section 3, we introduce algorithms of four rounding methods. Section 4 compares the methods, and summary in Section 5 concludes our findings and suggests ideas for further research.

II. Calendar Processing

An example structure of the calendar is shown in Figure 1. Slots 0 and 1 have identical sizes, while each consecutive slot is twice as big as the previous slot. Specifically, the 0^{th} slot is for $0 < TET \le 2^0$, the 1^{st} slot is for $2^0 < TET \le 2^1$, the 2^{nd} slot is for $2^1 < TET \le 2^2$, etc. Formally, the relation between slot number N_s and *TET* is as follows:

$$0 < TET \le 1 \text{ for } N_s = 0 \tag{1}$$

$$2^{N_s - 1} < TET \le 2^{N_s} \text{ for } N_s \ge 1$$

The latter inequality implies:

$$N_s = \lceil \log_2 TET \rceil \text{ for } TET > 1 \tag{2}$$

The calendar has a finite resolution. Consequently, *TET* of a frame being scheduled may place that frame anywhere between the slot boundaries. Depending on the rounding algorithm, *TET* is rounded down or up to the boundary of a slot.

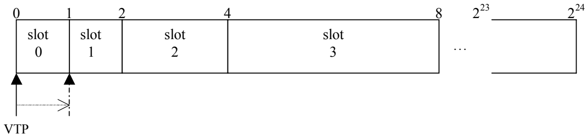

Figure 1. Structure of a calendar

For example, assume that for some frame, $TET = 1.2$, and that *TET* is always rounded up to the right boundary of a slot. Since *TET* is between 1 and 2, the boundaries of slot 1, it is rounded up by 0.8 and scheduled in slot 1.

A *virtual time pointer* (*VTP*) moves in increments of 1 unit, removing the scheduled frames from the current slot. In this context, "unit" corresponds to the size of the slot of the highest resolution. In other words, a "window" of width equal to the size of slot 0 moves on the calendar. Frames "visible" through this window are removed from the calendar.

Consecutive slots in a calendar have increasing sizes (see Figure 1). Although this design decreases resolution of the calendar, it also increases the range of *TET* values that can fit in the calendar. For example, assume there are two frames with *TET*s 5 and 7. The first frame has a greater weight because its *TET* is smaller. However, both frames are scheduled in the same slot 3. If the slots had equal sizes of 1, the second frame would be scheduled in the slot closer to the right boundary of a calendar, giving the first frame a higher priority.

The advantage of this design, however, is the increased range of *TET* values that can fit in the calendar, as well as easier implementation. The calendar from Figure 1 can accommodate frames whose $TET \le 2^{24}$ using just 25 slots. A calendar with equal-size slots would require 2^{24} slots to accomplish that.

Note also that a poor resolution of, e.g., 2^{22} is only experienced by frames whose values of *TET* are very large. High-weight frames, whose scheduling precision is particularly important, have small *TET* and are scheduled in high-resolution slots.

III. Rounding Methods

Initially, *TET* is determined by the value T_{inc} that we will call an *increment*. T_{inc} depends on the weight of a stream a given frame belongs to. With the increasing weight of a stream, T_{inc} decreases, allowing frames to be scheduled in slots of higher resolution (i.e., slots of smaller sizes). Such frames will be reached by *VTP* earlier and therefore they will be processed ahead of relatively less important frames with lower weights.

Depending on the rounding method, T_{inc} may be decreased or increased by a round-up or round-down value used in the calculation of *TET* for the previous frame (see below). This modified value of T_{inc} will be called *initial theoretical emission time*, and denoted by TET_{init}. Since the value of TET_{init} may place a frame anywhere between the boundaries of some slot, TET_{init} will be rounded down (or up, again depending method) to the nearest boundary of a slot. This rounded value of TET_{init} finally becomes a *theoretical emission time* and determines what slot a frame is scheduled in.

A. Up_down *method*

Let $R_p = R_c = 0$, and T_{inc} be an *increment* of a given frame

As long as there is a supply of frames, repeat the following process:

Calculate the initial theoretical emission time:

$$TET_{init} = T_{inc} - R_p$$

Find the slot number N_s for TET_{init} according to (1) and (2)

Calculate the round-up value $R_c = 2^{N_s} - TET_{init}$

Calculate theoretical emission time:

$$TET = TET_{init} + R_c = 2^{N_s}$$

The frame is scheduled in slot N_s

The TET_{init} of the *next* frame will be decreased by the round-up value $R_p = R_c$ used in scheduling of *this* frame

Initially, the increment of the frame being scheduled on the calendar is decreased by R_p, the value used to round up *TET* in the previous calculation (R_p is set to 0 for the first frame). Then, the algorithm finds slot number N_s for the calculated value TET_{init} using (1) and (2) from Section II. In the next step, the algorithm finds R_c that says how far the TET_{init} is from the right boundary of slot N_s. As signaled in Section II, the boundaries of slot N_s are 2^{N_s-1} and 2^{N_s}. Finally, the *TET* is calculated and the frame is scheduled in slot N_s.

Before the *TET* for the next frame is calculated, the round-up value R_c is assigned to R_p that will be used to round the initial value of *TET* in the iteration for the next frame.

Fairness is an advantage of this method. It can be shown that even if streams have similar weights, this method will differentiate between them in the long run, allowing frames with higher weights (and smaller increments) to be scheduled in slots of higher resolution.

To achieve fairness, the algorithm has to keep the round-up value used in one calculation and reuse it in the next. The *up_down* algorithm (as well as *down_up* presented below) rounds up or down the initial values of *TET* not only because of a finite resolution of the calendar. Every time TET_{init} for some frame is increased (rounded up), its weight is effectively decreased. That gives other frames with lower weights an unfair advantage over this frame. Since the round-up value is subtracted from TET_{init} for the next frame, thereby increasing its weight, the stream is able to regain the lost weight.

Thus, the presented algorithm allows preserving the weights of the streams represented in the scheduler by values of T_{inc}. Particular values of theoretical emission time may vary from the prescribed T_{inc} due to the limited resolution of the calendar but in the long term, the average *TET* equals T_{inc}.

There are two disadvantages of the *up_down* method: complexity and interference with frames scheduled in high-resolution slots. The previous round-up value is stored for use in the next calculation, and several addition and subtraction operations are performed for each frame. Interference is related to the fact that a frame with a high T_{inc} (i.e., low weight) may be scheduled in high-resolution slots, interfering with high-weight frames. Such interference is harmful as it may increase the jitter of delay-sensitive, high-weight frames.

B. Down_up *method*

Let $R_p = R_c = 0$, and T_{inc} be an *increment* of a given frame

As long as there is a supply of frames, repeat the following process:

Calculate the initial theoretical emission time:

$$TET_{init} = T_{inc} + R_p$$

Find the slot number N_s for TET_{init} according to (1) and (2)

Calculate the round-down value R_c:

$$R_c = TET_{init} - 2^{N_s - 1} \text{ for } N_s > 1 \text{, or}$$

$$R_c = TET_{init} \text{ for } N_s = 0$$

Calculate theoretical emission time: $TET = TET_{init} - R_c$

The frame is scheduled in slot $N_s - 1$ or 0, if $N_s - 1 < 0$

The TET_{init} of the *next* frame will be increased by the round-down value $R_p = R_c$ used in scheduling of *this* frame.

This method is somewhat similar to *up_down*. Now, however, the increment of the frame being scheduled on the calendar is *increased* by R_p, the value used to round down TET_{init} in the previous calculation. Then, as in *up_down* method, the algorithm finds slot number N_s for the calculated value TET_{init}. In the next step, however, the algorithm finds how far the TET_{init} is from the *left* boundary 2^{N_s-1} of slot N_s. Finally, the theoretical emission time is calculated by decreasing the TET_{init} by the round-down value of R_c, and the frame is scheduled in the slot $N_s - 1$, that is, the slot of immediately higher resolution than the slot in which the initial value theoretical emission time placed the frame.

As in *up_down* method, before the TET for the next frame is calculated, the value R_c is assigned to R_p later used to increase the initial TET_{init} in the iteration for the next frame.

Down_up method provides fairness to and differentiation between frames of different weights, similar to the *up_down* method. Complexities of these methods are also comparable. However, *down_up* method significantly decreases interference with high-weight frames. This is because frames are confined to slots N_s and $N_s - 1$ only (see Section IV). In the *up_down* method, frames may be scheduled even in slots of the highest resolution regardless of their weights and increment values.

C. Up_only *method*

This method is much simpler than *up_down* and *down_up* presented above. The increment T_{inc} of a scheduled frame is simply rounded up to the nearest slot boundary and scheduled in the relevant slot.

The obvious advantage of this method is its simplicity. There are fewer operations and the algorithm does not store any values from one frame to another. Also, frames with the given weight are always scheduled in the same slots which means that they do not interfere with high-weight frames scheduled in high-resolution slots.

The price to pay for these features is a lack of service differentiation between frames of slightly different weights. For example, two frames with increments 2.2 and 3.8 will be *always* scheduled in the same slot number 2. In other words, if increments of two frames are close enough, these frames will be scheduled in the same slot.

D. Down_only *method*

Now, the TET_{init} (which is T_{inc} in *up_only* and *down_only* methods) is rounded down to the nearest left boundary of a slot. This effectively schedules the frame in the slot of immediately higher resolution. Like in the *up_only* method, all frames with a given weight are scheduled in the same slot.

IV. Comparison of the Rounding Methods

Let us compare the four algorithms using the following criteria: complexity, differentiation in servicing frames of different weights, and interference with high-weight frames.

These three criteria have already been discussed in Section III. To better estimate interference with high-weight frames, let us introduce parameter *average slot number* N_{avg}.

In methods *up_down* and *down_up*, frames with the given weight are scheduled in slots of varying resolution. This allows differentiating between frames of slightly different weights, and providing higher-weight frames with more opportunities to be scheduled ahead of lower-weight frames. The average slot number N_{avg} will indicate which slot, on average, frames with the given increment T_{inc} are scheduled in.

Methods *up_only* and *down_only* are trivial – a frame with the given T_{inc} is always scheduled in the same slot. Thus, the average slot number N_{avg} is simply equal to the number of a slot in which the method schedules the frames.

Using the above three criteria, the best method would have a low complexity, it would differentiate between frames of slightly different weights, and finally, it would not allow low-weight frames to interfere with high-weight frames. Low interference will be represented by high values of average slot number N_{avg} as T_{inc} increases (and the weight decreases).

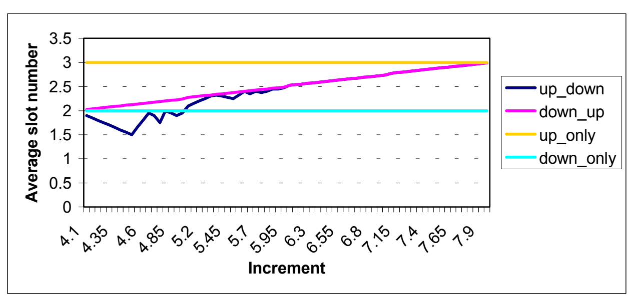

Figure 2. Average slot number N_{avg} vs increment T_{inc} for slot 3 ($4 < T_{inc} \leq 8$)

Figure 2. shows average slot number N_{avg} for increment values T_{inc} initially placing a frame in slot 3. N_{avg} is measured during a single *cycle* that spans the period from the initial moment at which $R_p = 0$, consists of subtracting/adding and rounding different intermediate values, and ends back at $R_p = 0$ after several iterations. First, let us analyze the simple *up_only* and *down_only* methods.

In the *up_only* method, the average slot number $N_{avg} = N_s$. This is because in this method, the *TET* is in fact the T_{inc} rounded up to the right boundary of the slot (see Section III.C). As Figure 2. indicates, *up_only* method provides the upper limit on the slot number in which a frame with a given T_{inc} may be scheduled.

In the *down_only* method, the average slot number $N_{avg} = N_s - 1$. This is because in this method, the *TET* is in fact the T_{inc} rounded down to the left boundary of a slot (see Section III.D). Since the left boundary of a slot belongs to the slot of immediately higher resolution, this effectively schedules the frame in that slot.

Figure 2. and the analysis suggest that *up_only* method is better than *down_only*. Both offer a low complexity and poor differentiation between frames of slightly different weights. However, frames whose T_{inc} values are rounded according to *up_only* method do not interfere with high-weight frames.

Behaviour of the *up_down* and *down_up* methods is more complicated and would require more space to explain. Here, let us observe important relations between these two methods.

If the increment T_{inc} initially places a frame in the 2nd half of slot N_s, the average values of slot number N_{avg} in these two methods are identical. Specifically, with the increasing T_{inc}, N_{avg} grows from $N_s - 0.5$ to N_s.

Behaviours of *up_down* and *down_up* methods, however, do differ when the increment T_{inc} initially places a frame in the 1st half of slot N_s. The average slot number N_{avg} of the *up_down* method drops down to $0.5 \times N_s$. At the same time, N_{avg} of the *down_up* method steadily grows, suggesting that frames are scheduled only in slots N_s and $N_s - 1$.

Clearly, *down_up* method is better than *up_down* method. Both have the same complexity and both well differentiate between slightly varying weights. However, for increments placing the frames in the 1st half of a slot, the average slot number of *up_down* method drops significantly, which means that frames are often scheduled in slots of high resolution, interfering with high-weight frames.

V. Summary, Conclusions and Future Work

In this paper, we presented and analyzed four rounding methods that determined how, in the calendar implementation of work-conserving service disciplines, frames are scheduled.

TABLE I. FEATURES OF THE ROUNDING METHODS

Method	Complexity	Differentiation	Interference
up_down	High	good	very high
down_up	High	good	low
up_only	Low	poor	none
down_only	Low	poor	medium

Table I summarizes the properties of all four methods, focusing on their complexity, differentiation between frames of different weights, and interference with high-weight frames scheduled in high-resolution slots.

Complex methods *up_down* and *down_up* provide a good service differentiation between frames of different streams. This means that in the long term, theoretical emission times at which the frames are removed from the calendar are equal to increments T_{inc} that correspond to their weights. The *down_up* method, however, is more attractive. Unlike the other method, it introduces little interference with high-weight frames, placing the scheduled frames in slots N_s and $N_s - 1$ only. The *up_down* method, in contrast, can place a frame in the slots of highest resolution even if the weight of the frame is very low.

Among the less complex methods *up_only* and *down_only*, the former is better. It does not interfere with high-weight frames at all, while the *down_only* method tends to penalize particularly frames of the the highest weights.

With the additional complexity, the *down_up* method is preferred. It provides little interference with high-weight frames and, unlike the two simple methods, it differentiates between frames of different weights. Otherwise, *up_only* method is a good compromise that despite a poor differentiation, introduces no interference with high-weight frames.

In the future, we plan to study other calendar structures and algorithms in the context of IP traffic (i.e., variable packet sizes and values of T_{inc}). We also plan to simulate our methods choosing a certain work-conserving service discipline and typical traffic patterns to see how the characteristics presented above affect real-life scenarios.

References

[1] Bennett, J.C.R. and Zhang, H., "WF^{2}Q: Worst-case fair weighted fair queueing", *Proc. IEEE INFOCOM'96*, pp. 120-128, San Francisco, CA, Mar. 1996.

[2] Brown, R., "Calendar queues: A fast *o(1)* priority queue implementation for the simulation event set problem", *Comm. of the ACM*, 31(10):1220-1227, Oct. 1988.

[3] Demers, A., Keshav, S. and Shenker, S., "Analysis and simulation of a fair queueing algorithm", *J. Internetworking Res. And Experience*, pp. 3-26, Oct. 1990.

[4] Ferrari, D. and Verma, D., "A scheme for real-time channel establishment in wide-area networks", *IEEE J. Selected Areas in Commun.,* vol. 8, pp. 368-379, Apr. 1990.

[5] Golestani, S., "A self-clocked fair queuing scheme for broadband applications", *Proc. IEEE INCOFOM '94*, Toronto, ON, June 1994, pp. 636-646.

[6] Parekh, A. and Gallager, R., "A generalized processor sharing approach to flow control – the single node case", *Proc. INFOCOM '92,* 1992.

[7] Zhang, H., "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", *Proc. IEEE*, vol. 83, no. 10, Oct. 1995.