A New Smoothed Fair Scheduling Algorithm based on Timeslot Reservation

Ji Li, Huaxin Zeng
School of Information Science and Technology
Southwest Jiaotong University
Chengdu, China, 610031
{jelecn, huaxinzeng1}@yahoo.com.cn

Dengyuan Xu
Computer and Information Technology College
Chongqing JiaoTong University
Chongqing, China, 400075
dengyuanxu1@yahoo.com.cn

ABSTRACT

Based on the background of SUPANET, this paper present a novel fair scheduling algorithm, which we call Smoothed Fair Scheduling based on Timeslot Reservation (TRSFS). TRSFS decomposes the data scheduling process into two stages: 1) data-queues generate and sent out schedule-requests at fixed rate according to the reserved timeslots and 2) The arbiter serves schedule-requests in the FIFO manner. By exactly emulating the idealized fair scheduling, TRSFS realized the design purpose of distributing the output traffic evenly. We also prove theoretically that TRSFS is a Guarantee Rate scheduling algorithm with good scheduling constant. Through parallel and distributed technology, TRSFS can be implemented easily in high-speed networks to provide quality of service due to its simplicity and the feature of asynchronous operation.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design –*Packet-switching networks*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems –*Sequencing and scheduling*

General Terms

Design, Algorithms, Performance

Keywords

fair scheduling, quality of service, timeslot reservation, SUPANET (Single physical-layer User-data-transfer Platform Architecture Network).

1. INTRODUCTION

Quality of Service (QoS) provisioning in Internet has drawn much attention from academic community as well as from industry. Moreover, with fast progress in communications, especially in DWDM (Dense Wavelength Division Multiplexing), bit rate in a lambda has reached 40/80/160 Gbps and this further demands Internet nodes with QoS-aware and high-speed switching capability catered for multimedia traffic. The Sichuan Network and Communication key lab has been working on a simplified network architecture called SUPANET (Single physical-layer User-data-transfer Platform Architecture Network) [6] [7] supported by a novel technique called EPFTS (Ethernet-oriented Physical Frame Timeslot Switching) [8] [9]. EPFTS uses a fixed-length frame format which we call Ethernet-oriented Physical Frame (EPF) to encapsulate max length of Ethernet MAC frames and utilizes the transmission time of an EPF as the timeslot for switching. EPFTS provides a permanent or switched Virtual Line (VL) service in delivering user data. To meet QoS requirements of different traffic, SUPANET need new switching fabric and better scheduling scheme suitable for high-speed switching.

Tactically, Existing fair scheduling algorithms are divided into two major categories: time-stamp based and round-robin based. Weighted Fair Queuing (WFQ) [4] (PGPS [11]) is an early time-stamp based scheduler, which emulates the ideal Generalized Processor Sharing (GPS) [11] by maintaining a virtual time clock. It schedules packets in the order of time -stamp. WFQ keeps a low local-delay bound and maintains good fairness, but it is difficult to be implemented for its O(N) time complexity. Even variants of WFQ, such as Virtual Clock [13], WF^{2}Q [1], WF^{2}Q+ [2], still have $O(logN)$ time complexity.

WRR [10] and DRR [12] are two typical round-robin schedulers. Though they are simple to implement with an $O(1)$ time complexity; they have poor upper delay-bounds, usually proportional to the number of queues (i.e. N) . Moreover, these schedulers prefer to transfer data in burst. A burst represents an instance of high channel utility; it may also cause buffer-overflow, or longer delay and jitters in downstream nodes. Reference [10] suggests easing the problem by controlling cell/packet interval to make an even output. Smoothed Round Robin (SRR) [3] further extends this idea to variable length switching and improves short-term fairness among queues by use of Weighted Matrix and Weight Spread Sequence (WSS). The trouble with SRR is that system performance may be greatly degraded if number of simultaneously active flows is too large or abrupt change in flow number occurs.

In this paper, we aim at designing a new fair scheduling scheme to realize service guarantee in SUPANET. We will present a novel fair scheduling algorithm called TRSFS (Smoothed Fair Scheduling based on Timeslot Reservation). In order to provide bandwidth and delay guarantee in SUPANET, TRSFS takes the advantage of timeslot reservation mechanism embedded in EPFTS to distribute output traffic evenly. TRSFS is designed to be easily implemented in high-speed environment for improving practical value.

INFOSCALE 2007, June 6-8, Suzhou, China
Copyright © 2007 ICST 978-1-59593-757-5
DOI 10.4108/infoscale.2007.913

2. Smoothed Fair Scheduling based on Timeslot Reservation

In order to explain the main idea behind the TRSFS, we first present a scheduling model. As shown in Figure1- (a), the model is formed of a shared output link with capacity C timeslots/s, N FIFO data-queues, where q_i (i=1… N) is used to queue the EPFs of i^{th} flow or i^{th} class of flows, and an arbiter. We take the total timeslots r_i reserved by the flows queuing at q_i as the reservation of data-queue q_i. The arbiter is responsible for deciding which data-queue is allowed to forward its head-EPF to the output link in a given timeslot. To avoid overbooking, the total reserved timeslots R for an output link is restricted by admission control to ensure $R = \sum_{i=1}^{N} r_i \le C$.

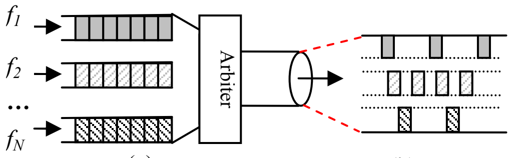


Figure 1. (a) Queuing model of shared output link (b) the idealized fairness scheduling model

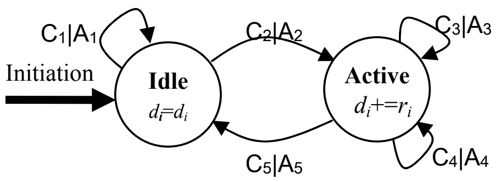
We consider an idealized fair scheduling model as follows (referring to Figure1- (b)): Each data-queue is assigned with a separate and independent channel from the output link, whose transmission capacity is exactly equals to the reservation of the corresponding data-queue. That is, the transmission rate of the channel corresponding to q_i is equals to r_i (timeslots/s) and the interval between adjacent EPFs forwarded from q_i is exactly $1/r_i$ seconds. In case that the total reserved timeslots are much less than C, to make fully use of the available transmission capacity, the free transmission capacity is reallocated to each data-queue proportional to its reservation.

The core idea behind TRSFS is to forward packets by emulating the referenced ideal model. Different from the ideal model, TRSFS forward packets in slot way since only one EPF can be sent out by the output link within a timeslot. In TRSFS, we define the time when a data-queue start to transmit an EPF in the emulated ideal model as the EPF's Virtual Start Time (VST). TRSFS divide EPF transmission process into two stages:

1. VST tracking and schedule-request generation: TRSFS track VSTs of all EPFs by a special mechanism described below. Once a "VST" is captured, a schedule-request from the data-queue is sent to arbiter.
2. Schedule-request management and arbitration: to resolve the schedule-request collision from multiple data-queues, the arbiter arranges schedule-requests into an FIFO request-queue according to arrival order and serves the request-queue when it is not empty.

In order to capture the VST exactly, two queue-states (active and idle) are defined and two counters: d_i and v_i are set for each data-queue in TRSFS. The value of d_i and v_i is updated once for each timeslot based on following principles:

- $d_i = d_i + r_i$ when q_i is active otherwise let d_i unchanged; if $d_i \ge R$, q_i send a schedule-request to the arbiter and $d_i = d_i - R$.
- $v_i = v_i + 1$when a new EPF arrived at q_i.
- $v_i = v_i - 1$ when q_i send out a schedule-request.



C_1: v_i=0.	C_2: v_i>0.
C_3: v_i>0 and $d_i \ge R$.	C_4: $d_i < R$.
C_5: v_i=0 and $d_i \ge R$.	A_1, A_4: null operation.

A_2: activate q_i and q_i sendout a schedule-request, $v_i = v_i - 1$.
A_3: $d_i = d_i - R$ and q_i sendout a schedule-request, $v_i = v_i - 1$.
A_5: $d_i = d_i - R$ and idle q_i.

Figure 2. The state transition sketch of queue q_i

In the initiation stage, all data-queues are on idle state. With the timeslot go on, the state of data-queues is changed according to Figure 2. The format string "$C_k|A_k$" aside every transition line in Figure 2 denotes the transition condition and action between the two states. All notations are described in the below of Figure 2. In TRSFS the state change of a data-queue belongs to one of five situations in any timeslot, that is, when C_k is satisfied, A_k is executed, k=1…5. And a data-queue is set to be active only when it is busy in the emulated ideal model. Thus it is easy to know that q_i will send a schedule-request out every R/r_i timeslots when active. No less that r_i EPFs will be forwarded from q_i in a round of C timeslots because $R \le C$.

The main procedure of state tracking in TRSFS can be formally described as follows:

```
Initiation at t=0: $d_i$=0; $v_i$=0; $R = \sum_{i=1}^{N} r_i$ ;
State Updating at t (≥1) for all data-queues:
if ($q_i$ is active ) {
    $d_i = d_i + r_i$ ;
    if ( $d_i \ge R$ ) {
        $d_i = d_i - R$ ;
        if($v_i$>0) { send scheduling request; $v_i$--; }
        else  idle $q_i$;  }  }
when a new EPF (P) arrived at $q_i$ {
    $v_i$++;
    if ( $q_i$ is idle)
        {sendout a schedule-request ; $v_i$--; active $q_i$; }  }
```

3. PROPERTIES OF TRSFS

In this Section, We firstly theoretically prove the conclusion that the occupation length of the request-queue is not greater than N, which directly indicate the long-term fairness of TRSFS. Then we further analyze the short-term fairness and delay performance of TRSFS based on this conclusion. Because of the page size limitation, we omit all proving procedure for brief expression in this section. More details can be got by contacting the authors.

3.1 Occupation of Request-queue

TRSFS allows a data-queue to forward an EPF only when the arbiter serves a schedule-request sent by the data queue. And the stay time of a schedule-request in request-queue depends on the occupation length of request-queue when the request is generated. Thus the occupation length of request-queue determines the guarantee performance of TRSFS. We first explain some notations to be used before further analysis.

- u_i: the fraction of the bandwidth shared by q_i, i.e., $u_i=r_i/R$.
- a_i: the normalized value of d_i by R, i.e., $a_i=d_i/R$.
- $|A|$: the number of element in set A.
- Ω : the set of the N data-queues.
- $A(t)$: the set of idle data-queues at the end of timeslot t.
- $B(t)$: the set of active data-queues at the end of timeslot t.
- $l(t)$: the length of request-queue at the end of timeslot t.
- $H(t)$: the set of data-queues ***idled*** in timeslot t.
- $M(t)$: the set of data-queues ***activated*** in timeslot t.
- $n(t)$: the number of schedule-requests generated in timeslot t because of $d_i \geq$R.
- $\sum(t) \equiv \sum_{q_i \in B(t)} a_i$; $h(t) \equiv |H(t)|$; $h(t) \equiv |H(t)|$.

Apparently in TRSFS we always have $\sum_{i=1}^{N} u_i = 1$ and $A(t)+B(t)=\Omega$, $A(t)\cap B(t)=\phi$. Moreover "d_i = d_i + r_i" and "$d_i \geq R$" are equivalent to "$a_i=a_i+u_i$" and "$a_i \geq 1$" respectively. Without loss of generality, we always assume EPFs arrive only at the timeslot boundary and leave immediately at the end of the timeslot when scheduled.

Lemma 1. For $t \geq 1$, we have $|A(t)| = \sum_{i=1}^{t}[h(i)-m(i)]$.

Lemma 2. For $t \geq 1$, we have $n(t)+h(t)+\sum(t) = \sum(t-1)+\delta(t)$, where $\delta(t)=\sum_{q_k \in B(t)} u_k$ and $\delta(t) \leq 1$.

We define a *busy period* of the arbiter as the timeslot duration that the request-queue is not empty continuously. For example, [t_s, t_e] is called a busy period of the arbiter iif $l(t_s-1)=0$, $l(t_e)=0$ and $l(t)>0$, $t_s \leq t < t_e$. Since the maximum value of $l(t)$ only appears in busy period of the arbiter, we can get following lemma.

Lemma 3. For any busy period [t_s, t_e] of the arbiter, we have $l(t_s) < |B(t_s)| + 1 - \sum(t_s)$.

Theorem 1. for any busy period [t_s, t_e] of the arbiter, we have $l(t) \leq |B(t)|$, $t_s \leq t \leq t_e$.

Because $|B(t)| \leq N$, it is easy to know from theorem 1 that the occupation length of request-queue is not greater than N. therefore, the long-term fairness of TRSFS is proved.

3.2 Local Delay Bound

The authors of [5] defined a class of GR scheduling algorithms based on the notion of Guaranteed Rate Clock (GRC). Many scheduling algorithms shown in [5] belong to the GR class. For example, both PGPS [11] and virtual clock (VC) [13] are GR scheduling algorithms with scheduling constant $\beta = L/C = 1$ timeslot, where L is the maximum length of packets. It can be theoretically proved that TRSFS is a GR scheduling algorithm too.

Theorem 2. TRSFS is a GR scheduling algorithm with schedule constant $\beta = N - R/r$ timeslots (where r is the reservation of the considered data-queue).

As a GR scheduling algorithm, TRSFS guarantee local delay bound to traffic that confirm to a certain traffic model. For example, if EPFs arrive at data-queue q with reservation r confirms to the token bucket model (σ, r), the tight local delay bound of the EPFs is $\frac{\sigma \times C}{\frac{R}{r}} + N - \frac{R}{r}$ timeslots which is better than that of PGPS when $N - \frac{R}{r} \leq 0$.

3.3 Short-term fairness

The generation process of schedule-requests indicates the short-term fairness of TRSFS. Assume that [t_0, t_e] is the timeslot duration when the two data-queues q_i, q_j are both active. Let $S_i(t_0,t)$ denotes the number of schedule-requests generated by q_i during [t_0,t]. We have following theorem with TRSFS.

Theorem 3. For any $t \in [t_0, t_e]$, we have

$$\left| \frac{S_i(t_0,t)}{r_i} - \frac{S_j(t_0,t)}{r_j} \right| \leq \max\{\frac{1}{r_i}, \frac{1}{r_j}\} \text{, where } \tau = t - t_0 + 1 .$$

Theorem 3 shows TRSFS have good short-term fairness, because the generation process of schedule-requests determines the output sequence of EPFs. The schedule-request is generated with good short-term fairness, which means the output traffic from different data-queues is distributed very evenly.

3.4 Complexity

The scheduling scheme based on TRSFS mainly consists of two functionalities as described in Section II. Firstly, the state of all data-queues need to be updated every timeslot, which is easy to be implemented when N is small. Because the state update operation at data-queues is independent to each other, we believe the implementation problem when N is relative large could be easily resolved by distributed and parallel processing technology. Secondly, the request-queue in the arbiter should support up to N+ 1 access per timeslot in worst case. With assumption that a request needs a length of *logN* bits to save information, the implementation of request-queue requires a *NlogN* bits memory space which access bandwidth is no less than (N+1)*logN* bits per timeslot. For example when 1 timeslot equal to 100 ns and N = 256, TRSFS requires a 256 Bytes memory space with access bandwidth of 20.48 Gbps, which is easily implemented under memory technology nowadays.

4. Simulation Results

We established a simulation model of a shared output link by software to compare the local delay and short-term fairness properties of TRSFS with that of two existing algorithms, DRR and SRR.

We set C=700 and the buffer size for each queue is unlimited. There are totally N=12 data-queues q_i (i=1,…,12) among which q_i (i=2,…,12) are used to buffer the guarantee performance flow f_i (i=2,3, … ,12) while a best effort flow f_1 queuing at q_1. All guarantee performance flows are generated conforming to the token bucket model (σ, r) where σ, r is the max burst length and reserved timeslots of the flow respectively. In the following experiments, we let f_1 has an arrival process with average rate of 150, but is only assigned reserved timeslots of 90. All experiments only test the guarantee performance of f_i (i=2,3, … ,12) with reserved timeslots from 5 to 105 in an increment step of 10.

In the first experiment, we set σ=1 for f_i (i=2,3,…,12) and let f_1 arrive according the Poisson arrival process. Actually f_i (i=2,3,…,12) are CBR flows when σ=1. The comparison of the average (solid line with suffix "-A") and maximum local delay (dashed line with suffix "-M") is shown in Figure 3. It can be seen that TRSFS outperform both DRR and SRR for CBR traffic. SRR favors large bandwidth traffic. DRR is not sensitive to the reserved bandwidth of traffic but have large delay. Only TRSFS show very low delay performance as well as good fairness.

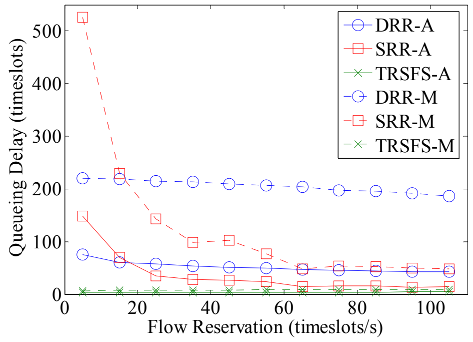


Figure 3. Comparison of delay performance

In the second experiment, the arrival process of f_1 confirms to the token bucket model (σ, 150). The simulation result of the worst local delay for flow f_i (i=2,…,12) with σ=8,16,32 is shown in Figure 4 where dashed lines depict the local delay bounds theoretically derived from theorem 2. The simulation result shows TRSFS can guarantee delay bound to flows constrained by token bucket model.

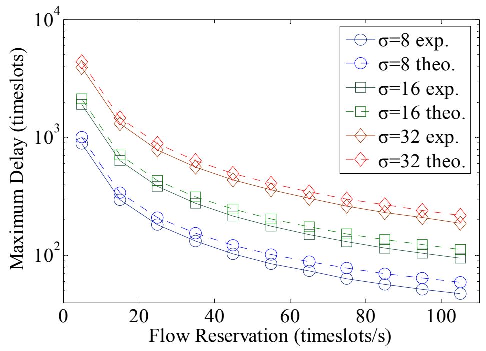


Figure 4. Worst delay of constrained traffic in TRSFS

We refer to the continuous EPF sequence sent out from the same data-queue as a *burst*. The third experiment is to compare the burstiness of output traffic, namely short-term fairness, of different schedulers. The EPF arrival process of flows is same as the second experiment. The statistic result of burst length is shown in Table II. Apparently traffic burstiness at output is far higher than at input for DRR, whereas the situation is on the contrary for SRR and TRSFS. TRSFS indeed appear a litter better burstiness than SRR, which indicate the good short-term fairness of TRSFS.

Table I. Comparison of output bursitiness

	σ	DRR	SRR	TRSFS
Mean burst size	σ =8	10.54363	1.000032	1.000009
	σ =16	15.05648	1.000052	1.000015
	σ =32	23.84582	1.000064	1.000017
Max burst size	σ =8	90	2	2
	σ =16	90	2	2
	σ =32	101	2	2

5. CONCLUSION

We have presented a new fair scheduling algorithm-TRSFS under the background of SUPANET. The features of SUPANET include using fixed-length EPF as the basic switching unit and taking timeslot reservation as the premise of QoS guarantee. TRSFS emulate the idealized scheduling model exactly base on the reservation on all data-queues. Thus the output of EPFs in TRSFS is distributed more evenly than that of the existing round robin schedulers. We Also proved TRSFS is a GR scheduling algorithms with smaller scheduling constant than PGPS under specific conditions. And because of the property of asynchronous operation, TRSFS is easy to be implemented in distributed and parallel method. Hence TRSFS is proper to apply in high-speed link especially when the number of data-queues N is small.

6. References

[1] Bennett, J.C.R. and Zhang, H. WF2Q: Worst-case Fair Weighted Fair Queueing. In: *Proc. of INFOCOM'96*, San Francisco: IEEE press, 1996. 120-128.

[2] Bennett, J. C. R, Zhang, H. Hierarchical Packet Fair Queueing Algorithms. *IEEE/ACM Transactions on networking*, Vol. 5., No. 5, October 1997.

[3] Chuanxiong Guo. SRR: an O(1) time-complexity packet scheduler for flows in multiservice packet networks. *IEEE/ACM Trans. on Networking*, 2004, 12(6):1144-1155.

[4] Demers, A., Keshav, S. and Shenker S. Analysis and simulation of a fair queueing algorithm. *ACM Computer Communication Review*, 1989, 19(4):1-12.

[5] Goyal, P. and Vin, H. M. Generalized guaranteed rate scheduling algortihms: a framework. *IEEE/ACM Trans. on Networking*, Vol. 5(4), pp. 561-571. Apr. 1997

[6] Huaxin Zeng, Jun Dou, Dengyuan Xu. Replace MPLS with EPFTS to build a SUPANET. In: *Proc. of HPSR 2005*. Hong Kong: IEEE press, 2005.39-43.

[7] Huaxin Zeng, Jun Dou, Dengyuan Xu. Single physical layer U-plane Architecture (SUPA) for Next Generation Internet. In: *Comprehensive Report on VoIP and enhanced IP Communications Services*, IEC, 2004.197-227.

[8] Huaxin Zeng, Dengyuan Xu, Jun Dou. Promotion of Physical Frame Timeslot Switching (PFTS) over DWDM. In: *Annual Review of Communications*, vol. 57, IEC, 2004.809-826.

[9] Huaxin Zeng, Dengyuan Xu, Jun Dou. On Physical Frame Time-slot Switching over DWDM. In: *Proc. of PACAT 2003.* Chengdu: IEEE press, 2003.535-540.

[10] Katavenis M, Sidiropoulos S, Courcoubetis C. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communication*, 1991, 9(8):1265~1279.

[11] Parekh, A.K. and Gallager, R. G. A generalized processor sharing approach to flow control in integrated services

networks: the single-node case. *IEEE/ACM Trans. On Networking*, 1993, 1(3): 344-357.

[12] Shreedha M, Varghese G. Efficient fair queuing using deficit round-robin. *IEEE/ACM Trans. on Networking*, 1996, 4(3):375-385.

[13] Xie, G.G. and Lam, S.S. Delay guarantee of virtual clock server, *IEEE/ACM Trans. on Networking (TON)*, v.3 n.6, p.683-689, Dec. 1995