Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

Baker

Finzi
Frances
Lochin
Mifdaoui
ISAE-SUPAERO
March 5, 2018

Priority Switching Scheduler
draft-finzi-priority-switching-scheduler-01

Abstract

We detail the implementation of a network scheduler that switches the priority of one or several queues. This scheduler aims at carrying and isolating time constrained and elastic traffic flows from best-effort traffic. We claim that the usual implementations with rate schedulers (such as WRR, DRR,...) do not allow to efficiently quantify the reserved capacity of the different classes. By using this credit based scheduler mechanism called Priority Switching Scheduler, we provide a more predictable available capacity.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Context and Motivation

To share the capacity offered by a link, many fair schedulers have been developed, such as Weighted Fair Queueing, Deficit Round Robin. However, with these solutions, the capacity available to a class is strongly impacted by the other classes. With this new scheduler denoted Priority Switching Scheduler (PSS), we wish to reduce this impact on some classes and provide them with a more predictable output rate, reporting the impact on other classes. Additionally, compared to well-known schemes, this solutions is simpler to implement as it does not require a virtual clock, and more flexible thanks to the many possibilities offered by the setting of different priorities.

1.2. Priority Switching Scheduler in a nutshell

The principle of PSS is the use of credit counters to change the priority of one or several queues. For each switching queue q, its priority, denoted p[q], switches between two values, depending on its associated credit counter. Then classic Priority Scheduler is used for the dequeuing process.

The main idea is that changing the priorities adds fairness to the Priority Scheduler. Depending on its credit counter parameters, the amount of capacity available to a queue is bounded between a minimum and a maximum value. Consequently, good parameterization is very important to prevent starvation of lower priority queues.

The new service we seek to obtain for the queue with the switching priority is more predictable: the minimum between a desired capacity and the residual capacity left by higher priorities. The impact of the input variations of higher classes is passed down to lower priority classes.

Finally, this new solutions offers much flexibility as we can have both i) queues with a reserved capacity (when two priorities are set), ii) and queues scheduled with a simple Priority Scheduler (when only one priority is set).

2. Priority Switching Scheduler

2.1. Specification

The PSS defines for each queue q a low priority, p_low[q], and a high priority, p_high[q]. For each queue q with p_high[q] > p_low[q], to manage the priority switching a credit counter is defined with:

- o a minimum level: 0;
- o a maximum level: LM[q];
- o a resume level: LR[q]
- o a reserved capacity: BW[q]
- o an idle slope: Iidle[q]=C*BW[q];
- o a sending slope: Isend[q]=C-Iidle[q];

The priority change depends on the credit counter as follows:

- o initially, the credit counter starts at 0;
- o the change of priority p[q] of queue q occurs in two cases:
  - * if p[q]=p_high[q] and the credit reaches LM[q];
  - * if p[q]=p_low[q] and credit reaches LR[q];

- o when a packet of queue q is transmitted, the credit increases with a rate Isend[q], else the credit decreases with a rate Iidle[q];

- o when the credit reaches LM[q], it remains at this level until the end of the transmission of the current packet;

- o when the credit reaches 0, it remains at this level until the start of the transmission of a queue q packet.

Figure 1 and Figure 2 show two examples of credit and priority behaviors of a queue q.

```
       ^ credit[q]
       |         |                   |
       |p_high[q]|       p_low[q]    | p_high[q]
   LM  +---------|-------------------|------------
       |      _ '|    |`             |        '
       |Isend| ' |    |  `           |       '
       | [q]  '  |    |    `  _      |      '
       |     '   |    |      ` |Iidle|     '
       |    '    |    |        `  [q]|    '
       |   '     |    |          `   |   '
   LR  +  '      |    |            ` |  '
    0  +.........|....|..............|..........>
                      |              |          time
       @@@@@@@@@@@@@@@@oooooooooooooo@@@@@@@@@@

                   @ queue q traffic  o other traffic
```

Figure 1: First example of queue q credit and priority behaviors

```
       ^ credit[q]
       |                        |
       |         p_high[q]      |     p_low[q]
   LM  + -----------------------|--------------------     '
       |                       '|  |`          '
       |               '|`   '  |  |  `       '
       | [q]          ' |  `    |  |    `   '
       |     '`      '  |   |   |  |      `
       |    '|  `   '   |   |   |  |      |
       |   ' |     |    |   |   |  |      |
   LR  +--'--|-----|----|---|---|--|------|--------
    0  +-'---|-----|----|---|---|--|------|-------->
             |     |    |   |      |      |        time
       @@@@@@oooooo@@@@@oooo@@@@@@@@oooooo@@@@@@@

                   @ queue q traffic  o other traffic
```

Figure 2: Second example of queue q credit and priority behaviors

Finally, for the dequeuing process, a Priority Scheduler selects the appropriate frame using the current p[q] values.

2.2. Implementation

The new dequeuing algorithm is presented in the PSS Algorithm. The credit of each queue q, denoted credit[q], and the dequeuing timer denoted timerDQ[q] are initialized to zero. The initial priority is set to the high value p_high[q]. First, for each queue with p_high[q] > p_low[q], the difference between the current time and the time stored in timerDQ[q], is computed (lines 2 and 3). The duration dtime[q] represents the time elapsed since the last credit update, during which no shaped packet was sent, we call this the idle time. Then, if dtime[q]>0, the credit is updated by removing the credit gained during the idle time that just occurred (lines 4 and 5). Next, timerDQ[q] is set to the current time to keep track of the time the credit is last updated (line 6). If the credit reaches LR[q], the priority changes to its high value (lines 7 and 8). Then, with the updated priorities, the priority scheduler performs as usual: each queue is checked for dequeuing (lines 12 and 13). When a queue q is selected with p_high[q] > p_low[q], the credit expected to be consumed is added to credit[q] variable (line 16). The time taken for the packet to be dequeued is added to the variable timerDQ[q] (lines 13 and 14) so the transmission time of the packet will not be taken into account in the idle time dtime[q] (line 2). If the credit reaches LM[q], the priority changes to its low value (lines 18 and 19). Finally, the packet is dequeued (line 22).

```
Inputs: credits, timerDQs, C, LMs,LRs,BWs,p_highs, p_lows
currentTime=getCurrentTime()
for each queue q with p_high[q] > p_low[q] do:
   dtime[q]=currentTime-timerDQ[q]
   if dtime[q]>0 then:
      credit[q]=max(credit[q]-dtime[q].C.BW[q],0)
      dtime[q]=currentTime
      if credit[q]<LR[q] and p[q]=p_low[q] then:
         p[q]=p_high[q]
      end if
   end if
end for
for each priority level pl, highest first do:
   if length(queue(pl))>0 then:
      q=queue(pl)
      if p_high[q] > p_high[q] then:
         credit[q]=min(LM[q], credit[q]+size(head(q)).(1-BW[q]))
         timerDQ[q]=currentTime+size(head(q))/C
         if credit >=LM[q] and p[q]=p_high[q] then:
            p[q]=p_low[q]
         end if
      end if
      dequeue(head(q))
   end if
end for
```

Figure 3: PSS algorithm

PSS algorithm also implements the following functions:

- getCurrentTime() uses a timer to return the current time;
- queue(pl) returns the queue associated to priority pl;
- head(q) returns the first packet of queue q;
- size(f) returns the size of packet f;
- dequeue(f) activates the dequeing event of packet f.

3. Usecase: benefit of using PSS in a Diffserv core network

3.1. Motivation

The DiffServ architecture defined in [RFC4594] and [RFC2475] proposes a scalable mean to deliver IP quality of service (QoS) based on handling traffic aggregates. This architecture follows the

philosophy that complexity should be delegated to the network edges while simple functionalities should be located in the core network. Thus, core devices only perform differentiated aggregate treatments based on the marking set by edge devices.

Keeping aside policing mechanisms that might enable edge devices in this architecture, a DiffServ stateless core network is often used to differentiate time-constrained UDP traffic (e.g. VoIP or VoD) and TCP bulk data transfer from all the remaining best-effort (BE) traffic called default traffic (DE). The Expedited Forwarding (EF) class is used to carry UDP traffic coming from time-constrained applications (VoIP, Command/Control, ...); the Assured Forwarding (AF) class deals with elastic traffic as defined in [RFC4594] (data transfer, updating process, ...) while all other remaining traffic is classified inside the default (DE) best-effort class.

The first and best service is provided to EF as the priority scheduler attributes the highest priority to this class. The second service is called assured service and is built on top of the AF class where elastic traffic such as TCP traffic, is intended to achieve a minimum level of throughput. Usually, the minimum assured throughput is given according to a negotiated profile with the client. The throughput increases as long as there are available resources and decreases when congestion occurs. As a matter of fact, a simple priority scheduler is insufficient to implement the AF service. Due to its opportunistic nature of fetching the full remaining capacity, TCP traffic increases until reaching the capacity of the bottleneck. In particular, this behaviour could lead to starve the DE class.

To prevent this and ensure to both DE and AF a minimum service rate, the router architecture proposed in [RFC5865] uses a rate scheduler between AF and DE classes to share the residual capacity left by the EF class. Nevertheless, one drawback of using a rate scheduler is the high impact of EF traffic on AF and DE. Indeed, the residual capacity shared by AF and DE classes is directly impacted by the EF traffic variation. As a consequence, the AF and DE class services are difficult to predict in terms of available capacity and latency.

To overcome these limitations and make AF service more predictable, we propose here to use the newly defined Priority Switching Scheduler (PSS). Figure 4 shows an example of the Data Plane Priority core network router presented in [RFC5865] modified with a PSS. The EF queues have the highest priorities to offer the best service to real-time traffic. The priority changes set the AF priorities either higher (3,4) or lower (6,7) than CS0 (5), leading to capacity sharing. Another example with only 3 queues is described in [Globecom17]. Thank to the increase predictability, for the same minimum guaranteed rate, the PSS reserves a lower percentage of the

capacity than a rate scheduler. This leaves more remaining capacity that can be guaranteed to other users.

```
                               priorities
                                ________
                      queues   |        `
       Admitted EF----->||-----+   1      `
                               |            `
     Unadmitted EF----->||-----+   2          `
                               |                `
               AF1----->||-----+ 3 or 6    PSS    -----------
                               |                '
               AF2----->||-----+ 4 or 7       '
                               |            '
               CS0----->||-----+   5      '
                               |________'
```

Figure 4: PSS applied to Data Plane Priority (we borrow the syntax from RCF5865)

3.2. New service offered

The new service we seek to obtain is:

- o for EF, the full capacity of the output link;

- o for AF the minimum between a desired capacity and the residual capacity left by EF;

- o for DE (CS0), the residual capacity left by EF and AF.

As a result, the AF class has a more predictable available capacity, while the unpredictability is reported on the DE class. With good parametrization, both classes also have a minimum rate ensured. Parameterization and simulations results concerning the use of a similar scheme for core network scehduling are available in [Globecom17]

4. Security Considerations

TODO

5. Normative References

[Globecom17]
Finzi, A., Lochin, E., Mifdaoui, A., and F. Frances, "Improving RFC5865 Core Network Scheduling with a Burst Limiting Shaper", Globecom , 2017, <acceptedPaper>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <https://www.rfc-editor.org/info/rfc4594>.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <https://www.rfc-editor.org/info/rfc5865>.

Authors' Addresses

Fred Baker
Santa Barbara, California 93117
USA

Email: FredBaker.IETF@gmail.com

Anais Finzi
ISAE-SUPAERO
10 Avenue Edouard Belin
Toulouse 31400
France

Phone: 0033561338735
Email: anais.finzi@isae-supaero.fr

Fabrice Frances
ISAE-SUPAERO
10 Avenue Edouard Belin
Toulouse 31400
France

Email: fabrice.frances@isae-supaero.fr

Emmanuel Lochin
ISAE-SUPAERO
10 Avenue Edouard Belin
Toulouse 31400
France

Email: emmanuel.lochin@isae-supaero.fr

Ahlem Mifdaoui
ISAE-SUPAERO
10 Avenue Edouard Belin
Toulouse 31400
France

Email: ahlem.mifdaoui@isae-supaero.fr