



**HAL**  
open science

## Investigation of Ethernet switches behavior in presence of contending flows at very high-speed

Sébastien Soudan, Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Primet, Ryousei Takano

► **To cite this version:**

Sébastien Soudan, Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, et al.. Investigation of Ethernet switches behavior in presence of contending flows at very high-speed. [Research Report] LIP RR-2006-38, Laboratoire de l'informatique du parallélisme. 2006, 2+21p. hal-02102407

**HAL Id: hal-02102407**

**<https://hal-lara.archives-ouvertes.fr/hal-02102407v1>**

Submitted on 17 Apr 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



*Laboratoire de l'Informatique du Parallélisme*

École Normale Supérieure de Lyon  
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

***Investigation of Ethernet switches behavior in  
presence of contending flows at very high-speed***

Sebastien Soudan,  
Romaric Guillier,  
Ludovic Hablot,  
Yuetsu Kodama,  
Tomohiro Kudoh,  
Fumihiko Okazaki,  
Pascale Primet,  
Ryousei Takano

November 2006

Research Report N° 2006-38

**École Normale Supérieure de Lyon**

46 Allée d'Italie, 69364 Lyon Cedex 07, France

Téléphone : +33(0)4.72.72.80.37

Télécopieur : +33(0)4.72.72.80.80

Adresse électronique : lip@ens-lyon.fr



**INRIA**



# Investigation of Ethernet switches behavior in presence of contending flows at very high-speed

Sebastien Soudan, Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Prinet, Ryousei Takano

November 2006

## Abstract

This paper examines the interactions between layer 2 (Ethernet) switches and TCP in high bandwidth delay product networks. First, the behavior of a range of Ethernet switches when two long lived connections compete for the same output port is investigated. Then, the report explores the impact of these behaviors on TCP protocol in long and fast networks (LFNs). Several conditions in which scheduling mechanisms introduce heavy unfair bandwidth sharing and loss burst which impact TCP performance are shown.

**Keywords:** ethernet switches, queue management, transport protocol, cross-layering.

## Résumé

Ce rapport présente les interactions entre les switchs de couche 2 (Ethernet) et TCP dans les réseaux à haut produit débit-délai. Tout d'abord, nous investiguons le comportement de plusieurs switchs Ethernet lorsque deux connexions se partagent un même port de sortie. Nous explorons ensuite l'influence de ces comportements sur le protocole TCP. Enfin nous exhibons plusieurs situations dans lesquelles le mécanisme d'ordonnancement introduit un partage inéquitable de la bande passante et des pertes en rafales pénalisant les performances de TCP.

**Mots-clés:** switchs ethernet, gestion de file d'attente, protocoles de transport, cross-layering.

## 1 Introduction

Most transport protocol designers addressing wired networks do not take link layer behaviors into account. They assume a complete transparency and determinist behavior (i.e. fairness) of this layer. However, Ethernet switches are store-and-forward equipments, which have limited buffering capacities to absorb congestions that can brutally occur, due to bursty nature of TCP sources [JD05]. Thus many Ethernet switches use contention algorithms to resolve access to a shared transmission channel [FKM04, AOST93, McK99]. These scheduling algorithms aim at limiting the amount of data that a subnet node may transmit per contention cycle. This helps in avoiding starvation for other nodes. The designers of these algorithms have to find a tradeoff between performance and fairness in a range of traffic conditions [MA98]. Considering the case of grid environment where many huge data transfers may occur simultaneously, we explore the interactions between this L2 congestion control mechanisms and TCP and try to understand how they interfere. The goal of this study is to investigate the way packets are managed in high speed L2 equipments (switches) and the implication on transport protocols under different congestion conditions. The question is to understand how the bandwidth and losses are distributed among flows when traffic profiles correspond to huge data transfers.

After a brief introduction on switching algorithms, the second section of this paper details the experimental protocol adopted and the observed parameters. In the third part, we present the steady-state behaviors of CBR (Constant Bit Rate) flows' packet scheduling and the switch's characteristic. In the fourth section, this report study the impact on this kind of flows and on TCP. The report ends by a discussion on the problems of switching algorithms in grid context.

## 2 Switching algorithms

Switching algorithms are designed to solve contentions that may occur on output ports of a switch when several inputs ports intend to send packets on the same port. In input-queued switches, this problem is known as head of line blocking problem. There are mainly two types of such algorithms. Some use a random approach by choosing randomly packets among contending input ports as PIM [AOST93] does whereas others (as iSLIP [McK99]) use round-robin. From a global point of view, both can achieve fairness among input port but from a local point of view results can be quite different [Var05, chapter 13].

A user doesn't know *a priori* which algorithms are used in its switches nor if they are of one of these two types.

## 3 Experiment description

In order to observe the bandwidth sharing and loss patterns on a congested output port of a switch, a specific testbed and a restricted parametric space were used to explore 1 Gbps Ethernet port behavior.

### 3.1 Objectives and Test plan

In order to characterize the switches and the impact of switches on TCP, several experiments have been performed.

The first set of experiments consists of running two contending flows at different constant rates. These two flows permit to observe different behaviors of the switch under different congestion level, to highlights the difference of behaviors of different switches using mean and variance of per-flow

output bandwidth. Fine-grained observation can also be made using sequence numbers to observe per-packet switching behaviors in presence of two flows.

Then as TCP tends to send packets per bursts, experiments with one CBR flow and one burst are run. The number of packets of the burst is observed as a function of burst's length.

And lastly two different variants of TCP (BIC and Reno) are evaluated on two different switches. Performance are measured under different situations: with and without SACK as it is designed to improve TCP performance under specific loss pattern and with a 0 ms and 50 ms RTT GigE link as TCP's performance are impacted by losses and high latencies.

These experiments are run the testbed describe in the next section.

### 3.2 Testbed

The topology of the experimental testbed is described on figure 1. One GtrcNET-1<sup>1</sup> [KKT<sup>+</sup>03] is used to both generate traffic and monitor the output flows. GtrcNET-1 is an equipment made at the AIST which allows latency emulation, bandwidth limitation, and precise per-stream bandwidth measurements in GigE networks at wire speed. GtrcNET-1 has 4 GigE interfaces (channels). Tests were performed with several switches but most of the results presented here are based on a Foundry Fast-Iron Edge X424 and a D-Link DGS1216-T. Firmware version of the Foundry switch is 02.0.00aTe1. "Flow control" is disabled on all used ports and priority level is set to 0. According to manufacturer's documentation, D-Link has 512 KB and according to command line interface `show mem` command, Foundry switch has 128 MB of RAM but for both the way memory is shared among ports is not known.

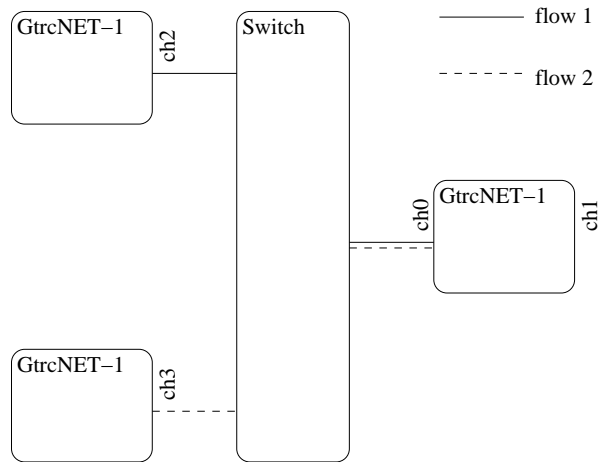


Figure 1: Experimental testbed

### 3.3 Parametric space

We assume L2 equipments do not differentiate UDP and TCP packets. Tests that have been made corroborate this fact. Experiments were conducted with UDP flows as they can be generated easily by GtrcNET-1 and as they can be sent at a constant rate.

<sup>1</sup><http://www.gtrc.aist.go.jp/gnet/>

The following parameters were explored: flows' rate, packet length and measure interval length. Different levels of congestion using different flow's rates were used : 800, 900, 950 and 1000 Mbps. These rates are transmission capacity (TC) used to generate UDP packets. Transmission capacity specify the bitrate (including Inter Frame Gaps and preamble) of an emulated Ethernet link. Experiments were strictly included in the period of packet generation. IP packet length are set to 1500 bytes as high-speed connections use full size packets. In order to observe output flow bandwidths, packets were counted on intervals of 400 and 1000  $\mu$ s (around 33 and 83 packets at 1 Gbps).

### 3.4 Test calibration

We first compare the flows generated by GtrcNET-1 with and without switch to be sure switches do not introduce too much noise. Here, the observation interval is 400  $\mu$ s.

With 1000 Mbps (wire speed) as transmission capacity, due to preamble and IFG, transmission bandwidth will be of  $1000 * (64 + 18)/(64 + 18 + 20) = 803$  Mbps with 64 bytes packets and  $1000 * (1500 + 18)/(1500 + 18 + 20) = 986$  Mbps with 1500 bytes packets. Table 1 summarizes these values.

GtrcNET-1's is able to generate packets at the specified rates. When GtrcNET-1's channel 3 (ch3) is directly connected to ch0 and ch3 generates UDP packets, monitored bandwidth on ch0 is constant.

Then, the measurements with one flow transiting through the Foundry Fast IronEdge X424 switch was done. Figures 3 represent the output bandwidths observed on ch0 when the flow goes through the switch. In these figures, measured bandwidths are not as stable as the ones obtained without switches (figure 2) but the obtained rates are nearly the same.

Transmission capacity (Mbps)	UDP bandwidth (Mbps)	
	64 B packets	1500 B packets
900	724	888
950	764	937
1000	803	986

Table 1: UDP flows bandwidth with respect to transmission capacity used

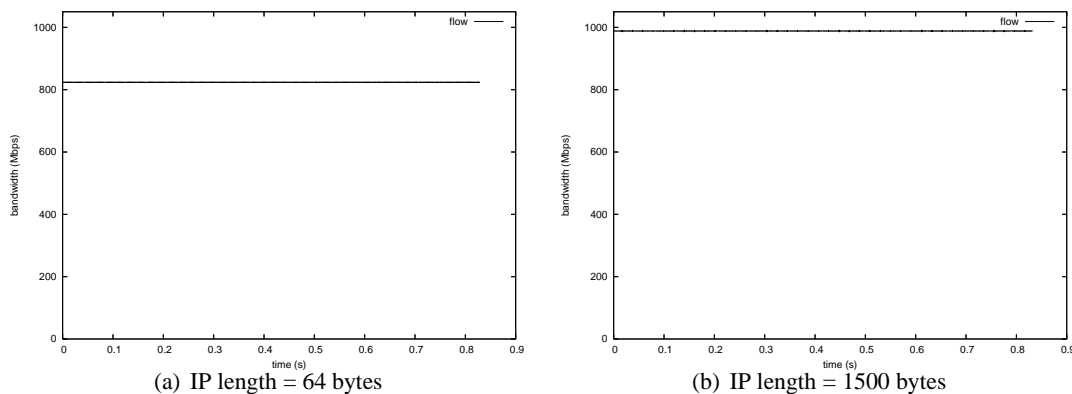


Figure 2: Without switch, TC = 1000

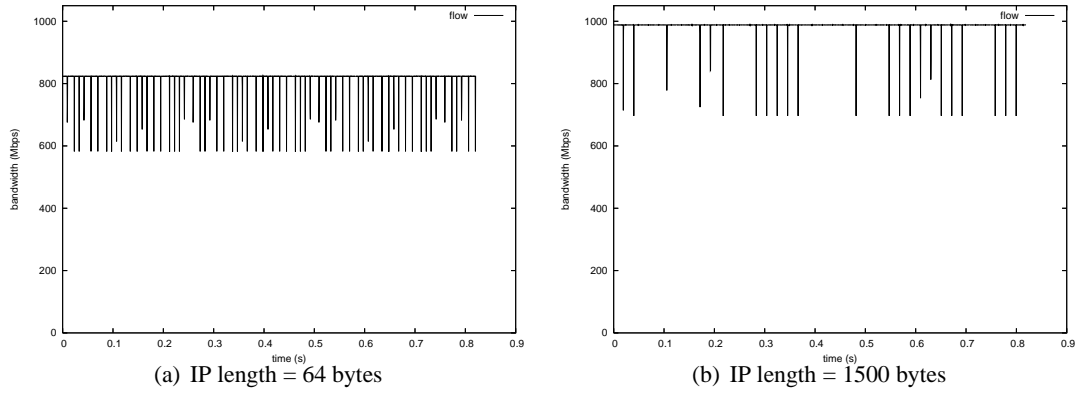


Figure 3: Switch, TC = 1000

## 4 Steady-state behaviors of CBR flows' packets scheduling

This section presents some of the bandwidth patterns observed on the output port of the Foundry Fast IronEdge X424 switch when two CBR (Constant Bit Rate) flows are sent through this output port. We only concentrate on 1500 bytes packets as high throughput flows use such packets size (or over with jumbo frames). Each figure of this section represents the output rate of the two flows (subfigures (a) and (b)). Sums of output bandwidths are always constant at 986 Mbps. Measures presented in this section are made using 1 ms intervals.

### 4.1 Two CBR flows with same rates

Figures presented in this section use the same input rate for the two flows.

In figure 4, only one flow is forwarded at a time. There are many transitions between the two flows but it seems to be completely random. It can be noticed that the aggregated bandwidth is nearly constant and that one flow can starve for more than 100 ms (for example: flow 1 between 1.6 s and 1.7 s).

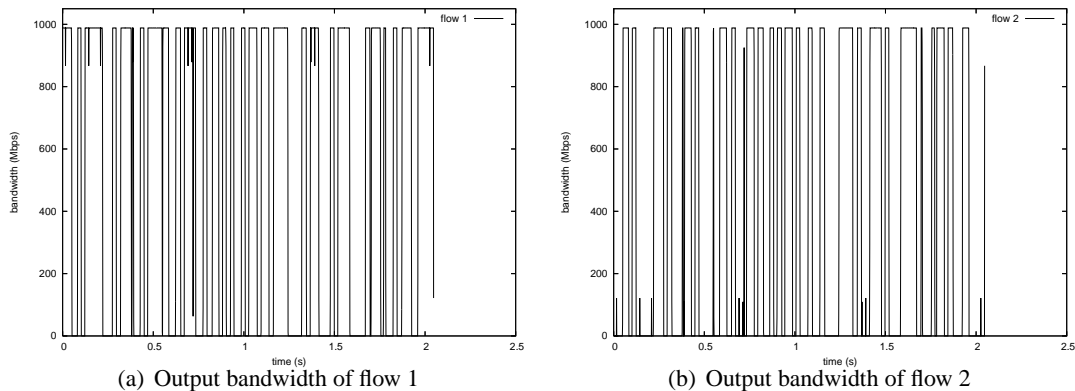


Figure 4: 1000 Mbps + 1000 Mbps (1500 bytes)

In figure 5, flows do not starve but a real unfair sharing is observed for more than 300 ms. From time 1.45 s to 1.75 s, one is running at more than 900 Mbps and the other at less than 50 Mbps.

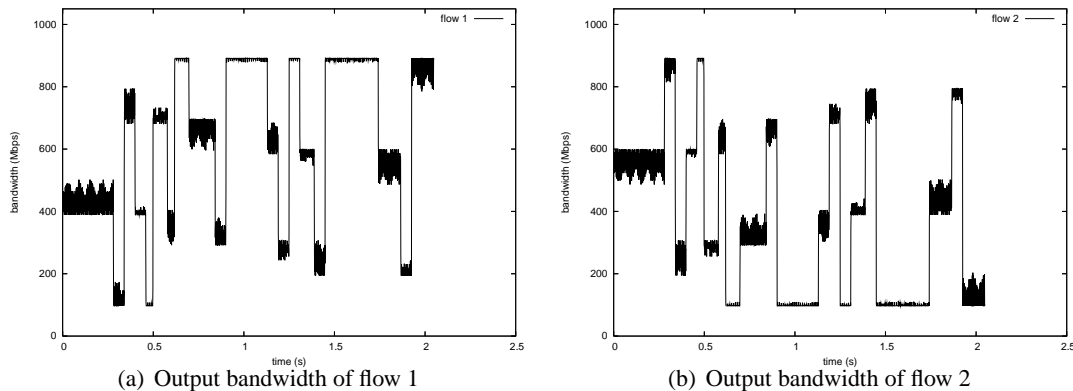


Figure 5: 900 Mbps + 900 Mbps (1500 bytes)

As seen in all the figures shown in this section, sharing among input ports can be really unfair on “short” time scale, which can have a dramatic impact as 100 ms may be a very long interval within TCP’s dynamic specifically at these rates. Around 8300 1500-bytes packets should have been forwarded at 1 Gbps within 100 ms.

## 4.2 Two CBR flows with different rates

In figure 6, it can be observed that when the two flows are sending at different rates (with one at wire speed), instant flow rate on the output port varies among a set of values. And finally, when none of the flows is at 1 Gbps and they have not the same rate, as in figure 7, the sharing of the bandwidth is closer to what we would expect to obtain as with 1 ms interval observation the throughput is nearly constant.

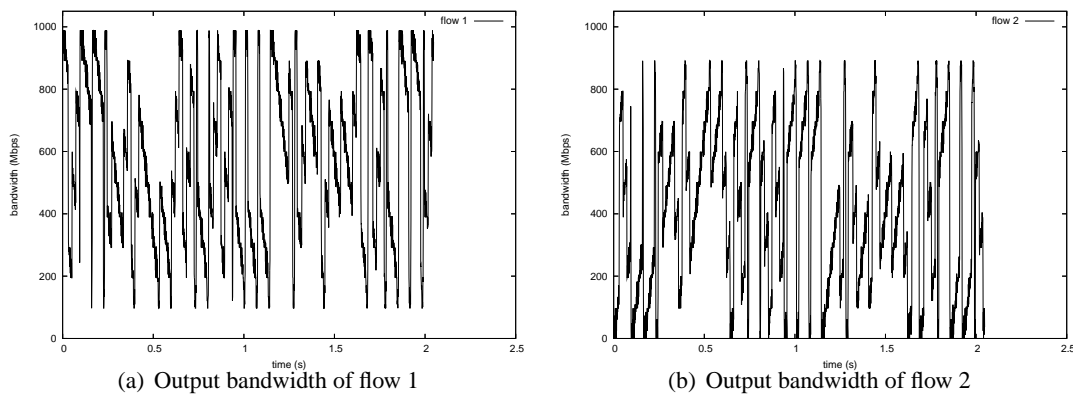


Figure 6: 1000 Mbps + 900 Mbps (1500 bytes)

As a conclusion of this section, when the two CBR flows have the same rate or one of the flow is at the maximum rate, the behavior of this switch is unfair on “short” time scale (100 ms) but is not



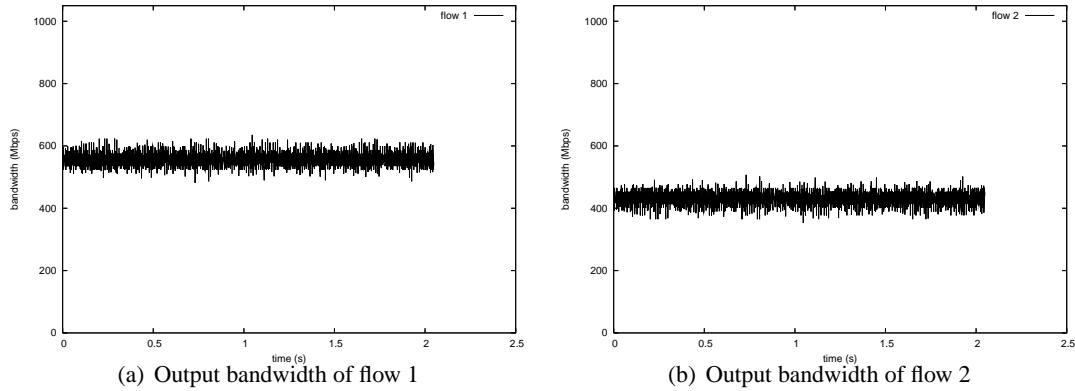


Figure 7: 950 Mbps + 900 Mbps (1500 bytes)

when none of the two rates is at wire rate. The following sections present some quantitative measures on these situations for several switches.

### 4.3 Sequence number analysis

In this section, instead of monitoring the output bandwidth, the sequence numbers of forwarded packets are monitored. Figure 8 shows the situation with two 1000 Mbps flows and figure 9 with two 400 Mbps flows on Foundry switch. In these figures is the sequence number of a packet at the date it was observed on the output port represented by an impulse. It can be noticed that on the first figure (figure 8), only one flow is forwarded at a time most of the time just like observed in figure 4, whereas on the second ones (figure 9) output packets are picked alternatively from the two flows (figure 9 (c) is a zoom-in of a short interval).

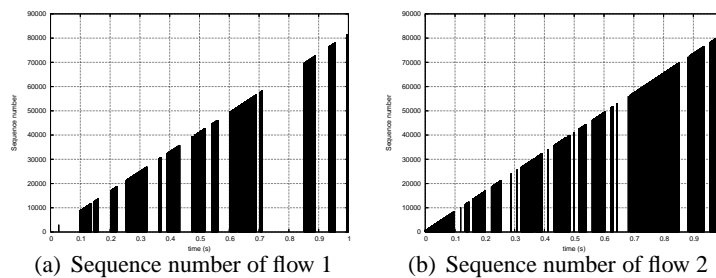


Figure 8: Evolution of sequence number of packets on the output port (1000 Mbps + 1000 Mbps (1500 bytes) on Foundry switch)

On D-Link switch, even when the two input rates are 1000 Mbps (figure 10), packets on output port come alternatively from the two port but packets are alternatively dropped too. Sequence numbers of forwarded packets are growing by from 1 to 3 (figure 10 (c)) as there is only 1000 Mbps of bandwidth on output port and some of the input's packets have to be dropped. This is different from the Foundry switches where packets are dropped by burst. The case with 400 Mbps flows and D-link is similar to Foundry one. This likely means the two tested switches have different queue management strategies.

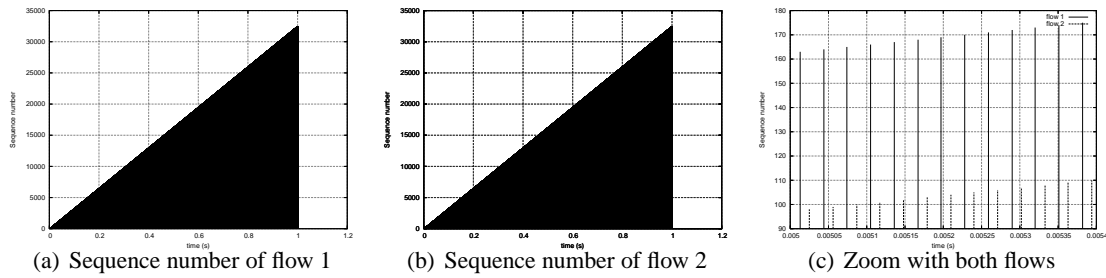


Figure 9: Evolution of sequence number of packets on the output port (400 Mbps + 400 Mbps (1500 bytes) on Foundry switch)

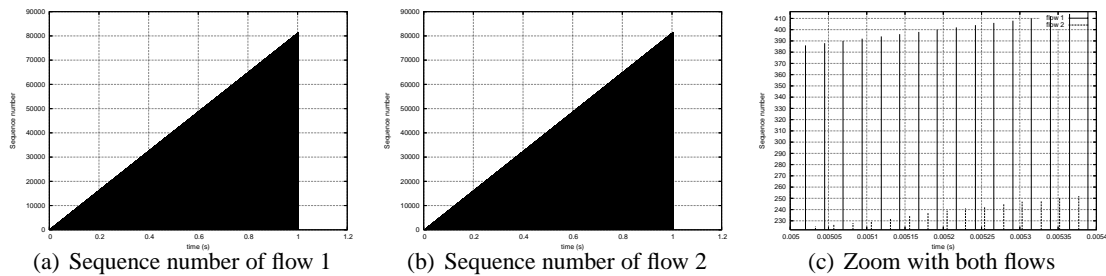


Figure 10: Evolution of sequence number of packets on the output port (1000 Mbps + 1000 Mbps (1500 bytes) on D-Link switch)

#### 4.4 Quantitative measures for CBR flows

Following tables summarize statistical metrics for the two flows on different switches with different input rates. The first two columns indicate the input rates in Mbps for flow 1 and flow 2, the next four columns show average, maximum and minimum throughput (Mbps) and its variance (square Mbps) for flow 1. The next four do the same for flow 2 and finally the last column indicates the interval used for throughput measurements.

It can be observed that tables 2, 3 and 4 show a very high variance and minimum throughput of 0 Mbps when the input rates are equal to 1000 Mbps whereas table 5, 6 and 7 don't. The three first switches are tend to make one of the flows starve for periods of time longer than 100 ms when the congestion is severe. These three switches also perform unfair sharing under high congestion whereas the three last always split the available bandwidth around 494 Mbps when the input rates are equals. In the case of D-Link switch, it occurs even when the input rates are different and the output port is congested.

To conclude this section, it seems switches divide in two different classes. In the first one starvation can occur and high variance under severe congestion can be experienced. In the second one low variance and no starvation occurs. As TCP connections have no knowledge of which switches the network is made of, it can be guessed that the behavior and performances of the connections can be highly and differently impacted as it will be shown in section 5.

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	800	988	666	19K	188	323	0	19K	100 ms
800	800	197	197	197	0	792	791	792	0	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	569	574	566	4	419	422.52	414	3	100 ms

Table 2: Two CBR flows on Foundry FastIron Edge X424

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	525	988	0	203K	463	988	0	203K	100 ms
800	800	785	787	635	233	203	353	200	233	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	567	575	547	12	419	441	413	12	100 ms

Table 3: Two CBR flows on Cisco 4948

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	172	988	0	113K	816	988	0	113K	100 ms
800	800	738	746	735	3	250	253	242	3	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	577	586	569	18	411	419	401	18	100 ms

Table 4: Two CBR flows on Cisco 3750

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	494	494	494	0	494	494	494	0	100 ms
800	800	494	494	494	0	494	494	494	0	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	497	497	496	0	491	491	490	0	100 ms

Table 5: Two CBR flows on D-Link DGS1216T

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	494	494	494	0	494	494	494	0	100 ms
800	800	494	495	494	0	494	494	494	0	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	565	565	565	0	424	424	423	0	100 ms

Table 6: Two CBR flows on Huawei S5648

Input rate		ch0(flow 1)				ch0(flow 2)				Interval
CH2	CH3	ave	max	min	var	ave	max	min	var	
1000	1000	496	497	494	1	493	494	491	1	100 ms
800	800	492	492	492	0	496	496	496	0	100 ms
500	500	494	494	494	0	494	494	494	0	100 ms
800	600	551	551	550	0	437	438	437	0	100 ms

Table 7: Two CBR flows on DELL 5224

#### 4.5 Steady-state Switch's characteristic for CBR flows

While previous section shown metrics in a small number of situation for several switches, this section will present some metrics for two switches for all input rates. In order to characterize switching behaviors, the ratio of output bandwidth divided by input bandwidth were measured for every input rates (from 0 to 1 Gbps by 20 Mbps). Variance of the flows among experiments were also measured. Each experiments last 12 seconds, measurements have been done on 1 ms intervals and have been repeated 3 times.

Figure 11 shows the isoline of the ratio of output bandwidth divided by input bandwidth of flow 1 (on subfigure (a) and of flow 2 on subfigure (b)) on Foundry switch. X axis is the input rate of the first flow and Y axis the one of the second flow. It can be observed that the isolines tends to join at one point – (1000,0) for flow 2. When there is no congestion (below the line joining (0, 1000) and (1000, 0)), the ratio is equal to 1. Figure 12 shows the graphic obtained for the D-link switch. Here, the behavior is completely different and probably related to the packet switching algorithms used. With this switch, if the input rate of the flow 2 is less or equal to 500 Mbps, its output rate is always equal to the input rate regardless of the input rate of flow 1.

We weren't able to find an explanation for strange pattern observed on the bottom right of figures 11 (a) and 12 (a) yet.

Figure 13 shows the standard deviation of the output bandwidth for the 2 flows with the Foundry switch (with 1 ms measurements' interval). The standard deviation is quite low in the usual case. But when the input rates are the same or when one of them is at the maximum, more deviation can be observed. The highest standard deviation is obtained when the two flows are at 1 Gbps. That is when alternate complete starvation of one of the flows was observed. Figure 14 is similar to the previous one but realized with the D-Link switch. Even below the congestion limit, output bandwidth vary on this switch. However magnitude of standard deviation does not grow as high as with the Foundry switch.

As variance isolines shown, predictions on the behavior of a switch are not easy to made *a priori*. Next section will show that the differences between the two switches tested in this section impact TCP performances.

## 5 Impact on TCP

In the previous sections, strange behaviors of switches were observed when the congestion level is very high. As TCP uses a congestion avoidance mechanism, one can assume this prevents the occurrence of such high congestion level on switches' output ports. However in the slow start phase as the congestion window is doubling at each RTT and during aggressive congestion window increase phases (as in BIC [XHR04]), flows can face severe congestions.

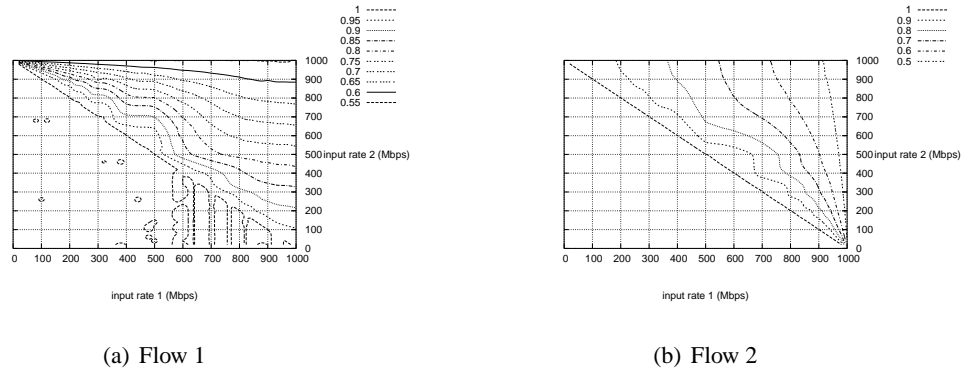


Figure 11: Isoline of output bandwidth over input bandwidth for the 2 flows on Foundry switch

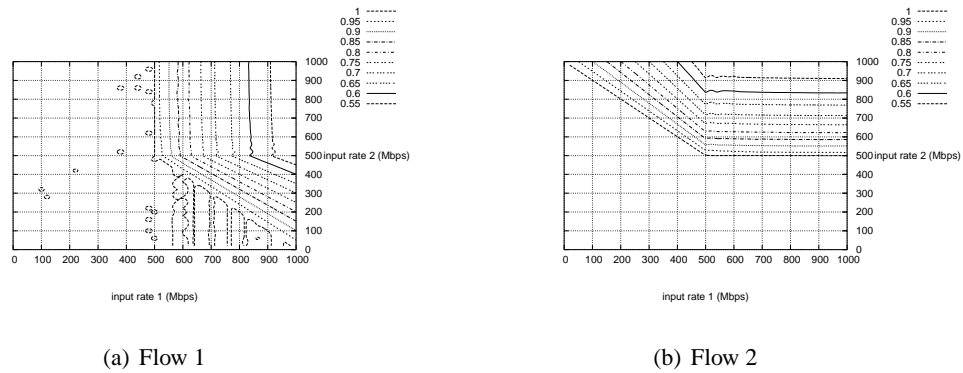


Figure 12: Isoline of output bandwidth over input bandwidth for the 2 flows on D-Link switch

## 5.1 Slow start

In this section, we study the impact of L2 packet scheduling algorithms on already established flows when a new connection starts. The testbed used is similar to the one presented before but the first CBR flow generated by GtrcNET-1 was replaced by a burst of variable length generated by *pktgen* linux kernel module. In this experiment, the bandwidths obtained by the CBR flow and the burst were measured. We assume that the amount of CBR flow's lost bandwidth corresponds to a number of packets lost as in a long run situation, the switch can't buffer all the packets. Figure 15 represents the estimated number of loss that the first flow experienced as a function of the length of the burst with different switches. It can be seen that generally the burst get most of his packets going through the output port, which causes a large dent on the CBR flow. But again two different behaviors can be observed. Figure 15 (b) shows very regular lines for the DELL, D-Link and Huawei switches whereas they are very noisy for the Cisco and Foundry switches (figure 15 (a)) which might indicate these switches use more sophisticated algorithms.

As switches differently drop packets and it impacts TCP, the next section will compare two TCP variants (BIC and Reno) under two latencies (0 ms RTT and 50 ms RTT), with and without SACK on D-Link and Foundry switches.

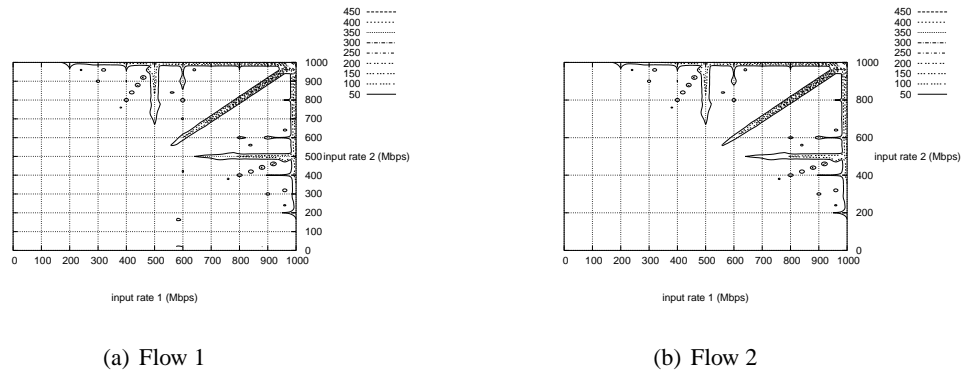


Figure 13: Isoline of standard deviation of output bandwidth of the 2 flows on Foundry switch

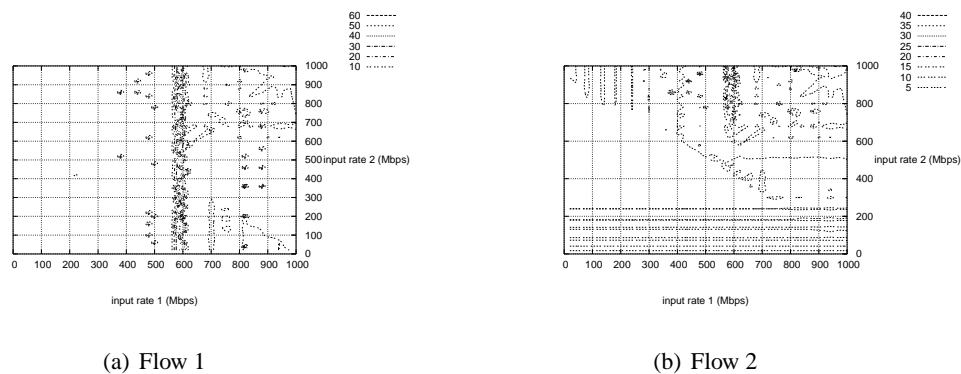


Figure 14: Isoline of standard deviation of output bandwidth of the 2 flows on D-Link switch

## 5.2 Transport protocols comparaison on different switches

Experiments presented in this section use four hosts: two senders and two receivers, all running iperf. The two flows involved share a 1 GbE link of configurable RTT: 0 ms or 50 ms. Bottleneck takes place in the switch before this link. We observe the two flows on this link using the GtrcNET-1 box. All the experiments share the same experimental protocol: first flow is started for 400 s, 20 s later second flow is started for 380 s.

In these experiments, TCP buffers were set to 25 Mbytes and `txqueueLen` to 5000 packets to avoid software limitation on end hosts.

First observation, the first flow manages to fill the link by itself in all situations except for Reno with 50 ms RTT on D-Link switch where a loss occurs during the first seconds (figures 28 (a) and 30 (a)).

The two next sections will present graphics for 0 ms RTT and then 50 ms RTT GigE links.

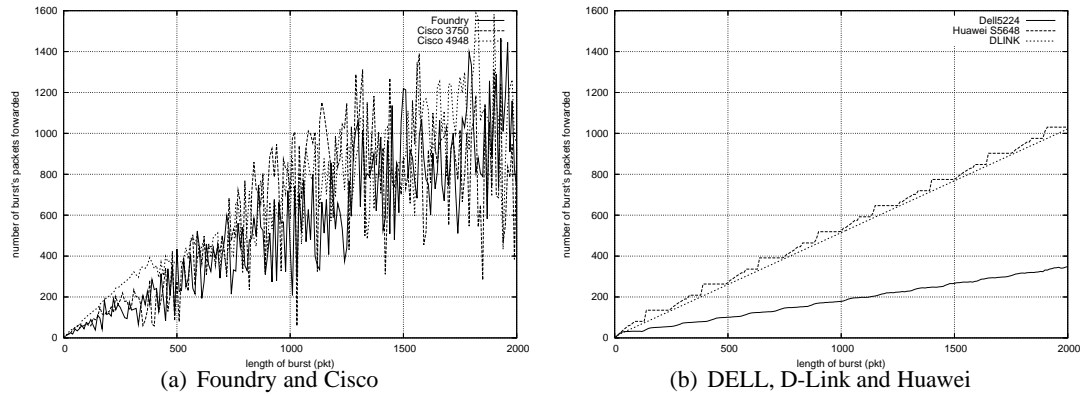


Figure 15: 1000 Mbps CBR flow's losses

### 5.2.1 0 ms RTT

Figures presented in this section represent flows' throughputs on a 0 ms RTT GigE link. Some figures present periods of time where one of the flow is nearly silent. We can observe such behaviors on figure 17 (a) between seconds 320 and 340, on figure 19 (b) between seconds 30 and 50 and between seconds 150 and 175 and finally on figure 25 (a) between 200 and 220. We didn't observe such starvation on D-Link switch nor with Reno TCP.

Comparison clearly shows the interest of using SACK as it "smooths" the bandwidth usage of the two flows, no more stop and go when packets are lost in the switch. Absence of SACK is the worst case for Foundry switch, as in this situation we can observe in figure 17 that only one flow is forwarded at a time. It can be noticed that TCP default configuration provided by current linux kernel (2.6.18) is BIC with SACK.

**BIC**

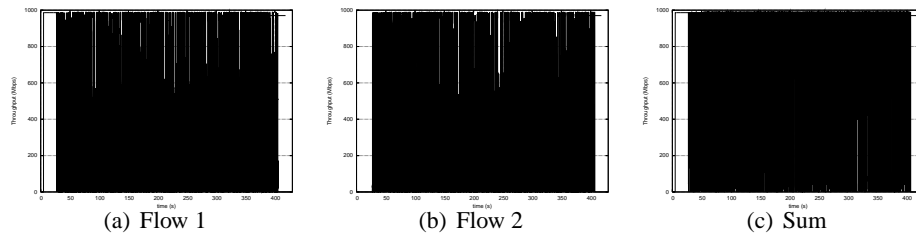


Figure 16: Two BIC flows without SACK (0 ms RTT) on D-Link switch

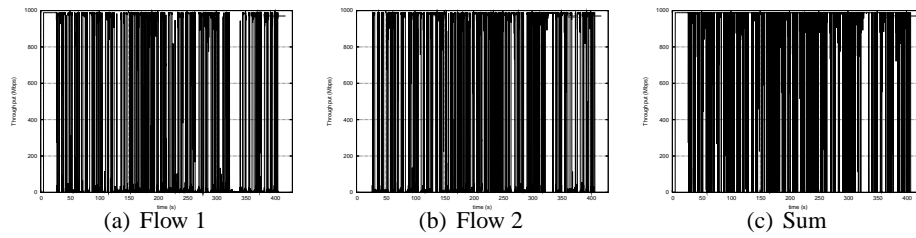


Figure 17: Two BIC flows without SACK (0 ms RTT) on Foundry switch

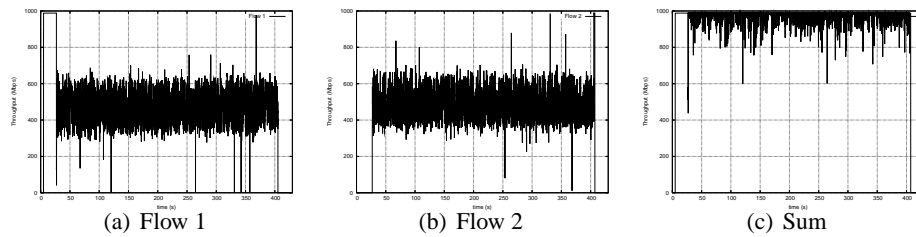


Figure 18: Two BIC flows with SACK (0 ms RTT) on D-Link switch

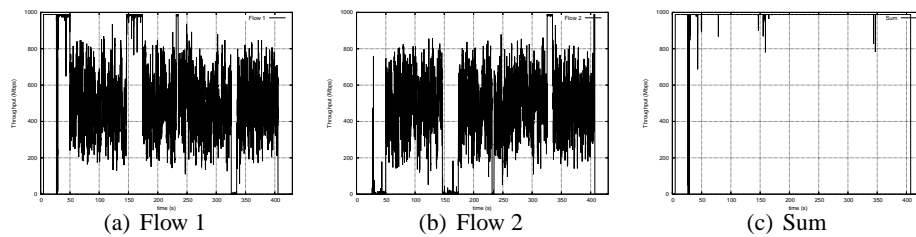


Figure 19: Two BIC flows with SACK (0 ms RTT) on Foundry switch



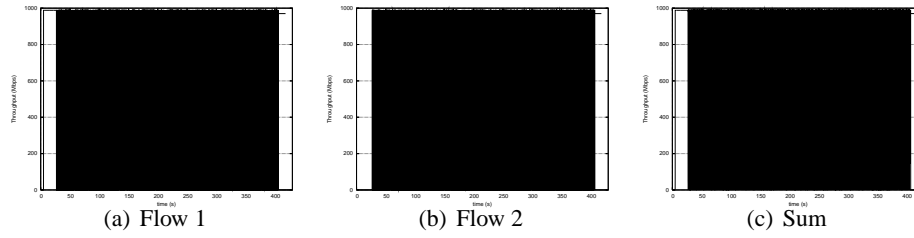
**Reno**

Figure 20: Two Reno flows without SACK (0 ms RTT) on D-Link switch

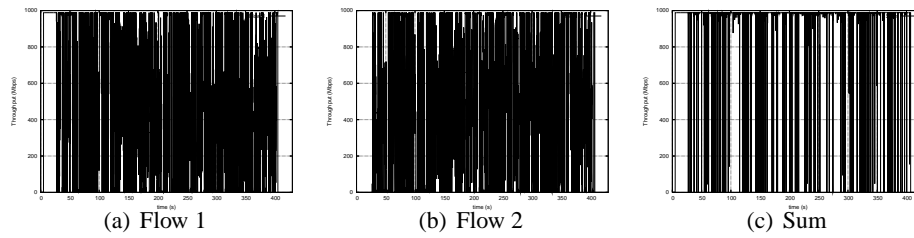


Figure 21: Two Reno flows without SACK (0 ms RTT) on Foundry switch

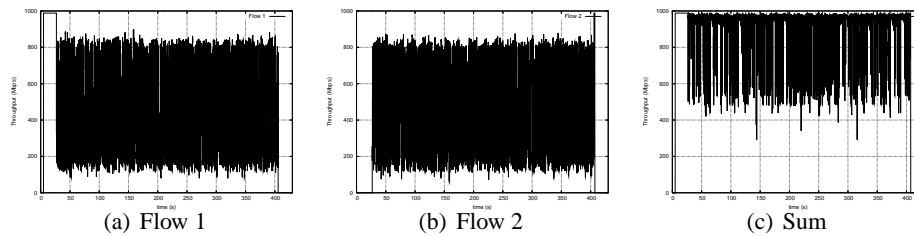


Figure 22: Two Reno flows with SACK (0 ms RTT) on D-Link switch

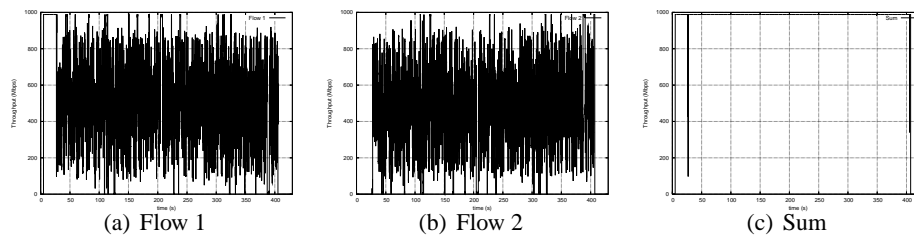


Figure 23: Two Reno flows with SACK (0 ms RTT) on Foundry switch

### 5.2.2 50 ms RTT

Figures presented in this section represent flows on an emulated 50 ms RTT GigE link. As the latency is higher, TCP variants are less reactive and losses have an impact on performance.

We can observe on figure 29 and 31 that Foundry switch is able to bufferize some packets from one RTT to next one. Smoothed RTT estimation (from Web100 [MHR03] variables) in the situation of figure 29, shows an increase from 52 ms at 75 s to 80 ms just before time 150 ms, which corroborates this observation.

### BIC

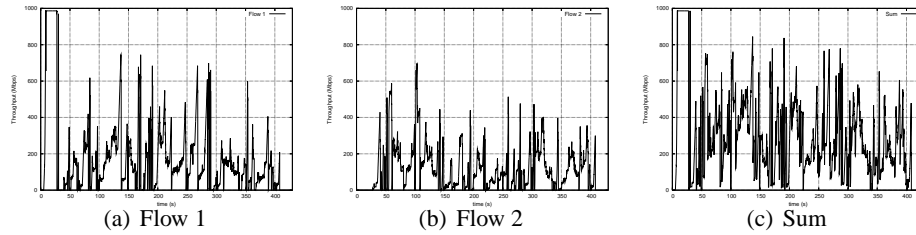


Figure 24: Two BIC flows without SACK (50 ms RTT) on D-Link switch

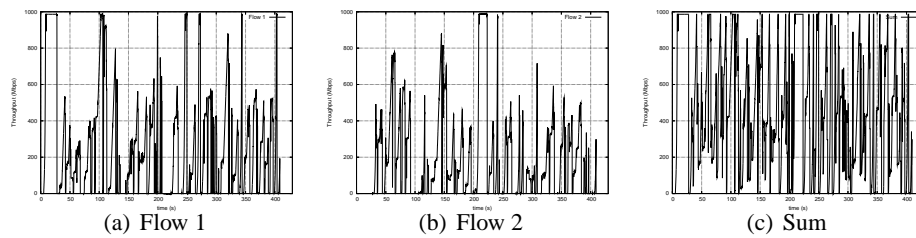


Figure 25: Two BIC flows without SACK (50 ms RTT) on Foundry switch

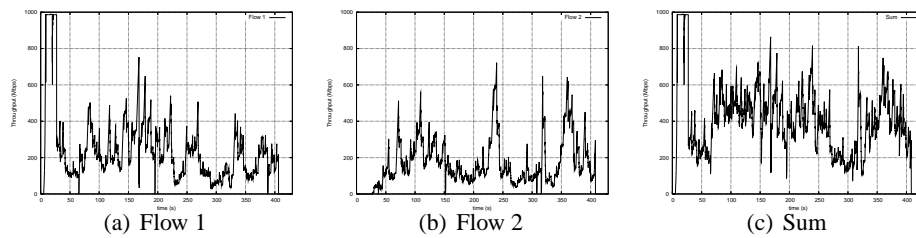


Figure 26: Two BIC flows with SACK (50 ms RTT) on D-Link switch

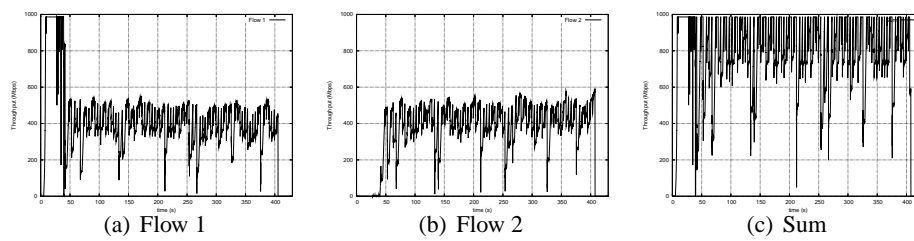


Figure 27: Two BIC flows with SACK (50 ms RTT) on Foundry switch

**Reno**

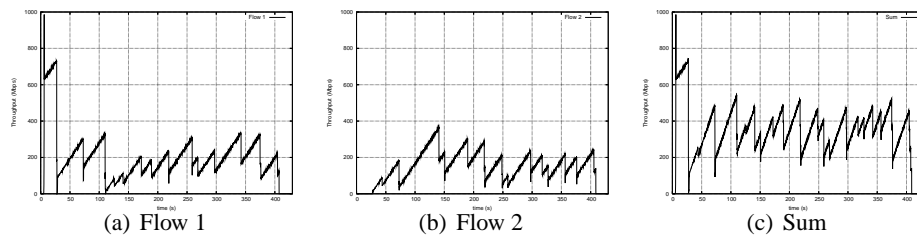


Figure 28: Two Reno flows without SACK (50 ms RTT) on D-Link switch

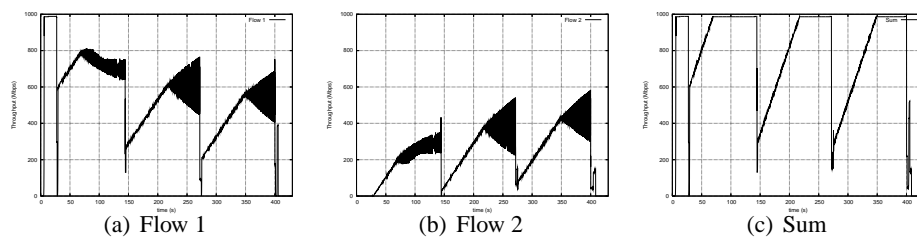


Figure 29: Two Reno flows without SACK (50 ms RTT) on Foundry switch

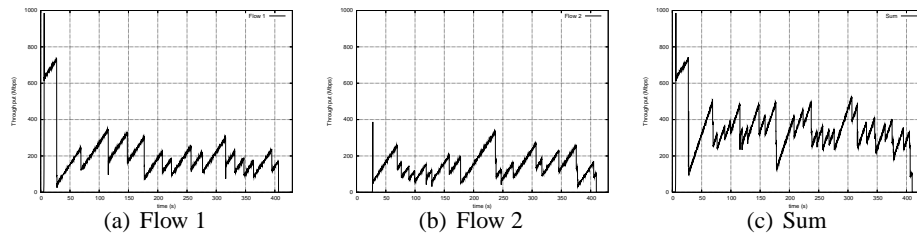


Figure 30: Two Reno flows with SACK (50 ms RTT) on D-Link switch

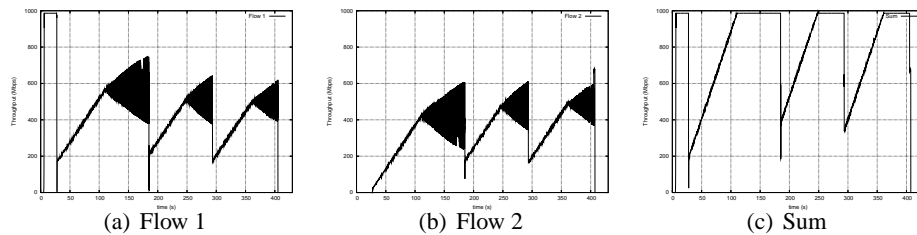


Figure 31: Two Reno flows with SACK (50 ms RTT) on Foundry switch

### 5.2.3 Quantitative comparison

We summarize in this section results of the previous sections. Data presented in this section have been collected using Web100 Linux kernel patch. Table 8 represents the mean goodput in Mbps of each flow in the different experiments, table 9 represents the number of retransmission per second and finally table 10 represents the number of timeouts per seconds.

On table 8, we can observe that mean goodput is higher when using Foundry switch than D-Link one. We can also notice highest aggregated goodput with 0 ms RTT is obtained on Foundry switch with Reno and SACK whereas the best results are obtained on D-Link with BIC and SACK. With 50 ms both best results are obtained with BIC and SACK. Table 9 highlights the fact that there is more retransmission with D-Link switch than with Foundry with 0 ms RTT but the situation is the opposite with 50 ms RTT. And finally, table 10 shows that more timeouts occur on D-Link switch and that there is more timeouts with 0 ms RTT than with 50 ms.

Figures of previous sections and tables of this sections have shown that the behaviors and performances of TCP variants on different switches can dramatically vary.

TCP variant	Sack?	Switch	Mean goodputs	
			0 ms RTT	50 ms RTT
BIC	Yes	Foundry	417 & 543	372 & 413
		D-Link	468 & 491	168 & 238
	No	Foundry	408 & 471	192 & 253
		D-Link	361 & 394	118 & 178
Reno	Yes	Foundry	461 & 505	345 & 434
		D-Link	434 & 461	141 & 201
	No	Foundry	433 & 475	254 & 535
		D-Link	364 & 407	154 & 198

Table 8: Mean goodputs of 2 flows sharing one port for 400 s (Mbps)

TCP variant	Sack?	Switch	Retransmission per seconds	
			0 ms RTT	50 ms RTT
BIC	Yes	Foundry	58.14 & 60.01	14.27 & 15.83
		D-Link	206.75 & 227.22	2.76 & 4.25
	No	Foundry	41.17 & 53.39	6.28 & 7.67
		D-Link	447.68 & 493.23	3.91 & 4.25
Reno	Yes	Foundry	47.65 & 50.02	0.34 & 3.25
		D-Link	183.84 & 192.32	0.17 & 0.59
	No	Foundry	45.10 & 46.78	0.39 & 0.46
		D-Link	100.58 & 89.32	0.10 & 0.12

Table 9: Number of retransmissions per seconds for 2 flows sharing one port for 400 s (pkt/s)

## 6 Discussion

In this present work switches are being evaluated in a extreme situation which is likely not to be the one for which algorithms were optimized as only three ports are used (two input ports and one output port) with a high congestion level.

TCP variant	Sack?	Switch	Timeouts per seconds	
			0 ms RTT	50 ms RTT
BIC	Yes	Foundry	0.18 & 0.63	0 & 0.01
		D-Link	1.60 & 1.74	0.007 & 0.01
	No	Foundry	1.94 & 2.05	0.05 & 0.06
		D-Link	2.37 & 2.55	0.09 & 0.11
Reno	Yes	Foundry	0.14 & 0.23	0 & 0
		D-Link	2.22 & 2.33	0 & 0
	No	Foundry	1.49 & 1.67	0 & 0.005
		D-Link	2.34 & 2.43	0 & 0.005

Table 10: Number of timeouts per seconds for 2 flows sharing one port for 400 s (timeouts/s)

While IP is the common layer of the Internet, Ethernet can be considered as the common layer of grids. For example, grids often use Ethernet over DWDM as long distance clusters interconnexion. In this situation, congestions between flows take place in Ethernet switches. We can face congestion on a long latency link or local links depending on which nodes are talking together. Papers on IP router buffer sizing assume that queue are droptail FIFO [KT05], but we observed (section 4) queue management in switches are not so simple and different from switch to switch. There are two design issues: queue length but also queue management.

This first work on interaction between transport protocols and layer two equipments highlighted different behaviors and level of performance of these protocols under similar situations. The situation studied in this report is not so uncommon in a grid context where large amount of data can be moved between nodes with a low multiplexing level and a non-uniform distribution of sources and destinations. Switching algorithms have not been designed to solve contention in this context. It is unlikely switches vendors publish precise description of their products. Further investigations are then needed to understand what really happen in switches and how to improve protocol in this particular context. For example, are “uplink” ports managed differently? How is the memory managed? What is the buffer length of a given port? Is there different switching strategies applied depending on inputs’ “load”? When are triggered backpressure through Ethernet PAUSE packets? Nevertheless designing switching algorithms and Ethernet equipments taking into account future contending large data movements seems quite important as performances observed are not optimal (the output link is not fully filled), the sharing is not fair and performance are not predictable.

## 7 Conclusion

Packet scheduling algorithms for Ethernet equipments have been designed for heterogeneous traffics and highly multiplexed environments. Nowadays Ethernet switches are also used in situations where these assumptions can be incorrect such as grid environments. This report shows several conditions in which these scheduling mechanisms introduce heavy unfairness (or starvation) on large intervals (300 ms) and loss burst which impact TCP performance. These conditions correspond to situations where huge data movements occur simultaneously. It also shows that behaviors are different from switch to switch and not easily predictable. These observations offer some tracks to better understand layer interactions. They may explain some congestion collapse situations and why and how parallel transfers mixing packets of different connections take advantages over single stream transfers.

We plan to pursue this investigation of layer two - layer four interaction and explore how to model

it and better adapt control algorithms to fit the new applications requirements. We also plan to do the same precise measurements with flow control (802.3x) and sender-based software pacing [TKK<sup>+</sup>05] enabled which both tend to avoid queue overflows.

## 8 Acknowledgments

This work has been funded by the French ministry of Education and Research, INRIA, and CNRS, via ACI GRID's Grid5000 project and ACI MD's Data Grid Explorer project, the IGTMD ANR grant, Egide Sakura program, NEGST CNRS-JSP project.

A part of this research was supported by a grant from the Ministry of Education, Sports, Culture, Science and Technology (MEXT) of Japan through the NAREGI (National Research Grid Initiative) Project and the PAI SAKURA 100000SF with AIST-GTRC.

## References

- [AOST93] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High-speed switch scheduling for local-area networks. *ACM Trans. Comput. Syst.*, 11(4):319–352, 1993.
- [FKM04] Morteza Fayyazi, David Kaeli, and Waleed Meleis. Parallel maximum weight bipartite matching algorithms for scheduling in input-queued switches. *IPDPS*, 01:4b, 2004.
- [JD05] Hao Jiang and Constantinos Dovrolis. The origin of TCP traffic burstiness in short time scales. In *IEEE INFOCOM*, 2005.
- [KKT<sup>+</sup>03] Y. Kodama, T. Kudoh, T. Takano, H. Sato, O. Tatebe, and S. Sekiguchi. GNET-1: Gigabit ethernet network testbed. In *Proceedings of the IEEE International Conference Cluster 2004*, San Diego, California, USA, September 20-23 2003.
- [KT05] Anshul Kantawala and Jonathan Turner. Link buffer sizing: A new look at the old problem. In *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)*, pages 507–514, Washington, DC, USA, 2005. IEEE Computer Society.
- [MA98] Nick McKeown and Thomas E. Anderson. A quantitative comparison of iterative scheduling algorithms for input-queued switches. *Computer Networks and ISDN Systems*, 30(24):2309–2326, December 1998.
- [McK99] Nick McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.*, 7(2):188–201, 1999.
- [MHR03] Matt Mathis, John Heffner, and Raghu Reddy. Web100: extended TCP instrumentation for research, education and diagnosis. *SIGCOMM Comput. Commun. Rev.*, 33(3):69–79, 2003.
- [TKK<sup>+</sup>05] R. Takano, T. Kudoh, Y. Kodama, M. Matsuda, H. Tezuka, and Y. Ishikawa. Design and evaluation of precise software pacing mechanisms for fast long-distance networks. In *In Proceedings of the International workshop on Protocols for Long Distance Networks (PFLDNET)*, Lyon, France, 2005.

- [Var05] George Varghese. *Network Algorithmics: an interdisciplinary approach to designing fast networked devices*. Elsevier, 2005.
- [XHR04] Lisong Xu, Khaled Harfoush, and Injong Rhee. Binary increase congestion control for fast long-distance networks. In *IEEE INFOCOM*, 2004.