LABORATOIRE D'INFORMATIQUE DE L'UNIVERSITE DE FRANCHE-COMTE

EA 4269

A Study on the Benefit of TCP Packet Prioritisation

Eugen Dedu — Emmanuel Lochin

Rapport de Recherche no RR 2011–????

THÈME 4 – Février 2009

L I F C

A Study on the Benefit of TCP Packet Prioritisation

Eugen Dedu, Emmanuel Lochin

Thème 4
OMNI

Février 2009

Abstract: This paper studies and analyses the benefits of favouring the transfer of packets of a TCP flow over a best-effort network. Specifically, we aim at studying whether we could improve the pace of short data request, such as HTTP request, by giving a high priority to TCP packets that are not previously enqueued inside a core router. Following the idea that long-lived TCP flows greatly increase the routing queue delay, the motivation of this work is to minimise the impact in terms of delay, introduced by long-lived TCP flows over short TCP flows. Thus, this forwarding scheme avoids to delay packets that do not belong to a flow already enqueued inside a router in order to avoid delay penalty to short flow. We define metrics to study the behaviour of such forwarding scheme and run several experiments over a complex and realistic topology. The results obtained present interesting and unexpected property of this forwarding scheme where not only short TCP flows take benefit of such routing mechanism.

Key-words:

Laboratoire d'Informatique de l'Université de Franche-Comté,
Antenne de Montbéliard — UFR STGI,
Pôle universitaire du Pays de Montbéliard,
25200 Montbéliard Cedex (France)
Téléphone : +33 (0)3 81 99 47 43 — Télécopie +33 (0)3 81 99 47 91

Une étude sur les gains de la prioritisation de paquets TCP

Résumé :

Mots-clés :

Laboratoire d'Informatique de l'Université de Franche-Comté,
Antenne de Montbéliard — UFR STGI,
Pôle universitaire du Pays de Montbéliard,
25200 Montbéliard Cedex (France)
Téléphone : +33 (0)3 81 99 47 43 — Télécopie +33 (0)3 81 99 47 91

1 Introduction

Favouring the TCP connection establishment packets or any others packets belonging to a TCP flow inside a core network is not a novel idea. James Kurose, in his famous text book Computer Networking, suggests that it would be useful to protect from losses TCP packets with a low time-to-live value in order to prevent retransmission of packets that have already done a long travel inside a core network. As another example and in the context of QoS networks, Marco Mellia et Al. [5] have proposed to protect from loss some key identified packets of a TCP connection in order to increase the TCP throughput of a flow over an AF DiffServ class. In this study, the authors observe that TCP performance suffers significantly in the presence of bursty, non-adaptive cross-traffic or when it operates in the small window regime, *i.e.*, when the congestion window is small. The main argument is that bursty losses, or losses during the small window regime, may cause retransmission timeouts (RTOs) which will result in TCP entering the slowstart phase. As a possible solution, the authors propose qualitative enhancements to protect against loss: the first several packets of the flow in order to allow TCP to safely exit the initial small window regime; several packets after an RTO occurs to make sure that the retransmitted packet is delivered with high probability and that TCP sender exits the small window regime; several packets after receiving three duplicate acknowledgement packets in order to protect the retransmission. In this study, the authors are focused on the TCP throughput guarantee which is a primary goal of the DiffServ/AF class [2, 4]. This motivates to protect against losses, packets that strongly impact on the average TCP throughput. In an obvious manner, we cannot guarantee a minimum throughput over a best-effort network. However, we propose to study the prioritisation of certain TCP packets in order to investigate whether we could minimise the transfer delay of short TCP flows without impacting on the long lived connections. In this paper, we study how to exploit router functionality to improve the performance of TCP flows in a best-effort network by giving a higher priority to the first packets of a TCP flow inside a router queue if no others packets belonging to the same flow are already en-queued. One of the main goals of this proposal, called FavourTail, is to investigate and understand the benefits of using a prioritisation forwarding scheme inside a core network. Intuitively, we expect to decrease the transfer delay of small TCP connections but surprisingly, we show that this routing behaviour does not only improve the performance of TCP flows and allows to improve the overall performance in terms of transfer delay when slight congestion occurs. We present the potential benefit of using such solution and analyse the benefit of our scheme over a realistic and complex network topology. We show that the proposed scheme can improve the transfer delay of TCP flows up to 25%.

This paper is organised as follows: the next section 2 presents the problem and gives the motivation of this work. Both sections 3 and 4 evaluate through simulations our FavourTail proposal. Then, we provide some related work section 5 and finally, section 6 concludes and gives the perspectives of this study.

2 Problem formulation and motivation

The purpose of storing packets at a router queue is to absorb temporarily burst of packets. The idea we want to develop in this study is to prevent short data flows to be enqueued due to a burst induced by long-lived traffics.

To solve this problem, a possible solution is to involve only the end points. Suppose a connection which only needs to send 10 packets. If we assume the sender is aware of this small number of packets to transmit, we could propose to let him decide to send them in burst. However, this violates the slow start phase and the congestion control mechanisms principle. As a matter of fact, it results that a mechanism to favour short flows must necessarily involve the routers.

Idea presentation Routers are mandatory to decide if a received packet is either rejected or marked or simply enqueued. In the FIFO scheduling policy, the packet is inserted at the top of the queue.

Our packet scheduling proposal, called FavourTail, is the following. The enqueuing packet process is changed. When a packet is enqueued, a check is made in the whole queue to seek another packet from the same flow. If no other packet is found, it becomes a priority packet, otherwise it is added as normally at the top of the queue. Priority packets are added at the beginning of the queue, right after the last priority packet (if there are any). The packet reordering inside a flow is thus avoided.

This scheme is quite similar to a priority queuing scheduling mechanism [7].

FavourTail changes the packet scheduling policy; as such, it can be used with any other queue management policy such as DropTail and RED. In the simulations below we choose DropTail.

Variants There are many variants of this idea which do not involve flow states on routers. We plan to work on them in order to understand their effect and drive performances comparison measurements. The following details some of possible variants:

- Instead of a binary function (insert the packet in the priority queue or in the normal one), use another function $f(n, m)$ giving the level in the queue where the packet will be inserted, where n is the number of packets of the same flow in the queue, and m is the total number of packets in the queue. f might be linear, exponential, logarithmic and so on. For example, in classical FIFO, $f(n, m) = m$, i.e. the packet is always added on top of the queue. (This is not the same as Fair Queuing, where f depends on the number of packets of other flows too);

- Instead of having only two queues, it is possible to have several queues, for example one for packets with 0 other packets from the same flow, another for packets with 1-2 packets, one for 3-5 packets and the least priority queue for all the other packets. However, this solution might introduce an important overhead;
- Act on dropped/marked packets too: when a packet is to be dropped and if it is a priority packet, then choose the last normal packet to act on instead of it.

Dimensioning We are interested to know if FavourTail might have good results in realistic cases. As an example, suppose a flow of 8Mb/s (equivalent to 1000 pkts/s with a 1000 bytes packet size). Suppose also an RTT of 50ms. It results 50 pkts/RTT (1000 pkts/s * 50ms/RTT). If each direction has half of them, then there are 25 data packets in flight. If there are 10 routers between source and destination, then there are 2.5 pkts/router in average. We therefore consider that there are a few routers where this flow is still prioritised. However, it is more consistent to consider the overall gain obtained by the flow rather than the number of times this flow got a packet with a high priority.

Characteristics

- FavourTail does not only favour the beginning of connection but also flows with few packets in flight, i.e. with small congestion window;
- There is no relation between the routers, i.e. a flow with ten packets in flight, one in each router, is favoured, while another flow with only two packets in flight, both on the same router, is not favoured. This is especially true in the TCP slow start phase: packets are generated in burst, so the probability to have several packets of this flow in a router is higher. Evenly-spaced packet sending, such as in TFRC, should be much more prioritised, because their packets are distributed across all the routers;
- There are still several cases where a flow with a very small transmission time has the same transmission time in both cases (DropTail and FavourTail): (1) it might have only one intermediate router, so

the chances to be in concurrence with other flows are smaller, (2) it might be because in FavourTail they are never high priority (it crosses routers which have only high priority packets), (3) it might be because in DropTail the routers are empty (so it is like they would be priority), (4) it might be because on the next router of the next router the prioritisation of the packet does not change anything;

- For priority flows, the more routers are in the path, the greater is the gain in terms of transmission time;
- In terms of security, sources cannot cheat by sending packets at a rate making them always priority, because they cannot estimate how many priority packets are inside routers (a priority packet may be put in the head if there is no other priority packet, but it may be put as number 10 is there are already 9 priority packets in the queue);
- FavourTail is NAT-compatible, if source port and destination port are used to identify flows (as they are unique for NAT too);
- The deployment is in the interest of the person who deploys. Indeed, when a router uses FavourTail, his own users are advantaged: clients receive faster Web pages, and servers reply faster to their Internet clients;
- Starvation may occur for long flows in a router which receives only priority packets. In order to remove this, the normal queue could be served from time to time even if the priority queue is not empty; we will tackle this in a future work;
- A solution which acts on two bursty packets exactly like two spaced packets, i.e. which spaces the bursty packets, would avoid TCP problem given below. We reserve this possible enhancement in a future study.

FavourTail has been implemented in ns2 as an extension of the DropTail queue. The next sections present the results.

3 Simulation results on a simple network

3.1 Network topology

We firstly evaluate FavourTail over the simple network topology given figure 1. The links are configured as follows: both access links have a capacity set to 2Mb/s while the bottleneck link has a 1Mb/s capacity. All links have a transfer delay set to 10ms. Two FTP/TCP traffics are generated: the first one, C1, from src1 to dest, and the second one, C2 (in the second simulation

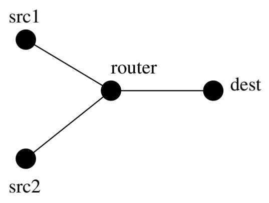


Figure 1: Topology of the simple network.

we use TFRC instead of TCP), from src2 to dest. C1 starts at sec. 0 and ends at sec. 5. C2 starts at sec. 1 and generates only 12 data packets. Obviously, at the beginning of the connection, a SYN packet is generated, and FavourTail will give it a high priority. However, it is important to keep in mind that not only the SYN packet will get a priority. The tests are made with both policies: DropTail and FavourTail.

3.2 Results

TCP The time elapsed between the last packet sent and the first one for C2 is 0.53 for DropTail and 0.43 for FavourTail. The total number of packets sent by C1 is 591 in both cases. No packet is lost.

This is a positive result for the user in terms of transfer duration. Indeed, C1 is not penalised at the end of the connection, while C2 finishes about 20% sooner. The reason is that the two first packets of C2 are prioritised by the router. In fact, when arrived at the router, the first packet overtakes 13 slots, while the second one overtakes 14 slots. The difference in time for C2 ($0.53 - 0.43 = 10$ ms) corresponds to the processing time of packets by the router ($13 + 14 = 27$ packets), time gained by C2.

TFRC If C2 uses TFRC instead of TCP, the results are the following. The time elapsed between the last packet sent and the first one for C2 is 0.54 for DropTail and 0.17 for FavourTail. The total number of packets sent by C1 is 591 in both cases.

This is again a very positive results for the user. C1 is not penalised at the end of the connection, while C2 transfer is about 3 times faster. In fact, 6 packets from C2 are prioritised, gaining each one between 14 and 17 slots.

3.3 Discussion

While the results with this network topology are positive, i.e. the small flow is indeed favoured, the question is why not all its packets are prioritised.

In fact, TCP is a protocol which sends packets in burst. The first burst contains one packet, so it is prioritised. The second burst, sent when the acknowledgement arrives, contains two packets. The first packet is prioritised on the router, but the second one arrives before the first one leaves the router (the link after the router, 10ms/1Mb, is slower than the link before the router, 10ms/2Mb), hence it is not prioritised. As the queue contains many packets from the long flow, this packet is still in the queue when the packets of the following burst arrive. In fact, because the queue has many packets, none of the following packets are prioritised.

On the other hand, TFRC is a congestion control which sends evenly-spaced packets, without burst. More packets than TCP are prioritised, but not all (6 out of 12/13). The reason is that for the seventh packet the throughput of the TFRC flow is a bit higher than the link at the right of the router can process, so it arrives at the router before the sixth packet has leaved it. The following packets suffer from the same problem.

4 Simulation results on a complex network

4.1 Network topology

In order to evaluate FavourTail over a more realistic case, a more complex topology is used in the next simulations. According to a small study of the xDSL backbone of a Internet provider[1], most of these networks are built around a central core where several loops are connected. These loops are composed of a small number of routers. The aim of the closed loop is to have a fault tolerance.

For the simulation, a flower with five loops is considered as backbone, each loop has 8 routers, shown in figure 2. Each router (except the 5 core routers) has 2 DSLAMs connected to it, and each DSLAM has 3 hosts connected to it. Each link has the following characteristics: 10Mb/s bandwidth, 10ms delay, DropTail (or our FavourTail).

We emit 500 FTP over TCP/Newreno connections (flows) with random and non identical hosts as source and destination. Each connection starts at a random time between 0 and 20 seconds and sends a random number between 10 and 600 packets.

4.2 Results

Several metrics are used to compare the classical DropTail and FavourTail, we divide them in global and short flows specific ones.

Global metrics Global metrics refer to metrics about the whole simulation. Table 1 presents some results.

FavourTail globally reduces the transmission time of all the flows. The time reduction is globally dispersed among the flows, as shown in figure 3, but among the flows with higher transmission time, as will be detailed later.

[1]`http://support.free.fr/reseau/province.png`, not available anymore as of August 2008.

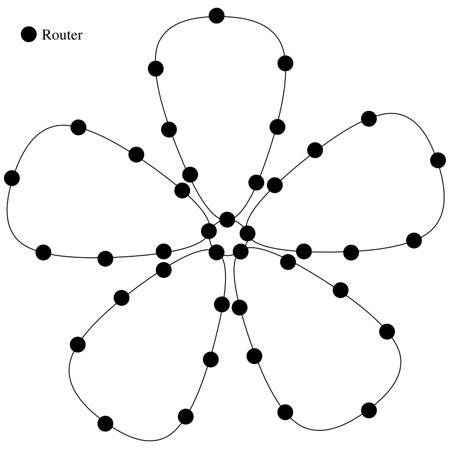


Figure 2: Backbone of the flower network.

	DropTail	FavourTail
Sum of transmission times (in seconds)	2618.68	2410.34
Number of lost packets	2470	1608
Number of lost data packets	913	626

Table 1: Global metrics.

As a matter of fact, if a scheduler treats first the shorter tasks, then the total finish time is smaller. This might explain the reason why we obtain a shorter transmission time.

We also notice that the number of lost packets, and of lost data packets, is much smaller with FavourTail by about 30%.

Short flow specific metrics Our idea favours packets when they are alone (i.e. no other packets from the same flow exist) on their current router. This leads to the idea that flows with few packets to send should be favoured. However, several results prove the contrary.

The transmission time of each flow for both variants is given in figure 4. They are ordered by the transmission time of the FavourTail variant, in figure 4(a), and the DropTail variant, in figure 4(b). In figure 4(a), it seems that flows with short transmission times are indeed favoured. However, this is not true. This is clearly shown in figure 4(b), where for short transmission times (on the left x-axis in this figure), the DropTail variant seems to have smaller times.

It results that using the DropTail or FavourTail transmission time is subjective. An objective comparison, i.e. order the abscissa based on the "length" of flow, is therefore needed. Several criteria may be chosen as length of flow: (1) number of packets sent by the flow, (2) number of packets divided

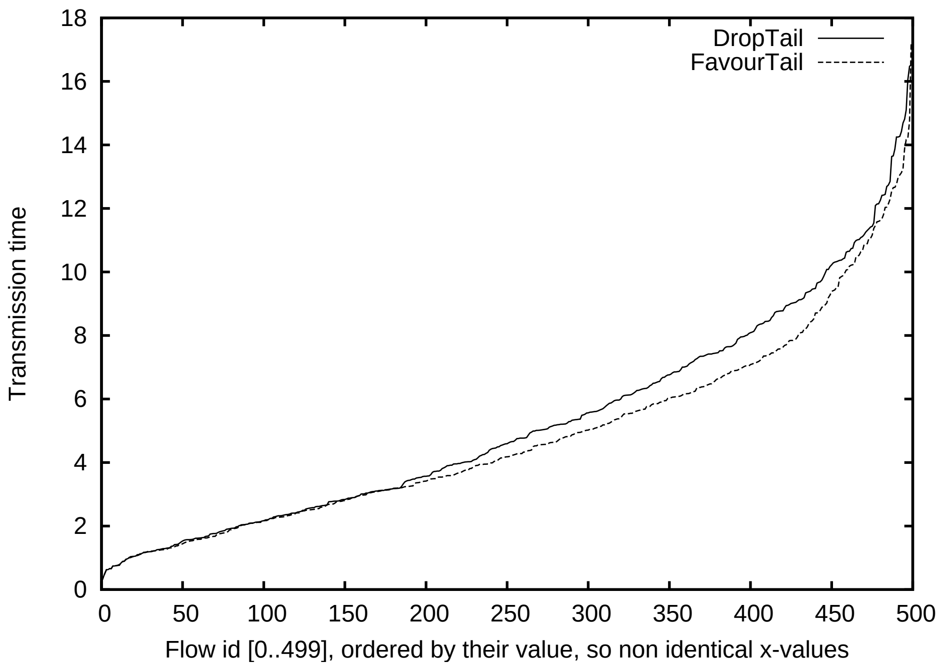
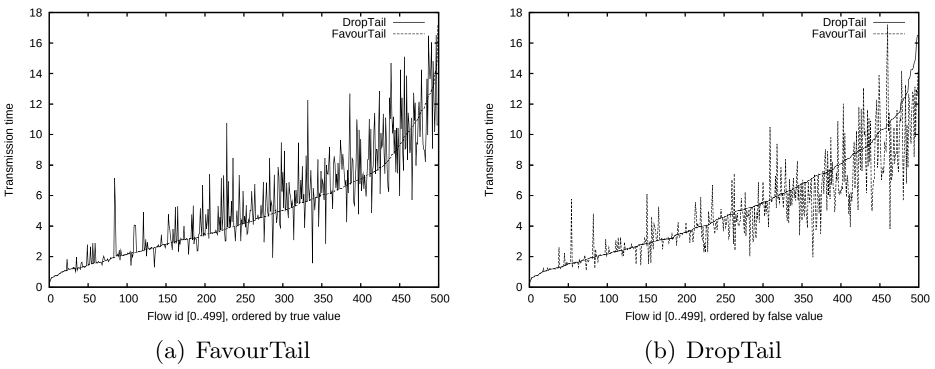


Figure 3: Transmission time of all the flows; the two curves have different values on abscissa.



(a) FavourTail (b) DropTail

Figure 4: Transmission time of all the flows, ordered by the transmission time of FavourTail variant, and DropTail variant.

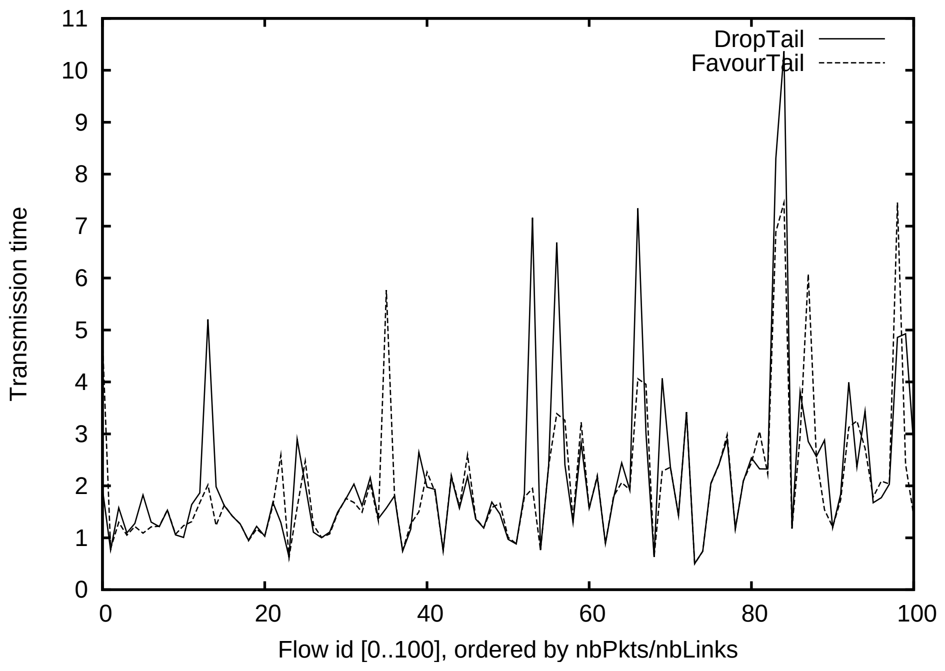


Figure 5: Transmission time of the first 100 flows, ordered by the number of packets divided by the number of flow links.

by the number of intermediate routers (or links), (3) number of packets divided by the number of concurrent flows inside path routers, and so on. For our purposes, i.e. favouring packets with few packets, the most appropriate criterion is the number of packets divided by the number of intermediate routers (we use here a simple ratio function for simplicity purpose). A smaller value means more favoured. In fact, the more the packets, the smaller the gain of FavourTail; the more the number of links, the higher the gain of FavourTail.

Figure 5 presents the transmission time ordered by the number of packets divided by the number of links. The curve for FavourTail is more often below the other curve, so FavourTail is slightly better. The difference between the two curves are not however high. The plot for ordering by the number of packets shows similar results (albeit the difference between both variants is a bit further reduced).

Several parameters of the simulation have been changed, the results are quite similar. The parameters changed are:

- all the traffic is TFRC instead of TCP;
- the data size of flows are not random, but use Pareto distribution;

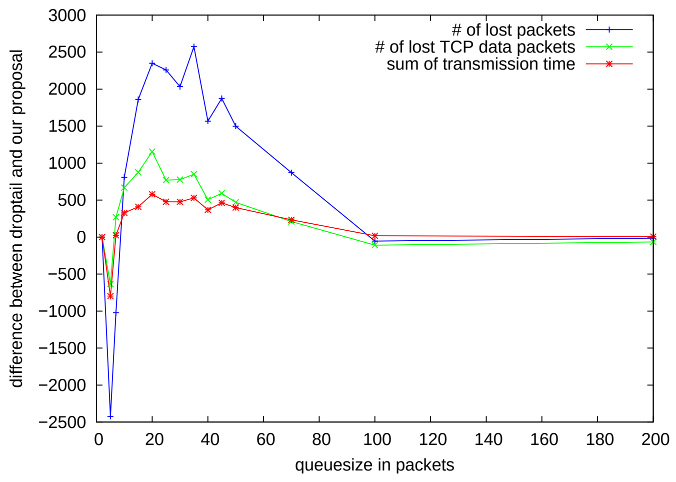
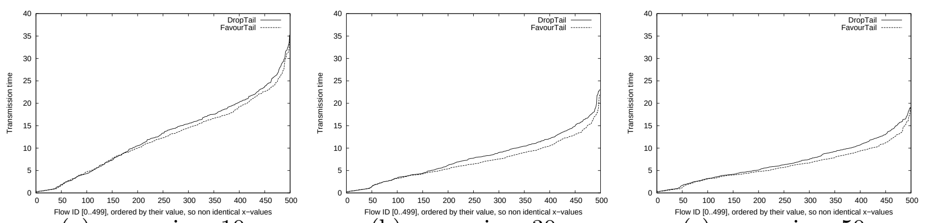


Figure 6: Comparison between DropTail and our proposal.

- the first 50 flows have between 1 and 10 packets, all the others have between 200 and 800 packets to send;
- each core router has attached only one DSLAM, and each DSLAM has only one host attached.

Experiments with variable queue buffer size In this experiment, we vary the queue size of each router buffer from 2 to 200 packets. This allows to emulate a congestion level inside the whole core network. We use the flower network topology previously presented with random size of data transfer. Then, we report both the number of dropped packets and TCP data packets as well as the sum of transmission time of all data transfer during the experiment. These metrics are those already presented at the beginning of this section.

In order to verify the benefit introduced by our solution at a macroscopic level, we show in figure 6 the difference between the results obtained by DropTail and our proposal. When this difference is positive, our proposal gets better performance than DropTail. As expected, and in order to verify our implementation, when the queue size is very small (*i.e.* when the queue size is set to two), the difference is nil meaning there is no advantage for both mechanisms. When the queue size is very large (*i.e.* higher than hundred packets), the performance realized by both mechanisms is quite similar. The fixed TCP window size of all sources explains this later result: the buffer occupancy of each router queue is stabilized since the emitted traffic is not growing anymore due to this fixed size. This result allows to stand that when the buffers size are large, meaning that there is no congestion inside the network, our proposal does not bring any advantage. However, when the



(a) queuesize=10 (b) queuesize=30 (c) queuesize=50

Figure 7: Transmission time of all flows as a function of the queue size.

queue size is ranging from 10 to 70, that can symbolize a congestion level considered as severe to slight, our proposal allows to decrease the number of dropped packets and as a result, the number of retransmitted packets which directly results by a lower transmission time for the TCP flows.

At a microscopic level, the results obtained by all flows are similar to those presented in figure 3 and despite the fact that the overall performance gets better, we cannot see a lower transfer time for short TCP flows as illustrated in figures 7 for various queue sizes.

5 Related work

Many papers are about the router-based methods to optimise the flow transfers. This section presents the most related to our FavourTail proposal.

In [6], routers memorise the number of bytes of each flow passing through them. Upon reception of a packet, the number of bytes of its flow is updated and it is added to the queue so that the packets in the queue be always sorted based on the number of bytes traversed. The consequence is that the flows are given higher priority when they are at the beginning of connection. The drawbacks are that routers have flow states, heavy computations are involved (sorting the queue), and the number of the packet in the flow must be known.

[1] removes this final condition by computing the number of the packet from on the TCP sequence number. For that, it introduces a hypothesis: the starting sequence number must have the last N bits ($N = 22$ proposed in the article) equal to 0. The drawbacks are that TCP senders have to be modified, the sequence number is less invulnerable to guessing attack, and the deployment is difficult: short flows on a standard TCP source will be penalised, since the sequence number of the first packets are misinterpreted by the router as being the Nth packets. Two queues are used and a threshold. Upon reception of a packet, if the number of bytes of that flow is inferior to threshold, it is put in the priority queue, otherwise in the normal queue. The priority queue is FIFO, while the normal one is sorted by the number of

packets traversed. The normal queue is used only when the priority queue is empty.

Another idea is presented in [3]. Only edge routers have flow states, even if not for ever. They count the packets of each flow and set the DiffServ bits of each packet. Core routers use only the DiffServ information. Edge routers mark packets as IN if the current number of packets is inferior to a certain threshold, and OUT if they exceed this threshold.

6 Conclusion

In this paper, we have presented and evaluated a novel forwarding scheme. We show that this scheme leads to interesting properties allowing to decrease the overall transfer delay of TCP flows in the context of best-effort networks. The results obtained are quite unexpected, as intuitively we would expect a stronger benefit of this mechanism to short TCP flows and on the contrary, measurements show an overall benefit for long flows without high impact over short ones with our proposal compared to DropTail. Indeed, the main findings are that FavourTail decreases the number of packets dropped and as a primary consequence, the sum of transmission times is thus reduced. However, the short flows are not noticeably favoured compared to the other longer flows.

In a future work, we aim at investigating, through a larger measurements campaign, this forwarding scheme and in particular with other transport protocols such as TFRC in order to verify whether we would benefit similar properties.

Acknowledgements

Authors thank Pascal Anelli and François Spies for their remarks about packet prioritisation.

References

[1] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows: predictability of the response time. In *INFOCOM*, volume 2 of *7*, pages 762–773, Hong Kong, Mar. 2004. IEEE.

[2] S. Blake, D. Blak, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, Internet Engineering Task Force, Dec. 1998.

[3] X. Chen and J. Heidemann. Preferential treatment for short flows to reduce web latency. *Computer Networks*, 41(6):779–794, Apr. 2003.

[4] D. Grossman. New terminology and clarifications for diffserv. Request For Comments 3260, IETF, Apr. 2002.

[5] M. Mellia, I. Stoica, and H. Zhang. Tcp-aware packet marking in networks with diffserv support. *Comput. Netw.*, 42(1):81–100, 2003.

[6] I. A. Rai, E. W. Biersack, and G. Urvoy-Keller. Size-based scheduling to improve the performance of short TCP flows. *IEEE Networking*, 19(1):12–17, Jan.-Feb. 2005.

[7] D. Zhang, H.; Ferrari. Rate-controlled static-priority queueing. In *In Proceedings of IEEE Infocom 1993*, Mar. 1993.