

Flow Aware Traffic Management

Alberto Blanc, Konstantin Avrachenkov, Sara Alouf, Georg Post

▶ **To cite this version:**

Alberto Blanc, Konstantin Avrachenkov, Sara Alouf, Georg Post. Flow Aware Traffic Management. 5th International Workshop on Traffic Management and Traffic Engineering for the Future Internet (EuroNFTraf '09), Dec 2009, Paris, France. hal-01446775

HAL Id: hal-01446775
https://inria.hal.science/hal-01446775

Submitted on 26 Jan 2017

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.

Flow Aware Traffic Management

Alberto Blanc*, Konstantin Avrachenkov*, Sara Alouf*, Georg Post†
*INRIA Sophia Antipolis Méditerranée †Alcatel-Lucent Bell Labs, Nozay
Email: first.last@sophia.inria.fr Email: first.last@alcatel-lucent.com

I. Background and Motivations

When the TCP and IP protocols were first proposed the emphasis was on a simple best-effort service. Later on TCP was modified in order to address the issue of congestion control, that is to make sure that the senders would not overwhelm the network by sending data too fast. Over the years these protocols have shown an excellent scalability and have allowed the Internet to grow from a network connecting a few research centers to one that connects a very large number of computers all over the world. In its simplest form, which is still by far the most widely used, TCP congestion control works by using drop tail queues that simply drop packets whenever there is no more space in the router's buffers. TCP senders react to these losses by reducing their sending window and, hence, their rate.

This mechanism, while simple and scalable, has several well known limitations: 1) often different flows experience synchronized losses, leading to lower link utilization [5], 2) when flows with different Round Trip Times (RTT) share the same bottleneck link, the flows with a smaller RTT will receive a larger share of the capacity [1]. Not only this allocation of capacity is non-optimal in general, but it cannot be modified as long as only drop tail queues and TCP are used. Over the years several solutions have been proposed to address these limitations, especially the first one. Random Early Detection (RED) [5] was the first, and by far the most widely known and implemented; Active Queue Management (AQM) algorithm aims at improving link utilization by desynchronizing packet losses for different flows, albeit without addressing the fairness issue. Furthermore, while RED can indeed improve performances in some cases, this is not always true [9] and tuning its parameters is non-trivial, hampering its widespread use. Many other AQM algorithms have been proposed (like REM [8], Blue [4], Stochastic fair blue [3], just to name a few), but, to the best of our knowledge, none of them is capable of addressing both issues while being easy to configure and supporting different fairness criteria.

Another advantage of using AQM algorithms is that routers can use Explicit Congestion Notification, in order to inform a sender that it needs to reduce its sending rate (that is to cut its sending window). This reduces the number of dropped packets and can improve performances as the sender does not need to rely on timeouts and/or duplicate acknowledgments in order to infer congestion.

We propose a new flow-aware traffic management mechanism that aims at addressing the two aforementioned limitations of TCP, while being self-configuring and supporting different fairness criteria. Clearly the fairness criteria has to be explicitly selected (configured) by the user; this corresponds to selecting a specific objective, leaving the optimal tuning of parameters to the system. This mechanism is part of Alcatel-Lucent's "Semantic Networking" paradigm [12]. Given that several studies indicate that 20% of the flows are responsible for 80% of the traffic, and that the overwhelming majority of these are TCP flows, we propose to concentrate on controlling these long-lived TCP flows, often called "elephants." In so doing it is possible to improve the Quality of Service/Quality of Experience offered to short-lived and streaming flows. By controlling each elephant individually it is also possible to better control the service received by each one and prevent flows from harming each other.

II. Flow-aware traffic management: our solution and a simplified implementation

In order to offer great flexibility in terms of fairness criteria and service offering we propose to control each long-lived flow individually, decoupling it from all the others. The core idea of the proposed mechanism can be described as a two step process:

1) decide a target rate ($\hat{\lambda}_i$) for each flow;
2) control each TCP flow in order to minimize the oscillations around the chosen target rate.

The second point can be formally expressed as:

$$\min. \frac{1}{T}\int_0^T (\lambda_i(t) - \hat{\lambda}_i)^2 dt, \tag{1}$$

where $\lambda_i(t)$ is the instantaneous rate of flow i. Our conjecture is that, in order to minimize (1), and based on a TCP Reno model, it suffices to send a congestion signal (i.e., mark/drop a packet) whenever the instantaneous rate reaches a peak value

$$k_i \triangleq \frac{3\hat{\lambda}_i(1+\beta)}{2(1+\beta+\beta^2)},$$

where β is the coefficient used by the sender to cut its window for each congestion event, that is, if w is the window size, $w = \beta w$. This forces the sending rate to oscillate deterministically between βk_i and k_i, as shown in Figure 1.

As the TCP window grows linearly (it is incremented by one each RTT) it suffices to mark (drop) packets periodically in order to obtain the behavior depicted in Figure 1. This can be easily implemented using packet counters. In order to compute the number of packets in each period one needs to know the RTT of the flow in question, but this can be estimated using the

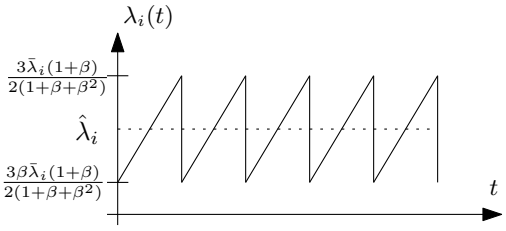


Figure 1. Periodic oscillations of the instantaneous sending rate.

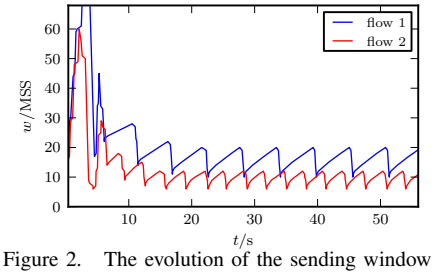


Figure 2. The evolution of the sending window with periodic marking.

algorithm presented in [2] that can estimate the RTT of a long-lived TCP flow on-line and in real-time by observing only one-way traffic. This algorithm can be efficiently implemented on a network processor using only integer arithmetic. Furthermore successive congestion signals are separated by several RTT, so that control decisions are made at a timescale larger (or at least equal) to the RTT.

Similarly to what has been previously suggested (e.g. [7]) we propose to concentrate on controlling only long-lived flows as these carry the overwhelming majority of the traffic. It is reasonable to assume that the number of concurrently active long-lived flows is going to be in the order of hundreds which is orders of magnitude smaller than the total number of active flows. This reduces significantly the needed requirements, in terms of memory and computing power. The only function that still needs to be carried out at wire-speed is packet classification, that is to establish, for each incoming packet, if it belongs to a long-lived flow and if yes to which one. But this is doable with modern hardware, for example by using counting Bloom filters to determine if a packet belongs to the set of long lived flows (counting filters have the advantage that they can be updated so that, as flows terminate, they can be removed from the set of long-lived flows).

So far we have illustrated how we plan to address the second step of our control process. We already have a simplified implementation of this step in ns-3: for the moment, instead of taking the target rate as input, for each flow, the implementation takes the number of packets between two successive marks. This has allowed us to verify that periodic marking is indeed enough to force a periodic behavior in the TCP sending window. It can also be used to analyze the effect of non-linearity in the growth of the window when queueing delays are non-negligible. As an example, Figure 2 shows the evolution of the window for two flows when packets are marked periodically.

In order for our solution to be complete we also need to find a way to compute the target rate for each flow. We describe some possible approaches in the next section.

III. Future Work

Selecting the target rate for each long-lived flow is not a simple task and we can envisage several possible solutions. Trying to achieve max-min fairness is possibly among the simplest ones. This solution has the advantage that it can be determined with local information only, as long as routers can measure the rate of each flow, in order to determine which flows never reach the allotted rate and then allocate the unused bandwidth among the remaining flows. In other words routers can implement the so-called "water-filling" algorithm.

One natural extension is to consider α-fairness [10] but we conjecture that this cannot be done by using only local information, requiring the use of distributed solutions. This introduces the problem of signaling overhead. Nevertheless, it could be worth exploring this solution as it would offer great flexibility in terms of available fairness criteria. Another possible solution is to consider a centralized server that, based on the state of the network, computes the target rate for each flow. Clearly this solution is not the most attractive one but it does have the advantage of being easy to implement. Finally solutions involving end-hosts are possible, similarly to what is done in XCP [6] and RCP [11].

Acknowledgements: This work was done in the framework of the INRIA and Alcatel-Lucent Bell Labs Joint Research Lab on Self-Organized Networks.

References

[1] P. Brown. Resource sharing of tcp connections with different round trip times. In *INFOCOM 2000*, volume 3, pages 1734–1741, 26-30 March 2000.

[2] D. Carra, K. Avrachenkov, S. Alouf, P. Nain, and G. Post. Method for estimating a round trip time of a packet flow, 2009. Patent No. 2416, Filing No. 09305207.4.

[3] W.-C. Feng, D.D. Kandlur, D. Saha, and K.G. Shin. Stochastic fair blue: a queue management algorithm for enforcing fairness. In *INFOCOM 2001*, volume 3, pages 1520–1529, 2001.

[4] W.-C. Feng, K.G. Shin, D.D. Kandlur, and D. Saha. The blue active queue management algorithms. *IEEE/ACM Trans. Netw.*, 10(4):513–528, 2002.

[5] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[6] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02*, pages 89–102, New York, NY, USA, 2002. ACM.

[7] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Minimizing the overhead in implementing flow-aware networking. In *ANCS '05: Proc. of the 2005 ACM symposium on Architecture for networking and communications systems*, New York, NY, USA, 2005. ACM.

[8] D. Lapsley and S. Low. Random early marking: An optimization approach to internet congestion control. In *ICON '99: Proceedings of the 7th IEEE International Conference on Networks*, pages 67–74, Washington, DC, USA, 1999. IEEE Computer Society.

[9] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *IWQoS '99: Proc. of the 7th International Workshop on Quality of Service*, pages 260–262, 1999.

[10] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.*, 8(5):556–567, 2000.

[11] D. Nandita. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. PhD thesis, Standford, 2007.

[12] L. Noirie, E. Dotaro, G. Carofiglio, A. Dupas, P. Pecci, D. Popa, and G. Post. Self-* features for semantic networking. In *FITraMEn '08: Proc. of International Workshop on Traffic Management and Traffic Engineering for the Future Internet*, December 2008.