Brief Announcement: B-Neck – A Distributed and Quiescent Max-min Fair Algorithm *

Alberto Mozo
Dpto. Arquitectura y Tecnología de Computadores
U. Politécnica de Madrid
Madrid, Spain
amozo@eui.upm.es

Jose Luis López-Presa
DIATEL
U. Politécnica de Madrid
Madrid, Spain
jllopez@diatel.upm.es

Antonio Fernández Anta
Institute IMDEA Networks
Madrid, Spain
antonio.fernandez@imdea.org

ABSTRACT

In this brief announcement we propose B-Neck, a max-min fair distributed algorithm that is also quiescent. As far as we know, B-Neck is the first max-min fair distributed algorithm that does not require a continuous injection of control traffic to compute the rates. When changes occur, affected sessions are asynchronously informed, so they can start the process of computing their new rate (i.e., sessions do not need to poll the network for changes). The correctness of B-Neck is formally proved, and extensive simulations are conducted. In them it is shown that B-Neck converges relatively fast and behaves nicely in presence of sessions arriving and departing.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Max-min fairness, quiescence, distributed algorithms

1. INTRODUCTION

The fair distribution of network resources among a set of sessions is a recurring problem. In this problem, each session connects via a single communication path a source node and a destination node in the network, with the objective of maximizing the transmission rate (i.e., throughput) between them. Since the links of the network have limited capacity, the solution of the problem must use some criterion to fairly distribute the network resources among the sessions.

A popular fairness criterion to share the available network capacity among a set of sessions without incurring in link overload is the, so called, *max-min fairness* [3]. The basic idea behind the max-min fairness criterion is to first allocate equal bandwidth to all contending sessions at each link, and if a session can not utilize its bandwidth because of constraints elsewhere in its path, then the residual bandwidth is distributed among the other sessions. Thus, no session is penalized, and a certain minimum quality of service is guaranteed to all sessions. More precisely, max-min fairness takes into account the path of each session and the capacity of each link. Then, each session i is allocated a transmission rate λ_i so that no link is overloaded, and a session can only increase its rate at the expense of a session with the same or smaller rate.

*This research was supported in part by Comunidad de Madrid grant S2009TIC-1692 and Spanish MICINN grant TIN2008–06735-C02-01.

PODC'11, June 6–8, 2011, San Jose, California, USA.
ACM 978-1-4503-0719-2/11/06.

2. RELATED WORK

We are interested in computing the max-min fair rate allocation for single path sessions. The max-min fair rates of the sessions can be efficiently computed in a centralized way with the Water-Filling algorithm [3]. Max-min fairness has usually been chosen as the target fairness criterion implemented by congestion control protocols to allocate the bandwidth of network links among the sessions that cross them. From a taxonomic point of view, centralized and distributed algorithms have been proposed. The latter have typically been implemented as congestion control protocols.

To our knowledge, the proposals of Gallager [7] and Katevenis [11] were the first to apply max-min fairness to share bandwidth among sessions in a packet switched network. Later on, when ATM networks appeared, several distributed algorithms were proposed to calculate virtual circuit max-min fair rates in the Available Bit Rate (ABR) traffic mode [1, 2, 4, 8, 12]. Charny et al. [4] seem to have been the first to analytically prove the correctness of their proposed algorithm. Hou et al. [8] generalized the Charny algorithm to extend the max-min fairness criterion with minimum rate requests and peak rate constraints. A problem of the algorithm in [4] (when pseudo-saturated links appear) was identified and documented by Tsai and Kim [12]. While the distributed algorithms mentioned need per-session state information at the routers of the network, distributed algorithms that only use constant state information in each router have also been proposed [1, 5].

Recent research trends in explicit congestion control protocols (XCP [10], RCP [6], PIQI-RCP [9]) implement efficient congestion controllers in routers, without the need to store and process state information for each session (in the case of RCP, even with a low per packet computational overhead), and guarantee that the max-min fair rate assignments are achieved when controllers are in steady state. These papers analyze the stability of these protocols, but the experi-

mental evaluations are done on simple topologies composed by a small number of links and sessions (far from real scenarios of large networks with transient dynamics).

None of the distributed algorithms mentioned above is quiescent, and so, traffic must be injected continuously in to the network in order to keep the system stable. It is not straightforward to transform any of these algorithms to achieve quiescence since they do not have mechanisms to detect convergence to the max-min fair rate.

3. CONTRIBUTIONS

In this brief announcement we propose B-Neck, a max-min fair distributed algorithm that is also quiescent. As far as we know, B-Neck is the first such algorithm. Instead of requiring a continuous injection of traffic to compute the max-min fair rates, B-Neck uses a limited number of control packets. In addition, each node only requires information of the sessions that traverse it. When changes occur, affected sessions are asynchronously informed, so they can start the process of computing their new rate (i.e., sessions do not need to poll the network for changes). Quiescence is a key design concept of B-Neck, because B-Neck routers are capable to detect by themselves changes in the convergence conditions (from instability to stability and vice-versa) of max-min fair session rates, and to notify them to the affected sessions. Additionally, affected sessions collaborate with routers to propagate atomically these changes to the rest of the routers in their paths. This behavior is not present by design in any of the non-quiescent published algorithms, and so, as mentioned, the transformation of any of these algorithms to a quiescent one is not a trivial problem.

We have formalized the interaction between the (applications that create and use the) sessions and B-Neck, with a set of primitives. Then, primitives to start and end sessions have been defined (namely, $API.Join$ and $API.Leave$). A primitive that B-Neck uses to notify a session of a change in its rate is also defined (namely, $API.Rate$). Finally, the interface allows a session to fix the maximum rate that it requires both at the time it is created (with $API.Join$) and by using a fourth primitive, defined to change the requested maximum rate (namely, $API.Change$).

The properties of B-Neck are formally proved. This proof has two parts. Firstly, we show its *correctness*, i.e., if sessions do not change (for a time period large enough) B-Neck correctly finds the max-min fair rates of all the sessions, and notifies these rates to them. Secondly, we show *quiescence*, i.e., after computing the rates, eventually B-Neck stops injecting traffic into the network. We want to note that, once B-Neck is quiescent, changes in the sessions (new arrivals, departures, or changes in the requested maximum rates) reactivate it, so that, once the changes end, the new appropriate rates are found and notified, and eventually B-Neck becomes quiescent again.

The properties of B-Neck have been tested with extensive simulations. In them, we have used networks of several sizes (with up to hundreds of thousands of nodes), with LAN and WAN characteristics, and with a wide range of session cardinalities (up to hundreds of thousands of sessions). To guarantee the correctness of our implementation of B-Neck, the max-min fair rates obtained have been compared with rates computed with a centralized algorithm (similar to the Water-Filling algorithm [3]). B-Neck has always converged to the right set of max-min fair rates. Our simulations have shown that B-Neck converges very quickly (30% faster than [2] in our simulations), even in the presence of many interacting sessions. We have also stressed the algorithm by, once quiescent, causing large number of simultaneous departures and rate changes. In all cases B-Neck has shown to be robust and efficient, reaching convergence and quiescence again quickly. The control traffic caused in the network by the algorithm is also shown to be limited, and only for highly dynamic systems with many sessions has more than a few packets per session. Finally, we have observed that, during transient behavior, B-Neck assigns temporal rate values to the sessions that are smaller than the max-min fair rates. Hence, it is expected that the network links will not suffer from packet overloading before convergence, due to these conservative temporal rate assignments.

4. REFERENCES

[1] Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: a simple and effective flow control scheme. *Computer Networks*, 32(3):277–305, 2000.

[2] Y. Bartal, M. Farach-Colton, S. Yooseph, and L. Zhang. Fast, fair and frugal bandwidth allocation in atm networks. *Algorithmica*, 33(3):272–286, 2002.

[3] D. Bertsekas and R. G. Gallager. *Data Networks (2nd Edition)*. Prentice Hall, 1992.

[4] A. Charny, D. Clark, and R. Jain. Congestion control with explicit rate indication. In *International Conference on Communications, ICC'95, vol. 3*, pages 1954 – 1963, 1995.

[5] J. A. Cobb and M. G. Gouda. Stabilization of max-min fair networks without per-flow state. In S. S. Kulkarni and A. Schiper, editors, *SSS*, volume 5340 of *Lecture Notes in Computer Science*, pages 156–172. Springer, 2008.

[6] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor sharing flows in the internet. In H. de Meer and N. T. Bhatti, editors, *IWQoS*, volume 3552 of *Lecture Notes in Computer Science*, pages 271–285. Springer, 2005.

[7] E. L. Hahne and R. G. Gallager. Round robin scheduling for fair flow control in data communication networks. In *IEEE International Conference in Communications, ICC'86*, pages 103–107, 1986.

[8] Y. T. Hou, H. H.-Y. Tzeng, and S. S. Panwar. A generalized max-min rate allocation policy and its distributed implementation using abr flow control mechanism. In *INFOCOM*, pages 1366–1375, 1998.

[9] S. Jain and D. Loguinov. Piqi-rcp: Design and analysis of rate-based explicit congestion control. In *Fifteenth IEEE International Workshop on Quality of Service, IWQoS 2007*, pages 10 –20, June 2007.

[10] D. Katabi, M. Handley, and C. E. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM*, pages 89–102. ACM, 2002.

[11] M. Katevenis. Fast switching and fair control of congested flow in broadband networks. *IEEE Journal on Selected Areas in Communications*, SAC-5(8):1315–1326, 1987.

[12] W. K. Tsai and Y. Kim. Re-examining maxmin protocols: A fundamental study on convergence, complexity, variations, and performance. In *INFOCOM*, pages 811–818, 1999.