Fibbing in action: On-demand load-balancing for better video delivery

Olivier Tilmans[‡*], Stefano Vissicchio[‡†], Laurent Vanbever[§], Jennifer Rexford[¶]

[‡] Université catholique de Louvain, [§] ETH Zürich, [¶] Princeton University

[‡] name.surname@uclouvain.be, [§] lvanbever@ethz.ch, [¶] jrex@cs.princeton.edu

ABSTRACT

Video streaming, in conjunction with social networks, have given birth to a new traffic pattern over the Internet: transient, localized traffic surges, known as flash crowds. Traditional traffic-engineering methods can hardly cope with these surges, as they are unpredictable by nature. Consequently, networks either have to be over-provisioned, which is expensive and wastes resources, or risk to periodically incur congestion, which infuriates customers. This demonstration shows how Fibbing [1] can improve network performance and preserve users' quality of experience when accessing video streams, by implementing a fine-grained load-balancing service. This service leverages two unique features of Fibbing: programming per destination load-balancing and implementing uneven splitting ratios.

CCS Concepts

•**Networks** → **Network protocols; Programmable networks; Network management;** *Network architectures;*

Keywords

Fibbing; traffic engineering; link-state routing

1. INTRODUCTION

Interactive applications (e.g., video streaming) impose hard constraints on the performance of networks (e.g., on losses or delay) in order to ensure a good quality of experience for their users. To guarantee good network performance, operators typically rely on traffic engineering (TE). However, traditional TE schemes pre-compute a network configuration for a predictable load (e.g., expressed as traffic matrices), which is hardly effective in the events of flash crowds [2]. For example, a sudden surge of traffic due to content shared over social networks could congest parts of the network, leading to service outages. Operators can thus either vastly over-provision their networks, hindering their profitability, or be at risk of service disruption.

This demonstration illustrates how Fibbing [1] can improve network performance in the events of a flash crowds targeting content delivery services in a network. Such a support is based on two unique abilities of Fibbing, that is, (i) to program multiple paths a on a per-destination basis with very limited control-plane overhead, and (ii) to enforce uneven load-balancing among those paths with no data-plane overhead. This also distinguishes Fibbing from previously-studied approaches, from IGP multi-path to MPLS RSVP-TE [3], and will be presented in Sec. 2. We will then demonstrate in Sec. 3 a Fibbing controller quickly removing the congestion in a small network, by combining these two building blocks, to successfully deliver videos to multiple users. This demo also shows that it is possible and profitable to build applications on top of the Fibbing machinery.

2. FLEXIBLE ECMP WITH FIBBING

Most IP networks run a link-state Interior Gateway Protocol (IGP). IGPs build the routing tables of each router by by computing the shortest-paths on shared weighted graph. Consider the sample network control-plane shown on Fig. 1a, where unspecified link weights are 1. These weights are the results of an IGP-TE optimization minimizing the maximal link load under normal condition. Assume that there is a sudden surge of traffic from two sources towards destinations belonging to the blue prefix. With the chosen set of link weights, the maximal link load in the network increases drastically, potentially overloading some links (see Fig. 1b).

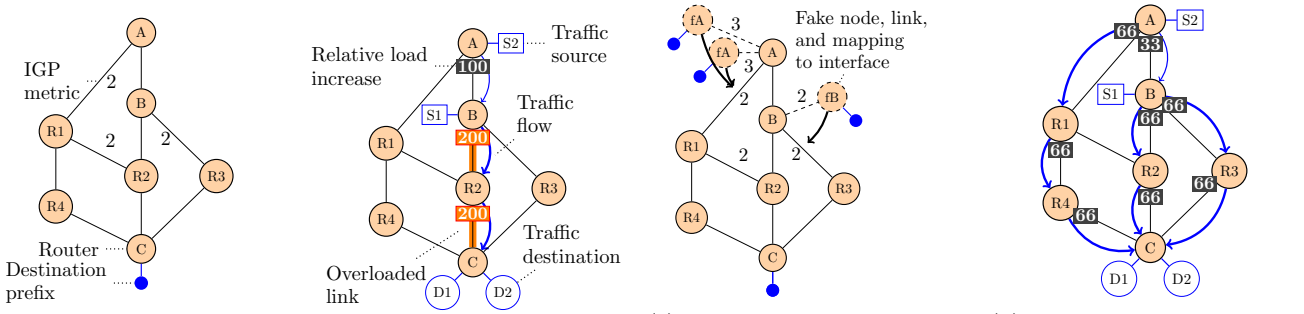
The original TE scheme lacks flexibility to react to such event. Indeed, adapting to the new traffic demands requires to change the link weights on a per-device basis. This process is too slow for a transient event, and

[*]O. Tilmans is supported by a grant from F.R.S.-FNRS FRIA.

[†]S. Vissicchio is a postdoctoral researcher from F.R.S.-FNRS.

SIGCOMM '16, August 22–26, 2016, Florianopolis, Brazil

DOI: http://dx.doi.org/10.1145/2934872.2959084



(a) The IGP shortest paths starting at A and B overlaps along B–$R2$–C.

(b) The data-plane traffic from sources to clients could overload links B–$R2$–C.

(c) Fibbing augments the topology with fake elements to create additional paths towards the blue prefix.

(d) The data-plane traffic is split across all paths, introducing uneven load-balancing at router A.

Figure 1: As Fibbing provides a fine-grained control over ECMP, it can dynamically add new paths in order to cancel the effect of a surge of traffic and reduce the maximal link load.

may impact negatively other traffic flows. Furthermore, as ECMP will load-balance the traffic evenly across all next-hops, reaching the optimal configuration is an intractable problem [4]. Using MPLS and RSVP-TE avoids the shortcomings of the IGP, but introduces overhead on both the control and data planes, by establishing a potentially-high number of tunnels, encapsulating packets, and performing statefull uneven load-balancing.

Fibbing provides better tools to react to flash crowds. By injecting fake nodes and links in the underlying IGP topology, Fibbing achieves full control over the available paths and their cost, on a per destination basis. Moreover, by adding redundant equal-cost paths, Fibbing implements on-demand fractional splitting of traffic. Fibbing can thus theoretically implement the optimal solution to the min-max link utilization problem [5], without pre-provisioning tunnels or changing link weights.

As shown by Fig. 1c, a Fibbing controller can inject a fake node fB attached to B, announcing the blue prefix at cost of 2. This will trick B in computing 2 equal-costs paths, B–$R2$–C and B–$R3$–C. Similarly, 2 fake nodes attached to A cause that router to compute 3 different paths. When installed into the routers' FIB, these fake nodes are then resolved to physical next-hops ($R3$ for fB, and $R1$ twice for fA). Fig.1d shows the resulting data-plane flows and link utilization.

3. DEMO

Setup. We emulate the network in Fig. 1a. The sources are video streaming servers, and the destinations are playback clients. A Fibbing controller, connected to $R3$, monitors link loads using SNMP, and is notified by the servers when they have a new client.

Experiment. Multiple clients ($D1$) request videos to $S1$. As the link loads increase, the Fibbing controller introduces ECMP on B by injecting a fake node for router B as shown on Fig. 1c in order to prevent congestion. Before the first series of videos finishes, a second set of clients ($D2$) requests video playbacks to $S2$.

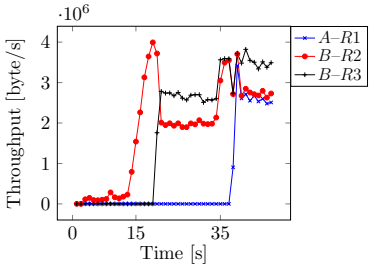


Figure 2: Fibbing decreases the maximal link load while the overall load of the network increases by adding equal-cost paths and controlling splitting ratios.

The controller then introduces uneven load-balancing at A, by injecting two fake nodes on A as in Fig. 1d.

Results. The video playbacks are smooth when the Fibbing controller is in use and stutter when disabled. Fig. 2 plots the throughputs over time over the links A–$R1$, B–$R2$ and B–$R3$, with a single flow from $S1$ to $D1$ (at $t = 0s$), then adding 30 more (at $t = 15s$), then adding 31 flows from $S2$ to $D2$ (at $t = 35s$). As the network load increases, we see that additional paths are used, preventing congestion from happening.

The source code for the Fibbing controller, the scripts to build a virtual machine and sample labs including this demo are available on `www.fibbing.net`.

4. REFERENCES

[1] S. Vissicchio *et al.*, "Central control over distributed routing," in *SIGCOMM*, 2015.

[2] I. Ari *et al.*, "Managing flash crowds on the internet," in *MASCOTS, IEEE*, 2003.

[3] N. Wang *et al.*, "An overview of routing optimization for internet traffic engineering," *Communications Surveys & Tutorials, IEEE*, 2008.

[4] M. Chiesa *et al.*, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," in *INFOCOM*, 2014.

[5] R. Ahuja *et al.*, *Network flows: theory, algorithms, and applications.* Prentice Hall, Inc., 1993.