HFTraC: High-Frequency Traffic Control

Ning Wu
Cornell University
nw276@cornell.edu

Yingjie Bi
Cornell University
yb236@cornell.edu

Nithin Michael
Waltz Networks, Inc.
nithin@waltznetworks.com

Ao Tang
Cornell University
atang@ece.cornell.edu

John Doyle
California Institute of Technology
doyle@caltech.edu

Nikolai Matni
California Institute of Technology
nmatni@caltech.edu

ABSTRACT

We propose high-frequency traffic control (HFTraC), a rate control scheme that coordinates the transmission rates and buffer utilizations in routers network-wide at fast timescale. HFTraC can effectively deal with traffic demand fluctuation by utilizing available buffer space in routers network-wide, and therefore lead to significant performance improvement in terms of tradeoff between bandwidth utilization and queueing delay. We further note that the performance limit of HFTraC is determined by the network architecture used to implement it. We provide trace-driven evaluation of the performance of HFTraC implemented in the proposed architectures that vary from fully centralized to completely decentralized.

KEYWORDS

Network control and management; Communication networks

ACM Reference format:
Ning Wu, Yingjie Bi, Nithin Michael, Ao Tang, John Doyle, and Nikolai Matni. 2017. HFTraC: High-Frequency Traffic Control. In *Proceedings of SIGMETRICS '17, Urbana-Champaign, IL, USA, June 05-09, 2017,* 3 pages.
DOI: http://dx.doi.org/10.1145/3078505.3078557

1 INTRODUCTION

Due to the network dynamics induced by both varying traffic demand and resource supply, responsive approaches of network control that update network configurations at a fast timescale becomes increasingly appealing. The goal of such approaches is to improve resource utilization by reacting to network changes, rather than by over provisioning with respect to its typical behavior. The emergence of Software-Defined Networking (SDN) [5] not only makes it more feasible for network control and management technology to operate at higher frequency, but also enables the flexibility of their architecture design.

In this work, we propose high-frequency traffic control (HFTraC) that dynamically controls, at around RTT timescales, the link service rate from in-network routers based on the information exchange with each other including queue length and transmission rate. HFTraC targets at better utilizing available buffer space and thus achieving higher network utilization and smaller average buffer size than static rate-limiting policies. The feature that HFTraC works by only rate-limiting the routers' interface and does not change queuing management or routing strategy allows its compatibility with any standard AQM schemes [3, 7] and TE approaches [2, 4]. The scheme of information exchanging among the routers has an important effect on the responsiveness which determines the system performance at the time-scale that we consider. Therefore we further provide four different implementation architectures of HFTraC, varying from completely centralized to completely decentralized, and evaluate the associated latency/performance tradeoff.

SIGMETRICS '17, June 05-09, 2017, Urbana-Champaign, IL, USA

ACM ISBN 978-1-4503-5032-7/17/06.
DOI: http://dx.doi.org/10.1145/3078505.3078557

2 DESIGN

We consider a network consisting of a set of switches V and a set of directed links L . We let $\mathcal{L}_v^{in}$ and $\mathcal{L}_v^{out}$ denote the set of incoming and outgoing links respectively for a switch $v \in V$. The network is shared by a set I of source-destination (SD) pairs and the traffic demand is denoted as d_i for SD pair $i \in I$. Let x_l^i be the arrival rate into the egress buffer associated with link l and let f_l^i be the transmission rate on link l due to SD pair i. Switch v splits flows along outgoing links according to split ratios $\alpha_{l,v}^i$ satisfying $\sum_{l \in \mathcal{L}_v^{out}} \alpha_{l,v}^i = 1$ for each SD pair i that utilizes link l. To capture the effect of traffic fluctuations, we model the demand fluctuation $\Delta d_i(t)$ at time t as $\Delta d_i(t) = d_i(t) - d_i^*$, where d_i^* is the average demand over a TE update interval. Similarly, the fluctuation of arrival and transmission rates are given by $\Delta x_l^i(t) = x_l^i(t) - (x_l^i)^*$ and $\Delta f_l^i(t) = f_l^i(t) - (f_l^i)^*$. The fluctuation of aggregated arrival rate and transmission rate on link l are defined as $\Delta x_l(t) = \sum_{i \in I} \Delta x_l^i(t)$ and $\Delta f_l(t) = \sum_{i \in I} \Delta f_l^i(t)$ respectively.

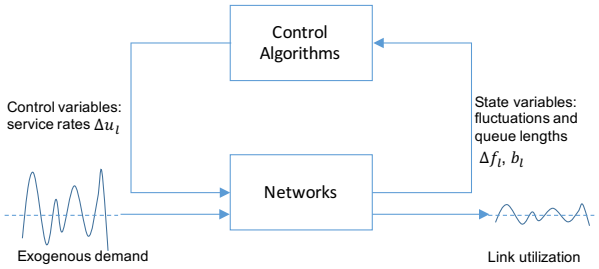


Figure 1: HFTraC Feedback Control System

We construct the feedback control system of HFTraC shown in Fig 1. The control variables in the system are the changes of service rate $\Delta u_l(t)$, and the control action is done by rate limiting

link l to the dynamic service rate $u_l(t) = (f_l)^* + \Delta u_l(t)$. The state variables are the network state information including transmission rate fluctuations $\Delta f_l(t)$ and queue lengths $b_l(t)$. The control algorithm outputs the service rate change $\Delta u_l(t)$ for each link at RTT timescales, taking the measurement of state variables as input. The service rates under the dynamic control will in turn affect the current network state.

We pose the discrete-time optimal control problem as:

$$\min_{\Delta u_l(n)} \quad \lim_{n\to\infty} \sum_l [\mathbb{E}[\Delta f_l(n)]^2 + \lambda_l \mathbb{E}[b_l(n)]^2]$$

$$\text{s.t.} \quad b_l(n+1) = b_l(n) + \tau(\Delta x_l(n) - \Delta u_l(n)),$$

$$\Delta f_l(n+1) = \Delta u_l(n),$$

$$\Delta x_l(n) = \sum_{i\in I} \Delta x_l^i(n),$$

$$\Delta x_l^i(n) = \begin{cases} \Delta d_i(n) & \text{if } l \text{ is edge link,} \\ \alpha_{l,v}^i \sum_{k\in\mathcal{L}_v^{\text{in}}} \beta_k^i \Delta f_k(n-n_k) & \text{otherwise,} \end{cases}$$

$$\Delta u_l(n) = \gamma_l(\mathcal{I}_l(n)) \text{ for all } l \in L.$$

The objective function is a weighted sum (specified by the weights $\lambda_l \geq 0$) of the variance in transmission rate and buffer length. τ is the sampling interval satisfying $\tau n_k = \delta_k$ for some integers n_k. We make the simplifying assumption that $\Delta f_l^i(n) = \Delta f_l(n)\beta_l^i$ where $\beta_l^i := \frac{(f_l^i)^*}{\sum_{k\in I}(f_l^k)^*}$. The service rate is constrained by $\mathcal{I}_l(n)$, which is the set of state information available to the algorithm taking the control loop latency into account, i.e.

$$\mathcal{I}_l(n) := \{(b_k(n-n_{lk}))_{k\in L}, (\Delta f_k(n-n_{lk}))_{k\in L}\},$$

where n_{lk} is taken to be the communication delay imposed on information exchange from link k to link l. The communication delays vary in different architecture, thus driving us to the exploration of architecture design. Here we define four candidate architectures that range from completely centralized to fully decentralized.

The GOD architecture: The Globally Optimal Delay free (GOD) architecture assumes that a logically and physically centralized controller can instantaneously access global network states as well as compute and execute control laws Δu_l for each router. It is clearly not implementable in practice, but it represents a benchmark against which all other architectures should be compared.

The centralized architecture: A logically and physically centralized controller makes control decisions. The control loop latency is determined by $n_{\max} = \max_{l\in L} n_{lk}$, the longest RTT from any router to the centralized controller. It takes $\frac{1}{2}n_{\max}$ for the controller to collect complete network information and another $\frac{1}{2}n_{\max}$ for the router to receive the control decisions.

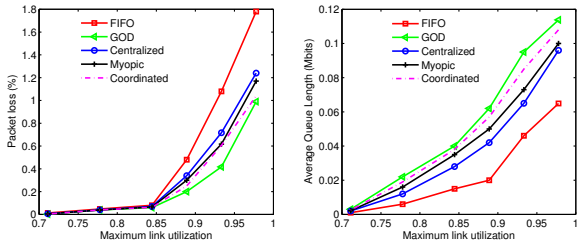
The coordinated architecture: The controller is logically and physically distributed, with each local controller computing Δu_l based on shared network state information. n_{lk} are specified by the delays of collecting network state information at router associated with link l. The local control actions will be taken immediately once being computed by the controller with the knowledge of timely local information and delayed shared information.

The myopic architecture: This is a completely decentralized architecture, in which local controllers compute their control laws Δu_l using local information only, i.e., the delays available to the controller associated with link l satisfy $n_{ll} = 0$, $n_{lk} = \infty$.

3 RESULTS

We here provide our trace-driven evaluation results of HFTraC on a triangle topology with three nodes running Open vSwitch (OVS) [6]. Links between the switches have capacity of 30 Mbps, RTT of 20 ms and buffer limit of 0.2 Mbits. There is one host node connecting to each of the switch. Edge links that connects the switch and host have larger capacity of 42 Mbps and buffer limit of 0.3 Mbits. The real Internet traces are extracted from CAIDA anonymized dataset [1] recorded with nanosecond scale timestamps. We replay the trace data on the topology from a source host to a destination host. Given by the routing solution to some TE method, 75% of the traffic is transmitted on the shorter path with a single hop, and 25% is transmitted on the longer path with two hops. We choose a sampling time of $\tau = 10$ ms – hence, the control laws are updated once every 10 ms.

We evaluate how the total loss rate and average queue length change as maximum link utilization is increased in GOD, centralized, myopic and coordinated schemes and compare the performance with the standard FIFO scheme which is in fact static rate limiting by link capacity. Fig. 2 shows that HFTraCs are able to reduce packet loss by absorbing some of the traffic demand randomness into the buffers – not surprisingly, this in general leads to slightly larger queue length when HFTraC is used. Thus we see that HFTraC, regardless of architecture, effectively reduces the packet loss rate especially when link utilization is over 85%.



(a) Loss rate (b) Average queue length

Figure 2: Trace-driven evaluation of HFTraC.

REFERENCES

[1] *The CAIDA UCSD Anonymized Internet Traces - February 2012.* http://www.caida.org/data/passive/passive_2012_dataset.xml.

[2] Anwar Elwalid, Cheng Jin, Steven Low, and Indra Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of IEEE INFOCOM*, pages 1300–1309, 2001.

[3] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking (ToN)* 1(4):397–413, 1993.

[4] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, and others. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review* 43(4):3–14, 2013.

[5] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[6] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, and others. The Design and Implementation of Open vSwitch. In *Proceedings of NSDI*, pages 117–130, 2015.

[7] Kadangode Ramakrishnan and Sally Floyd. *A proposal to add explicit congestion notification (ECN) to IP.* Technical Report, 1998.