TCPTokens: Introducing Currency into Data Center Congestion Control

Anand Jayarajan
University of British Columbia
anandj@cs.ubc.ca

Michael Przystupa
University of British Columbia
michael.przystupa@gmail.com

Robert Reiss
University of British Columbia
rreiss@cs.ubc.ca

Fabian Ruffy
University of British Columbia
fruffy@cs.ubc.ca

ABSTRACT

Datacenter networks have become a hotbed for research in the recent past. Many works have been published on improving the bisection bandwidth, reducing cost, and improving congestion control as well as latency. As a core contribution of this research, centralized scheduling has entered the stage as dominant strategy to manage flows. Benefitting from a global view and full control over the network, centralized schedulers are able to enforce fine-grained traffic control, frequently achieving near-optimal bandwidth optimization. However, modern congestion control algorithms and schedulers are still fundamentally reactive. Many algorithms are designed to only respond to packet loss or significant increase in latency, not to actively prevent it. In this project, we plan to explore the opportunities of employing a preemptive and tightly controlling central network scheduler. Using tokens as a global tool to enforce traffic limiting and policies, the scheduler is able to proactively control the flow of traffic as well as respond to application requests. "Knowledge" and predictive analysis in networks is a growing trend in research which we intend to leverage in this system. We investigate the potential in such a new form of interactive congestion control and analyze it against to state-of-the-art solutions. Our tool of choice to model our system is Mininet, a rapid prototyping emulator for datacenter networks. We will compare our "TCPToken" system against established TCP-congestion systems such as Hedera and DCTCP. Based on the findings and measurement results, we will reassess the feasibility and prospect of a token-based, predictive congestion control algorithm.

KEYWORDS

TCP, Congestion Control, SDN, Data center

1 INTRODUCTION

Initially, research in network congestion was dominated by the assumption of a decentralised, autonomous network - end-hosts only have control over the amount of traffic they send, and are unaware of the intentions or traffic rates of their peers. Similarly, switches and routers are unaware of global traffic patterns and only forward based on their local notion of optimality.

In line with these assumptions, TCP was designed to optimize traffic globally on a simple fairness principle: nodes react to packet loss and assume that others will behave accordingly. An equilibrium is reached when all end-hosts achieve a traffic rate that, in sum, conforms to the maximum available bandwidth of the most congested link. TCP works well in scenarios where many different distrustful participants compete for limited bandwidth. Still, TCP is a *reactive protocol*; the fact that packet loss and latency increases occur in the network already indicates a problem. Packet loss is primarily due to overflowing queues in forwarding elements, implying that traffic has not been optimally distributed. Ideally, a network should always be "zero-queue", i.e., latency will merely be induced by propagation, and not queuing delay.

Queueing has generally not been a dominating issue in wide-area and enterprise networks, as traffic is sufficiently distributed and diverse, with only few "hot" target hosts. [1, 3] Traffic optimization is a substantial challenge; network operators have no control over the individual network elements nor its participants. Under these conditions, TCP and its extensions can be considered a best-effort solution.

Developments in the past decade have changed the general networking environment. Datacenters have emerged as an exciting new research frontier, posing novel design challenges and opportunities. Driven by minimization of costs and maximization of compute power, data centers must run at maximum utilization to achieve an optimal compute/cost ratio. Inefficient routing can quickly lead to bufferbloat [6] and the eventual collapse of a high-load network, requiring more sophisticated approaches to solve congestion control. On the other hand, operators now have the ability to freely control and adapt their network architecture, leading to highly customized systems and fine-grained optimization. As a result , Software-Defined Networking (SDN) has emerged as a new networking paradigm. Moving away from the principle of distributed communication and routing, SDN introduces the notion of "centralized management". A single controller with global knowledge is able to automatically modify and adapt the forwarding tables of all switches in the network, while notifying end hosts of changes in the network. These two new trends in system design facilitated impactful new innovation opportunities in the space of TCP congestion research. Traffic can now be managed in a *centralised* fashion based on *global knowledge* of the entire topology and traffic patterns. A new line of centralised schedulers has emerged that can achieve close to optimal bandwidth utilization. [1, 3, 7, 8, 11] However, these schedulers are still *reactive* in nature. The central controller responds to changes in the network or requests by applications, which may cost valuable roundtrip latency. Often, short-term flows or bursts are unaccounted for, which causes undesirable packet loss and backpropagating congestion.

A much more desirable solution is a global, centralised arbiter which is able to predict and fairly distribute flows in the network before bursts or congestion occurs. By treating the networkâĂŹs compute and forwarding power as a single finite resource, a controller acts like the OS scheduler distributing CPU time slices to processes. This design approach follows SDNâĂŹs aspiration of introducing operating systems abstractions to the networking domain space.

In this project, we plan to explore the possibilities of a centralised, proactive flow scheduler. We ask ourselves the following research questions:

(1) Is it possible to design a centralised token-based scheduling network?
(2) Is it possible to predict traffic and preemptively schedule flows and token distribution in a datacenter context?
(3) Using this approach, are we able to achieve better performance and utilization than existing solutions?

In the scope of this course, we attempt to answer question 1 and design a simple token-based scheduler in Mininet. If successful, we will benchmark our results and evaluate the level of utilization compared to contemporary scheduling systems.

2 RELATED WORK

Most of the work in congestion control relies on a centralised scheduler which periodically gets updates about the flows in the network and reserves specific paths for long running flows.

MicroTE [3] does predictive OpenFlow routing and scheduling using a central SDN controller. Hedera [1] schedules large so called "elephant" flows by gathering switch statistics. However it pays poor attention to short term flows and network latency. FastPass [11] on the other hand, proposes a zero-queue network and promises a very low latency network. The feasibility of this proposal remains questionable when considering the high cost of implementation.

Recent proposals on decentralised scheduling like ExpressPass [5] have heavily influenced our research. ExpressPass posits a credit based system similar to TCPTokens in order to control the congestion in the network. Effecting flow rerouting using ML techniques in KDN [10] and other works shows promising signs of further improvements in this area.

3 DESIGN / PROPOSED APPROACH

3.1 Simple System

In our initial simple TCPToken design, a centralized controller regulates all node traffic by provisioning end-hosts with tokens. These tokens act as the "currency" of the system. The total amount is fixed to some factor of the aggregate bandwidth available in the network, and the traffic window size of sending end hosts is calculated based on the host's current token availability. If a node or switch is overburdened, it may notify the scheduler, which will adjust the traffic window of responsible sending nodes remotely. Management and control operations including token distribution and updates are priority-queued to guarantee a fast and low-latency response. If an end-host requires more bandwidth, it can also notify the controller, which may comply based on priority and availability. Initially, the controller will compute optimal route configuration based on the topology and link bandwidth using a simple heuristic bin-packing approach. End-hosts will be initialized with a fixed low-to-medium bandwidth guarantee that will be adjusted over time.

3.2 Advanced System

The aforementioned system is intended only as a proof of concept. It is still a synchronous and reactive approach, dependent on notifications and requests by overburdened switches and underprovisioned hosts. Ideally, all management in the TCPToken system should be asynchronous and preemptive.

3.2.1 Paying for traffic. In a complete design, tokens will be used as a transport pass instead of simply acting as traffic shaping parameters. On a per-flow basis, nodes will be able to attach tokens to traffic as a form of payment. Enforcement is performed on the switch level, any flow that is not âĂIJpaidâĂİ by a token will be dropped. Nodes may be handed different types of tokens to serve load-balancing and traffic shaping purposes. Applications operating on traditionally short-lived flows and bursts may sign their traffic with a different token type than long-lived permanent processes. Switches will forward traffic and balance flows accordingly with only minor interference from the central controller.

3.2.2 Predicting traffic. ML-based scheduling is an emerging research field, also known as Knowledge-Defined Networking. Datacenter traffic follows patterns, which can be predicted using statistical methods. Combined with using tokens as a traffic engineering enforcement, a preemptive scheduling strategy in datacenters may be feasible.

We do not intend to implement the advanced system for this project, but are using it as a guideline to motivate our research. Based on the experiences and knowledge gained from implementing the basic system, we will reevaluate the feasibility and practicality of the second concept.

3.2.3 Fault-Tolerance and Scalability. To handle fault-tolerance we plan to integrate a fallback-mechanism to conventional TCP, if the node has not received a TCPToken in a prefixed amount of time. End-hosts will only be marginally affected by a controller or link failure, as they are guaranteed a fixed amount of bandwidth based on their token availability. To avoid overburdening the controller, we envision a distributed hive of schedulers (e.g., Onos) each managing their own region in the final design. [4] In case of failure, a shadow node or neighbouring device may take over computation temporarily, until the main node has recovered. [12]

4 IMPLEMENTATION

We intend to emulate our system in Mininet [9] to observe traffic patterns and infer a suitable token algorithm. Mininet has proven itself to be a viable tool to model new congestion control algorithms [13], and will help us prototype our concept efficiently. We will build a custom SDN controller that interacts with traditional OpenFlow software switches as well as end-hosts. End-hosts will run a custom real-world traffic generation script which adjusts based on information packets sent by the controller. If time permits, we may expand our implementation to MaxiNet, which can emulate

large-scale network stress tests of thousands of nodes on multiple physical hosts.

We initially considered a second implementation alternative in C/C++ based on the FastPass [11] source code. This would provide us with a fully deployable system which we could fork our implementation from. A major advantage of this approach is the ability to test scenarios and traffic algorithms using real software code. However, several concerns made us favour a Mininet emulation instead. Firstly, FastPass relies on DPDK integration, which requires actual hardware interfaces. The central arbiter in the FastPass design would need to run on a dedicated machine, which increases prototyping and development complexity substantially. Secondly, the FastPass code is highly specialized and optimized research code with only little available documentation. Modifying and evolving the source code will require thorough understanding of kernel and networking development, a significant time-sink. For a class project, these may be major initial hurdles, taking away from the research aspect of the design concept. Consequently, we have decided to pursue an approach which allows us to quickly develop an understanding of the problem without being obstructed by engineering work.

5 EVALUATION

To evaluate the effectiveness of our system, we plan to measure against existing centralized as well as decentralized solutions. We will run datacenter traffic on a fat-tree topology of various depth and observe the effect each congestion algorithm will have on the overall bandwidth utilization and latency. The centralized design will be based on Hedera [1], a common and influential datacenter scheduler. The decentralized congestion control mechanism will be DCTCP [2], a state-of-the-art TCP congestion algorithm. As baseline we will also compare to the default TCP congestion algorithm to estimate how our scheduler fares against the default case. As our initial simple design is very constrained in its ability to route and control traffic we expect to achieve less optimality against DCTCP and Hedera, but gain a substantial advantage over common TCP.

6 TIMELINE

- October 15th: Finalize draft and related work / background section.
- October 22nd: Iterate over literature and thoroughly analyze related work. Set up an initial prototyping environment in Mininet on which each team member can work on.
- October 30th: Finalize the initial simple design and explore naive congestion modelling in Mininet. Integrate existing Mininet emulations of Hedera and DCTCP.
- November 12th: Develop initial naive TCPTokens prototype involving controller and end-host design.
- November 19th: Develop intelligent centralised congestion control heuristic.
- November 26th: Run evaluations and microbenchmarks.
- December 3rd: Iterate and improve the scheduler, explore topology and bandwidth constraints. Test at scale.
- December 10th: Write report and prepare presentation.

REFERENCES

[1] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks.. In *NSDI*, Vol. 10. 19–19.

[2] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center tcp (dctcp). In *ACM SIGCOMM computer communication review*, Vol. 40. ACM, 63–74.

[3] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. 2011. MicroTE: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 8.

[4] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. 2014. ONOS: towards an open, distributed SDN OS. In *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 1–6.

[5] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 239–252.

[6] Jim Gettys and Kathleen Nichols. 2011. Bufferbloat: Dark buffers in the internet. *Queue* 9, 11 (2011), 40.

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

[8] Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. 2014. Dynamic scheduling of network updates. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 539–550.

[9] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 19.

[10] Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés-Mulero, David Meyer, Sharon Barkai, Mike J Hibbett, et al. 2017. Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review* 47, 3 (2017), 2–10.

[11] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2015. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Computer Communication Review* 44, 4 (2015), 307–318.

[12] K. Phemius, M. Bouet, and J. Leguay. 2014. DISCO: Distributed multi-domain SDN controllers. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. 1–4. https://doi.org/10.1109/NOMS.2014.6838330

[13] Lisa Yan and Nick McKeown. 2017. Learning Networking by Reproducing Research Results. *ACM SIGCOMM Computer Communication Review* 47, 2 (2017), 19–26.