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

KEYWORDS

TCP, Congestion Control, SDN, Data center

1 INTRODUCTION

Initially, research in network congestion was dominated by the assuming 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 âĂIJzero-queueâĂİ, 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 âĂIJhotâĂİ target hosts. [1, 2] 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 [3] 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 highly customized systems and fine-grained optimization. As a result Software-Defined Networking (SDN) emerged as a new networking paradigm. Moving away from the principle of distributed communication and routing, SDN introduces the notion of âĂIJcentralized managementâĂİ. A single controller with global knowledge is able to automatically modify and adapt the forwarding tables of all switches in the network, as well as notify end hosts of changes in the network. These two new trends in systems 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, 2, 4–6] 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 could act like the OS scheduler distributing CPU time slices to processes. This design approach also 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 we succeed, we will benchmark our results and evaluate the level of utilization compared to contemporary scheduling systems.

2 RELATED WORK

3 DESIGN

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 âĂIJcurrencyâĂİ of the system. The total amount is fixed to the aggregate bandwidth available in the network and the traffic window size of sending end hosts is calculated based on the 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. Any management or control operations such as token distribution and updates are priority-queued to guarantee a fast and low-latency response. In the case of an end-host requiring 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, which will be adjusted over time.

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 TCP-Token system should be asynchronous and preemptive. 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.

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.

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 each managing their own region in the final design. Add ONOS CITATION In case of failure, a shadow node or neighbouring device may take over computation temporarily, until the main node has recovered. CITATION

4 IMPLEMENTATION

We intend to emulate our system in Mininet 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 CITATION, 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 that adjust based on information packets send 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 CITATION 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. First, 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

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] 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.

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

[4] 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.

[5] 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.

[6] 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.