Omniscient: Towards realizing near real-time data center network traffic maps

Diana Andreea Popescu
University of Cambridge
diana.popescu@cl.cam.ac.uk

Andrew W. Moore
University of Cambridge
andrew.moore@cl.cam.ac.uk

ABSTRACT

In order to make measurement-based placement, an optimiser must make informed decisions. Currently, it is difficult to assign routes or assign resource commitments to network paths in data centers, as applications do not declare what is carried within each flow. We propose to provide insight into the traffic that traverses each network link, realizing a near real-time map of a data center's network traffic. We present Omniscient, a system that aims to increase the visibility into the data center network traffic by computing link utilization broken down by application instance using OpenFlow stats in an SDN-enabled data center. The goal of the system is to inform application instance placement and redundant network path assignment in order to improve application performance and resource utilization.

CCS Concepts

•**Networks** → **Network monitoring;** *Data center networks;*

1. INTRODUCTION

Data centers have become an essential part of today's network communications. A typical data center runs a huge number of application instances (jobs) on tens of thousands of hosts. Many of these applications are user facing (web search for example) and thus their performance is critical. Academia and industry alike are seeking to improve the data center network fabrics and software to ensure that applications run smoothly. To this end, we need to have readily available data on network traffic patterns in a data center. Knowing the path taken by the flows of a certain application instance could serve in better placement of the application instances or better route assignments. Besides informing future network designs, we would understand better the application performance.

Measurement systems in data centers have recently focused on network debugging [12, 13, 14]. Previously, a number of systems focused on identifying large flows [2, 6], computing the traffic matrix [4] or measuring link utilization [9, 11], in order to use this information to reroute flows.

We propose to realize a near real-time traffic map for a data center, where the utilization on each link is computed, broken down by the application instance that produces that traffic, or at a coarser granularity by the type of application. Our aim is to use the collected data to improve application performance and resource utilization. We introduce Omniscient, a system that combines three simple concepts: tagging packets with a unique identifier for each application instance, OpenFlow flow rule matching on the tagged packets and statistics collection through OpenFlow statistics requests. While NetFlow [5] and sFlow [8] can provide sampled packets that can be processed to provide information about the link utilization, our approach in Omniscient has two advantages over sampling. Firstly, it eliminates the bias inherent to sampling. As it is known, sampling biases against small flows, but these form the majority of the traffic in data centers [3, 10]. Secondly, it offers a flexible way of choosing for which application instance we want to collect this data by using the tagging mechanism.

If we consider web search as an application, in order to answer a web search request, data needs to be retrieved from a large number of servers in parallel and later aggregated into a response at the requesting server. By using the same tag for all the flows determined by the web search request, we can conveniently identify the flows originating from the same application instance. Furthermore, by knowing the link utilization determined by a certain application instance, we can provision accordingly the needed bandwidth for future requests and load balance the requests taking into account the current capacity of the network links. Additionally, we think of several other ways in which our proposed

CoNEXT Student Workshop'15 December 01 2015, Heidelberg, Germany

© 2015 ACM. ISBN 978-1-4503-4066-3/15/12...$15.00

DOI: http://dx.doi.org/10.1145/2842665.2843558

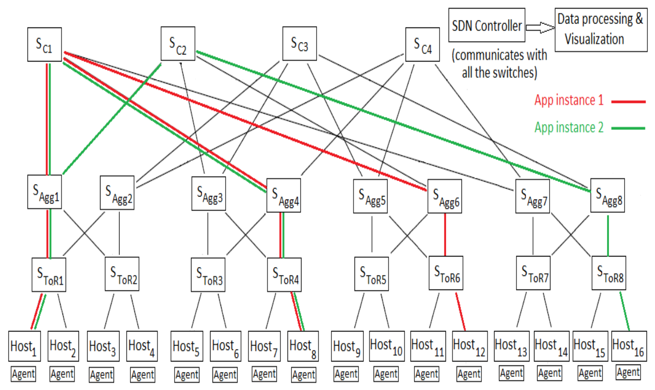

Figure 1: Omniscient running in a data center measuring the link utilization determined by the flows of two application instances.

system could help in solving other issues that arise in data centers. In order to detect intra-datacenter distributed denial-of-service (DDoS) attacks we need visibility of network resources and network-resource measurements. Our system stores past traffic patterns which can be consulted in case there is a deviation from normal traffic patterns for certain application instances or that can be used in capacity planning.

2. PROPOSED DESIGN

Our proposed design leverages the software-defined networking (SDN) paradigm. SDN is a paradigm that separates the control plane from the data forwarding plane, enabling the centralization of network control and offering the possibility of programming the network. The control plane is represented by a controller and the data plane consists of networking devices, like switches and routers. This separation is made possible by a programming interface, which allows the controller to communicate with the forwarding devices, for example installing forwarding rules at switches. OpenFlow [7] is the most popular such communication protocol. For our approach we assume a data center whose switches support the OpenFlow protocol. In an OpenFlow network, the controller can collect statistics about the flows (duration, number of packets, number of bytes) by polling the switches using flow statistics requests. These statistics can be either per flow values or aggregates across multiple flows that match a rule. We combine this OpenFlow feature with tagging packets using a unique identifier for each application instance. In this way, the polled statistics per flow will represent link loads caused by a certain application instance. As OpenFlow switches can support multiple flow tables, where each flow is matched successively against multiple flow tables, the tagging rules can be kept in a separate table, and as such it will not interfere with the other rules used in forwarding.

In what follows, we describe the design of Omniscient in detail. An end-host runs an *agent* that tags the packets originating from an application instance with a unique *identifier*. The identifier must be known by the network controller in order to be able to construct the rule that will match on the packets having this identifier. The controller can generate and propose the identifier to the application instance. The controller will also install the rules at each switch or only at selected switches in the data center. The identifier can be embedded into fields in the packet header or using VLAN tags [12]. In the case of fields, such examples are the ToS byte, or the IP ID field, or the 20-bit IPv6 flow-label field [13]. If we use n bits for encoding, we can encode 2^n application instances. Unlike previous work [12, 13], the identifiers are not used for path tracing, but for identifying the flows originating from a certain application instance. Given the number of application instances running in a data center, we may not be interested in such a fine granularity as tracking the load counters for each application instance. In this case, multiple application instances can have the same identifier, or we can perform aggregation on the load link counters when processing the polled statistics after collection, or install a rule that aggregates multiple identifiers. In this way, we can focus only on the flows of the application instances that we are interested in, resulting in a reduced number of entries in the flow table.

Statistics collection from switches is done through polling at a defined time interval. The overhead incurred on the switches when polling stats must be taken into account when choosing the polling interval, but also the interval should be small enough to provide timely information within a few seconds. Our aim is to construct a network traffic timeline, thus the arrivals of the polling requests at the switches need to be synchronized at the beginning of a new polling interval in order to obtain a coherent view of the traffic. The design also comprises data collection and storage systems and a visualization tool of the data center map annotated with collected information. The data can be stored using a tool such as RRDtool [1].

Challenges in implementing our design are represented by switch hardware limitations (limited space for rules), number of unique identifiers that can be embedded in the packet header field, overhead caused by polling and synchronization of polling requests.

3. CONCLUSION AND FUTURE WORK

We have motivated the need for increased visibility into a data center's network traffic. We have proposed Omniscient, a system whose goal is to collect network traffic patterns data to be used to improve application performance and resource utilization in data centers. We are currently working on implementing and evaluating our design.

Acknowledgments. Diana Andreea Popescu is supported by the EU FP7 ITN METRICS (607728) project.

4. REFERENCES

[1] RRD tool. http://www.rrdtool.org/. Accessed 21-September-2015.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 19–19, Berkeley, CA, USA, 2010. USENIX Association.

[3] T. Benson, A. Akella, and D. A. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280, New York, NY, USA, 2010. ACM.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang. MicroTE: Fine Grained Traffic Engineering for Data Centers. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 8:1–8:12, New York, NY, USA, 2011. ACM.

[5] B. Claise. Cisco systems NetFlow services export version 9, 2004. RFC 3954.

[6] A. R. Curtis, W. Kim, and P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *INFOCOM 2011*, pages 1629–1637, 2011.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[8] P. S. Phaal P. and M. N. Inmon corporation's sFlow: A method for monitoring traffic in switched and routed networks, 2001. RFC 3176.

[9] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca. Planck: Millisecond-scale Monitoring and Control for Commodity Networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 407–418, New York, NY, USA, 2014. ACM.

[10] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 123–137, New York, NY, USA, 2015. ACM.

[11] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. B. Carter. OpenSample: A Low-Latency, Sampling-Based Measurement Platform for Commodity SDN. In *ICDCS*, pages 228–237. IEEE Computer Society, 2014.

[12] P. Tammana, R. Agarwal, and M. Lee. CherryPick: Tracing Packet Trajectory in Software-defined Datacenter Networks. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, SOSR '15, pages 23:1–23:7, New York, NY, USA, 2015. ACM.

[13] H. Zhang, C. Lumezanu, J. Rhee, N. Arora, Q. Xu, and G. Jiang. Enabling Layer 2 Pathlet Tracing Through Context Encoding in Software-defined Networking. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 169–174, New York, NY, USA, 2014. ACM.

[14] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng. Packet-Level Telemetry in Large Datacenter Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 479–491, New York, NY, USA, 2015. ACM.