Overcoming the Memory Limits of Network Devices in SDN-enabled Data Centers

Antonio Marsico, Roberto Doriguzzi-Corin and Domenico Siracusa
CREATE-NET - Fondazione Bruno Kessler, Trento, Italy
Email: {amarsico, rdoriguzzi, dsiracusa}@fbk.eu

***Abstract*—In extremely connected and dynamic environments, such as data centers, SDN network devices can be exploited to simplify the management of network provisioning. However, they leverage on TCAMs to implement the flow tables, i.e., on size-limited memories that can be quickly filled up when fine-grained traffic control is required, eventually preventing the installation of new forwarding rules. In this work, we demonstrate how this issue can be mitigated by means of a novel flow rule *swapping* mechanism. Specifically, we first show the negative effects of a full TCAM on a video streaming service provided by an SDN-enabled data center. Then, we show that our swapping mechanism helps in overcoming the inability to properly access a media content available in the data center, by temporarily moving the least matched flow rules from the TCAM to a larger memory outside the SDN device.**

I. Introduction

Many data centers rely on distributed architectures to improve the availability and the reliability of the services they offer to the customers. This architecture is fully exploited by cloud-based applications, which are composed of many instances spread in different locations of a data center. For instance, cloud-based applications for on-demand video streaming are typically composed of a User Interface (UI), such as a web page, and a collection of video contents, possibly hosted in different hardware or software appliances. Such applications may require many *intra-datacenter* connections to maintain the communication between the different application instances, e.g., for data synchronization.

In SDN-enabled networks, the control logic is moved to an external controller that talks to the datapath over the network itself. The controller decides how the packets are forwarded by installing/manipulating the forwarding rules in the *flow tables* of the switches via control interfaces. *Flow tables* are usually saved in a high performance memory called Ternary Content Addressable Memory (TCAM). Since such memories are very expensive and power hungry, vendors tend to install TCAMs with very limited capacity which can quickly get full, especially in environments like SDN-based data centers.

When a network device runs out of memory, it starts refusing the installation of new forwarding rules. Eventually, the buffer of the network device, which holds the packets waiting for forwarding instructions, becomes full and starts dropping the buffered packets. This will lead to a degradation of the user's quality of the experience in terms of low throughput (see also [1]) and high delays when accessing the online service provided by the data center, such as on-demand video streaming.

To mitigate the effects of limited memory of network devices, we have proposed a novel flow rule memory *swapping* mechanism [1]. The memory swapping is a function of a more generic component for SDN controllers called Memory Management System (MMS).

The MMS [2] aims at optimizing the TCAM memory utilization by providing two different functionalities: (i) the *memory deallocation* and (ii) the *memory swapping*. The *memory deallocation* automatically deletes the flow entries installed in the TCAM by Software-Defined Networking (SDN) applications that are no longer running/active. The *memory swapping* mitigates network performance degradations caused by the network devices operating in full memory condition, by temporarily moving the least frequently matched flow entries to a slower –but larger– memory. This SDN component is currently developed for the ONOS controller [3] and is available for testing as open an source project [4].

In our demonstration we present the *memory swapping* and we show how it improves the performance of a data center in case of TCAM memory overflows in SDN-enabled switches. The proposed demonstration shows how a video streaming service, whose contents are distributed on several servers in the data center, can easily saturate the memory of a Top of Rack (ToR) switch, preventing the customers to access the media contents. We show how the MMS solves this issues.

II. Demonstration

To show the effectiveness of our *memory swapping* mechanism, we propose a network scenario where a SDN-enabled data center hosts a distributed on-demand media streaming application. We demonstrate how an insufficient capacity of the TCAM memory of SDN devices can affect the availability of the media content for the customers. In particular, we show how a full TCAM memory prevents part of the customers to access the media content and, then, we demonstrate how the problem can be solved by enabling the MMS in the SDN controller to optimize the usage of the TCAM memory.
The MMS moves the least used forwarding rules from the TCAM to an external databased maintained by the MMS itself. The newly available TCAM memory space can be then used by the controller to handle the customers' requests. This process is called *memory swap out* and, in this specific demo scenario, involves the rules pro-actively installed by the controller to allow the intra-datacenter communication, as such rules are rarely matched. We also show that such a

978-3-901882-89-0 @2017 IFIP

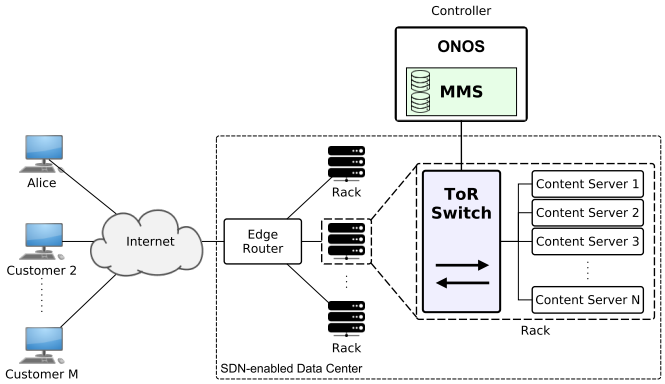

Fig. 1. Demonstration Scenario.

server-to-server connectivity is not broken. Instead, the MMS automatically re-installs the swapped-out rules into the TCAM memory of the switches when they are required (e.g., during the synchronization of the media content between the servers).

A. Demo setup

As represented in Fig. 1, we demonstrate the relevance of the memory swapping by implementing a SDN-enabled data center where the ToR switches are controlled by an SDN controller. Specifically, the right-hand side of the figure illustrates the internal architecture of one of the data center racks.

The network topology, emulated with Mininet [5], includes a ToR switch controlled by the ONOS controller, a pre-configured edge router that connects M customers to the data center, and N content servers which host the media content. The values of M and N determine the occupancy of the ToR memory: (i) $N \times (N-1)$ flow rules with infinite timeout are pro-actively installed by the controller to implement a peer-to-peer synchronization system between the content servers (i.e., full connectivity). (ii) $2 \times M$ flow rules are needed for the customers' traffic (bi-directional communication), thus, the controller reactively installs 2 new rules every time a new customer accesses the data center. Such rules are installed with finite *idle timeout*, so that they are automatically evicted from the ToR switch memory once the customer disconnects. We also customize the size of the emulated TCAM of the ToR switch by setting the flow table size of the Open vSwitch [6] instance created in Mininet.

During the demonstration, we will vary the values of *M* and *N* and the memory size of the emulated ToR switch.

B. Demo workflow

The demonstration is divided in four different steps. As mentioned above, each step can be repeated with different flow table sizes and by varying the number of customers and content servers to demonstrate the effectiveness of the memory swapping mechanism in different conditions.

We start with a flow table of size $2 \times (M-1) + N \times (N-1)$ flow rules (i.e., we intentionally limit the memory of the ToR switch so that it can serve up to $M-1$ customers, beyond the N content servers of the rack), then we repeatedly change the ratio between M, N and the memory size:

1. M-1 customers are streaming video contents from the data center. In this situation, the flow table of the ToR switch is full, and when Alice tries to access the streaming service, her connection is refused as the controller cannot install any new forwarding rule. In this situation, Alice's connection is delayed until one of the other customers disconnects.
2. We enable the MMS in the ONOS controller. The MMS detects the memory full condition and activates the *swapping mechanism*. The least used rules are *swapped out* from the ToR switch to a database hosted in the controller and managed by the MMS. During our demo, we will show that the swapped-out rules are the ones that implement the connectivity among the content servers, as they are rarely used compared to the ones matching the customers' traffic.
3. We start a synchronization process between *Content Server 1* and *Content Server 2* to show how the MMS automatically restores the forwarding rules that have been previously swapped out to allow Alice to access the media contents available in the data center. In this case, the MMS only restores the rules needed for bi-directional connectivity between Servers 1 and 2, while the others remain in the external database.
4. Finally, we vary the size of the flow table and the values of *N* and *M*. When we repeat steps 1-3, we will see the relevance of the MMS increasing when the ratio between the number of connected hosts (customers and servers) and the flow table size increases.

III. Conclusions

This work exhibits the properties of the *memory swapping* mechanism of the MMS, a component for SDN controllers that optimizes the usage of network devices' memory. The experimental setup shows how the *swapping* mechanism can prevent delays for the users when accessing an online video service, hosted in a data center adopting SDN switches with inadequate memory capacity.

Acknowledgments

This work has been partially sponsored by the EU FP7 project NetIDE, grant agreement 619543.

References

[1] A. Marsico, R. Doriguzzi-Corin, and D. Siracusa, "An Effective Swapping Mechanism to Overcome the Memory Limitation of SDN Devices," *to appear in IEEE/IFIP IM 2017*.

[2] R. Doriguzzi-Corin, D. Siracusa, E. Salvadori, and A. Schwabe, "Empowering Network Operating Systems with Memory Management Techniques," in *IEEE/IFIP NOMS*, 2016.

[3] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *ACM HotSDN*, 2014.

[4] "MMS source code." [Online]. Available: https://github.com/fp7-netide/Tools/tree/master/memory-management-system

[5] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible Network Experiments Using Container-based Emulation," in *ACM CoNEXT*, 2012.

[6] The Open vSwitch Project. [Online]. Available: http://openvswitch.org/