Which Flows Are Hiding Behind My Wildcard Rule? Adding Packet Sampling to OpenFlow

Philip Wette
University of Paderborn
Warburger Straße 100
33098 Paderborn, Germany
philip.wette@uni-paderborn.de

Holger Karl
University of Paderborn
Warburger Straße 100
33098 Paderborn, Germany
holger.karl@uni-paderborn.de

ABSTRACT
In OpenFlow [1], multiple switches share the same control plane which is centralized at what is called the OpenFlow controller. A switch only consists of a forwarding plane. Rules for forwarding individual packets (called flow entries in OpenFlow) are pushed from the controller to the switches.

In a network with a high arrival rate of new flows, such as in a data center, the control traffic between the switch and controller can become very high. As a consequence, routing of new flows will be slow. One way to reduce control traffic is to use wildcarded flow entries. Wildcard flow entries can be used to create default routes in the network. However, since switches do not keep track of flows covered by a wildcard flow entry, the controller no longer has knowledge about individual flows. To find out about these individual flows we propose an extension to the current OpenFlow standard to enable packet sampling of wildcard flow entries.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: [Network communications, Packet-switching networks];
C.2.4 [Distributed Systems]: [Network operating systems]

General Terms
Software-Defined Networks, Packet Sampling

Keywords
OpenFlow

1. INTRODUCTION
OpenFlow [1] is a protocol for communication between the forwarding plane of switches and a (logically) centralized control plane in a Software-Defined Network (SDN). In the OpenFlow model, switches only consist of a forwarding plane that is equipped with what is called a flow table. A flow table is a collection of flow entries that identify traffic flows and determine how each packet of a flow is processed. There are 9 packet header fields to match on, where matching can either be done exact (where each single field has to match) or wildcarded (where particular header fields can be omitted). Wildcarded flow entries thus group several individual flows together and treat them in the same way.

Whenever a switch does not have an appropriate flow entry for an incoming packet, this packet is encapsulated in a PACKET_IN message and sent to the controller. The controller then computes an appropriate route for the flow and pushes the corresponding flow entry to the switch. This way, all subsequent packets of that flow are handled by the newly installed rule. For a network with a high arrival rate of new flows, the amount of PACKET_IN messages is very high. However, even current high-end ToR switches like the IBM G8264 are only capable of creating up to 200 PACKET_INs per second [2], which massively restricts its usage in such networks. To cope with these limitations, wildcard flow entries have to be installed in the switch. Thus, default routes for different groups of flows are defined in a proactive manner. But due to grouping, the fine-grained control of the network is lost. The only way to see which single flows are hiding behind a wildcarded flow is to delete the wildcard flow entry from the switch. Then, each subsequent packet that was previously covered by the wildcard flow entry creates a PACKET_IN that is sent to the controller. This is clearly a very inefficient way to look behind a wildcard flow entry.

This paper proposes to add a packet sampling mechanism to the OpenFlow standard to efficiently unveil which individual flows are hiding behind a wildcard flow entry. Since we use packet sampling, the gained results are only approximations. Unlike in conventional packet sampling techniques (like sFlow), samples are not taken from all incoming packets but only from the wildcarded flow entries that are under suspicion. This is more economical: for a given level of accuracy, less samples are required.

2. OPENFLOW PACKET SAMPLING
2.1 Protocol Extension
To add sampling support to OpenFlow, we propose the following extensions to the standard: To invoke the sampling process on a switch for a specific flow entry we extend the ofp_stats_type by the new type OFPST_SAMPLING. The body of such a message is defined in Figure 1. On reception of this message, a switch marks every matching wildcard entry with the requested sampling probability and duration. In the message, sampling_period specifies the average sampling
Thus, the switches have to be populated with wildcard default routes in the network. In case these default routes get to the controller each time a new flow arrives, it is not preferable to install flows reactively (by sending a new flow entry). Instead, it is more efficient to install flows proactively (by sending a wildcard flow entry).

2.2 Sample Application 1: Default Routing

When operating latency-critical networks with OpenFlow, it is not preferable to install flows reactively (by sending a PACKET_IN to the controller each time a new flow arrives). Thus, the switches have to be populated with wildcard entries to speed up forwarding. These wildcard entries create default routes in the network. In case these default routes get congested, it is preferable to reroute some of the flows to less utilized links. This, however, would require knowledge about the flows which are hiding behind the wildcard flow entry. With our extension, the wildcard flow entry can be sampled and a new route for the extracted flows can be installed.

2.3 Sample Application 2: Garbage Collection

Flow tables are stored on ternary content-addressable memory (TCAM), which is expensive and therefore a limited resource in OpenFlow switches. As a consequence, the amount of flow entries that can be stored on a single switch is very limited. After the limit has been reached, it is not possible for the switch to process any more additional flows. In such a situation it is favorable to group multiple flow entries and represent these by using a wildcard flow entry.

To match a packet on wildcard flow entries, in most practical implementations the corresponding packet headers have to be compared to each installed wildcard flow entry of the switch. Thus, with increasing wildcard flow entries at the switch, the latency increases, too. To counteract this, wildcard flow entries covering only a small number of individual flow entries should be removed from the switch and re-installed as individual flow entries if possible. This, however, requires a function to estimate the number of flows before deletion. To accomplish this, our packet sampling extension can be used.

3. CONCLUSION

With our OpenFlow packet sampling extension it is possible to efficiently find out which individual flows are hiding behind a wildcard flow entry. This is a powerful tool when building OpenFlow applications that reside on wildcard flow entries, but requires detailed knowledge of the traffic information after a wildcard flow entry is installed.

4. ACKNOWLEDGMENTS

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre “On-The-Fly Computing” (SFB 901).

5. REFERENCES
