FlowInsight: Decoupling Visibility from Operability in SDN Data Plane

Yuliang Li, Guang Yao, and Jun Bi
Institute for Network Sciences and Cyberspace, Tsinghua University
Department of Computer Science, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology (TNList)
liyuliang001@gmail.com, yaoguang@cernet.edu.cn, junbi@tsinghua.edu.cn

1. INTRODUCTION

In most designs of SDN scenarios, there is only one flow table in each switch, e.g. OpenFlow 1.0. Even with the adoption of multiple flow tables in later versions of OpenFlow, this feature is still not widely used.

In the current design of OpenFlow, a controller has two ways to get the view of a flow: 1) the controller passively learns a flow if the flow is mismatched in the flow table; 2) the controller proactively pulls the counters of each rule. If only one flow table is used, whether the controller can see a flow is wholly determined by the flow table, i.e. the visibility is coupled to the operability. However, the coupling has the following problems:

1. **Invisible flows**. The first packet of the flows may be required by some applications to learn the flows, but gets lost due to the limitation of the design. For example (Figure 1(a)), a wildcard rule installed by a forwarding application will mask all the matched micro-flows which should be visible to a firewall application. To ensure every micro-flow can be seen by the controller, Frenetic [2] installs an exact-match rule after each micro-flow's arrival instead of installing wildcard rules in advance, while introducing significant latency in forwarding.

2. **Unnecessary visibility**. The visibility of a part of flows may be unnecessary to any of the applications, but it is provided with extra resource cost or network performance degradation. For example (Figure 1(b)), whenever a load-balance application queries elephant flows from the flow table, all the mice flows are also unnecessarily got[1].

3. **Rule explosion**. Applications may query the counters of the flows, and the queries should be composed with the forwarding policy. However, the composition may cause rule explosion. For example (Figure 1(c)), in a single switch network where the forwarding behavior are only depend on the dstIP, the monitor wants to get the statistics from each srcIP. Separately, The number of rules needed by the forwarding APP and the monitor APP is respective O(n) and O(m). However, when they are composed into one flow table, the number becomes O(nm).

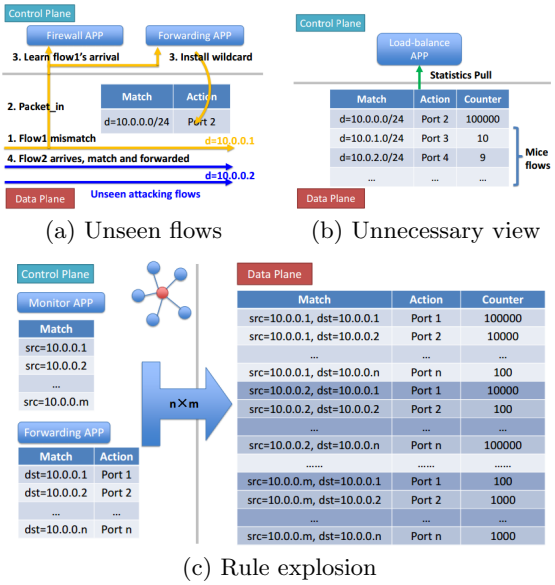

Figure 1: The problems introduced by the coupling between the visibility and the operability

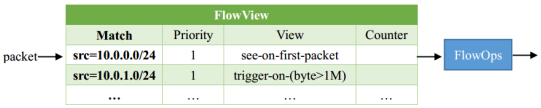

Figure 2: The architecture of FlowInsight

These problems originate from the coupling between the visibility and operability. In this demo, we propose FlowInsight, which separates the provision of visibility and operability. Moreover, we use a 2-stage pipeline of flow tables in OpenFlow 1.3 to implement a prototype, which can naturally solve the rule explosion problem.

2. FLOWINSIGHT

2.1 Architecture

In the data plane, there are two pipelined tables in FlowInsight: **FlowOps** and **FlowView**. Packets are processed by FlowView and FlowOps sequentially (Figure 2).

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631468 .

FlowOps determines the operations on each flow. It works the same as traditional flow table, i.e. matches against packets and applies forwarding to matching packets.

FlowView defines the flows that should be visibile to the controller. Basically, each rule in FlowView has a match field, a priority, a view field and counters. In the view field, one can specify what kinds of view of the match flow should be provided to the controller. There are two kinds of views:

1. **see-on-first-packet**. The first packet of each micro-flow which matches a rule with see-on-first-packet should be copied and sent to the controller. For example, in Figure 2, if two flows with respective src=10.0.0.1 and src=10.0.0.2 arrives, both their first packets should be copied and sent to the controller.

2. **trigger-on-condition**. As introduced by DevoFlow [1], the trigger-on-condition is specified with a threshold, and when the counter meets the threshold, the switch should notify the controller. However, in DevoFlow, the triggers are on the forwarding rules, so functionalities like triggers on the aggregate of several flows cannot be achieved, which can be easily achieved in FlowInsight with FlowView.

Additionally, there is another functionality called **query-by-condition**. Applications can query the FlowView with conditions. For example, applications can query flows whose total bytes reach 1M, or it can query the TCP flows whose destination port is 80.

2.2 Workflow

When a packet arrives, it is first matched against the rules in the FlowView. The rule with the highest priority updates its counter, and performs the functionalities specified in the view field. Then the packet is matched against the rules in the FlowOps, and is forwarded correspondingly.

The controller can send the query-by-condition to the switches proactively, and the switches should send all flows that meet the condition to the controller upon receiving the query.

2.3 How FlowInsight solves the problems

The see-on-first-packet directly solves the invisible flows problem. Because the forwarding action can be installed in FlowOps in advance without interfering the visibility, this will not introduce latency in the forwarding. The trigger-on-condition and query-by-condition together can solve the unnecessary visibility problem associated with flow size. Finally, since the FlowView is independent from FlowOps, the rule explosion problem is naturally solved, as shown in the next section.

3. IMPLEMENTATION AND DEMONSTRATION

We use the feature of multi-table pipeline in OpenFlow 1.3 to implement an experiment. Since see-on-first-packet and query-by-condition are not directly supported by OpenFlow 1.3, we leave them to our future work, only having a separate table with counters. Our experiment shows that it can significantly reduce the number of rules compared to the case where only one table is used, so it can solve the rule explosion problem.

In this demo, We perform a case study, where there is only one switch connecting 48 hosts. There are two applications. The first is a learning switch, which learns the <MAC, port> mapping and forwards packets based on the destination MAC address. The second applications is a monitor, which monitors the traffic from each source MAC address every time unit.

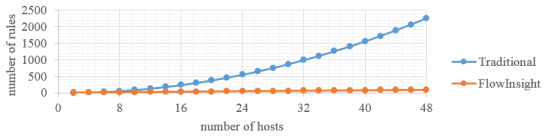

Figure 3: Comparison of the number of rules between traditional method and FlowInsight

In traditional implementation, for each packet <*src*, *dst*> sent to the controller, a corresponding rule with match field <src_MAC=*src*, dst_MAC=*dst*> should be installed. When monitor wants to get the statistics, it should pull all the counters, and groups them by their source addresses. In our implementation, for each packet <*src*, *dst*>, the monitor installs a rule with match field <src_MAC=*src*> in table 0, and the learning switch learns the *in_port* corresponding to the *src* and installs a rule with match field <dst_MAC=*src*> and action field <output_port=*in_port*> in table 1. When monitor wants to get the statistics, it should pull all the counters in table 0.

We tested the number of rules used with different number of hosts in mininet. The result is shown in Figure 3. The number of rules needed by traditional method grows quadratically with the number of hosts. Our methods reduces it down to linear growth.

4. CONCLUSION

Our contributions are, 1) presenting the problems introduced by the coupling between the visibility and the operability in the data plane, 2) proposing FlowInsight to solve the problems, and 3) implementing with OpenFlow 1.3 to show the feasibility and effectiveness of our method. In the future, we plan to add other functionalities to fully solve the problems.

Acknowledgment

This work is supported by the National High-tech R & D Program ("863" Program) of China(No.2013AA013505), the National Science Foundation of China (No.61161140454), National Science & Technology Pillar Program of China (No.2012BAH01B01), the National Science Foundation of China (No. 61303194), and the China Postdoctoral Science Foundation (No.2013M530047). Jun Bi is the corresponding author.

5. REFERENCES

[1] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proc. SIGCOMM*, 2011.

[2] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. In *ACM ICFP*, 2011.