FlowInsight: Separating Visibility and Operability in SDN Data Plane

Guang Yao, Yuliang Li, 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)

Abstract

The flow table of current SDN data plane is overloaded: it determines the operability whereas implies visibility of flows. A number of problems are introduced due to the operability and visibility are tightly coupled. In this article, we propose FlowInsight, which has separated operability and visibility provisions in the data plane. FlowInsight enables a more explicit, flexible, and efficient flow visibility provision in SDN.

1. Introduction and Problem Statement

One of the most significant differences between SDN and traditional networks is that the applications generate flow entries based on the visibility of flows. For example, in SDN, an access control application may determine whether a flow should be allowed through checking the header of the first packet of each flow [1]; a load-balance application may require getting the elephant flows and their rates to perform flow scheduling [2]. Thus, adequate visibility should be provided by the SDN data plane to make the applications and the whole network work properly.

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 statistics performed on each flow entry. It can be found whether the controller can see a flow is wholly determined by the flow table. However, the visibility defined in the flow table has following problems:

1. **Invisible flows**. The visibility of a part of flows may be required by some of the applications, but gets lost due to the limitation of visibility provision capacity in the current design. For example, a wildcard flow entry which is installed by a forwarding application will mask all the matched micro-flows which should be visible to an access control application; the elephant flows have finished between two pulls are invisible to the load-balance application.
2. **Unnecessary visibility**. The visibility of a part of flows may be unnecessary to any of the applications, but it is provided with costing extra resource or reducing the network performance. For example, whenever a load-balance application queries elephant flows from the flow table, all the mice flows are also unnecessarily got.
3. **Performance degradation.** The provision of the visibility may significantly degrade the performance of the data plane. For example, to ensure every micro-flow can be seen by the controller, wildcard rules cannot be used and significant latency is introduced in forwarding.

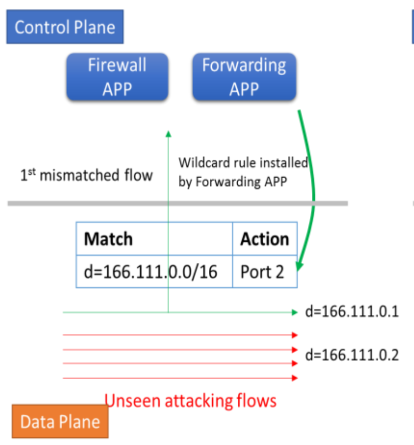
Though the visibility provided by the SDN data plane has been much better than the legacy data plane, it is still somewhat ambiguous, inflexible and inefficient. We consider these problems origin from the overload of flow table: on one hand, the flow table defines the operations to be performed on flows; on the other hand, it implies the visibility provided by the data plane. As a result, the provision of visibility is tightly coupled with the operations. We believe the visibility and the operability of flows should be decoupled to achieve better visibility. In this article, we proposes FlowInsight, which is a novel architecture of the SDN data plane. FlowInsight separates the provision of visibility and operability, therefore it can support a more explicit, flexible, and efficient visibility provision. Though there have been a number of works which separate visibility and operability in the control plane, e.g., Frenetic [3], we believe the separation in data plane is more natural and effective.

2. FlowInsight

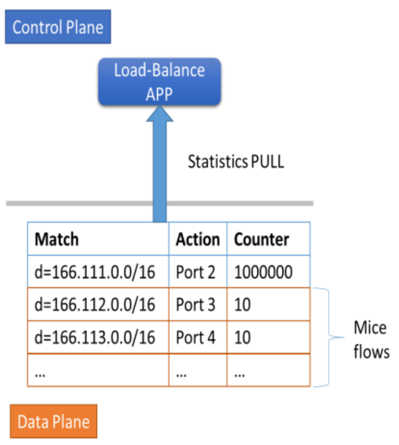
There are two conceptual parallel modules in FlowInsight: FlowOps and FlowView. FlowOps, which works the same as current OpenFlow data plane, determines the operations on each flow; FlowView determines whether each flow should be visible to the controller. Flows are processed by the two modules in parallel. In another word, the visibility and operability of flows are independent.

The FlowView module has three components: FlowDB, ViewDB and FlowView Agent. FlowDB is a database keeps the statistics for all the micro-flows, ViewDB keeps the visibility rules from applications. FlowView Agent handles requests from applications, translates the requests to rules in ViewDB and reports flows to the controller.

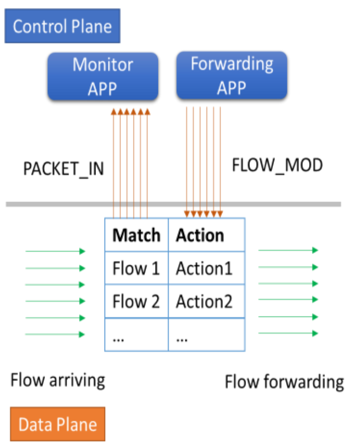
* Jun Bi is the corresponding author.



(1) Unseen flows



(2) Unnecessary view



(3) Forwarding latency

Figure 1. The problems introduced by the overload of flow table

The FlowView module supports 3 types of visibilities:

1. See-on-first-packet: applications can specify a scope, in which the first packet in this scope, or the first packet of each micro-flow in this scope, will be redirected to the controller. In this way, wildcard rules can be used to forward packets, whereas the visibility of all the micro-flows can be achieved. Besides, because flows are processed separately by the FlowOps module and FlowView module, the provision of visibility will not introduce latency in flow forwarding.
2. Query-by-condition: applications can query the FlowDB with conditions. For example, applications can query the TCP flows whose destination port is 80. Then, the applications can see the flows which they actually cares about without fetching the whole flow table.
3. Trigger-on-condition: applications can specify a condition. Once a flow meets the condition, the flow will be reported to the controller. For example, a load-balance application can require whenever the total bytes of a flow reach 1M, the flow should be reported to the controller.

The FlowView module can be implemented in the slow path as it does not have to process each packet. It can use the first packet of each flow to setup entries in FlowDB. Then an exact match rule is added in the FlowOps module to generate statistics. The FlowView module periodically gets the statistics to update the FlowDB.

3. Conclusion

In this article, we present the problems of flow visibility provision in current SDN data plane and claim they are introduced by the overload of flow table. We propose FlowInsight which has separated operability and visibility provision modules, and illustrate the architecture and mechanisms of FlowInsight.

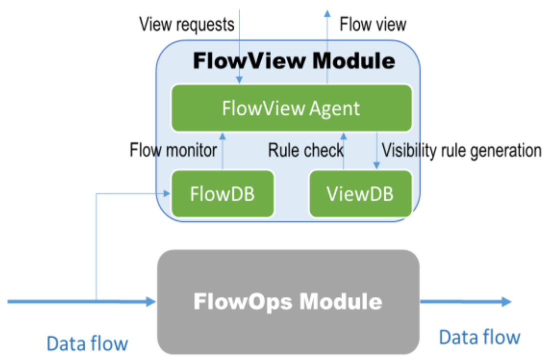


Figure 2. The architecture of FlowInsight

Acknowledgements

This work is supported by the National High-tech R&D Program ("863" Program) of China(No.2013AA010605), 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 Post-Doctoral Funding of China(2013M530047).

References

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise.", SIGCOMM 2007.
[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," NSDI 2010.
[3] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. "Frenetic: A Network Programming Language.", ACM ICFP 2011.