Tag-Based Classification for Software-Defined Networking

Hamid Farhady, The University of Tokyo, Tokyo, Japan
Akihiro Nakao, The University of Tokyo, Tokyo, Japan

ABSTRACT

Software-Defined Networking (SDN) increasingly attracts more researchers as well as industry attentions. Most of current SDN packet processing approaches classify packets based on matching a set of fields on the packet against a flow table and then applying an action on the packet. The authors argue that it is possible to simplify this mechanism using single-field classification and reduce the overhead. They propose a tag-based packet classification architecture to reduce filtering and flow management overhead. Then, they show how to use this extra capacity to perform application layer classification for different purposes. The authors demonstrated their evaluation results to indicate the effectiveness of the proposal. Furthermore, they implemented a customized user-defined SDN action that addresses some security challenges of one of their previous works and showed performance evaluation results.

Keywords: Data Plane, Network Virtualization, Northbound, Software-Defined Networking, Southbound

1. INTRODUCTION

Software-Defined Networking (SDN) increasingly attracts more researchers as well as industry attentions. Last year, Google announced that it is using SDN to improve their internal network between data centers. An out of the box outline of SDN roadmap shows significant improvements in network control and management. While there are many efforts towards developing control plane architecture and applications such as OpenFlow (Open-NetworkingFoundation, 2012), ForCES (Yang et al., 2005) and rule based forwarding (Popa, Egi, Ratnasamy, & Stoica, 2010), there is a limited attention to design effective APIs and methods for data plane.

OpenFlow is a wide-spread API for Software-Defined Networking. This API installs a set of \(<match, action>\) rules on network switches. Any flow matching to the rule, receives the corresponding action. The matching process classifies particular flows for the action. Similar to many other protocols (e.g., SNMP (Stallings, 1998)) to classify incoming flows, OpenFlow parses the packet against predefined protocol headers (e.g., Ethernet, IP and TCP). Then, it matches extracted fields against a set of rules. If enough fields match to a rule, it applies the corresponding action (e.g., drop or
forward to port X) to the packet. One of the problems in this process is that the protocol headers should be predefined. Updating such patterns in a timely fashion as well as keeping the performance at a reasonable level is not a trivial task. Because of performance considerations, matching process is implemented on hardware (e.g., Ternary Content Addressable Memory).

1.1. Flexibility of Flow Classification

An obvious solution is to support more protocols via adding more fields. Probably this is the solution OpenFlow is following. It is also possible to add some general fields (i.e., smarter wildcard rules) so that users can freely add arbitrary bits to the packet bit stream. The process of adding fields may (practically) converge after a decade. Before convergence, hardware upgrades may cause some overhead costs for users. That is, the same problem current hardware-centric networks suffer from. If we neglect all these issues, flexibility of flow classification can be a major shortcoming. Low level classification based on some static or wildcard matching rules suffer from inflexibility. Furthermore, traditional header matching is already overloaded in different ways. For example, port 80 is now used for many applications that can have different processing/forwarding requirements. We cannot classify these applications easily using field-matching classification of OpenFlow. This is the main drawback of field-matching classification. Hence, any extra support for emerging protocols is costly.

1.2. Performance of Flow Classification

Even though OpenFlow is advertising the idea of SDN, the classification part is now implemented (from several vendors e.g., NEC and HP) on Ternary Content Addressable Memory (TCAM) which is a hardware. That is because researches suggest offloading classification and matching part of OpenFlow from software to hardware results at least in a $\%40$ throughput gain (Tanyingyong, Hidell, & Sjodin, 2011). It is the case when using commodity hardware. Obviously, using purpose-built hardware (e.g., ASIC) we can gain a much faster classification. Increasing requests to support new protocols/fields pushed 15 fields defined in OpenFlow 1.1 to about 40 fields in version 1.3 (OpenNetworkingFoundation, 2012). Accordingly, if a network owner buy a switch supporting OpenFlow 1.3, after next protocol specification upgrade, the hardware needs to be replaced resulting in a high CAPital EXpenditure (CAPEX). This is the problem almost any emerging technology faces while growing: to support already existing technologies as well as new ones.

Aforementioned problems are of the type “flexibility versus performance tradeoff”. In order to have the TCAM performance and at the same time open up TCAM programmability limitations, community suggests Network Processors (EZchip, 2013). Network processor Units (NPUs) are a family of ICs using system-on-a-chip technology. They provide more efficient communication specific functions than general purpose CPUs. Network processors usually include a set of APIs to program the chip in high level languages. Therefore, we may gain more flexibility than TCAM based solutions. However, while using NPUs is a step forward to use commodity hardware, they are still inefficient for core switching heavy loads. In conclusion, still we need to lighten the flow classification overhead in addition to adding faster hardware.

Our contribution in this paper are three folds. First, we propose a light classification architecture as a replacement for current classification methods which offloads the classification load to the network edge and frees some extra processing capacity. Second, we show how to use the released capacity for application layer classification. Finally, we introduce user-defined actions as a replacement for current hardware-centric actions in OpenFlow. The rest of the paper organized as follow. After explaining related work we discuss our system architecture and finally we move on to the evaluation section where we discuss implementation and measurement details.
Adaptive Processor Allocation for Moldable Jobs in Computational Grid
www.igi-global.com/article/adaptive-processor-allocation-moldable-jobs/2165?camid=4v1a