SODA: Enhancing the Data Plane Functionality of Software Defined Networking

Dan Li, *Tsinghua University,* Yirong Yu, *Tsinghua University,* Kang Li, *Kaiwang Technology*

Abstract

We design and develop a system called SODA, which enhances the data plane of SDN by providing richer semantics and more flexible programming interfaces. SODA's innovation on the data plane includes arbitrarily setting the keys of forwarding rules, enabling (compute, action) operations on the forwarding rules, packet payload processing, and stateful processing on a sequence of packets. We develop a software implementation of SODA switch prototype based on CPU and GPU.

I. Introduction

As the de facto standard of SDN, OpenFlow defines a way to abstract the data plane functionality, i.e., the flow table. The flow table in OpenFlow opens the network functionality in the following ways. First, rather than using only the destination IP address as the key of forwarding rules, OpenFlow allows network operators to flexibly choose the combination of a number of protocol fields in the packet header as the key. Second, rather than running a single routing protocol (e.g., OSPF) to calculate the routing path in the granularity of destination IP address, OpenFlow provides operators the flexibility to set the routing paths in flow level.

OpenFlow is successful in opening the definition of "forwarding path" to network operators. In this work, we argue that although OpenFlow can satisfy the requirement of many network paradigms, there are also scenarios where network operators require much more flexibilities in the network's data plane. In particular, in data center network which supports cloud computing, different cloud applications require different network functionalities, e.g., network virtualization, in-network computation, etc. Note that whatever programming capability is provided in the control plane, data plane is the core to realize the network functionality customized by operators or users. As a result, richer semantics in the data plane of SDN is a necessity in many networks.

In this paper, we present the design and development of an SDN architecture called SODA, which enhances the SDN data plane by providing more flexible operation interfaces. SODA is compatible with OpenFlow, but makes four aspects of innovation in the data plane. First, SODA uses byte fields to define the keys of forwarding rules. Hence, operators can define new packet headers and forward packets based on the new fields. Second, SODA uses (compute, action) to define the forwarding behaviors, not only (match, action). Third, SODA supports process on the packet payload, not only the packet header. Fourth, SODA's data plane can process on a sequence of packets, instead of only individual packets. SODA controller provides the communication interfaces to SODA switches and programming interfaces to operators to support the semantics in SODA's data plane. Our software implementation of SODA switch can achieve a throughput of 80Gbps.

II. SODA Design

Fig. 1 shows the basic functionalities provided in SODA's data plane. SODA's data plane abstract is compatible with OpenFlow, but makes innovations to provide more flexible operations. In what follows we describe the SODA's data plane semantics as well as the design of SODA controller.

A. Arbitrary Keys of Forwarding Rules

In SODA switch, the keys of forwarding rules is defined by byte fields in the format of {(locator 1, offset 1), (locator 2, offset 2), ..., (locator k, offset k)}, instead of using standard protocol fields such as destination IP address, destination MAC address, VLAN tag, etc. As a result, network operators and users can introduce new packet headers and arbitrarily choose the fields to define the keys of forwarding rules. It is especially useful for new data center network protocol stacks such as BCube [1], in which the forwarding rules are defined upon the an invented layer 2.5 packet header.

B. (Compute, Action) Operation on Forwarding Rules

OpenFlow uses a (match, action) way to abstract the operation on the forwarding rules in the flow table. Essentially, 'match' is a '$=$' comparison operation between a key in the packet header and a key in the flow table. SODA generalizes the way to use flow table, namely, using a (compute, action) semantic instead of (match, action) only. Here the 'compute' operation can be a general operation such as '$=$', '$>$', '$<$', '$+$', '$-$', '$\%$', etc. There are several benefits to extend the operation on the flow table besides a simple match. First, more advanced packet forwarding rules can be supported. For instance, the incoming packets of a flow can be hashed (by '$\%$' operation upon the number of output interfaces) to balance the load on all the feasible output

The work was supported by the 973 program in China under Grant 2014CB347800, and the NSF China under Grant No.61170291, No.61161140454.

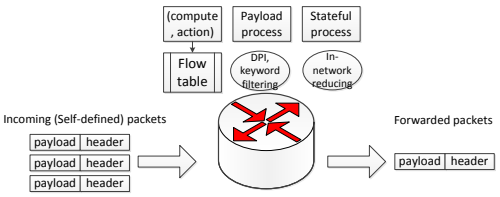

Fig. 1. Basic functionalities of SODA's data plane.

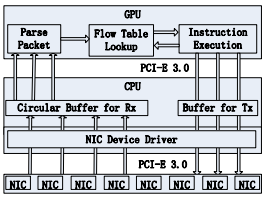

Fig. 2. Implementation of SODA software switch.

interfaces. On the contrary, the current 'group table' in OpenFlow switches can only support either forwarding the packets to all the interfaces or a randomly selected interface. Second, the size of flow table can be significantly reduced. For example, a single forwarding entry based on '$>$' operation can replace all the entries enumerated based on '$=$' operation.

C. Payload Processing

SODA not only supports forwarding rules based on packet header, but also embraces packet payload processing such as deep packet investigation (DPI) or keyword filtering. A specific processing logic is defined by operators how to deal with the payload. Typically, a flow table abstract cannot be directly used for payload processing. For instance, keyword filtering needs to lookup the whole packets to find out whether a certain keyword is included, rather than matching specific fields of the packet.

D. Stateful Processing

In legacy switches and OpenFlow switches, forwarding/processing logic is defined on each individual packet, which we call *stateless processing*. SODA switches encompasses processing on a sequence of packets, which we call *stateful processing*. We here give two examples that requires stateful processing on a number of packets. First, in cloud computing tasks such as MapReduce, if we can conduct in-network reducing, i.e., combining the mapping results of several packets into a single packet in a switch, the traffic volume in the network can be significantly reduced. The idea was proposed in Camdoop [2], but no existing network devices can directly support this kind of operation based on our knowledge. Second, in DDoS attack detection, a classical way is to analyze the traffic pattern after collecting the information of a number of packets and then find out whether the pattern is 'abnormal'. SODA switch supports this kind of stateful processing on a sequence of packets by embedded semantics.

E. SODA Controller

SODA controller collects topology and traffic information from SODA switches, configures the forwarding and processing rules onto SODA switches, and provides programming interfaces to network operators and users. The communication protocol between SODA controller and SODA switches is compatible with OpenFlow. In other words, SODA controller can also talk to OpenFlow switches. The programming interfaces SODA controller provides to programmers include specifying the locator and offset of each key used to define forwarding rules, the 'compute' instruction on certain keys (unary operation, binary operation, or multi-operands operation), processing logic on the packet payload, as well as the way to deal with a sequence of packets.

III. Implementation

Given the rich functionality provided in SODA's data plane, it is difficult to implement it in hardware. As a result, we turn to a software solution. A challenge here is how to achieve high forwarding speed by pure software implementation. We implement SODA switch based on Indigo software switch [3], and make several technical contributions to increase the forwarding speed. First, we depend on the parallel processing capability of GPU to provide enough processing power. Second, we customize the main board to increase the I/O speed by PCI-E 3.0. Eight 10G NIC cards and two GPU processors can be simultaneously supported. Third, we optimize the driver of NIC cards, user-level and kernel-level interaction and flow table lookup algorithm to increase the throughput and reduce the latency for packet forwarding. Fig. 2 illustrates the implementation architecture of SODA software switch. Current test shows that our preliminary implementation can achieve higher than 60Gbps forwarding speed. SODA controller is implemented based on Floodlight [4]. We realize all the functionalities in OpenFlow 1.3, and extend the communications protocol and programming interfaces to embrace the richer data plane functionalities in SODA.

References

[1] C. Guo, G. Lu, D. Li, etc., "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers", ACM SIGCOMM'09.
[2] P. Costa, A. Donnelly, A. Rowstron, and G. Shea, "Camdoop: exploiting in-network aggregation for big data applications", NSDI'12.
[3] http://www.projectfloodlight.org/indigo/
[4] http://www.projectfloodlight.org/floodlight/