Control-data plane intelligence trade-off in SDN

Yash Sinha
Siddharth Bhatia
Birla Institute of Technology and Science, Pilani, India

Introduction

With the decoupling of network control and data planes, the upcoming Software Defined Networking (SDN) paradigm advocates better network control and manageability. It introduces logical centralized control, network programmability and abstraction of underlying infrastructure from network services and applications. With global visibility of network state and central control that eases real time monitoring, policy alterations etc., it certainly enhances network security inherently. However, the separation of planes opens up new challenges like denial of service (DoS) attack, saturation attack, man-in-the middle attack and so on.

Many of the issues of controller availability, controller-switch communication delay and scalability can be solved separately by distributed controllers, out-of-band communication links and parallelization respectively. Control-data plane intelligence trade-off has the potential to solve all of these. It increases controller availability, reduces latency for traffic engineering & decision making, and improves controller scalability. Moreover, control-data plane intelligence trade-off enables the control-data plane communication to be more secure. This will tremendously offload the processing load on the controller. We present how to realize control-data plane intelligence tradeoff extending OpenFlow.

Issues at CDPI

Controller-switch semantic gap

Stateful applications such as firewalls heavily depend on the communication between the switch and the controller and the controllers among themselves. If network state changes, latency in distribution of this information can lead to incorrect behavior.

The distribution of access control supporting aggregated flows, multi-tenant controllers, and multiple controllers in a single domain can create configuration conflicts.

Control-data plane intelligence trade-off

There are recommendations by the researchers to delegate the decision making of the controller partially to the switches to overcome the issues due to latency in switch-controller communication, partial controller unresponsiveness due to load etc. This adds further complexity to maintain control plane states, discover and avoid security loopholes and mitigate delayed response. Nevertheless, it can help mitigate issues of latency, availability, fast reactivity and security.

Extending Open Flow

We propose to relax separation of control operations at the controller and include following operations in the forwarding elements

Network Monitoring

Monitoring networks and collecting statistics is just a repetitive task and this cannot be classified strictly as a control plane task. If the switch can get to know from the controller certain parameters regarding what to monitor and what to store, it can very well perform this task. This will offload significant load on the controller as well as reduce latency for controller-switch communications.

1. Message Generation

Similar to PortsStats and FlowStats requests sent by the controller to request statistics from the switches, a particular switch, say root of the spanning tree, can send similar packets to the switches connected in the tree and accumulate statistics.

This can be realized by a general message generator and processing function on OpenFlow switches.

2. Message Response

Message response is already supported by the switches in response to controller's request for statistics. This functionality can be extended to react to statistics' request from other switches.

Link Encryption

To prevent man in the middle attack, it is crucial to have secure connections between the switches. To expedite decision making for routing flows and thus improve upon controller-switch communication latency, switches need to share stateful information. We have described in subsections C and D below. This requires links to be encrypted. Similar to network monitoring, link encryption is used here just as a mechanism and not as a network controlling/managing entity.

Flow rules installation based on local heuristics

In switches that support dual stack, traditional protocols like Open Shortest Path First (OSPF) have been used along with the SDN controller to improve traffic engineering in hybrid SDN models. Therefore, we recommend having a similar low level heuristic to route flows at the switch, in case communication with the controller is delayed.

This can utilize the local, real-time data collected by the network monitoring module about the local vicinity similar to OSPF Hello messages.

Thus, it is more efficient in terms of spatial and temporal locality for collecting network monitoring data.

By only sending aggregated statistics to the controller, there will be lot of reduction in controller load. This will increase available bandwidth in the controller-switch communication channel thus enabling better scalability.

Network State Sharing

State sharing using east-west bound APIs at the controller yet again cannot be classified strictly as a control plane task, if it is implemented as a pull based API rather than push based. If the switches are able to serve low level network state available with them, then a controller in a distributed controller environment can request the stateful information as and when it requires.

The state information not only includes network statistics and locally traffic engineered paths as outlined before; it also includes firewall information, current elephant and ant flows, processor loads of the various controllers that it was connected to in the recent past etc. The concerned controller can get this data from various sources and filter it based on timestamps

Conclusion

While assigning control functions at the switches partially instead of a central remote network controller, we enhance security and scalability extending OpenFlow. Particularly in the direction of network monitoring, link encryption, local decision making and sharing network states the functionality can be shared by the data plane.