Computer Networks 92 (2015) 330–340

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Scalable and fair forwarding of elephant and mice traffic in software defined networks

Saumya Hegde*, Shashidhar G. Koolagudi, Swapan Bhattacharya

Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, India

ARTICLE INFO

Article history:
Received 31 October 2014
Revised 20 May 2015
Accepted 15 September 2015
Available online 25 September 2015

Keywords:
Software defined networking
Data center networks
Source routing
Mice and elephant traffic

ABSTRACT

A software defined network decouples the control and data planes of the networking devices and places the control plane of all the switches in a central server. These flow based networks do not scale well because of the increased number of switch to controller communications, limited size of flow tables and increased size of flow table entries in the switches. In our work we use labels to convey control information of path and policy in the packet. This makes the core of the network simple and all routing and policy decisions are taken at the edge. The routing algorithm splits the elephant traffic into mice and distributes them across multiple paths, thus ensuring latency sensitive mice traffic is not adversely affected by elephant traffic. We observed that label based forwarding and traffic splitting work well together to enable scalable and fair forwarding. Our approach is topology independent. We present here a few preliminary simulation results obtained by running our routing algorithm on random network topologies.

1. Introduction

Data center networks must evolve to better adapt to the challenges brought about by virtualization and cloud computing. Traditional networking switches are not capable of handling these without degrading the performance. These switches are vertically integrated where the application specific hardware chips, the hardware and the entire software stack are singly sourced. This tight coupling makes it hard to program the switches. It also makes it difficult for the switches to evolve fast enough to keep up with the demands of virtualization, cloud computing and new applications developed.

Hence it is desirable to separate the control path from the data path, i.e. decouple the network control (software, protocol and state) from the network hardware. This separation is what software defined networking (SDN) aims to achieve [2]. With SDN the control path of the switches resides in a central controller on a server and the data path continues to reside on the switch as shown in Fig. 1.

SDN architecture is characterized by three important features: separation of control plane from the data plan; centralized control and centralized view of the network; programmability of the network by external applications using the controller. Data centers have a single administrative domain whose control is centralized, thereby making it suitable to use SDN in data centers.

In SDN architecture, the ingress switch encapsulates the first packet of a new flow and sends it to the SDN controller. The controller runs the required module to identify a path for the flow. The forwarding rules are then installed in the flow tables (FT), on all the switches along the path of the flow. Only the flows matching these flow entries in the switches are acted on as per the controller's instructions. Packets which fail to match the flow entries are discarded or sent to the controller. This communication between the controller and the switches is shown in Fig. 2 and is done

* Corresponding author. Tel.: +91 9482513958.
E-mail addresses: hegdesaumya@gmail.com (S. Hegde), koolagudi@yahoo.com (S.G. Koolagudi), bswapan2000@yahoo.co.in (S. Bhattacharya).

http://dx.doi.org/10.1016/j.comnet.2015.09.014
1389-1286/

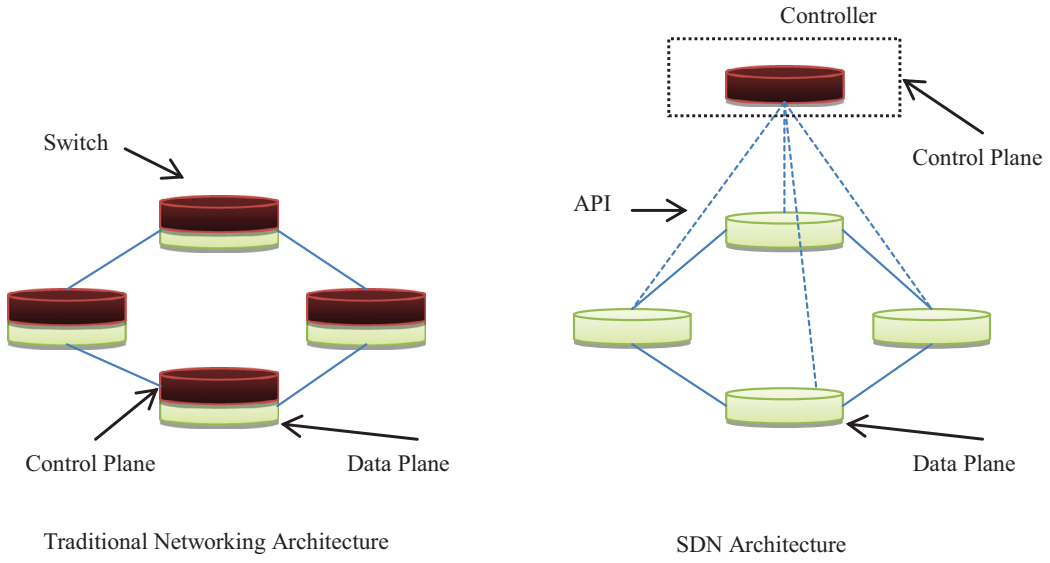

Fig. 1. Traditional network architecture vs. SDN architecture.

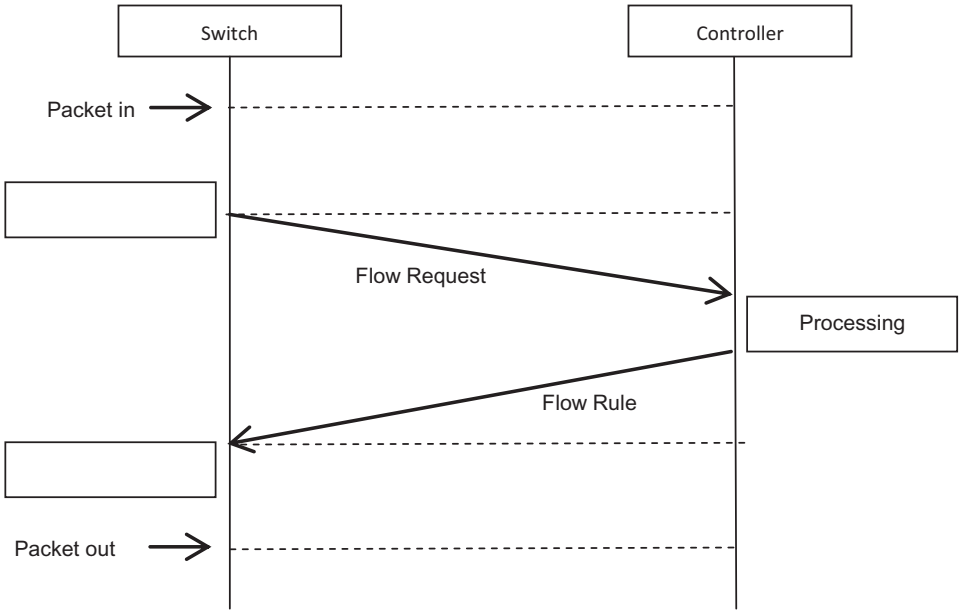

Fig. 2. Routing in SDN.

thorough various APIs, that allow the controller to address a wide variety of operating requirements.

Since SDN provides a global view of the network, it is now possible to take centralized decisions about routing and react to changes in the network state dynamically and redirect flows. This is different from the distributed routing approach of traditional networks, where decisions about routing, redirection etc. had to be taken locally. Routing which was essentially a distributed function can now be done centrally by the controller. Path computation can be done proactively before the flow begins, using the complete view of the network available at the controller. Network policy enforcement which had to be done on a switch by switch basis can now be implemented programmatically using the controller.

Although SDN provides a centralized view and allows for programmability of the network, it has some inherent scalability issues. Fig. 3 shows an example network where a packet is forwarded from source host s to destination host d. Five communications between the controller and switch are required, in order to forward the packets of this flow. Also the flow tables of switches S_1, S_2, S_3 and S_4 have to store the forwarding rules. The scalability issues here are twofold: (i) communication between the switch and the controller, (ii) memory required on the TCAM switches to store the forwarding rules. Source routing alleviates this problem since