Computer Networks 92 (2015) 300–315

Contents lists available at ScienceDirect

Computer Networks

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

JumpFlow: Reducing flow table usage in software-defined networks

Zehua Guo [a], Yang Xu [b,*], Marco Cello [c], Junjie Zhang [b], Zicheng Wang [b], Mingjian Liu [b], H. Jonathan Chao [b]

[a] School of Electronics and Information, Northwestern Polytechnical University, Xian 710072, China
[b] Department of Electrical and Computer Engineering, New York University Polytechnic School of Engineering, NY 11201, USA
[c] Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, University of Genoa, Genoa 16145, Italy

ARTICLE INFO

Article history:
Received 15 November 2014
Revised 5 June 2015
Accepted 25 September 2015
Available online 5 October 2015

Keywords:
Software-defined networking
Forwarding
Flow table management

ABSTRACT

The forwarding scheme in Software-Defined Networking (SDN) is usually coupled with flow table management. To reduce the redundancy in the flow tables of OpenFlow switches, some recent studies propose forwarding flows using stacked MPLS labels, in which each label in the stack indicates the forwarding decision at one hop of the forwarding route. However, using multiple MPLS labels in each packet introduces significant transmission overhead, especially in networks with large diameters.

In this paper, we propose JumpFlow, a forwarding scheme that achieves low and balanced flow table usage in an SDN by properly and reactively placing flow entries on switches. To reduce the transmission overhead, JumpFlow uses the available VLAN identifier (VID) in the packet header to carry routing information. Constrained by the limited space of the VID, a flow's complete routing information must be divided into several sections and loaded separately at different switches on the flow's forwarding route. To achieve low and balanced flow table usage, we formulate and solve the reactive flow entry placement problem. We evaluate JumpFlow against the per-hop configuration-based forwarding of OpenFlow for both unicast and multicast scenarios in a real network topology with different traffic patterns. For the unicast scenario with different new flow arrival rates, JumpFlow postpones the time when the first flow rejection occurs, reduces the flow rejection percentage by 37.06%, and reduces the control messages for route configuration by 53.52% on average. For the multicast scenario with a high new multicast group arrival rate, JumpFlow increases the ratio of accepted multicast groups by 83.90%, and reduces the ratio of average control messages for a multicast group configuration by 32.68%.

1. Introduction

In recent years, as network applications have experienced rapid growth, Software-Defined Networking (SDN) has attracted significant attention from academia and industry [1]. With the separation of the control plane from the data plane and the logically centralized control, SDN enables flexible network control and thus provides tremendous opportunities for network innovations, such as traffic engineering [2–4], load balancing [5,6], cost saving [7–10]. One key enabler of SDN is OpenFlow, which uses a flow entry-based abstraction to enable multiple network functions, such as layer 2 forwarding, layer 3 routing, and layer 2–4 admission control. Owing to the consideration of power

* Corresponding author.
E-mail addresses: guolizihao@hotmail.com (Z. Guo), yang@nyu.edu (Y. Xu), marco.cello@unige.it (M. Cello), jz733@nyu.edu (J. Zhang), zw640@nyu.edu (Z. Wang), mingjian.liu1@gmail.com (M. Liu), chao@nyu.edu (H.J. Chao).

http://dx.doi.org/10.1016/j.comnet.2015.09.030
1389-1286/

consumption and manufacturing cost, OpenFlow switches are usually equipped with limited flow table space, which may become insufficient when there is a large number of flow entries demanded by the network [11–13]. Thus, designing a scheme that can efficiently manage flow tables is a critical problem for practically deploying and applying SDN technology in large-scale production and data center networks.

In SDNs, flow entries can be placed in flow tables either proactively, during the network initialization, or reactively, when flows enter the network for the first time. Whereas there are some proactive flow entry placement schemes that address the problem of limited flow table space [11,14–16], less effort has been devoted to reactive flow entry placement, which is also an important part of flow table management. On the one hand, proactively placed and reactively placed flow entries are used for different applications: proactively placed flow entries [11,14–16] are usually used for applications that require bandwidth and/or delay guarantees (e.g., virtual private network and enterprise applications), whereas reactively placed flow entries are employed to provide best-effort service, such as on-demand applications (e.g., online load balancing). On the other hand, even for the same application, reactive flow entry placement provides a better chance to improve the network performance than proactive flow entry placement: as new flows enter the network, they can be directed to less congested routes or forwarded to less loaded servers based on the current network status [6].

The flow table management scheme is usually coupled with the forwarding scheme. Following the basic OpenFlow principle, the per-hop configuration-based forwarding scheme (denoted as OpenFlow) uses multiple flow entries along a route to forward a single flow and thus introduces huge redundancy in flow tables (see Section 2.1). To reduce the redundancy in flow tables, recent works in [17–20] propose an MPLS label-based forwarding scheme in which switches forward packets based on multiple MPLS labels that carry the routing information (i.e., the forwarding port number of each switch on the route, similar to that of source routing). With the MPLS label-based forwarding scheme, the controller must only reactively install one flow entry in the ingress switch to encapsulate MPLS labels for each flow. Unfortunately, encapsulating the routing information in MPLS labels introduces extra transmission overhead, especially for small packets and networks with large diameters, resulting in bandwidth waste (see Section 2.2).

In this paper, we propose an efficient forwarding scheme for SDNs named JumpFlow, which makes use of the MPLS label-based forwarding scheme concept and prevents bandwidth waste. JumpFlow uses the available VLAN identifier (VID) of the packet header to carry the routing information. Considering that the VID has limited space and can carry only limited routing information, the controller must to divide a flow's complete routing information into several sections and load them on a few selected *contact switches* on the flow's forwarding route. We formulate the reactive flow entry placement problem in JumpFlow with the objective of achieving low and balanced flow table usage by properly selecting the placement and the number of contact switches. Note that JumpFlow can be applied to reactively place both exact-matched and wildcard-matched flow entries. For ease of understanding the proposed JumpFlow scheme, we present only the case of reactive exact-matched flow entry placement, and in the rest of paper, the reactive exact-matched flow entry placement will be called reactive flow entry placement. We evaluate JumpFlow against the per-hop configuration-based forwarding of OpenFlow for unicast and multicast scenarios in a real network topology with different traffic patterns. For the unicast scenario with different new flow arrival rates, JumpFlow postpones the time when the first flow rejection occurs, reduces the flow rejection percentage by 37.06%, and reduces the control messages for route configuration by 53.52% on average. For the multicast scenario with a high new multicast group arrival rate, JumpFlow increases the ratio of accepted multicast groups by 83.90%, and reduces the ratio of average control messages for a multicast group configuration by 32.68%.

The major contributions of this paper are listed as follows:

1. We summarize the existing per-hop configuration-based forwarding of OpenFlow and the MPLS label-based forwarding and analyze their advantages and disadvantages. Based on the analysis, we propose JumpFlow, a forwarding scheme that eliminates bandwidth waste in the MPLS label-based forwarding scheme by embedding the routing information in the unused fields (e.g., VID) of the packet header.
2. We include a piecewise function used in Internet traffic engineering in our cost function to achieve joint low and balanced flow table usage and then formulate and solve the reactive flow entry placement problem by properly selecting the placement and the number of contact switches for each flow.
3. We evaluate JumpFlow in a real network topology with different traffic patterns. Simulation results demonstrate that JumpFlow achieves better performance for unicast and multicast scenarios than baseline schemes.

The rest of the paper is organized as follows. Section 2 introduces the problem background and discusses the motivation for this paper. Section 3 presents JumpFlow's framework. Section 4 formally formulates and solves the reactive flow entry placement problem. Section 5 presents the evaluation strategy and compares JumpFlow to baseline schemes. Section 6 differentiates JumpFlow from some related schemes and discusses some concerns for JumpFlow. Section 7 reviews the related work. Section 8 concludes the paper.

2. Background and motivation

2.1. Per-hop configuration-based forwarding of OpenFlow

OpenFlow provides a per-hop configuration-based forwarding, which works as follows: when the first packet of a new flow enters the network from an edge switch, the switch has no specific entry in its flow table for the flow and sends a flow setup request to the controller, which will calculate a route for the flow and install flow entries coupled with specific actions in the flow tables of the switches on the route.

The above operation introduces huge redundancy in flow tables as illustrated in Fig. 1. When a flow traverses five