Computer Networks 103 (2016) 44–55

Contents lists available at ScienceDirect

Computer Networks

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

An efficient routing algorithm based on segment routing in software-defined networking

Ming-Chieh Lee, Jang-Ping Sheu*

Department of Computer Science, National Tsing Hua University, Taiwan

ARTICLE INFO

Article history:
Received 9 September 2015
Revised 18 December 2015
Accepted 28 March 2016
Available online 8 April 2016

Keywords:
Routing algorithm
Software-defined networking
Segment routing
Traffic engineering

ABSTRACT

Software-defined networking (SDN) is an emerging architecture that offers advantages over traditional network architecture. Segment routing (SR) defines the path of information through the network via an ordered list of multi-protocol label switching (MPLS) mechanisms on the packet headers at the ingress device, and this system makes SDN routing management more simple and efficient. SR can also solve some scalability issues in SDN. In this paper, we propose a routing algorithm for SDN with SR that can meet the bandwidth requirements of routing requests. Our algorithm considers the balance of traffic load and reduces the extra cost of packet header size in a network. Simulation results show that the performance of our algorithm is better than that of previously developed algorithms in terms of the average network throughput and the average rejection rate of routing requests.

1. Introduction

Software-defined networking (SDN) [1] is an emerging architecture that is flexible, manageable and responsive to rapid changes. This architecture enables the network controls to be directly programmed through an open standardized interface called OpenFlow [2]. The SDN paradigm provides clear advantages over traditional network architecture, and it has attracted growing attention from many businesses in recent years. SDN provides fine-grained traffic distribution control in a network, as it can decide forwarding behavior based on different combinations of packet header fields, unlike traditional coarse-grained systems of destination-based forwarding. However, the SDN attribute implies that an SDN switch requires a larger sized flow table than that of a traditional switch for storing the same number of flows [3]. Flow tables are implemented by ternary content addressable memory (TCAM) [4], which is an extremely expensive and power-hungry resource. In general, TCAM can support from a few hundred to several thousand entries which direct SDN in facing challenges such as network scalability [5].

SR is currently undergoing an Internet Engineering Task Force (IETF) draft [6], which is driven by Cisco and is supported by many leading telecom companies. In general, the SR approach shows promise as an alternative network-operating model. SR is a network technology that offers a new method of packet forwarding that minimizes the need for keeping large numbers of network information states and therefore helps to overcome the TCAM deficiency problem. This approach can definitely add capacity to IPs and multi-protocol label switching (MPLS) networks, as the SR data plane can be applied to IPv6 and MPLS. There are many advantages when we integrate SR in the data plane and in SDN-based control layer technologies. As for scalability and agility, SR avoids the requirement for millions of label codings to be stored in each network device along the path, and it reduces the number of forwarding rules in the TCAM. Furthermore, SR eliminates the complexity of maintaining large numbers of forwarding rules, so there are no communication delays between network devices and the SDN controllers [7].

In this paper, we study the traffic engineering issue in SDN [8] with SR. We show that through traffic engineering, we can achieve the goals of optimizing network performance and network resource utilization. With SR, the SDN controller only needs to encode end-to-end route information into the packet header as an ordered list of labels to the ingress network device. The controller does not need to add forwarding rules to each individual device along the path, and it can re-send a packet without the need for new forwarding state redistribution in the related switches. In addition, the intermediate nodes do not need to maintain any per-flow state. Intermediate nodes simply act on the routed behavior, which is recorded in the segment header. This approach provides a much more simple and scalable solution for traffic engineering. However, as OpenFlow [9] defines each MPLS label with 32 bits (which is a large label), SR has a drawback in that it requires extra per-packet header size overhead. This requirement arises because

* Corresponding author.
E-mail addresses: s102062516@m102.nthu.edu.tw (M.-C. Lee), sheujp@cs.nthu.edu.tw (J.-P. Sheu).

http://dx.doi.org/10.1016/j.comnet.2016.03.017
1389-1286/

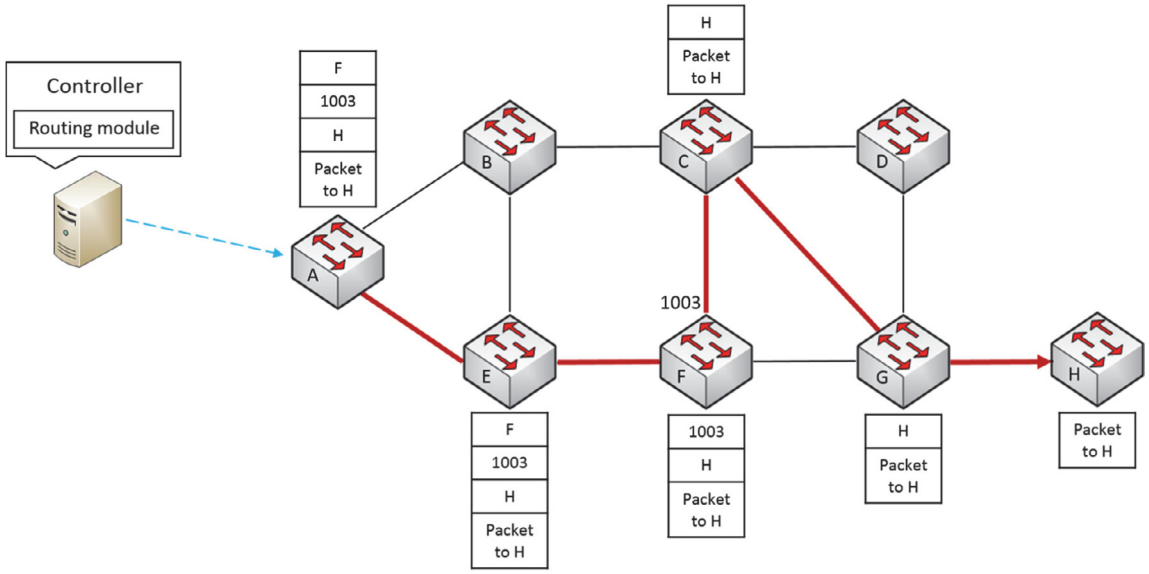

Fig. 2.1. SR traffic engineering with an SDN controller.

SR represents the route information as a set of labels, and it implements the labels by using the MPLS label field in the packet header.

In this paper, we propose a heuristic routing algorithm with a bandwidth guarantee. The proposed algorithm not only achieves the goal of using traffic engineering to balance the network traffic load, but it also promises to reduce the network overhead cost. In other words, our algorithm adopts an extra-hop limitation, and it considers a link's residual bandwidth and link criticality. Link criticality is based on the concept of the minimal interference routing method (MIRA) [10]. This concept is applied to minimize the possibility of rejecting requests when the network becomes overloaded, with an aim of upgrading the network's performance. We obtain a better network performance than that of other traditional routing algorithms such as the shortest path first (SPF) [11], widest shortest path (WSP) [12], shortest widest path (SWP) [13] or MIRA [10] algorithms. Extensive simulations show that our algorithm can lower the unsatisfied request rate and raise the bandwidth satisfaction rate compared to that of the previous approaches.

The rest of this paper is organized as follows. We introduce the related previous works in Section 2. In Section 3, we present our proposed algorithm. The performance evaluation is presented in Section 4, and Section 5 concludes the paper.

2. Related works

2.1. Preliminary of segment routing

For each pair of host communications within the same SDN/SR domain, a network uses an interior gateway protocol (IGP), such as the open shortest path first (OSPF) protocol, to enable routing to the destination by default. SR [6] uses a *node segment* to represent the action that a packet should follow to take the shortest route. In other words, every node in the same SR domain keeps a *node segment* information link to each of the other nodes in its forwarding table, as the default rules and the *node segment* represents global awareness. In addition, SR adopts an *adjacency segment* to control traffic, which represents the action that the packet should transfer to a specific egress data link with an adjacent node, and this *adjacency segment* represents local awareness. However, unlike in the *node segment*, each node only needs to install its local *adjacency segment* rules in its forwarding table. The combination of *node segments* and *adjacency segments* forms a sequence list of labels that are applied to the packet header by the SDN controller, and they are reflected instantly as the desired traffic path. Therefore, SR brings several orders of scaling gains, as it does not hold any state for the flows in the intermediate devices. However, SR contributes to another noticeable problem, because it uses an MPLS label field to place the segment labels. As a result, SR may require a larger packet header, which reduces the available bandwidth.

We illustrate an overview of intra-domain SDN-based SR in Fig. 2.1. The mark on each switch is the *node segment*, and the number next to the switch, *F*, is the *adjacency segment* of switch *F*. The SDN controller computes an explicit route by the routing module, and it configures the forwarding table of the ingress switch with an ordered list of segments. For example, assume that there is a traffic demand from switch *A* to *H*. The path information {*A-E-F-C-G-H*} is encoded in the packet header as a set of MPLS label stacks, which completely removes the need for installing rules in the switches along the path. First, the switch *A* processes the top label, which is a node segment, and then forwards the packet along the shortest path toward the network device with the *node segment F*. Then, switch *F* pops the top label, and the following top label is 1003. Therefore, switch *F* forwards the packet toward its output port 1003. After arriving at switch *C*, the switch forwards the packet to *H* along the shortest path, and so on. Thus, SR can reduce the number of forwarding rules in TCAM.

The following paragraphs summarize some of the previous studies related to the SR technology. First, the authors in [14] consider the problem of determining the optimal parameters for SR in offline and online cases. The authors propose a traffic matrix oblivious algorithm for the offline case, and another algorithm for the online case. In the online case, the network has a centralized controller that can use an online approach to solve the SR problem, as is done in our SDN-based environment. These authors give formulas and linear programs for defining the optimal parameters of SR. Their paper focuses on determining the optimal parameters as traffic split values. These values are applied to minimize the worst-case link utilization by taking equal-cost multipath routing (ECMP) into account in offline cases. The traffic split values also serve to minimize rejections of requests in online cases. However, our research is dedicated to designing an efficient routing algorithm for better network performance (such as improved