Computer Networks 106 (2016) 161–170

Contents lists available at ScienceDirect

Computer Networks

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

BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch

L. Boero[a], M. Cello[b,*], C. Garibotto[a], M. Marchese[a], M. Mongelli[c]

[a] University of Genoa, Via all'Opera Pia 13, 16145, Genova, Italy
[b] Nokia Bell Labs, Blanchardstown Business & Technology Park, Snugborough Road, Dublin 15, Ireland
[c] National Research Council, Via De Marini 6, 16149, Genova, Italy

ARTICLE INFO

Article history:
Received 17 August 2015
Revised 25 March 2016
Accepted 20 June 2016
Available online 24 June 2016

Keywords:
SDN
OpenFlow
Packet loss
Traffic engineering

ABSTRACT

Current OpenFlow specification is unable to set the service rate of the queues inside OpenFlow devices. This lack does not allow to apply most algorithms for the satisfaction of Quality of Service requirements to new and established flows. In this paper we propose an alternative solution implemented through some modifications of Beacon, one popular SDN controller. It acts as follows: using 'almost'-real-time statistics from OpenFlow devices, Beacon will re-route flows on different queues to guarantee the observance of deadline requirements (e.g. the flow is still useful if, and only if, is completely received by a given time) and/or an efficient queue balancing in an OpenFlow SDN switch. Differently from the literature, we do not propose any new primitive or modification of the OpenFlow standard: our mechanism, implemented in the controller, works with regular OpenFlow devices. Our changes in the SDN controller will be the base for the design of a class of new re-routing algorithms able to guarantee deadline constraints and queue balancing without any modification of the OpenFlow specification, as well as, of OpenFlow devices.

1. Introduction

Software Defined Networking (SDN) is revolutionizing the networking industry by enabling programmability, easier management and faster innovation [1,2]. These benefits are made possible by its centralized control plane architecture which allows the network to be programmed and controlled by one central entity.

The SDN architecture is composed both of SDN enabled devices (switches/routers)[1] and of a central controller (SDN controller). An SDN device processes and delivers packets according to the rules stored in its flow table (forwarding state), whereas the SDN controller configures the forwarding state of each SDN device by using a standard protocol called OpenFlow (OF) [2]. The SDN controller is responsible also to build the virtual topology representing the physical topology. The virtual topology is used by the application modules that run on top of the SDN controller to implement different control logics and network functions (e.g. routing, traffic engineering, firewall actions).

Currently the Quality of Service (QoS) management in OF is quite limited: in each OF switch one or more queues can be configured for each outgoing interface and used to map flow entries on them. Flow entries mapped to a specific queue will be treated according to the queue's configuration in terms of service rate, but the queue's configuration *takes place outside the OF protocol*. For example, the queue's service rate cannot be modified by OF.

Supposing that a flow is traversing a chain of queues from the source to the destination node, and that the flow data rate increases, a possible consequence is that queues increase their occupancy, and a bottleneck may be generated with consequent network congestion. The impossibility to change the bottleneck queue's service rate through real-time OF directives can lead to a severe performance degradation for the flows traversing that queue because, without a proper rate assignment, it is very difficult to guarantee Quality of Service requirements to the flows [3].

A possible solution to mitigate the performance degradation involves the re-routing of the flows experiencing a violation of deadline constraints (e.g. the flows that are totally received beyond the fixed time constraint) [4] on less congested paths or queues. The underlying idea is that, since we cannot change the service rate of

* Corresponding author. The work has been performed while M. Cello was employed at University of Genoa.
E-mail addresses: luca.boero@edu.unige.it (L. Boero), marco.cello@nokia-bell-labs.com (M. Cello), chiara.garibotto@edu.unige.it (C. Garibotto), mario.marchese@unige.it (M. Marchese), maurizio.mongelli@ieiit.cnr.it (M. Mongelli).

[1] In the following we will use the terms: SDN device, OpenFlow device, OpenFlow switch, interchangeably, even if the term "OpenFlows switch" or simply "switch" indicates an SDN enabled device in most SDN literature.

http://dx.doi.org/10.1016/j.comnet.2016.06.025
1389-1286/

the queues, we act on the ingress traffic, moving a subset of flows on different paths or queues in case of need. In order to be 100% compatible with current OF hardware, we impose no changes to OF specifications and directives. Instead we propose to modify one popular SDN controller: Beacon [5]. The proposed solution, *BeaQoS*, applied to a single SDN switch, is an extension of our previous work presented in [6]. Our new updated controller will receive statistics about queues, flows and ports from OF switches and will compute an estimation of the flow rates and of the packet loss of the queues. Based on customizable policies, BeaQoS will be able to select a subset of flows experiencing congestion over the bottleneck queue and to re-route them on another and less congested queue, so improving the switch performances. The action of flow re-routing may be exploited not only for deadline management but also for efficient queue load balancing. On the other hand load balancing is often seen as an action to prevent congestion and, consequentially, to limit and delay performance detriment.

The remainder of this paper is structured as follows. We describe related works on this field in Section 2. Concerning the main contributions of the paper:

- We explain the motivations that lead to consider multi-queue interfaces with variable service rate to support deadline management in Section 3;
- We describe the basic idea concerning the re-routing mechanisms introduced in this paper in Section 4.1, where we also show how it can be usefully applied in case of multi-core architectures and load balancing issues among queues;
- We describe the modifications of the Beacon controller required to implement re-routing in Section 4.2;
- We propose five effective re-routing strategies in BeaQoS: two of them aimed at improving deadline management and three of them aimed at balancing the load among queues in a SDN switch in Section 5.

We show the performance analysis of our proposed algorithms in Section 5. We report a discussion about the obtained results together with the conclusions in Section 7.

2. Related works

Despite traffic engineering (TE) approaches are often ruled by MPLS-TE [7,8], the ability of the SDN controller to receive (soft) real-time information from SDN devices and to make decisions based on a global view of the network, coupled with the ability of "custom"-grained flow aggregation inside SDN devices, makes TE one of the most interesting use cases for SDN networks.

Global load balancing algorithms are proposed in [9] that addresses load-balancing as an integral component of large cloud services and explores ways to make load-balancing scalable, dynamic, and flexible. Moreover [9] states that load-balancing should be a network primitive, not an add-on, and presents a prototype distributed load-balancer based on this principle.

[10], shows that the controller should exploit switch support for wildcard rules for a more scalable solution that directs large aggregates of client traffic to server replicas. [10] also presents algorithms that compute concise wildcard rules that achieve a target distribution of the traffic and automatically change load-balancing policies without disrupting existing connections. Furthermore, the authors implement these algorithms on top of the NOX OpenFlow controller, evaluate their effectiveness, and propose avenues for further research.

The work presented in [11] shows a system that re-configures the network's data plane to match current traffic demands by centrally controlling the traffic that each service sends on a backbone connecting data-centres. [11] develops a novel technique that leverages a small amount of scratch capacity on links to apply updates in a provably congestion free manner, without making any assumptions about the order and timing of updates at individual switches. Further, to scale to large networks in the face of limited forwarding table capacity, [11] greedily selects a small set of entries that can satisfy current demands and updates this set without disrupting traffic.

Reference [12] analyses a partially deployed SDN network (a mix of SDN and non-SDN devices) and shows how to exploit the centralized controller to get significant improvements in network utilization as well as to reduce packet losses and delays. [12] shows that these improvements are possible even in cases where there is only a partial deployment of SDN capability in a network. The authors formulate the SDN controller's optimization problem for traffic engineering with partial deployment and propose a fast Fully Polynomial Time Approximation Schemes (FPTAS) to solve it.

This last problem is also tackled in [13] that introduces a traffic management method to divide, or to "slice", network resources to match user requirements. [13] presents an alternative to resort to low-level mechanisms such as Virtual LANs, or to interpose complicated hypervisors into the control plane, by introducing an abstraction that supports programming isolated slices of the network. The semantics of slices ensures that the processing of packets on a slice is independent of all other slices. They define their slice abstraction, develop algorithms to compile slices, and illustrate their use by using examples. In addition, [13] describes a prototype implementation and a tool to automatically verify formal isolation properties.

In our previous work [6], we propose a solution based on SDN, which implements a software strategy to cope with non-conformant traffic flows inside a class-based system. This approach is therefore independent of the underlying hardware, as it is conceived to run as an algorithm inside the SDN controller. The proposed strategy will manage non-conformant flows, based on a set of statistic data gathered by a modified version of the Beacon controller, in order to mitigate the quality degradation of flows traversing the network.

In order to support traffic engineering in the SDN environment, OpenFlow Management and Configuration Protocol (OF-Config) has been proposed. OF-Config [14] is a protocol developed by the Open Networking Foundation used to manage physical and virtual switches in an OpenFlow environment. This tool gives network engineers an overall view of the network and also provides the ability to set policies and to manage traffic across devices.

3. Motivations

Some approaches consider a single queue for each outgoing interface. In order to support QoS mechanisms and traffic differentiation, it is common to configure multiple queues in advance [3]. The importance of traffic differentiation is highlighted by the first group of simulations (Table 1) reported in the following.

Flow entries mapped to a specific queue will be treated according to that queue's configuration in terms of service rate. Most of the previously mentioned approaches assumes the ability of SDN/OpenFlow to set the service rate of the queues in each SDN

Table 1
Performance metrics of the traffic for 1-queue and 3-queues configurations.

Performance metric	Queue configuration	
	1-queue	3-queue
BF - packet loss	25%	71.16%
DF1 - percentage of flows matching the deadline	11.43%	74.29%
DF2 - percentage of flows matching the deadline	17.39%	19.57%