Dynamic aggregation of traffic flows in SDN Applied to backhaul networks

Kentis, Angelos Mimidis; Caba, Cosmin Marius; Soler, José

Published in:
Proceedings of 2016 IEEE NetSoft Conference and Workshops

Link to article, DOI:
10.1109/NETSOFT.2016.7502459

Publication date:
2016

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Kentis, A. M., Caba, C. M., & Soler, J. (2016). Dynamic aggregation of traffic flows in SDN Applied to backhaul networks. In *Proceedings of 2016 IEEE NetSoft Conference and Workshops* (pp. 136-140). IEEE. https://doi.org/10.1109/NETSOFT.2016.7502459

Dynamic aggregation of traffic flows in SDN

Applied to backhaul networks

Angelos Mimidis, Cosmin Caba, José Soler
DTU Fotonik – Networks Technology & Service Platforms Group
Lyngby, Denmark
{agmimi, cosm, joss}@fotonik.dtu.dk

***Abstract*—A challenge in the adoption of the OpenFlow (OF)-based SDN paradigm is related to the limited number of OF rules supported by the network devices. The technology used to implement the OF rules is TCAM, which is expensive and power demanding. Due to this, the network devices are either very costly or they can support a very limited number of OF rules. One way to cope with this limitation, is to perform the same logic but with fewer OF rules in the devices. As a demonstration of this operational strategy, the current paper proposes a service for traffic flow aggregation which reduces the number of OF rules needed in the network devices, without impacting the control plane logic. The proposed traffic flow aggregation service is tested on a set of topologies specific to the backhaul network, since they aggregate a large amount of traffic flows. The results illustrate significant reductions in the number of OF rules in network devices, thus a lower demand on TCAM capacity.**

Keywords—SDN; OF; aggregation; backhaul; QoS.

I. Introduction

When network devices analyze incoming packets, the OF protocol allows for matching on any of the packet headers, including the possibility to use wildcards (i.e. don't care bits) for parts of the headers that are not considered for the matching. This wildcarding feature is a characteristic of the Ternary Content Addressable Memory (TCAM). In consequence, the common way to implement the flow tables in OF devices, in order to fully leverage the wildcard matching, is by using TCAM [2]. A major challenge that the OF-based SDN paradigm raises is related to the limited space in the flow tables. TCAM memory, the underlying technology implementing the flow tables, is very expensive and power demanding. Due to this, the devices have in general limited resources to implement the flow tables, leading to a limited number of OF rules that can be installed in a device.

There is a correlation between the number of the traffic flows in the data plane, the number of OF rules installed in the forwarding devices, and the granularity at which the traffic flows are processed. More OF rules are needed if the processing of the traffic flows is done using fine granular OF rules (matching on Layer 4 headers). Alternatively, fewer OF rules are needed if the processing of the traffic flows is done with a coarser granularity (matching on L1 headers).

By exploiting the correlation presented above, one can design a control plane solution which processes the same number of traffic flows but with fewer OF rules in the forwarding devices. This is possible if several smaller traffic flows (i.e. microflows) are aggregated (multiplexed) into fewer, but bigger flows (termed *aggregates* throughout the paper). Ideally, this aggregation process should be performed at the edge of the network, while the core of the network will forward only the aggregated traffic flows, thus having fewer OF entries per forwarding device. The savings in the number of OF rules entries in the forwarding devices depend on the *degree of aggregation*. This is the number of microflows that are multiplexed into bigger traffic flows, which further depends on the traffic patterns and the topology of the network.

This paper proposes a solution to reduce the number of OF entries in the flow tables based on dynamically applying aggregation of traffic flows at various points in the network. Moreover, the proposed solution considers the implications that the aggregation process has on the QoS of the traffic flows. The implemented service has been tested in order to observe the savings in terms of space in the flow tables. Out of the several possible network setups to execute the tests (e.g. data centers, aggregation networks, etc.), the backhaul network has been chosen for this work. The reason for this choice is outlined in section II.

The remaining of the paper is structured as follows. Section II details the concept of OF-based SDN, argues for the necessity of traffic flows aggregation and presents the related work. In section III the description of the prototype is given and section IV contains the test results. The entire work is concluded in section V.

II. Background and Related Work

A. OF-based SDN Architecture

An OF forwarding device processes data packets by using the concept of flows. If an incomming packet matches an OF rule in the flow table of a device, then the packet is processed according to that rule. Otherwise, a notification containing the packet headers is sent to the controller (i.e. PACKET IN notification), which will then decide on how should the respective packet be processed. Eventually, the controller may install a new OF rule in the forwarding device, so that all the future packets belonging to the same traffic flow will be processed without requiring further intervention from the controller. In this way, the controller is aware of the traffic flows and the OF rules installed in the forwarding devices.

The most common underlying technology for implementing OF flow tables is TCAM [2], because it allows for fast implementation of matching rules that can support flexible wildcarding, for any of the packet headers. Since this type of memory is expensive and it consumes significant amounts of power [3], it becomes costly to build and deploy forwarding devices that can store a large number of OF rules. This further limits the adoption of OF-based SDN for environments where there is a large volume of traffic flows that must be processed (e.g. forwarded, firewalled, etc.). As an example, there are current comercial OF switches that have space for a couple of thousands of flow rules, which is insufficient even in small environments such as enterprise networks [2][3][4].

B. Aggregation Service: Overall Description

A possible way to reduce the number of traffic flows in the network is to treat several flows as a single *aggregate,* operation which is termed *traffic flow aggregation* (or simply aggregation) in this paper. The service proposed herein applies aggregation in the network in a dynamic fashion in order to cater to the limitations outlined above with respect to TCAM. By doing this, instead of installing several OF rules to match and process the individual flows, the controller installs fewer OF rules to process the flows as an aggregate.

An identifier is needed in order for the controller to be able to manage the aggregate as a single entity. The traffic flows that are to be treated as a single aggregate will then be marked with the identifier and processed accordingly in the network. Several identification mechanisms are possible such as Multi-Protocol Label Switching (MPLS) labeling, VLAN tagging, and other encapsulation formats. These mechanisms are well known from traditional network architectures and they are well suited also for the aggregation service proposed in this paper. For real life deployments the choice of the identifier is very important and it usually depends on several aspects such as support from hardware, encapsulation overhead, number of available identifiers, etc. Nevertheless, for the purpose of the current work the aggregation mechanism does not influence the efficiency of the proposed service, since the only metric that matters is the number of OF rules in the flow table, which does not depend on the identifier. Hence, for the sake of simplicity and familiarity with the development, VLAN tagging has been used to identify aggregates. Additionally, for the purpose of the current work, the limitation imposed by the VLAN on the number of available identifiers (approx. 4000) does not affect the experiments and results.

The aggregation service works in an autonomic and dynamic way, such that upon receiving a notification for a new traffic flow, the controller decides whether the flow can be aggregated with existing flows in the network. Two criteria are used for the decision: (1) the possibility that the route computed for the new traffic flow overlaps with the existing flows in the network, and (2) the possibility that the QoS of the new traffic flow matches the QoS of existing flows. Both decision criteria can be configured in various ways (e.g. the path overlapping degree, etc.), to make the service adjustable to network changes and operator management policies.

C. Literature and Related Work

Several works try to solve the challenge of reducing the number of OF rules in the flow table by investigating various configurations and characteristics of possible hardware implementations for the flow tables (e.g. types of memory). For example, [4] proposes a solution based on combining the SRAM memory, which is significantly cheaper, with the TCAM. More specifically, they argue that the simple packet processing such as MAC and VLAN based forwarding, can be done in SRAM while more complex processing involving wildcarding can be done using TCAM-based flow tables. In this way, the need for having a large TCAM is reduced.

Another work suggesting a mixture of SRAM and TCAM is [5]. An algorithm that combines several finer grained wildcards into fewer coarser grained wildcards is proposed in order to reduce the number of OF entries. This solution is applied in the context of server load balancing hence the paper does not provide tests and results to assess the impact on the degree of reduction in the flow table space.

The solutions proposed in [7] and [8] are based on the concept of *flow table aggregation,* that is restructuring of the OF rules and the wildcard matching, in order be able to match the same number of traffic flows but with fewer OF rules. This idea takes advantage of the fact that several wildcards may have a common prefix hence they can be aggregated.

The work which is most similar to the current one is presented in [6]. It is argued there that a dynamic flow aggregation mechanism is needed in order to have a more manageable number of OF rules in the core switches. While the solution proposed in [6] is not described in detail, the criteria used for the aggregation are similar to the ones presented in the current paper: path similarity and similarity of traffic types. The transport protocol and port numbers are used to distinguish between different traffic types. An important difference in the current work is that the aggregation service relies on the Differentiated Services Coped Point (DSCP) marking in the IP headers in order to distinguish between the various QoS classes for the traffic flows. In this way, it is easier for the aggregation service to take into account the QoS requirements, without deciding what QoS level should each type of traffic have. This decision is taken by a different entity when marking the packets with the DSCP codes. Another distinction from [6] is that the current paper provides a quantitative analysis of the aggregation service for various key topologies. This analysis gives a precise indication about the degree of savings that the flow aggregation service produces in the flow tables.

D. Backhaul Networks

Even though the proposed aggregation service is designed to work in any OF-based SDN environment, its efficiency in reducing the number of flow table entries depends on the network topology and the traffic patterns. Intuitively, the aggregation service is needed in environments with a large number of traffic flows that must be processed, since more OF rules in the flow tables are needed to process them. Moreover, there are topologies that better facilitate aggregation due to, for example, multiple flows sharing the same path. As a consequence, a specific set of network topologies and traffic patterns have been chosen to test the proposed aggregation service, in order to better emphasize its efficiency.

The *backhaul network* aggregates the traffic from the access networks (mobile or fixed), towards the core. With the increasing number of mobile users and the tendency towards small cell deployment, the mobile backhaul must scale up in terms of capacity and become more flexible in order to cope with the large number of user traffic flows. At the same time, the QoS is a critical aspect for service provisioning, which must be taken into account in all the areas of the network (from access to core). Due to this, the backhaul network is a suitable area to apply traffic flow aggregation, and it is the area chosen for further investigation in this paper.

The proposed aggregation service, which is based on OF, works at layer 3 in the TCP/IP protocol stack. However, the backhaul networks usually have a different protocol stack, making it challenging to use the OF protocol. For example, the backhaul for the mobile network uses the General Packet Radio Service (GPRS) Tunneling Protocol (GTP) to carry the user traffic. GTP is not supported by OF. Nevertheless, SDN has the potential to simplify the protocol stack and the traffic management in the backhaul network, and this has been investigated in the literature [9]. To conclude, the current paper focuses on the aggregation service itself and its efficiency with regard to the backhaul network, leaving out the study of integrating OF in the mobile domain.

III. Prototype Implementation

A. *Prototype Architecture*

A prototype of the aggregation service was implemented by using the open source SDN controller, Floodlight [12]. The architecture of the prototype is depicted in Figure 1.

The created aggregation service can dynamically decide whether it should perform traffic aggregation (by taking care of the forwarding process itself and stopping the pipeline processing of the packet within the SDN controller), or to allow for the normal forwarding of the packet when aggregation is not a viable option. In either case (regular forwarding or aggregation), the controller also utilizes the built in *Topology Manager,* which allows for an up-to-date and global view of the network's state. After processing the packet, the controller sends back to the OF switch the required instructions on how the packet should be processed, by using OF. To reduce the OF related delay and overhead, the controller sends instructions to all the OF switches that are part of the packet's route through the network.

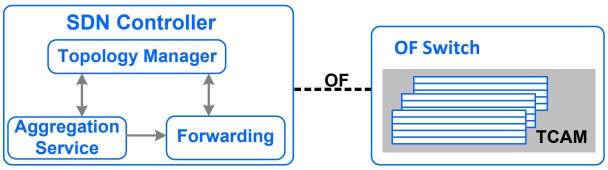


Fig. 1. Prototype architecture

B. *Aggregation algorithm*

In the proposed algorithm, traffic aggregation is performed based on two criteria, the network path that the traffic flows follow and their DSCP values. The network path was selected so that the aggregation process would be as less intrusive to the behavior of the network as possible. The DSCP values were selected as an additional criterion to ensure that the generated traffic aggregates would have similar QoS needs and as such QoS treatment could be performed by the network administrator. The distinction of the different traffic aggregates is achieved by tagging the traffic flows that belong to them with different VLAN tags (one for each traffic aggregate). The VLANs are assigned and released by a separate function that ensures each VLAN is used by at most one traffic aggregate. By utilizing this VLAN function it is also possible to specify a range of VLAN values to be used for traffic aggregation. This can be useful if another network service needs to manage its own separate VLAN pool.

There are two events that can trigger the traffic aggregation algorithm, the arrival of either a forwarding instruction request or a notification that a flow has been removed from the flow table of a switch.

1) *Arrival of a forwarding instruction request*

Whenever an OF message of this type is sent to the controller by one of the forwarding devices, the traffic aggregation module is triggered. The first step is to check whether or not the packet is an IP packet (line 2 in Figure 2). This is because the algorithm is only designed to process IP packets, so any other packets cause the aggregation algorithm to terminate and pass the packet for normal forwarding. If the packet is an IP packet, then the algorithm extracts the network path that the packet has to follow through the network, using the source and destination IP address information. If QoS differentiation is enabled, it also extracts the DSCP value of the packet. Then it checks if it is possible to aggregate the corresponding flow of this packet with an existing aggregate (lines 9-12 in Figure 2). If not, the algorithm then checks whether it is possible to create a new traffic aggregate with other un-aggregated traffic flows (lines 14-17). If aggregation is not possible in any way, the algorithm terminates and the traffic flow is processed with normal forwarding (line 19). If the flow can be aggregated only through a portion of its path, then the algorithm crops the flow's path by removing the aggregated part. The aggregation operation is then repeated, this time with only the cropped path (lines 11-12 and 16-17). This ensures that the traffic flow will be aggregated as many times as possible through its path in the network.

2) *Arrival of a flow removal notification*

In this implementation, whenever a flow entry expires in a switch, a flow removal notification is sent to the SDN controller. This is done to make sure that the controller has always an up-to-date view of the active traffic flows, so that it can effectively aggregate them when it is possible.

When a flow removal notification arrives at the SDN controller, a distinction is made whether the traffic flow entry that was removed from the flow table was a traffic aggregate or an individual flow (Figure 3). If it was an aggregate, then the information that is stored in the controller regarding the currently active traffic aggregates has to be updated and if necessary the VLAN associated with this aggregate is released. A different course of action has to be taken if the removed flow entry is associated with only a single traffic flow. In the case that the notification refers to a traffic flow that belongs to a traffic aggregate then the controller does not need to follow any further action, as the removal notification is an expected side-effect of the flow now being forwarded by its corresponding aggregate flow rules. On the other hand, if the notification refers to a flow that is not part of any aggregate then the controller must update the information regarding the active traffic flows of the network by removing the entry that corresponds to this specific traffic flow.

```
P ← received packet
if (P ≠ IP packet)
    pass P to forwarding module
else
    route ← extract route for P
    code ← extract DSCP from P
    while (true)
        update the flow entry database
        if route can be aggregated with existing flows
            aggregate(route, code)
            r ←crop(r)
            continue
        flag ← check if the new flow can form a new
aggregate with other un-aggregated flows
        if (flag==true)
            aggregate
            r ←crop(r)
            continue
        else
            stop
```

Fig. 2. Aggregation algorithm: forwarding instruction request

```
f ← extract the related flow
if (f is an aggregate)
    remove the aggregate from the aggregates database
    update the VLAN pool if necessary
else
    flag ← check if the flow has been aggregated
    if (flag==true)
        stop
    else
        remove the flow from the flows database
```

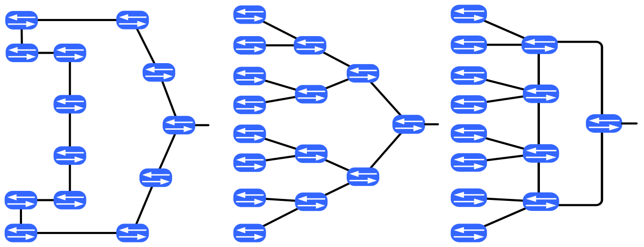
Fig. 3. Aggregation algorithm: flow removal notification

IV. Results

A. Specifying the experiments

In order to assess the performance of the traffic aggregation algorithm, a number of network topologies and traffic patterns were simulated using Mininet [11] and the algorithm was implemented for the Floodlight [12] (FL) SDN controller. In order to quantify the performance of the aggregation algorithm, the same set of experiments were repeated with and without traffic aggregation. The performance metric that was monitored was the total number of OF flow entries within the forwarding devices. Three distinct topologies were used in total, and a set of different traffic patterns was applied in each topology. The topologies that were used (Figure 4) are a ring, a tree and a ring of trees.

These topologies were selected because they represent those used in real-life mobile backhaul topologies [10]. The sets of different traffic patterns were designed to introduce varying levels of traffic flow numbers in the network, so the scaling of the algorithm could also be assessed. To generate these traffic flows, a number of end-users was connected to each node in the network and generated a number of unique ICMP ping traffic flows, all destined to the exit node of the corresponding topology. To achieve a balance between the QoS requirements of these sets of traffic flows, each end-user generated traffic flows with varying DSCP values associated with them. The varying levels of traffic flows were achieved by changing either the number of end-users connected to each node or by changing the number of unique traffic flows that each end-user generated.



a) Ring topology b) Tree topology c) Ring of trees topology

Fig. 4. Test topologies

B. Experimental setup and procedure

The following setup was used to conduct the experiments needed to measure the performance of the aggregation service. Two virtual machines (VM) were used, one running the FL SDN controller and one running the simulated network using Mininet. The reason that the controller and the network were in different VMs was to better simulate a real world scenario, in which the controller and the network itself are separated. This separation into different VMs also minimizes any performance interference between the controller and the network, as the two VMs have dedicated resources (CPU and RAM) allocated to them.

In order to ensure the validity of the collected results the following procedure was used during all the different experiments: (1) Each experiment was defined by the network topology in use, the number of hosts in the network, the traffic pattern that these hosts generated, and the setup of the SDN controller (aggregation enabled /disabled). (2) Each experiment was repeated until the mean value of the monitored performance criteria (total number of flow entries in the network nodes) remained relatively stable. This process ensured that any results that deviated from the norm were not drastically affecting the final results.

C. Results analysis

The results presented in this section quantify the performance of the aggregation service. To do so, the following graph shows the correlation between the number of unique traffic flows in the network (horizontal axis in Figures 5 and 6) and the percentage of reduction in the number of OF entries that the aggregation service has managed to offer (vertical axis in Figures 5 and 6). To collect each data point presented in these figures, the number of OF entries in the network, with and without traffic aggregation, was monitored and then the percentage of reduction was calculated using these two values.

From Figure 5 it is clear that the algorithm manages a substantial reduction in the total amount of traffic flow entries, which ranges between 16% and 48% approximately, depending on the situation at hand.

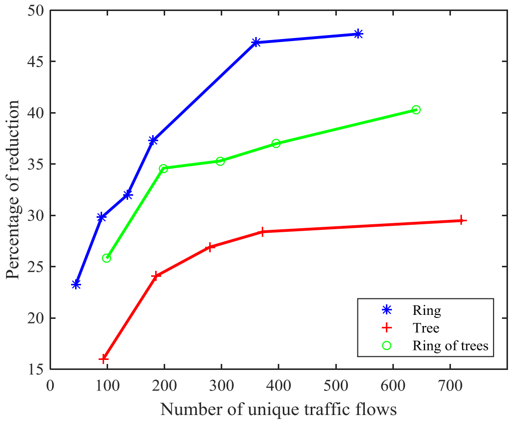


Fig. 5. Performance results

Regarding the performance of the algorithm in the different topologies there are two main conclusions. It is clear that the tree topology is the worst candidate for enforcing traffic aggregation. This behavior can be based on the fact that tree topologies have a high path separation, which means that it is harder for any two (or more) flows in the network to share a path, especially when compared with ring based topologies. Regarding the performance of the ring and ring of trees topologies, the results collected from the experiments cannot be used to conclude safely as to which is more suited for traffic aggregation. Finally, it is important to note that during all the different repetitions of the experiments the standard deviation of the number of monitored flows was trivial when compared with the magnitude of the monitored value. This fact further enhances the benefits of the implemented algorithm, as it means that the aggregation process has a reliable performance and behavior.

V. CONCLUSION

This paper proposed a solution that solves a burning issue regarding the OF-based SDN architecture. Dynamically aggregating the user traffic flows into bigger flows reduces the number of OF rules in the devices with up to 48%, depending on the topology and the traffic patterns. This makes the flow tables smaller thus the devices will eventually be able to accommodate more traffic. Further, it allows a wider adoption of cheap, commodity OF forwarding devices in a variety of network scenarios such as enterprises, data centers, campus networks, backhaul networks, etc.

Apart from the noticeable benefit in reducing the flow table size, the aggregation service impacts in a positive way the management of traffic. Because of the aggregation, it becomes easier to perform traffic engineering in the network since the traffic engineering service can reason in terms of aggregates, which are fewer in number and also created based on QoS classes. Moreover, it is easier to reroute the traffic flows in the network since there are fewer OF rules that have to be changed in the process.

Because the solution is based on simple encapsulation of traffic flows, it is easy to integrate it in existing network deployments since the devices already support the various encapsulations suggested herein (VLAN, MPLS).

Finally, some challenges that may arise for the aggregation service and that need further study are: (1) the integration of OF in environments where the aggregation service could be very useful, such as mobile backhaul networks, (2) the possibility to have several layers of encapsulation in order to carry already tagged traffic or to support overlapping addresses, and (3) reducing the encapsulation overhead by using some more innovative encapsulation mechanism.

In the current implementation the aggregation service does not perform traffic rerouting when creating the traffic aggregates. So each traffic flow will follow the same path before and after being aggregated. Another approach would be to perform traffic rerouting by inserting a traffic flow to an aggregate that follows a different path through the network. This approach inevitably increases the complexity of the aggregation process. But since the SDN controller has a global view of the network at all times, this process can increase the performance of the aggregation service, as more efficient traffic aggregates will be created.

ACKNOWLEDGMENT

This work was partially sponsored by the 7th Framework Programme for Research of the European Commission COSIGN project, under grant number 619572 – COSIGN.

REFERENCES

[1] Open Networking Foundation, "OpenFlow Switch Specification, version 1.3.0", June 2012.

[2] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization", Proceedings of IEEE INFOCOM, Toronto, ON, 2014.

[3] R. Narayanan, S. Kotha, G. Lin, S. Rizvi, W. Javed, H. Khan, and S. A. Khayam, "Macroflows and Microflows: Enabling Rapid Network Innovation through a Split SDN Data Plane", Proceedings of European Workshop on Software Defined Networking, Darmstadt, 2012.

[4] B. Stephens, A. Cox, W. Felter, C. Dixon, J. Carter, "PAST: Scalable Ethernet for Data Centers", Proceedings of CoNEXT, Nice, France, 2012.

[5] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-Based Server Load Balancing GoneWild", Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Boston, MA, 2011.

[6] S. Das, Y. Yiakoumis, G. Parulkar, N. McKeown, P. Singh, D. Getachew, P. D. Desai, "Application-Aware Aggregation and Traffic Engineering in a Converged Packet-Circuit Network", Proceedings of Optical Fiber Communication Conference and Exposition (OFC/NFOEC), Los Angeles, CA, US, 2011.

[7] B. Leng, L. Huang, X. Wang, H. Xu and Y. Zhang, "A Mechanism for Reducing Flow Tables in Software Defined Network", Proceedings of IEEE ICC Next Generation Networking Symposium, London, 2015.

[8] S. Luo, H. Yu and L. Li, "Fast incremental flow table aggregation in SDN", Proceedings of 23rd International Conference on Computer Communication and Networks (ICCCN), Shanghai, 2014.

[9] J. Costa-Requena, "SDN integration in LTE mobile backhaul networks", International Conference on Information Networking (ICOIN), Phuket, 2014

[10] Ceragon, "Wireless Backhaul Topologie: Analyzing Backhaul Topology Strategies", August 2010.

[11] Mininet, www.mininet.org, accessed October 2015.

[12] Floodlight, www.projectfloodlight.org/, accessed October 2015.