POLITECNICO DI TORINO
Repository ISTITUZIONALE

Inter-controller Traffic in ONOS Clusters for SDN Networks

Original
Inter-controller Traffic in ONOS Clusters for SDN Networks / Muqaddas, A.S., Bianco, A., Giaccone, P., Maier, G.. - ELETTRONICO. - (2016). (IEEE International Conference on Communications (ICC) Kuala Lumpur (Malaysia) May 2016) [10.1109/ICC.2016.7511034].

Availability:
This version is available at: 11583/2631909 since: 2016-09-17T07:33:15Z

Publisher:
IEEE

Published
DOI:10.1109/ICC.2016.7511034

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 November 2026

Inter-controller Traffic in ONOS Clusters for SDN Networks

Abubakar Siddique Muqaddas[†], Andrea Bianco[†], Paolo Giaccone[†], Guido Maier[‡]
[†] Dip. di Elettronica e Telecomunicazioni, Politecnico di Torino, Italy
[‡]Dip. di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Italy
E-mails:{abubakar.muqaddas,andrea.bianco,paolo.giaccone}@polito.it, guido.maier@polimi.it

***Abstract*—In distributed SDN architectures, the network is controlled by a cluster of multiple controllers. This distributed approach permits to meet the scalability and reliability requirements of large operational networks. Despite that, a logical centralized view of the network state should be guaranteed, enabling the simple development of network applications. Achieving a consistent network state requires a consensus protocol, which generates control traffic among the controllers whose timely delivery is crucial for network performance.**

We focus on the state-of-art ONOS controller, designed to scale to large networks, based on a cluster of self-coordinating controllers, and concentrate on the inter-controller control traffic. Based on real traffic measurements, we develop a model to quantify the traffic exchanged among the controllers, which depends on the topology of the controlled network. This model is useful to design and dimension the control network interconnecting the controllers.

I. Introduction

The standard centralized approach for SDN is based on a single controller managing all network devices. Even if this simplifies the network management and the development of network applications, it poses severe limitations to network scalability and reliability. Indeed, a single centralized controller is a single point of failure. Moreover, a single controller may not be able to handle a large number of devices, because the communication load and the processing overhead for the controller increases with device number. Finally, in very large networks (as in WANs), devices can be physically very far from the controller and, due to the propagation delays, flow modifications in network devices can experience large latency.

Distributed SDN controllers face with all the above impairments. Multiple instances of the controller manage the whole network, which is divided into different domains, each of them under the control of one controller instance. Distribution of the controller functions over multiple physical servers improves the robustness of the control plane, by providing backup control resources for each network node. Furthermore, large networks can be handled, because the device control is distributed among the controllers and the processing load can be balanced. Finally, being the control servers also geographically distributed across the network area, they can reduce the device-to-controller delay, thus improving the controller reactivity as perceived by the network node.

However, a logical centralized view of the network state has to be guaranteed also with controller distribution, to ease the development of advanced network applications. This transparent behavior for the network operator comes at the cost of keeping all the shared data structures synchronized among the controllers through some consensus protocol. For example, the same network topology must be known at each controller to take correct routing decisions. However, since each controller is responsible of a subset of devices, it is of paramount importance to distribute any topology update in a timely fashion to avoid routing misbehaviors (e.g. loops), as highlighted in [1].

In large networks, the control plane distributed among the controllers is implemented in-band, without the possibility of relaying on an out-of-band high-performance network as in the data center scenario [2]. This poses technical challenges in designing the control network, that not only interconnects devices to controllers, but also supports the communication between controllers. Due to the complexity of the adopted consensus protocols, the reactivity of the controllers as perceived by the devices depends also on the bandwidth and delays experienced in inter-controller communications.

In our work we focus on the control traffic exchanged among the controllers, which is often neglected in the literature. We consider the state-of-art ONOS controller [3], which is an open-source project developed by ON.Lab [4] and supported by a large community of network operators and vendors. Differently from the well-known OpenDaylight project [5], ONOS has been designed specifically to cope with reliability and scalability issues arising in large ISP/WAN networks. It natively supports a distributed version of the controller, running on a cluster of control servers.

A. Our contributions

We run an experimental testbed mastered by a cluster of ONOS controllers and evaluate the amount of traffic exchanged among the controllers. We vary the topology (in terms of nodes and links) of the network under control to develop a model,

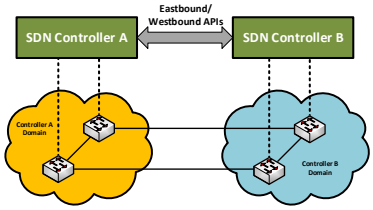

Fig. 1: Distributed SDN architecture with two controllers

fitting the experimental results, able to estimate the required bandwidth for the inter-controller communications, given an *arbitrary* network topology and any partition into different controller domains. The model is of paramount importance for the proper design and dimensioning of the network supporting the control plane in cluster of ONOS controllers.

B. Organization of the paper

Sec. II introduces the general architecture of distributed SDN controllers and describes the two main consistency models adopted to synchronize the data structures. In Sec. III we concentrate on the specific distributed architecture of ONOS and describe the two main protocols to achieve the consensus on the data structures. In Sec. IV we present the experimental evaluation of the inter-controller traffic, and we show the results for clusters of 2 and 3 ONOS controllers. These values are used to devise and fit the model to estimate the bandwidth for any network topology and any domain partition. The main results will be summarized in two claims, Property 1 and 2. Finally in Sec. V we draw our conclusions.

II. Consistency in Distributed SDN controllers

To coordinate the behavior of the controllers, some control traffic, denoted as *inter-controller traffic*, must be exchanged among the controllers, through their so called east-west interfaces. This traffic includes information related to: network topology, controller reachability information (through heartbeat messages), consistency protocols (described in the following) and flow setup coordination [6]. The east-west traffic grows always with the size of the network as well as with the number of controllers.

A. CAP theorem

Consistency of shared data in distributed systems is a well known and deeply investigated property. This property is achieved with quite complex protocols and algorithms [7]. The consistency dilemma is explained thoroughly using the famous CAP theorem [8] which states the impossibility of enjoying the following three properties at the same time: Consistency, i.e. all the data reads access the latest written version of the data; Availability, i.e. all data are accessible and can be updated; Partition, i.e. the system is tolerant to node partitions.

Even if the proof of this theorem is complex, a convincing scenario to understand this property is a storage system with the data replicated locally in two servers connected through a communication link. If availability and consistency are required at the same time (CA case), i.e. each server should be able to update the local data and access the most recent version of it, network partitions are not allowed, since the two servers must be able to communicate always an update to the other. Similarly, if availability and tolerance to partitions is required (AP case), i.e. each server should be able to update the local copy of the data, then consistency cannot be anymore guaranteed when partitions occur. Finally, if consistency and tolerance to partitions is required (CP case), i.e. the servers must access the most recent version of the data also in the case of partitions, availability cannot be guaranteed since each server cannot update the local copy in case of partitions. Depending on the pair of required guarantees (CA, AP or CP) in a distributed system, many consistency protocols and algorithms have been devised.

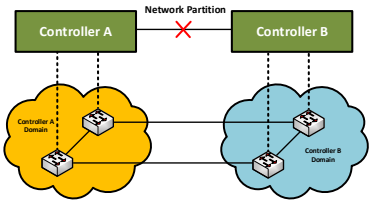

Fig. 2: Network partition in the control plane between two SDN controllers

In a distributed SDN scenario, consistency means that all the controllers view the same network state, e.g. have the same local copy of the network topology and of the node/link availability state. If the controllers have inconsistent view, the network policies could not run correctly and can lead to potential routing loops or packet drops in the network. For example in Figure 2, if the east-west interface between controllers is down, it results in control network partition. In this case if there is a change in topology in controller B's domain, then it will not be propagated to controller A. Consequently controller A could take routing decisions based on the old topology of the network in controller B's domain that could lead to unexpected behaviors. In the following section, we will discuss different consistency models that are currently adopted in SDN networks.

B. Consistency models in SDN networks

In distributed systems, many consistency models can be defined. We concentrate here on just two of them, which have a direct application in SDN networks.

1) Eventual Consistency model: This model provides a weak form consistency, in sense that data update on a certain controller will be *eventually* updated on all the nodes. This implies that for some time some nodes may read values different from the actual updated ones; but after some time, all the nodes will have the updated values, given that they are able to communicate. This model is employed in systems which require high availability. The anti-entropy protocol implemented in ONOS and described in Sec. III-A, supports this consistency model.

2) Strong Consistency model: This model ensures that each controller reads always the most updated version of a data. If certain data are not yet updated to all (or most of) the controllers, then they are not allowed to be read, thereby giving availability less priority in favor of consistency. The RAFT consensus protocol, implemented in ONOS and described in Sec. III-A, supports this consistency model.

III. Distributed ONOS

We now focus on the specific distributed architecture of ONOS controller, which allows to achieve a large scalability and availability, since a distributed cluster of controllers can coordinate to provide resilience and fault-tolerance, which is required if any of the controllers fails. Each controller in

the cluster is responsible of managing the switches under its domain, and updating their state on the distributed data stores. Each network device (i.e. a switch) can connect to multiple ONOS controllers, for reliability reason, but only one will be its master with full control on it in terms of read/write capabilities on the switch forwarding tables. Anytime a cluster of controllers is setup, each controller interacts with all the other controllers, thus the controllers are always logically connected in a full mesh in a peer-to-peer fashion, using a specific TCP port (9876) for their interaction. The controllers send and accept keep-alive messages to/from other controllers to monitor the cluster members.

A. Distributed stores and consistency protocols

The *stores* implement the distributed databases in ONOS. The main ones are the following:

- *Mastership store* which keeps the mapping between each switch to its master and it is managed by a strongly consistent protocol, using RAFT consensus algorithm, described below.
- *Network topology store* which describes the network topology in terms of links, switches and hosts; consistency is achieved using an eventually consistent protocol called anti-entropy, described below.

All the updates on the distributed stores are tracked using logical timestamps, which allow to find out the most recent update in case of conflicts.

Two consistency protocols are implemented in ONOS to manage the distributed stores, each protocol tailored to a specific level of consistency to guarantee.

1) Anti-Entropy Protocol: It is used to manage the network topology store. It is based on a simple gossip algorithm in which each controller chooses at random another controller in the cluster every 5 seconds and then sends a message to compare the actual content of its store with the one available at the other controller. This synchronization message includes the information about the elements (switches, links and hosts) present in the topology, as well as the removed elements, known as the "tombstones". In the case of temporary network partitions, keeping tombstones minimizes the variation in the internal topology store, and thus the allocation/deallocation of internal data structures. After the synchronization messages are exchanged and the stores are updated based on the age of each entry (i.e. the most recent updates are at higher priority than the older ones), the two controllers become mutually consistent. This ensures that all the controllers achieve consensus on the network topology, according to an eventually consistent model.

2) RAFT Protocol: It is a recently proposed scheme [9] which provides strong consistency for the mastership store in ONOS. A RAFT implementation requires a cluster of nodes each having a database termed as the "log" which is replicated in all the nodes: each update is appended to this shared data structure. The consistency is coordinated by a leader node in the cluster, which is responsible for receiving update requests from all the other nodes and then relaying log updates to the other nodes. Once the majority of the followers have acknowledged the update, this is actually committed to the log. In the case of network partitions, only the side with the majority of the nodes will be able to update the log, thus avoiding contemporary and conflicting updates in two different network partitions. In ONOS, for scalability issues, multiple instances of RAFT algorithm run simultaneously, each of them responsible of the synchronization of a subset of data. This implies that the stores are actually partitioned into different parts, each of them managed by a different RAFT instance.

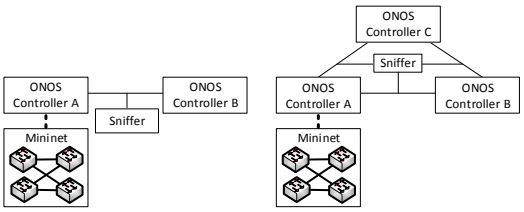

Fig. 3: Experimental testbed for 2 and 3 ONOS controllers

IV. Experimental results and model validation

We present here the approach used for the experimentation, aimed at evaluating the traffic exchanged among the controllers, based on any network topology and any partition of the network into controller domains. We consider a time-variant topology in which the number of active switches and active edges changes with the time; we expect that the inter-controller traffic will change due to the required modifications in the topology store as well as in the mastership store.

A. Methodology

We have considered two main scenarios, the first one with 2 controllers and the second one with 3 controllers belonging to the same cluster. The test setup for a 2-controller scenario is shown in Fig. 3 and is based on a standalone Ubuntu 14.10 server machine with ONOS 1.2 instances as distributed controllers installed in Linux Containers (LXC). The use of LXCs was chosen for its lightness and for the lack of "background" traffic, which allows to identify easily all the traffic generated by each instance of the controller. We denote with A and B the two instances of the controller, running on the same controller cluster. In the 3-controller scenario, the configuration is shown in Fig. 3 with the third controller denoted as C. In ONOS, the controllers are always logically connected in a full mesh. To emulate the network controlled through OpenFlow protocol, we adopted Mininet-2.1.0. Let S be the number of switches in the emulated network and L be the corresponding number of bidirectional links. Some test network topologies have been adopted for our investigation to highlight the individual contribution of each network element (switch or link). In the *isolated topology* we have S isolated switches without links ($L = 0$). In the *linear topology*, S switches are connected linearly ($L = S-1$). This methodology allowed us to obtain general results holding for *any* topology.

For the experiments, each topology is configured in Mininet; then the controller is connected to Mininet by specifying the target controller to connect to at each switch in the Mininet network. In the case when the topology is removed from the controllers, we issue the ONOS `wipe-out` command at each controller instance, which removes all the

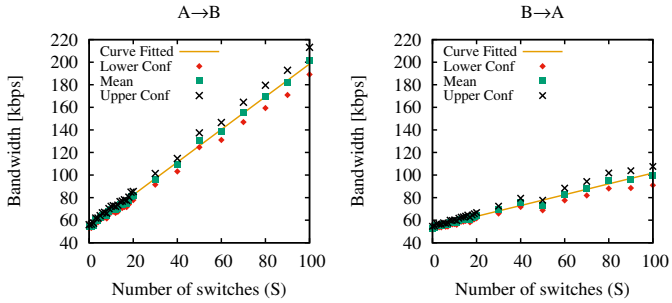

Fig. 4: Isolated topology added to controller A in the scenario with 2 controllers

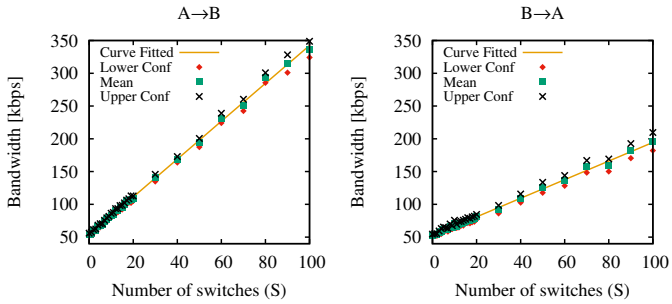

Fig. 5: Linear topology added to controller A in the scenario with 2 controllers

switches, links and hosts. To repeat an experiment, the topology is removed from each controller and the LXC containers are rebooted so that no tombstones persist in the inter-controller traffic.

Following the configuration in Fig. 3, we run Wireshark as a sniffer to capture the inter-controller traffic between any pair of controllers by capturing all the TCP traffic on the interface of a controller towards the other controller(s). ONOS uses port 9876 for the inter-controller communications, thus it is simple to identify such traffic. The total inter-controller traffic is sampled every 0.1 s to compute the consumed bandwidth. The bandwidth samples are averaged through a sliding window of 10 s. We start to discuss the results obtained for a simple topology, in order to highlight the transient and steady states we observed in our experiments. Starting from Sec. IV-B all the bandwidth results are measured in steady state. We repeated the experiments 20 times and computed the 95% confidence intervals.

B. Scenario with 2 controllers

We investigate the traffic exchanged by A and B for different sizes of the topology under the control of A. Fig. 4 shows the bandwidth from A to B and vice versa, when an isolated topology is added to controller A. We show also the confidence intervals and one linear curve fitting the experimental measurements. Similarly, Fig. 5 shows the bandwidth when a linear topology is added to controller A. Both graphs show that the bandwidth is increasing linearly in both communication directions. This is coherent with the linear growth of the internal data structures, based on hash tables. Moreover, the bandwidth for $A \rightarrow B$ is larger than $B \rightarrow A$. If we consider that the topology store is distributed with the anti-entropy protocol, we should expect a symmetric behavior. Instead, the mastership store is managed centrally by the leader and thus we can expect an asymmetric behavior due to the central role of the leader.

TABLE I: Notation for the bandwidth in the scenario with two controllers

Symbol	Meaning
B, (B^I, B^L)	generic unidirectional bandwidth (in isolated or linear topology)
b^0	generic zero bandwidth
b^s	average unidirectional bandwidth per switch
b^l	average unidirectional bandwidth per intra-domain link
b^d	average unidirectional bandwidth per inter-domain link
B^I_{AB}, B^I_{BA}	bandwidth ($A \rightarrow B$, $B \rightarrow A$) in isolated topology
B^L_{AB}, B^L_{BA}	bandwidth ($A \rightarrow B$, $B \rightarrow A$) in linear topology
b^s_{AB}, b^s_{BA}	average bandwidth $A \rightarrow B$, $B \rightarrow A$ per switch
b^l_{AB}, b^l_{BA},	average bandwidth $A \rightarrow B$, $B \rightarrow A$ per link

Due to the internal data structures, whose memory occupancy grows linearly with the number of elements (nodes and links), we can assume that the exchanged traffic B in each direction is proportional to the size of the topology store:

$$B = S \times b^s + L \times b^l + b^0$$

where we used the notation in Table I. Now, considering the two different scenarios in our experiments, we can write the following system of equations, with the topology added only to controller A:

$$\begin{cases} B^L_{AB} = S \times b^s_{AB} + (S-1) \times b^l_{AB} + b^0 \\ B^L_{BA} = S \times b^s_{BA} + (S-1) \times b^l_{BA} + b^0 \\ B^I_{AB} = S \times b^s_{AB} + b^0 \\ B^I_{BA} = S \times b^s_{BA} + b^0 \end{cases}$$

that can be solved by considering the linear interpolating curve fitting the experimental data.

Note that we used always the same zero bandwidth value $b^0 = 53$ kbps (obtained with 2% accuracy at 95% confidence level) for the both directions, given our measurements. Thus, we obtained

$$b^l_{AB} = 1.45 \text{ kbps} \qquad b^l_{BA} = 0.94 \text{ kbps} \tag{1}$$
$$b^s_{AB} = 1.45 \text{ kbps} \qquad b^s_{BA} = 0.48 \text{ kbps} \tag{2}$$

So far the considered topologies have been configured under the same controller domain. In order to extend our model to any topology, arbitrarily partitioned among the two controller domains, we need to understand the effect of inter-domain links, i.e. connecting one switch in one domain with another in the other domain. We considered the star topology in Fig. 6, in which we varied the number of switches and thus of the inter-domain links. Now the observed bandwidth in one direction should be obtained by summing the following contributions: B^I for 1 switch to model the switch in A domain; B^I for $S-1$ switches to model the $S-1$ switch in B domain; $S-1$ times the average bandwidth per inter-domain link b^d. Using the same methodology before, and exploiting the estimated values obtained so far, we were able to estimate that the average bandwidth per inter-domain link is

$$b^d = 1.55 \text{ kbps} \tag{3}$$

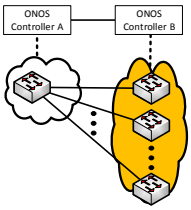

Fig. 6: Scenario with network links between two controller domains

TABLE II: Notation for the network topology in the scenario with 2 controllers

Symbol	Meaning
S_A	number of switches in controller A domain
S_B	number of switches in controller B domain
L_A	number of intra-domain links in controller A domain
L_B	number of intra-domain links in controller B domain
L_{AB}	number of inter-domain links

By combining the results so far and the estimated bandwidths in (1), (2) and (3), we can claim the following:

Property 1: In a scenario with 2 controllers A and B, managing a generic network topology arbitrarily partitioned among the two controllers, the exchanged bandwidth can be evaluated as follows:

$$B_{AB} = 53 + 1.45 \times S_A + 1.45 \times L_A + 0.48 \times S_B + 0.94 \times L_B + 1.55 \times L_{AB} \quad \text{[kbps]}$$
$$B_{BA} = 53 + 1.45 \times S_B + 1.45 \times L_B + 0.48 \times S_A + 0.94 \times L_A + 1.55 \times L_{AB} \quad \text{[kbps]}$$

where we used the notation in Table II

We were able to validate the above formulas by considering multiple scenarios, including full mesh topology, ring topology, irregular topology. All the experimental results were always compatible with the model prediction of Property 1 within 95% confidence interval.

C. Scenario with 3 controllers

The methodology adopted in the previous scenario is now extended to the 3 controllers scenario using the configuration shown in Fig. 3.

We start by adding the topology to controller A. For symmetry, the bandwidth for $A \to B$ and $A \to C$ is same and termed as $A \to B, C$; as no topology is added to controllers B and C, hence $B \to A$, $B \to C$, $C \to A$ and $C \to B$ are same and termed as $B, C \to A$. Fig. 7 shows the bandwidth from A to B and C and vice versa, when an isolated topology is added to controller A.

Fig. 8 shows the bandwidth when a linear topology is added. As compared to Fig. 4 and 5, the bandwidth values in the 3-controller case are lower than the 2-controller case. This is due to the anti-entropy protocol: periodically, each controller randomly selects another controller to synchronize the network topology. Say the synchronization rate for each controller is λ. Thus the average contribution of this process on each link is $3\lambda/6 = \lambda/2$, since 6 possible links are present with 3 controllers. Instead, in the case of 2 controllers, the average contribution was $2\lambda/2 = \lambda$. Thus, a reduction of a factor 2 in the bandwidth due to the enti-entropy is expected. In the figure, the reduction is much lower, due to the low impact of this consistency protocol with respect to RAFT. Globally, the bandwidth exchanged B in each direction is still proportional to the size of the topology store. Considering that the topology is only added to controller A, then the following system of equations can be written with the notation in Table III:

$$\begin{cases} B^L_{A\to BC} = S \times b^s_{A\to BC} + (S-1) \times b^l_{A\to BC} + b^0 \\ B^L_{BC\to A} = S \times b^s_{BC\to A} + (S-1) \times b^l_{BC\to A} + b^0 \\ B^I_{A\to BC} = S \times b^s_{A\to BC} + b^0 \\ B^I_{BC\to A} = S \times b^s_{BC\to A} + b^0 \end{cases}$$

which can be solved by considering the linear interpolating curve fitting the experimental data. The zero bandwidth between any two controllers here is $b^0 = 43$ kbps (obtained with 5.5% accuracy at 95% confidence level). Thus, we computed

$$b^l_{A\to BC} = 0.93 \text{ kbps} \qquad b^l_{BC\to A} = 0.53 \text{ kbps} \quad (4)$$
$$b^s_{A\to BC} = 1.12 \text{ kbps} \qquad b^s_{BC\to A} = 0.35 \text{ kbps} \quad (5)$$

To extend model to any topology, arbitrarily partitioned among the two controller domains, the star topology in Fig. 6 was considered albeit with 3 controllers with no switch added to controller C, in which we varied the number of switches and thus the inter-domain links. In this scenario, the bandwidth originating from each controller to the other two controllers will be different, since different number of switches are added to each controller. Hence, the three bandwidths in consideration are: $A \to BC$, $B \to AC$ and $C \to AB$. Furthermore,

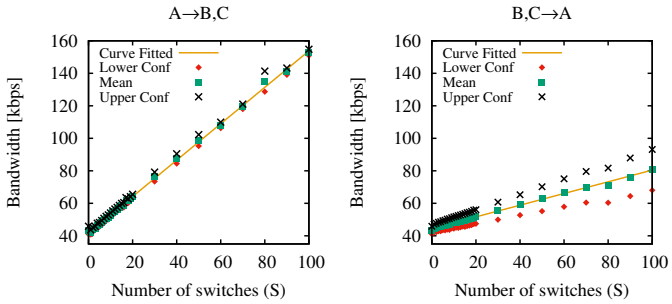

Fig. 7: Isolated topology added to controller A in the scenario with 3 controllers

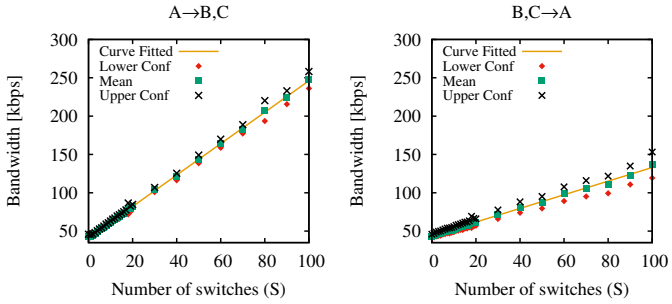

Fig. 8: Linear topology added to controller A in the scenario with 3 controllers

TABLE III: Notation for the bandwidth in the scenario with 3 controllers

Symbol	Meaning
B, (B^I, B^L)	generic unidirectional bandwidth (in isolated or linear topology)
b^0	generic zero bandwidth
b^s	average unidirectional bandwidth per switch
b^l	average unidirectional bandwidth per intra-domain link
b^d	average unidirectional bandwidth per inter-domain link shared also by target controller
b^e	average unidirectional bandwidth per inter-domain link external to target controller
$B^I_{A\to BC}$	bandwidth $A \to B$ and $A \to C$ in isolated topology
$B^I_{BC\to A}$	bandwidth $B \to A$, $C \to A$, $B \to C$ and $C \to B$ in isolated topology
$B^L_{A\to BC}$	bandwidth $A \to B$ and $A \to C$ in linear topology
$B^L_{BC\to A}$	bandwidth $B \to A$, $C \to A$, $B \to C$ and $C \to B$ in linear topology
$b^s_{A\to BC}$	average bandwidth $A \to B$ and $A \to C$ per switch
$b^s_{BC\to A}$	average bandwidth $B \to A$, $C \to A$, $B \to C$ and $C \to B$ per switch
$b^l_{A\to BC}$	average bandwidth $A \to B$ and $A \to C$ per link
$b^l_{BC\to A}$	average bandwidth $B \to A$, $C \to A$, $B \to C$ and $C \to B$ per link

the average unidirectional bandwidth per inter-domain link in this case will be b^d for controllers A and B but will be b^e for controller C, since the links are external to it but of inter-domain type. Now the observed bandwidth in one direction is obtained by summing the following contributions: B^I for 1 switch to model the switch in A domain; B^I for $S-1$ switches to model the $S-1$ switch in B domain; $S-1$ times the average bandwidth per inter-domain link b^d and $S-1$ times the average bandwidth per external inter-domain link b^e.

Using the same methodology before, and exploiting the estimated values obtained so far, we were able to estimate that the average bandwidth per inter-domain link as:

$$b^d = 0.87 \text{ kbps} \quad b^e = 0.7 \text{ kbps} \tag{6}$$

By combining the results so far and the estimated bandwidths in (4), (4) and (6), we can claim the following, by referring to Fig. 9:

Property 2: In a scenario with 3 controllers A, B and C, managing a generic network topology arbitrarily partitioned among the three controllers, the exchanged bandwidth can be evaluated as follows:

$$B_{A\to BC} = 43 + 1.12 \times S_A + 0.93 \times L_A + 0.35 \times (S_B + S_C) + 0.53 \times (L_B + L_C) + 0.87 \times (L_{AB} + L_{AC}) + 0.7 \times L_{BC} \text{ [kbps]}$$

$$B_{B\to AC} = 43 + 1.12 \times S_B + 0.93 \times L_B + 0.35 \times (S_A + S_C) + 0.53 \times (L_A + L_C) + 0.87 \times (L_{AB} + L_{BC}) + 0.7 \times L_{AC} \text{ [kbps]}$$

$$B_{C\to AB} = 43 + 1.12 \times S_C + 0.93 \times L_C + 0.35 \times (S_A + S_B) + 0.53 \times (L_A + L_B) + 0.87 \times (L_{AC} + L_{BC}) + 0.7 \times L_{AB} \text{ [kbps]}$$

where we used the notation in Table IV

V. Conclusions

We considered a distributed SDN architecture in which a cluster of SDN controllers manages all network devices. We focused our investigations on the traffic exchanged between the controllers, which is mainly due to the required consensus protocols running among controllers to reach a consistent view of the network state.

We considered an experimental testbed based on a cluster of 2 and 3 ONOS controllers and evaluated experimentally the inter-controller traffic under different network configurations. This allowed us to develop a model, empirically validated, to estimate the control traffic among controllers under any network topology and any partition of the switches in the controllers' domains. This permits to analyze how the inter-controller traffic scales with the size of the network under control, and can be used for the proper dimensioning of the control plane supporting the interaction among the controllers.

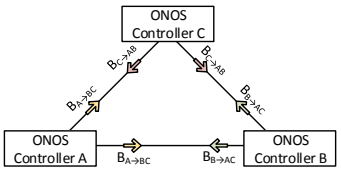

Fig. 9: Experimental testbed for 2 ONOS controllers

TABLE IV: Notation for the network topology in the scenario with 2 controllers

Symbol	Meaning
S_A	number of switches in controller A domain
S_B	number of switches in controller B domain
S_C	number of switches in controller C domain
L_A	number of intra-domain links in controller A domain
L_B	number of intra-domain links in controller B domain
L_C	number of intra-domain links in controller B domain
L_{AB}	number of inter domain links between A and B
L_{AC}	number of inter domain links between A and C
L_{BC}	number of inter domain links between B and C

References

[1] A. Panday, C. Scotty, A. Ghodsiy, T. Koponen, and S. Shenker, "CAP for networks," in *HotSDN*. ACM, 2013, pp. 91–96.

[2] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, Apr. 2014.

[3] "ONOS 1.2 Wiki," https://wiki.onosproject.org/.

[4] "On.Lab website," http://onlab.us/.

[5] "OpenDaylight website," https://www.opendaylight.org/.

[6] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[7] P. Bailis and A. Ghodsi, "Eventual consistency today: Limitations, extensions, and beyond," *Communications of the ACM*, vol. 56, no. 5, pp. 55–63, May 2013.

[8] E. Brewer, "CAP twelve years later: How the "rules" have changed," *IEEE Computer*, vol. 45, no. 2, pp. 2–13, March 2012.

[9] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annual Technical Conference*, 2014, pp. 305–320.