ICONA: a peer-to-peer approach for Software Defined Wide Area Networks using ONOS

Matteo Gerola*, Francesco Lucrezia†, Michele Santuari*, Elio Salvadori*
Pier Luigi Ventre‡, Stefano Salsano‡, Mauro Campanella§
*CREATE-NET, †Polytechnic University of Turin,‡CNIT / Univ. of Rome Tor Vergata, §Consortium GARR

***Abstract*—Several Internet Service Providers (ISP) are planning to innovate their infrastructures through a process of network softwarisation and programmability. The Software-Defined-Network (SDN) paradigm aims at improving the design, configuration, maintenance and service provisioning agility of the network through a centralised software control plane which is in charge of managing the entire system. This is easily achievable for local area networks, typical of data centres, where the benefits of having programmable access to the entire network is not restricted by latency. However, in Wide Area Networks, a centralised control plane limits the speed of responsiveness in reaction to time-constrained network events due to unavoidable latencies caused by physical distances. A logical step towards robustness in SDN is to distribute the load of the control plane between entities, each taking care of a portion of the entire geographical network and each providing an east-west communication interface to enable programmability of the entire network. Moreover, a key objective of an SDN control plane targeting an ISP networks is the east-west interface with external domains under the control of other providers. In this article we present ICONA (Inter Cluster Onos Network Application), a tool that has the objective of enabling programmable networks to span multiple clusters of controllers within either a single or multiple administrative domains. In particular, the paper describes the architecture behind ICONA and provides an initial evaluation obtained on a preliminary version of the tool, built on top of the cutting-edge network controller ONOS, Hummingbird release.**

***Index Terms*—Software Defined Networking, Open Networking Operating System (ONOS), WAN, multi administrative domains, east-west interface, BGP**

I. Introduction

The SDN control plane reliability, scalability and availability were among the major elements of attention expressed by Service and Cloud Providers. Existing deployments show that standard IP/MPLS networks natively offer fast recovery in case of failures. Their main limitation lies in the complexity of the distributed control plane, implemented in the forwarding devices. IP/MPLS networks fall short when it comes to designing and implementing new services that require changes to the distributed control protocols and service logic. The SDN architecture, that splits data and control planes, simplifies and speeds up the introduction of new services, by moving the intelligence and most of network state from the physical devices to a logically centralised Network Operating System (NOS), also known as controller, in charge of all the forwarding decisions. It is also clear, as described in the work of Heller et al. [5], that even if a single controller may suffice to guarantee round-trip latencies on the scale of a typical mesh restoration delays (200 msec), this is not enough for all network topologies. Furthermore, ensuring an adequate level of fault tolerance (i.e., avoiding excessive packet loss and session termination) can be guaranteed only if controllers are spaced apart in different locations of the network. To guarantee the proper level of redundancy in the control plane, several distributed NOS architectures have been proposed in the last years: ONIX [1], Kandoo [2], HyperFlow [3] to name a few. Mainly, these architectures fall into two categories: *(i)* hierarchy of controllers and *(ii)* peer-to-peer interconnections between controllers. While the former gives adequate scalability for resources under the control of the same domain, the latter offers more benefits in case of a multi administrative domain solution, removing a top-level entity, possibly managed by a third party, controlling the interconnections between networks belonging to different providers. One of the most promising solutions to properly deal with control plane robustness in SDN is ONOS (Open Networking Operating System) [4]. ONOS is a distributed NOS and it is supported by an active community of vendors and operators. In the ONOS architecture, a cluster of controllers shares a logically centralised network view: network resources are partitioned and controlled by different ONOS instances in the cluster. Resilience to faults is guaranteed by design, with automatic traffic rerouting in case of node or link failure. However, despite the distributed architecture, ONOS is designed to be placed in a single geographical location, because its distributed architecture requires negligible communication delays between instances.

Given this consideration, the authors engineered an open-source ONOS application called ICONA (Inter Cluster Onos Network Application). ICONA is designed to work in a single administrative WAN network scenario, increasing the robustness to network faults by redounding ONOS clusters in several geographical locations and decreasing event-to-response delays, as well as in a multi administrative domain scenario. To better support the latter use-case, ICONA is based on a peer-to-peer architecture, and implements configuration policies between clusters (i.e., domains belonging to different owners), that ensure the full control of services and events between domains.

The structure of the paper is as follows. Sect. II provides an overview of ONOS, while Sect. III presents the ICONA architecture. Some preliminary results are then discussed in Sect. IV and Sect. V discusses the state-of-the-art in the field.

2379-0369/17 $31.00 © 2017 IEEE
DOI 10.1109/EWSDN.2016.12

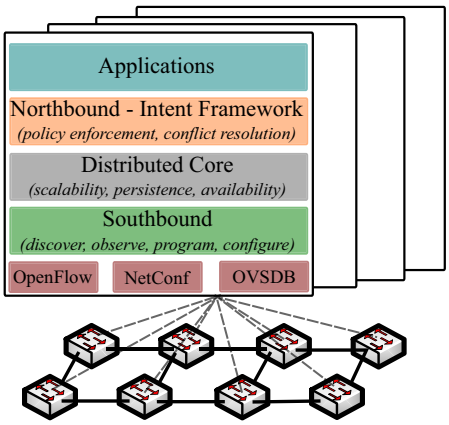


Fig. 1: ONOS distributed architecture

Finally, Sect. VI draws conclusions and indicates future works.

II. An overview of ONOS

The Open Network Operating System (ONOS) is a software defined networking (SDN) OS for Service Providers, that is targeting scalability, high availability, high performance and abstractions to make it easy to create apps and services. Created by ON.Lab it is an open-source joint community effort with substantial contribution from various partners including AT&T, NEC and Ericsson [5].

ONOS implements a distributed architecture in which multiple controller instances share multiple distributed data stores with different level of consistency. The entire data plane is managed simultaneously by the whole cluster. However, for each device a single controller acts as a master, while the others are ready to step in if a failure occurs. With these mechanisms in place, ONOS achieves scalability and resiliency. Figure 1 shows the ONOS internal architecture within a cluster of four instances. ONOS is based on software modules managed by the Apache Karaf suite [6], a set of java OSGi based runtime and applications. It provides a container into which various component can be deployed, installed, upgraded, started and stopped at runtime, without interfering other components. The southbound modules manage the physical topology, react to network events and program/configure the devices leveraging on different protocols. The distributed core is responsible to maintain coherent information, to elect the master controller for each network portion and to share information with the adjacent layers. In case of a failure in the data path (switch, link or port down), an ONOS instance becomes aware of the event through the southbound modules, computes alternative paths for all the traffic crossing the failed element, and notifies them to the distributed core; then, each master controller configures accordingly its portion of the network. The northbound subsystem offers an abstraction of the network and the interface for applications to interact and program the NOS. Finally, the Application layer offers a container in which third-party applications can be deployed. Applications on top of ONOS can benefit of the Intent Framework. An intent is an abstraction used by applications to specify their high-level desires in form of policies. The ONOS core accepts the intent specifications and translates them into actionable operations on the network environment. These actions are carried out by the intent installation process, such flow rules being installed on a switch, or optical lambdas (wavelengths) being reserved.

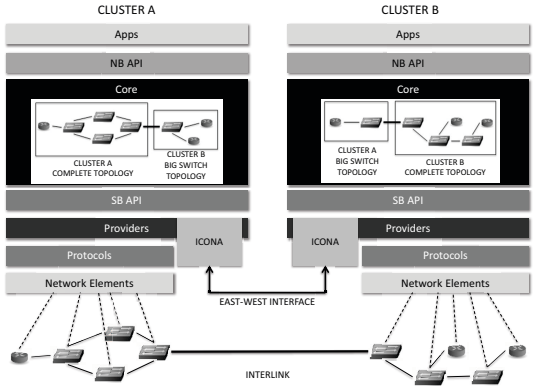
III. ICONA Architecture

ICONA is a new southbound ONOS provider, that offers an east-west interface and a powerful abstraction layer to allow a single ONOS cluster to be interconnected with several other clusters, both in the same and in different administrative domains. ICONA is completely transparent to the ONOS core systems and to other applications, thus offering the same functionalities of ONOS, but extended to a geographically distributed environment, including multiple administrative domains. From an application perspective, all the features offered by ONOS are then available in an multi administrative domain composed of several ICONA clusters.

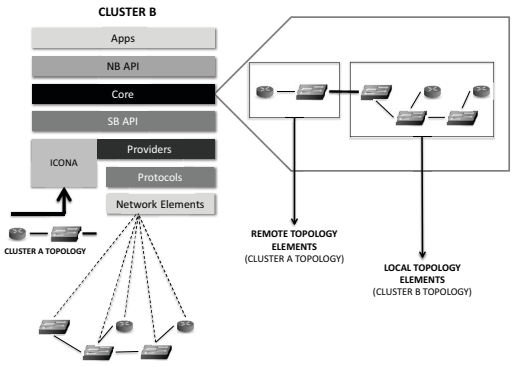
The main architectural goals of ICONA are to:

- *Enable east-west communication between ONOS clusters*. In a single-domain, this implies partitioning the Service Providers network into several geographical regions, each one managed by a different cluster of ONOS instances. The network architect can select the number of clusters and their geographical dimension depending on requirements (e.g., leveraging on some of the tools being suggested within the aforementioned work [5]), without losing any features offered by ONOS, neither worsening the system performance. In a multi-domain scenario, several ONOS clusters, belonging to different administrative domains, can exchange network services based on respective policies and network abstractions.
- *Provide an abstraction to ONOS*, able to: *(i)* abstract and communicate external topologies (i.e., devices, links and ports not directly managed by the local cluster), *(ii)* configure these external devices from local applications, leveraging on the Intent Framework and *(iii)* enforcing policies between clusters.

Clusters, policies and topology abstractions can be easily injected in ICONA through the ONOS configuration service. ICONA extrapolates the local topology from the ONOS core, abstracts it based on the configuration and finally exposes it to remote clusters; likewise, it receives the external topologies from the remote clusters and notifies them to ONOS. In case of a multi domains, this external topology is exposed as a single big switch. Moreover, it takes care of reporting relevant updates to the remote clusters about changes affecting the abstracted topology by listening to events reported by the ONOS subsystems (e.g., devices, links, ports and edge hosts). The east-west communication between clusters is not bounded to a single peer-to-peer mechanism, but it allows different implementations, leveraging on the ICONA Southbound Interface (ISBI).



(a) ICONA as a peering provider



(b) Topology exchange

Fig. 2: ICONA topology abstraction.

Figure 2(a) depicts two clusters, A and B, sharing their topologies through ICONA. In Figure 2(b), *cluster A* exposes its 4 switches topology as a single big switch, that is abstracted and communicated to the local ONOS core by the ICONA provider of *cluster B*.

The communication between clusters relies on the ISBI interface, that can be implemented by different mechanisms. To make ICONA as flexible as possible, its structure is vertically split in two logical layers:

- the ICONA Provider which contains the main logic and lays between the ONOS core and the ISBIs.
- multiple ISBI drivers, each one tied to a specific for a communication mechanism. For each remote cluster, it's possible to specify a different mechanism, thus allowing several ISBI implementations to be used simultaneously.

This architecture allows to integrate several communication mechanisms, just by implementing the ISBI logic, without any modifications in the ICONA provider.

A. ICONA Provider

The provider contains the main logic behind ICONA, and performs various functionalities. As a **Topology Manager** (see III-A1), it builds an abstraction of the local topology to be exposed to remote clusters. Currently ICONA supports two topology abstractions: *BigSwitch* (single switch representing the entire network with edge ports) and *FullMesh* (network topology in which there is a direct link between all pairs of edge nodes). While the former shrinks the entire topology in a single switch, the latter builds a full virtual mesh topology between all the edge switches (e.g. the ones with edge ports) of the network. The provider, after receiveing remote topologies from the southbound mechanisms, abstracts them to the local ONOS core, and reacts to network events which could reflect a change in the topology exposed to/from the remotes. As a **Service Manager** (see III-A2), it manages service requests coming from the local applications to remote clusters and vice versa. Finally, as a **Policy Manager** (see III-A3), it enforces configuration policies between clusters. The next sections detail the features offered by the three provider modules, depicted in Figure 3.

1) Topology Manager: The Topology Manager (TM) is responsible to *(i)* analyse the remote topologies and install them in ONOS with the relevant metric and annotations, and *(ii)* reacts to network events, both local and remote. Each TM shares with the other clusters an abstraction of the local topology, that may vary from cluster to cluster, based on the configuration. The topology is composed of:

- Inter-links (IL): links belonging to different ONOS clusters. Each IL is provided trough the ONOS configuration subsystem and it's tagged by some metrics, such as the link delay, available bandwidth and number of flows crossing the link.
- Virtual devices and intra-links: links within the local cluster/domain.
- End-points (EP): interconnection ports between the customer's gateway router/switch and the ONOS network.

2) Service Manager: The Service Manager (SM) is the ICONA component, that provides inter-cluster path computation whenever a northbound application, on-top of ONOS, requires connectivity between two or more EPs crossing multiple clusters. The SM intercepts an intent requests from the ONOS core targeting one or more virtual devices of the remote topology and sends it to the target remote cluster. The recipient translates the remote intent request into a local intent request and submits it to the ONOS Core. The translation consists in resolving the mapping between the abstract ingress and egress points of the original request into local ingress and egress points of the underlying network. The check of the feasibility of the intent installation is also performed, both in terms of policy and capabilities. If the request is accepted, the SM waits for the ONOS core to accomplish the task of installing the intent and then notifies the requester of the outcome.

3) Policy Manager: A multi administrative domain scenario is characterized by the presence of networks under control of different authorities. Usually, the mutual trust between these domains is limited to specific agreements, which identify a list of constraints to be applied at the edge of the network. To

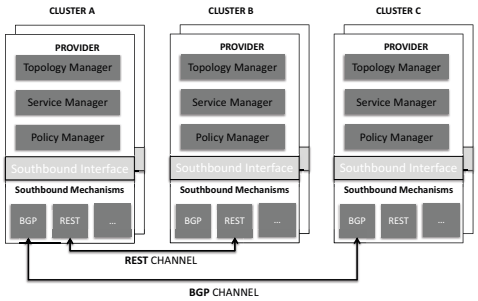


Fig. 3: High level view of the ICONA components.

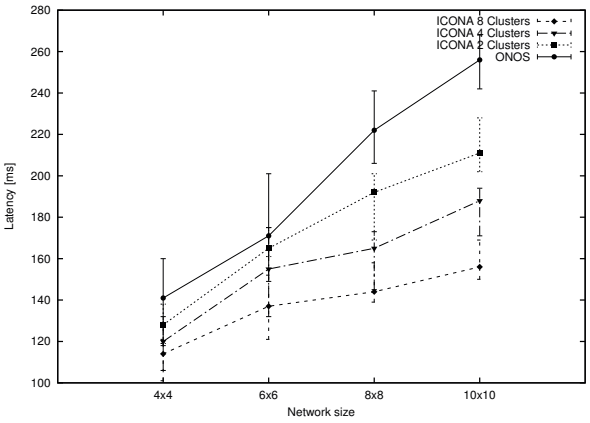


Fig. 4: Average, maximum and minimum latency to reroute 100 paths in case of link failure for ONOS and ICONA (2, 4 and 8 clusters)

support such use-case, ICONA is policy-oriented and enforces, through configuration, the compliance to those agreements. Currently ICONA allows to set at runtime several parameters, such as *(i)* the external peering clusters information, *(ii)* the topology abstraction exposed to each domain, *(iii)* the list of EPs, *(iv)* the type and number of intents installable by a remote cluster, *(v)* the ILs and their metrics (i.e., bandwidth, delay and type), *(vi)* the preferred path for specific classes of traffic (based on L2 and L3 fields). However, the authors are currently analysing the common design patterns of BGP policies, that are typically used by ISPs [7], to implement innovative policy mechanisms in the future releases.

B. ICONA Southbound Mechanisms

A southbound mechanism is an implementation of the communication system between clusters (Fig. 3). Basically, it's a software component in charge of translating the provider's requests into protocol-specific, network operations and the remote clusters messages into abstracted notifications via the ISBI. This component performs the exchange of the information with message encoding and decoding, and does not retain any system state except the status of the remote clusters. Currently ICONA supports two different mechanisms:

- BGP: an extension of the BGP protocol with a new Type-Length-Value (TLV) field to offload the communication system to an external router (e.g. Quagga based) and to use BGP as a pure transport protocol for data exchange.
- REST: a REST client/server peer-to-peer architecture has been implemented between clusters. The client is in charge of sending local topology elements and to request service installation, while the server is responsible to receive remote topology elements and service requests from the other clusters. We implemented a distributed architecture, in which every ICONA instance is responsible to interconnect the local cluster with a set of remote clusters and the communication is balanced among multiple endpoints providing resiliency.

IV. Evaluation

The purpose of the experimental tests described in this section is to compare ICONA with a standard ONOS setup, and evaluate the performances of the two solutions in an emulated environment. It is important to highlight that these evaluation has been achieved with a preliminary version of ICONA, working on an old ONOS version (Blackbird). For these reasons, the results presented in this section should not be considered as benchmark, but they can offer a comparison between the fully centralized solution versus the peer-to-peer architecture offered by ICONA.

The control plane is composed of several virtual machines, each configured to use 4 Intel Core i7-2600 CPUs @ 3.40GHz and 8GB of RAM. For the ONOS tests we used 8 instances, while with ICONA we created 2, 4 and 8 clusters, respectively with 8, 5 and 3 instances each. The data plane is emulated by Mininet [8] and Netem [9]: the former creates and manages the network based on OpenFlow 1.3, while the latter emulates the properties of wide area networks, intriducing variable delays, throughput and packet loss. Both solutions (ONOS and ICONA) have been tested on top of "grid" networks and of the GÉANT [10] topology.

A. Reaction to Network Events

1) Grid network: The first measured performance metric is the overall latency of the system for updating the network state in response to events; examples include rerouting traffic in response to link failure or moving traffic in response to congestion. To evaluate how the system performs when the forwarding plane scales-out, few standard grid topologies (from 4*4 to 10*10) have been chosen, with a fixed link delay of 5ms (one-way) and the latency needed to reroute a certain number of installed paths when an inter-cluster link fails is compared between ONOS and ICONA with various clustering settings.

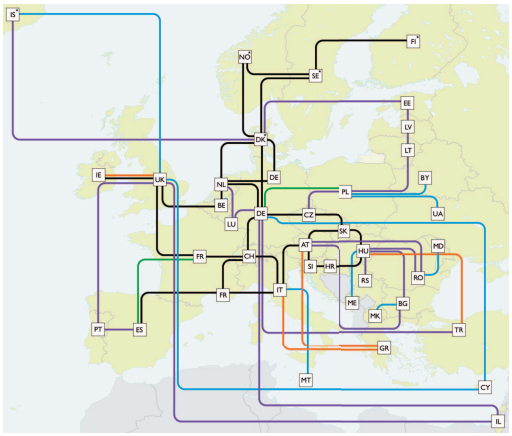


Fig. 5: GÉANT pan-European network

The total latency is defined as the amount of time that ONOS or ICONA requires to react to the failure. It is computed as the sum of: *(i)* the amount of time taken by the OpenFlow messages (`PORT_STATUS` and `FLOW_MOD`) to traverse the control network, *(ii)* the alternative path computation, *(iii)* the installation of new flows in the network devices and *(iv)* the deletion of the pre-existing flows. In particular, we have been running several simulations by installing 10^3 paths in the network and then causing failure of an inter-cluster link carrying al least 10^2 flows.

Figure 4 shows the latency (avg, min, max) required for the different case to execute the four tasks previously mentioned. Each test has been repeated 10^3 times. Despite the same mechanism used by ICONA to compute and install the new paths, the difference is mainly due to the following reasons: *(i)* each ICONA cluster is closer to the devices, thus reducing the amount of time required for OpenFlow messages to cross the control channel and *(ii)* the ICONA clusters are smaller, with fewer links and devices, thus decreasing the time used for computation and the overall numbers of flows to be installed and removed from the data plane.

2) GEANT network: The same metrics have been evaluated on the GÉANT topology (see Figure 5). Circuits have various one-way delays (from 10 to 50ms) and throughputs (from 1 to 100Gbps).

Control plane	Avg latency [ms]	Min latency [ms]	Max latency [ms]
ONOS	297	284	308
ICONA2	272	261	296
ICONA4	246	232	257
ICONA8	221	199	243

TABLE I: GÉANT network: average, maximum and minimum latency to reroute 100 paths in case of link failure for ONOS and ICONA (2, 4 and 8 clusters)

Table I depicts similar results as the previous test. While the GÉANT network is smaller than the grid topology, with 41 switches and 58 bi-directional links, the higher delay in the data plane adds an additional time before convergence to a stable state.

B. Startup Convergence Interval

This second experiment measures the overall amount of time required for both solutions to re-converge after a complete disconnection between the control and data planes. The tests have been performed over the GÉANT topology, and replicated 10^3 times. Table II shows the average, maximum and minimum values in seconds.

Control plane	Average Time [s]	Minimum Time [s]	Maximum Time [s]
ONOS	6,98	6,95	7,06
ICONA	6,96	6,88	7,02

TABLE II: Amount of time required to obtain the network convergence after disconnection for ONOS and ICONA

The result shows that ICONA and ONOS require comparable time intervals to return to a stable state, in case of a complete shutdown or a failure of the control plane.

V. Related Work

The logical centralization of the control plane, proposed in general by the SDN approach, requires a specific design to obtain adequate performance of the control network, scalability and fault tolerance. Most of the open source controllers currently available are more focused on functionalities rather than on scalability and fault tolerance. This section provides a review about distributed architectures for the SDN control plane, that address the scalability and fault tolerance issues. ONIX [1] provides an environment on top of which a distributed NOS can be implemented with a logically centralized view. The distributed Network Information Base (NIB) stores the state of network in the form of a graph; the platform is responsible for managing the replication and distribution of the NIB, the applications have to detect and resolve conflicts of network state. Scalability is provided through network partitioning and aggregation. Regarding the fault tolerance, the platform provides the basic functions, while the control logic implemented on top of ONIX needs to handle the failures. ONIX has been used in the B4 network, the private WAN [11] that inter-connects Google's data centers around the world. This high level design is similar to ICONA. ICONA however is not tailored to a specific use case, providing a reusable framework on top of which it is possible to build specific applications. The Kandoo [2] architecture addresses the scalability issue by creating an architecture with multiple controllers: the so-called root controller is logically centralized and maintains the global network state; the bottom layer is composed of local controllers in charge of managing a restricted number of switches. The Kandoo architecture does not focus on the distribution/replication of the root controller and on fault tolerance neither in the data plane nor in the control plane. HyperFlow [3] focuses on both scalability and fault tolerance. Each HyperFlow instance manages a group

of devices without losing the centralized network view. A control plane failure is managed by redirecting the switches to another HyperFlow instance. The applicability of such approach to WAN scenarios with large delays between the different HyperFlow instances is not considered. DISCO [12] architecture considers a multi-domain environment. This approach is specifically designed to control a WAN environment, composed of different geographical controllers, that exchange summary information about the local network topology and events. This solution overcomes the HyperFlow limitations, however it does not provide local redundancy: in the case of a controller failure, a remote instance takes control of the switches, increasing the latency between the devices and their primary controller. ElastiCon [13] and Pratyaastha [14] aim to provide an elastic and efficient distributed SDN control plane to address the load imbalances due to static mapping between switches and controllers and spatial/temporal variations in the traffic patterns. SMaRtLight [15] considers a distributed SDN controller aiming at a fault-tolerant control plane. It only focuses on control plane failures, assuming that data plane failures are dealt with by SDN applications on top of the control platform. Service Provider-SDN (SP-SDN) [16] envisages for an extended SDN architecture with the introduction of a service orchestration layer which spans several administrative domains supporting both network and cloud services. The multi-domain applications run on top of the service layer. Similarly, Lifecycle Service Orchestration (LSO) [17] proposes an orchestration layer on top of the SDN control layer. Compared to SP-SDN, LSO envisages for a hierarchical orchestration layer. The lower layer has narrow scope, in fact the LSO components run on top of the single SDN controllers in the different domains. On top of this layer there is a global LSO that orchestrates the intra-domain LSOs. In OpenDaylight [18], an initial work on clustering has been provided in the Helium release using the Akka framework [19] and the RAFT consensus algorithm [20]. Finally, hierarchical SDN Orchestration solutions for WANs are presented in [21] and [22]; [22] discusses general design considerations and alternatives when considering multiple controllers in a parent-child relationship, while [21] focuses on the orchestration of heterogeneous technologies, referred as domains, of the underlying network.

VI. Conclusion

In this paper we have presented ICONA (Inter Cluster ONOS Network Application), an ONOS provider application, which has the objective to enable clustering of ONOS instances across different locations and administrative domains. In fact, even if the current ONOS release scales and performs well on a variety of network topologies, it may not be adequate for all cases, especially when there are requests for latency bounds in restoration scenarios. ICONA is based on a pragmatic approach, that inherits ONOS features while improving its responsiveness to network events like link/node failures or congested links, that impose the rerouting of large set of traffic flows. Moreover, the policies configuration available in ICONA enables the east-west communication with clusters belonging to different administrative domains, thus offering a simple solution for inter-domain SDN scenarios. ICONA is available under open-source license in the ONOS application repository. Future improvements of ICONA will focus on enhancing the policy mechanism with innovative techniques and on new southbound mechanism solutions to interact with other SDN control planes using the east-west interfaces.

References

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *9th USENIX Conference on Operating Systems Design and Implementation*, 2010.

[2] H. Y. Soheil and Y.Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In *First workshop on Hot topics in in software defined networking*, 2012.

[3] A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In *2010 internet network management conference on Research on enterprise networking*, 2010.

[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. Onos: towards an open, distributed sdn os. In *Third workshop on Hot topics in in software defined networking*, 2014.

[5] Onos website - http://onosproject.org/.

[6] Apache karaf - http://karaf.apache.org/.

[7] M. Caesar and J. Rexford. Bgp routing policies in isp networks. *IEEE Network*, 19(6):5–11, Nov 2005.

[8] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *9th ACM Workshop on Hot Topics in Networks*, 2010.

[9] Netem - http://www.linuxfoundation.org/collaborate/ workgroups/networking/netem.

[10] Geant - the core national research and education networks (nrens) european backbone - http://www.geant.net/pages/default.aspx.

[11] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *ACM SIGCOMM Computer Communication Review*, 2013.

[12] K. Phemius, M. Bouet, and J. Leguay. Disco: Distributed sdn controllers in a multi-domain environment. In *IEEE Network Operations and Management Symposium*, 2014.

[13] A.A. Dixit, F. Hao, S. Mukherjee, T.V. Lakshman, and R. Kompella. Elasticon: an elastic distributed sdn controller. In *Tenth ACM/IEEE symposium on Architectures for networking and communications systems*, 2014.

[14] A Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson. Pratyaastha: an efficient elastic distributed sdn control plane. In *Third workshop on Hot topics in in software defined networking*, 2014.

[15] F Botelho, A Bessani, F Ramos, and P Ferreira. *SMaRtLight: A Practical Fault-Tolerant SDN Controller.* ArXiv e-prints, 2014.

[16] James Kempf, Martin Korling, Stephan Baucke, Samy Touati, Victa McClelland, Ignacio Mas, and Olof Backman. Fostering rapid, cross-domain service innovation in operator networks through service provider SDN. In *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014*, pages 3064–3069, 2014.

[17] A. Mayer and S. Mansfield. The third network: Lifecycle service orchestration vision. Technical report, MEF, 02 2015.

[18] Opendaylight - http://www.opendaylight.org/.

[19] Akka framework - http://akka.io/.

[20] Raft consensus algorithm - https://raftconsensus.github.io/.

[21] R. Vilalta, A. Mayoral, R. Munoz, R. Casellas, and R. Martinez. Hierarchical sdn orchestration for multi-technology multi-domain networks with hierarchical abno. In *Optical Communication (ECOC), 2015 European Conference on*, pages 1–3, Sept 2015.

[22] R. Ahmed and R. Boutaba. Design considerations for managing wide area software defined networks. *IEEE Communications Magazine*, 52(7):116–123, July 2014.