DISCO : Distributed SDN Controllers in a Multi-domain Environment

Kévin Phemius, Mathieu Bouet and Jérémie Leguay
Thales Communications & Security
4 avenue des Louvresses, 92230 Gennevilliers, France
{kevin.phemius, mathieu.bouet, jeremie.leguay}@thalesgroup.com

***Abstract*—Software-Defined Networking (SDN) is now envisioned for Wide Area Networks (WAN) and deployed constrained networks. Such networks require a resilient, scalable and easily extensible SDN control plane. In this paper, we propose DISCO, a DIstributed SDN COntrol plane able to cope with the distributed and heterogeneous nature of modern overlay networks and deployed networks. A DISCO controller manages its own network domain, communicates with other DISCO controllers to provide end-to-end network services and share aggregated network-wide information. This east-west communication is based on a lightweight and highly manageable control channel which can self-adapt to network conditions.**

I. Introduction

In this paper we propose DISCO [1], an open *DIstributed SDN COntrol plane* for multi-domain SDN networks. It relies on a per domain organization, where each DISCO controller is in charge of an SDN domain, and provides a unique lightweight and highly manageable control channel used by *agents* that can be dynamically plugged into the different domain controllers composed of an intra-domain part and an inter-domain part (Sec. II). We demonstrate how DISCO dynamically adapts to heterogeneous network topologies while providing classic functionalities such as end-point migration and being resilient enough to survive disruptions and attacks (Sec. III). Contrary to state of the art distributed SDN control planes which posses Distributed Databases that are costly in energy and mostly in a cloud environment, DISCO focuses on WAN constraints and discriminates heterogeneous inter-domain links such as high-capacity MPLS tunnels and SATCOM interconnections with poor bandwidth and latency. We implemented DISCO on top of the Floodlight [3] OpenFlow controller and the AMQP [2] protocol.

II. Architecture and Implementation

DISCO is a distributed multi-domain SDN control plane which enables the delivery of end-to-end network services. A DISCO controller is in charge of a network domain and communicates with neighbor domains to exchange aggregated network-wide information for end-to-end flow management purposes.
The architecture, detailed in this paper [1], is composed of two parts: an intra-domain part which gathers the main functionalities of the controller and an inter-domain part to manage the communication with other DISCO controllers (reservation, topology state modifications, disruptions, …).

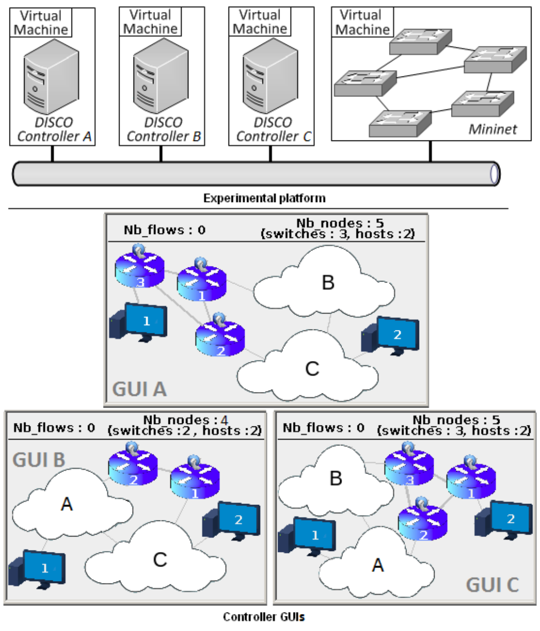

Fig. 1. Experimental setup (top) and controllers' GUIs (bottom).

It is composed of two key elements: (i) a *Messenger* module which discovers neighboring controllers and maintains a distributed publish/subscribe communication channel, and (ii) different *agents* that use this channel to exchange network-wide information with intra-domain modules. *Messenger* provides an open communication bus on top of which any *agent* can be plugged dynamically. It can subscribe to topics published by other *agents* and start publishing on any topic. Each *agent* publishes and consumes messages on a required subset of topics that they manage to ensure the consistency in the system.

III. Demonstration

A. Platform Setup

The network topology considered in the evaluation is composed of interconnected SDN domains. Each network domain A, B and C is managed by a local DISCO controller, which coordinates with its neighbor DISCO controllers. This setup is representative from a typical enterprise network where several sites (edge networks or datacenters) are interconnected with different WANs. The hosts connected to the network domains can be either user terminals or Virtual Machines (VM).

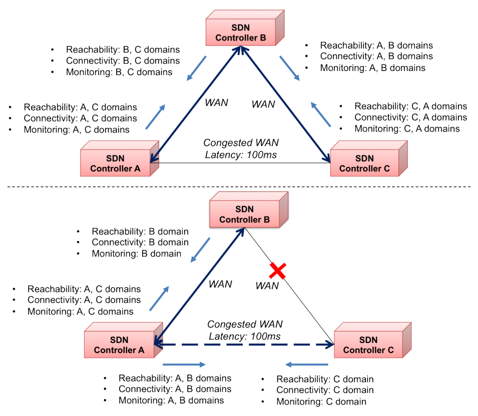

Fig. 2. Adaptive Information Exchange in DISCO.

The testbed was enclosed in a private cloud but can be transposed in a small server or a personnel computer (see Fig. 1 (top)). The network is emulated using *Mininet* [4], a tool used to create rich topologies and instantiate Open vSwitch [5] switches and virtual hosts. *Mininet* is hosted on a dedicated VM while the controllers are hosted on separate VMs. The different links latencies and bandwidths are enforced using Linux's *tc* command. This setup allows a fine control over the network. We can see the results on the setup in Fig. 1 (bottom).

B. Adaptive Information Exchange Use Case

This scenario shows how the message exchanges can self-adapt to the network conditions. In order to reduce the network footprint of control information exchanged between domains, *agents* adopt a twofold strategy: (i) they identify alternative routes to offload traffic from weak outgoing interconnections (e.g., low-bandwidth satellite connection, congested link), and (ii) they reduce the frequency of control messages for these links if they do not find an alternative route. Each *Monitoring agent* usually sends information every $2s$. This period increases to $10s$ for weak interconnections. The *Connectivity* and *Reachability agents* also send their information using alternative routes whenever possible. However, contrary to the *Monitoring agents*, they only exchange messages in a reactive manner, that is when an event occurs.

Upon bootstrap and discovery, the three controllers reach the situation described on top of Fig. 2. In this scenario, the link between the domains A and C is congested. Its latency equals $\geq 50ms$. B is thus relaying control information for A and C in order to offload the congested link. In case the inter-domain link between B and C fails, *Monitoring agents* reconfigure themselves to the situation presented at the bottom of Fig. 2 where monitoring traffic is passed through the weak link $A \rightarrow C$, but at a lower frequency.

C. Intra-domain and Inter-domain Link failure Use Case

This scenario emphases on what happens whenever a link failure occurs, both inside a domain and in a peering link. Thanks to its algorithm, a link failure inside a domain does not necessarily impact the system, even if a flow was using that link. The DISCO controller dynamically recomputes a path for that flow (whenever possible) and does not share that information with its neighbors.

In contrast, if an inter-domain link is affected by a failure, the DISCO controller will (i) recompute a new route for affected flows and (ii) immediately send a message about the disruption. In consequence a remote controller will not count on this peering link when computing flow routes.

For now, the flow are forwarded in Best Effort with the shortest path route. The algorithm can be improved to include more metrics like link's latency or region restrictions.

D. Host Migration Use Case

Our final scenario presents how the distributed control plane deals with end-points migration. This migration can be due to the fact that an administrator moves a VM from a site to another or because a mobile host hops from a wireless Access Point to another. At the beginning, a UDP flow is established between two end points. At some point of the run, the destination host moves from the domain C to B. We will see that the traffic is dynamically redirected to the new position of the destination host with minimal losses.

During the transition, the following happens :

- The destination host is disconnected from the domain C
- C's DISCO controller sends a *Host_Removed* message
- If a controller (like A) detects that a flow was going to the removed host, it will stop that flow until it has new knowledge about its position
- The destination host is reconnected to B
- B's controller sends a *Host_Added* message
- A's controller resumes the flow with a route to the new destination's position

Any flow running to an unknown destination will automatically be blocked by the controller to prevent resource waste. Conversely it will immediately start again when the destination is available. This strong connection between a host and its position mimics LISP.

IV. Conclusion

In this paper we have presented how DISCO, while being able to work on a single domain, can dynamically adapt to heterogeneous multi-domain network topologies to assure end-point migration and link failure mitigation. DISCO is easily extendable and can integrate more capabilities while keeping the underlying mechanism simple and open.

References

[1] K. Phemius, M. Bouet and J. Leguay, "DISCO: Distributed Multi-domain SDN Controllers", in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[2] "AMQP." [Online]. Available: http://www.amqp.org

[3] "Floodlight OpenFlow Controller." [Online]. Available: http://floodlight.openflowhub.org/

[4] "Mininet." [Online]. Available: http://mininet.org

[5] "Open vSwitch." [Online]. Available: http://openvswitch.org/

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner,"OpenFlow: enabling innovation in campus networks",SIGCOMM Comput. Commun. Rev. vol 38, 2008