Decentralizing SDN's Control Plane

Mateus A. S. Santos*, Bruno A. A. Nunes†, Katia Obraczka‡, Thierry Turletti†,
Bruno T. de Oliveira* and Cintia B. Margi*
*University of Sao Paulo, Brazil †INRIA, France ‡UC Santa Cruz, USA

***Abstract*—Motivated by the internets of the future, which will likely be considerably larger in size as well as highly heterogeneous and decentralized, we propose Decentralize-SDN, D-SDN, a framework that enables not only physical– but also logical distribution of the Software-Defined Networking (SDN) control plane. D-SDN accomplishes network control distribution by defining a hierarchy of controllers that can "match" an internet's organizational– and administrative structure. By *delegating* control between *main controllers* and *secondary controllers*, D-SDN is able to accommodate administrative decentralization and autonomy.It incorporates security as an integral part of the framework. This paper describes D-SDN and presents two use cases, namely network capacity sharing and public safety network services.**

I. Introduction

The growing need to facilitate network evolution motivated the emergence of the Software-Defined Networking (SDN) paradigm. SDN's premise is to decouple the network control- and data planes and thus make deploying new network services and protocols viable especially in production networked environments. However, SDN techniques to-date, including OpenFlow, have mostly targeted "managed networks". As such, they promote logically centralized control which is ill-suited not only to the scale but also to the level of administrative decentralization and episodic connectivity that may be present in future internets.

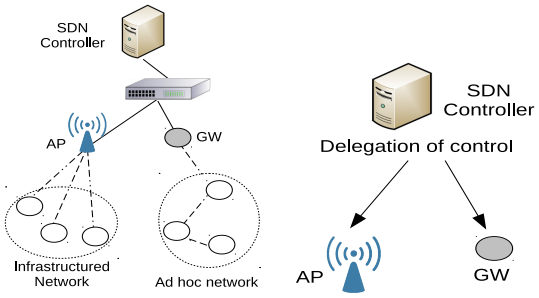

Fig. 1. SDN control distribution

Most approaches aiming at increasing the scalability and robustness of the SDN control plane have also targeted "managed" networks, e.g. data centers and intranets, where it is reasonable to assume the existence of a single, logically centralized administrative authority, as shown in the left part of Figure 1. However, this assumption does not hold in heterogeneous internets that may include a variety of autonomously administered networks, such as infrastructureless self-organizing networks (as illustrated in Figure 1).

We propose Decentralize-SDN, or D-SDN, an SDN framework that allows SDN control distribution both physically and logically by defining a *control hierarchy* of *main controllers (MCs)* and *secondary controllers(SCs)*. In "smart spaces" type applications, for example, devices within the home are controlled by the "home" controller independent of the "smart neighborhood" controller and the ISP's controller. D-SDN enables logically decentralized control through control delegation between different levels of the control hierarchy, as shown in the right part of Figure 1. Another distinguishing feature of D-SDN is that it incorporates security as integral part of the framework and its underlying protocols.

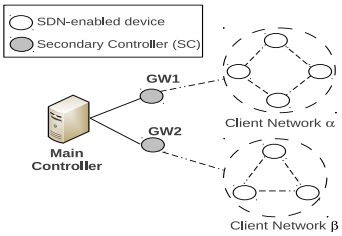

Fig. 2. Control delegation from MC to SCs in a heterogeneous internet.

As proof of concept, we apply the D-SDN framework in two use cases, namely: (1) network capacity sharing, in which control decentralization enables nodes in a infrastructure-less network to connect to the Internet via other (connected) nodes, and (2) public safety network (PSN) scenario that showcases control decentralization in emergency response services.

II. D-SDN Overview

As mentioned previously, D-SDN defines two types of controllers: *Main Controllers* (MCs) and *Secondary Controllers* (SCs). The main difference between them is that SCs require that MCs authorize and delegate control to them before SCs are able to act as SDN controllers. In addition, we envision that SCs will typically be responsible for managing SDN switches in a sub-domain within the MC's domain.

Let us take the scenario shown in Figure 2. Under centralized control, the MC controls the two ad-hoc networks (MANETs). SDN-capable mobile devices in the MANETs need to rely on the MCs forwarding decisions. Thus, every new flow in the MANET generates a request to the MC, which then needs to respond with the appropriate flow modification message(s). Alternatively, using D-SDN's decentralized control plane, $GW1$ and $GW2$ can act as SCs upon MC's authorization and delegation. As a result, new flows arriving at MANET nodes will not need to reach the MC and could be handled directly by the corresponding SC.

Hierarchy of Controllers: In D-SDN, control distribution is based on a hierarchy of MCs and SCs which can also be used to improve control plane availability and fault tolerance.

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

Following the hierarchy, MCs can *delegate* control of certain devices to a particular SC. For example, an SC would not be allowed to write new flow entries to a device's flow table without a delegation from the corresponding MC. Note that SCs must have been previously authenticated by the MC or some other trusted third-party authority before being able to participate in the network control plane.

Control Delegation: An MC can delegate the control to an SC with respect to a set of SDN-enabled devices. Delegation can be initiated by an MC or can occur upon a request from the SC. A delegation request can be triggered by different kinds of events. For example, when a new SC is deployed geographically closer to a set of devices, it could request delegation from the MC to control these devices. Another example is a scenario in which mobile devices in a MANET need connection to the Internet through a gateway node. The gateway can then request authorization from the MC for playing the role of an SC to new devices joining the network.

MC-SC Communication: As previously pointed out, a network device is only able to act as an SC upon the authorization of the corresponding MC through a `Control Delegation` message. In addition, MC-SC communication usually happens within the same administrative domain. Control delegation is illustrated in Figure 3 and proceeds as follows:

- `Check-in Request`: an SC requests authorization for managing a specific SDN-enabled device.
- `Check-in Response`: the MC, upon accessing its database, authorizes or denies access by the requesting SC.

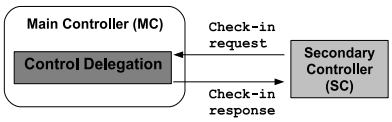

Fig. 3. Delegation of control from main- to secondary controller.

SC-SC Communication and Fault Tolerance: SCs use D-SDN's SC-SC protocol to implement fault tolerance in case of failure of the current SC. Inspired by OpenFlow (OF) 1.3 [7], we differentiate *master* controllers from *slave* controllers in order to provide fault tolerance. Slave controllers do not receive messages from a switch. However, they can become *masters* by sending a *role request* message to the corresponding switch.

The current master controller sends periodic `Hello` messages as keep-alive signaling. Slave controllers detect that the master failed after not receiving `Hello` messages for a predefined period of time. When that happens, slaves will start an election process to select a master among them. If a new master is elected, it will inform the corresponding devices that will be under its control. These devices will then remove the old master from the master role.

The master also sends `Update` messages to slave controllers to make sure that their state is in sync. `Update` messages contain the application modules currently running on the master as well as the list of switches it controls.

III. Implementation

In our testbed, mobile nodes are *SDN-enabled* through the use of software switches, e.g., Open-VSwitch (OVS). SDN-enabled nodes can thus be responsible for forwarding incoming traffic, maintaining flow tables, and communicating with the controller when needed.

Our current implementation is comprised of a *server-side* and a *client-side*. The server-side exposes an interface to a hierarchy of controllers. The client-side provides accounting data to the servers as well as management of cryptographic material that is used for providing security services such as data confidentiality and authentication.

Regarding security, we use Identity Based Cryptography (IBC) [9]. which requires a Trusted Third Party (TTP) responsible for secret key generation. There is synergy between controllers and TTPs. In particular, MCs can play the role of a TTP. Using the notation presented in Table I, we describe the main protocols D-SDN components use in order to communicate.

ID_X, ctr	identity of X and counter, respectively
S_X, P_X	private and public key of X, respectively
$K_{X,Y}$	key established between nodes X and Y
$\texttt{authenc}(\cdot, k)$	authenticated encryption using key k
$\texttt{enc}(\cdot, k)$, $\texttt{dec}(\cdot, k)$	encryption/decryption using key k
mac, s	authentication tag and master secret key

TABLE I
Notation.

Setup: As public keys are derived from identities, the TTP (i.e., the controller) maps the node identity, ID_X, to a point in the elliptic curve, P_X. This mapping is a public parameter, since a node is allowed to generate any device's public key. The TTP generates a master secret key s and calculates each node's private key as $S_X = sP_X$. This value should be either sent privately by the TTP or pre-deployed on the device (i.e., SC or end-host device).

Authenticated Key Agreement: Pairings[8] provide practical implementation for authenticated key agreement (AKA) over IBC, which is an elegant alternative to non-authenticated schemes such as the Diffie-Hellman interactive key exchange.The AKA procedure considered here has the main goal of avoiding public key encryption. It means that, once a key is agreed between two nodes using public key cryptography, they can use the shared key for confidentiality and data authentication.

Handshaking: In the handshaking procedure, a new coming device, or requesting node (RN), is required to respond to a challenge, so that the authenticator is able to verify the devices' identity. This allows them to compute a shared key, which is used for authenticated encryption of the challenge. Figure 4 shows in detail this process.

Availability: The proposed framework is available for download from `http://inrg.cse.ucsc.edu/community`.

IV. Evaluation

A. Secure Capacity Sharing

For the secure capacity sharing use case, we assume the network model illustrated in Figure 2, where a node in a *client*

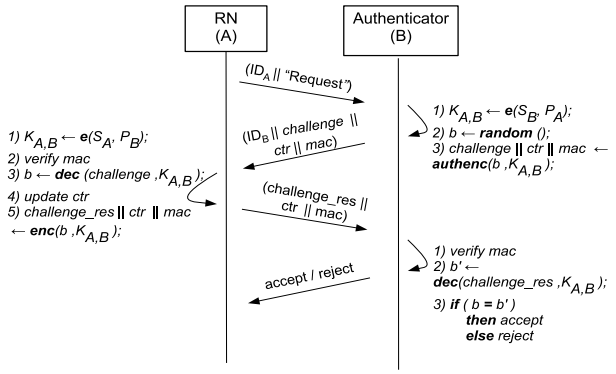

Fig. 4. Detailed handshaking procedure.

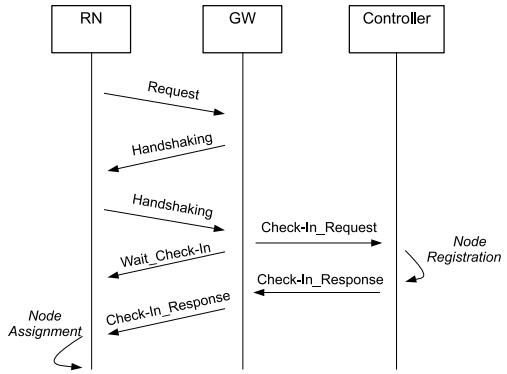

Fig. 5. High level description for node authorization through main controller.

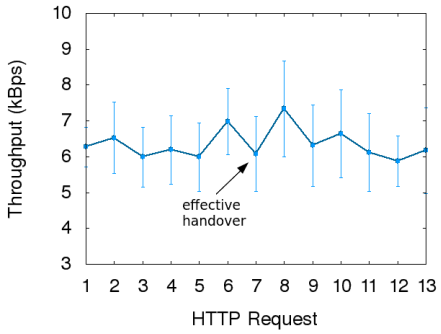

Fig. 6. Throughput before and after handover.

Fig. 7. Throughput before and after the event of activating a redundant gateway with no QoS enforcement policy.

network, called here the "Requesting Node" (RN), wishes to connect to the Internet and accesses, for example, the World Wide Web. However, it is unable to connect to the existing network infrastructure (e.g., because the RN is out of range of the closest AP). Another node, called gateway node 1 - "GW1", advertises its gateway services providing RN the option to connect to the Internet through it. Note that RN can connect to GW1 directly or through a wireless, multi-hop ad-hoc network (MANET) using some existing MANET routing protocol to route packets towards GW1.

The main steps to achieve secure network capacity sharing using D-SDN (illustrated in Figure 5), are as follows:

- **Gateway discovery:** GW nodes send periodic messages, announcing their gateway capabilities. The potential users, on the recipient of such messages will choose a GW, by sending a `Request` message to the most suitable candidate;
- **Handshaking**: a GW node responds to a user request and initiates a handshaking procedure for node authentication;
- **User check-in**: the GW requests authorization to the main controller, which queries its database in order to approve allocation of resources to the designated customer.

If a user is authorized, the main controller adds the new flow-table entries to the forwarding devices on user data path towards the Internet. The procedure of user check-in includes the *delegation of control* from the MC to the gateway with respect to user device administration.

Secure Handover: Here, we adopt the scenario in which a user notices that a *more suitable* GW becomes available. The user itself can send a request to the new candidate and perform a handshaking procedure. Then, the MC can orchestrate flow creation and removal in the new and old gateways, respectively.

In order to demonstrate the handover, we generated a sequence of HTTP requests to an external web server (located outside the local network) and measured the throughput. We collected 10 samples for each element of the sequence and report a 95% confidence level in our results. Figure 6 shows the results, in which *effective handover* points to the first HTTP request after the new gateway took over. It can be seen from the figure that the throughput fluctuates so that the handover cannot be observed among different HTTP requests. In this particular case, both gateways presented similar performance. We emphasize that our goal is not to increase performance among gateways, but to provide seamless handover.

QoS and Gateway Redundancy: Quality of service can be enforced by MCs or SCs by using ingress policy rates. In the same scenario of Figure 2, a gateway would prevent RNs from allocating more than a determined fraction of the total bandwidth provided by the ISP.

We carried out experiments using one single gateway with the ingress policy set to 3 kBps. Then, another gateway with no restrictions becomes available as a redundant channel to the infrastructured network. We measured the throughput during sequences of HTTP requests to a server. Figure 7 shows that network performance is limited to the configured throughput as long as the redundant GW is not activated.

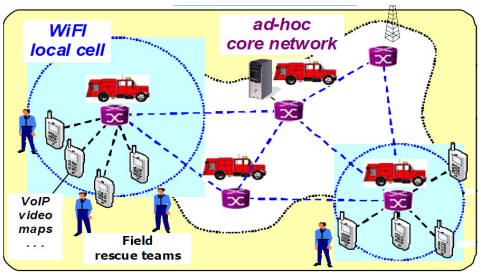

Fig. 8. Broadband scenario for inter-agency communication. Source: [1]

GOAL: Evaluation of fault tolerance in a MANET, in which SC_i is the active controller. SC_j takes over after SC_i failure.

1) SCs exchange periodic `Hello` messages with their identities and roles for each SDN-enabled device, if any exists, under their scope;
2) SC_i fails;
3) An election protocol is triggered among SCs due to a *timeout* for receiving `Hello` messages from the *master* controller;
4) The elected controller, say SC_j, requests the administration of the corresponding SDN-enabled devices and effectively replaces the failed controller;
5) `Role Reply` messages from devices confirm that SC_j took over.

Fig. 9. Scenario for fault tolerance among the SCs inside a MANET.

B. Public Safety Networks

PSNs are built to detect and/or handle disaster events [3]. Such networks are set to provide communication and coordination for emergency responders and operations. Many of the challenges in the PSN field come from the variety of systems and agencies involved in the crisis response and from their mobility at the disaster site [3]. By decentralizing the control plane, our proposed framework allows rapid deployment, reliability and interoperability.

We envisage a scenario in which public safety authorities can organize themselves for exchanging valuable information regarding an emergency situation. We showcase our proposal over such a scenario, illustrated in Figure 8. In this figure vehicles are capable of serving as GWs to a network of different agency actors (e.g., firefighters and police officers).

Our testbed instantiates SCs at the agencies' vehicles. A single agency can have many decentralized SCs that exchange messages with other agencies' SCs. They should rely on our framework in order to continue operating correctly in the event of link failures. Figure 9 describes the proposed scenario for implementing tolerance to failures.

Methodology and Results: The experiments were carried out using four controllers and one switch. A single node was set as master for the switch. All the nodes, including the switch, were configured in a wireless ad hoc network. We integrated the Paxos election protocol proposed with our framework.

Before presenting the results, we elaborate on the main parameters of the system. Let t_h be the time between periodic `Hello` messages sent by the master controller. Let t_{out} be the timeout, or in other words, the time a non-master controller waits for receiving the next `Hello` message. Given that $0 < t_h \leq t_{out}$, the worst case scenario for controllers to detect a failure is when the master actually fails just after sending a `Hello` message. In our experiments, we used $t_{out} = 5$ seconds and $t_h = 3$ seconds.

Minimum	Maximum	Average (95% confidence interval)
2.3	6.7	4.2 (3.7, 4.7)

TABLE II
TIME IN SECONDS TO RECOVER FROM A FAILURE ($t_{out} = 5$ AND $t_h = 3$).

We collected 20 samples and computed a 95% confidence interval. We used a random failure time at each sample. Table II shows the recovery time, which is not only the time to detect a failure, but also the time it takes for the new master to take control of the switch. The minimum time (i.e., 2.3 seconds) is close to the best case scenario mentioned earlier.

V. Related Work

Previous work such as [5] propose a logically centralized but physically distributed control plane by means of a distributed file system. The trade-offs on distributing the control plane under a logically centralized scheme are investigated by Levin et al. [6]. An example of hierarchical control is Kandoo [2], which allows the deployment of *local* controllers with no network-wide state. Nevertheless, Kandoo still needs a logically centralized *root* controller. Phemius et al. proposed DISCO [4]. Even though DISCO is decentralized, it neither considers controller hierarchy nor deals with fault tolerance.

VI. Conclusion

We proposed Decentralize-SDN, a general framework enables a wide range of current- as well as future network services and applications through the decentralization of the SDN control plane. D-SDN supports control distribution by defining a hierarchy of controllers in which main controllers can delegate functions to secondary controllers. As future work, we envision new D-SDN based network services and applications, such as inter-domain routing and load balancing.

Acknowledgments

This work is partly funded by the Community Associated Team between INRIA and UCSC, the French ANR under the "ANR-13-INFR-013" project, the Sao Paulo Research Foundation (FAPESP) under grant 2013/15417-4, the National Council for Scientific and Technological Development (CNPq) under grant 245588/2012-4, and by NSF grant CNS 1150704.

References

[1] EADS Defence and Security Systems, "CHORIST final report SP0.R7," 2009, available online at `http://www.chorist.eu/`.
[2] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 19–24.
[3] Iapichino, Giuliana, D. Camara, C. Bonnet, and F. Filali, *Public Safety Networks*. IGI Global, 2011.
[4] J. L. Kevin Phemius, Mathieu Bouet, "DISCO: Distributed multi-domain sdn controllers," *CoRR*, vol. abs/1308.6138, 2013.
[5] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.
[6] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the workshop on Hot topics in software defined networks*. New York, USA: ACM, 2012, pp. 1–6.
[7] Open Networking Foundation, "Openflow switch specification 1.3," 2012.
[8] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *SCIS00*, 2000, pp. 26–28.
[9] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*, ser. LNCS, G. Blakley and D. Chaum, Eds. Springer Berlin Heidelberg, 1985, vol. 196, pp. 47–53.