Self-configuration Mechanisms for SDN Deployment in Wireless Mesh Networks

Mohamed Labraoui[1,2], Michael Boc[1], Anne Fladenmuller[2]
[1]CEA, LIST, Communicating Systems Laboratory
Commissariat l'Energie Atomique (CEA)
F-91191 Gif-sur-Yvette CEDEX, France
[2]Laboratoire d'Informatique de Paris 6 (LIP6/CNRS)
Université Pierre et Marie Curie-Paris VI
4 Place Jussieu, 75005 Paris, France
{mohamed.labraoui,michael.boc}@cea.fr, anne.fladenmuller@lip6.fr

***Abstract*—Software-Defined Networking (SDN) is a framework conceived to make network infrastructures more programmable and more easily manageable. We explore the integration of the latter in Wireless Mesh Networks (WMNs) through the design of an opportunistic and self-configurable SDN solution. Opportunistic to be able to cope with the intermittent connectivity and network partitioning and merging scenarios caused by the dynamic nature of the wireless medium; and self-configurable to be able to redirect dynamically SDN-enabled Wireless Mesh Routers (WMRs) to the most suitable SDN controller. This paper proposes mechanisms allowing to make automatic an SDN-based WMN deployment. Our proposed solution enables (1) for each WMR to discover the reachable SDN controllers, then to select and connect by itself to the most suitable one; (2) the setting up of new SDN controller(s) among the WMRs in case of controllers unavailability. We validated our work using NEON, an SDN solution developed by CEA LIST that supports fast devices configuration and services deployment in dynamic and unconfigured infrastructures contexts.**

***Keywords*—Software-Defined Network, Wireless Mesh Network, self-discovery, auto-configuration, bootstrapping, SDN deployment.**

I. INTRODUCTION & CONTRIBUTION

The integration of an SDN [1] solution into a WMN [2] involves a certain number of prerequisites: it is necessary to plan ahead the SDN controller location and to set up each WMN device (mesh router) to establish the connection to the SDN controller. If now, the SDN controller has to be relocated (due to network topology modification for instance), all WMN devices have to be reconfigured. Besides, due to the unstable nature of WMNs (the wireless medium can cause intermittent connectivity, interference, asymmetric communications, to name a few) the SDN controller can often be unreachable preventing normal operations.

Despite the significant progress accomplished on the reliability and scalability aspects in the field of distributed SDN control plane architectures, there are very few works focusing on the deployment-related challenges in dynamic environments. Indeed, while some of those solutions assume (in particular during the "master controller" election phases) that communications between the different SDN controllers are stable [3], others require an initial configuration and/or an additional component such as a DHCP server [4] [5]. Insofar as these two solutions classes are only effective within stable mediums (in terms of communications quality), they are not suitable for WMNs.

In this paper, we propose mechanisms to achieve a fully automated and reliable deployment solution for SDN-based WMN architectures:

- A mechanism to allow WMRs (Wireless Mesh Routers) to discover SDN controllers present in the network.
- A solution to elect the master controller in a scenario with multiple concurrent SDN controllers. The solution considers their potential heterogeneous natures (in terms of computational capabilities), this aspect has a significant impact on the network scalability.
- A mechanism to set up an SDN controller on the fly among the WMRs when necessary (in case of controllers unavailability). The WMRs will be able to manage themselves, creating hence an opportunistic SDN-based wireless mesh network.

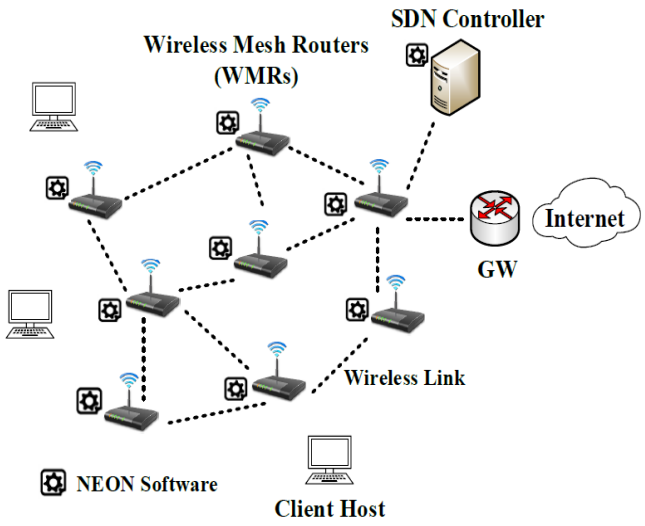

Fig. 1: Network Architecture Scenario

To address these challenges, we used NEON [6], an SDN framework for dynamic networks. NEON was developed by CEA LIST and aims at enabling fast devices configuration and services deployment - within seconds - in dynamic and unconfigured infrastructure contexts. NEON has also the capacity to control the configuration of other local applications and libraries such as Open vSwitch (OpenFlow [7] application). At the base defined as an SDN southbound protocol API, we enhanced NEON in such a way to integrate also its own SDN controller as well as a set of services. In our network architecture scenario (Figure 1), each device is equipped with

978-1-5386-2723-5/17/$31.00

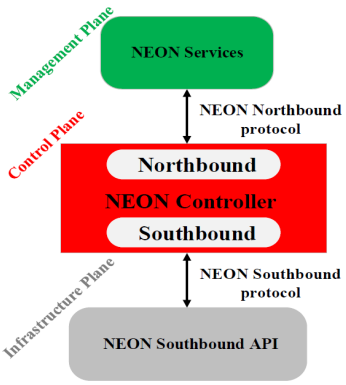

Fig. 2: Overview of NEON SDN reference architecture NEON software.

Typically, a device equipped with NEON software (Figure 2) can have two possible working modes depending on the status of its NEON SDN controller [7]: (i) "master mode" if its controller has the master status, (ii) "slave mode" if its controller has the slave status.

II. AUTOMATING SDN DEPLOYMENT

Unlike the wired networks, routing packets in the software-defined wireless mesh networks represents a key challenge [8]. Through our previous work covering the SDN-assisted routing in wireless mesh networks [9], our insight is that distributed routing protocols for WMNs have been used for a long time and are an integral part of mesh networks, nevertheless the SDN approach provides advantages that can positively help the distributed routing protocol operations. Consequently, in the present deployment, a classical distributed routing protocol (babel routing protocol in our case [10]) will guarantee the reachability between devices, while the SDN solution ensures all aspects of management and reliability of the network.

WMRs are configured to boot by default in slave mode, however they are able to switch into master mode when necessary. In contrast, an SDN controller is programmed to be always in a master mode. The operating principle of these two modes is detailed below:

A. *Slave Mode*

As shown in Figure 3, when a WMR is newly added to the network, its default mode is set to "slave mode" (1). Then through its local NEON-based service, it joins automatically the mesh network, allowing it to communicate with the rest of the network (2). Once this step achieved, it broadcasts periodically over a given interval of time (fixed by a timeout) messages of type "Presence Indicator" announcing its existence (3). The same operation is carried out by all devices (i.e. WMRs and SDN controllers) in the mesh network. And in parallel, a dedicated database (located in the WMR newly added) is populated with all detected devices across the network (4-5). If the timeout is reached (7), the WMR stops announcing its presence as well as populating the database. At this stage, it checks whether SDN controllers are present in the database. There are two possible cases:

- *Presence of SDN controllers in the database:* the WMR selects the most suitable controller (Section III) and connects to it (8,11), passing hence to a "managed status". In this status, routing rules originating from the controller have systematically a higher priority

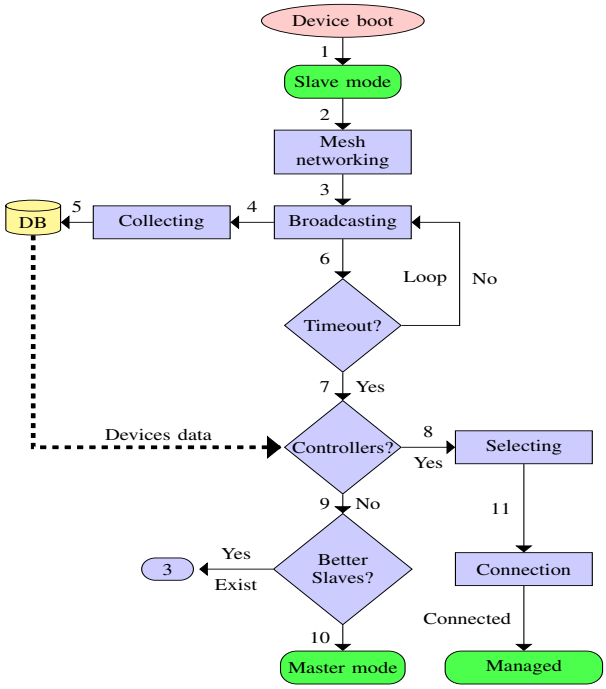

Fig. 3: Operating diagram of a device in slave mode

compared to those incurred by the distributed routing protocol. In addition, a WMR in this status has also to manage messages of type "Connection Request" from its controller if the latter wishes to assign it to another one. Besides, as soon as it loses the connection with its controller, it returns to step 3.

- *No SDN controllers in the database:* According to the controller election procedure defined in Section III, the WMR compares its capabilities with those of other slave devices. If it considers itself as the most adequate device to become the master controller, it switches to master mode, otherwise it returns to step 3.

B. *Master Mode*

The operating diagram of a device in master mode is depicted in Figure 4. In master mode, the device (i.e. WMR or SDN controller) ensures the WMN management. In addition, it participates in the automatic SDN deployment using the following procedures:

- It broadcasts periodically messages of type "Presence Indicator" to announce its existence (2). At the same time, it receives the same type of message from all devices present in the same mesh network (3). This will enable it to populate its database (4).
- It ensures the presence of a single "master controller" in the network (5). Indeed, the WMNs are often led to split into multiple partitions, or rather the opposite, to merge between them. Consequently, several master controllers might coexist managing only a subset of the whole network. Therefore, a control plane centralization procedure is required. This procedure consists in selecting, among the master controllers, the best candidate to manage the whole network.

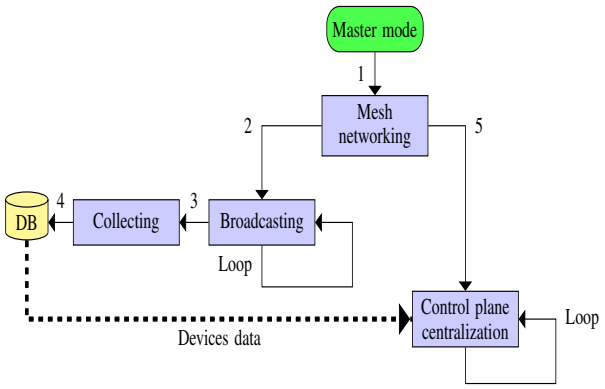

Fig. 4: Operating diagram of a device in master mode

III. CONTROLLER ELECTION PROCEDURE

Considering the fact that all WMRs in the network are configured to launch NEON application at boot-up, and consequently the potential of each one of them to play the role of an SDN controller, the election process can be summarized as a "master assignment problem" [11] [3]. A naive approach of master-selection in a SDN-based WMN scenario with multiple concurrent SDN controllers has already been proposed in [3]. Considering the issues incurred by the wireless mesh environment (topology changes, links unreliability, ...), the authors suggest that the master controller selection procedure should be made directly at the mesh routers level. Similarly, we propose to implement in each mesh router a mechanism that enables it to select autonomously its own master controller. However, our mechanism considers more additional aspects compared to what was proposed in [3].

Indeed, one of the major characteristics of our network architecture scenario is the heterogeneous nature of the physical network infrastructure (in terms of hardware resources). To the best of our knowledge, none of the existing deployment solutions for SDN-based WMNs (with multiple concurrent controllers) considers the case where the SDN controllers would have highly-variable hardware characteristics. Therefore, we can easily highlight the necessity of having a mean through which one could classify controllers based on their hardware capacities. This is particularly necessary considering the fact that the aim behind the SDN solution deployment is to ensure all aspects of management and reliability in the network, consequently an SDN controller needs to manage a large amount of data. From there, we will focus on the following criteria:

A. *The Controller Capacity*

As mentioned above, we aim to find out an approach that would allow routers to select the SDN controller which may lead to the best network management capability. This latter depends largely on the SDN controllers hardware resources (i.e. computational capabilities) given that the NEON software stack is the same for all. The issue is that, classifying the controllers on the basis of their highly-variable hardware characteristics (CPU, Number of CPU Cores and RAM) is not an easy task in the sense that it's impossible to consider all scenarios and all existent technologies [12]. Moreover, the design of a module in NEON software that analytically predicts the controller capacity based on the hardware characteristics is not the optimal option. In fact, if we change the current programming and execution model of NEON software, the entire performance prediction model should be updated accordingly.

Algorithm 1: Testing SDN controller capacity

```
Data: N fake switches (N=16 by default)
Create N OpenFlow sessions to the controller;
for each session do
    while buffer not full do
        queue packet_in's;
        count flow_mod's as they come back;
    end
end
Result: average flow setup throughput (flows/s)
```

In order to find another approach for controllers capacity classification, we conducted an exhaustive experimental study on the benchmarking methodology for SDN controllers. We find out that the average flow setup throughput (flows/sec) [13] is the adequate metric to reflect the SDN controller computational performance. This metric is usually defined as the maximum number of new flows rate that an SDN controller can support. Obviously, the more this metric is high, the more suitable an SDN controller is for the network management. Therefore, we think that the controller election procedure should be initially based on the controllers' average flow setup throughput estimations. In this sense, each controller S will be characterized by its average flow setup throughput estimation, referenced by the term Capacity(S). Consequently, using that metric, WMRs will select the most powerful controller.

B. *Integrating a Controller Testing and Analysis Module*

In order that each device involved in the deployment can inform all others of its average flow s etup t hroughput, a controller performance analyser module has been integrated in NEON software. This module is based on Cbench algorithm [13] summarized in the Algorithm 1.

This controller testing operation is performed at each device boot phase. Table I summarizes the parameters used for tests. Once calculated, the average flow s etup throughput (flows/sec) m etric i s i ncluded i n " Presence I ndicator" messages, and sent to the other devices.

TABLE I: CONTROLLER BENCHMARKING PARAMETERS

PARAMETER	DESCRIPTION	VALUE
Devices Number	Number of fake OpenFlow devices connected to the controller	50
Version	OpenFlow protocol version	1.4
Test Loops	Number of tests carried out	5
Test Duration (ms)	Duration of each test	1000

IV. VALIDATION

The purpose is to validate the controller election procedure which represents the key phase of the SDN deployment architecture. The proposed metric is based on the controllers' average flow setup throughput estimations. Therefore, it's important to validate that the latter is effective and allows routers to select the most powerful SDN controller.

A. *Experiment Setup*

We created our network using the Common Open Research Emulator (CORE) [14]. In order to quickly create a large number of virtual network components (links, hosts, switches,

routers and so on), CORE uses a virtualization technology based on the Network Namespaces functionality in Linux Containers (LXC). In our setup, we emulated a Wireless Mesh Network (WMN) consisting of 8 routers (Figure 5).

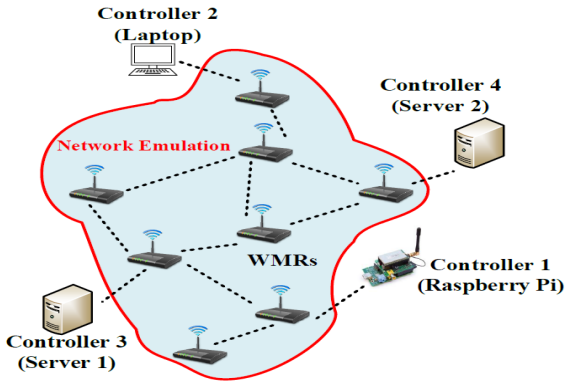

Fig. 5: Validation Architecture Scenario

Considering the case where the SDN controllers would have highly-variable hardware characteristics, we have connected to the network 4 controllers running on different hardware platforms (Laptop, Raspberry Pi and two servers). In the table II, one can find t he h ardware c haracteristics (CPU, Number of CPU Cores and RAM) associated to each SDN controller. Besides, Each one of them was configured t o start the NEON application at boot-up.

TABLE II: HARDWARE CHARACTERISTICS

Controllers	1	2	3	4
Hardware Characteristics				
CPU frequency (MHz)	900	933	3312	3312
Number of CPU Cores	4	4	14	23
RAM (MBytes)	862.40	3720	18930	35580

B. *Results*

In the table III below, we summarize the principal results of this validation. As already explained, the controllers testing phase is performed at boot time at the level of each one, the average flow setup throughput estimations are noted in the part "NEON Evaluation" of the table. As expected, the controller 4 has been chosen by the routers given its evaluation result.

To validate this choice, we have tested the four SDN controllers separately. For each one, the network management capacity (i.e. the data flow rate that can be managed in the WMN) has been evaluated. We can note that the controller 4 presents the best network performance, which confirms the smooth operating of our controller selection procedure.

V. CONCLUSION

This paper proposes mechanisms permitting to deploy an SDN-based WMN architecture automatically. The key target for us was to bypass preconfiguration phases usually required. To address this challenge we extended NEON, a light-weight SDN software for dynamic networks developed by CEA lab. Equipped with NEON, WMRs are able to discover the SDN controllers present in the network and then to connect by themselves to the most efficient one, without any pre-configuration.

TABLE III: VALIDATION RESULTS

Controllers	1	2	3	4
NEON Evaluation				
Average flow setup throughput (flows/s)	42495	55265	94439.67	128539.30
Relative Standard Deviation (%)	6.5	1.8	2.33	4.84
Measurement of the average data flow rate treated by the WMN				
Network performance (flows/s)	39123	45933	80042.21	100399.62

Besides, in case of controllers unavailability, the WMN can manage itself by setting up SDN controllers on the fly among the routers. In the context of this work, an SDN-based Mesh deployment service based on the NEON capabilities has been implemented. By using the CORE emulator, we have been able to validate our controller selection procedure. The latter is performed by the routers in order to select the SDN controller leading to the best network management performances.

Acknowledgements : The work presented in this paper is supported by the French-German project BERCOM jointly funded by ANR and BMBF under the grant number ANR-14-PICS-0001.

REFERENCES

[1] N. M. et al., "Openflow: enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*. ACM New York, NY, USA, March 2008, pp. 69–74.

[2] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, 2005.

[3] S. Salsano, G. Siracusano, A. Detti, C. Pisa, P. L. Ventre, and N. Blefari-Melazzi, "Controller selection in a wireless mesh SDN under network partitioning and merging scenarios," *CoRR*, vol. abs/1406.2470, 2014.

[4] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Automatic bootstrapping of openflow networks," in *Local Metropolitan Area Networks (LANMAN), 2013 19th IEEE Workshop on*, April 2013.

[5] R. Katiyar, P. Pawar, A. Gupta, and K. Kataoka, "Auto-configuration of sdn switches in sdn/non-sdn hybrid network," in *Proceedings of the Asian Internet Engineering Conference*, ser. AINTEC '15. New York, NY, USA: ACM, 2015, pp. 48–53.

[6] S. Decremps, S. Imadali, and M. Boc, "Fast deployment of services in sdn-based networks: The case of proxy mobile {IPv6}," *Procedia Computer Science*, vol. 40, pp. 100 – 107, 2014, moWNet'2014.

[7] Open Networking Fundation, "OpenFlow Specification 1.4.0." [Online]. Available: https://www.opennetworking.org

[8] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 27:1–27:11, Nov. 2014.

[9] M. Labraoui, M. M. Boc, and A. Fladenmuller, "Software defined networking-assisted routing in wireless mesh networks," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sept 2016, pp. 377–382.

[10] J. Chroboczek, "RFC6126 - The Babel Routing Protocol," *IETF*, 2011.

[11] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. ACM, 2013, pp. 7–12.

[12] S. Gurun, C. Krintz, and R. Wolski, "Nwslite: A general-purpose, nonparametric prediction utility for embedded systems," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 32:1–32:36, May 2008.

[13] R. SHERWOOD and K.-K. YAP, "Cbench controller benchmarker," 2010. [Online]. Available: http://archive.openflow.org/wk/index.php/Oflops

[14] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, Nov 2008, pp. 1–7.