

# Software Defined Networking for Community Network Testbeds

Emmanouil Dimogerontakis  
Universitat Politècnica de Catalunya  
Department of Computer Architecture  
Barcelona, Catalonia, Spain  
Email: edimoger@ac.upc.edu

Ivan Vilata  
Pangea.org NGO  
Barcelona, Catalonia, Spain  
Email: ivan@pangea.org

Leandro Navarro  
Universitat Politècnica de Catalunya  
Department of Computer Architecture  
Barcelona, Catalonia, Spain  
Email: leandro@ac.upc.edu

**Abstract**—Wireless Community Networks have received increasing attention the latest years. In an effort to set the cornerstone for an internet without central authorities and monopolies, network engineers throughout the world have started creating community networks. To enhance this effort, Community-lab, a wireless community networks testbed, was created allowing researchers to experiment with new protocols and applications in a realistic environment. Nevertheless, this testbed does not offer the ability to perform L2 experiments. To address this gap, we developed a platform that allows Community-Lab researchers to perform L2 experiments. Moreover, we decided to reach our goal using Software Defined Networking(SDN) techniques, due to the attention they received lately and their promise for a complete networking solution. Overall, we propose an architecture that allows researchers to perform L2 experiments in a generic community networks testbed. To prove the feasibility of our architecture we implemented it for Community-Lab using the OpenFlow SDN protocol, enabling researchers to manage the L2 topology of their experiment.

## I. INTRODUCTION

In order to create new infrastructures and technologies that would promote and enhance Wireless Community Networks (CNs) there was the need for an environment where researchers could deploy and test their ideas. For this reason Community-Lab<sup>1</sup> was created, a community networking testbed which runs on top of existing CNs and enable researchers to experiment in an realistic environment.

Working with Community-Lab infrastructure we encountered the need to allow experimenters to perform L2 experiments. To achieve that goal we needed a way to perform L2 topology virtualization and thus present to the researchers not a low level API but a set of abstract and manageable L2 resources. In spite of the wide variety of ways to achieve L2 topology virtualization the most prominent approach is the one of Software Defined Networking (SDN), as we explain later. Network resource virtualization using SDN offers more benefits than simple resource handling. The SDN community is working on a complete solution where except from virtualized resources, users will be provided with a configuration and management plane. This is the reason why we chose OpenFlow Protocol (OFP), an SDN protocol, to be one of the basic components of our architecture.

As a result of all the above, our effort was focused in creating a component for CN testbeds, like Community-Lab,

where experimenters would be able to handle their L2 topology and experiment with it. Furthermore, we want to use the OFP to virtualize the existing L2 resources, targeting a complete virtualization solution. Our approach is a first step towards a generic SDN system for L2 network experiments in CN testbeds. Additionally, handling experiments above L2 could be ideally implemented as a simple extension of our system.

The rest of the paper is organized as follows. Section II provides an introduction to the different technical areas related to this work. Then, section III describes the fundamental challenges related to our work and an overview of the related research. In section IV we present the related work. Section V introduces the architecture of our system. Section VI describes our effort to map the proposed architecture to a prototype implementation. Section VII presents the evaluation of our system. In section VIII we present a more abstract discussion about our system and our choices. Finally, section IX contains the future work we propose and section X concludes this document by summarizing its main points.

## II. BACKGROUND

In this section we present an overview of the technologies which set the base for our work.

### A. Community-Lab

Community-Lab [2] is a CNs testbed, set up by the CONFINE<sup>2</sup> European project. Community-Lab is a testbed over existing CNs where researchers are able to conduct their experiments in order to test their own protocols and applications deploying them on real community networks.

Three main entities exist in Community-Lab, as depicted in 1. The testbed controller maintains information about the users of the testbed, the experiments and the testbed nodes. The testbed gateways allow the inter-CN communication. The testbed nodes are the devices that host the experiments. All these entities are interconnected through the management network. This interconnection is usually established through the underlying CN links. The researcher can create and deploy his experiment through the testbed server and the testbed nodes.

Every node has a community device, through which it connects to the CN, and a research device where the experiments

<sup>1</sup>Community-Lab <http://community-lab.net/>

<sup>2</sup>CONFINE <http://confine-project.eu/>

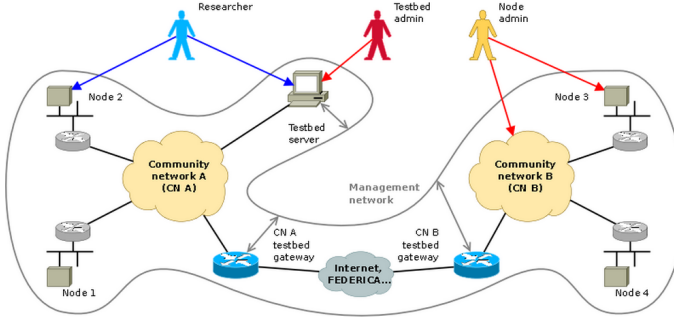


Fig. 1. Community-Lab architecture.

are run. In order to achieve resource sharing in the research device the ideas of slivers and slices from Planetlab<sup>3</sup> are deployed. Each experiment is represented by a slice, which is a collection of slivers. A sliver consists of the isolated resources associated with a slice inside a node.

### B. Software Defined Networking

The SDN approach proposes that all network components should be translated into a set of resources reachable by a defined API. To this date, there exists only one main representative of the SDN approach, the OpenFlow protocol.

OpenFlow [3] (OF) is an SDN protocol, which creates abstractions of L2 network components, like switches. The key idea behind OpenFlow is the separation of control plane and data plane. The control plane is moved from the network resource, here the switch, to a controller. The OF controller provides an API to the user through which the forwarding logic can be implemented. Typically, the switch uses a different communication channel for the data plane and the OFP. The Flow Table stores the flow rules that are sent from the remote controller, which are used during the switching procedure. In a real deployment usually an out-of-band communication channel is used for the OFP communication.

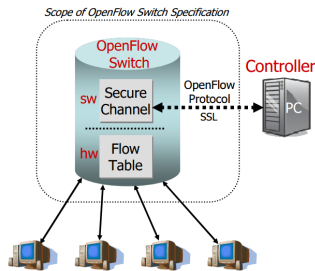


Fig. 2. Typical OpenFlow environment.

Flow control with OF can be either reactive or proactive. In the reactive approach, the switch informs the controller for every new incoming packet that arrives and waits a reply with the action to be taken. In the proactive approach, the controller populates the flow tables of the switches beforehand and is not concerned about every new flow. The proactive approach leads to a small communication overhead and in case of connection failure between the switch and the controller the system can still function properly. On the other hand, the reactive approach

offers a dynamic way of handling flows, increasing though the system latency and the control plane traffic.

## III. CHALLENGES

While deploying the OFP in CNs is a challenging topic, we, additionally, have to resolve problems that are related with the nature of testbeds. In this section we try to enumerate the most important of these challenges. It is worth mentioning that the OFP protocol is designed to manage a homogeneous set of switches that are part of a wired environment.

### A. Wireless Mesh Networks

CNs present many similarities technical with WMNs. Thus, the challenges we present in this section apply also in CNs. The main characteristics of the wireless channels are the origins for many problems faced in WMNs. Here we refer to the most common WMN issues and how they affect our effort.

*Challenge 1: Link Quality Instability:* Wireless link quality is highly unstable, especially in the case of WMNs where distances between nodes can get significant. Kim and Shin in [4] explain that links in WMNs, due to their deployment in large areas and heterogeneous environment, often experience significant quality fluctuations and performance degradation or weak connectivity. This instability is translated to the WMN as a high node churn rate. As a result, we have to assume that in our network there are going to be often topology changes.

*Challenge 2: Link Capacity:* The wireless link capacity is known to be significantly smaller than the one of the wired links, one to two orders of magnitude. Akyildiz and Xudong in [1], do not mention it clearly, but strongly imply that modeling and improving link capacity is one of the major concerns in WMN research. As this problem remains unsolved, WMN routing protocols try to minimize their control plane traffic.

The OFP on the other hand, was designed for wired networks and it does not take into account this kind of restrictions. Using the OFP for managing control plane in WMNs may introduce significant overhead. The often communication between the switches and the controller may both reduce the available bandwidth but also prevent the correct function of the OFP service. Even if the OF is used in proactive mode, when control plane traffic is supposed to be minimal, the communication is often due to the frequent messages sent by the switches to check if the controller is alive.

### B. Community Networks and Community Network Testbeds

CNs share the WMN problems, since they are actually implemented as WMNs, but they also add extra challenges. One of the greater problems in CNs is the diversity of hardware, software and protocols. The difficulty level is further increased in a testbed on top of multiple existing CNs.

*Challenge 3: Device and Protocol Diversity:* A characteristic of WMNs is their heterogeneous nature, as mentioned also before. This is a fact also for CNs but not only in terms of physical links, also in terms of devices as highlighted in [2], by Neumann et al. . This is a great obstacle in finding a global strategy to handle L2 resources. The homogeneity that the OFP assumes increases further the difficulty of the situation.

<sup>3</sup>Planetlab <http://www.planet-lab.org/>

#### *Challenge 4: Communication with Non-Testbed Nodes:*

Inside a CN we can assume that there can exist some kind of homogeneity. For example standard routing protocols that are agreed to be used network-wide or in specific islands of nodes. In a CN testbed created on top of existing CNs, though, new nodes have to be introduced in various locations and various CNs creating the need to manage diverse sets of nodes and protocols, as well as their combination applying minimal or no changes to them [2]. Thus, in a CN testbed, it might be the case that all the testbed nodes are similar but this cannot be claimed for the adjacent CN nodes.

This implication leads to a problem similar with the one faced in *Challenge 3*. For a testbed wide solution, we would need to manage a set of diverse L2 resources, in terms of links and nodes. This is a challenge that complicates the virtualization of L2 resources since it is hard to have a view of the complete L2 topology. Furthermore, the deployment of the OFP is harder since, as mentioned above, the OFP assumes an homogeneous environment with OF compatible nodes.

*Challenge 5: Out-of-band Channels:* When a CN environment is considered, we cannot assume that out-of-band channels are available. As we mentioned earlier in II-B, it is important for the proper functionality of the OFP that it is used in a dedicated channel of communication. Thus, ideally, we would like an out-of-band channel for control plane (OFP) and an in-band channel for the data plane.

## IV. RELATED WORK

Our goal, as stated in I, implies that the first and crucial step is to deploy the OFP in a CN testbed environment. The rest of the problems are going to be handled on top of this infrastructure. As a result, in this section, we present research related with deploying OF in CN-like environments. We have categorized the related work in two main sections: SDN in WMNs and SDN in rural and heterogeneous environments.

Before reviewing the state of the art we would like to mention that no one has addressed the problem of deploying the OFP in an heterogeneous environment like a CN. There exist approaches concerning deploying the OFP in WMNs but considering for example that all the mesh routers involved are OF compatible. Furthermore, most of the efforts are directed in using the OFP protocol to substitute the forwarding plane. In our case, we want OF to experiment in L2. The lack of existing similar approaches makes our effort more challenging but at the same time more innovative.

### A. SDN in Wireless Mesh Networks

The accomplishment most similar to our work, is presented by Dely and Kassler in [5]. This paper describes an architecture for WMNs where OF is used to substitute the forwarding plane, instead of existing routing protocols for WMNs like AODV, OLSR or B.A.T.M.A.N.. The nodes that connect to the mesh split their physical interfaces in two virtual ones, one for the control plane (OF) and one for the data plane. Each virtual interface has a different SSID, achieving that way the separation of control and data plane. The data plane IP connectivity is established through OF rules but the control plane IP connectivity should be established by a normal L3 mesh routing protocol. In their implementation OLSR is used

in order to establish IP connectivity for the control plane virtual interfaces. The OF controller lies in another network which is accessible through the mesh gateways. Furthermore, their architecture was evaluated as an experiment in a real WMN testbed where they identified with two worth mentioning results. Firstly, slow mesh routers can present problematic behavior handling big number of rules. Secondly, using OF instead of OLSR in the forwarding plane when there is a considerable number OF rules introduces significant overhead, which increases proportionally to the number of rules. Dely and Kassler's approach is quite complete taking into account the WMN environment. Their proposed architecture tackles *Challenges 1,2 and 5* but in a different context. Their approach uses the OFP in L3, assuming that all mesh routers are OF compatible, implying a homogeneous environment.

Chung et al. in [6], based mainly on Dely and Kassler's architecture, present their efforts to deploy OF in WMNs. Their work is focused in presenting the difficulties faced, focusing more on hardware obstacles. Though their experiments are not extensive enough, they suggest that Dely and Kassler's approach show acceptable performance compared to using 802.11s in the forwarding plane. More importantly, they conclude, that due to performance issues OF should not be used to substitute mesh networking protocols, but to provide additional functionality in the WMN.

### B. SDN in Rural and Heterogeneous Environments

Hasan et al. in [7], discuss deploying SDN techniques in rural networks in order to allow these networks to operate as infrastructure providers for ISPs. The paper describes the opportunities and challenges involved in this effort. While rural networks present many similarities with CNs, this work approaches the problem abstractly without providing solutions.

A similar but more generic approach was presented in [8], Mendonca et al. The authors provide an abstract description of how should SDN techniques be applied in heterogeneous WMNs. The authors define the main use cases, a set of requirements that heterogeneous SDN would need to enable the use cases. From those requirements they derive a set of research challenges to be confronted. Unfortunately no architecture or technical advices are proposed, but we consider that their effort is important in encouraging an discussion that would lead to a faster establishment of this research domain.

As it is shown above, SDN in WMNs or related fields are quite immature. Also, we can notice, as we claimed before, that there is no actual research in deploying OF in heterogeneous environments. We hope that our effort will be able to contribute in the evolution of this topic.

## V. ARCHITECTURE

To present properly our architecture we should first introduce the environment where the L2 experiments will take place. We assume a testbed architecture similar to Community-Lab, which is a valid assumption since it follows approaches from already existing successful testbeds like Planetlab modified to fulfill the needs of WMNs and CNs. Thus, our environment is formed by testbed nodes, community nodes, a testbed server and testbed gateways. The experiments actually take place in the testbed nodes but the traffic can traverse

also community nodes. The testbed nodes have at least two network interfaces. Through one interface they connect to the testbed management network and also reach the testbed server. Additionally, each node should have at least one for local wireless connections to communicate with adjacent community nodes. The users connect to the testbed server to create and deploy their experiment. When a user wants to create an experiment he creates a new slice and some slivers that belong on this slice and they will run as VMs inside testbed nodes.

#### A. Decisions

Assuming the environment described above, we want to provide the experimenters with the ability to perform L2 experiments using SDN techniques. The specific category of experiments we choose to address is the ability to manage the L2 topology of all the slivers that belong in a slice.

We first present basic infrastructure decisions.

*Decision 1: OF Switches on the host side of testbed nodes:* Since we want OF support for experiments, each sliver should have its own dedicated OF switch. These switches could either be located inside the sliver or in the host. We choose to place the OF switches in the host so that users would use OF without having to care about technical details. Also, this approach allows OF to be used from testbed administrators to manage sliver network resources, for example to deploy QoS policies.

*Decision 2: OF Controller in Testbed Server:* To administer the OF switches we need an OF controller. The OF controller should have connectivity with all the OF switches and hence the related slivers. Furthermore the controller should be accessible from the user as it is the entity through which the switches can be managed. Taking all these into account, together with the fact that our system should be deployed as a testbed service, we decided to place the OF controller in the testbed server. The testbed server is accessible from all the testbed nodes and additionally provides an interface to the user to manage his experiments.

We continue with the functionality decisions.

*Decision 3: L2 mesh routing protocol for multihop L2 connectivity:* The lowest level accessible from the user is L2. Thus, we need to achieve L2 connectivity between the slivers. In an environment as described above the non-testbed (community) nodes make this step non-trivial. Assuming we want to let the experimenters define their own L2 virtual topology for their slivers and in the future even adjust the physical L2 links we need to provide L2 connectivity between the slivers, hence the testbed nodes. As we mentioned before we assume that the node has at least two network interfaces, a management and a local one.

If the management interface is used we would have to create a L2 overlay over the existing management overlay. In case the management overlay functions on L2, it already introduces significant traffic overhead in the CN. In case the management overlay functions on L3, like in Community-Lab, L2 tunnels could be created on top of the L3 overlay. That way, we would create a L2 overlay on top of the L3 management overlay. This approach, depicted on the left stack in the figure 3, has several drawbacks. First, we would have to use the management connection both for control plane and

data plane. Some isolation mechanism would have to be used to separate the different planes requiring modifications to the existing testbed infrastructure and significantly affecting the testbed performance. Second, with this approach there is no direct mapping between the service L2 topology and the actual L2 topology, which would affect the quality and realism of the users experiments.

The second option involves exploiting the local network interface. This approach is more sensible since we do not have to build a L3 network and then degrade it to L2 using an overlay. On the other hand there exists the problem of establishing L2 connectivity with nodes where direct connection does not exist, taking into account the diversity of existing nodes. To make the testbed realistic, the traffic should traverse non testbed community nodes. Nevertheless, the CN environment allows us to assume that there will be islands where mesh routing protocols are deployed. Thus, we require that the slivers of the experiment are placed in nodes that belong to the same island. What we would ideally need is a mesh routing protocol that provides connectivity in L2, unlike OLSR or BMX6 which are the most widespread protocols but function on L3. At the current moment, this can be achieved by two protocols, IEEE 802.11s and B.A.T.M.A.N.-advanced. A L2 mesh routing protocol creates ideally a fully connected L2 network of the participating nodes. So at the IP layer, every packet sending operation is seen as a one-hop communication. These kind of overlays are also called L2.5 overlays. This approach is depicted on the right stack in the figure 3. With this approach there exists a trivial mapping between the upper L2 overlay and the physical nodes, and a more difficult one, but of reasonable complexity between the upper L2 overlay and the physical links. Also, performance-wise, we delegate the responsibility to the L2 mesh routing protocol.

For all the reasons described above we decided to follow the second approach.

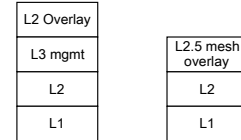


Fig. 3. The two possible approaches for achieving L2 connectivity.

*Decision 4: Control plane through management interface, data plane through local interface:* This decision is a derivative of the previous decision, we wanted though to state it more clearly. The control plane communication will take place locally between the OF switch and the local proxy OF controller and remotely between the central OF controller in the testbed server and the local proxy controller. The remote communication will take place through the management network interface, since this is the interface the nodes use to communicate with the testbed server. The data plane communication will take place through the local interface, through which the testbed node can access other community nodes in the same island through the L2.5 overlay.

Finally, we present the optimization decisions.

*Decision 5: Use OF in proactive mode:* As we presented in section IV, using the OFP in the control plane can create significant overhead to the system, depending on the number

of flow rules. Moreover, in section II-B, we mentioned that the OF controller can be programmed to act in proactive or reactive mode. Using OF in proactive mode can significantly reduce control plane traffic. Furthermore, a proactive approach does not eliminate the dynamic properties of the system, since the controller can still decide to modify rules based on inputs different than the switches incoming traffic.

*Decision 6: Local Proxy OF controller in testbed nodes:*

Using OF in proactive mode we minimize the control plane communication, that concerns exchanging flow rules. Nevertheless, the communication between the OF switch and the OF controller contains also some type of management plane communication. The switch periodically sends requests to the controller to ensure that he is alive. Moreover, the controller periodically requests statistics from the switches. To minimize the overhead of this type of communication we introduce a new idea in the SDN community, the idea of the proxy OF controller. The proxy OF controller is collocated with the OF switch and performs all these tasks in a local level. Thus, the switch does not know the health of the central controller and the statistics are gathered by the local proxy controllers. Since we have actually decoupled the switches from the central controller we can design our own asynchronous communication system to do health checks and gather statistics.

Taking all the previous decisions into account we present our architecture in the figure 4.

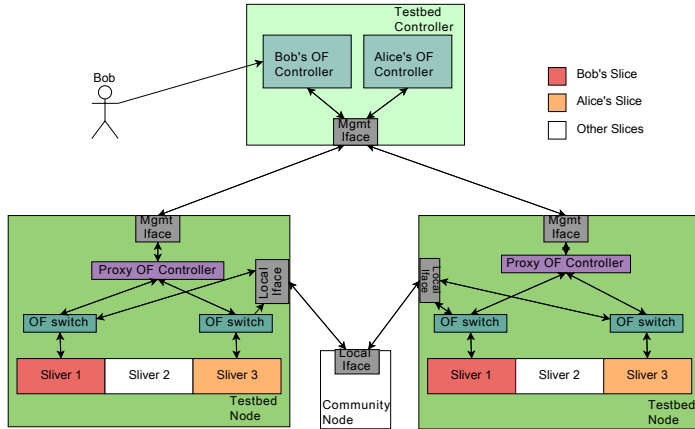


Fig. 4. Overview of the architecture.

## VI. IMPLEMENTATION

Our implementation consists of two main components. Proxy, a proxy for the POX OF controller and Pongo, an attempt to integrate POX, Django and Community-Lab. Proxy plays the role of the local proxy as described in V. Pongo, on the other hand, is a centralized service that corresponds to the specifications described in V. The two components are totally independent but are able to communicate using the OFP.

These software components together with Open vSwitch<sup>4</sup>, a software OF switch, batman-adv<sup>5</sup>, a Linux kernel module that implements the B.A.T.M.A.N. advanced protocol, and the CONFINE software collection<sup>6</sup> are technologies that helped us build a proof-of-concept prototype system.

### A. Proxy - A proxy for POX OF controller

Proxy is a proxy for the controller-switch OFP connection based on POX<sup>7</sup> OF controller. Proxy gives the ability to place a basic or custom POX OF controller in the middle of every OF connection since it acts as a transparent proxy. Based on the OF protocol, Proxy is not restricted by which OF controller or which OF switch lies on the other side. As shown in figure 5, Proxy acts as a controller on the switch side and as a switch on the controller side. Proxy is consisted of two main components which are presented below.

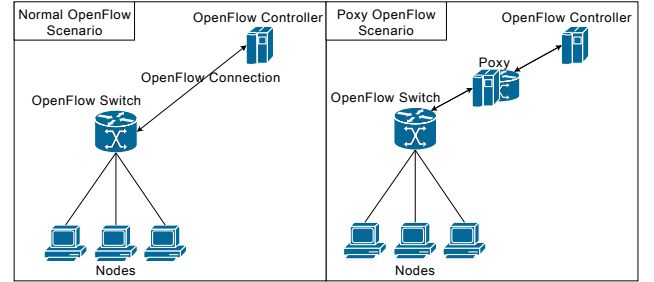


Fig. 5. Basic idea of Proxy

1) *A local OF Controller:* This part of Proxy is used to communicate with switches. Proxy uses as its OF controller the existing basic OF controller of POX overriding some of its functionality. This controller is responsible to establish the connection with the switch and propagate to the switch the messages that arrive forwarded from the proxy. At the same time, it listens for incoming packets from the switches and it propagates them to the proxy. Furthermore, the controller maintains all the information about the connected switches, their features and their state.

2) *A Proxy:* The proxy implements the main behavior of Proxy. On the one end, it forwards packets from the OF switch to the remote OF controller. This is achieved by receiving and redirecting the packets that arrive to the local OF controller through the OF connection. On the other end, the proxy has to simulate a switch connection to the remote controller. An OF controller does normally not use information from protocols of network layers lower than the layer that OFP works (though it is able) so a connection with a switch can be totally simulated without being an actual connection to the switch.

### B. Pongo - An integration of POX, Django and Community-Lab

Pongo<sup>8</sup> is an attempt to integrate POX with Django<sup>9</sup> in order to administer L2 experiments in a collection of nodes. Pongo is a Python software component built on top of POX OF controller which acts as an extended OpenFlow controller. The extensions are efforts to integrate POX with Django and Community-Lab. Pongo provides a very simple interactive Django web interface to Community-Lab researchers that wish to perform experiments using our system.

The main functionality of Pongo lies in allowing the user to define his own L2 topology adding and deleting links

<sup>4</sup>Open vSwitch <http://openvswitch.org/>

<sup>5</sup>batman-adv <http://www.open-mesh.org/projects/batman-adv/wiki/>

<sup>6</sup>CONFINE Software System <http://redmine.confine-project.eu/>

<sup>7</sup>POX <http://www.noxrepo.org/pox/about-pox/>

<sup>8</sup>Pongo <https://github.com/emmdim/Pongo>

<sup>9</sup>Django <https://www.djangoproject.com/>



between slivers through the web interface. The choices of the links are translated to OFP rules and through POX they are propagated to the switches. This feature assumes that L2 connectivity between the switches already exists. Through the web interface, a researcher can see his slivers and the corresponding OF switches. Moreover he can see the existing connections between the OF switches.

### C. L2 experiments in Community-Lab

All the software components and tools described above are combined in order to build a system for L2 experiments in for a CN testbed. The CN testbed is Community Lab and we are using an architecture which is depicted in figure 6. The two main entities are the controller and the node.

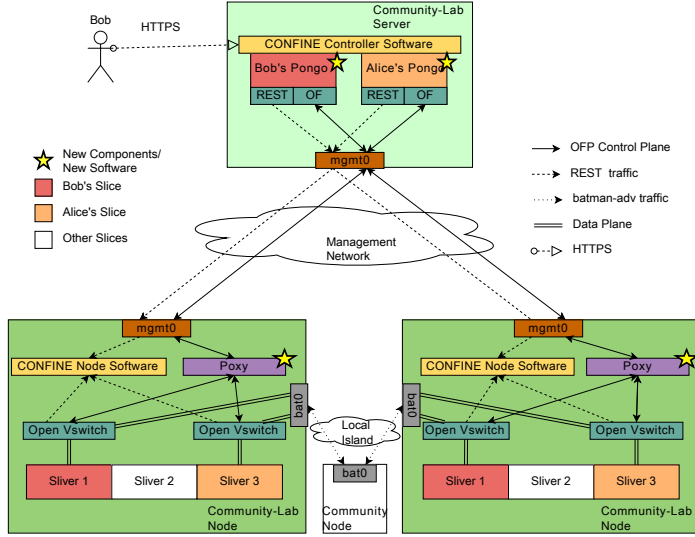


Fig. 6. Overview of the implementation design.

1) *Controller*: The controller is based on the CONFINE Controller software. The controller runs a different Pongo instance for each of the researchers that wish to conduct experiments using our system. The user can access the web UI through the web interface of the CONFINE Controller software. Pongo communicates with each node's CONFINE node software system making HTTP requests to it. Also Pongo communicates with the Open vSwitches located on the nodes through Poxy using the OFP.

2) *Nodes*: The nodes are based on the CONFINE Node software system. Each node runs one Poxy instance which can be used by different experiments at the same time. Each user's sliver has a dedicated Open vSwitch. The main Poxy application does not forward traffic to multiple OF controllers, thus an extension was added that implements the idea of different users. The OFP traffic, the control plane, is routed through the management interface from Pongo. The data plane of the virtual switches is routed through the bat0 interface which is the virtual interface created by batman-adv. This way the separation of control plane and data plane is achieved. The nodes communicate with each other through the batman-adv overlay. For this, we have to assume that all the nodes of an experiment that uses our system run to batman-adv island. This assumption implies that the OF switches of the researcher are fully connected between them. Thus, when a new switch is

connected to Pongo all the links with the rest switches of the slice are assumed to exist. Any data plane traffic is finally propagated through Open vSwitch inside the slivers.

Assuming the architecture depicted in 6 expanded in three Community-lab nodes, we present in figure 7 how users see their topology: Bob (on the left) and Alice (on the right).

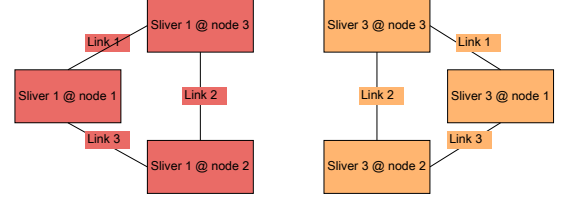


Fig. 7. User view of the topology.

## VII. EVALUATION

Considering that we have developed a platform there exist two evaluation subjects: functionality and performance.

### A. Functional Evaluation

We managed to set up our system in a virtualized environment with three testbed nodes and one testbed server. Then we set up our Pongo in the testbed server, the Open vSwitches, batman-adv and Poxy in each of the nodes. After the installation, we had a view of our slivers and their connections from the web interface of Pongo. Then we created traffic between slivers which is routed successfully. Finally, by dropping the link from the Pongo web interface the traffic is blocked, while recreating the link the traffic is resumed.

### B. Performance Analysis

In this section we present a performance analysis of our system, tracing the bottlenecks and the performance critical parts. Based on this analysis we propose a set of experiments that would, in our opinion, support our arguments. We describe separately our analysis concerning the communication overhead and the computation overhead. It is left for future work to perform the evaluation performing the experiments.

1) *Communication Overhead*: There are two critical areas for the communication overhead, the management network and the local island, as those are depicted in figure 6.

The first network of our interest is the management overlay. We want this communication to be very minimal, so that it won't interfere with the proper functionality of the testbed. The REST communication, that performs sliver info retrieval from the node, happens only once for each node, when the OF switch connects to the corresponding PONGO, so it is negligible. Moreover, OF is used in proactive mode so the communication is necessary only when a switch is connected or when the user decides to remove/add links. No other communication overhead is produced from our system in this area guaranteeing. Experimenting with network overhead in this area is unnecessary as our system guarantees almost constant amount of message exchanges.

The second interesting area contains the communication that takes place in the island of the L2 mesh routing protocol.

In this communication part there are no restrictions concerning interfering with the testbed or the community network. On the other hand, this part can easily become the bottleneck of our system. As we suggested in *Decision 3* in section V, we delegate the performance concerns of these communication channels to the L2 mesh routing protocol. The implementations of L2 mesh routing protocols have improved significantly over the last years, but it is our opinion that they are still not ready to support hundreds of nodes and large amount of traffic. Thus, the throughput of batman-adv links may degrade the performance of our system. We assume that the throughput decreases as the distance (in number of hops) increases. Also the control plane used by batman-adv adds overhead to the system but the implementation is designed in order to minimize this overhead. Performing some trivial file transfer experiments for one hop distance with and without batman-adv, in a wired-like environment we found that the throughput is not decreased significantly in the latter case. In general we expect that batman-adv itself does not introduce the overhead, but the wireless nature (for example wireless interference) that creates the problems, even though we cannot expect the same performance from a L2 multihop environment (virtual L2) and an L2 collision domain. A simple experiment could be performed, where our system is used on top of a switched wireless network (WLAN for example) compared with usage on top of a batman-adv network in a WMN environment. Furthermore, experiments that indicate the relation between number of nodes and throughput could be conducted.

2) *Computation Overhead*: Computationally, our system is separated in the server and the nodes.

The Pongo applications run on the server, which is located on the testbed controller. As it is depicted in the figure 6 and was explained in VI, there can exist multiple Pongo applications, one for each user. The testbed server is designed to run on powerful server hardware, so the computation overhead introduced by the Pongo applications is not important. Additionally, our choice to run OF in proactive mode makes the computation overhead even smaller. No experiments are necessary for the computation overhead of the server part.

The nodes on the other hand, according to Community-Lab specifications, are low performance and low cost devices. For that reason, we chose to have only one Poxy application running on each node, as shown in figure 6. The communication between the Open vSwitches and Poxy happens using traditional Unix sockets which run on the user space. Nevertheless, this overhead is not that important, since it only involves some message exchanges every few seconds. On the other hand, the packet switching in the Open vSwitches and in the real network interface that lies below the batman-adv interface could be computationally intensive. The overhead introduced by the switches depends almost only on the amount of data traffic, which is natural and cannot be changed. As a result the only troubling point, is packet handling for the batman-adv control plane. As we explained previously, this concerns the performance and capabilities of L2 mesh routing protocols. The most interesting experiment concerning the computation overhead in the node would be measuring the overhead caused by the physical batman-adv interface and comparing it with the actual data (payload) that are transmitted or received.

## VIII. DISCUSSION

In this section we review our system and argue about the success of our decisions and our general approach.

### A. Tackling the Challenges

The main assumption of our architecture was that the environment of the testbed has some similar properties with Community-Lab, which are reasonable. Under this assumption we describe how our architectural decisions, presented in V, tackle the main challenges as they are described in III.

*Solution for Challenge 1 - Link Quality Instability*: The actual problem caused by this challenge was the frequent changes in the topology. In our case we actually care about the testbed nodes. Nevertheless, their L2 connectivity is affected by topology changes. Therefore, we let our L2 mesh protocol handle both the changes and the instabilities. Thus, *Decision 3* resolves this challenge.

*Solution for Challenge 2 - Link Capacity*: Using OF in proactive mode and introducing the entity of OF proxy controller significantly reduce the link usage by the OFP. Thus, *Decisions 5,6* resolve this challenge.

*Solution for Challenge 3 - Device and Protocol Diversity*: The assumption that the testbed nodes will be located in the same island with a common L2 mesh routing protocol are enough to overcome the existing diversity. Thus, *Decision 3* resolves this challenge, though significantly restricting the possible application environments.

*Solution for Challenge 4 - Communication with Non-Testbed Nodes*: Addressed in a way similar to *Challenge 3*. *Decision 3* resolves this challenge.

*Solution for Challenge 5 - Out-of-band Channels*: As we described before, we use totally different interfaces to achieve data plane and control plane communication. Thus, *Decision 4* resolves this challenge.

Therefore, we can claim that we succeeded in overcoming the challenges and building an architecture that can provide the required functionality.

### B. Distributed properties of the system

Despite our achievements, it is worth to mention that our system is not fault-tolerant. There are two kinds of fault-tolerance that could exist in our system, component oriented and network oriented. A component failure could be either a failure that concerns the controller or a failure on the node side. On the node side, if the sliver or the node fails the OF switches are able to recover the rules upon the component recovery. Thus, any failure related to the nodes can be surpassed. On the other hand, the controller is a single point of failure and maintains no state. As a result, if the controllers fails the system cannot recover. Network failure mainly refers to the communication channels failure. In general, simple link failures do not affect our system because in a CN environment there are usually multiple routes and the L2 mesh routing protocol can also handle them. Our system, though, will not be able to recover lost update during a network partition. This update may not reach all the relevant switches. An inconsistency on the switches may lead to system to malfunction. Handling

inconsistencies is an interesting topic that we plan to explore in the future. At the current moment, there are no consistency guarantees concerning rule propagation to the OF switches. This raises the need to invent a method to monitor the distributed state of the switches.

### C. Generalization for CNs or WMNs

One of the interesting points for discussion is the possibility and the benefits of applying our architecture not only in CN testbeds, but also normal CNs or WMNs. Concerning WMNs, our architecture would be very similar with the one proposed in [5], but adapted for L2. Our approach could be used only in case the WMN already uses the L2 mesh routing protocol. Similarly, for CNs our architecture could prove useful in islands that have deployed L2 mesh routing protocols. The separation of control plane and data plane should be implemented like proposed in [5], breaking each physical interface into two virtual ones. Additionally, our contribution of having a proxy OpenFlow controller would be really helpful taking into account the distributed nature of the environment.

There exists, though, a theoretical contradiction. Routing would be already implemented into L2, why would a forwarding plane be needed? There is no doubt that OF and SDN can provide really interesting applications for CNs and WMNs, like described in [8], but what would we earn in case OF was functioning in L2 instead of L3? We believe that especially in the case OF could interact with the L2 mesh routing protocol there are a lot of interesting applications. Most importantly, if a system of distributed OF controllers was developed, it could provide full control in L2 CN-wide or mesh-wide. Such a claim is very strong but we believe it is valid under the assumption we stated about OF communication with the mesh protocol. Nevertheless, we should consider the performance cost. Therefore, the answer lies on the tradeoff between the desired functionality and the performance overhead.

## IX. FUTURE WORK

The next step would be to complete the evaluation by performing the experiments we described in VII. We believe that these experiments will confirm our hypotheses. Another interesting approach would be to research the consistency of the system by checking the distributed state of the switches, as proposed in VIII. The safety and the progress of the system should also be guaranteed in a technical way. Another interesting future approach would be to detach the OF service from the controller server and place it in multiple servers throughout the CN using a distributed communication protocol between OF controllers. This approach seems very challenging but very promising for future CNs, as it would lead to scenarios similar to the ones described in VIII-C. A different approach would be to explore and deploy more the SDN capabilities. For example new L2 (or upper layer) use cases could be developed based on the current OF infrastructure and newer versions of OF could be exploited. Finally, in order to exploit the full advantages of using SDN, it is in our future interests to explore the necessity and feasibility of having a management plane, like OFCONFIG<sup>10</sup> or OVSDB<sup>11</sup>. That would create a

complete platform for performing experiments in a CN testbed but also would allow us to test whether it can be used for the infrastructure of the testbed itself.

## X. CONCLUSIONS

Throughout this document we presented how we achieved to create a system that allows L2 experiments in CN testbeds using SDN techniques. We proposed a generic architecture that fulfills our goal and implemented this architecture for an existing CN testbed, Community-Lab. Our implementation consists of two, FOSS licensed, software components. First, we developed Poxo, a proxy for OF traffic. Second, we implemented Pongo, an integration of POX OF controller, Community-Lab testbed server and Django. Using these software components we deployed the proposed architecture in Community-Lab, creating an application that allows managing the L2 topology of a researcher's nodes. Thus, we proved the feasibility of our architecture and the proper function of our implementation. Moreover, we performed a performance analysis of our system reaching the conclusion that the wireless multihop environment and the use of L2 mesh routing protocol are the greatest cause of overhead. Finally we discussed how our effort satisfies our goals and under which conditions our architecture could be used in a generic CN or WMN environment.

## ACKNOWLEDGMENTS

This work is supported by the European Community Framework Programme 7 within the Future Internet Research and Experimentation Initiative (FIRE), Community Networks Testbed for the Future Internet (CONFINE), contract FP7-288535.

## REFERENCES

- [1] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, pp. S23–S30, 2005.
- [2] A. Neumann, I. Vilata, X. Leon, P. Garcia, L. Navarro, and E. Lopez, "Community-lab: Architecture of a community networking testbed for the future internet," in *Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2012 IEEE 8th International Conference on, pp. 620–627, 2012.
- [3] 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, pp. 69–74, Mar. 2008.
- [4] K.-H. Kim and K. G. Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, MobiCom '06, (New York, NY, USA), pp. 38–49, ACM, 2006.
- [5] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks," in *Computer Communications and Networks (ICCCN)*, 2011 Proceedings of 20th International Conference on, pp. 1–6, 2011.
- [6] J. Chung, G. Gonzalez, I. Armuelles, T. Robles, R. Alcarria, and A. Morales, "Experiences and challenges in deploying openflow over a real wireless mesh network," in *Communications (LATINCOM)*, 2012 IEEE Latin-America Conference on, pp. 1–5, 2012.
- [7] S. Hasan, Y. Ben-David, C. Scott, E. Brewer, and S. Shenker, "Enhancing rural connectivity with software defined networks," in *Proceedings of the 3rd ACM Symposium on Computing for Development*, ACM DEV '13, (New York, NY, USA), pp. 49:1–49:2, ACM, 2013.
- [8] M. Mendonca, K. Obraczka, and T. Tuletli, "The case for software-defined networking in heterogeneous networked environments," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, CoNEXT Student '12, (New York, NY, USA), pp. 59–60, ACM, 2012.

<sup>10</sup>OFCONFIG <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.1.pdf>

<sup>11</sup>OVSDB <http://tools.ietf.org/html/draft-pfaff-ovsdb-proto-02>