SEVENTH FRAMEWORK PROGRAMME

Theme 3

Information and Communication Technologies

Deliverable D3.9

EXPRESS requirements, initial functional specification and overall design

Grant Agreement number: 287581

Project acronym: OpenLab

Project title: OpenLab: Extending FIRE testbeds and tools

Funding Scheme: Large scale integrating project (IP)

Project website address: www.ict-openlab.eu

Date of preparation of deliverable: 30/11/2013

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document properties

Document responsible:	Stefano Salsano (CNIT)
Author(s)/editor(s):	Stefano Salsano (editor), Nicola Blefari-Melazzi, Giuseppe Bianchi, Luca Veltri, Andrea Detti, Claudio Pisa All authors are affiliated to CNIT
Version:	1.0

Abstract:

The main objective of EXPRESS is designing an innovative, resilient SDN system capable to extend the SDN applicability domain from fixed networks to intermittently connected network, like wireless mesh networks. The EXPRESS solution will be implemented and then deployed in an experiment involving the three OpenLab testbeds NITOS, W-iLab.t and PlanetLab Europe. In this document we define the requirements for the EXPRESS systems, both in terms of architecture and functionality and in terms of deployment over the OpenLab testbed. This document include the high level design of the solution. EXPRESS will use an SDN approach based on the OpenFlow protocol for the controller-to-switch communication. EXPRESS will rely on OLSR distributed routing protocol to setup the control plane for the controller-to-switch communication and the communication among the controllers. A distributed controller solution based on a hierarchy of controllers will be designed, capable to handle network partitioning and merging. On the data plane, the SDN/OpenFlow approach will be capable to support advanced routing strategies, going beyond the shortest-path routing provided by OLSR.

Table of Contents

1 Introduction

The main objective of EXPRESS (EXPerimenting and Researching Evolutions of Software-defined networking over federated test-bedS) is designing an innovative, resilient SDN system capable to extend the SDN applicability domain from fixed networks to intermittently connected network, like wireless mesh networks. The system will be able to withstand attacks, failures, mistakes, natural disasters and able to keep operating also in fragmented and intermittently connected networks. Such a system will also be able to easily glue together separated networks or to form networks (e.g. mesh networks). In this scenario SDN has pros (ability to quickly update the forwarding plane) and cons (single point of failure, remote control). We aim at exploiting the pros while mitigating the cons by improving promptness and robustness of the SDN infrastructure. In a way, this project is investigating on the limits of the SDN approach, because the wireless and intermittently connected scenario that we are considering is highly challenging for the SDN approach based on the centralization of control functionality.

In our reference scenario, wireless infrastructure nodes, or Wireless Mesh Routers (WMR) equipped with SDN, build up a set of Wireless Mesh Networks (WMNs). Different wireless mesh networks are interconnected by a set of wired SDN nodes (see Fig. 1), forming the core network. This topology is typical of so called “community networks” and of wireless mesh providers. In this project we address the issues related to using SDN over such a severe network environment, assuming that the failure of network links is not a rare event and the network could even become partitioned. In these conditions we aim at providing a resilient SDN control-plane, which is the foundation for any data-plane restoring mechanism.

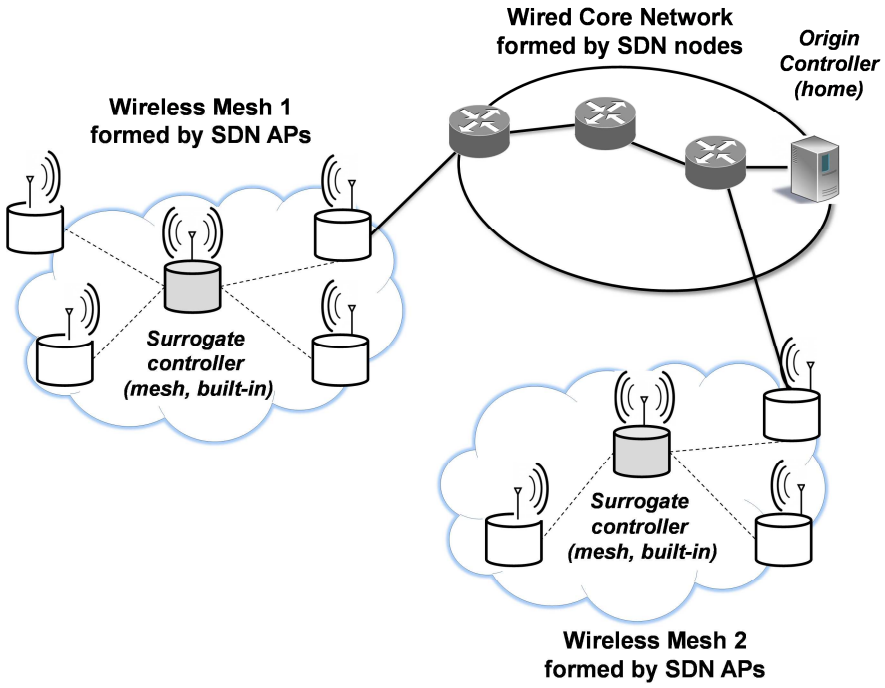

Fig. 1 – Reference scenario: a provider or community network composed of a set of Wireless Mesh Networks

The proposed solution will need to deal with the setup and maintenance of the routing functionality necessary to discover the network topology and install and maintain the routing protocols, and for the setup and maintenance of the SDN controller(s) and of the communication between the controllers and the switches. Once the infrastructure is up and running, SDN will allow to change the device configurations, if needed; to install/update policies for access control, tailored to specific environments and categories of users, or traffic engineering rules; to program security or launch monitoring actions, as a function of anomaly detection warnings; to offer a large set of virtualization functionalities.

The second objective of EXPRESS is to implement a demonstrator of the defined SDN solution, in representatives scenarios of real world environments such as wireless mesh community networks and wireless mesh networks. To deploy our demonstrator, we will exploit a federation of three OpenLab testbeds (PlanetLab, NITOS and W-iLab.t), where PlanetLab plays the role of the core fixed backbone and NITOS and W-iLab.t play the role of the wireless mesh networks.

2 Requirements

2.1 General architectural requirements

In this section we list the general architectural requirements. These requirements are referred to as GRx (General Requirements).

GR1	Express will be able to operate a Wireless Mesh Network (WMN) formed by a set of Wireless Mesh Routers (WMRs) by providing IP connectivity services to a set of access networks. The access networks will be connected to the WMRs via wired or wireless interfaces.
GR2	Express will be able to operate a set of Wireless Mesh Network as above defined, interconnected by a wired backbone, providing IP connectivity services to a set of access networks.
GR3	Express will consider fragmented network operations: • each WMN can run in isolation or can be connected with the backbone • a WMN in turn can be fragmented in different parts • two WMN fragments can merge into a bigger fragment (or into the whole WMN)
GR4	Express will use an SDN approach based on the OpenFlow protocol to configure the operations of the WMRs and control the behaviour of the IP data plane.
GR5	Express will assume that different controllers can take control of parts of the network and then merge back into a single network, when partitions are removed. Therefore different controllers do not necessarily share the same vision of the network. Different nodes may need to take the controller role and start controlling an isolated portion of the network. Simplified operations can be considered in these circumstances, with the goal of providing at least emergency operations (highest priority services), and then support lower priority services only if possible.
GR6	IP routing protocols like AODV, BATMAN or OLSR are typically used to create IP connectivity in the mesh. With these protocols, it is hard to implement advanced features like flow based routing or load balancing. The Express SDN approach will offer the possibility to implement fine grained control on the use of resources. (NB Express work will not focus on finding optimal solutions but to provide the basic tools to implement such features).
GR7	On the Data Plane, IP connectivity services will be offered to hosts in the access networks. The proposed architecture will not support layer 2 connectivity: the Wireless Mesh & Access node will act as an IP router for the clients.

GR8	We assume that the system can face transient issues leading to temporary loss of connectivity. Such failures needs to be recovered in time frames comparable with convergence time of distributed routing protocols like OLSR. In other words, Express does not aim at faster convergence times than the existing distributed routing protocols. The SDN approach will be used to offer more flexibility and more advanced services in the data plane (e.g. traffic engineering, load balancing).

2.2 Requirements related to OpenLab testbeds

In this section we list the requirements related to the experiments over the OpenLab testbeds. These requirements are referred to as TRx (Testbed Requirements).

TR1	The experimental Wireless Mesh Network of Express will be deployed over both the NITOS and W-iLab.t wireless testbeds
TR2	Express will perform preliminary experiments over isolated wireless testbeds (NITOS and/or W-iLab.t), then it will perform experiments over a federated testbed.
TR3	The backbone network interconnecting the two experimental Wireless Mesh Network will be the OpenLab SDN testbed over PlanetLab Europe, as described in OpenLab Deliverable D4.3 “OpenFlow enhancements for PLE”
TR4	The Express experiments will start by creating separate logins on the different testbeds, then if possible Express will try to use the myslice federated solution ().

3 Initial functional specification and overall design

Taking into account the above outlined requirements, the EXPRESS approach will be based on a control plane IP connectivity realized using the OLSR routing protocol. We will deploy the SDN mechanisms on top of the control plane IP connectivity: the communication among the controllers will use the control plan IP connectivity, as well as the communication between the controllers and the OpenFlow switches. Enhancement of the OLSR routing protocol will be designed to support the SDN approach.

We will design the operation of the Wireless Mesh Routers in order to support the exchange of OLSR messages and then the coexistence of control plane traffic and data plane traffic handled using SDN/OpenFlow mechanisms.

3.1 High level architecture

The proposed approach for using SDN to control the wireless meshes is to use an IP ad hoc routing protocol (OLSR) among the nodes of the mesh to establish a basic IP connectivity (see Fig. 2). Such connectivity will constitute the control plane and will support all controller-to-switch OpenFlow messages as well as controller-to-controller messages that are needed to coordinate the allocation of "master" controller role for the switches. The use of OLSR ensures the proper reaction to changing topology events, like addition/removals of mesh nodes and wireless links among them. The data plane will be based on an OpenFlow approach.

As for the wireless channels, we use a single SSID for both the control traffic and the data traffic, therefore we can classify it as an "in-band" control strategy from the OpenFlow protocol perspective.

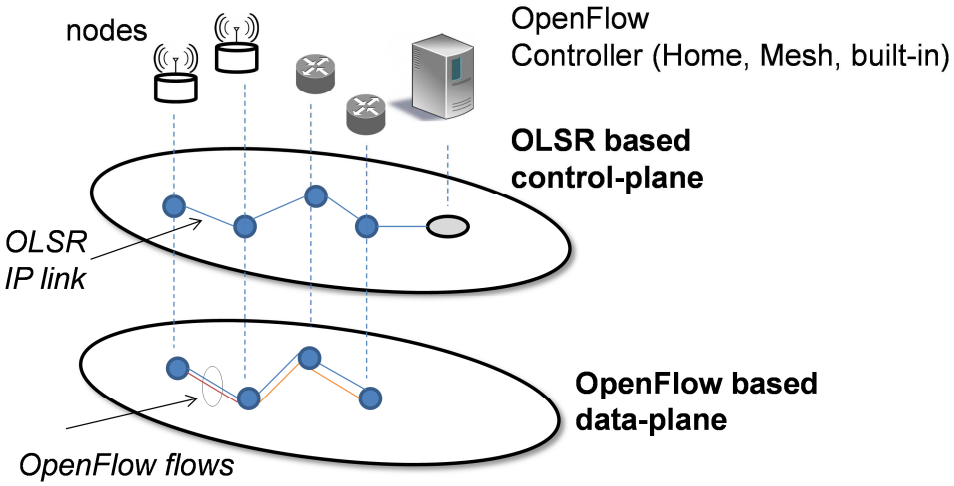

Fig. 2 – Control and data planes

Each Wireless Mesh Router will run an OpenFlow switch. Over the control plane, the OpenFlow switch will try to contact a set of default controllers: the Home Controller (HC), running in the fixed network, and the Mesh Network Main Controller (MMC), i.e. the main controller in the mesh. Moreover the switch can use other "lower priority" Mesh controllers that can take over in case all other controllers are not reachable, these "lower priority" Mesh controllers can implement a subset of the full OpenFlow-based available services, they will be referred as SMC – Secondary Mesh Controllers. A switch node will also have a *built-in* controller located in the switch itself for handling emergency services or, more in general, partitioned networks without a pre-defined Home/Mesh controller. This built-in controller does not need to be a full compliant OpenFlow controller, rather it is a process that is able to inject OpenFlow rules in the local OpenFlow switch. We will call this entity "EFTM - Embedded Flow Table Manager". The hierarchy of controllers and flow table manager is shown in Fig. 3.

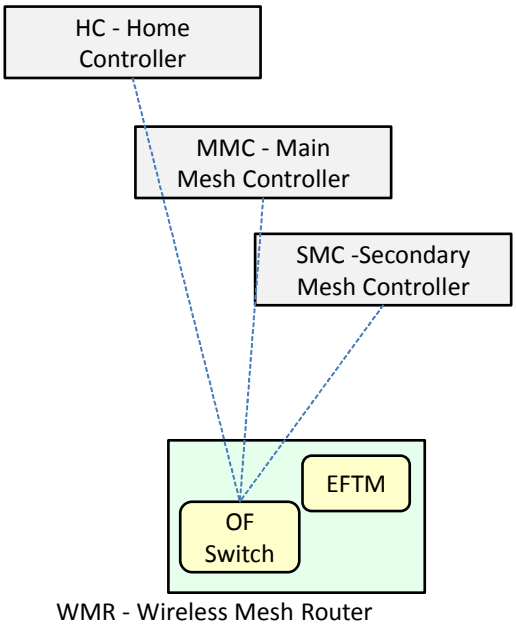

Fig. 3 – Hierarchy of Controllers and Embedded Flow Table Manager

The OpenFlow controllers in the hierarchy can be used to engineer the routing of data traffic, while OLSR is used to locally set up the control-rules used by OpenFlow control traffic, through the use of the EFTM entity. Moreover, OLSR is also used to push *emergency-rules* in the switch. Such rules route data traffic in emergency conditions, during which the OpenFlow controller fails or is unreachable.

The controllers in the hierarchy coordinate each other, supporting controller failures and dynamic topology modification, including network partitioning and joining.

The high level architecture of a Wireless Mesh Router node, showing the interplay between the OLSR protocol and the EFTM entity is shown in Fig. 4. The OpenFlow switch in the WMR is configured by the EFTM so that the OLSR daemon can send and receive OLSR packets over the wireless interfaces. The OLSR daemon provides topology information to the EFTM, which is needed to set up

the flow entries for the control traffic. The OF switch interacts with the OpenFlow controllers that can configure the flow table for data plane flows.

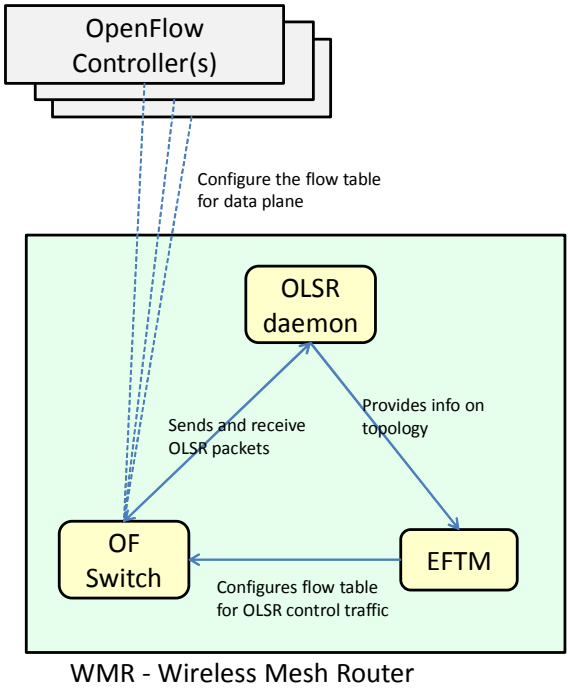

Fig. 4 – WMR node architecture

It is noteworthy that in this architecture we have two different levels of entities setting up OpenFlow rules: a local distributed manager (the ETFM) taking care of control-rules, and the set of remote controllers taking care of rules for data traffic.

3.2 Detailed design of networking and of WMR node internals

The reference network scenario is shown in Fig. 5. A WMN is composed of Wireless Mesh Routers (WMRs) which provide connectivity to a set of Access networks (either offering a wired or wireless interface to user terminals). A subset of the WMRs operate as Gateways and provide connectivity towards the wired backbone. A set of OpenFlow controllers can operate in the wireless mesh (indicated as Main Mesh Controller and Secondary Mesh controller), connected to a WMR through a wireless/wired connection. The Home Controller is on the wired backbone.

Within the whole control network, each controller is uniquely identified by its IP address in the control network range. It means that a private IP range will be allocated for the control network and each controller and mesh node will be statically given its IP address. Under these assumptions the control plane connectivity can be built by using OLSR, spanning across the mesh network and the wired core

network. Control traffic and data traffic use different IP subnets. For instance, the subnet 10.0.0.0/16 can be used for control traffic, while other subnets are used for data traffic. The controllers and the WMR wireless interfaces use addresses of the control subnet, while other interfaces of the network get an IP address belonging to different subnets, e.g. 192.168.x.0/24, each announced in OLSR as an "HNA network" (HNA stands for Host and Network Association).

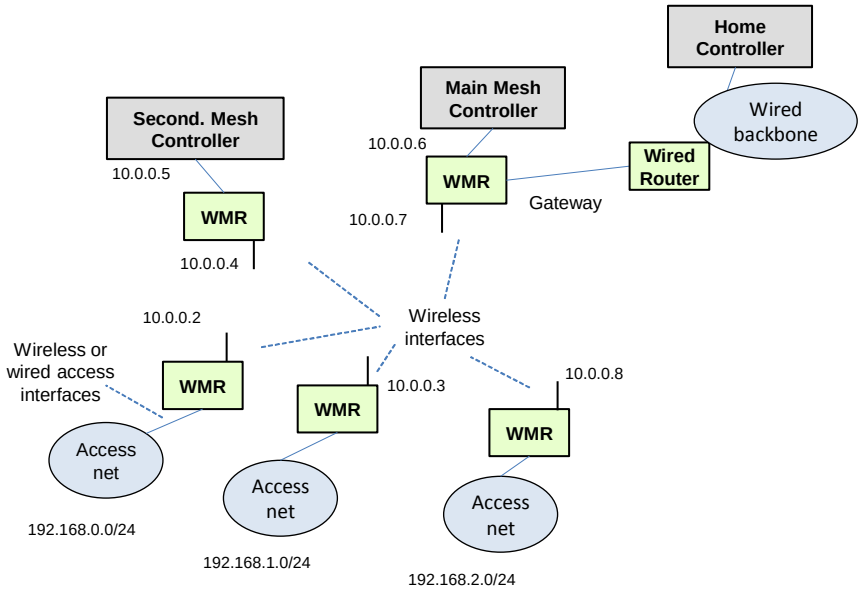

Fig. 5 – Reference network scenario

3.2.1 Control plane forwarding/routing

In our WMR node, we assume that the OpenFlow switch will be directly connected to the wireless interfaces (see Fig. 6). In order to deploy our in-band control mechanism, we need to locally set-up the control-rules to forward OpenFlow control packets, which are packets with destination IP address belonging to the control-subnet. To this aim we use the OLSR routing protocol to learn the topology of the control-subnet and then exploit this knowledge to setup the control-rules. Accordingly, an OLSR routing instance runs on each WMR node and the IP address of the controller is also advertised by OLSR using a Host and Network Association (HNA) messages with /32 mask.

Fig. 6 reports the main entities of a WMR involved in the interplay between OLSR and OpenFlow. The control-rules used by OpenFlow message are configured by a module running in the ETFM called OLSR-to-OpenFlow (O_2O). O_2O module operates by inspecting an IP routing table handled by the OLSR daemon. This IP routing table is a "dummy" table, i.e. not actually used by the operating system when forwarding IP packets. Under the Linux OS used in our implementation, this is a "user defined" routing table, different from the kernel main one, and never referenced in the Routing Policy Data Base.

An entry of the dummy routing table has the form <control-subnet IP address/32, next-hop, output interface>; the O_2O module converts it in a rule of the OpenFlow table whose match is "IP destination == control-subnet IP address" and whose action is "change source MAC address with the MAC

address of outgoing interface and the destination MAC address with the MAC address of the next-hop". Therefore the O_2O module needs to know the MAC addresses of the WMRs; this IP-to-MAC translation can be provided offline or can be distributed by a novel OLSR plug-in, so that each WMR can learn the MAC addresses of all other WMRs. Note that while an ARP or ARP-like mechanism could be viable to learn the MAC addresses of nodes at one-hop distance, the central controller however needs to know all the associations between IP addresses and MAC addresses of all network nodes, therefore the solution of extending the OLSR seems to be the best one.

In order to follow the topology changes, the OpenFlow table should be updated immediately when the OLSR protocol receives the information about the change and re-computes the dummy IP routing table. In order to simplify the implementation, a periodic "polling" procedure can be used to check if some route has changed. The simplest approach foresees to periodically update the Flow Table according to the dummy IP routing table. In this simplified approach, the O_2O module sets a timeout (e.g. 60s) to the inserted control-rules and at the timeout expiration the dummy IP table is dumped again on the OpenFlow Table.

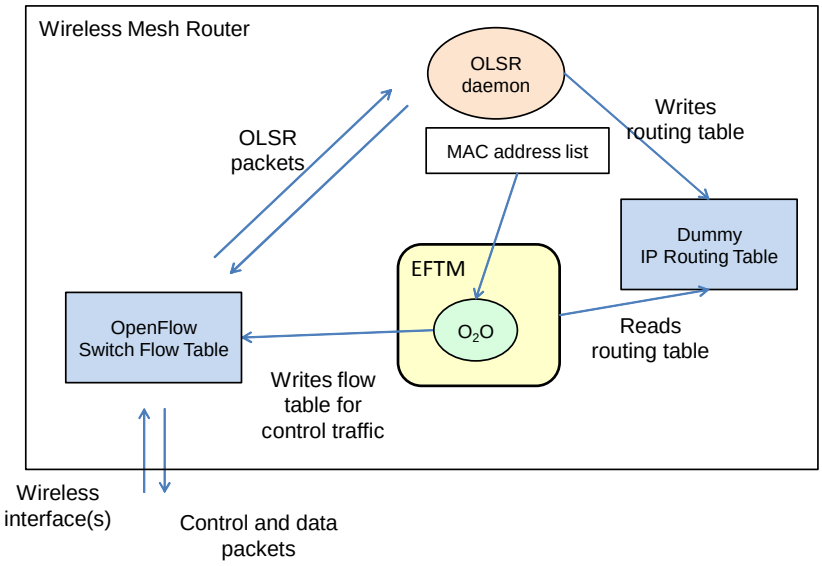

Fig. 6 – OpenFlow and OLSR interaction

In addition to the control-rules used to route OpenFlow traffic, the flow tables are also filled by the EFTM with other control-rules needed to support the OLSR operations. These rules are used to forward the incoming OLSR packets to the OLSR daemon in the WMR node and to let the outgoing OLSR packets exit from the proper interfaces.

3.2.2 Data plane forwarding/routing

Let us now consider how to handle the traffic for IP destinations outside the control-subnet, i.e. either to the access networks or to the Internet. Assume that a packet is generated in a host of the access network and destined to an Internet address outside the wireless mesh network (but the same will also

apply to packets destined to a host of the access network as this occurs when packets come back from the Internet or for mesh internal communications).

The packet will be received by the WMR on its access network interface. Then a match is searched in the flow table. In case a match is found, the related action is carried out. Otherwise, the IP packet is embedded in a OpenFlow packet-in, which is transferred to the controller currently in charge of the WMR using the in-band control network. When the controller receives the packet-in, it can apply the desired routing logic and install data plane entries in the flow table.

To distribute the topology information of the data plane, the IP subnets of the Access Networks are advertised by WMRs and gateway WMRs by using OLSR Host and Network Association (HNA) messages. Moreover, gateway WMRs may also advertise the default route 0.0.0.0/0. With this approach, each WMR node knows the full network topology. The controllers inquiry the connected WMR to learn this topology information, which is fundamental to implement traffic engineering logic for data traffic. This approach is different from the traditional OpenFlow topology discover in wired layer 2 network, performed using LLDP messages [1].

3.2.3 WMR node design

The architecture of a WMR node is shown in Fig. 7. It includes: a wireless interface belonging to the Wireless Mesh Network (wlan0); a virtual interface br0, which is a software bridge using OpenFlow switching logic, e.g. Open vSwitch [2]; an optional wired interface used as a gateway to the wired backbone (ethX in the figure); an optional set of wired or wireless interfaces towards client Access Networks (ethY, wlanZ in the figure. A generic “real” WMR node may additional wireless interfaces toward the WMN that can be bridged to br0 if a multi-channel WMR is used.

The br0 interface has an IP address belonging to the control-subnet, wlan0 does not have an IP address, ethZ and wlanZ have an address of the Access Networks subnet and ethX of the subnet connected to the wired backbone. OLSR is connected to br0, and br0 is used as destination for any packets generated by the node and directed towards the WMN. To this aim we used the solution to insert a fake IP address (e.g. 10.0.254.254) in the main routing table of Linux as gateway of all the routes whose outgoing interface is br0 (i.e. of the routes directed toward the WMN). To avoid ARP generation, we also statically insert in the ARP table a fake MAC address for the fake IP address.

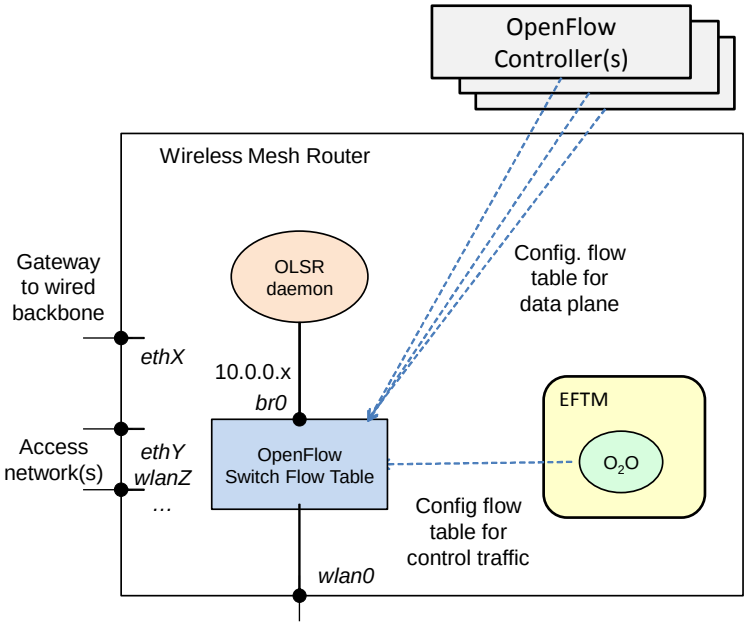

Fig. 7 – Detailed WMR architecture

3.2.4 Emergency conditions

The EFTM entity in the WMR will continuously check if the WMR is connected to an active controller. In case of controller failures (e.g. due to hardware or communication issue) the ETFM will trigger the start of an "emergency condition".

Some policies are needed to decide how to handle data traffic under this emergency condition. It could be needed to classify the data traffic and allow to forward only a subset of this data traffic which has a higher priority.

A simple solution that will be described hereafter is to allow all data traffic, using the same routes used for control traffic. In this solution, when O_2O enters in the emergency status it removes all the rules inserted by the remote controllers from the flow table and dumps all the OLSR routing table, i.e. including the routes outside the control-subnet and the default route advertised by the gateways. In doing so the routing of the mesh becomes substantially controlled by OLSR, while the forwarding is always carried out through OpenFlow mechanisms. When the controller becomes reachable, the O_2O leaves the emergency status and removes from the flow table the rules associated to routes outside the control-subnet, thus forcing ongoing data flows to send packet-in data units to the controller, which will decide how to re-route them.

3.3 Design of controller distribution architecture

According to our requirements, Express needs a distributed solution that allow different controllers to take control of our WMR nodes and of routers in the wired backbone.

We considered in our design the following two main aspects:

1) The different controllers need to synchronize about which controller is master for each switch, performing a “master election” procedure.
2) The different controllers need to share a common view of topology and of the network events that are relevant to take decisions in the controller layer.

In our scenarios, the “master election” procedure needs to be repeated each time that a portion of network become partitioned or when different partitions are joined together in a larger partition. Express adopts a hierarchical approach, as illustrated in Fig. 3. The basic idea is that the control will be taken by the controller connected to the WMR with the highest level in the hierarchy. The “master election” procedure will therefore select for each switch the highest level controller that can control the switch. With this approach, we focus on the problem of partitioning/merging of network portions, while we are deliberately not focusing on load sharing issues. In fact, if the Home Controller will be visible from each switches, it will be selected as the master controller for the whole network.

As for the common view of topology and events, we assume that OLSR topology distribution mechanism can be exploited by OpenFlow controllers. The Express controllers will learn the topology and will receive topology updates using OLSR. For the purpose of Express experiments, the overall map of potential controllers and WMNs can be statically configured in all controllers (e.g. using some configuration file). We will analyse if the OLSR protocol can be further extended to support functionality related to our specific scenario, but it is not of highest priority to design and implement such extensions.

4 Discussion and state of the art

In this section we provide some background discussion about the design choices we have made and some information about the state of the art of research and related works. We first discuss the aspects of failures of control plane communication links and then the aspects related to distribution of functionality among a set of controllers and to controller failures.

The failure of control-plane communications is typically faced by a wired SDN/OpenFlow deployment by means of an “out-of-band” switched network, virtually implemented through a pre-configured VLAN, realized by means of pre-loaded rules in the switch flow-table. In this case the OpenFlow switch routes the control traffic towards the controller, using plain Ethernet auto-learning mechanisms, including the Spanning Tree Protocol. In case of link failure, after a timeout, the MAC lookup table of the switch will be properly updated with an alternate route. In our scenario, this Ethernet auto-learning approach for setting up and maintaining the control-plane communications is not viable, because it requires point-to-point links, rather than single broadcast interfaces as in a Wireless Mesh Network environment. Furthermore, also the out-of-band strategy [3] may be difficult to handle in a mesh environment, where multi-hop VLAN support is not provided off-the-shelf. These consideration led us to define the in-band control approach described in section 3.2.

The design of a distributed controller architecture is an open research issue, existing work has been focused on solution for load sharing and resilience in wired environment (mostly data center environment or WAN interconnection of data centers). This has led to “flat” solutions where a layer of controllers is able to take control of a set of switches. In these solutions, the different controllers are used to distribute the control load, to increase the proximity between the controller and the controlled switch and for backup, so that a “secondary” controller can become masters of a set of switches if their default primary controller fails. There is no hierarchy among controllers in these approaches [4][5][6].

The failure of the controller-unit is typically “protected” by replicating the controller [6], in a static and pre-defined way, so having in the network an *origin controller* and a *surrogate controller*. In case of failure of the origin controller, the “hot standby” surrogate controller can take over. Likewise [5] focuses on the issue of migrating the control of a switch between one controller and another one in the controller layer, as a reaction to change of load conditions on the controllers.

In our Express scenarios, the system should react to partition and merge events so we cannot assume a flat layer of controllers that are typically always available. Therefore we have focused on a hierarchical approach.

5 Conclusion

In this document we have reported the Express requirements and the initial design of the Express architecture. We have described the proposed solution for integrating SDN concepts, in particular using the OpenFlow protocol, in a Wireless Mesh Network scenario.

Our next steps will be the implementation of the proposed solution and the deployment of the solution, first in the separated OpenLab wireless testbeds NITOS and W-iLab.t, then in an integrated experiment involving PlanetLab Europe testbed to interconnect the wireless testbeds.

6 References

[1] Volkan Yazıcı, “Discovery in Software-Defined Networks”, blog post, http://vlkan.com/blog/post/2013/08/06/sdn-discovery/

[2] Open vSwitch website: http://openvswitch.org/

[3] P. Dely, A. Kassler, N. Bayer, “OpenFlow for Wireless Mesh Networks”, IEEE International Workshop on Wireless Mesh and Ad Hoc Networks (WiMAN 2011), Hawaii, USA, August 2011

[4] T. Koponen et al., “Onix: A Distributed Control Platform for Large-scale Production Networks,” in OSDI, 2010.

[5] Advait Dixit, Fang Hao, Sarit Mukherjee, T.V. Lakshman, Ramana Kompella, “Towards an Elastic Distributed SDN Controller”, HotSDN 2013

[6] A. Tootoocian and Y. Ganjali, “HyperFlow: A distributed control plane for OpenFlow”, in INM/WREN workshop, 2010.