Software Defined VPNs

Stavros Konstantaras & George Thessalonikefs
stavros.konstantaras@os3.nl george.thessalonikefs@os3.nl

Supervised by
Drs. Rudolf Strijkers

University of Amsterdam
System & Network Engineering MSc

August 5, 2014

Contents

Table of Figures

1 Introduction

Virtual Private Networks (VPN) is a secure way for large companies and organizations to expand their network beyond their physical infrastructure by deploying secure tunnels across the Internet. Hence, VPNs are a successful solution to the problem of interconnecting LANs located in different countries/continents or providing the capability for secure network access for remote users.

New networking concepts like Software Defined Networking (SDN) promise to offer more flexibility and advanced administration capabilities into a field, which grows significantly every year. Major ISPs like Deutsche Telekom, Telefonica and SURFnet are examining possible use cases where SDN is able to help them offer new services to clients in minimum time.

A type of VPN technology is Virtual Private LAN Service (VPLS). It allows organizations to interconnect their local Ethernet networks in a scalable way where Internet Service Providers (ISPs) are responsible for bridging the virtual links. Despite the fact that vendors have developed optimized solutions for delivering VPLS services to customers, it is still uncertain if SDN concepts can be used to implement VPLS effectively as well.

The uncertainty is being originated by the modern conceptual idea of SDN to provide flow based connectivity between the hosts, while VPLS uses a decentralized packet labeling approach. Therefore, VPLS functions need to be redesigned with SDN based solutions and this is what this research project offers to the scientific community.

1.1 Scope of the work

Throughout this research project we will design a VPLS solution by using SDN concepts and focus on the mapping of VPLS functions to the OpenFlow 1.3 switch specification interface[1]. An overview of our design can be seen in Figure 1. Besides that, we will also evaluate if the 1.3 version is powerful enough for an efficient implementation of VPLS.

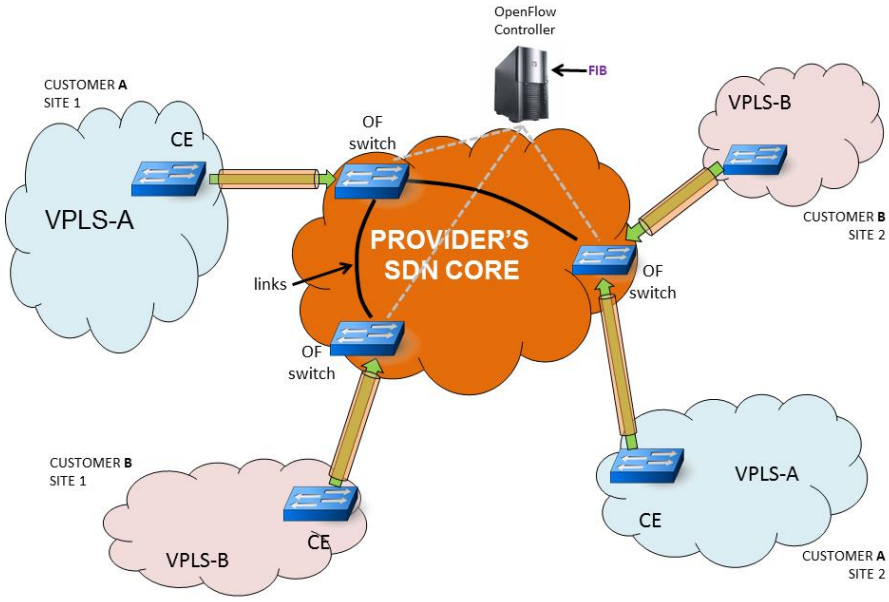

Figure 1: Design overview

Using the Community Connect project (CoCo) [2] as a use case for on-demand VPLS, we will also examine the possibilities of multi-domain, auto-configured VPLS instances. The CoCo project is a combined effort of TNO and SurfNet to provide on-demand private and secure virtual networks to research and scientific communities participating in NREN and the GEANT network.

In addition, our research will also examine possible practical problems, which can be raised from this approach, like traffic aggregation and user mobility. Thus, the scope of our project is to deliver a software defined network architecture based on state of the art technologies.

1.2 Problem Statement

The flexibility that SDN provides to network administrators for developing and deploying new on-demand services to customers and organizations is the key of our approach. However, requirements such as scalability, efficiency and effectiveness are guiding our research. Being scalable in respect to the increasing number of hosts is critical as our design aims to interconnect hosts of large organizations in different VPNs. Efficiency in terms of network resources (i.e., effective link usage, number of flows) is also important and requires measures such as traffic aggregation. We are going to study the possibility to adopt these requirements into our solution and examine what are the requirements and possible benefits of turning this approach into a real implementation.

Therefore, the main research question is the following:

How can VPLS be implemented efficiently by using the OpenFlow 1.3 switch specification interface?

The main research question can be divided into the following sub-questions:

- Can SDN be an underlay layer for building on-demand VPLS services?
- Is it possible to support a multi-domain environment?
- Is SDN flexible enough to support at least as scalable, efficient and effective implementation of VPLS as existing solutions?

1.3 Outline

In this report we will first investigate how the VPLS technology works and what mechanisms are implemented for providing Layer 2 connectivity between the hosts. After that, we examine the existing technologies that can help us build an efficient network architecture based on SDN concepts.

We continue by describing the conceptual functionalities of two network approaches, described in detail on chapter 3, that are able to provide solutions to several requirements. In addition, we proceed to a deep analysis for both approaches in operational, networking and scalability fields and we present the modifications required for the architecture to be adopted by the CoCo project. Moreover, we include some optimizations and ideas that can be easily embedded for traffic aggregation or covering functionality gaps.

Finally, we discuss our work along with the advantages and the disadvantages of our implementation and we conclude the report with our personal opinion and suggested future work.

2 Related work

The requirement of migration strategies from traditional network protocols to the networking model of SDN allows engineers to rethink their mechanisms. Despite VPLS exists and matured in the industry for more than ten years, its mechanisms cannot be adopted by SDN without considering the impact.

In [3], it is discussed that in order to provide Layer 2 VPNs the natural choice of operators is to use VPLS/MPLS. Based on an MPLS-free network like the one that belongs to SWITCH, the national Swiss research organization for developing communication technologies, the idea to use SDN in order to provide Layer 2 VPNs was nurtured. According to their argumentation, it will be easily embodied in their current network and current off-the-shelf components can be used to build powerful platforms delivering raw packet-forwarding speeds comparable to ASIC implementations in the domain of 10GE.

The use of OpenFlow as a means to provide Dynamic VPNs was the target of research done in [4]. Current implementations using VPLS/MPLS were compared with a possible implementation of OpenFlow 1.3. It was shown that the centralized nature of OpenFlow could replace the various MPLS-related technologies used with MPLS. These related technologies form a complex protocol stack needed in order to provide additional functionalities such as topology discovery (OSPF) and path provisioning (RSVP/LDP) on top of the pure forwarding functionality of MPLS. It was discussed that the adoption of OpenFlow could provide the network operator with a more manageable interface towards its network but several concerns were raised regarding the as of yet undefined/unstandardized Northbound and East/Westbound interfaces that limit portability and scalability.

In [5], the idea to provide a unified implementation for multi-domain SDN/OpenFlow is discussed and a solution based on an orchestrator is presented. The OpenNaaS management platform could be used as such an orchestrator that would coordinate previous management systems as services, while yet delegating the execution of operations locally to each domain.

2.1 Software Defined Networking

Software Defined Networking (SDN) is a modern promising networking concept born in Stanford University which is based on a simple idea: the clear separation between the control plane and the data plane of a Layer 2 switch. While the data plane remains in the switch, the control plane is transferred in a new centralized network element named “Controller”. All switches that participate in the SDN network establish a permanent TCP communication with the Controller in order to receive commands and install rules.

Until now, only the Switch-to-Controller communication has been standardized with a Southbound API named “OpenFlow”. This protocol provides huge flexibility to network engineers as different SDN switches manufactured from different vendors can communicate with the same controller. In addition, the OpenFlow Controller is usually software running inside a powerful commodity server but some vendors have released commercial versions based on more sophisticated implementations. Researchers and small/medium organizations are able to implement their own Controller based on their needs, or modify one of the well-known open source implementations.

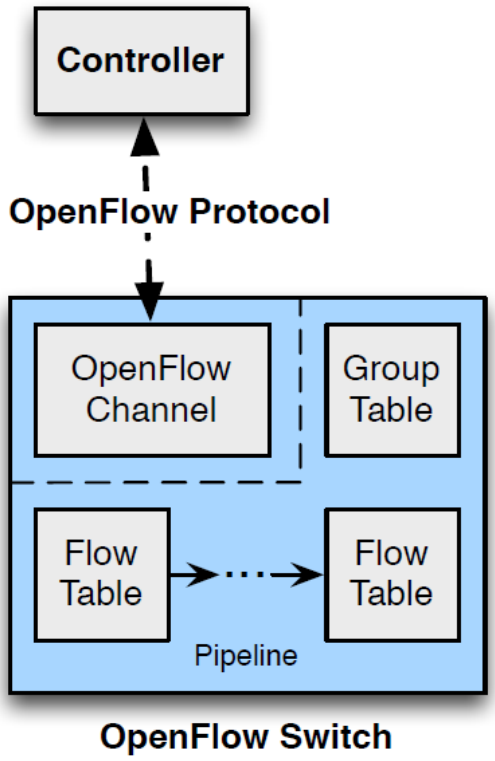

Figure 2: Main components of an OpenFlow switch (source: OpenFlow Switch specification 1.3.0)

Figure 2 demonstrates the basic components of OpenFlow version 1.3, which was released by Open Networking Foundation (ONF) at the middle of 2012. Flow tables allow the controller to install multiple rules inside the switch for matching different protocols of the incoming packets. Moreover, the Group table allows for grouping of different actions that need to be executed when a packet matches a rule.

Another advantage that OpenFlow offers since version 1.1 is the embedded support for VLAN and MPLS. Both protocols are widely used by network engineers in local and core networks for managing virtual network topologies. Thus, we consider OpenFlow to be a production ready technology and important tool for Network functions Virtualization (NFV).

It is important to mention that OpenFlow is an event-driven protocol where functions of the Controller are triggered when the switch sends various type of messages, based on network events. A type of event is the "Packet-In" event which is sent to the Controller when the incoming packet does not match any installed rule on the Switch. The OpenFlow Packet-In messages notify the Controller the port of the switch the packet arrived on and also encapsulate the original packet in the payload. The Controller can extract valuable information from the Packet-In messages (e.g., MAC addresses, IP addresses, Layer 4 protocol used for both source and destination) and proceed to routing decisions accordingly.

2.2 VPLS Overview and Requirements

In MPLS/VPLS [6], a Service Provider (SP) offers a Layer 2 VPN service between a Customer's different sites. Effectively, the Provider emulates a Layer 2 switch (Layer 2 broadcast domain) which interconnects the Customer's different LAN segments. The Provider's network is invisible to the Customer and the different LAN segments share the illusion of being directly connected. The common practice is for SPs to use VPLS on an MPLS core network.

As shown in the figure below (Figure 3) the main building blocks of VPLS are the following:

Customer Edge (CE) devices are VPLS agnostic and provide connectivity to the Provider's network for the LAN devices. By keeping the CE devices VPLS agnostic, minimal configuration is needed from the client-side.

Provider Edge (PE) routers are the key devices in a VPLS implementation providing all the necessary functionalities in order for the Customer's LAN segments to share a single Layer 2 broadcast domain.

Attachment Circuit (AC) refers to the form of access shared between PE and CE devices. It is irrelevant to the VPLS and could be any connection carrying Ethernet frames, from a physical or logical (tagged) Ethernet port to even an Ethernet pseudowire.

Pseudo Wires (PW) are signaled between PE routers that share the VPLS in order for them to be interconnected in the Provider's network.

Forward Information Base (FIB) is used to associate MAC addresses to (logical) ports on which they arrive. They need to be used per VPLS in order to guarantee traffic isolation.

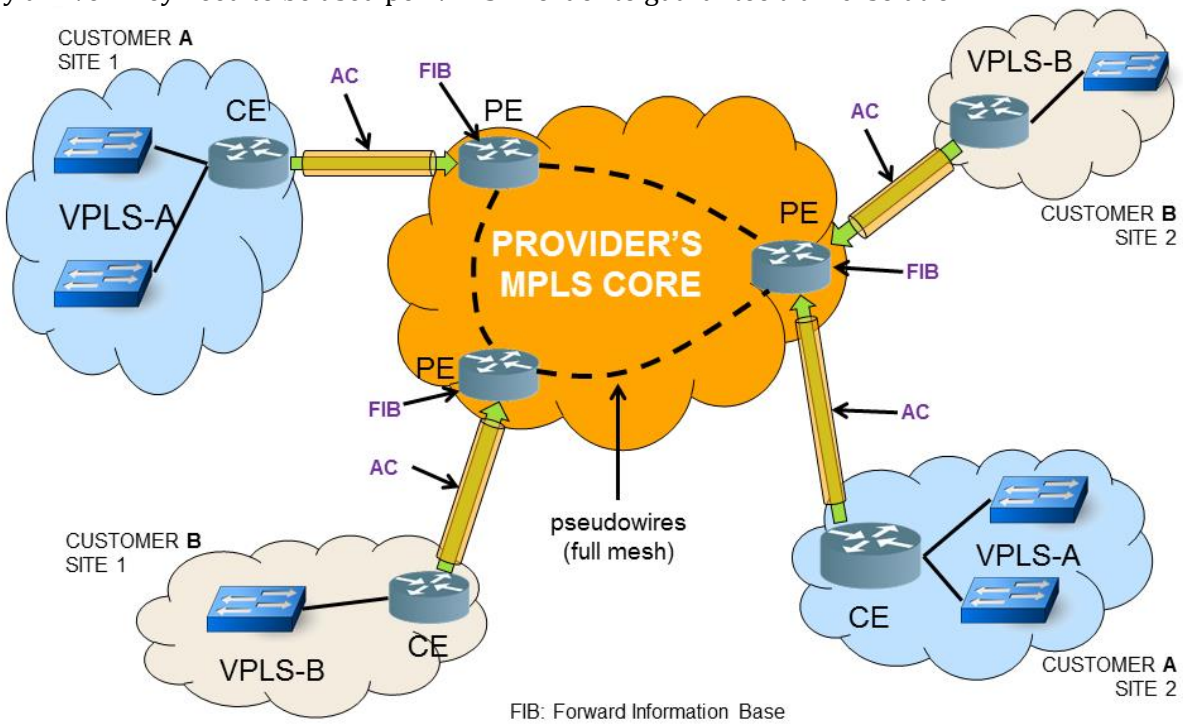

Figure 3: MPLS/VPLS conceptual architecture

Each PE can support multiple VPLS instances assigned to different kind of traffic, thus separating traffic between different customers or even traffic of the same customer. The separation of traffic is based on agreements between Provider and Customer on the use of the related AC (e.g., use of specific VLAN tags to mark different kind of traffic originating from the customer).

A requirement for current VPLS implementations is for the PEs to be connected in a (logical) full mesh topology. This strategy will prevent broadcast loops between the PEs which consume valuable network resources as a result of the broadcast packets transmitted by the hosts. By deploying virtual direct links, pseudowires, between them, each PE examines the incoming packets and decides to forward or drop them based on the attached hosts. The split-horizon technique which is also used, instructs a PE to send broadcast packets to all the ports except the one of the incoming packet. By adopting both techniques in a VPLS commercial implementation, it is possible to prevent loops, lower the usage of the network resources and keep the solution more efficient.

Mechanisms for PE auto-discovery and signaling of their PWs are also needed for automating the required configuration. Currently, two main solutions exist: using BGP as an auto-discovery and signaling mechanism and using manual configuration along with LDP for signaling. Auto-discovery simplifies the demanding configuration of the PEs needed, given the nature of their full mesh connectivity.

Each PE has to be configured independently and through BGP their configuration will spread to other PEs of the same VPLS allowing for automatic signaling using the underlying, already established MPLS network. When not using any auto-discovery mechanism, manual configuration of all the PEs is required. The configuration of individual PEs depends on all the other PEs participating in the same VPLS. LDP is then normally used in the MPLS network to signal the VPLS PWs between the PEs. Functionalities that need to be supported by the PEs in order to provide a Layer 2 broadcast domain are presented in Table 1.

VPLS Functionality name	VPLS Functionality explanation
MAC Learning	The PE learns the MAC address of the attached hosts and associates them to specific VPLS.
MAC Aging	The PE counts the time passed since a MAC address was learnt.
MAC Withdrawal	The PE commands the other PEs to delete a MAC address.
MAC Flooding	The PE floods a packet to all the other PEs in the same VPLS.
Handling broadcast traffic	The PE receives and handles a broadcast packet accordingly.
Handling multicast traffic	The PE receives and handles a multicast packet accordingly.

Table 1: VPLS functionalities

Since the Provider's network emulates a layer 2 switch, the MAC addresses functionalities are necessary. MAC learning refers to the ability of the PEs to associate MAC addresses to (logical) ports they arrive on. In VPLS it is achieved by using separate FIBs for traffic isolation. MAC aging is required in order for a PE to relearn a MAC address in case of relocation of a host and to also help reduce the size of the FIBs by only holding information about the active MAC addresses of the

VPLSes. MAC flooding is required in order to flood unicast traffic to all other PEs in the same VPLS in case a destination MAC address is not present in the PE's FIB.

Traffic with a destination of the well-known Ethernet broadcast address or the set of easily recognized multicast MAC addresses should also be flooded to all other PEs participating in the same VPLS. As Layer 2 multicast traffic is essentially broadcast traffic that a limited number of hosts are configured to accept it, certain mechanisms may be used in order to detect these multicast addresses and forward the traffic only to interested hosts instead of all the hosts in the VPLS.

Vendors have developed ASIC equipment to offer VPLS solutions. Based on the fact that VPLS offers a layer 2 solution, the specialized equipment is not designed to handle traffic based on upper layer protocols. Problems, specifically with broadcast traffic, can emerge and Providers need to rely on external solutions in order to compensate for unwanted traffic (e.g., AMS-IX's use of "ARP sponge" to mitigate excessive ARP traffic)[7].

2.3 Related technologies

Modern requirement for supporting multi-tenancy is achieved with virtualization, where each physical server hosts multiple Virtual Machines (VMs), each one managed by different customer for different purposes. However, a complexity is raised when a customer is responsible for managing many VMs. If all of them are located in the same commodity server, the problem can be addressed easily and Inter-VM communication is not forwarded in the local network. However, if the customer's VMs are located in different servers and need to communicate between as being in the same broadcast domain, a problem occurs of how to address traffic efficiently.

Virtual eXtensible LAN (VXLAN) enables virtual machines to share a LAN even if they are separated by different networks. The current draft of IETF [9] describes the concept and its fundamental functionalities but a richer version is expected in the near future. It succeeds in creating a common broadcast domain for numerous VMs by using Virtual Endpoints (VTEPs), which are embedded inside a Hypervisor.

Each physical server is the start or the end of a VXLAN tunnel where the Hypervisor is responsible to deliver the packets to the correct VM. Host to host communication is established by using common IP routing protocols, an efficient solution for keeping core network simple. The packets originated from VMs are encapsulated from Hypervisors inside UDP packets and being sent to the corresponding VTEP.

The service provided by VXLAN is that different VMs located in different servers can now belong in the same overlay network and exchange data as being in the same broadcast domain. In order to achieve that, the VMs are grouped under a unique identifier called Virtual Network Identifier (VNI) where the VTEPs have multiple roles on them:

- Automatic update of which servers have which VNIs in order to expand/decrease the overlay network,
- Keeping track of which VMs belong to which VNIs,
- Traffic isolation between the different overlay networks by using the VNI as authenticator.

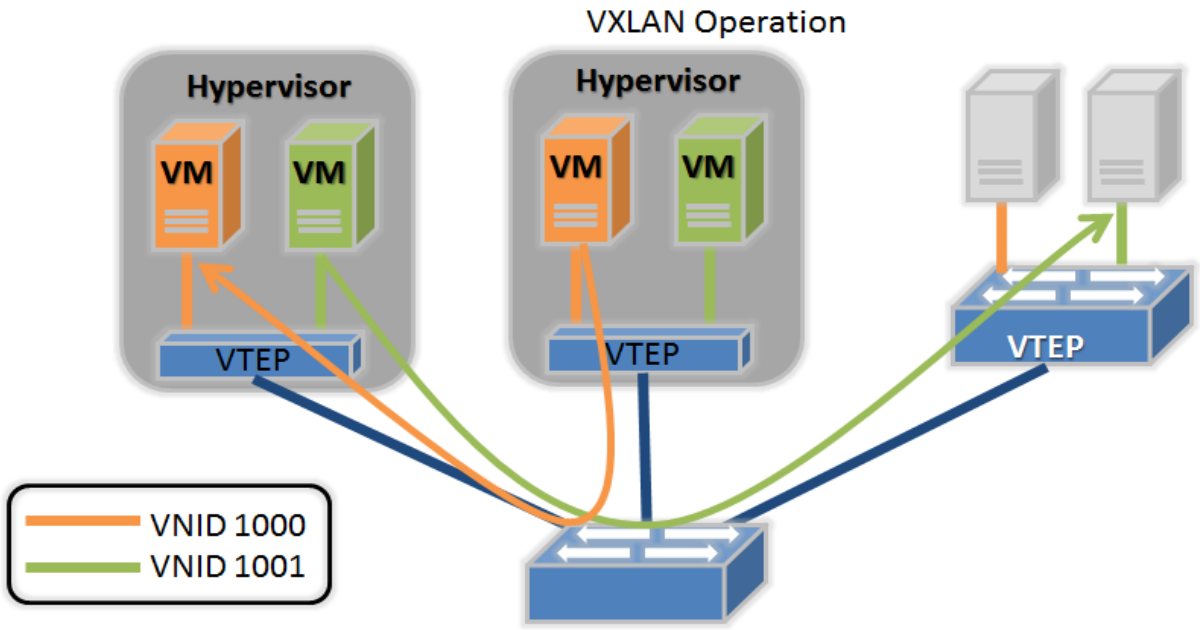

Figure 4: VXLAN conceptual architecture (source: www.definethecloud.com)

Figure 4 demonstrates a simple VXLAN architecture. Traffic isolation is achieved by using VTEPs and grouping VMs under the same VNI. Each Hypervisor has the knowledge of which VMs belong to which VNIs and exchanges this knowledge with other Hypervisors. At the end, all the participating VTEPs have a MAC table which allows them to distinguish and forward traffic accordingly. When a VM sends a packet to another VM in the same broadcast domain, the VTEP receives the packet, examines the destination MAC address and according to the internal MAC tables forwards the packet to the corresponding VTEP. In case of broadcast traffic, the VTEP sends the broadcast packet to the VTEPs which have VMs participating in the VNI of the sender.

However, VXLAN is not suitable for creating on-demand VPLSs because of the following reasons:

- VTEPs already handle the traffic generated by the VMs in an intelligent way,
- The virtual tunnels are encapsulated inside a UDP packet but OpenFlow switches are not able to match inner packets,
- The type of network traffic is hidden from the Controllers and therefore intelligent handling is difficult to be applied (i.e. managing the ARP requests).

3 Designing an SDN based VPLS

One major concern when designing a Layer 2 multipoint-to-multipoint VPN architecture is the distribution of knowledge. The information of which hosts belong to which VPN needs to be known to the devices responsible for traffic forwarding. As already mentioned in section 2.2, VPLS accomplishes that by using FIBs residing in the PEs.

3.1 Design requirements

In order to provide a fully functional and modern architecture which can be adopted easily in the near future, we need to fulfill the following requirements for having a scalable VPLS service:

- Layer 2 solution
  Each team of hosts that shares a virtual private LAN should have the freedom to choose its own Layer 3 address scheme. Therefore, we need to address all the principal problems at Layer 2 and this increases the complexity of our research.

- Traffic isolation between VPNs
  This is a default requirement when offering private LAN services. Traffic between different VPNs traverses the same network but hosts of one private LAN service must not be able to communicate with hosts of another private LAN.

- Scalable multi-domain support
  Private LAN services must be able to span over different administrative domains. Based on the number of hosts and the amount of traffic needed to interconnect the different hosts, a scalable approach needs to be considered. Also, coordination is required between the different domains to provide unified virtual Layer 2 networks. As a result, hosts in different administrative domains are able to participate in the same private LAN.

- Host's on-demand multi-VPLS participation
  Hosts must be able to participate in more than one VPN on-demand. By providing a pure Layer 2 solution the combination of a host's MAC address along with the VPLS information must be globally unique in order to effectively identify the different hosts.

- Traffic aggregation
  We need to provide connectivity between islands which are located in different domains and create multi-domain paths that are able to aggregate traffic and route both unicast and broadcast packets efficiently in the core network. We also need to be conservative at the number of flows required for forwarding traffic. By using OpenFlow we can create flows that match only the necessary fields in order to provide aggregation.

- VPLS configuration mechanism
  A configuration mechanism for setting up the virtual LANs is required. Information regarding the domains and islands that participate in the network is required to create communication channels between them. Through these channels we are able to exchange VPLS information between interested parties. We also need to know which hosts are meant to take part in which private LANs in order to establish access control. This way the privacy part of the VPLS is satisfied.

- MAC learning/aging/flooding/withdrawal support
  By providing a pure Layer 2 solution the standard MAC address mechanisms are needed for the following reasons. MAC learning is used to dynamically learn a host's location as it associates a MAC address with a port. MAC aging is used to minimize the size of the MAC tables by invalidating hosts that have not generated any traffic over a specific period of time. It is also used to address host mobility by determining the most current location of a given host. MAC flooding is used when a host's location is unknown and traffic is then flooded to the network until the given host is learned through the MAC learning mechanism. MAC withdrawal is used to invalidate a given MAC address. Invalidation of a MAC address is needed for mobility and link-failure reasons, for example when a host maintains two different links to the network for redundancy purposes. When the primary link fails, the MAC withdrawal mechanism indicates that the host's primary location is invalid.

3.2 Design problems

The key design problems are presented in the following paragraphs:

Number of VPLSes
When designing a multi-domain VPLS architecture we need to consider the scalability issues. Having a multitude of hosts that can participate in different VPLSes simultaneously raises concerns about the total number of VPLSes supported by the architecture.

Associating hosts with VPLSes
The problem of associating hosts to VPLSes is directly related to traffic separation. The usual approach for separating traffic in the network is for switches to group ports to different virtual LANs. This is achieved for example by using VLAN and tagging packets based on the incoming ports. However, one of the design requirements is the host's ability to participate in multiple private LANs simultaneously. As a result, network devices are incapable of deciding how to label traffic coming from the same host and traffic labeling needs to be transferred from the network devices to the hosts. In that way, the hosts themselves dictate in which VPN they take part in

VPLS privacy
Without proper network configuration a host, for example, can arbitrary label its traffic and participate in a private LAN while he is not authorized to do so. Furthermore, broadcast traffic needs to reach every machine that is part of the same private LAN. A typical Layer 2 action is to flood the packet to all ports in order for the traffic to reach every known and yet-unknown machine in the network. Due to the privacy requirements, meaning traffic isolation, accompanying a VPN approach, flooding a packet to all possible ports is forbidden. We also cannot rely on MAC learning alone to identify VPN ports because a host may not have generated any network traffic yet and thus remains unknown. These rise security concerns and extra measures are needed to ensure the privacy of the virtual LANs.

Joining/leaving VPLSes
The privacy concerns discussed in the previous paragraph can be addressed by knowing beforehand which hosts belong to which VPNs. For this to work along with the requirement for on-demand multi-VPLS participation, a kind of mechanism/interface that will allow for host registration/deregistration is needed.

Unicast traffic
The usual approach to create flows to forward Layer 2 unicast traffic is to match the source and destination of the packet. In a multi-domain environment where all the hosts can communicate with each other, installing flows in this manner can quickly increase the number of flows needed. Given the number of hosts in the network and the capabilities of the networking equipment in use, scalability concerns are raised.

Broadcast traffic
In addition to unicast, broadcast and multicast packets also need special treatment in order to avoid difficult situations. Host machines produce broadcast packets for several reasons (i.e. ARP requests) and since the virtual LANs are expanded in different islands, these packets also cross the core network. If broadcast traffic is not handled properly it can lead to broadcast loops and aimless consumption of network resources.

MAC learning
A major ingredient when designing a Layer 2 solution is the MAC learning mechanism as we need to know where the different hosts are located. In classical networking, MAC learning is practiced by the switches themselves. By using SDN a different approach to MAC learning needs to be taken, as the gathering of information is now a responsibility of the OpenFlow controllers.

Multi-domain flow efficiency
Having a multi-domain environment containing a multitude of islands and hosts, certain network devices could be overwhelmed by the number of flows required. Such devices are the switches that act as transit nodes while traffic passes through. For example switches that interconnect domains can be easily overwhelmed if flows are installed for every host.

3.3 The SDN/VPLS Architecture

The following network entities as presented in Figure 5 exist in our architecture:

- **Island Controller**, which is located inside a customer's site and manages an OpenFlow switch. The island controller has the responsibility to accept and forward packets to the provider's domain,
- **Domain Controller**, which is responsible to manage several OpenFlow switches inside the core network and forward traffic between islands existing in the same or another provider's domain.
- A **DE** device is defined as Domain Edge device and is an OpenFlow switch that connects Island(s) to the Domain,
- A **D** device is defined as Domain device and is an OpenFlow switch in the Domain network,
- A **DBE** device is defined as Domain Border Edge device and is an OpenFlow switch that interconnects different Domains,
- An **IE** device is defined as Island Edge device and is an OpenFlow switch that connects an Island to a Domain.

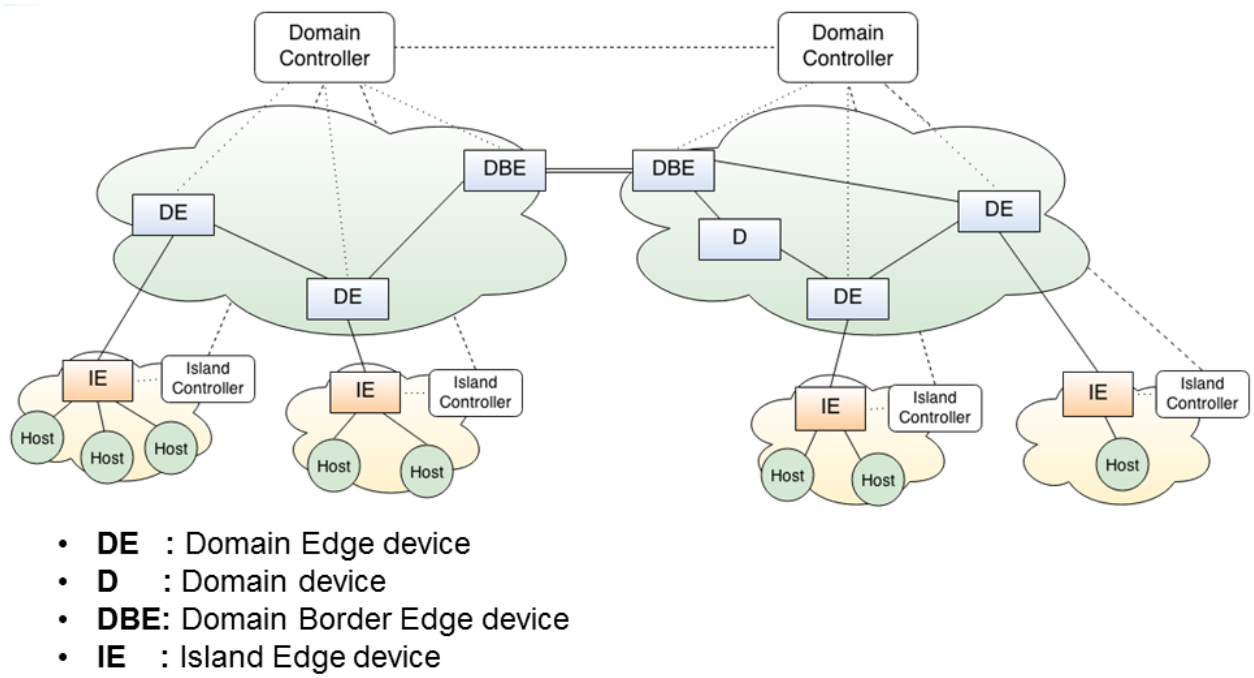

Figure 5: The SDN/VPLS Architecture

The OpenFlow controllers are assumed to know the topology of their local network, meaning how their switches are connected with each other and behind which ports there is communication with local Islands or other Domains. This could be achieved by manual configuration or automatic learning (see section 4.1).

Also OpenFlow controllers communicate with each other in a hierarchical way as shown in Figure 5 via Controller to Controller communication (CTRL-to-CTRL). Communication is defined according to the situation as:

- Island Controller to Domain Controller communication,
- Domain Controller to Local Island(s) Controller(s) communication,
- Domain Controller to Domain Controller communication.

The following are definitions for different (logical) ports:

- PORT - A port on a switch or a (switch, port) combination if viewed from a controller's perspective ,
- Host port - A port on an Island switch that is assigned to a host machine,
- Domain port - A port on an IE that is assigned to a DE,
- Inter-domain port - A port on a DBE that is assigned to a DBE on another domain,
- Island port - A port on a DE that is assigned to an IE of a local Island.

There are three different connectivity scenarios from the perspective of the host. They are presented in Table 2 in accordance with the amount of configuration required by the administration of the host's network.

Nature of host's network	Configuration required
OpenFlow Network	None
Campus Network	Little (VLAN configuration)
Legacy Network	Considerable (VPN configuration)

Table 2: Connectivity scenarios

The first and most straightforward is for the host to be part of an OpenFlow Island. In this case the host is connected to the Provider's network through an OpenFlow switch, which we can control through an OpenFlow controller. The second scenario is for the host to be part of a campus network with existing infrastructure which we cannot control. In this case, minimal configuration is required by the campus' administrator. A VLAN ID needs to be configured in the campus' infrastructure in order for the required traffic to be tunneled from an OpenFlow switch inside the campus' network towards the Provider's network. The third and last scenario is for the host to be part of a legacy network where no configuration is possible. In this case a VPN connection is needed in order to connect the host to a location that is already configured to properly forward traffic to the Provider's network.

3.3.1 Number of VPLSes and associating hosts with VPLS

As stated in the design concerns, traffic labeling at the host is needed. Our choice for host traffic labeling is VLAN as it provides a Layer 2 solution that is also supported by operating systems. But VLAN comes with a limit on the available VLAN IDs that is a hindrance to the architecture's scalability. In order not to confine the global number of total VPNs by the small number of possible VLAN IDs, we will use two different definitions to distinguish between local and global representation of VPNs as shown in Table 3 and explained further in the next paragraphs.

Hosts use VLAN in order to label their traffic and each Island chooses independently which VLAN IDs represent which VPNs. VLAN IDs are therefore used to locally represent a VPN from an Island's point of view and they are unique in the context of the Island only. In the core network, since traffic is already labeled by the hosts themselves, we use another labeling mechanism for forwarding VPLS traffic that is supported by OpenFlow 1.3, MPLS. MPLS offers a Layer 2 labeling solution that is also more scalable than VLAN, 2^{20} possible MPLS labels instead of 2^{12} VLAN tags. We refer to the MPLS label that is used to globally represent a VPN as VPLS_ID. Every VPLS_ID is globally unique and represents a distinct VPN.

In this way the global number of VPLSes is bound only by the maximum number of possible MPLS labels, 2^{20}. However, by using VLAN inside the Islands, the local number of VPLSes that can be present in a Island is limited by the maximum number of possible VLAN tags, 2^{12}. To clarify, there are 2^{20} global VPLSes and each Island can only participate in 2^{12} of them. The mapping between local and global representation of the VPLSes is stored on the appropriate controllers, in the form of tables (see APPENDIX A), in order to correctly manipulate and forward traffic.

Local and global representation of VPNs			
VPN	**Island A (local)**	**Core network (global)**	**Island B (local)**
VPN_A	VLAN_ID: 1	VPLS_ID: 10	VLAN_ID: 2
VPN_B	VLAN_ID: 2	VPLS_ID: 11	VLAN_ID: 3

Table 3: Example of local and global representation of VPNs

As a result, a HOST is defined as a unique combination of MAC address and VLAN/VPLS ID. To be more specific, from an Island's point of view a HOST is defined as a unique combination of (MAC address, VLAN ID). However, from a Domain's point of view a HOST is defined as a unique combination of (MAC address, VPLS ID).

3.3.2 Joining/leaving VPLS and VPLS privacy

As already mentioned in section 3.2, VPLS privacy can be ensured if we know which hosts participate in which VPNs. A control mechanism can be defined where a user can register/deregister a host to a certain VPN. The registration could be by means of associating and recording the host's location (switch, port, Island) with a specific VLAN/VPLS ID. Through this control interface, the information of which ports belong to which VPNs can be known to the controllers. On an Island controller this translates to a (VLAN ID, port) combination, whereas on a Domain controller to a (VPLS ID, PORT) combination. This way broadcast traffic can be properly flooded only to the same VPN ports.

3.3.3 Multi-domain flow efficiency

To provide efficiency and scalability in the core network regarding the amount of flows needed in the switches we introduce a location identifier, the ISLAND_ID. ISLAND_ID is a global and unique identifier which represents the Islands that participate in the complete network. It is used as an MPLS label in the packets and logically points to the Island of the destination host. In this way flow aggregation can be achieved. For example, all unicast traffic destined for several hosts on one Island now needs only one flow to be properly forwarded. The exact usage of ISLAND_ID differs according to the following designs. We present two network designs that handle the distribution of knowledge in a distinct way. Namely:

- Core Labeling, and
- Island Labeling

"Core Labeling" uses the edges of the core network to label the various VPN traffic. In this approach we try to confine the amount of information needed in each part of the network. Thus, an Island needs only to know about its local configuration and a Domain needs only to know about its own and its Islands' configuration. However, this will not always be possible.

"Island Labeling" uses the edges of the Islands to label the various VPN traffic. In this approach we try to keep the core of the network, the Domains, as information agnostic as possible by only keeping the minimum information required for forwarding the packets. All the information about the various VPNs is gathered in every Island instead.

3.3.4 Design 1: Core Labeling

A major concern when using OpenFlow is the efficiency regarding the choice of flows that are going to be installed in an OpenFlow switch as discussed in previous sections. In order to aggregate traffic to few flows one has to match fields on the packet that identify the recipient. This approach leads to a behavior where all the traffic from multiple machines destined to one specific machine can be handled by just one flow.

Following this idea we match packets based on the destination's identification, meaning the MAC address and the VLAN/VPLS ID used. That way traffic to that specific host can be aggregated to only one flow. In order for out-of-Island traffic destined to the same host to also match the same flow, packets arriving to the Island need to be ready for processing by the switch. Any actions needed in order for the packets to be correctly forwarded through the Provider's network are taken outside of the Island, therefore at the DE.

The same holds true for broadcast traffic. An Island controller can install flows to match a BROADCAST_MAC and VLAN_ID combination and forward it to the appropriate ports. Split-Horizon is also used to avoid sending broadcast traffic back to the originator. This is easily accomplished by matching the IN_PORT in broadcast packets and preventing traffic to be forwarded back.

Unicast traffic

The IEs send packets to the DEs as-is and expect packets to arrive with the right VLAN IDs that are used inside the Island. The Domain controller knows the mapping between VLAN IDs and VPLS IDs used by each local Island. Furthermore, it maintains a MAC table of all the hosts participating in the global network in order to take forwarding decisions.

Regarding unicast traffic on the core network, a DE should be able to forward VLAN tagged Ethernet packets. Since VLAN IDs are unique only on an Island's scope, extra labeling of the packets takes place when they traverse the core network. For that we use the globally unique VPLS_ID as an MPLS label. It will provide all the necessary information in order to map packets to specific VPNs.

The Domain controller can forward unicast traffic based only on the (VPLS_ID, MAC destination) combination but that could rapidly increase the number of flows needed inside the core network. In order to aggregate traffic to fewer flows, a location indicator is needed. We use a second MPLS label, which contains the ISLAND_ID. The Domain controller knows the location information for all Islands (via topology learning) and can easily forward traffic to them.

Based on these ideas unicast forwarding on the core network is easily achieved as follows:

1. The two labels are pushed to the packet,
2. Traffic traverses the core network based on the ISLAND_ID,
3. When it reaches the destination (DE responsible for the given Island), the VPLS_ID is examined in order to map the packet to a specific VPN, the labels are popped and the VLAN_ID is changed according to the local (VLAN_ID, VPLS_ID) mapping.

The whole unicast procedure from source to destination is depicted in the following figure, Figure 6:

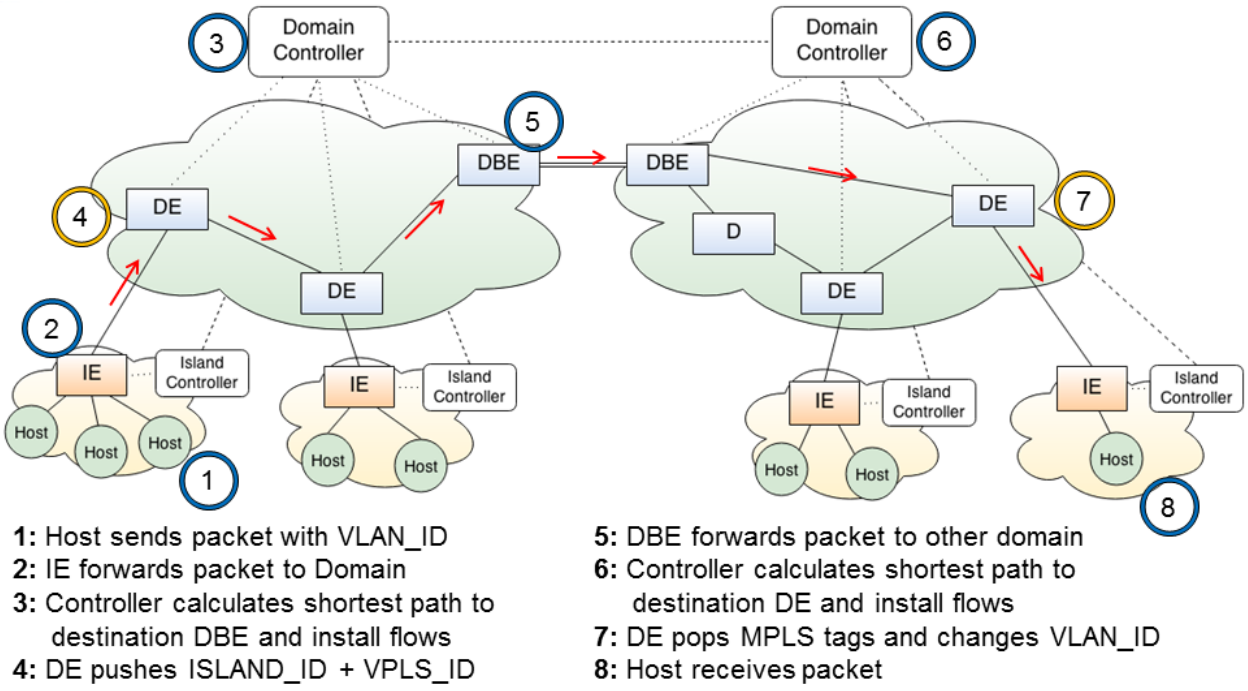

Figure 6: Core Labeling Unicast functionality

Broadcast traffic

For broadcast traffic the same double labeling approach is followed. The VPLS_ID is used as the first pushed MPLS label but for the second pushed MPLS label the BROADCAST_MPLS is used. BROADCAST_MPLS is a reserved MPLS label value where all the bits are set to '1'. It is used as a location indicator and logically points to all the Islands that participate in the same VPN.

The Domain controller knows its network topology and which Islands are part of which VPNs. Given this information, broadcast traffic is only forwarded to Islands participating in the same VPN. This is easily accomplished by using multicast trees created by the Domain controller.

Based on these ideas the broadcast forwarding in the core network can be easily achieved as shown in Figure 7:

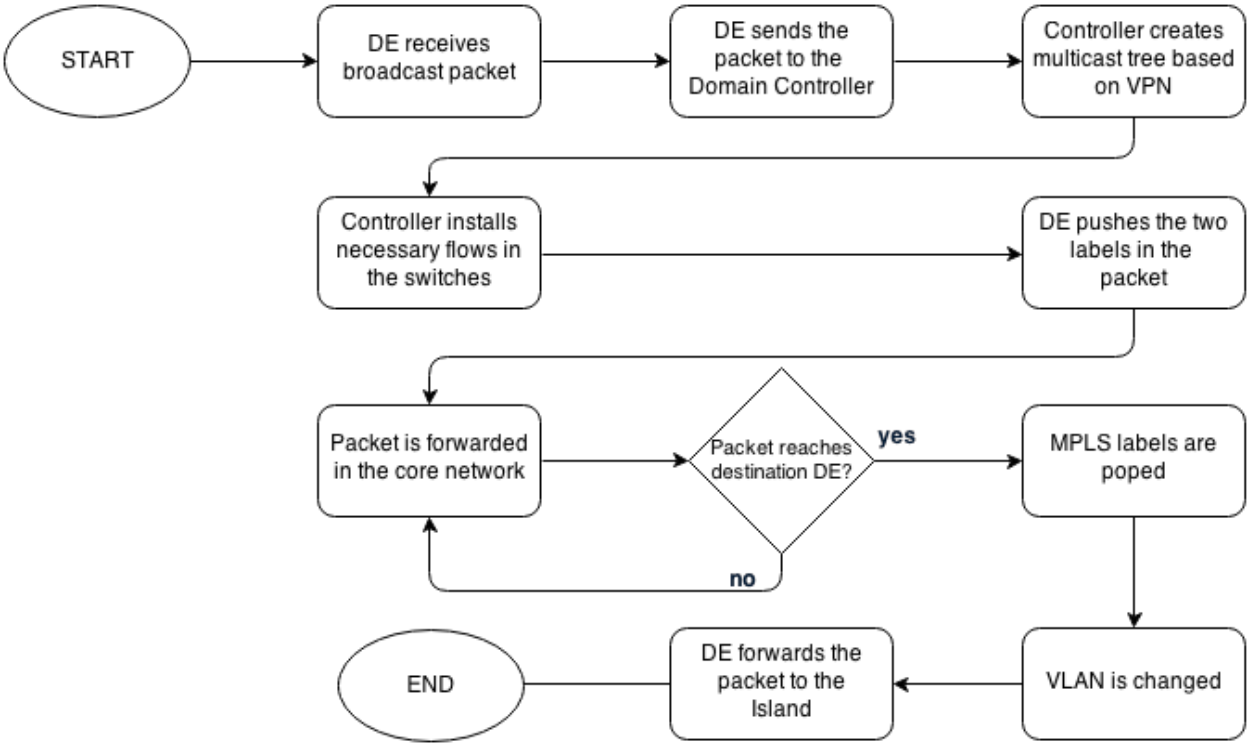

Figure 7: Flowchart of core broadcast traffic in Core Labeling

The whole broadcast procedure from source to destinations is depicted in Figure 8:

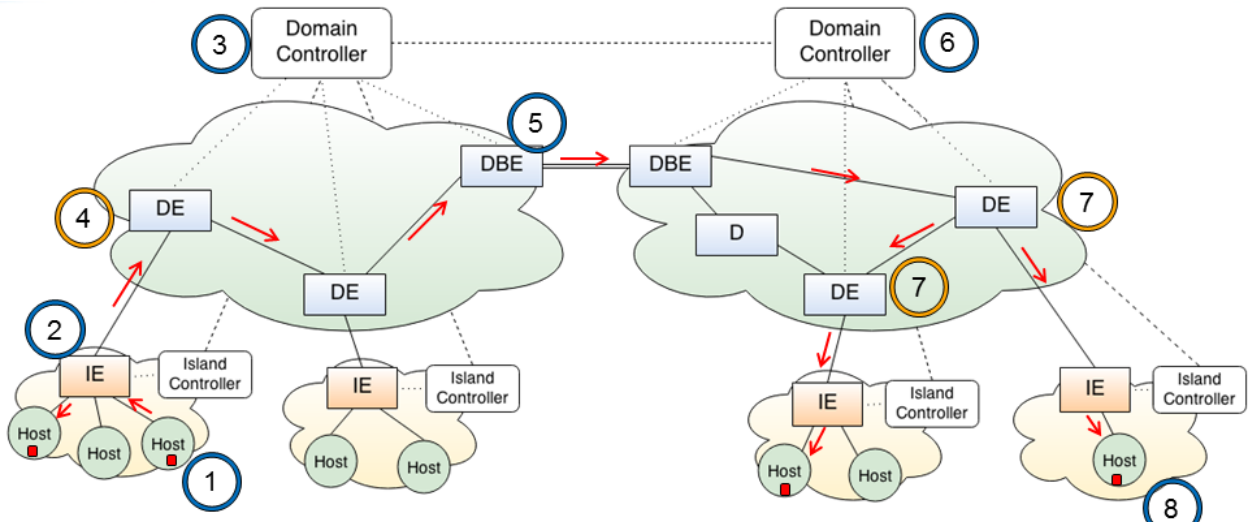

Figure 8: Core Labeling Broadcast functionality

Forwarding multi-domain VPLS traffic

The same practices as the ones already discussed about required information and forwarding of unicast and broadcast traffic are still valid. The main points regarding the multi-domain approach are:

- MAC tables are populated by MAC addresses of all the hosts participating in the global network,
- ISLAND IDs remain globally unique but they must be known to all Domain controllers,
- Each Domain controller can only forward unicast traffic up to the DBE, the other Domain controller then picks up the traffic and takes its own forwarding decisions,
- Likewise, each Domain controller creates a local multicast tree. The receiving Domain controller is responsible to create its own local multicast tree based on the VPLS_ID and use split-horizon to avoid sending traffic back to the originating Domain. However, depending on the Domains' interconnectivity, further considerations should be taken into account to avoid loops in broadcast traffic (see section 4.4),
- The inter-domain connection between the DBEs could be a physical or virtual link. No changes are necessary in both situations.

This concludes the Core Labeling design. Based on our requirements and concerns we were able to define an SDN design that can support multipoint-to-multipoint, multi-domain VPLS traffic. The Island Labeling design that follows is based on the principle ideas of Core Labeling but emphasizes the key differences required for a new distinct approach.

3.3.5 Design 2: Island Labeling

As we mentioned before, the major difference between Core Labeling and Island Labeling is the way that VPN knowledge is distributed across the global network. Furthermore, in the Core Labeling approach each Domain controller is responsible to provide the Islands with correctly tagged Ethernet packets. However, in the Island Labeling approach the Islands are themselves capable of handling both tagged and labeled Ethernet packets based on the global knowledge they possess, therefore supplying the core network with ready to be forwarded traffic. As a consequence, the Domain controllers do not need to have any host or VLAN information.

Same as in Core Labeling, flows for every host participating in the global network are still needed in every IE. In addition, the following information is present in every Island controller in order to apply the appropriate actions:

- MAC addresses of all the hosts participating in the global network,
- The VPLS instances (VPLS_IDs) that are currently active,
- The Islands that participate in each VPLS instance,
- The mapping between MAC addresses and VLAN_IDs for every host,
- The mapping between (VLAN_IDs, VPLS_IDs) for every participating Island.

Unicast traffic

The next figure, Figure 9, depicts the Unicast functionality applied in Island Labeling:

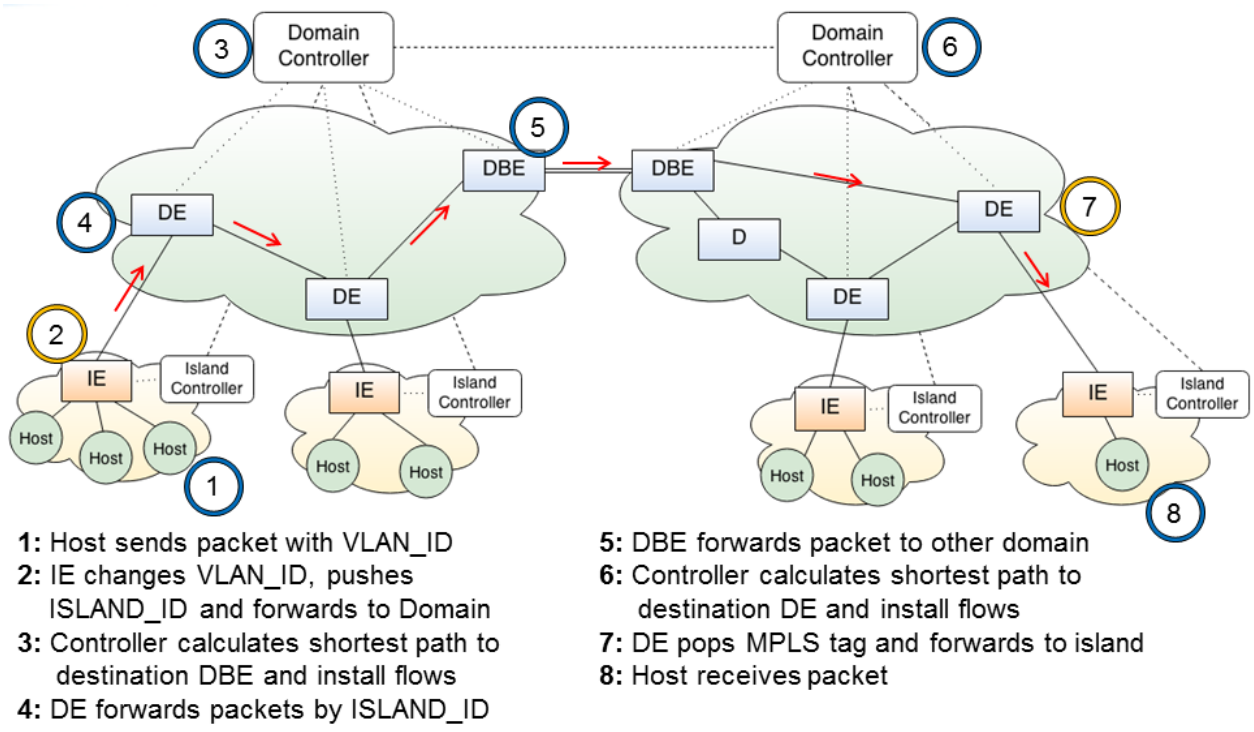

Figure 9: Island labeling Unicast functionality

When a unicast packet sent by a host inside an Island arrives at the IE, it matches one of the flows based on the destination MAC address and the VLAN_ID. If the destination host is a machine inside the Island, the only action that is applied to the packet is to be sent to the appropriate port. However, if the destination host is a machine in another Island the packet is forwarded to the provider's domain, which is completely unaware of MAC addresses and VLAN associations. Thus, the Island's OpenFlow switch has to proceed to the following actions in order to prepare the leaving packet to match the flows of the destination switch:

1. Change the packet's VLAN_ID to the corresponding VLAN_ID of the destination island,
2. Push an MPLS tag containing the destination ISLAND_ID,
3. Send the packet to the Domain Port.

In order for an island controller to be able to apply this strategy, it needs to have the required knowledge (which MAC addresses participating in which VPLS_ID with which VLAN_ID) before the hosts initiate their communication. This operational requirement is fulfilled by CTRL-to-CTRL where local configuration is being exchanged between the participants.

The second important difference between the two approaches is the number of MPLS labels inserted at a unicast packet. In Island Labeling only one label is inserted, indicating the destination island. This fundamental design detail makes the second approach being also efficient by requiring less packet overhead.

At the domain, the corresponding OpenFlow controller is responsible to route the incoming unicast packet according to the destination ISLAND_ID. Therefore, it needs to have the following information:

- All the ISLAND_IDs that participate in the complete network topology,
- The location (switch and port) of each island.

We assume that this information has been given to the controller manually (pre-configuration file) or by an automatic remote mechanism. Following the example of Core Labeling, since the Domain controller knows the topology of the network and also the ingress and egress ports, it is feasible to calculate the shortest path by using a well-known algorithm (i.e., Dijkstra). It can then install the appropriate flows to the necessary D devices that the traffic needs to cross for reaching the destination. When the incoming packet reaches the DE it matches a flow based on the MPLS label and is then forwarded to the appropriate port.

At the other side of the path, the DE that is attached to the destination island is responsible for one extra action except for forwarding the packet to the destination IE, it has to remove the MPLS label. This extra action will allow the packet to match the flow with (destination MAC address, VLAN_ID) that may already exist in the destination IE. Thus, we can take advantage of the existing flows and actions that already exist and keep their number at a minimum.

Broadcast traffic

In case of broadcast traffic, we need to follow a different strategy than the one we used in Core Labeling. The island controller installs a flow in the IE for every active VPLS_ID in the customer's site, each one matches the Broadcast MAC address, the input port and the corresponding VLAN_ID. But each of them is responsible to forward the broadcast packets only to the hosts participating in the same VPN.

When the IE receives a broadcast packet from a host, it matches one of the Broadcast flows and automatically is duplicated to all the ports where hosts participating in the same VPN are located. If the VPN is also expanded to other islands, the IE applies the following actions before the packet reaches the DE:

1. Push an MPLS label with the corresponding VPLS_ID,
2. Forward the packet to the Domain port.

Inside the core network, Broadcast packets are forwarded by using the VPLS_ID which is a piece of information known to the Domain controller. The broadcast functionality is depicted in Figure 10.

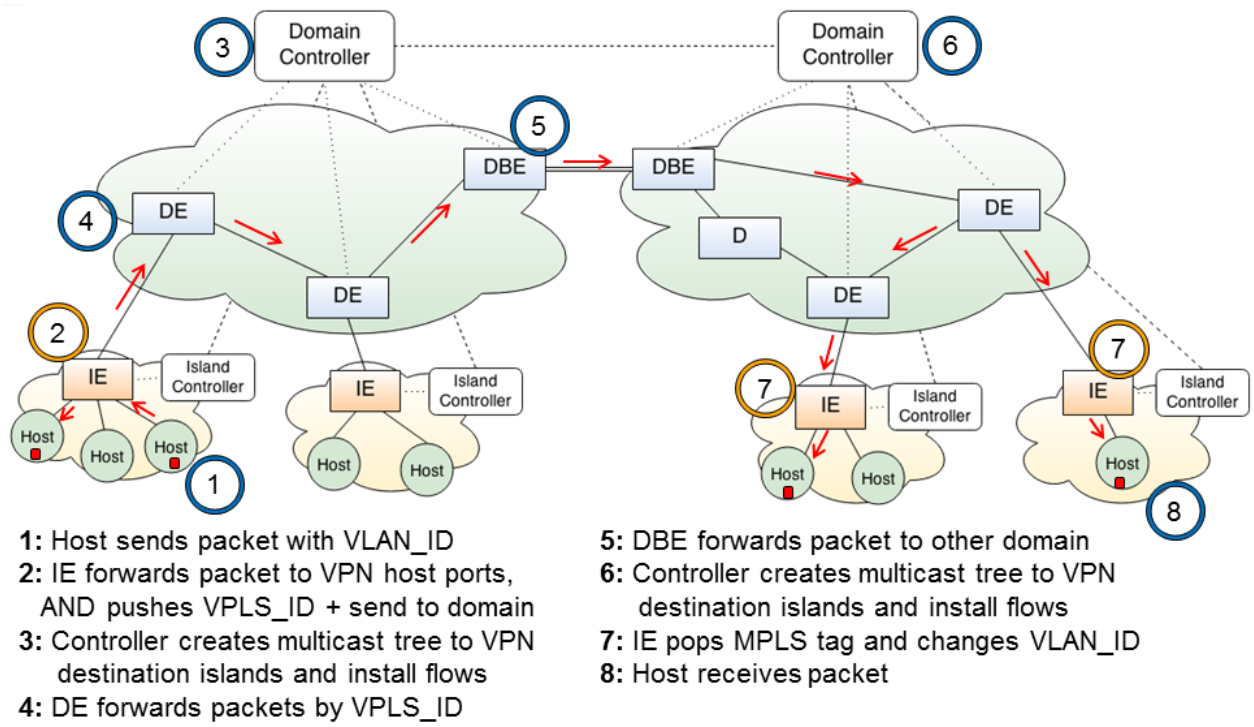

Figure 10: Island Labeling Broadcast functionality

To understand if the label of the incoming packet is a VPLS_ID or ISLAND_ID, the controller needs to examine the destination MAC address. The procedure is shown in Figure 11.

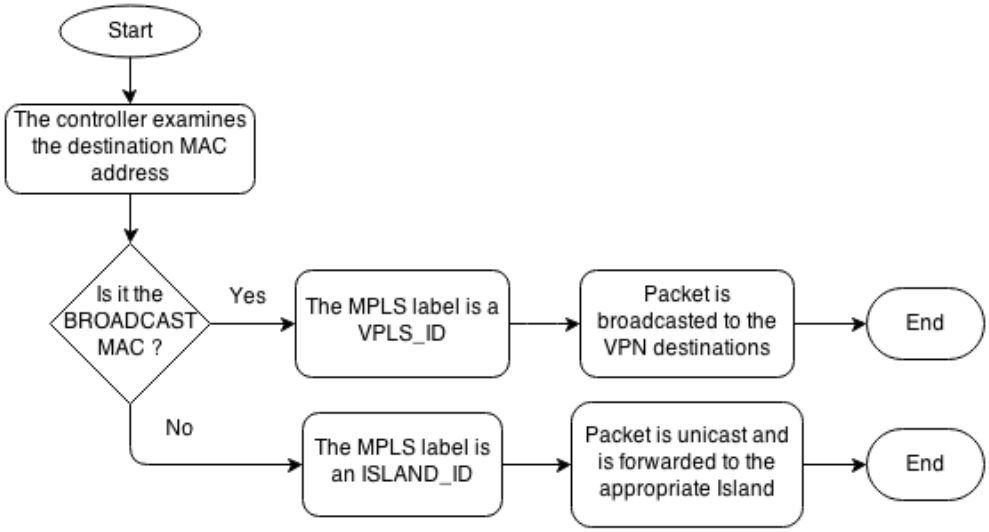

Figure 11: Flowchart of MPLS label distinction in Island Labeling

Based on these ideas the broadcast forwarding can be easily achieved as follows:

1. The Domain controller receives the broadcast packet and creates a multicast tree based on the VPN. Flows for the broadcast traffic are installed,
2. Traffic traverses the core network based on the (BROADCAST_MAC, VPLS_ID) combination,

3. When the packet reaches the destination (DE responsible for a given Island), traffic is forwarded to the IE,
4. In the IE the VPLS_ID is examined in order to map the packet to a specific VPN, the label is popped and the VLAN_ID is changed according to the local (VLAN_ID, VPLS_ID) mapping.

Forwarding Multi-domain VPLS traffic

Advancing to the multi-domain topology, the same practices as the ones already discussed about required information and forwarding of unicast and broadcast traffic are still valid. The main points regarding the multi-domain approach remain the same as the ones presented in Core Labeling (see section 3.3.4).

3.3.6 The MAC Learning mechanism

The MAC learning mechanism, which is a fundamental functionality of MPLS/VPLS as shown in section 2.2 and has been introduced in section 3.1, is responsible to associate hosts with their location by storing the (MAC address, port) combination in a PE's memory. The importance of having this mechanism embedded allows the network elements to proceed to fast and efficient routing decisions without flooding the packets to all available links. The advantage of keeping network knowledge allows for more efficient implementations and lower consumption of network resources.

In an SDN implementation, this mechanism operates by using the Packet-In event, which have been described in section 2.1. As the Controller receives these messages containing all the required information, it is possible to store the MAC addresses along with their location (switch and port). As a result, having the location of two end points it is easier for the controller to calculate the shortest path between them and install the necessary flows inside the switches.

MAC ageing and MAC withdrawal mechanisms are also needed in order to deal with user mobility and link failures. MAC ageing is achieved by using timers along with the entries of the MAC tables (MAC address, port, VPN information) for indicating if a given MAC address is still valid. Timers refresh themselves when traffic is observed from hosts. Through timers a host's current location can be determined in case of host mobility. When a timer for a given host runs out, this specific host is considered invalid and a MAC withdrawal mechanism based on Controller to Controller communication (see appendix A) is started in order to erase the host record from every MAC learning controller. The MAC withdrawal mechanism is also used when a controller receives a link-down event, indicating connection loss to a given host(s). When the connection to the host(s) is reestablished, the MAC learning mechanism is used to record the new entries.

On our SDN/VPLS architecture we are facing the following problem by using flows that match packets based on destination MAC addresses. Traffic with a destination MAC address for which the appropriate flows already exist in the OpenFlow switches, will not result in a packet-in event. Packets will not reach the controller and the MAC learning mechanism will not be triggered. In that case if the source MAC address was not yet known to the controller it will remain unknown. Replying to the same source MAC address will cause the traffic to be flooded in order to reach the unknown MAC address in the network.

For example, suppose that three hosts (A, B, C) are connected through an OpenFlow network where flows are installed based on destination MAC addresses. When there is communication between

host A and host B, flows matching the MAC addresses of host A and host B in the destination MAC field are installed. If host C decides to also communicate with host A, the following chain of events will occur as depicted in Figure 12:

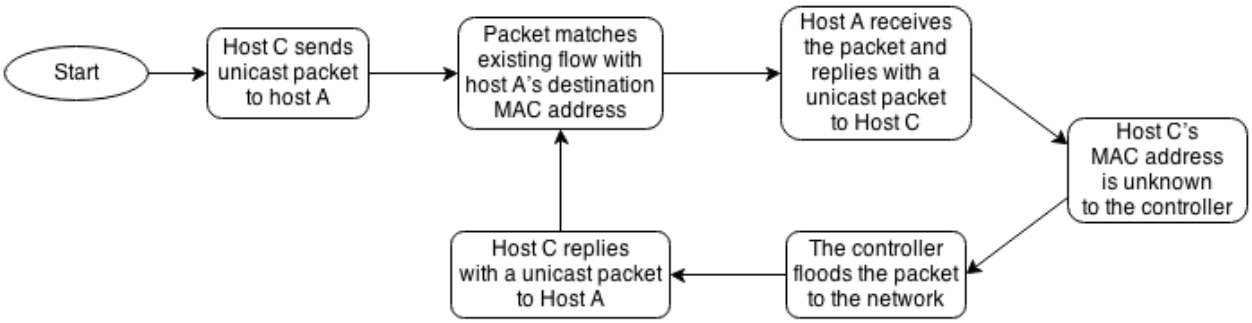

Figure 12: Flowchart of the unknown unicast problem

Host C will not be learned until both Host C and Host B stop communicating with Host A and the flow eventually expires. One can argue that Host C will start communicating by first sending an ARP request (broadcast traffic). It could still be the case that also the flow for broadcast traffic is present in the OpenFlow switch and thus leading to flooding of traffic again.

In a large scale multi-domain OpenFlow network with thousands of hosts the above scenario is not uncommon. In order to compensate for the SDN/VPLS unknown destination problem, a MAC learning mechanism to locate and learn unknown MAC addresses based on Controller to Controller communication is used.

The following steps are involved in the MAC learning mechanism of the SDN/VPLS architecture:

1. When a unicast packet to an unknown destination arrives at an Island controller, instead of flooding it in the network, the controller sends the packet in question to the Domain controller that is responsible to forward the command to all the appropriate local Islands' controllers and other Domains' controllers in the same VPN. This is accomplished via Controller to Controller communication.
2. When the Island controllers receive the command, they install a filter flow in the IE and flood the packet to the Host ports. The filter flow will match the response (source MAC address) and will send the packet to the controller, essentially creating a packet-in.
3. The Island controller upon receiving the specific packet-in, learns the MAC address, removes the filter flow, installs any flow necessary and initiates the "Force MAC learning" procedure. The "Force MAC learning" procedure essentially instructs all the MAC learning controllers in the same VPN to learn the specific MAC address.

The steps of the "Force MAC learning" procedure are the following:

1. The Island controller sends a Force MAC learning command to the Domain controller.
2. The Domain controller if it is a MAC learning controller (depends on the approach discussed earlier) learns the MAC address.
3. The Domain controller sends a MAC learning command to the appropriate local Islands' controllers and also to other Domains' controllers.
4. Eventually all Islands participating in the VPN have now learned the MAC address in question.

It should be noted that due to the nature of the Island Labeling approach, meaning the inability to flood unicast traffic throughout a VPN, the "Force MAC learning" mechanism must be triggered in every MAC learning event.

Finally, for supporting user mobility, the "Force MAC learning" mechanism should also be used when a host needs to be re-associated in a different port. This means that the host has moved to another location (port, island, etc.) and all the MAC learning controllers need to update their information.

3.3.7 Summary of designs

The Core Labeling design is based on the concept of confining the amount of information needed in each part of the network and using MPLS labels at the DEs in order to properly forward traffic. In a multi-domain scenario, the only knowledge shared between the Domains is the necessary host MAC addresses, the globally unique VPLS_IDs and the globally unique ISLAND_IDs. An optimization is discussed in section 4.2 in order to decrease the number of flows required for inter-domain unicast traffic, which also treats the ISLAND_ IDs as local information to a Domain.

The Island Labeling design uses minimal knowledge at the core network. Despite the fact that the Domain Controllers are completely unaware of the local information of each Island, it is feasible to forward unicast and broadcast traffic efficiently based on the global identifiers. Controller to Controller communication is heavily required for this design. As in the Core Labeling, the architecture is able to operate unmodified in the multi-domain environment.

The next table, Table 4, presents the differences of the two designs:

CORE LABELING	ISLAND LABELING
Usage of one MPLS tag	Usage of two MPLS tags
MAC table present on domain controllers	Domain controllers do not keep MAC addresses
Islands need only local VLAN/VPLS mapping	Islands need global VLAN/VPLS mapping
Packets are being prepared to match island flows at the provider's domain	Packets are being prepared to match island flows at the sending islands

Table 4: Core and Island Labeling differences

4 Open issues when using SDN

The components and the protocols that we used in our architecture helped us solve the majority of the problems that we faced in order to provide a complete design. However, we observed that there are some areas that can be further improved for automation, performance or overall functionality.

4.1 Multi-domain discovery

As we mentioned in chapter 3, the architecture that we developed needs to fulfill the requirement of establishing Multi-Domain connectivity between hosts located in different islands belonging in different providers. The MAC Learning mechanism and the flows that can be installed in the OpenFlow switches have been developed with the principle that our overall solution is going to operate in a dynamic environment which has the potential to grow in size. Therefore, new domains can be connected and the overall network topology can be expanded.

In order to achieve connectivity between the Domain controllers, network administrators need to reconfigure the domain controllers with the new ingress/egress ports and confirm that the new topology changes are consistent. Our idea to automate this procedure is based on expanding the Link Layer Discovery Protocol [10], which has already been adopted by the vast majority of the open source OpenFlow Controllers. According to the specification, it is possible to create custom LLDP packets by using the type 127 in the TLV format. Thus, we can expand the protocol with the following three variables:

- IP, the IP address of the OpenFlow controller orchestrating the corresponding topology
- Level**,** indicates the level of each domain in order to create the hierarchical model which will allow the providers to apply network policies,
- D-ID**,** the ID of each domain which will allow us to proceed to domain level traffic aggregation.

Both approaches introduced in chapter 3 use CTRL-to-CTRL communication for solving problems related to MAC learning but this type of communication is expected to take place above Layer 3. Thus, all the controllers need to know the IP addresses of their neighbor controllers and the IP parameter can help us automate this procedure.

The Level parameter can be used for helping each controller clarify its position on the network hierarchy. Domain controllers play a different role from the island controllers and different type of information is needed to be stored in their memory. Therefore, the Level parameter can help its controller application distinguish what type of neighbor is the connected to the corresponding port and apply the required policies.

Finally, the D-ID parameter is an optimization variable that can be used for aggregating traffic per domain. Instead of installing flows per island at core OpenFlow switches for every customer site that participates in our architecture, we can install flows pointing to each provider's domain. Thus, it is possible to match all the packets travelling to the same domain whether they come from the same or a neighboring domain. The idea of traffic aggregation per domain is going to be described in detail at the next chapter.

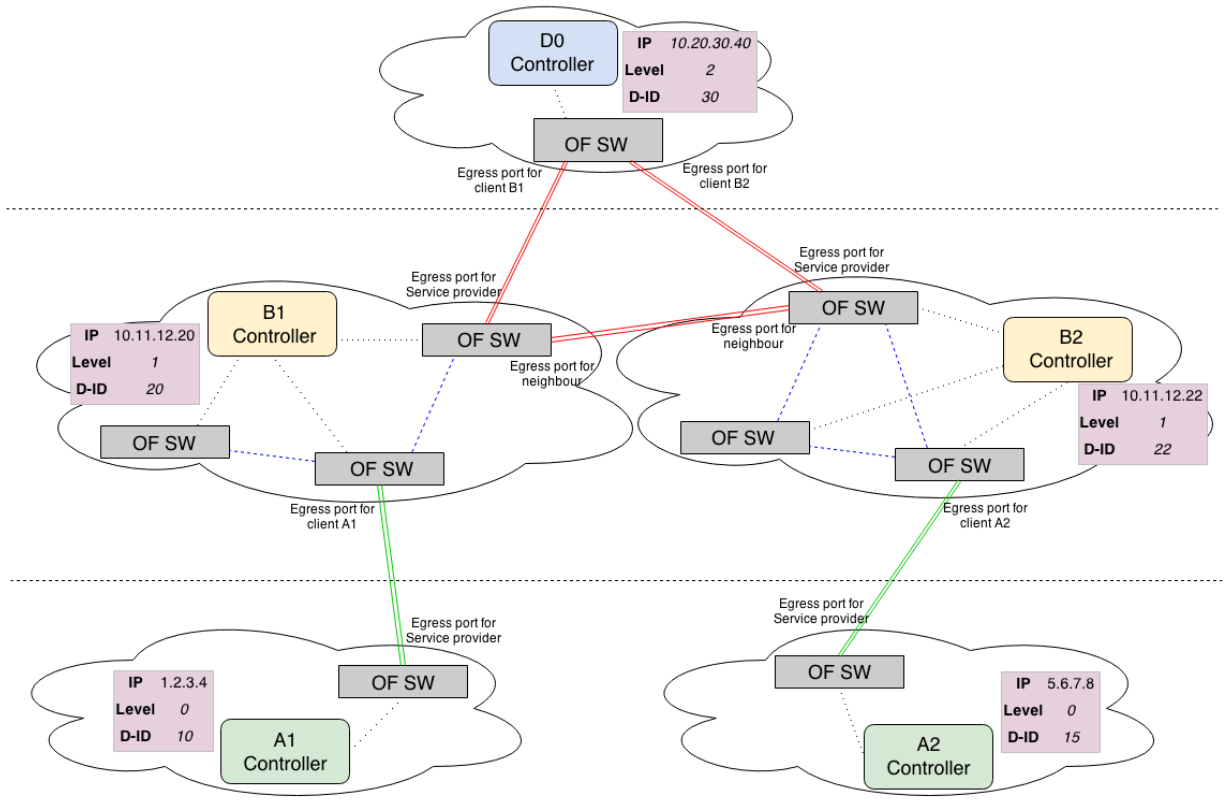

Figure 13: Multi-domain LLDP discovery mechanism

The topology of Figure 13 is an example demonstrating the expected result of having controllers in a Multi-Domain environment sending custom LLDP packets with the mentioned parameters. Through OpenFlow Packet-In events each controller not only receives its own LLDP packets but also the packets that are being sent by the neighbor controllers. Hence, it can construct a map of its own topology and use the LLDP packets of the other controllers to identify the ingress and egress ports of its domain.

In addition, the information included inside each packet allows for administration automation and traffic aggregation. The ultimate purpose of this idea is to make network administrators of each domain participating in the topology to be responsible for providing only the basic configuration of their OpenFlow controller.

4.2 Traffic aggregation at core network

The number of flows at the core network of each provider is a crucial part of our architecture as a high amount of traffic is expected to be exchanged between the domains. Therefore, the number of flows at the core switches needs to be kept to a minimum for fast lookups and low memory consumption.

This idea is based on the fact that unicast packets targeting islands which belong to the same foreign domain can match one flow instead of multiple ones. Since they have to follow the same path and similar actions need to be taken, it is possible to use a global identifier marking all these

packets. The suggested identifier that will allow this idea to be implemented is a Domain Identifier (DOMAIN_ID), which is unique between the participating domains.

Different approaches of using the DOMAIN_ID have been developed through our research, each one with different size of this variable. The first approach suggests a 20-bit DOMAIN_ID, which can be easily inserted inside an MPLS label. Hence, Domain Controllers can install flows matching MPLS labels carrying different DOMAIN_IDs and guide them correctly to the corresponding egress ports.

This suggestion is scalable, since it allows for 2^{20} different domains to coexist in the overall network topology. However, in order to be implemented it requires, from both the Core and Island Labeling approaches, to insert an additional MPLS label on every packet.

In case of simpler topologies where few provider domains coexist and packet overhead plays an important role for network architects, the following suggestions can be more efficient without changing the scope of the idea. They both follow the idea that the MPLS label containing the ISLAND_ID can be divided in two parts where DOMAIN_ID and ISLAND_ID coexist but each one has different length and position inside the MPLS Label.

Since the number of islands is expected to be greater than the number of provider domains, it is suggested for DOMAIN_ID to have a fixed size of 8 bits allowing for the rest 12 bits of the MPLS label to be used for the ISLAND_ID. So it is possible for the value of the label to include both identifiers and use either the first one or the second one according to the routing requirements. For example the controller could install flows that match packets at the first 8 bits of their MPLS label in order to aggregate them and forward them to another domain. Packets that need to stay inside the provider's domain and be forwarded to another island, could match only the other 12 bits of the MPLS label.

Unfortunately, OpenFlow 1.3 does not allow MPLS Label masking and this idea is unfeasible to be implemented nowadays. In order to provide a more feasible one, it is possible to use the first bit of the MPLS label as a separator, which defines the value of the following bits. For example, if the first bit of the MPLS label is 0 (providing a range from $0\text{-}2^{19}$), the value of the label could be the ISLAND_ID and therefore the controller can route the packet accordingly.

However, if the first bit is 1 indicating that the value of the label is a DOMAIN_ID, the controller could install flows that aggregate all the Domain traffic and forward the corresponding packets to the same egress port. This idea is applicable only to the Core Labeling approach since the Domain controllers hold knowledge about each host (MAC address + VLAN), which is required to take further routing decisions when the aggregated packet arrives at the DE device.

4.3 ARP Host discovery

In this section we discuss an alternative to the unknown MAC destination problem introduced on section 3.3.6 by installing flows which match on destination ignoring the source of the traffic. It can be used if the VPN traffic is solely IPv4.

Our alternative approach is to use ARP in order to locate and learn an unknown host. An additional MAC table (UNKNOWN_MAC_TABLE) is required on MAC learning controllers, which holds the following values:

- MAC_DST, the unknown destination MAC address,

- IP_DST, the IP of the unknown destination host,
- VPN_INFO, information about the VPN the host belongs to. It could be the VPLS_ID or the VLAN_ID based on the previous designs,
- TIME, information on the MAC address' time of entry. It will be further used for comparison with threshold values in order for different actions to take place. Two threshold values are available: Local, Remote.

A generic description of the procedure is given but it could be implemented in both our approaches with slight changes according to the specific architecture. The following steps are taken in order to resolve an unknown host using ARP:

1. When a controller receives a packet with an unknown destination MAC address it creates a record in the UNKNOWN_MAC_TABLE.
2. The controller creates a custom ARP request packet with (MAC_DST, IP_DST, CUSTOM_MAC_SRC, CUSTOM_IP_SRC, VLAN) and floods it to the same VPN host ports of its Island. An answer is to be expected within the Local threshold. Custom values should be used in order to reliably identify the reply.
3. In case the Local threshold is reached the other Islands' controllers are contacted with the (MAC_DST, IP_DST, VPN_INFO) information via Controller to Controller communication in order to create custom ARP requests and flood them to the same VPN host ports. An answer to the originating controller is expected within the Remote threshold.
4. When the Remote threshold is reached a number of actions can be taken:
    a. Restart the procedure when the same unknown MAC address is encountered, or
    b. Remove the record from the UNKNOWN_MAC_TABLE and install a flow with a hard timeout to instruct the switch to drop packets with the unknown MAC destination for a period of time. The procedure will restart when the flow has expired and a packet with the same unknown MAC address is encountered. This can act as a countermeasure to a DoS attempt.

While the timer is active, meaning neither the Local nor the Remote threshold is reached, traffic to the unknown destination should be dropped via a flow for example. This way, controller resources will not be consumed until an action can be taken.

When an ARP reply is received the already described "Force MAC learning" mechanism is used in order for the MAC address to be known. Any MAC learning mechanism also needs to check if the MAC address is present in the UNKNOWN_MAC_TABLE and remove it.

Compared to our first solution a number of advantages exist:

- It is not required for the whole packet to be sent via Controller to Controller communication, only the relevant information to craft the ARP request is required,
- The commands given by the Island controller to the switch are reduced to only one, flood the ARP request, instead of flooding the packet, installing the filter flow and later removing the filter flow.

However, this solution only works if IPv4 is used at the Internet layer.

4.4 Multi-domain broadcast loops

In our proposed architecture we use split-horizon to effectively avoid broadcast loops. In this way broadcast loops can be avoided in the Island and Domain scope. However, problems with broadcast loops arise when we consider multi-domain connectivity because split-horizon heavily relies on the loop-free topology of the network. Figure 14 depicts such a case:

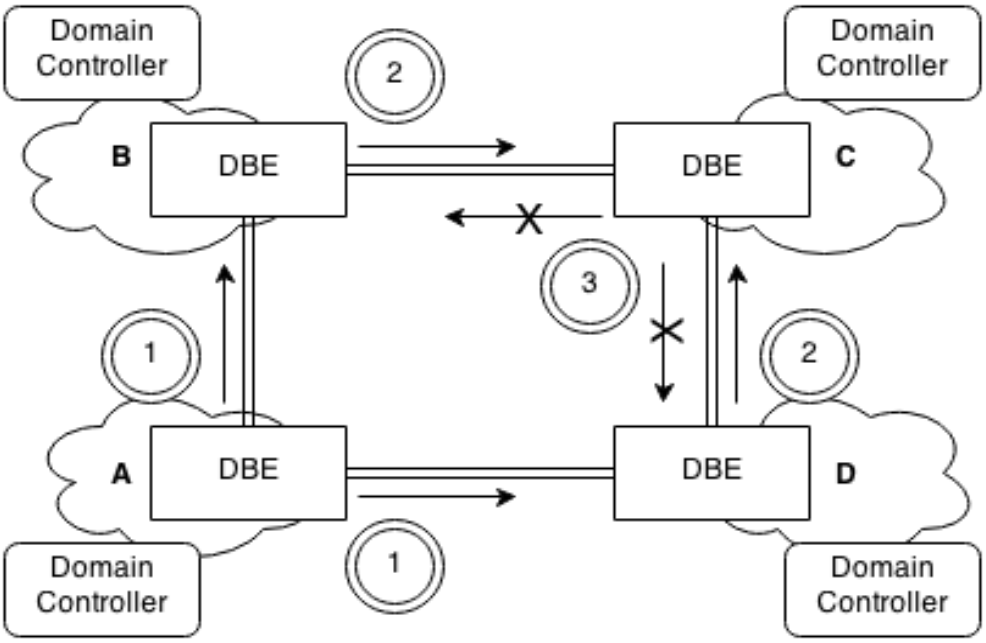

Figure 14: Multi-Domain broadcast loops

The following steps are taken in the above figure:

1. Broadcast traffic is generated from Domain A,
2. In the arriving domains split-horizon is used and traffic is forwarded to Domain C,
3. Broadcast traffic problem:
    a. Domain C receives the same broadcast traffic from two different ports, thus resulting in flooding the same traffic twice in its Islands,
    b. Split-horizon is still used but nevertheless, traffic is forwarded back to Domains B and D from traffic arrived from Domains D and B respectively.

Possible solutions for dealing with the broadcast loops while practicing split-horizon forwarding are:

- Ensure the loop-free connectivity between domains,
- Full mesh connectivity between the domains. Along with the split-horizon method, the domains should not forward broadcast traffic that came from an inter-domain connection to other domains, similar to how iBGP works.
- Use of a Spanning Tree Protocol to ensure a loop-free topology. STP can run on the Domain controllers and address only the Inter-domain ports. Instead of disabling the links, as the classic STP does, it will mark the Inter-domain connections as unusable for broadcast traffic. For example, in Figure 14 the connection from Domain A to Domain B could be marked

unusable for broadcast traffic, but unicast traffic from Domain A to Domain B could still traverse the same link.

On one hand, ensuring the loop-free connectivity between domains is a straight-forward solution. However, careful planning is required whenever topology changes are needed. For example when a domain leaves the network, the network topology may need to be redesigned to avoid broadcast loops. On the other hand, full mesh connectivity is trivial to achieve by interconnecting the domains, but comes with a high cost on setup and maintenance. Finally, using STP on the Inter-domain ports increases the complexity of the architecture but offers the less human intervention.

4.5 A web portal for SDN based VPLS

The critical problem of learning the hosts which participate in the various VPLS instances has been solved through the MAC learning mechanism, described in detail in section 3.3.6. The mechanism not only allows the OpenFlow controllers to associate hosts with VPLS instances and ports automatically, but also solves the problem of unknown unicast, which is common in our architecture.

In case all the necessary information regarding the hosts (e.g., MAC address, switch-port) can be provided beforehand, OpenFlow Controllers can receive this information and establish the necessary paths. A similar project, CoCo [2], goes for a similar idea where a central web portal provides the required information to the network elements responsible for providing end-to-end connectivity.

Therefore, a web portal which will allow for user registration/deregistration will play an important role in our architecture. This portal will provide a way to match a specific machine (MAC address) behind a certain port on the IE to a certain VPN (VPLS_ID, VLAN_ID). This information could then be passed to the appropriate controllers. How this mechanism could work is out of the scope of this research but a brief description is given. An application could be downloaded to the machine and by sending a special packet, that will be captured by the controller, the web portal could learn the (MAC address, switch, port) information via a controller communication. The web portal could then push all the necessary configuration to the appropriate controllers (MAC address, VLAN_ID, VPLS_ID, etc.) and finally instruct the application to assign a specific VLAN_ID to the machine's interface.

By having all this information beforehand the on-demand VPLS is facilitated and the need for any MAC learning/ageing mechanism is eliminated. Packet-Ins do not trigger MAC learning and timers are not used for MAC ageing purposes. In addition the unknown MAC destination problem no longer exists as all the MAC addresses are universally known. In contrast, MAC withdrawal mechanisms are still needed for deregistration purposes.

The total number of changes needed in our architecture if the web portal solution is adopted, have been described in detail in APPENDIX C. Based on these modifications, our architecture becomes simpler due to the absence of MAC learning/ageing mechanisms. As a result Controller to Controller communication functions are also reduced.

5 Discussion

We conducted a deep technical analysis in our network architecture for both designs in three different fields:

- The size of the information tables that OpenFlow Controllers keep in memory,
- The type and number of unicast and broadcast flows that need to be installed in the OpenFlow switches,
- The functions and their parameters required for implementing the mechanisms of the architecture,

The results can be found in APPENDIX A and APPENDIX B**.** The total number of supported VPLSes and Islands is 2^{20} and is bound by the 20-bit length of the MPLS label. Each Island can support participation in up to 4096 of those VPLSes for its hosts. This is a limit of the 12-bit length of the VLAN tag that is used to represent the local VPNs. In addition, the number of unicast flows at the OpenFlow switches increase linearly by the number of hosts and the number of broadcast flows varies depending on the combination of (IN_PORT, VLAN/VPLS ID). Thus, it is very convenient for network engineers to predict the amount of resources required based on the number of the hosts involved.

Comparing the existing MPLS/VPLS architecture with the SDN/VPLS architecture, the major difference is the absence of the classical PE devices located at the edges of a provider's domain. They are replaced by common OpenFlow switches, which communicate with the domain controller for receiving the required instructions. The OpenFlow controller, which runs on a commodity server, is now the only entity on the domain responsible for gathering network information. The controller's MAC learning capabilities are further expanded as commodity servers do not have the low memory limitations that accompany ASIC devices (i.e., PEs). Due to the increasing amount of information that follows a multi-domain VPLS architecture, the centralization of information can contribute to lower setup and maintenance costs and thus to the economy of scale.

In addition, since the OpenFlow controller is able to hold the knowledge of the network topology that it is managing, there is no need for full mesh connectivity between the switches. The controllers are able to calculate the shortest path between two single points of the network and install the necessary flows. Moreover, SDN technology removes the need of creating pseudowires and extra functions like signaling and label exchange are no longer required.

Our architecture is able to provide flexible on demand VPLS in a scalable way which is able to operate in a multi-domain environment. In addition, according to our hypothesis, an implementation is feasible since all the network level decisions that we made are fully compliant with OpenFlow 1.3. The only part of the architecture which needs to be developed from the beginning is the Controller-to-Controller communication.

In order to implement all these extended functionalities that are required for both the designs that we developed, more sophisticated controllers are required. The existing open source implementations provide a good platform for implementing a custom controller for our architecture. However, it is also required for a communication channel or northbound interface to be defined in order to achieve Controller-to-Controller communication.

Moreover, the usage of SDN technology means that we have to use software to manage the network topology, which automatically leads to procedure automation and design freedom. This means that

the overall solution is easily extendable and provides a green field for new experimental implementations.

Finally, both designs combine OpenFlow with VLAN and MPLS in order to provide end-to-end, multipoint-to-multipoint connectivity solutions. Therefore, the scalability of our architecture is bound to the scalability of these protocols.

6 Conclusion

This report is a preliminary study on a network architecture that aims to provide VPLS connectivity by using the benefits of the SDN technology. Following the successful example of MPLS/VPLS, the project is targeting on providing a complete, flexible and scalable solution for creating and establishing virtual networks on demand.

Through our work we were able to provide answers to our requirements by combining OpenFlow 1.3 with existing protocols like VLAN and MPLS. Therefore, we developed two different designs which we named “Core Labeling” and “Island Labeling”. The key differences between them are the way they handle the MPLS labels and the amount of Island information that is required on a Domain OpenFlow controller. This leads to the conclusion that network architects can choose the design which is suitable for their needs according to their available resources. Hence, it is difficult for us to choose the architecture of our preference since each one has its own positives and requirements.

Further analysis of both designs yielded positive results regarding scalability and consumption of network resources, which lead to the hypothesis that a possible implementation of the architecture is feasible in the near future. The SDN/VPLS architecture promises increased flexibility as the users can join VPNs on demand. System administrators can benefit from the minimum configuration required while the software nature of SDN allows for further automation. Moreover, proposed ideas and optimizations promise to provide a more efficient and complete architecture.

7 Future work

As of now, our architecture does not take into account any QoS requirements. QoS can be implemented by using the Meter table functionality introduced in OpenFlow Switch specification 1.3. The Meter table allows defining per-flow meters which can be a way to implement simple QoS functions such as rate-limiting. If it is to be also combined with the existing per-port queues of the switch, more advanced QoS operations such as DiffServ could be achieved.

Our architecture could be further expanded by taking multicast traffic into consideration and providing solutions for better traffic shaping and aggregation. Multicasting is by default a layer 3 functionality and mostly used in the form of IP multicasting using IP multicast groups. Ethernet multicasting is achieved by using well-known Ethernet multicast addresses in the destination MAC address field. Packets having an Ethernet multicast address in the destination MAC address field still need to be flooded in the network but they are only accepted by specific hosts, e.g., bridges accept packets with the well-known Spanning Tree Protocol multicast MAC address.

The behavior of the network regarding Ethernet multicasting (e.g., forwarding, flooding, dropping multicast packets) can be shaped according to administrative needs. Furthermore, RFC 1112 defines extensions to the Ethernet Local Network Module in order for IPv4 and IPv6 multicast addresses to be mapped to specific Ethernet multicast addresses.

Regarding the MAC aging timers, a well-known and reliable protocol such as RARP can be used in order to update them and correctly identify if a host is down. However, this approach can be used only in case of having hosts adopting the TCP/IP stack. In case of using other experimental protocols, for example in the CoCo environment where a VPLS could be used for networking experiments, a more abstract strategy should be defined and adopted by the network engineers.

Research [11][12] on the performance of OpenFlow presents promising results but the experiments are now outdated as they used old OpenFlow versions or OpenFlow-enabled (limited capabilities) hardware switches. An updated performance evaluation on OpenFlow 1.3 using pure OpenFlow hardware switches combined with a prototype of our architecture could provide better insight on OpenFlow's efficiency in providing software defined VPNs.

8 APPENDICES

8.1 APPENDIX A - Technical Description

We present the minimum information that is needed inside the OpenFlow controllers in order to install the appropriate flows and forward unicast and broadcast traffic correctly. The variety of flows that has to be installed in all type of intermediate switches is also examined. Lastly, we observe how both architectures scale when the number of hosts increases.

Operational details

This section presents the information needed to be stored in Island or Domain controllers for both "Island labeling" and "Core labeling".

This section presents the information needed to be stored in Island or Domain controllers for both "Island labeling" and "Core labeling".

Core Labeling - Information tables

The Island Controller maintains the following tables:

- MAC Table
  `[ MAC, Local_VLAN, PORT ]`

  Holds the required information to associate a MAC address to a port and a VPN. Used for forwarding unicast packets.

- Broadcast Table
  `[ PORT, Local_VLAN ]`

  Holds the information required for flooding broadcast packets.

The Domain Controller maintains the following tables:

- MAC Table
  `[ MAC, VPLS_ID, ISLAND_ID ]`

  Holds the required information to associate a MAC address to a specific island and VPN.

- Local Island Information
  `[ ISLAND_ID, Local_VLAN, VPLS_ID ]`

  Holds the information required for flooding broadcast packets. Local_VLAN is needed because the Domain controller is responsible to change the VLAN tag to the appropriate VLAN_ID value used inside an Island for a given VPLS_ID.

- Remote Island Information
  `[ ISLAND_ID,  -  , VPLS_ID ]`

The use is the same as the Local Island Information table but no VLAN information is needed for remote Islands.

- Island Location
  `[ SWITCH_ID, PORT, ISLAND_ID]`

  Simple mapping between an ISLAND_ID and the actual location of the Island in regard to the topology.

Island Labeling - Information tables

The Island Controller maintains the following tables:

- MAC Table
  `[ MAC, VLAN_ID, VPLS_ID, ISLAND_ID, PORT ]`

  Due to the Island controller being responsible for most of the decisions for packet forwarding and marking, it needs all the information available for a given host. It should be noted that the VLAN_ID refers to the VLAN_ID used in the home Island of the host.

- Broadcast Table
  `[ VPLS_ID, VLAN_ID, ISLAND_ID, PORT ]`

  Holds the information required for flooding broadcast packets to the Islands. When the controller starts operating it needs to know only its local information. The rest of the table can be constructed while MAC learning is practiced.

The Domain Controller maintains the following table:

- Information Table
  `[ SWITCH_ID, PORT, ISLAND_ID, VPLS_ID ]`

  The minimum information needed in the Domain controller in order to forward unicast traffic to an Island or handle broadcast traffic for specific VPN Island targets.

Network details

This section presents the flows that can be found inside the OpenFlow switches in order to match and forward the packets as needed. The different numbers that may appear in the actions indicate actions that can be taken exclusively or inclusively depending on the situation. Exclusivity can be achieved in OpenFlow by using the Group table along with Action Buckets.

Core Labeling - Flows

The following flows could be present in an IE device according to the nature of the traffic:

- Unicast - Flow to local or remote HOST
  ```
  ( Match : MAC_DST, VLAN
    Action: Send to Port )
  ```

  By matching the VLAN and the destination MAC address for unicast traffic, traffic aggregation can be achieved.

- Broadcast - Flow

```
( Match : MAC_BROADCAST, VLAN, IN_PORT
 Action: Send to Port(s) )
```

For broadcast traffic, except for the broadcast MAC address and the VLAN, one needs also to match the ingress port of the packet in order to exercise split-horizon and not forward traffic back to the source.

The following flows could be present in a DE device according to the nature of the traffic:

- Unicast - Flow to remote Island HOST

```
( Match : MAC_DST, VLAN, (IN_PORT)
  Action: Push VPLS_ID, ISLAND_ID
          Send to Port )
```

In order for unicast traffic to reach a HOST in a remote Island (Island not attached to this DE), the destination MAC address along with the VLAN can be matched. Based on this distinction the appropriate MPLS labels can be pushed. IN_PORT is optional and is used in case the DE is attached to multiple Islands.

- Unicast - Flow to local Island HOST

```
( Match  : ISLAND_ID, VPLS_ID
  Actions: Pop ISLAND_ID, VPLS_ID
           Change VLAN
           Send to Island Port )
```

In order for unicast traffic to reach a HOST in a local Island (Island attached to this DE), the ISLAND_ID and VPLS_ID MPLS labels can be matched. Based on this distinction the MPLS labels can be popped and the appropriate VLAN value can be used based on the Local Island Information table.

- Broadcast - Flow to other Islands

```
( Match: BROADCAST_MAC, VLAN, IN_PORT
  Action1(Remote Islands): Push VPLS_ID, BROADCAST_MPLS
                           Send to Port(s)
 Action2(Other Local Islands): Change VLAN
                               Send to Port )
```

For broadcast traffic originated from a local Island the broadcast MAC address along with the VLAN need to be matched. Based on this information the appropriate VPLS_ID and BROADCAST_MPLS MPLS labels can be pushed for traffic that needs to reach remote Islands. In case traffic needs to reach another Island in the same DE it simply needs to change the VLAN to the appropriate value. Again the ingress port is matched for split-horizon purposes.

- Broadcast - Flow to local Island(s)

```
( Match   : BROADCAST_MPLS, VPLS_ID, IN_PORT
  Action1 : Pop BROADCAST_MPLS, VPLS_ID
            Change VLAN
             Send to Island Port(s)
```

```
 Action2(Transit Broadcast Traffic): Send to Port(s) )
```

For broadcast traffic originated from a remote Island the BROADCAST_MPLS and VPLS_ID MPLS labels need to be matched. Based on this information the MPLS labels can be popped and the appropriate VLAN based on the Local Island Information table can be used. In case the DE is not a leaf in the multicast tree, the packets need to continue traversing the tree unchanged, hence the second action. Again the ingress port is matched for split-horizon purposes.

The following flows could be present in a D device according to the nature of the traffic:

- Unicast - Transit Flow

```
( Match : ISLAND_ID
  Action: Send to Port )
```

D devices need only to match the ISLAND_ID in order to properly forward unicast traffic.

- Broadcast - Transit Flow

```
( Match : BROADCAST_MPLS, VPLS_ID, IN_PORT
  Action: Send to Port(s) )
```

D devices need only to match the BROADCAST_MPLS and VPLS_ID in order to properly forward broadcast traffic. Again the ingress port is matched for split-horizon purposes.

Island Labeling - Flows

The following flows could be present in an IE device according to the nature of the traffic:

- Unicast - Flow to local HOST

```
( Match : MAC_DST, VLAN
  Action: Send to Port )
```

By matching the VLAN and the destination MAC address for unicast traffic to a local host, traffic aggregation can be achieved.

- Unicast - Flow to remote HOST

```
( Match : MAC_DST, VLAN
 Actions: Change VLAN
           Push ISLAND_ID
           Send to port )
```

By matching the VLAN and the destination MAC address for unicast traffic to a remote host, traffic aggregation can be achieved. In this case the VLAN needs also to be changed, based on the MAC table, in order to reflect the appropriate value used by the destination Island. The destination Island's ISLAND_ID MPLS label needs to be pushed as a final action before forwarding the packet.

- Broadcast - Flow for packets generated inside the Island

```
( Match                       : MAC_BROADCAST, VLAN, IN_PORT
 Action1(Send to local HOSTS) : Send to Port(s)
 Action2(Send to remote HOSTS): Push VPLS_ID
                                Send to DOMAIN_PORT )
```

For broadcast traffic generated inside the Island, the broadcast MAC address along with the VLAN need to be matched. Based on the Broadcast table the appropriate VPLS_ID MPLS label can be pushed. In case there are recipients inside the Island the packet needs to be flooded unchanged to these host ports. Again the ingress port is matched for split-horizon purposes.

- Broadcast - Flow for packets arriving at the Island

```
( Match  : MAC_BROADCAST, VPLS_ID, IN_PORT
 Actions: Pop VPLS_ID
          Change VLAN
          Send to port(s) )
```

For broadcast traffic arriving at the Island, the broadcast MAC address along with the VPLS_ID need to be matched. The MPLS label then gets popped and based on the Broadcast table the appropriate VLAN value can be used and the traffic can be flooded to the appropriate hosts. Again the ingress port is matched for split-horizon purposes.

The following flows could be present in a DE device according to the nature of the traffic:

- Unicast - Flow to remote Island HOST

```
( Match : ISLAND_ID
  Action: Send to Port )
```

For unicast traffic to a host residing in a remote Island only the ISLAND_ID needs to be matched in order to properly forward the packets.

- Unicast - Flow to local Island HOST

```
( Match  : ISLAND_ID
  Actions: Pop ISLAND_ID
           Send to Island Port )
```

For unicast traffic to a host residing in a local Island only the ISLAND_ID needs to be matched and popped before sending the packet to the Island.

- Broadcast - Flow to remote or local Islands

```
( Match  : MAC_BROADCAST, VPLS_ID, IN_PORT
  Actions: Send to Port(s) )
```

For broadcast traffic to remote or local Islands, only the broadcast MAC address along with the VPLS_ID need to be matched in order to properly forward packets. Again the ingress port is matched for split-horizon purposes.

The following flows could be present in a D device according to the nature of the traffic:

- Unicast - Transit Flow

```
( Match : ISLAND_ID
  Action: Send to Port )
```

D devices need only to match the ISLAND_ID in order to properly forward unicast traffic.

- Broadcast - Transit Flow

```
( Match : MAC_BROADCAST, VPLS_ID, IN_PORT
  Action: Send to Port(s) )
```

D devices need only to match the broadcast MAC address and VPLS_ID MPLS label in order to properly forward broadcast traffic. Again the ingress port is matched for split-horizon purposes.

Functional details

This section presents the functions that are required to be implemented inside the OpenFlow controllers for both designs.

Core labeling - Controller Functions

The functions present in an Island Controller are the following:

- LearnTopology()
  Learn the topology of the managed network including the different OpenFlow switches and the Domain Controller,

- LearnMAC (*mac_src, vlan, port*)
  Associate a specific MAC address with a port and a VLAN. Used for MAC learning purposes,

- ForgetMAC ([*mac,vlan*])
  Remove the given combinations of MAC addresses and VLAN_IDs from the MAC table. Used for MAC withdrawal purposes,

- FloodPacket (*packet, vlan*)
  Flood the given packet to all ports participating in the given VLAN. Used as part of the unknown host solution in order to locate the unknown host,

- CTRL-to-CTRL: ForceMAC (*domain_ctrl, mac, vlan*)
  Send command to Domain Controller to initiate the "Force MAC learning" mechanism for a given MAC address,

- CTRL-to-CTRL: LocateHost (*domain_ctrl, packet*)
  Send command to Domain Controller to initiate the unknown host problem's solution and locate an unknown host. The packet that triggered the event is also sent in order to be flooded, by the FloodPacket function, in the other Islands.

The functions present in a Domain Controller are the following:

- LearnTopology()
  Learn the topology of the managed network including the different OpenFlow switches, the Domain's Island Controllers and other Domain Controllers,

- UpdateVPLSConf (*island, l_vlan, vpls*)
  Update the required VPLS configuration, meaning the association between VPLS_ID and VLAN_ID for a given Island,

- LearnRemoteVPLSConf (*island, vpls*)
  Learn the required VPLS configuration for distant Islands. No VLAN information is needed for Islands residing in another Domain,

- LearnMACNative (*mac_src, vlan, island*)
  Associate a specific MAC address from a local Island with the specific Island and VPLS_ID. The VPLS_ID can be derived by the VLAN information combined with the Local Island Information table. Used for MAC learning purposes,

- LearnMACRemote (*mac_src, vpls, island*)
  Associate a specific MAC address from a distant Island with the specific Island and VPLS_ID,

- ForgetMAC ([*mac, vpls*])
  Remove the given combinations of MAC addresses and VPLS_IDs from the MAC table.
  Used for MAC withdrawal purposes,

- CalculatePath (*sw1*, *port1*, *sw2*, *port2*)
  Calculate the shortest path between the given ports. A shortest path algorithm such as Dijkstra's algorithm can be used,

- BuildTree (*vpls*)
  Create a multicast tree containing the local Islands which share the same VPN,

- CTRL-to-CTRL: WithdrawMACLocal ([*IslandCtrls*], [*mac, vpls*])
  Send command to Island Controllers to forget the specific combinations of MAC addresses and VLAN_IDs. It is triggered by a link down event on a DE-IE connection. The MAC addresses belonging to that Island must be withdrawn. It triggers the ForgetMAC function of the Islands's Controllers with the appropriate VLAN_ID for these MAC addresses. It also triggers its own ForgetMAC function,

- CTRL-to-CTRL: WithdrawMACRemote ([*DomainCtrls*], [*mac, vpls*])
  Send command to other Domains' Controllers to forget the specific combinations of MAC addresses and VPLS_IDs. It is triggered by a link down event on a DE-IE connection. The MAC addresses belonging to that Island must be withdrawn. It will eventually trigger the WithdrawMACLocal function of those controllers,

- CTRL-to-CTRL: ForceMACLocal ([*IslandCtrls*], *mac, vpls*)
  Send command to Island Controllers to associate a specific MAC address with the given VPN. It will eventually trigger the LearnMAC function of the Island Controller.The port that the MAC address will be assigned to is the Domain Port of the IE,

- CTRL-to-CTRL: ForceMACRemote ([*DomainCtrls*], *mac, vpls*)

Send command to other Domains' Controllers to associate a specific MAC address with the given VPN. It will eventually trigger the ForceMACLocal function of the Domains' Controller. The port that the MAC address will be assigned to is the Inter-Domain Port of the DBE,

- CTRL-to-CTRL: LocateHostLocal ([*IslandCtrls*], *packet, vpls*)
  Send command to Island Controllers to locate an unknown host by installing a filter flow. It is triggered by the LocateHost function of an Island Controller. The packet that triggered the event is also sent in order to be flooded, by the FloodPacket function, in the other Islands,

- CTRL-to-CTRL: LocateHostRemote ([*DomainCtrls*], *packet, vpls*)
  Send command to other Domains' Controllers to locate an unknown host. It will eventually trigger the LocateHostLocal function of the Domains' Controller.

Island labeling - Controller Functions

An Island Controller needs to have the following actions implemented:

- LearnTopology
  Learn the topology of the managed network including the different OpenFlow switches and the Domain Controller,

- LearnMAC (*mac_src, vlan, port, vpls, island*)
  Populate the MAC table with all the required information for the specific MAC address. Used for MAC learning purposes,

- ForgetMAC ([*mac, vpls*])
  Remove the given combinations of MAC addresses and VPLS_IDs from the MAC table. Used for MAC withdrawal purposes,

- FloodPacket (*packet, vlan*)
  Flood the given packet to all ports participating in the given VLAN. Used as part of the unknown host solution in order to locate the unknown host,

- CTRL-to-CTRL: ForceMAC (*domain_ctrl, mac, vlan, vpls, island*)
  Send command to Domain Controller to initiate the "Force MAC learning" mechanism for a given MAC address,

- CTRL-to-CTRL: LocateHost (*domain_ctrl, packet, vpls*)
  Send command to Domain Controller to initiate the unknown host problem's solution and locate an unknown host. The packet that triggered the event is also sent in order to be flooded, by the FloodPacket function, in the other Islands.

A Domain Controller needs to have the following functions implemented:

- LearnTopology
  Learn the topology of the managed network including the different OpenFlow switches, the Domain's Island Controllers and other Domain Controllers,

- UpdateVPLSConf (*island, vpls*)

Update the required VPLS configuration, meaning the association between a VPLS_ID and a given Island,

- BuildTree (*vpls*)
  Create a multicast tree containing the local Islands which share the same VPN,

- CalculatePath (*sw1*, *port1*, *sw2*, *port2*)
  Calculate the shortest path between the given ports. A shortest path algorithm such as Dijkstra's algorithm can be used,

- CTRL-to-CTRL: WithdrawMACLocal ([*IslandCtrls*], [*mac, vpls*])
  Send command to Island Controllers to forget the specific combinations of MAC addresses and VLAN_IDs. It is triggered by a link down event on a DE-IE connection. The MAC addresses belonging to that Island must be withdrawn. It triggers the ForgetMAC function of the Islands' Controllers with the appropriate VLAN_ID for these MAC addresses,

- CTRL-to-CTRL: WithdrawMACRemote ([*DomainCtrls*], [*mac, vpls*])
  Send command to other Domains' Controllers to forget the specific combinations of MAC addresses and VPLS_IDs. It is triggered by a link down event on a DE-IE connection. The MAC addresses belonging to that Island must be withdrawn. It will eventually trigger the WithdrawMACLocal function of those controllers,

- CTRL-to-CTRL: ForceMACLocal ([*IslandCtrls*], *mac, vpls, vlan, island*)
  Send command to Island Controllers to populate the MAC table with all the required information for the specific MAC address. It will eventually trigger the LearnMAC function of the Island Controller.The port that the MAC address will be assigned to is the Domain Port of the IE,

- CTRL-to-CTRL: ForceMACRemote ([*DomainCtrls*], *mac, vpls, vlan, island*)
  Send command to other Domains' Controllers to populate the MAC table with all the required information for the specific MAC address. It will eventually trigger the ForceMACLocal function of the Domains' Controller. The port that the MAC address will be assigned to is the Inter-Domain Port of the DBE,

- CTRL-to-CTRL: LocateHostLocal ([*IslandCtrls*], *packet, vpls*)
  Send command to Island Controllers to locate an unknown host by installing a filter flow. It is triggered by the LocateHost function of an Island Controller. The packet that triggered the event is also sent in order to be flooded, by the FloodPacket function, in the other Islands,

- CTRL-to-CTRL: LocateHostRemote ([*DomainCtrls*], *packet, vpls*)
  Send command to other Domains' Controllers to locate an unknown host. It will eventually trigger the LocateHostLocal function of the Domains' Controller.

8.2 APPENDIX B - Scalability Analysis

We examine how the size of the information tables and the number of flows increase according to the number of hosts. In addition, we pose our limits based on the packet marking technologies that we use in the Island and Core network.

Scalability on Core labeling

First, we examine the scalability of the Controller tables used for forwarding unicast and broadcast traffic.

Scalability on Information Tables

Island Controller

- MAC Table

  `[ MAC(48), Local_VLAN(12), PORT(16) ]`

Each record inside the Island's MAC table has a total length of 76 bits.
Let N_HOSTS be the number of global hosts, N_VLAN be the number of VLAN_IDs present in an Island and MAX_VLAN be the maximum number of VLAN_IDs present in an Island(4096).
The MAC table's size varies depending on N_HOSTS and N_VLAN because each host can participate in many VPNs. In the worst case scenario the size of the table will be N_HOSTS x MAX_VLAN if we consider that each MAC address can participate in MAX_VLAN different VPNs.

```
MAC Table Total Size = 76 bits x MAX_VLAN x N_HOSTS
                     = 76 bits x 4096 x N_HOSTS
                     = 38 Kbytes x N_HOSTS
```

- Broadcast Table

  `[ PORT(16), Local_VLAN(12) ]`

One record in the Broadcast table is 28 bits.
Let MAX_PORTS be the total number of used ports.
Each Island can participate in at most MAX_VLAN different VPNs. Thus, the broadcast table's size varies depending on N_VLAN and MAX_PORTS.

```
Broadcast Table Total Size = 28 bits x MAX_VLAN x MAX_PORTS
                           = 28 bits x 4096 x MAX_PORTS
                           = 14 Kbytes x MAX_PORTS
```

Domain Controller

- MAC Table

  `[ MAC(48), VPLS_ID(20), ISLAND_ID(20) ]`

Each entry of the MAC table has a total length of 88 bits.
Let N_VPLS be the number of VPLS IDs present in the global network and MAX_VPLS be the maximum number of VPLS IDs(2^{20}) present in the global network.
The MAC table's size varies depending on N_HOSTS and N_VPLS because each MAC address needs to be unique in the context of a VPN.
In the worst case scenario the size of the table will be N_HOSTS x MAX_VPLS if we consider that each MAC address can participate in MAX_VPLS different VPNs.

```
MAC Table Total Size = 88 bits x N_VPLS x N_HOSTS
```

- Local Island Information

```
[ ISLAND_ID(20), Local_VLAN(12) , VPLS_ID(20) ]
```

Let N_ISLANDS be the total number of local Islands present in a domain.
Each entry of the Local Island Information table has a total length of 52 bits.
The Local Island Information table's size varies depending on N_ISLANDS and N_VLAN because each Island can participate at MAX_VLAN VPNs at most. In the worst case scenario all the Islands would use MAX_VLAN.

```
Local Island Information Table Total Size = 52 bits x MAX_VLAN x
                                                    x N_ISLANDS
                                = 52 bits x 4096 x N_ISLANDS
                                = 26 Kbytes x N_ISLANDS
```

- Remote Island Information

```
[ ISLAND_ID(20),      - (0)      , VPLS_ID(20) ]
```

Each entry of the Remote Island Information table has a total length of 40 bits.
The Remote Island Information table's size varies depending on N_ISLANDS and N_VLAN as each Island can participate at MAX_VLAN VPNs at most. In the worst case scenario all the Islands would use MAX_VLAN.

```
Remote Information Table Total Size =
                              = 40 bits x MAX_VLAN x N_ISLANDS
                              = 40 bits x 4096 x N_ISLANDS
                              = 20 Kbytes x N_ISLANDS
```

- Island Location

```
[ SWITCH_ID(64), PORT(16), ISLAND_ID(20)]
```

Each entry of the Island Location table has a total length of 100 bit.
The Island Location table's size varies depending on N_ISLANDS.

```
Island Location Table Total Size = 100 bits x N_ISLANDS
```

Scalability on Number of flows

We examine the scalability of the number of flows used for unicast and broadcast traffic.

Island flows

Inside the flow table of an IE, the number of unicast flows varies only depending on N_HOSTS as we match packets based on destination MAC address in order to aggregate traffic.

```
Total Unicast Flows = N_HOSTS flows
```

Let MAX_PORTS be the total number of used ports on a switch.
For broadcast traffic the number of flows varies depending on N_VLAN and MAX_PORTS. The number of ports is required as we also match the ingress port for split-horizon purposes. The worst case scenario would be for all the available VLAN_IDs to be used.

```
Total Broadcast Flows = N_VLAN x MAX_PORTS flows
```

In total, also the worst case scenario, the maximum number of flows required in an IE device is:

```
Total Number of Flows = Total Unicast Flows +
```

```
                                    + Total Broadcast Flows
                          = N_HOSTS + MAX_VLAN x MAX_PORTS
                          = N_HOSTS + 4096 x MAX_PORTS flows
```

Domain Flows

Let ISLAND_HOSTS be the number of hosts present in a given Island.
Inside the Flowtable of a DE the number of unicast flows varies depending on the number of remote hosts and N_VLAN. An additional number of flows is needed to forward unicast traffic inside the Island for local hosts. This number relies only on N_VLAN as traffic is aggregated for all packets arriving at the Island using the ISLAND_ID.

```
Total Unicast Flows =
                    (N_HOSTS - ISLAND_HOSTS) x N_VLAN + N_VLAN flows
```

Let N_ISLAND_PORTS be the total number of ports attached to Islands.
For broadcast traffic we have 3 types of flows. Transit Broadcast, Incoming Broadcast and Outgoing Broadcast flows.

Transit Broadcast flows are for traffic passing through the switch in order to reach its destination and their number depends on N_VPLS and MAX_PORTS.

```
Total Transit Broadcast Flows = N_VPLS x MAX_PORTS flows
```

Incoming Broadcast flows are for traffic destined to the Island(s) attached to the DE device and their number depends on (N_VLAN, N_ISLAND_PORTS)[1] and the number of non-Island ports.

```
Total Incoming Broadcast Flows =
 = N_VLAN x N_ISLAND_PORTS x (MAX_PORTS - N_ISLAND_PORTS) flows
```

Outgoing Broadcast flows are for broadcast traffic generated in an attached Island and their traffic depends on (N_VLAN, N_ISLAND_PORTS)1.

```
Total Outgoing Broadcast Flows = N_VLAN x N_ISLAND_PORTS flows
```

In total, also the worst case scenario where a DE device is also a transit node for all broadcast traffic, the maximum number of flows required in a DE device is:

```
Total Number of Flows = Total Unicast Flows +
                      + Total Transit Broadcast Flows +
                      + Total Outgoing Broadcast Flows
```

Scalability on Island labeling

As we did in the analysis of the Core labeling, we are starting by examining the scalability of the Controller tables used for forwarding unicast and broadcast traffic.

Scalability on Information Tables

We examine the scalability of the Controller tables used for forwarding unicast and broadcast traffic.

Island Controller

- MAC Table

```
[ MAC(48), VLAN_ID(12), VPLS_ID(20), ISLAND_ID(20), PORT(16) ]
```

Each record inside the Island's MAC table has a total length of 116 bits.

Let N_HOSTS be the number of global Hosts, N_VLAN be the number of VLAN_IDs present in an Island and MAX_VLAN be the maximum number of VLAN_IDs present in an Island(4096).
The MAC table's size varies depending on N_HOSTS and N_VLAN.
In the worst case scenario the size of the table will be N_HOSTS x MAX_VLAN if we consider that each MAC address can participate in MAX_VLAN different VPNs.

```
MAC Table Total Size = 116 bits x MAX_VLAN x N_HOSTS
                     = 116 bits x 4096 x N_HOSTS
                     = 58 Kbytes x N_HOSTS
```

- Broadcast Table

```
[ VPLS_ID(20), VLAN_ID(12), ISLAND_ID(20), PORT(16) ]
```

One record in the Broadcast table is 68 bits.
Let N_ISLANDS be the number of global Islands.
Each Island can participate in at most MAX_VLAN different VPNs. Thus, the broadcast table's size varies depending on N_ISLANDS and N_VLAN.

```
Broadcast Table Total Size = 68 bits x MAX_VLAN x N_ISLANDS
                           = 68 bits x 4096 x N_ISLANDS
                           = 34 Kbytes x N_ISLANDS
```

Domain Controller

- Information Table

```
[ SWITCH_ID(64), PORT(16), ISLAND_ID(20), VPLS_ID(20) ]
```

Each entry of the Information table has a total length of 110 bits. The Information table's size varies depending on N_ISLANDS and N_VLAN because each Island can participate at MAX_VLAN VPNs at most. In the worst case scenario all the Islands would use MAX_VLAN.

```
Information Table Total Size = 110 bits x MAX_VLAN x N_ISLANDS
                             = 110 bits x 4096 x N_ISLANDS
                             = 55 Kbytes x N_ISLANDS
```

Scalability on Number of flows

We examine the scalability of the number of flows used for unicast and broadcast traffic.

Island flows

Inside the flow table of an IE, the number of unicast flows varies only depending on N_HOSTS as we match packets based on destination MAC address in order to aggregate traffic.

```
Total Unicast Flows = N_HOSTS flows
```

Let N_HOST_PORTS be the number of ports attached to local hosts, N_DOMAIN_PORTS be the number of ports attached to DE devices and MAX_PORTS be the total number of used ports.
For broadcast traffic, the number of flows varies depending on N_VLAN, N_HOST_PORTS and N_VLAN, N_DOMAIN_PORTS combinations. (N_VLAN, N_HOST_PORTS) combinations are for broadcast traffic generated inside the Island while the (N_VLAN, N_DOMAIN_PORTS) combinations are for broadcast traffic arriving at the Island. The latter is greatly reduced as in normal scenarios the Island will be attached to only one Domain.

```
Total Broadcast Flows = (N_VLAN x N_HOST_PORTS) +
                                  + (N_VLAN x N_DOMAIN_PORTS)
                      = N_VLAN x (N_HOST_PORTS +
                                          + N_DOMAIN_PORTS)
                       = N_VLAN x MAX_PORTS flows
```

In total, also the worst case scenario, the maximum number of flows required in an IE device is:

```
Total Number of Flows = Total Unicast Flows +
                                    + Total Broadcast Flows
                      = N_HOSTS + MAX_VLAN x MAX_PORTS
                      = N_HOSTS + 4096 x MAX_PORTS flows
```

Domain flows

Inside the flowtable of a Domain switch the number of unicast flows varies depending on N_ISLANDS as a result of the unique ISLAND_ID and the flows that match only the ISLAND_ID.

```
Total Unicast Flows = N_ISLANDS flows
```

Let N_VPLS be the number of VPLS IDs present in the global network and MAX_VPLS be the maximum number of VPLS IDs(2^{20}) present in the global network. For broadcast traffic, the number of flows varies depending on N_VPLS and MAX_PORTS.

```
Total Broadcast Flows = N_VPLS x MAX_PORTS flows
```

In total, also the worst case scenario, the maximum number of flows required in a DE device is:

```
Total Number of Flows = Total Unicast Flows +
                       + Total Broadcast Flows
                       = N_ISLANDS + MAX_VPLS x MAX_PORTS flows
```

8.3 APPENDIX C - Web portal based modifications

The following changes are needed per design when implementing a web portal in the architecture, as discussed in section 4.5.

Core Labeling

The following functions are no longer needed in the Island Controller:

- CTRL-to-CTRL: ForceMAC (*domain_ctrl, mac, vlan*)
  Send command to Domain Controller to learn a MAC address

- CTRL-to-CTRL: LocateHost (*domain_ctrl, packet*)
  Send command to Domain Controller to locate an unknown host

The following functions are no longer needed in the Domain Controller:

- CTRL-to-CTRL: ForceMACRemote ([*DomainCtrls*], *mac, vpls*)
  Send command to other Domain Controllers to learn a MAC address

- CTRL-to-CTRL: LocateHostLocal ([*IslandCtrls*], *packet, vpls*)
  Send command to Island Controllers to locate an unknown host

- CTRL-to-CTRL: LocateHostRemote ([*DomainCtrls*], *packet, vpls*)
  Send command to other Domain Controllers to locate an unknown host

Island Labeling

The following functions are no longer needed in the Island Controller:

- CTRL-to-CTRL: ForceMAC (*domain_ctrl, mac, vlan, vpls, island*)

Send command to Domain Controller to learn a MAC address

- CTRL-to-CTRL: LocateHost (*domain_ctrl, packet, vpls*)
  Send command to Domain Controller to locate an unknown host

The following functions are no longer needed in the Domain Controller:

- CTRL-to-CTRL: ForceMACRemote ([*DomainCtrls*], *mac, vpls, vlan, island*)
  Send command to other Domain Controllers to learn a MAC address

- CTRL-to-CTRL: LocateHostLocal ([*IslandCtrls*], *packet, vpls*)
  Send command to Island Controllers to locate an unknown host

- CTRL-to-CTRL: LocateHostRemote ([*DomainCtrls*], *packet, vpls*)
  Send command to other Domain Controllers to locate an unknown host

Based on these modifications, our architecture becomes simpler due to the absence of MAC learning/ageing mechanisms. As a result Controller to Controller communication functions are also reduced.

8.4 References

[1] Open Networking Foundation: OpenFlow 1.3 Switch Specification
URL:*https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf*

[2] CoCo: On Demand Community Connection Service for eScience Collaboration
URL:*http://www.geant.net/opencall/Software_Defined_Networking/Pages/Home.aspx#CoCo*

[3] GALL, Alexanger. Leveraging SDN to provide Layer-2 VPNs as a Service on a MPLS-free core, November 2013 in *TNC 2014*.

[4] APPELMAN, Michiel. Architecture of dynamic VPNs in OpenFlow, July 2013.
URL: *http://work.delaat.net/rp/2012-2013/p14/report.pdf*

[5] PODLESKI, Łukasz; JACOB, Eduardo; AZNAR-BARANDA, José; JEANNIN, Xavier; BAUMANN, Kurt; ARGYROPOULOS, Christos; Multi-domain Software Defined Network: exploring possibilities in *TNC 2014*.

[6] RFC 4762: Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling. URL: *http://tools.ietf.org/html/rfc4762*

[7] SIJM, Niels; WESSEL, Marco. Effects of IPv4 and IPv6 address resolution on AMS-IX ant the ARP Sponge, July 2009.
URL: *http://work.delaat.net/rp/2008-2009/p23/report.pdf*

[8] IETF: Network Virtualization Overlays (NVO3)
URL: *https://datatracker.ietf.org/wg/nvo3/charter/*

[9] IETF: VXLAN – A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks (Internet Draft), June 2014.
URL: *https://datatracker.ietf.org/doc/draft-mahalingam-dutt-dcops-vxlan/*

[10] IEEE: Link Layer Discovery Protocol
URL: *http://www.ieee802.org/1/files/public/docs2002/lldp-protocol-00.pdf*

[11] APPELMAN, Michiel; DE BOER Maikel. Performance Analysis of OpenFlow Hardware, February 2012.
URL: *http://work.delaat.net/rp/2011-2012/p18/report.pdf*

[12] SUENNEN, Dany. Performance Evaluation of OpenFlow Switches, February 2011.
URL: *ftp://ftp.tik.ee.ethz.ch/pub/students/2011-FS/SA-2011-07.pdf*