UNIVERSITY OF AMSTERDAM

MSC SYSTEMS & NETWORK ENGINEERING

Architecture of dynamic VPNs in OpenFlow

By:
Michiel APPELMAN
michiel.appelman@os3.nl

Supervisor:
Rudolf STRIJKERS
rudolf.strijkers@tno.nl

July 7, 2013

Summary

Contents

List of Figures

List of Tables

1 Introduction

Network operators today use Network Management Systems (NMSs) to get control over their devices and services that they deploy. These systems have been customized to their needs and in general perform their functionalities adequately. However, operators run into obstacles when trying to expand their business portfolio by adding new services. Which will potentially require *a*) new Application Programming Interface (API) calls to be implemented towards their NMS, *b*) their NMS to be able to cope with potentially new protocols, and *c*) added expertise by engineers to define the possible feature interactions and restrictions of these protocols [1]. This limits the flexibility of the operators network when deploying new or adjusting existing services.

To manage resources efficiently in a carrier network operators have been using Virtual Private Networks (VPNs) between customers. By differentiating traffic between VPNs they can control their traffic flow at a granular level. However, the set of interactions between different protocols and management interfaces to them are intricate. The provisioning of VPNs requires expertise and a significant amount of changes to the protocol stack used to provide the service.

Until recently operators were not concerned by the inflexibility in their services as their networks were in fact primarily static. However, the demand for application specific networks (e.g. video, voice or payment networks) is growing. Therefore operators are looking for a more flexible approach in the form of Dynamic VPNs (DVPNs). DVPNs are private networks over which end-users can communicate, deployed by their common Service Provider (SP). They differ from normal VPNs in the sense that they are relatively short-lived. Using DVPNs, SPs can react more rapidly to customer requests to configure, adjust or tear down their VPNs. However, due to the aforementioned complexity DVPNs services have not been implemented on a large scale.

A potential candidate to solve the complexity of implementing DVPNs is OpenFlow [2] and Software Defined Networking (SDN). SDN is an architecture that allows for the programmability of the control plane of networking devices. The architecture is not standardized but a generalized structure has been given in the OpenDaylight project [3]. OpenFlow is a lower level and increasingly supported API protocol towards networking devices. Implementing the SDN architecture promises *a*) CAPEX savings due to hardware being more generic and flexible, *b*) OPEX savings because of the integration of NMSs and the control interface of the devices, thereby increasing automation, and *c*) increased network agility by using the open interfaces to program network devices directly [4].

The momentum that SDN is getting might be explained by a general need for change in the networking industry. Operators primarily want to get more control over their networks, something which using the current stack of protocols is relatively complicated to get. The original OSI reference model [5] touches on the "Management Aspects" of each layer in the model, a way for management entities in the highest layer to control the behavior of lower layers. Unfortunately in the swift evolution of Transmission Control Protocol (TCP)/Internet Protocol (IP), these management interfaces are often limited or absent all together.

1.1 Research Question

In the case of DVPNs it is unclear if and how a real-world OpenFlow and SDN implementation will actually provide any simplicity, additional flexibility or cost savings when compared to contemporary technologies [1]. In this research we will therefor answer the question "Can DVPNs be implemented using contemporary technologies?". And additionally provide insight in how OpenFlow works and if "DVPNs can be implemented using OpenFlow?". When the two implementation have been described, we can then look at what the key differences between the two are.

1.2 Scope

The focus will primarily be on deploying Provider-provisioned VPNs (PPVPNs) at Layer 2 of the OSI-model between end-users. We haven chosen to do so because these Ethernet VPNs are characterized by their transparency to the end-user, who will be placed in a single broadcast domain with its peers and can thus communicate directly without configuring any sort of routing. Furthermore, the provider will also be mostly agnostic to the use of the customer, who can choose to use IPv4 or IPv6.

Previous research in [6] has proposed an implementation for programmable networks to deploy on-demand VPNs but it predates the OpenFlow specification, and also omits a comparison with how this would look using contemporary technologies.

1.3 Approach

In the Section 2 we will define the conceptual design of DVPNs. This will result in a list of required features for the technologies to provide such a service. Section 3 will list the technologies available and will additionally determine their usability for implementing DVPNs when taking into account the requirements set forth in Section 2. In Section 4 we will distill the advantages and limitations of the different implementations and substantiate how the compare to each other. Finally, Section 5 summarizes the results and provides a discussion and future work on this subject.

2 Definition of a DVPN

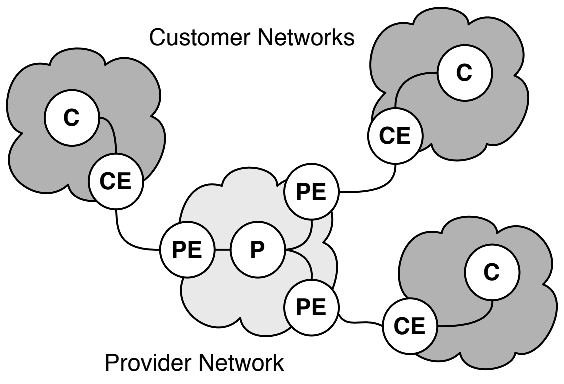
To define the DVPN service, we first take a look at the concepts of non-dynamic, or static VPNs. This section will start by defining what a standard VPN service looks like, how it's carried over the provider network and the information needed to implement it. Next we will look at how these normal VPN services can be evolved into Dynamic VPNs.

2.1 VPNs

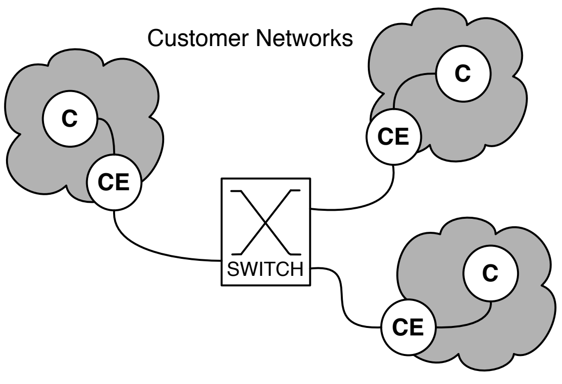
2.1.1 Service

VPNs can be classified depending on the OSI layer which it virtualizes, the protocol that is being used and the visibility to the customer. In an IPSec VPN for example, the customer needs to setup his Customer Edge devices (CEs) at each site to actually establish the Layer 3 IP VPN. As we have already established in Section 1.2, we limit the use-case to an multi-point Ethernet Layer 2 VPN which is provisioned by the provider (PPVPN) and thus requires no action on the CE. Primarily because these Ethernet VPNs are characterized by their transparency to the end-user, requiring no routing information from them, and the fact that the client can choose to use IPv4 or IPv6. Throughout this paper the definition of PPVPN related terms will be used as described in RFC 4026 [7] and an overview is given in Figure 1a.

What a Layer 2 PPVPN provides to the CE is a transparent connection to one or more other CEs using a single Ethernet broadcast domain. Another term to describe such a VPN service is a Virtual Private LAN Service (VPLS). It enables the interconnect of several LAN segments over a seemingly invisible carrier network. To do so, the Provider Edge device (PE) needs to keep the Customer MACs (C-MACs) intact and also support the forwarding of broadcast and multicast traffic. All PEs (and of course Provider devices (Ps)) will not be visible to the CE, who will regard the other CEs as part of the VPLS as direct neighbors on the network as illustrated in Figure 1b.



(a) Terminology used to describe PPVPN devices.



(b) Appearance of VPLS from customer point-of-view.

Figure 1: Visualization of used terminology.

To summarize, from a service level perspective a VPN needs to provide a network to the customer which:

1. provides a Layer 2 broadcast domain,
2. does not require configuration on the CE,
3. is transparent to CEs.

2.1.2 Transport

Transporting a Layer 2 frame between two CEs starts at the PE. The ingress PE learns the Source Address (SA) of the frame behind the port connected to the CE, then it needs to forward the frame to the PE where the Destination Address (DA) is present. It will need to do so while separating the traffic from other DVPNs, it has to make the traffic unique and identifiable from the rest of the VPNs transported over the network. This is done by giving the frame some sort of 'color' or 'tag' specific to the customer DVPN. Additionally PE1 presumes the that P devices are not aware of the DVPN and do not know the C-MAC addresses. This is because the network will have to scale to thousands of DVPNs and possibly millions of C-MACs divided over those DVPNs. To provide this so called MAC Scalability, only PEs should learn the C-MACs and the backbone will forward frames based on PE endpoints. Forwarding through the provider network happens based on a prefixed header indicating the endpoint PE or the specific path towards that PE. Figure 2 gives an example of transporting VPN frames over the provider backbone consisting of the following steps which we will discuss further in this section:

1. Ethernet frame comes in from CE1.
2. PE1 'colors' the frame according to the DVPN it is a member of, to separate traffic between VPNs.
3. PE1 looks up CE2 MAC in DVPN MAC table but has no mapping, so decides to flood the frame to PEs with DVPN member ports.
4. PE1 adds a label to the frame needed for the Ps to forward the frame to PE2 over a predetermined path.
5. The P device forwards the frame according to the label.
6. PE2 receives frame, removes header.
7. PE2 looks up CE2 MAC in DVPN MAC table, finds it behind port 1.
8. PE2 removes VPN 'color' and forwards frame to CE2.
9. PE2 sees CE1 MAC received from PE1, installs PE1 as destination for CE1 MAC.

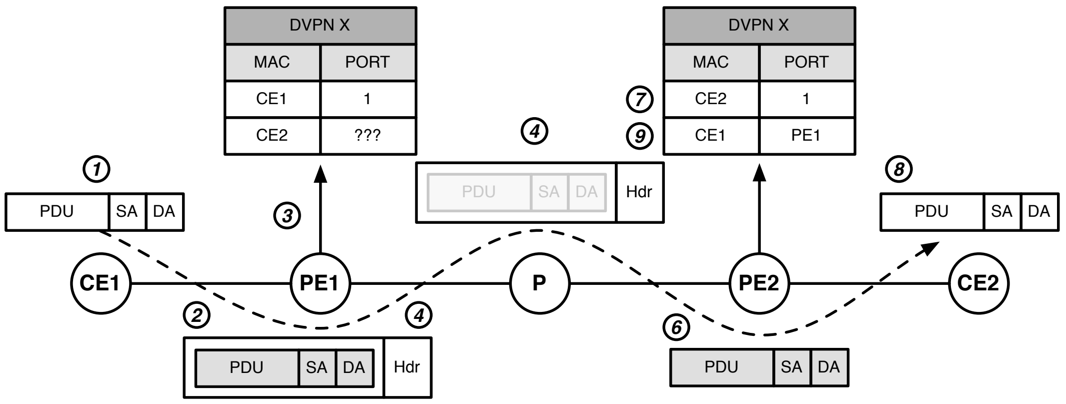


Figure 2: DVPN traffic as it travels through the provider network.

Forwarding from ingress PE to egress PE happens over a path of several Ps. Every PE connected to a CE member of a particular DVPN, should have one or more paths available to each and every other PE with members of that DVPN. The determination of the routes of these paths takes place through a form of topology discovery. This mechanism should dynamically find all available PEs and Ps with all the connections between them and allow for the creation of paths which are not susceptible to infinite loops.

The links comprising the paths have a certain capacity which will need to be used as efficiently as possible: the links comprising a path will need to have enough resources available, but other links need not be left vacant. Also, if the required bandwidth for a DVPN exceeds the maximum capacity of one or more of the links in a single path, a second path should be installed to share the load towards the egress PE. In short, the traffic going over the provide backbone should be tightly managed to prevent congestion.

Continuing with the processing of the ingress customer frame, when it arrives at the ingress PE with a DA unknown to the PE, the frame will be flooded to all PEs participating in the VPN. Upon arrival there, the egress PE stores the mapping of the frames SA to the ingress PE and if it knows the DA will forward out the appropriate port. Figure 2 shows this at points *3* and *8*, *10* displaying the contents of the Media Access Control (MAC) address table of both PEs. Because the DVPN is a virtual broadcast domain, all Broadcast, Unknown unicast and Multicast (BUM) traffic will need to be flooded to the participating PEs. To limit the amount of BUM traffic in a single DVPN rate limits or filters will need to be in place to prevent the DVPN from being flooded with it.

With multiple DVPNs present on the network it can happen that one DVPN affects the available bandwidth of others. Therefore rate limits will need to be in place for the overall traffic coming in to the CE-connected ports.

By assigning a minimum and maximum bandwidth rate to each DVPN instance, it is possible to preprovision paths over the network according to the required bandwidth. By also monitoring the utilization of individual links, DVPN paths can be moved away from over-provisioned links while they are in use. However, the impact on traffic when performing such a switch must be minimized and should ideally last no longer than 50 ms.

To monitor and troubleshoot large carrier networks Operations, Administration and Management (OAM) functionalities need to be supported by the network. Monitoring end-to-end activity needs to available through automatic continuity check messages, but also by supporting 'traceroutes' through the network manually. This enables the network react proactively to network failures by using a similar method as presented above when switching DVPNs to a different path, also known as 'fast failover.'

To sum up the requirements discussed in this section, the network needs to provide the following functionalities:

1. identify traffic from separate DVPNs by tagging,
2. scalable up to thousands of DVPNs and C-MACs,
3. topology discovery,
4. provision paths over the network,
5. efficient use of, and control over all network resources (Traffic Engineering (TE)),
6. share the load of traffic over multiple paths,
7. rate limiting or filtering of BUM traffic,
8. rate limiting of total DVPN traffic per port,
9. fast failover times (<50ms) to provide continuity to critical applications,
10. provide Operations, Administration and Management features to monitor and troubleshoot the network.

2.1.3 Provisioning

A VPN service consists of multiple member ports, which are identified by their PE device and the port on that PE. The service is also defined by its minimum and maximum available bandwidth which can be used to determine the paths that the VPN will get assigned. When member ports reside on different PEs, a path will need to be created through the network. The route of the path will depend on *a*) the liveness and administrative availability of the links, *b*) the administrative costs of the links, and *c*) the resources available on the links towards the PE. The exact algorithm used to choose the paths lies outside of the scope of this document. Paths are defined by the physical ports that they traverse through the network, with the PEs as the first or last in the list.

More paths may be added over different routes and paths may be adjusted during the lifetime of the VPN. This may for example be necessary when a certain link in the path fails, or when it nears its peak capacity and has to be rerouted. Individual port utilization will be monitored and when a certain link shows high utilization, the corresponding paths and VPNs using those paths can be looked up using the information base. Also other monitoring and troubleshooting processes will profit from this information.

After the paths between the PEs with VPN members have been setup the traffic can start flowing. However, as has been mentioned before, the rate limiting feature will need to be applied to the ingress ports to prevent the VPN from using up all the networks resources.

To provision VPNs in the network, the operator should be able to:

1. setup the base network required for VPN routing,
2. determine routes that can be used for the paths,
3. monitor links and reroute paths on failure or peak capacity,
4. set the rate limits on ingress PE ports.

2.2 DVPNs

The VPN service that can be implemented using the aforementioned features is typical for current operator networks. The configurations are mostly static and changes in the network will require (manual) effort to provision. To evolve from these inflexible VPN setups to Dynamic VPNs the provisioning systems in use will need to support some additional features.

First, the implementation of DVPNs needs to be automated. This means that instead of the manual configuration by an engineer, the NMS will take care of configuring the network with the correct CE endpoint ports on all PEs, making sure that the paths are installed and verifying connectivity. This also includes calculation of the optimal paths and setting the correct rate limits, similar to the requirement given in Section 2.1.3.

Second, the NMS should be able to change the configuration of DVPNs when requested. And when a DVPN is no longer in use, it will need to be automatically purged from the network to free up resources for other services. This requires the NMS to keep an overview of the provisioned DVPNs and can optionally set an idle time out for the service.

Finally, the changes in the network resources and DVPN requirements should be acted upon without manual intervention. When certain links are running low on available bandwidth or fail altogether, this will cause a recalculation of the provisioned paths. Also, when new hardware is added, it need to be configured to take part in the network, immediately making use of its added resources. Similarly, when the requirements of a DVPN change, the NMS will verify that the current paths meet these requirements, or otherwise find new routes to provision for the DVPNs.

To provide a DVPN service, the operator requires – next to the aforementioned VPN features – the following functionalities from the NMS:

1. Automated creation, modification and deletion of DVPNs,
2. Adapt to changes in the form of:
   (a) deminishing resources,
   (b) added resources, and
   (c) changed requirements for DVPNs.

3 Architecture

This section will first describe the technical details of Multi Protocol Label Switching (MPLS) with regards to the features required for implementing DVPNs. We will design an architecture to implement this service and point out the limitations. Next, we will describe the SDN architecture and the OpenFlow specification. Also looking at the features they provide to implement a DVPN service. In Appendix A.2 we have also taken a look at Asynchronous Transport Method (ATM) and Shortest Path Bridging (SPB), both technologies which can be used to implement some form of VPNs but are missing key requirements.

3.1 MPLS

MPLS is known for its scalability and extensibility. Over the past decade additions have been made to the original specification to overcome a plethora of issues within carrier networks. This initially started with trying to implement fast forwarding in legacy switches using labels (or tags) at the start of the frame [8]. When this issue became surmountable using new hardware, MPLS had already proven to be capable of transporting a wide arrange of protocols on the carrier backbone network, all the while also providing scalability, TE and Quality of Service (QoS) features to the operators.

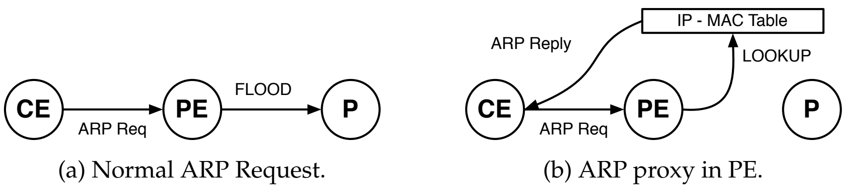
MPLS itself is more a way of forwarding frames through the network, without facilitating any topology discovery, route determination, resource management, etc. These functions are left to a stack of other protocols. Without IP reachability throughout the network these protocols cannot exchange traffic and so, as a prerequisite, MPLS relies on an Interior Gateway Protocol (IGP) like Open Shortest Path First (OSPF) to discovery the topology.

The distribution of labels has to be facilitated as well, which is done using Label Distribution Protocol (LDP) and/or Resource Reservation Protocol (RSVP). These protocols run between each device in the path between two PEs and exchange the labels that the will assign to a certain path, thereby setting up a Label-switched Path (LSP). The labels assigned are always of 'local significance,' meaning that the P/PE device that needs to forward the labels, will announce its own chosen labels. The LDP protocol does this by distributing its labels from the egress PE up towards the ingress PE based on IGP costs. RSVP, on the other hand, signals its paths from the ingress PE towards the downstream PE based on constraints, potential explicit hops or as a last resort using the IGP next hop. Label distribution is still determined from egress to ingress, but the actual path is determined at the head-end. To determine the best path to take, RSVP uses the Constrained Shortest Path First (CSPF) algorithm which can take into account link characteristics like bandwidth or Fast Reroute (FRR) support. This allows RSVP LSPs to take more well informed paths through the network and together with support for defining explicit paths, allows for granular TE features which LDP lacks. Both LDP and RSVP also allow for the use of multiple paths over the network to share traffic load towards a PE.

The aforementioned FRR feature is unique to RSVP and provides the network with fast failure recovery. It does so by preprovisioning a so-called backup LSP next to the primary LSP. When a failure is detected on the primary LSP, traffic is immediately shifted towards the standby path, yielding a sub-50ms failover. Obviously, this value also depends on the time it takes for the failure to be detected. Therefore it is important to have some sort of detection mechanism in place. One that is commonly used and integrates with OSPF is Bidirectional Forward Detection (BFD). This protocol sets up sessions between devices and triggers an alarm when the session does not behave as expected. At which point FRR swaps the traffic to the preprovisioned backup path. To differentiate traffic coming from a normal, 'protected' path and traffic taking a 'detour' path, FRR adds another MPLS tag to the MPLS label stack.

VPNs are also provided by additional protocols. Layer 3 VPNs make use of Border Gateway Protocol (BGP) to distribute client prefixes to the edges of the carrier network. The core is only concerned with the forwarding of labels and has now knowledge of these IP prefixes. Layer 2 VPNs make use of VPLS, a service which encapsulates the entire Ethernet frame and pushes a label to it to map it to a certain separated network. Again, the core is only concerned with the labels and only the edges need to know the clients MAC addresses. When setting up a VPLS instance (a VPNs), LDP sessions are setup between all PEs part of the same VPLS instance. Consecutively, the PEs will exchange their chosen labels for that instance between each other.

The C-MACs in a VPLS instance are normally learned through the data plane. That is, when a frame comes in from a CE, the PE learns the SA behind the corresponding port. If it doesn't know the DA, it will flood the frame to other PEs with member ports in that instance. These PEs in turn learn the SA as well behind the ingress PE. This is also illustrated in Figure 2. When a large number of C-MACs are present within a VPLS instance this can cause a lot of broadcast traffic, specifically Address Resolution Protocol (ARP) traffic. To solve this, the Ethernet VPN (E-VPN) standard has been proposed [9]. This technique provides MAC learning in the control plane by exchanging learned C-MACs between PEs using Multi-Protocol BGP. Additionally it learns the IP addresses associated with the C-MACs and distributes those to other PEs. The PEs are thereby able to act as an ARP proxy, as illustrated in Figure 3b.



(a) Normal ARP Request. (b) ARP proxy in PE.

Figure 3: Processing of ARP requests at the PE.

The different protocols all depend on each other, as illustrated in Figure 4. Each PE device runs this stack, while P devices run a subset which is shaded.

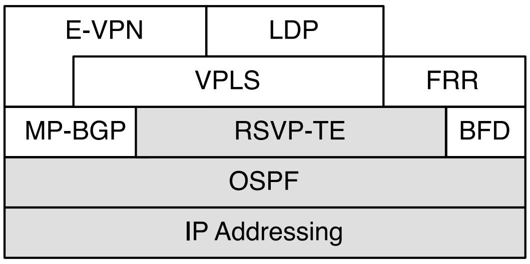


Figure 4: Dependency stack of MPLS-related technologies.

To configure a DVPN using MPLS first the participating PEs need to be configured with the new VPLS instance to which the member CE ports will be added. Next, constraints are defined by the NMS, which can be in the form of an explicit route to make a static route or by defining loose constraints based on bandwidth limits which can be used the CSPF algorithm. Using these constraints, paths are installed at each PE towards every other participating PE. These paths are then added to the VPLS instance, allowing LDP sessions to be setup between the PEs. Next, for FRR, backup LSPs need to defined similarly to the primary LSP but over a different path, which can again be done using constraints to exclude the other links. Utilization of the links in the network has to be monitored as well and when a path has a link which is nearing capacity, new LSPs have to be provisioned and some VPLS paths move to those LSPs. And finally the ingress traffic on the CE ports need to be rate limited. This procedure is not standardized and is dependent on support of the hardware.

The procedure above implies that the backbone network has been setup with the following protocols and features already enabled: IP addressing, OSPF routing, MPLS forwarding, RSVP with FRR and BFD. After initial setup of the backbone network the NMS is only concerned with the PEs, as can also be seen in Figure 5. However, the NMS needs to be aware of the PEs it needs to consult, the intermediate Ps if it wants to use TE, the network resources in the path and of course the actual member ports.

Implementing VPNs using MPLS has been shown to be possible using the protocols that have been engineered to provide all the required features. The technologies are built upon each other instead of being evolved upon. This causes their dependencies to grow more complex over time which in turn makes it difficult for the NMS to deal with all the different protocols. Add to that the lack of a common abstracted interface to the network devices, and it becomes apparent that developing and maintaining an efficient NMS is also getting more difficult.

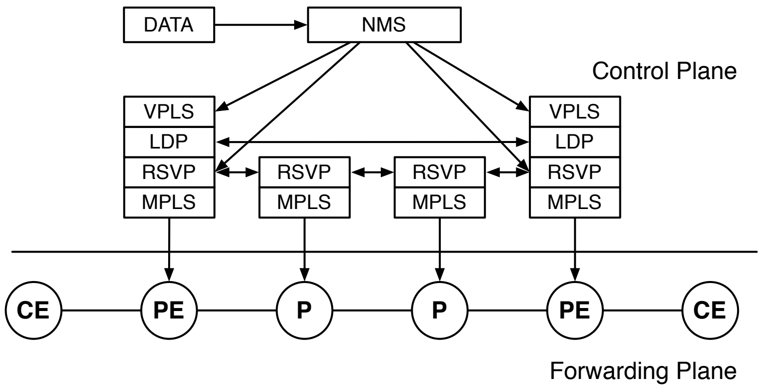


Figure 5: Provisioning a DVPN using MPLS.

3.2 OpenFlow

Software Defined Networking is the general principle of designing flexible networks using open interfaces towards the hardware. OpenFlow is a subcomponent of this new principle which provides a protocol between the forwarding plane of the networking devices and a centralized controller. A general overview of the SDN architecture and OpenFlow is given in Figure 6. OpenFlow provides the controller with an API that can be used to install flow entries directly in the forwarding plane of the devices. Flow entries consist of six fields:

Match	This field contains a list of frame/packet characteristics that will need to be present to match to this entry, e.g. DA, IP source address or MPLS tags.
Priority	The precedence of this flow entry over other flow entries to which a certain frame matches.
Counters	Frames matching to this entry are counted for monitoring purposes.
Instructions	When a frame is matched using the match field, it is processed according to a list of instructions, which may include forwarding out of a port, rewriting headers and/or applying meters.
Timeouts	The time that a flow entry can be live until it is discarded.
Cookie	Value assigned by controller to identify the flow (not used to forward frames).

The frame fields that can be matched upon have changed over the lifetime of the OpenFlow specification. For example, version 1.0 could only match and/or act upon tagged traffic using a single outer-VLAN tag. Version 1.1 added matches and actions for Q-in-Q tags and MPLS labels, and version 1.3 could also match Provider Backbone Bridging (PBB) tags. See Table 1 for a comparison of key features in different versions of OpenFlow spec.

Note that rate limiting on a per port basis has been available in OpenFlow since version 1.0. Additionally, version 1.3 added support for per flow rate limiting using so called 'meters' which can be assigned to specific flows. By doing so it becomes possible to also rate limit flows on certain aggregation ports rather than just at the ingress port of the CE.

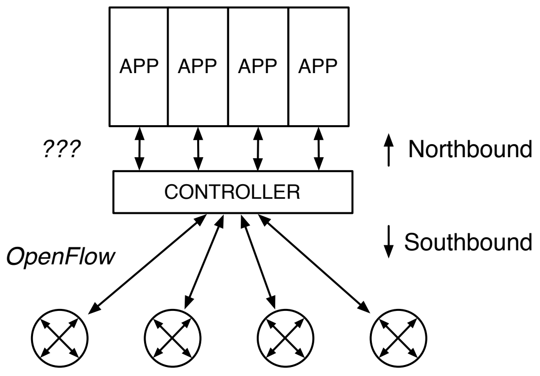


Figure 6: Architecture of SDN and OpenFlow.

	1.0	1.1	1.2	1.3
VLAN Tags	✓	✓	✓	✓
Q-in-Q Tags		✓	✓	✓
MPLS Tags		✓	✓	✓
PBB Tags				✓
Groups		✓	✓	✓
Rate limiting	✓	✓	✓	✓

Table 1: Comparison of OpenFlow versions regarding key features for DVPNs.

The installation of flow entries is done by the controller, governed by the applications running on it. Applications can be written to provide functions like topology discovery, routing, etc. Moreover, without these installed applications, the network will be unable to forward any traffic. The interface between the applications and the controller is also being referred to as the 'northbound interface'. This interface, in contrast to the southbound OpenFlow interface, has not been specified and varies between different controller implementations, limiting the portability of the network applications.

Unlike contemporary technologies that require inter-device communication before any paths can be set up, the forwarding tables of OpenFlow devices are empty. The only prerequisite is that the devices all have a management connection to the controller from which they can receive their forwarding information.

The controller is the combination of hardware and software components that are preferably setup redundantly and share a complete view of the network, which they share with the applications running on it. To provide the network and its control with redundancy in case of failures, the 'controller' component can refer to a logical controller entity, not specifically a single physical server. However, the actual design and implementation of such a system is beyond the scope of this paper.

Current OpenFlow versions are not yet supported by all controllers. We are inspecting version 1.3.1 of the specification which supports the features described. However, at the time of writing there are only two controllers supporting the 1.3 version:

- Ryu by NTT [10], and
- NOX extensions by CPqD research center from Brazil [11].

The controller developed under the OpenDaylight project lacks support for version 1.3.

Implementing DVPNs using OpenFlow relies mostly on the applications running on the controller. When they are written and configured as desired, provisioning a DVPN would only require the input of the data. After which the applications and controller install flow entries into all the network devices in the DVPN path, as can be seen in Figure 7.

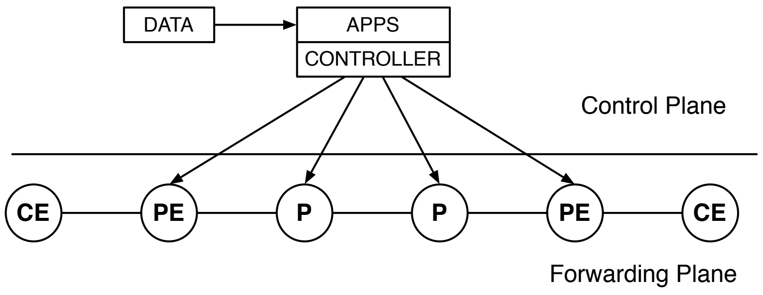


Figure 7: Provisioning a DVPN using OpenFlow.

3.2.1 DVPN Provisioning

To implement DVPNs, a combination of applications need to be installed on the controller that contemporary technologies solve using distributed protocols:

Topology Discovery Network devices do not require IP connectivity between each other to exchange route information. Instead, they rely on their connection to the controller to provide them with information. A topology discovery applications will instruct network devices to send out unique discovery messages which are then received by other devices, which forward the message back up to the controller. By keeping information on which packet goes in and comes out where, the application can get an overview of the network. By centralizing the topology discovery process, network operators can benefit from more efficient synchronization and faster convergence.

DVPN Provisioning The DVPN input will provide the application with at least the CE port and preferably the corresponding MAC and/or IP address. It can then instruct the path provisioning component to setup a path between the PEs participating in the DVPN.

Path Provisioning Using the discovered topology, paths are setup over those routes between PEs with member CE ports in a common DVPN. This means that the flow entries are installed proactively in the Ps and PEs when a DVPN is setup. Instead of waiting for one of them to send a message to the controller asking what to do with an unknown incoming frame, this proactive approach allows for better scalability (less requests to the controller) and faster initial forwarding (no buffering while consulting controller).

OAM The controller and applications provide a complete overview of the network but troubleshooting and monitoring still has to be done at the network level as well. Different approaches can be taken to do so, one of which could be sending out periodic keep-alive messages in the same path from the controller down to the ingress PE that the egress PE should forward back up to the controller. Another example is implementing an already defined OAM protocol in OpenFlow, as has been done in [12] which implemented Ethernet 802.1ag OAM.

Traffic Engineering The path setup procedure uses data from the DVPN input and the discovered network resources to provide the most optimal path between two PEs. Constraints for the paths taken by each DVPN can be configured by the operator and influence the route selection directly over the whole platform. Also, using input from the OAM monitoring applications paths may be preferred or deprecated based on their performance.

C-MAC Filtering This application is concerned with keeping the provided or dynamically learned MAC addresses up-to-date. Additionally, a flow can be installed matching on the ARP EtherType that sends the ARP request towards the controller. If the application also keeps track of the IP addresses of the CEs it can then act as an ARP proxy and reply to the requesting CE with the correct IP address.

Fast failover and Equal Cost Multi Path (ECMP) can be accomplished using the aforementioned port 'groups', which are available since version 1.1. Groups can be defined as a destination in a flow entry and contains a list of ports. The type of group defines the action of the group: '**all**' sends the frame out the frame out of all ports in the group (broadcast/multicasting); '**select**' outputs it to one of the ports (providing ECMP); '**indirect**' is a single port group which can be used by multiple flow entries (aggregation); and '**fast failover**' which choses the first *live* port out of which it will forward the frame. To support 'fast failover' a *liveness monitoring* technique needs to be implemented supported by the switch. However, apart from monitoring the state of the physical link, there has been no technique defined to monitor the liveness of inter-device links, such as BFD. The same holds true for full path liveness monitoring which currently needs to be done using the controller, yielding a higher failure recovery time.

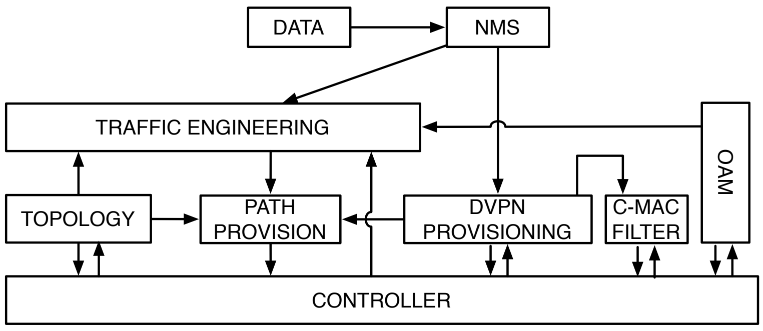


Figure 8: Interactions between applications to implement DVPNs.

The interaction between the different SDN applications has been illustrated in Figure 8. Due to the abstractions introduced by the intermediate applications the NMS doesn't need to be aware of the network below. The applications will map the port member info from the NMS to the correct devices, and configure the DVPN. Because of the abstractions and minimal initial configuration, adding extra hardware can also be taken care of without consulting the NMS: the applications will add it to the network and reroute traffic over the new device.

4 Comparison

Table 2 displays the key differences between the two technologies that we have discussed in 3. In this section we will take a close look at the differences between the MPLS and OpenFlow implementations with regards to the listed requirements.

	MPLS	OpenFlow / SDN
Tagging of VPN Traffic	VPLS (MPLS)	PBB / MPLS
MAC Scalability	yes	yes
Topology Discovery	OSPF	centralized
Path Provisioning	RSVP / LDP	centralized
Traffic Engineering	RSVP	centralized
ECMP	yes	yes, using Groups
BUM limiting	dependent on HW	yes, using Metering
Exchange C-MACs	E-VPN (draft)	centralized
Ingress Rate Limiting	dependent on HW	yes, using Queues or Metering
Fast Failover	FRR	yes, using Groups (but limited)
OAM	LSP Ping / BFD	centralized
Forwarding Decision	MPLS labels	flow entry
BUM traffic handling	flood	sent to controller

Table 2: Feature requirements available in discussed technologies.

4.1 Service

From a customer point-of-view it should be of no concern how the DVPN service is implemented in the provider network. Moreover, the PEs and the rest of provider network should be completely transparent. As such, the two technologies do not show any difference in their implementation. Both technologies are able to *1*) provide a Layer 2 broadcast domain, *2*) connect CEs without any required VPN configuration on them, and *3*) be completely transparent to the CEs.

4.2 Transport

Both implementations use MPLS labels to transport frames through the network. OpenFlow benefits from the MPLS technology by using these labels allowing for a generic and extensible way to define paths. One alternative to tag traffic would be PBB tags but these are less extensible because of the clear purpose of each tag, whereas MPLS labels can be added arbitrarily.

By using labels to identify and route traffic over paths instead of a hop-by-hop based routing protocol that uses an egress PE identifier, both technologies allow for granular TE features. Using OpenFlow operators may also define very specific traffic flows to provide TE, called 'microflows'. While this will give them more precise control over their traffic, it will fill up the flow tables of Ps fairly quickly in a network with thousands of customers. These flow tables are implemented using Ternary Content-Addressable Memory (TCAM) which is finite and is also expensive to produce [13]. Using these labels to minimize the amount of flows in the core is thus advised.

In contrast with the MPLS architecture, using OpenFlow the P devices also need to be updated using the forwarding information. After all, there are no label distribution protocols running between the devices. The labels do not need to be locally significant to the networking devices. In fact, by using the same unique label per path, one can imagine that troubleshooting will be more transparent as well.

A prerequisite for providing any kind of service over a network is the knowledge of the network topology. Again the distinction between decentralized and centralized is easily made. Arguments can be made about the faster convergence of large networks using a centralized controller, however these claims are largely dependent on the implementation. Transporting frames using either one of the two technologies will not change depending on the implementation chosen.

OpenFlow has been able to provide ECMP since version 1.1 using Groups with the **select** type. It basically means that a flow can point to this group and it will choose one of the output ports, based on a hashing algorithm. And although the terminology is different from Link Aggregation Groups, the procedure is indeed the same. Moreover, due to the lack of a definition for the hashing algorithm, both implementations depend on the hashing algorithm implemented by the vendor to provide efficient load sharing.

In a contemporary setup devices support fast failover by setting up BFD sessions between each other to monitor liveness of the path. This is done within the forwarding plane of the device and with very small timeouts so failures will be apparent within milliseconds. Currently OpenFlow devices lack the ability to install some sort of packet generator in the forwarding plane to perform the same functionality. SDN researchers have proposed to use a monitoring function closer to the data plane in [14] but until that has been implemented monitoring of paths will need to use the controller, causing higher recovery times. Monitoring of individual physical links is possible using the **fast failover** Group type though. This allows the network device to quickly reroute without needing to consult the controller.

4.3 Provisioning

Discovering the network topology of a network typically requires connectivity between the devices on Layer 2 or 3 (depending on the IGP) before any information can be exchanged. This means setting up a network to provide DVPNs will require some up-front configuration from the NMS as well. Using OpenFlow on the other hand, the only requirement is setting up a connection to the controller from all networking devices.

Traffic Engineering can benefit from centralization as well. In fact, operators already need to store information on application traffic flows and requirements in the NMS when using MPLS to route traffic in a certain way. When using RSVP with explicit routes, the NMS requires a complete view from the network to correctly define paths. Only when using loose constraints, the TE functionality is partly solved decentralized using CSPF. However, the NMS still needs to configure the constraints for each flow. An SDN setup provides the operator with a complete view from the network as seen by the controller which can be used together with input from VPN constraints to optimize the paths. The advantage lies in the fact that the the TE application on the controller can get the current topology directly from the discovery application. Whereas the MPLS setup would require the NMS to retrieve the topology from the network, or the topology should be predefined. Either way, this might lead to inconsistencies, depending on the implementation.

Research has already shown that OpenFlow can be used to implement the MPLS stack [15]. Over a two month period researchers also succeeded in provisioning VPNs and provide operators with a simulated environment with very granular TE [16].

4.4 DVPNs

Table 2 at the beginning of this section already showed most of the technical differences between the two technologies. It also shows the plethora of acronyms in use by the MPLS architecture that all need to be managed. Additionally, the current available systems lack a consistent management interface and thus need to be managed through a form of command-line or NetConf interface. These interfaces can differ between vendors and sometimes even between models from the same vendor. Maintaining the code for all these different protocols and management interfaces while also providing a dynamic and agile NMS interface for the operator to use will prove to be quite difficult.

The OpenFlow implementation can solve these problems by providing operators with abstractions of the network on which they can write applications to configure their network. However, this also means that the intelligence of the network devices has to be reimplemented completely in the controller of the SDN architecture. These applications will then provide the core functionalities of the network, also providing the configuration to the devices, without the NMS needing to be aware of the specific topology or hardware.

By using the SDN architecture and having a global view of the network, the controller can also adapt more efficiently to changes in network resources. The TE will recalculate paths as links go up or down and immediately push these paths to the network. Whereas with the MPLS network setup this would require polling the devices for changes, configuring the interfaces, calculating paths and finally installing them.

By centralizing the path calculation application, the network can also apply more complex algorithms to distribute the traffic. The *knapsack problem* for example, is concerned with finding the optimum combination of flows to go over a certain set of links. This problem is hard to solve and would be better of to be run on servers with the appropriate resources than on the small CPUs of the routers.

The initial needed configuration for devices to be added to the DVPN network is higher when using the MPLS implementation. To be added, they need settings for IP addresses, routing protocols, labels, etc. These processes all need information from the NMS, which needs to be filled with information by the operator. With OpenFlow the device can be added dynamically, only requiring a management interface and the applications on the controller can then configure it as they would deem necessary, without any information from the NMS or operator.

The development of SDN applications is still behind on the actual OpenFlow implementation. The architecture is missing a defined Northbound API which would allow for portability of the applications. This limits the flexibility of the applications that are written today, because there is no guarantee that they will interoperate with future controllers/applications.

5 Conclusion

The networking industry sees their networks growing and changing. This research has defined the DVPN service as a way to adapt to this growth by providing flexibility and manageability to these networks. This is accomplished by providing operators with interfaces towards their networking equipment allowing them to get more granular control.

While discussing the implementation of MPLS for DVPNs we found that the stack was overly engineered and due to its cumulative nature, getting more and more complex. The stack consists of multiple protocols that been designed and integrated on top of each other over the last years. The dependencies they have on each other have made changing or extending these protocols very difficult and thus the complexity of NMSs have increased as well. This has also been caused by the lack of a common interface towards the devices, meaning that portability of these systems is low. These limitations will make implementing DVPNs in a real-life scenario a complex undertaking.

We also used the SDN architecture to design an implementation of the DVPN service. The design requires the development of several applications to be used by the controller to provide this service. However, by doing so the operator is provided with a more manageable interface towards its network. Or at least to an abstraction thereof, presented by the controller and its applications.

Using these abstractions and the global network view operators can also develop their own control programs to provide new features in the network. We have shown an example implementation to provide MAC address exchange between PEs, a functionality that is still in draft for the MPLS architecture (E-VPN [9]). However, the SDN architecture still lacks a definition for the Northbound interface which limits the portability of the applications and thus their flexibility.

The OpenFlow protocol has been defined and we have used it in our implementation to provide the interface from the controller to the network devices. The implementation uses MPLS labels to separate traffic flows and provision paths over the network. Additionally combined with the controller and the applications it can accommodate most other features of the DVPN service. However, the simplicity of the protocol also has a down side in the form of preventing forwarding plane monitoring. This means that in to detect failures in the network path, the controller needs to be consulted yielding higher recovery times [14].

Following is a list of all advantages and disadvantages of the MPLS implementation (on the left and the right side, respectively):

- ✓ Known technology which is mature, well supported and understood throughout the networking industry.

- × Large protocol stack with intricate dependencies.
- × Lack of a consistent management interface towards the networking devices.
- × Deploying an NMS to configure this stack and also be portable to different networking devices will lead to a complex piece of software, equal in inflexibility to the MPLS stack itself.
- × The E-VPN standard providing C-MAC exchange and Unknown Unicast filtering is still in draft.

And a similar list for the SDN architecture using OpenFlow as the Southbound interface:

- ✓ Ability to learn from the MPLS implementation, while being able to improve upon the architecture [17] by adding operator control.
- ✓ Provide controller and applications with global network view, allowing for more granular control of network resources and paths.
- ✓ Use network abstractions to develop applications providing new features, e.g. MAC Exchange on PEs.
- ✓ Since version 1.1 OpenFlow allows rate limiting per Flow, giving more control over network resources at aggregation points.

- × Centralization limits the independent decision-making of network devices, which harms failure recovery procedures, e.g. forwarding plane monitoring.
- × The Northbound interface connecting the applications to the controller has not be defined, limiting portability.
- × By removing the intelligence from the networking devices, functionalities providing that intelligence have to be developed using the centralized applications.

5.1 Recommendations

Using OpenFlow and SDN to implement DVPN services or use it in any carrier network will require some further development on the interfaces to and from the controller. First, the OpenFlow southbound interface is still limited by a set of functionalities that lacks features that are required in the forwarding plane. However, the SDN architecture allows for the definition of abstractions for the network. This has been shown by the OpenDaylight architecture using the Service Abstraction Layer, as can be seen in Figure 9. Using these abstractions, the available interfaces towards the networking devices may be extended by new standards or vendor specific interfaces which do provide these functions. It should be noted however that the OpenDaylight specification is in its infancy but is a step in the right direction.

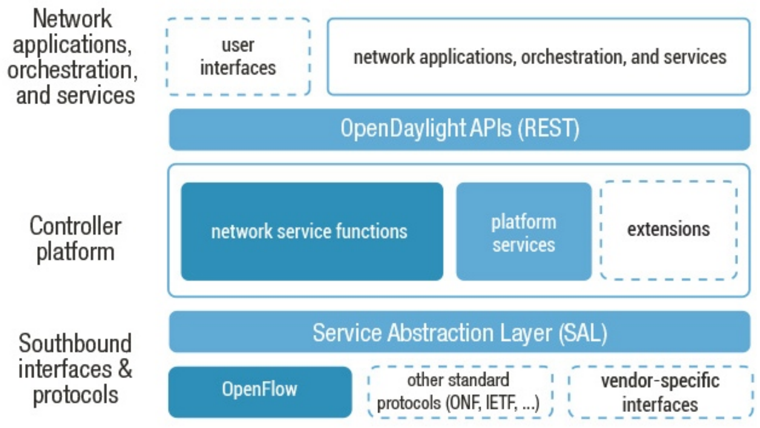

Network applications, orchestration, and services
user interfaces
network applications, orchestration, and services
OpenDaylight APIs (REST)
Controller platform
network service functions
platform services
extensions
Service Abstraction Layer (SAL)
Southbound interfaces & protocols
OpenFlow
other standard protocols (ONF, IETF, ...)
vendor-specific interfaces

Figure 9: The proposed OpenDaylight architecture.

Next, the northbound interface will need to be defined. Here the OpenDaylight proposes a standardized API to be implemented on the controller platform. By doing so, the applications that are developed for certain controller versions can be ported to other platforms. This clear separation allows for the evolution of the individual components, adding to the flexibility of the network.

Implementing DVPNs using OpenFlow also leads to other questions which need to be answered for specific implementations of controllers, hardware and applications.

5.2 Future Work

Other use cases:

- multi-domain
- mobility
- smart metering

A Related Work

A.1 ATM

For the sake of completeness, we will briefly look at ATM. ATM is a legacy protocol that has been used by operators to carry traffic over the internet backbone since the 1990s. Where as Ethernet and IP are developed as packet-routing connection-less protocols, ATM is a cell-switched connection-oriented protocol. This poses a number of problems when trying to transport IP over ATM. First, the variable length of the packets don't map efficiently to the fixed size cells of ATM. Drops of a single cell would cause the entire frame to become unusable. Then there is the added overhead of encapsulating IP over ATM, which causes inefficient use of the network resources when compared to running an all IP network. Finally, the QoS features of ATM are left unused [18]. These problems are some of the reasons that operators have moved away from ATM based backbones, to all IP ones.

A.2 SPB

SPB is an evolution of the original IEEE 802.1Q Virtual LAN (VLAN) standard. VLAN tags have been in use in the networking world for a long time and provide separation in campus networks. However, when VLAN-tagging was done at the customer network, the carrier couldn't separate the traffic from different customers anymore. This resulted in 802.1Qad or Q-in-Q which added an S-VLAN tag to separate the client VLANs from the SP VLANs in the backbone. This was usable for the Metro Ethernet networks for a while but when SPs started providing this services to more and more customers, their backbone switches could not keep up with the clients MAC addresses.

To provide the required scalability with regard to MACs in the backbone, PBB (802.1Qay or MAC-in-MAC) was introduced. It encapsulates the whole Ethernet frame on the edge of the carrier network and forwards the frame based on the Backbone-MAC of the egress PE. It also separated client VPNs using a Service Instance Identifier (I-SID), which with 24 bits is able to supply the carrier with 16 million separate networks. The downside of PBB remained one that is common to all Layer 2 forwarding protocols: the possibility of loops. Preventing them requires Spanning Tree Protocol (STP) which will disable links to get a loop-free network. Disadvantages of STP include the relatively long convergence time and inefficient use of resources due to the disabled links. This has been solved by using IS-IS as a routing protocol to discover the network topology. After which each PE creates a Shortest Path Trees (SPTs) originating from each edge device to every other PE. This is called SPB or 802.1aq.

SPB benefits from the maturity of the Ethernet protocol by reusing protocols for OAM and Performance Measurement (PM). This allows for fast error detection and extensive troubleshooting tools by using the Institute of Electrical and Electronics Engineers (IEEE) 802.1ag and International Telecommunication Union - Technology (ITU-T) Y.1731 standards respectively. The Intermediate System-Intermediate System (IS-IS) implementation has also been adapted to rapidly detect errors however, no fast recovery function has been defined, besides complete IS-IS reconvergence. This would result in a traffic impact of several hundreds of milliseconds in large networks [19].

However, due to its Ethernet STP forwarding-based nature it lacks TE features. The paths that the VPN traffic takes are not explicitly configureable and provide limited scalability due to limited amounts of available paths (or trees in this case). As such, operators can not define constraints or explicit paths to take over the network to distribute traffic in an efficient manner. The lack of available paths also negatively affects its ECMP functionalities. However, future, additional algorithms with multiple paths maybe introduced using extensible Equal Cost Trees (ECT) algorithms [20].

Provided that IS-IS has been configured on all provider devices, Figure 10 illustrates the interfaces needed to provision a DVPN in SPB. It excels in its simplicity by providing 'single-point provisioning'. This means that the NMS only needs to add the member CE port to a certain DVPN I-SID, after which the PE floods this binding through the IS-IS network and other PEs sharing this I-SID will install the path towards the ingress

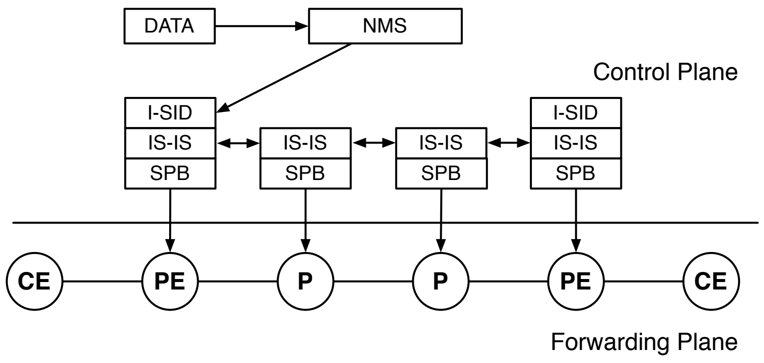


Figure 10: Provisioning a DVPN using SPB.

PE. The rate limiting of the CE ports is a vendor-specific feature however, and may vary per hardware platform.

The simplicity of the architecture comes at a cost. The protocols has:

- limited explicit or constraints-based routing, meaning few TE features,
- limited ECMP functionality due to the infancy of the standard to support it, and
- because failure recovery depends on IS-IS reconvergence, no fast failover.

These limitations are being worked on by the community, e.g. IEEE 802.1Qbp which provides extensive ECMP functions. And since the technology has only been officially standardized since March 2012, it will also need to mature before it is suitable for carrier implementations.

B Acronyms

API	Application Programming Interface
ARP	Address Resolution Protocol
ATM	Asynchronous Transport Method
BFD	Bidirectional Forward Detection
BGP	Border Gateway Protocol
BUM	Broadcast, Unknown unicast and Multicast
CE	Customer Edge device
C-MAC	Customer MAC
CSPF	Constrained Shortest Path First
DA	Destination Address
DVPN	Dynamic VPN
ECMP	Equal Cost Multi Path
ECT	Equal Cost Trees
E-VPN	Ethernet VPN
FRR	Fast Reroute
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IGP	Interior Gateway Protocol
IP	Internet Protocol
I-SID	Service Instance Identifier
IS-IS	Intermediate System-Intermediate System
ITU-T	International Telecommunication Union - Technology
LAN	Local Area Network
LDP	Label Distribution Protocol
LSP	Label-switched Path
MAC	Media Access Control
MPLS	Multi Protocol Label Switching
NMS	Network Management System
OAM	Operations, Administration and Management
OSI	Open System Interconnect
OSPF	Open Shortest Path First
PBB	Provider Backbone Bridging
P	Provider device
PE	Provider Edge device
PM	Performance Measurement
PPVPN	Provider-provisioned VPN
QoS	Quality of Service
RSVP	Resource Reservation Protocol
SA	Source Address
SDN	Software Defined Networking
SPB	Shortest Path Bridging
SPT	Shortest Path Tree
SP	Service Provider
STP	Spanning Tree Protocol
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TE	Traffic Engineering
VLAN	Virtual LAN
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network

C Bibliography

[1] J. Van der Merwe and C. Kalmanek, "Network Programmability is the answer," in *Workshop on Programmable Routers for the Extensible Services of Tomorrow (PRESTO 2007), Princeton, NJ*, 2007.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] "OpenDaylight Project." http://www.opendaylight.org/.

[4] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, "Packet and circuit network convergence with OpenFlow," in *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, pp. 1–3, IEEE, 2010.

[5] H. Zimmermann, "OSI reference model–The ISO model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[6] B. Yousef, D. B. Hoang, and G. Rogers, "Network programmability for VPN overlay construction and bandwidth management," in *Active Networks*, pp. 114–125, Springer, 2007.

[7] "RFC 4026: Provider Provisioned Virtual Private Network (VPN) Terminology." http://tools.ietf.org/html/rfc4026.

[8] Y. Rekhter, B. Davie, E. Rosen, G. Swallow, D. Farinacci, and D. Katz, "Tag switching architecture overview," *Proceedings of the IEEE*, vol. 85, no. 12, pp. 1973–1983, 1997.

[9] A. Sajassi, R. Aggarwal, N. Bitar, W. Henderickx, S. Boutros, F. Balus, K. Patel, S. Salam, A. Isaac, J. Drake, R. Shekhar, and J. Uttaro, "BGP MPLS Based Ethernet VPN," 2013. http://tools.ietf.org/html/draft-ietf-l2vpn-evpn-03.

[10] "Ryu, a component-based software-defined networking framework." http://osrg.github.io/ryu/.

[11] "nox13oflib – NOX Zaku with OF 1.3 support." https://github.com/CPqD/nox13oflib.

[12] R. Van der Pol, "Ethernet OAM enabled OpenFlow Controller," 2011. https://noc.sara.nl/nrg/presentations/SC11-SRS-8021ag.pdf.

[13] C. R. Meiners, A. X. Liu, and E. Torng, "TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pp. 266–275, IEEE, 2007.

[14] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Skoldstrom, "Scalable fault management for OpenFlow," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 6606–6610, IEEE, 2012.

[15] S. Das, A. R. Sharafat, G. Parulkar, and N. McKeown, "MPLS with a simple OPEN control plane," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pp. 1–3, IEEE, 2011.

[16] A. R. Sharafat, S. Das, G. Parulkar, and N. McKeown, "MPLS-TE and MPLS VPNs with OpenFlow," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 452–453, ACM, 2011.

[17] Schenker, S., "Software-Defined Networking at the Crossroads," Presented at Stanford EE Computer Systems Colloquium, May 2013. http://www.stanford.edu/class/ee380/.

[18] T. Mickelsson, "ATM versus Ethernet," in *Proceedings of the HUT Internetworking Seminar*, 1999.

[19] P. Ashwood-Smith, "Shortest Path Bridging IEEE 802.1aq – NANOG 49," 2010. https://www.nanog.org/meeting-archives/nanog49/presentations/Tuesday/Ashwood-SPB.pdf.

[20] "RFC 6329: IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging." http://tools.ietf.org/html/rfc6329.

Acknowledgements

Thanks to Rudolf Strijkers for his supervision during this project.