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 1, 2013

Summary

Hello world.

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]. And so the question arises: *"How much can operators benefit from using OpenFlow when implementing Dynamic VPNs in comparison with contemporary technologies?"* This requires research into whether DVPNs can be implemented with current technologies, as well as with OpenFlow. Finally we will make a comparison between the two architectures.

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.

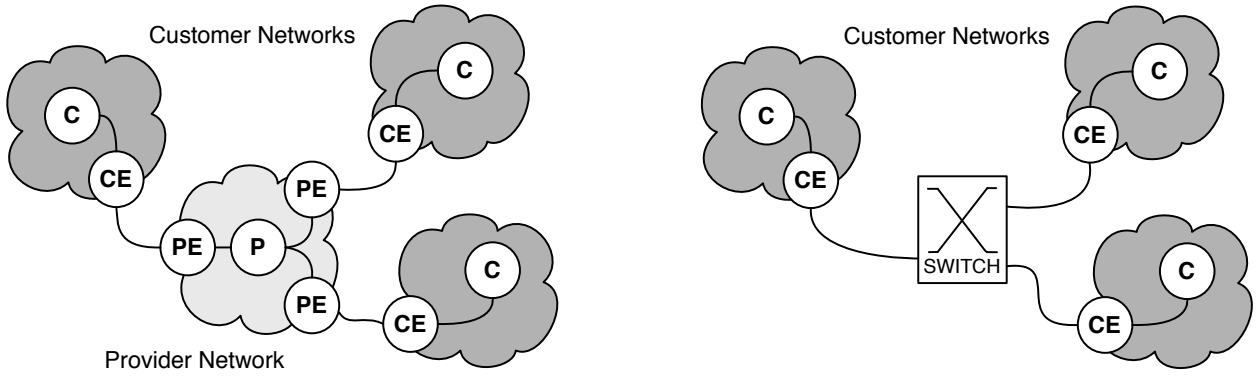
2 Dynamic VPNs

In Section 1.2 we already gave a short description of DVPNs. In this section, we will further look at the actual concept. Starting with defining what it actually provides, how it's carried over the core, the information needed to implement a VPN, from where that information is available and finally working towards a list of technical requirements that the network will need to provide.

2.1 Service

To define the DVPN service, we first take a look at the concepts of non-dynamic, or static VPNs. They 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.

All these functionalities apply to VPNs as well as DVPNs. DVPNs however, also are flexible in nature. They can be configured, adapted and deconfigured within relatively short timespans. Current Layer 2 VPNs are mostly configured statically and changes in their configurations will require manual labor from the engineers. To convert them to DVPNs new tools are needed to automate this provisioning process which we will get back to in Section 2.3.

To summarize, from a service level perspective a DVPN 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.

2.2 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 at PE1 in Figure 2, 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.

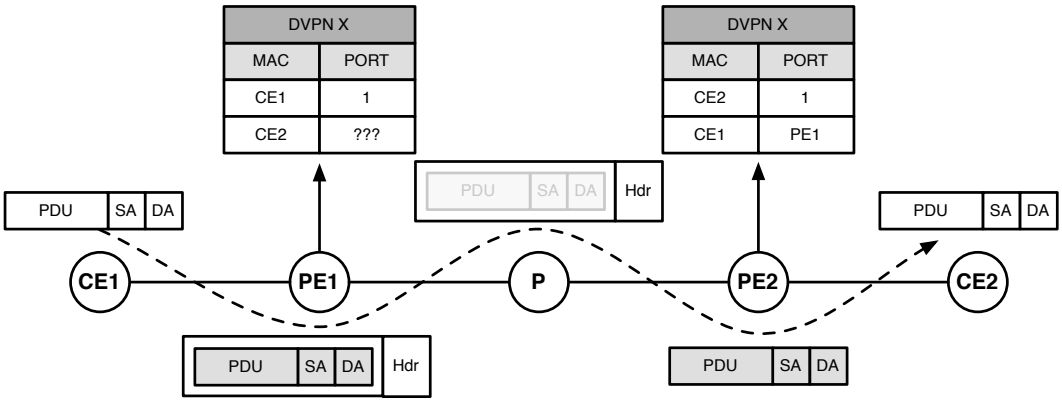


Figure 2: DVPN traffic as it travels through the provider network.

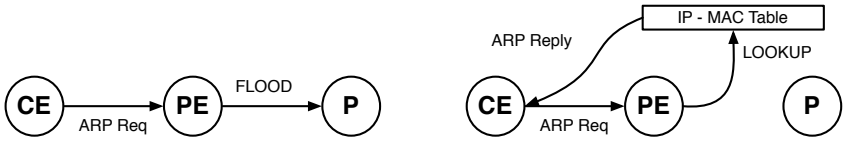
Forwarding from ingress PE to egress PE happens over a path of several Ps. Every PE connected to a CE member of a particular DVPN, should have one or more paths available to each and every other PE with members of that DVPN. The determination of the routes of these paths takes place through a form topology 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. This means that the links comprising a path will need to have enough resources available, but that 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.

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 participating PEs. 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 by 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.

Another addition to rate limiting unknown unicast traffic is by pre-populating the MAC tables of the PEs. This requires that, besides the ingress PE only learning the SA from the CE, it will also actively distribute the SA to all other PEs with members of the same DVPN. Then, instead of flooding unknown unicast frames

to the PEs, the ingress PE drops the frame, knowing that the other PEs will not recognize it either. This can also be extended to limit broadcast Address Resolution Protocol (ARP) traffic if the PEs also exchange the IP address belonging to each C-MAC. When the ingress PE receives an ARP request for a certain IP address, it can look it up in its table and without flooding the frame, reply to the CE with the correct MAC.



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

Figure 3: Processing of ARP requests at the PE.

With multiple DVPNs present on the network it can happen that one DVPN affects the available bandwidth of others. Therefor rate limits will need to be in place for the overall traffic coming in to the CE-connected ports. Policing rates of different DVPNs in the core is nearly impossible, the hardware cannot police traffic of separate DVPNs. And, because it burdens the core with another responsibility while it should only be concerned with fast forwarding, is also undesirable. However, by assigning a minimum and maximum bandwidth rate to each DVPN instance, it is possible to preprovision the 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.

2.3 Provisioning

A DVPN instance consists of multiple member ports, which are identified by their PE device and the port on that PE. The instance also contains values for its minimum and maximum available bandwidth which can be used to determine the paths that the DVPN will get assigned. When member ports reside on different PEs, a bidirectional path will need to be created through the network. The route of the path will depend on *a*) the liveness and administative 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.

When the path between two PEs has been setup for the DVPN, it can be put in the DVPN description. More paths may be added over different routes and paths may be adjusted during the lifetime of the DVPN. 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 DVPNs using those paths can be looked up using the information base. Also other monitoring and troubleshooting processes will profit from this information.

After the complete paths between PEs with DVPN 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 DVPN from using up all the networks resources.

To provision DVPNs in the network, the NMS in short should be able to:

1. take input as certain ports to be placed in a DVPN,
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.

3 Implementation

Using the requirements set forth in Section 2 we can compile a list of features provided by contemporary technologies that meet the following criteria: *1*) providing an Ethernet PPVPNs between multiple sites as defined in Section 2.1 , and *2*) its protocol stack must be supported in hardware available at time of writing. This section will start of with an overview of how these current technologies can be used to implement DVPNs and will finally present a theoretic OpenFlow implementation.

3.1 Contemporary Technologies

3.1.1 SPB

Shortest Path Bridging (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 decent 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, Provider Backbone Bridging (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 [8].

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 Equal Cost Multi Path (ECMP) functionalities. However, future, additional algorithms with multiple paths maybe introduced using extensible Equal Cost Trees (ECT) algorithms [9].

Provided that IS-IS has been configured on all provider devices, Figure 4 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 PE. The rate limiting of the CE ports is a vendor-specific feature however, and may vary per hardware platform.

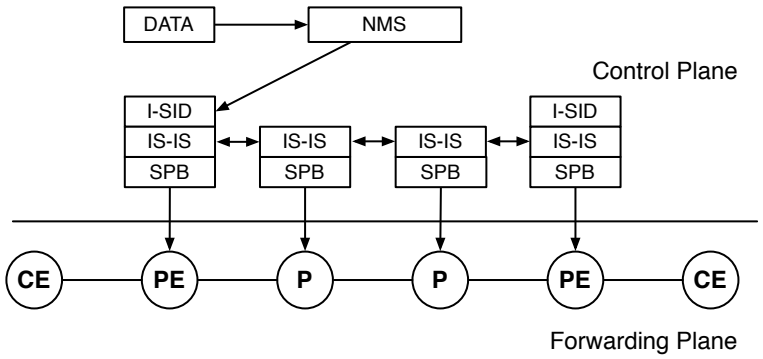


Figure 4: Provisioning a DVPN using SPB.

3.1.2 MPLS

Multi Protocol Label Switching (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 [10]. 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 kicks in. 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. When a large number of C-MACs are present within a VPLS instance this can cause a lot of broadcast traffic, specifically ARP traffic. To solve this, the Ethernet VPN (E-VPN) standard has been proposed [11]. This technique provides MAC learning in the control plane by exchanging learned C-MACs between PEs using Multi-Protocol BGP. Additionally it may also learn the IP address associated with the C-MAC and distribute that as well. Thereby being able to act as an ARP proxy, as earlier illustrated in Figure 3b.

The different protocols all depend on each other, as illustrated in Figure 5. Each PE device runs this stack, while P devices run a subset which is shaded.

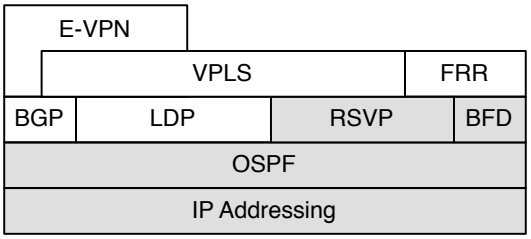


Figure 5: Dependency stack of MPLS-related technologies.

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 again, 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 6.

3.2 OpenFlow

Section 1 gave a short introduction into what SDN and OpenFlow entail and what it promises in terms of cost savings and agility. 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. Essentially, taking over the role of the distributed control planes of the network devices. A general overview of the SDN architecture and OpenFlow is given in Figure 7. OpenFlow provides the controller with an API

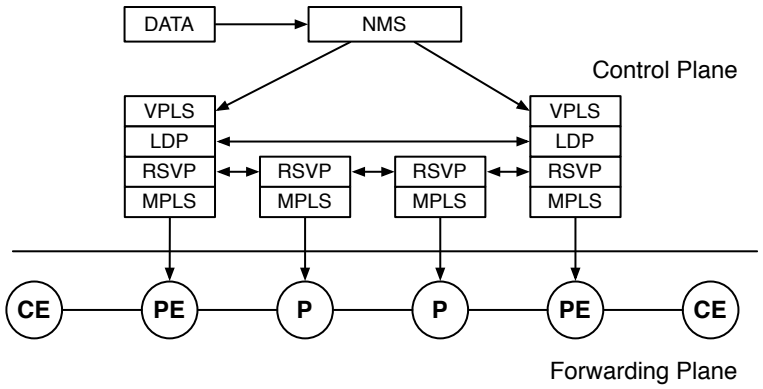


Figure 6: Provisioning a DVPN using MPLS.

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).

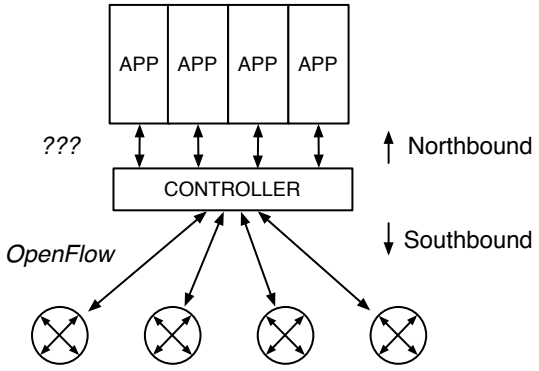


Figure 7: Architecture of SDN and OpenFlow.

Flow entries have been understandably compared to ACL entries, however it is also apparent that there are many more possibilities with regards to matching and performing actions. 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 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

	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.

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.

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.

Fast failover and 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.

One thing to note is that in this paper when discussing the 'controller' component we are referring to a logical controller entity, not specifically a single physical server. 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. The actual design and implementation of such a system is beyond the scope of this paper.

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 8.

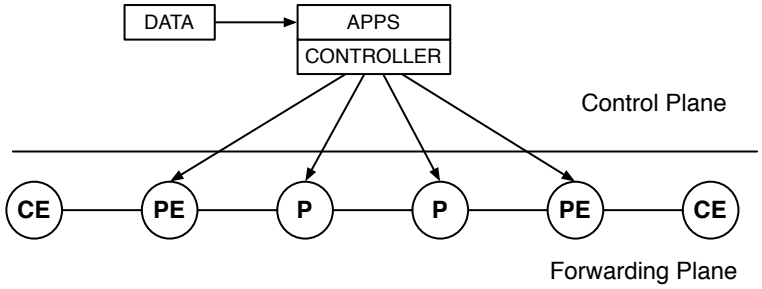


Figure 8: Provisioning a DVPN using OpenFlow.

3.2.1 DVPN Applications

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.

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. The application is also concerned with keep 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.

Path Provisioning Using the discovered topology, paths are then 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).

To provide MAC scalability over the backbone network and provide explicit path routing, the traffic will need to be tagged using MPLS labels. In contrast with the MPLS architecture 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.

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.

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 keepalive 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.

The interaction between the different applications has been illustrated in Figure 9.

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.

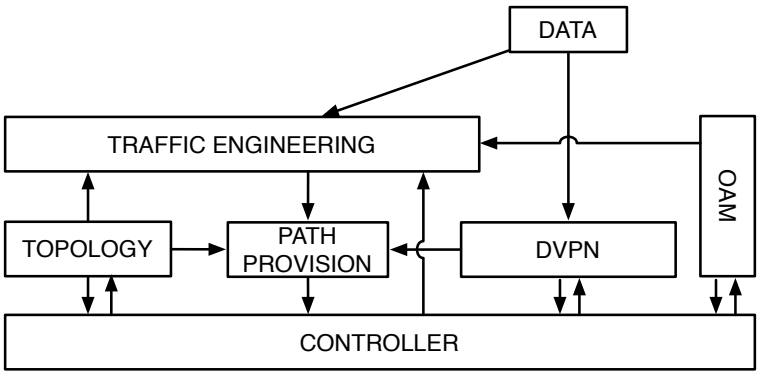


Figure 9: Interactions between applications to implement DVPNs.

4 Results

The features and limitations of the three discussed technologies in Section 3 are given in Table 2. In this section we will compare the contemporary technologies with the OpenFlow implementation when implementing the DVPN service as discribed in Section 2.

	SPB	MPLS	OpenFlow / SDN
Tagging of VPN Traffic	PBB	VPLS (MPLS)	PBB / MPLS
MAC Scalability	yes	yes	yes
Topology Discovery	IS-IS	OSPF	application
Path Provisioning	SPT	RSVP / LDP	application
Traffic Engineering	limited	RSVP	application
ECMP	limited	yes	yes, using Groups
BUM limiting	dependent on HW	dependent on HW	yes, using Metering
Exchange C-MACs	no	E-VPN (draft)	application
Ingress Rate Limiting	dependent on HW	dependent on HW	yes, using Queues or Metering
Fast Failover	no	FRR	yes, using Groups
OAM	802.1ag / Y.1731	LSP Ping / BFD	application
Forwarding Decision	PBB tags	MPLS labels	flow entry
BUM traffic handling	flood	flood	sent to controller

Table 2: Feature requirements available in discussed technologies.

4.1 SPB

The SPB architecture allows for a scalable carrier network supporting thousands, even millions of DVPNs using the I-SID in the PBB frame. From our theoretical implementation in Section 3.1.1, it became evident that setting up a DVPNs requires little configuration. The I-SID only needs to be configured on the PE connecting the CE port and the IS-IS routing protocol distributes this binding to the other corresponding PEs. Another benefit is the use of Ethernet OAM standards that are mature and extensive to allow for precise monitoring and troubleshooting.

However, 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.

Because of the shortcomings of SPB with regards to the use-case set forth in Section 2, we will omit this technology in our comparison. Instead we will focus on comparing the MPLS setup from Section 3.1.2 with the SDN/OpenFlow architecture as designed in Section 3.2.

4.2 Comparison

To get an overview of the key differences between the two architectures we will follow the structure of the DVPN requirement list defined in Section 2.

4.2.1 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.

4.2.2 Transport

Both implementations use MPLS labels to transport frames through the network. 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. The usage of paths also means that the Ps will forward the traffic without relying on or being aware of any C-MACs, providing scalability. OpenFlow supports MPLS labels since version 1.1. Version 1.0 can only separate traffic using C-VLAN tags, which means that *a*) the customer MACs will need to be present in the backbone, *b*) the customer can not use VLANs over the provider network, and *c*) forwarding will be based on the egress PE, not the path, eliminating TE. This latter limitation can of course be overcome using more specific match entries for every traffic flow that needs to be forwarded in certain way, however this would negate the benefit of using MPLS labels to minimize the amount of flows needed in the backbone. These specific flows are called 'microflows' and while giving more precise control over traffic, they will fill up the flow tables of Ps fairly quickly in a provider network with thousands of customers. The only way for OpenFlow to scale up to a carrier network level is by using version 1.1 or later.

Comparing MPLS to the Virtual Circuit (VC)-based Asynchronous Transport Method (ATM) protocol which required configuration of VCs throughout the network, we find that MPLS has the benefit of automatically distributing labels which allows for scalable and easily configurable carrier networks. The LSPs in MPLS are very comparable to the VCs of ATM [13]. Managing and configuring VCs was a problem in the ATM days though, mainly because of the lack of integration between the ATM switches and IP routers. So LDP was a huge advantage in the eyes of the carriers in the early days of MPLS. However, with the advent of explicit routes using RSVP for Traffic Engineering, operators are now trying to do away with the automatic paths setup by LDP. With these strict forwarding controls they are but a small step removed from once again manually setting up VCs.

As mentioned before, OpenFlow will also need to provide path-based forwarding to provide scalability in the network and this has to be implemented with strict control over the labels in each PE and P. However, the advantage that operators have today is that the complete control plane of the network will be in the OpenFlow controller and its applications. The ATM setup required separate management for the ATM switches and the IP routing subsystems such as the IGP and BGP with limited integration between the two.

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 any of the two technologies will not change depending on the implementation chosen. We will however take a look at how they differ in provisioning procedures in the Section 4.2.3.

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.

Rate Limiting

MAC learning in control in draft E-VPN

4.2.3 Provisioning

Discovering the network topology of a distributed network 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 DVPN 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.

MPLS:

- initial setup complicated
- DVPN setup: only each PE with member port
-

OF:

- initial setup nonexistent, no VPNs = no flows (except LLDP/OAM)
- DVPN setup: every PE with member port + Ps in path
- 1.0 supported almost everywhere, 1.1 and 1.2 are not. 1.3 slowly coming.
- TE more intricate algorithms on faster hardware (knapsack problem)

1.3 Controllers: Ryu by NTT [15] NOX extensions by CPqD research center from Brasil [16] MüL from kulcloud (South-Korea) is coming [17]

OpenDaylight controller still lacks 1.3 support

The strength of these applications however, is the fact that operators can integrate their NMS and control plane even more. This allows for a more granular control over their traffic.

MPLS automation software written for DVPNs, but not due to lack of programmable consistent interface to HW, not portable to other vendor, sometimes even model!

complexity high due to intricate dependencies of different protocols

---

OF applications need to solve from ground up, topology, etc... nortbound interface undefined, limited portability of apps between controllers.

MPLS VPNs in OpenFlow: [18]

MPLS control plane in OpenFlow: [19]

Also: access layer intelligence.

5 Conclusion

6 Recommendations

7 Future Work

Other use cases:

- multi-domain
- mobility
- smart metering

A Acronyms

ACL Access Control List

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

TCP Transmission Control Protocol

TE Traffic Engineering

VC Virtual Circuit

VLAN Virtual LAN

VPLS Virtual Private LAN Service

VPN Virtual Private Network

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

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

[10] 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.

[11] 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.

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

[13] I. Pepelnjak, "MPLS is not Tunneling." http://blog.ioshints.info/2011/10/mpls-is-not-tunneling.html.

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

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

[17] "MüL OpenFlow controller." http://sourceforge.net/projects/mul/.

[18] 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.

[19] 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.

Acknowledgements

Thanks to Rudolf Strijkers for his supervision during this project.