University of Amsterdam

Project report

Investigative Research of an IP Peering Service for NetherLight

August 21, 2020

ing. Arnold Buntsma
arnold.buntsma@os3.nl
UvA ID: 12818674

ing. Mar Badidas Simó
mar.badidassimo@os3.nl
UvA ID: 12838268

Assessor:
Prof. dr. ir. Cees de Laat
delaat@uva.nl
University of Amsterdam

Supervisors:
Gerben van Malenstein
SURFnet
gerben.vanmalenstein@surfnet.nl
Migiel de Vos
SURFnet
migiel.devos@surfnet.nl
Max Mudde
SURFnet
max.mudde@surfnet.nl
Marijke Kaat
SURFnet
marijke.kaat@surfnet.nl

Course:
Research Project 2

Abstract

In this project, we investigate how NetherLight, an exchange that facilitates high bandwidth point-to-point and multipoint connections for its clients, can set up a peering service where all its clients can connect. We define its requirements, study the best practices and gather common problems that need to be considered when building such service. We describe how route servers can be used, which IP address space is most suitable and what security measures should be taken. Then, we explored two different solutions, one using EVPN protocol and the other using OpenFlow with the Faucet controller. For both options, we described how they can be managed, monitored and how scalable they are. Finally, we compare both options and conclude that EVPN requires less effort to implement given current NetherLight infrastructure, but OpenFlow is a more advanced solution that offers a more fined-control of the traffic, less management effort and vendor independency.

Keywords— Peering service, EVPN, OpenFlow, Software Defined Network (SDN), Internet eXchange Point (IXP)

Contents

Please notice that in this project, the word "client" will be used with the pronouns he/him/his. However, "client" does not represent a single person, male or female, but an entire organisation or institution.

1 Introduction

NetherLight is an exchange built and operated by SURFnet and facilitates high bandwidth connections for its clients [1]. SURFnet is the organization that created, operates, and maintains the National Research and Education Network (NREN) of The Netherlands. Although NetherLight is part of SURFnet, their networks are separated. NetherLight connects research and education networks with, for example, service providers by facilitating a layer 2[1] domain. On this layer 2 domain, NetherLight offers point-to-point and multipoint connections between its clients. The logical overview of NetherLight's infrastructure can be seen in figure 1.

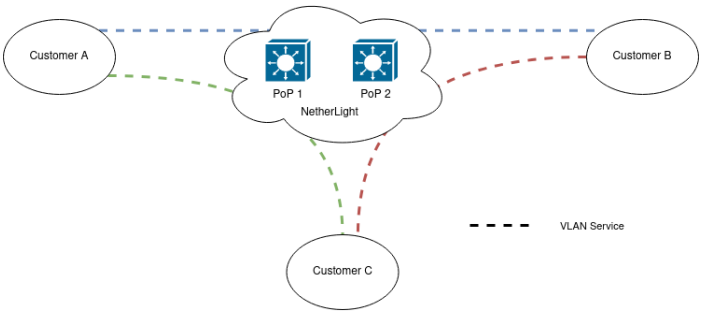

Figure 1: Logical Overview of NetherLight's infrastructure.

NetherLight investigates offering a new service that allows their clients to set up BGP peerings with each other in a fast, manageable, predictable, scalable, and reliable way. Preferably with minimal or no intervention from NetherLight's management and operations.

The concept of a peering service is widely used around the world on Internet eXchange Points (IXPs). IXPs offer peering services where different parties, like ISPs, social media networks, and CDNs, connect and exchange traffic. The clients of NetherLight are different from clients of an IXP. For NetherLight, the clients are mostly research facilities and exchange research data.

The main goal of this research project is to investigate what options there are to create a layer 2 underlay to facilitate layer 3 peering, as well as investigate how customers can connect to the peering network. Also, to investigate the operational environment, and give recommendations on which solution is best suited for the use case of NetherLight. The following main research question was answered:

How can NetherLight facilitate a state-of-the-art peering service for its clients which is flexible, secure, manageable and has a uniform set up?

To answer the main question, the following sub-questions have been defined:

1. What are the requirements for NetherLight (and its customers) for setting up the peering service?
2. What are the options and best practices for setting up a peering service that fulfils the requirements of NetherLight(and its clients)?
3. How do (used) protocols behave in such environment and which problems can arise?
4. What would the uniform onboarding procedure be for clients who are going to use the peering service?

The requirements are set by NetherLight for the peering service and contain all the technical, functional and non-functional demands of the solutions.

For answering the second question, we want to make use of the best-suited technologies to build a service that meets NetherLight's requirements. Best practices include considering route servers, as well as which IP ranges to use, security and manageability considerations.

In the third sub-question, we look at how the peering service protocols work. We research if those have problems like an ARP storm or how ARP spoofing can be prevented. Once known how these protocols will work, NetherLight can predict what will happen in the network and proactively act on this.

In the last sub-question, we look at how clients can join the peering service in a uniform way that is easy and scalable, without compromising the security of the service. We investigate how the configuration can be verified before connecting to the actual peering network. We also look at how this service can be managed with minimum or no involvement of the NetherLight team.

[1] Original OSI model: `https://standards.iso.org/ittf/PubliclyAvailableStandards/s014258_ISO_IEC_7498-4_1989(E).zip`

This report is organised in the following sections: In *section 2* (*Methodology*), the methodology is presented. Then, in *section 3* (*Background*), NetherLight and their current infrastructure is described and some best practices implemented by IXPs. *Section 4* (*Requirements*), answers the sub question regarding the requirements from NetherLight. In *section 5* (*Generic Components*), there are described the generic components that are needed in all the solutions found. Then, in *section 6* (*Solutions*) the different researched solutions are described including their options on monitoring and manageability. In *section 7* (*On- and off-boarding workflow*), the process to join and leave the peering service is described. *Section 8* (*Comparison between solutions*) provides a comparison of the researched solutions on the metrics manageability, security, and scalability. In *section 9* (*Discussion*) we discuss our findings, explain how they comply to the given requirements, motivate the given advice and answer our research questions. Then after the discussion, in *section 10* (*future work*) can the future work be found. Finally, closing off with the conclusion in *section 11* (*Conclusion*).

2 Methodology

For conducting this project, the following approach was used: first step was to define the requirements of NetherLight for the IP peering service. In appendix D the initial list of requirements can be viewed that the NetherLight team provided. These requirements where further defined via interviews with the NetherLight team. The requirements have been used to set up the criteria used to define and evaluate the solutions.

Secondly, a literature study has been performed to research possible solutions for the peering service and to find pitfalls and their mitigation. We studied the current best practices on IXPs, as well as, the current state-of-the-art approaches for setting a peering service. This study was done using scientific publications, protocol specifications and documentation. Other sources of information regarding IXP practices were euro-ix[2], and the database *peeringdb* [3].

Likewise, several Internet Exchanges were contacted to determine several solutions for the peering service and gather their best practices. A list of contacted IXPs can be seen in appendix A and a list of IXPs that replied in the list below. The questions asked are also in the appendix (see appendix B).
IXPs that answered our questions:

- **AAIX** - Klagenfurt, Austria. No web page available, information can be found at `https://www.peeringdb.com/ix/377`
- **AMS-IX** - Amsterdam, The Netherlands. `https://www.ams-ix.net/ams`
- **CATNIX** - Barcelona, Spain. `https://www.catnix.net/`
- **Rheintal IX** - Schaan, Liechtenstein. `https://www.rheintal-ix.net/`
- **TouIX** - Toulouse, France. `http://touix.net/en`
- **TREX Regional Exchanges** - Tampere, Finland. `http://www.trex.fi/`

We contacted the IXPs by mail or using their web form if no mail was available. Our questions were answered by mail (see appendix C) except for TouIX, with whom we had two online meetings with their operator, Marc Bruyere.

From these communications, we learned that even though VPLS is used in a lot of IXPs, migrating to EVPN is highly recommended because it was designed to overcome all of VPLS its shortcomings. A better description of VPLS is given in paragraph 6.1. All IXPs use public IP addresses in the peering service. Route servers are not always used, and they are necessary only after the IXP has over a hundred members, but introducing them at that point is difficult. Also, it is a common practice not to accept unnecessary protocols (Spanning Tree Protocol variants, routing protocols other than BGP, link local protocols, and others (DHCP, GARP, GVRP, VTP, ...). All IXPs agree that ARP storms are a severe problem, and they should be avoided, as they can bring down the entire network. Another common best practice is to allow one MAC address per port to prevent spoofing.

Finally, several solutions for a peering service have been investigated, compared, and proposed. These solutions meet NetherLight's requirements and include a detailed description of their advantages and disadvantages, possible shortcomings, and best practices.

3 Background

This section will provide some background information on NetherLight and its topology, and current best practices from IXPs.

3.1 NetherLight

NetherLight's network is separate from SURFnet's network. However, both networks are interconnected by multiple 100Gb/s links. Clients connected to the SURFnet network can use NetherLight's services using the SURFnet connection because of their interconnection. Besides SURFnet, other parties connected to NetherLight include, but are not limited to, research and education facilities' networks, and cloud service providers around the world. An overview of connected parties can be viewed in

[2] `https://www.euro-ix.net/`
[3] `https://www.peeringdb.com/`

figure 2. This also shows the type of client they are, for example, an open lightpath exchange or (NREN) network. This figure is not up-to-date[4], but will give an indication of the clients of NetherLight. NetherLight used to be a member of GLIF (Global Lambda Integrated Facility) and is now a member of GNA-G [5].

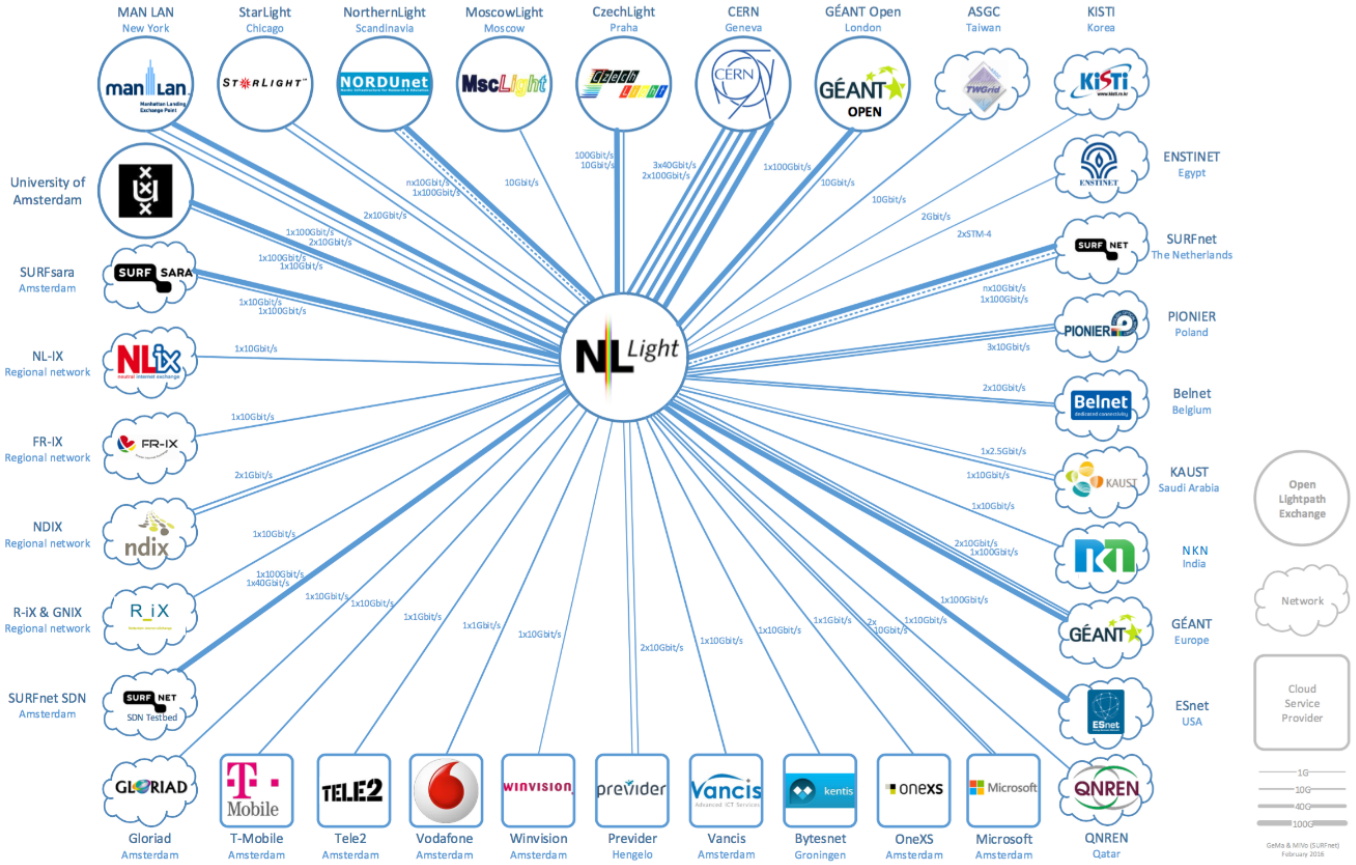

Figure 2: NetherLight's connected networks (outdated) [1].

NetherLight currently offers VLANs (IEEE 802.1Q and IEEE 802.1ad) point-to-point and multipoint connections between their clients. The VLANs are built using Ethernet VPN (EVPN) on top of an MPLS underlay network. A more detailed picture of NetherLight's topology and the relation between the network of SURFnet can be seen in figure 3. The dotted lines represent a layer 2 domain spanning across the networks. A customer can have multiple links to a node or both nodes of NetherLight. The two nodes are in different locations in Amsterdam.

3.2 Best practices on peering services

It is known that peering services have some pitfalls when building traditionally, as most IXPs have built their peering service using VPLS. These pitfalls are mostly related to protocols that send broadcast or multicast traffic. In a broadcast domain that consists of many clients, it is possible that a 'broadcast storms' can cripple the network [2]. Therefore these storms need to be taken care of, and will not cause the network to become inoperable.

Because a peering service is a network where many clients are connected, it is of utmost importance that traffic going over the network is legitimate and not bogus. Therefore, it is wise to limit the type of packets that may enter the network. How and which protocols must be filtered needs to be investigated in this research. For example, it is not advised to have DHCP, or OSPF packets going over the network [2]. This is not only a waste of bandwidth but could also be a security risk if clients receive control packets that are not intended for them but process them.

In a peering service, it is all about peering relations. Therefore, it is essential to keep those secure and no BGP hijacking[6] can take place, or incorrect prefixes are accepted. If this happens, the integrity of the routes will be compromised.

So to summarise, the following pitfalls need to be mitigated when creating a peering network:

[4] An up-to-date version was not available
[5] `https://www.gna-g.net/`
[6] `https://en.wikipedia.org/wiki/BGP_hijacking`

- Broadcast storms.
- Unwanted packet types.
- BGP hijacking.

There is a global initiative for the security of the peering service, that has set norms for routing security. This is provided by MANRS, Mutually Agreed Norms for Routing Security [7]. This initiative "provides crucial fixes to reduce the most common routing threats" (sic) [3]. There is a special 'MANRS IXP Programme' were IXPs can join to get the MANRS compliant status. To achieve this status, an IXP must prove that it does at least three of the actions listed, where the first two are mandatory. The action set is cited from their website [8].

1. "Prevent propagation of incorrect routing information. *The IXP implements filtering of route announcements at the route server based on routing information data (IRR and/or RPKI).*"
2. "Promote MANRS to the IXP membership. *The IXP provides encouragement or assistance for members to implement MANRS actions." This can be done by one or more of the actions provided on their website* [9]
3. "Protect the peering platform. *The IXP has a policy of traffic that is not allowed on the peering fabric.*"
4. "Facilitate global operational communication and coordination between network operators. *The IXP facilitates communication among members by providing mailing lists and member directories. The IXP and each of its members has at least one valid, active email address and one phone number that other members can use in case of abuse, security, and operational incidents.*"
5. Provide monitoring and debugging tools to the members. *The IXP provides a looking glass* [10] *for its members.*"

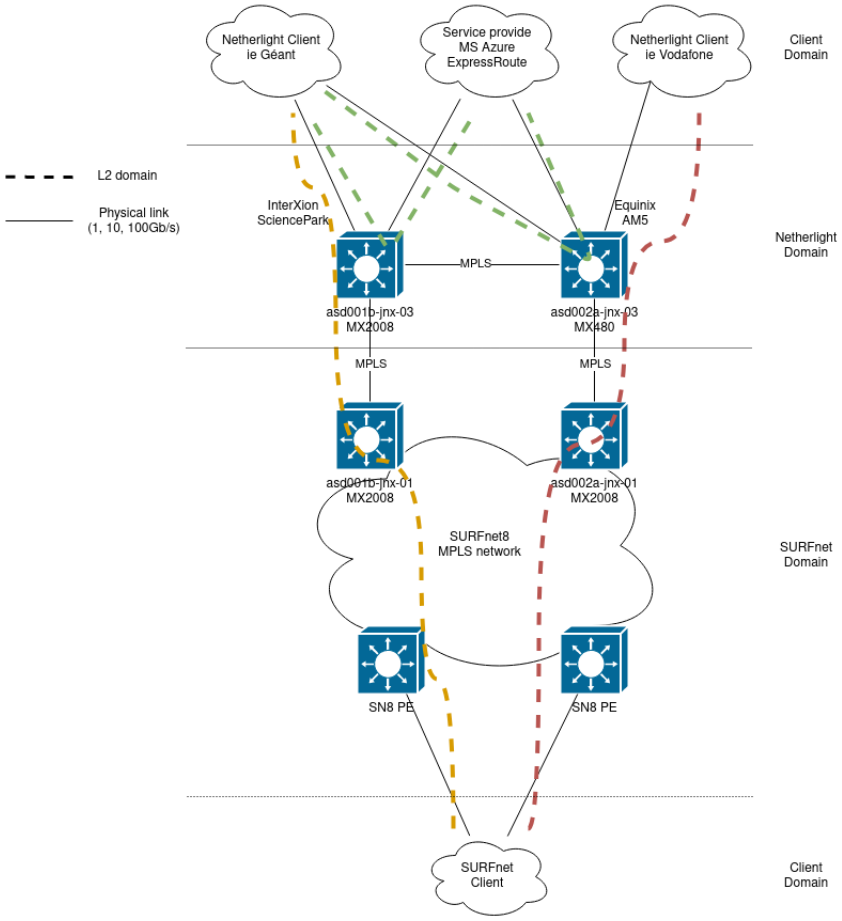

Figure 3: Overview of NetherLight's topology

[7] https://www.manrs.org/about/
[8] https://www.manrs.org/ixps/
[9] https://www.manrs.org/ixps/
[10] https://en.wikipedia.org/wiki/Looking_Glass_server

4 Requirements

The first research question is to find out the requirements of the peering service. In appendix D we can find the first list we received from NetherLight. Then, trough multiple discussions with the NetherLight team, we detailed and specified them. The following list contains the final requirements from NetherLight and their clients. The researched solutions should adhere to these requirements.

- **A detailed explanation of what the service is and how it works.** 'The service' refers to the peering service. The detailed explanation includes a protocol stack, a description of how the solution operates, can be managed and scales.
- **Clients set up BGP peerings with each other.** The peering service has the objective to facilitate the BGP peerings and traffic exchange between the clients. Therefore clients will setup BGP peerings with each other. Not all clients have to peer with all other clients.
- **Clients connect to the service using either one or two VLAN tags towards NetherLight.** Clients that are connected to NetherLight already use specific VLANs for specific services. Some clients use IEEE 802.1ad tags (a.k.a. double tag) for current services. The requirement is that they will connect to the new service similarly, with either one or two VLAN tags.
- **There is a clear onboarding process.** The full process of a client, that is physically connected to NetherLight and wants to join the peering service needs to be described.
- **The service needs to be well-manageable.** *Well-manageable* means in this case that the clients will uniformly use the service. Unwanted behaviour and configuration errors should be eliminated as much as possible. Minimal management should be needed from NetherLight's operational perspective.
- **The service is secure and scalable.** *Secure* means in this context that clients should not be able to interfere with connections of other clients. This entails, for example, spoofing of MAC and IP addresses, and BGP hijacking. *Scalable* means in this context that the peering service can scale to at least hundreds of clients.
- **At least one of the solutions from this research work can be implemented on the current NetherLight hardware (Juniper MX2008).** This requirement makes sure that NetherLight can implement one solution without having to change their current hardware setup.

5 Generic Components

In this section, the generic components have been presented that a peering service needs to consider. These do not depend on the specific technology used. We have described route servers, IP address allocation, and security. These services should used in each solution.

5.1 Route servers

A Route Server (RS) is a brokering system for exchanging network reachability information, BGP routes, between the peering service clients [4]. These clients announce their BGP routes to the RS using an eBGP session. Then, the RS forwards this information to each connected client according to its configuration. The type of peering that the clients set up using an RS is called multilateral peerings (i.e., many-to-many), in contrast with bilateral peerings (i.e., one-to-one), where each client peers with another individually. RSs use BGP to exchange Network Layer Reachability Information (NLRI) with each of its clients, but it does not forward traffic, so it is not a router. Notice that the presence of an RS in exchanges does not prohibit the creation of bilateral peerings between clients, but offers extra service. Therefore, bilateral peerings and route servers coexist in peering services.

RSs are key enablers for providing the clients with more efficient and sophisticated peering options in an exchange [5]. Being n the number of clients' routers connected to the peering service, the largest amount of possible BGP sessions is $n * (n - 1)/2$. As the number of clients increases, the administrative overhead to set and maintain the BGP connections also increases. With RSs, the amount of BGP peerings for a new client can be reduced from $n * (n - 1)/2$ to the number of RSs [6]. In other words, RSs will allow BGP connections over the peering service with low administrative overhead for the clients. Also, multilateral connections through the RSs can be used as a backup if a bilateral session to a member becomes inactive. RSs also provide several security mechanisms, to avoid prefix hijacking, for example, as described in 5.1.1 (*Filtering and communities*).

As stated by Fenioux [7], an RS can be considered as a single point of failure because multilateral peerings depend on it. Therefore it is recommended to have at least two instances of an RS operating in a mirror configuration. Also, the RSs should be located in a separate AS, as euro-ix [8] states it. This way, the RS behaves as a client of the exchange on its own, and clients peer with it as they would do with any other client.

There are several daemons available for enabling Route Servers such as Quagga, FFRouting, ExaBGP, OpenBGPD and BIRD. Making a comparison between these is out of scope for this project but one can be found at [11]. Using this comparison and the

[11] `https://www.bizety.com/2018/09/04/bgp-open-source-tools-quagga-vs-bird-vs-exabgp/`

best practices found from IXPs, BIRD was chosen because of its powerful filtering mechanisms [9]. All IXPs that we contacted who were using route servers, used the BIRD daemon. BIRD[12] (Bird Internet Routing Daemon, a recursive acronym) is an open-source daemon for routing Internet Protocol packets on Unix-like operating systems.

BIRD maintains a Routing Information Base (RIB) with the paths it receives from its peers. BIRD supports any number of routing tables. There is a default table, called *master* and other tables have to be defined in the configuration. These tables can exchange routes between them. Filters stand between tables and can modify, reject or accept routes. It is a common practice among IXPs [5] to maintain peer-specific RIBs, i.e., client-specific RIBs, and use the filtering to enable control for clients and the route server administrator over what prefixes are exported and imported using the route server.

5.1.1 Filtering and communities

Filtering needs to be done properly otherwise, RS clients can get routes that should not be distributed to them, and BGP security issues can arise. We have looked at various filter policies[7], [10]–[12] from IXPs that implemented RSs and we have listed the most common ones:

- **Filter bogon and martian prefixes**, e.g. 192.168.0.0/16, 10.0.0.0/8 (RFC 1918 [13]). This prevents leaks of internal routes and can be done using the list from Team CYMRU [14].
- **Filter bogon AS Numbers**. Paths that contain reserved and private AS Numbers need to be filtered.
- **Filter prefixes that are too large**. IP prefixes that have a size smaller than /8 for IPv4 and than /19 for IPv6 [15].
- **Verify the AS_PATH**. Confirm there is at least one AS in AS_PATH and the left-most AS is the peer's AS. Filter announcements that do not comply.
- **Verify next-hop IP**. The next-hop IP address should be the same as the source IP address of the packet for preventing next-hop hijacking. An exception can be made to allow participants with multiple connections to advertise their other IPs.
- **Sanity check**. Filter prefixes with an empty AS_PATH (i.e. internal AS routes) or longer than 64 AS numbers.
- **RPKI validation.** Validate the Route Origin using ROAs. If RPKI is valid, then accept route. If RPKI is invalid, then drop route. If RPKI is unknown, then revert to standard IRRDB prefix filtering. This prevents hijacking of prefixes.
- **IRRDB filtering**. For ensuring the origin AS is in the set from the member AS-SET.

Clients need to be able to decide how their routes are announced, and from whom they receive routes. This process can be done using BGP communities. The amount of detail that can be given to the clients for filtering announcements is vast. We present four basic functionalities:

- Block announcement of a route to a certain peer.
- Announcement of a route to a certain peer.
- Block announcement of a route to all peers.
- Announcement of a route to all peers.

Communities would allow NetherLight' clients to decide how their routes are distributed in the RS. Without them, they would not have control of who receives their announced routes and connecting to the RS would mean announcing all routes to all clients equally, preventing the clients to control their routes distribution.

Filtering and communities allow very fine-grained control of the prefixes that are announced through the RSs from the clients of the peering service. If an RS is implemented, proper control of the announced prefixes is crucial. As mentioned, the possibilities are huge, and communities offered to the client can be extended. Due time limitations, we will not do into a deep analysis on communities on the NetherLight specific case. Communities are described in RFC 1997, RFC 4360 and RFC 8096 [16]–[18]. Extended usage of communities can be found on CATNIX web page on route servers [13].

5.2 IP space

The peering service is a layer 2 network that is realised by NetherLight. However, clients using the network are going to speak IP on top of the layer 2 domain. Therefore an IP address strategy is needed. It is recommended to use a public range which is at least a /23 subnet for IPv4 and /64 for IPv6. It should be a public range to ensure that they are globally unique. The reservation should at least be a /23 because of growth expectations of "hundreds of clients", and a /23 allows up to 510 hosts in the subnet. For IPv6 a /64 is recommended because a prefix smaller than /64 should not be used according to RFC 5375 [19]. When NetherLight would grow beyond their expectations the IPv4 space needs to be extended with a new IP address block. The strategy for this was out of scope for this project. The IPv6 space is sufficient even if NetherLight would grow beyond their expectations.
The IP addresses are statically assigned to the clients and should be 1 IPv4 address and 1 IPv6 address per interface. The

[12] `https://bird.network.cz/`
[13] `https://www.catnix.net/en/route-servers/`

orchestration system can choose the IPv4 addresses by checking which is the next available one. It should be administrated which IP address is assigned to which client and interface. For the IPv6 addresses, are a couple of options available to choose the addresses. These are stated in RFC 5963 and listed below [20]. For IPv4, the first option could be used.

1. Decimal encoding of the client's ASN. For example, if the peering service uses the prefix of 2001:db8::/64 and a client has ASN 43231, the IPv6 address will be 2001:db8::43:231:1/64.
2. Hexadecimal encoding of the client ASN. This is less human-readable but can be preferred. Using the same example as above, the IPv6 address would be 2001:db8::a8df:1.
3. Relate a portion of the IPv6 address with the IPv4 address. For example, one could use the host part of the IPv4 address in either decimal or hexadecimal encoding in the IPv6 address. For example, if the IPv4 address would be 145.100.96.103/24, then the IPv6 address can be 2001:db8::103:1/64.
4. Base the IPv6 address on the ID used to identify the client in the administration system of NetherLight uniquely. For example if a client's ID is 0x43231a1 the IPv6 address could be 2001:db8::432:31a1:1/64.

NetherLight should consider all options and choose one. We recommend using the first option because it makes it easy to recognise which IPv6 address belongs to which ASN and thus which client.
RFC 5963 [20] and Bruyere [2] state that some IXPs use a separate VLAN for IPv6 traffic with the advantage of easily creating statistics for IPv6 traffic. However, this does require an extra VLAN to be set up, which leads to extra room for errors and more management efforts. Therefore we advise using the Dual-Stack option. This means that IPv4 and IPv6 run over the main peering service.

5.3 Security

The following security best practices were collected by contacting different IXPs as listed in the methodology (section 2):

- Allow one MAC, and IP address combination per connected (virtual) interface to the peering network. Assign these statically and drop anything that does not adhere to these rules. This can be done with layer 2 ACLs (on VLANs or ports, either is possible) or FlowRules [21].
- Only allow the Ethernet types [2]:
  - 0x0800 - IPv4
  - 0x86DD - IPv6
  - 0x0806 - ARP
- Drop all broadcast and multicast traffic from the client that is not broadcast ARP or multicast ICMPv6 ND.
- Drop all link-local protocols except ARP and ICMPv6 ND.
- Disable/Quarantine the port if abuse or malicious activity is detected, or reported by clients.
- Clients that do not want to peer with every one to set communities according to those defined in the filtering section 5.1.1 (*Filtering and communities*).

6 Solutions

In this section are the investigated solutions described. This was mostly a literature study, in addition to that, we have had interviews with Marc Bruyere from TouIX, and had contact with multiple IXPs how they realised their peering service. It is explained how the peering service can be realised and how the requirements are met. The following options are presented:

- EVPN
- SDN/OpenFlow

6.1 EVPN

Ethernet Virtual Private Network, or EVPN for short, is a protocol that is specified in 2015 for the BGP MPLS-Based EVPN. This protocol is specified in RFC 7432 [22]. EVPN is a protocol defined to overcome certain limitations of Virtual Private LAN Service (VPLS). The limitations are on "multihoming and redundancy, multicast optimisation, provisioning simplicity, flow-based load balancing, and multipathing" (sic) [22]. VPLS is a protocol designed to create a layer 2 domain that can span across multiple locations. As mentioned in section 3 (*background*), NetherLight currently uses EVPN to offer its services. It has been running for over a year now and has been stable throughout that period of time. For this reason, together with the VPLS issues that EVPN overcomes, VPLS will not be considered as an option for this research. EVPN can be run on different underlay networks, such as MPLS or VXLAN. Currently, NetherLight uses MPLS as underlay technology.

Running EVPN on a VXLAN underlay is possible if one has IP connectivity between the sites that need to be connected. However, if traffic engineering is needed, it can only be achieved with either MPLS and RSVP-TE or with the newer SR-MPLS (Segment Routing). VXLAN is reliant on IP, on which it is not easy to do traffic engineering because it relies on third parties,

i.e. peers, that need to respect the traffic engineering that was done, i.e. MED in BGP [23]. However, a VXLAN underlay network is much easier to set up than an MPLS network. VXLAN only needs support at the edge nodes, while for MPLS, it is needed to be supported by all nodes. Thus, if requirements do not include traffic engineering and the explicit use of MPLS, then VXLAN would be a right and easy solution. However, if there is already an MPLS (core) network, traffic engineering is needed, or if the core network is complicated and exists of many nodes, then MPLS as underlay network would be best suited. EVPN works similarly on both underlay networks. An advantage of MPLS over VXLAN is that it has much less overhead. VXLAN has a 50 byte overhead while MPLS only has 4 bytes overhead [24], [25]. This leads to better performance on the network in terms of goodput. Both VXLAN-EVPN and MPLS-EVPN use MP-BGP as control plane, and the concepts are the same. The next subsection will go into more detail of the EVPN solution. No distinction is made between the two underlay networks as they have no other significant impact on the working of the protocol.

6.1.1 EVPN Overview

In this subsection, we give an overview of how EVPN operates. EVPN has a specific terminology that first needs to be understood. The terminology is from RFC 4732 [22]:
CE: Customer Edge device. In most cases, this would be a router, but it does not have to be.
EVI: EVPN Instance spanning the Provider Edge devices participating in that EVPN.
ES: Ethernet Segment, the set of Ethernet links that are used by a customer site to connect to one or more PEs.
ESI: Ethernet Segment Identifier, unique non-zero identifier for an ES.
Ethernet Tag: Identifies a particular broadcast domain, for example, a VLAN. An EVI consists of one or more broadcast domains.
MAC-VRF: Virtual Routing and Forwarding table for MAC addresses on a PE.
PE: Provider Edge device. In most cases a router. This device is on the edge of the MPLS infrastructure.
Single-Active Redundancy Mode: An ES is operating in this mode when only a single PE, among all attached PEs of an ES, is allowed to forward traffic to and from that ES for a given VLAN.
All-Active Redundancy Mode: An ES is operating in this mode when all PEs are allowed to forward traffic to and from the connected ES.

Now that the terminology is known, an overview of the protocol will be given. Starting with the EVI. An EVI is made up of CEs that are connected to PEs, and the PEs provide the layer 2 domain for the CEs. A key benefit of EVPN is that the MAC learning between the PEs happens in the control plane, instead of in the data plane where it would traditionally occur [22]. By transferring the MAC learning from the data plane to the control plane, there are options available to control the MAC learning process, e.g. control who learns what and apply policies. The control plane for EVPN is Multiprotocol BGP (MP-BGP) [22]. The MAC addresses that are distributed via the control plane are the ones from the CEs attached to the PEs. Another advantage of learning from the control plane is that it enables load balancing to multihomed clients, and improves convergence times when a link failure occurs [22]. This is in addition to the load balancing done in the MPLS core. The PEs, however, learn the MAC addresses of connected CEs using a method that is best suited for the CE. The learning could be, among others, done via data-plane learning, IEEE 802.1x, LLDP, ARP. In Figure 4, an overview can be seen where all the components are placed and work together in EVPN.

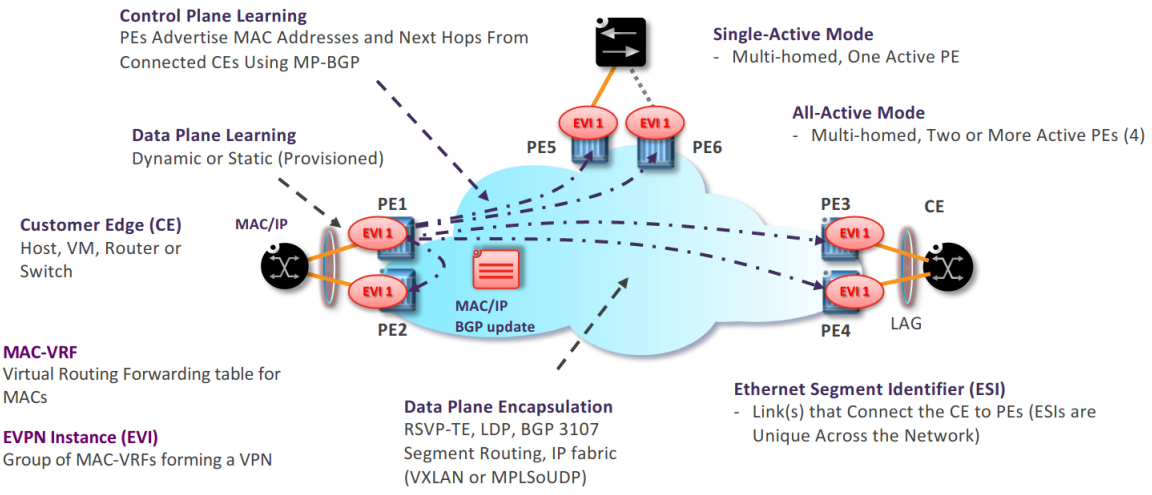

Figure 4: EVPN Fundamental Concepts by Khatri [26]

In the NetherLight topology, two Juniper routers currently work as PEs, while the clients' routers are the CEs (see figure 3).

6.1.2 EVPN implementation

Creating a peering service can be done with EVPN by creating an EVI with a VLAN where all clients connect. This connection can be made using a specific VLAN tag on the client's side, which will be directed to the peering service EVI. On the EVI is a VLAN which is the main peering service.

NetherLight can create multiple VLANs in the peering EVI to separate multiple research groups. However, this requires additional management effort and is only scalable to 4095 groups and one peering VLAN. A reason for this could be to eliminate ARP/IPv6 ND traffic, but this will not cause storms in EVPN because of the PEs act as an ARP/ND proxy. Thus ARP and IPv6 ND traffic is not forwarded on the data plane but answered by the PEs and exchanged between the PEs via the MP-BGP control plane. EVPN changes broadcast traffic by modifying it to unicast traffic. Because PEs have learnt all destinations via the control plane, the broadcast traffic can be unicasted to the destination.

Putting this all together in a protocol stack shows the packet structure that will go over the network (see figure 5, 6). In the PEs are the MPLS/VXLAN and EVPN labels stripped and added. What is interesting to see here, is that the VXLAN version has an extra header than MPLS. Now it is made more clear why VXLAN has more overhead then MPLS.

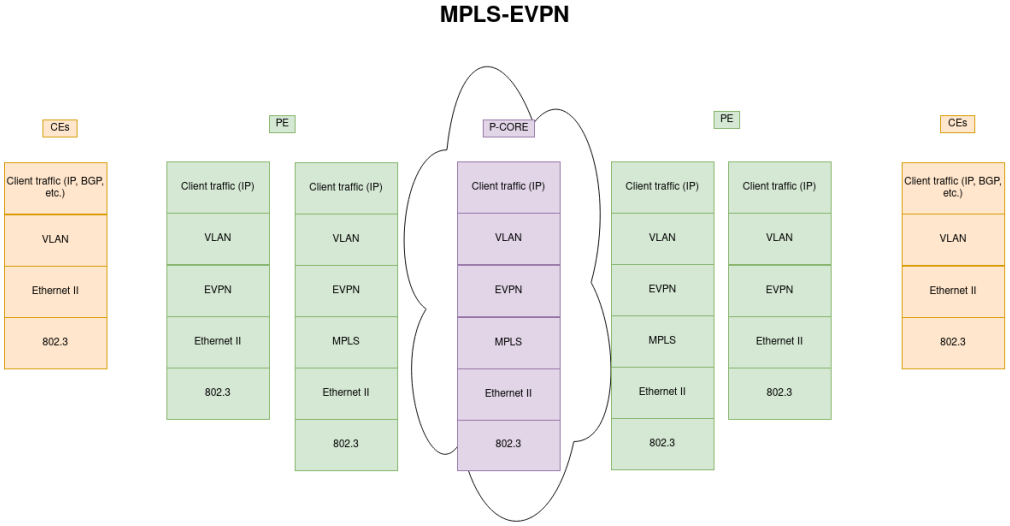

Figure 5: MPLS-EVPN protocol stack

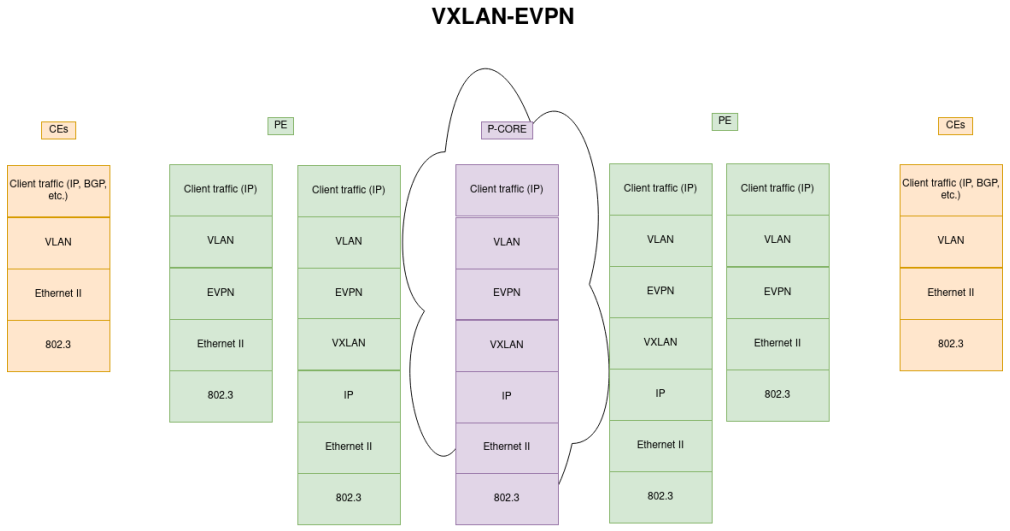

Figure 6: VXLAN-EVPN protocol stack

An overview of how NetherLight should implement the EVPN solutions can be seen in figure 7.

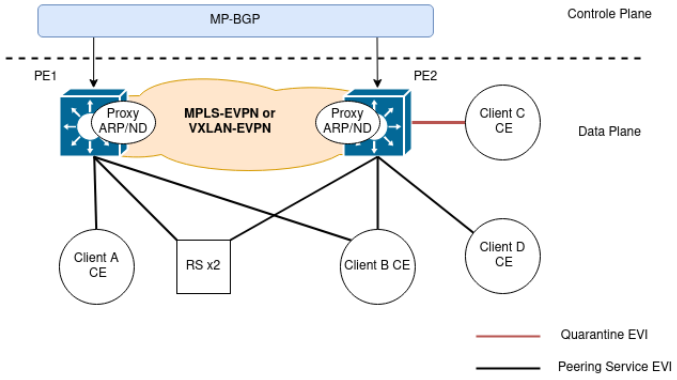

Figure 7: Overview of proposed EVPN solution

6.1.3 Management

(MPLS-)EVPN can be managed most easily using an automation platform which pushes the correct configuration to the nodes in the network. Other management systems like IPAM for IP management aid the automation platform. Currently, NetherLight has the automation and orchestration tool Network Services Orchestrator (NSO) by Cisco [14] in production. This tool uses YANG for modelling the services and NSO will put that model through a Network Element Driver (NED) which will generate the configuration for a node. NSO can then push the configuration via Netconf, REST API, or CLI to the nodes [27]. NetherLight has already templates for creating VLANs, EVIs, and adding clients to these. We recommend to use these and adapt them so it will point to the peering service EVI and VLAN to keep management effort to a minimum.

6.1.4 Monitoring

For monitoring, currently, SNMP is used from the current Juniper routers. A problem with this is that it can not show differentiation between protocols and look inside the packets. SNMP has only basic statistics. sFlow, is, however, sample-based, meaning it will sample once every x packets. Therefore it will never be 100% accurate. sFlow uses a dedicated chip in the device to offload the CPU of this task. The sample rate for NetherLight is hard to determine beforehand. A good balance should be found in accuracy and resource usage (of the sFlow chip). Following the directives of sFlow [15], the recommended sampling rate for the NetherLight links are:

- 100 Gb/s link 1:5000
- 10 Gb/s link, 1:2000
- 1 Gb/s link, 1:1000

It can sample all types of traffic. It is also recommended for high bandwidth environments because it does not put the load on the CPU and is sample-based and thus will not have to process all the packets [28]. Because of the dedicated chip and being sample-based, it is a perfect solution for a high bandwidth environment. Also, all IXPs that we have contacted use sFlow for their monitoring of traffic. sFlow packets consist of the sampled packet, in and out interface, sample rate, information about how it was forwarded, and interface counters [29]. Those packets are then sent to an sFlow collector and analysed using an sFlow analyser which will collect and analyse the packets respectively. The different aspects of the network that will be monitored are out of the scope of this project.

6.2 SDN/OpenFlow

Software-Defined Networking is a paradigm that advocates for decoupling the network data plane and the control plane. Its objective is to abstract the underlying infrastructure from the applications and network services [30]. OpenFlow is a standard for implementing SDN in network equipment [31]. OpenFlow allows a network controller to make high-level switching decisions and instruct the hardware on how to handle data packets. It consists of three main parts: flow tables installed in the hardware; a controller that manages the network; and the OpenFlow protocol that allows the communication between the hardware and the controller. A representation of an OpenFlow enabled network is shown in figure 8.

We advocate that an OpenFlow based solution can offer an innovative and state-of-the-art solution for NetherLight peering service. We consider that a peering service that connects research groups is an ideal place to foment innovation in the Internet ecosystem. SDN programmability, together with its fine-grained control capabilities, can provide high responsiveness, easy network management and vendor independency. It would also provide higher levels of abstraction for network services and

[14] https://www.cisco.com/c/en/us/products/cloud-systems-management/network-services-orchestrator/index.html
[15] https://blog.sflow.com/2009/06/sampling-rates.html

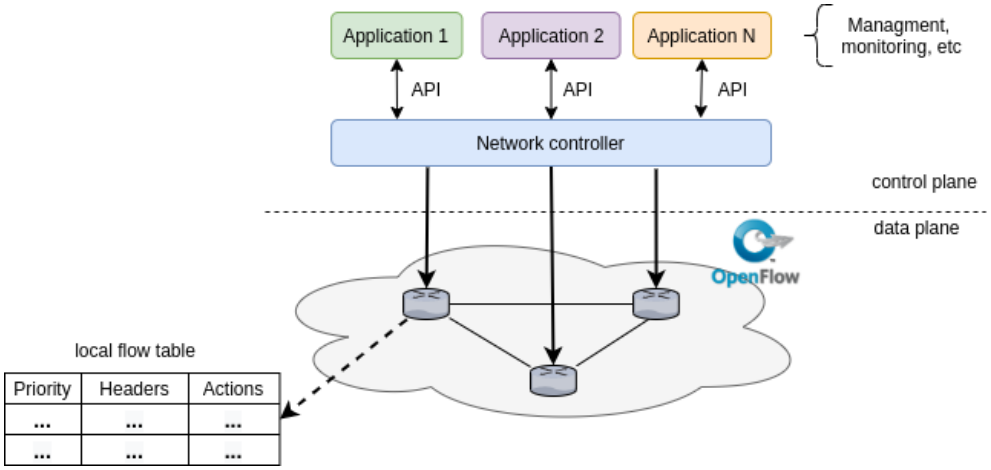

Figure 8: SDN paradigm with OpenFlow.

allow NetherLight, in the future, to create a broader range of programmable services.

Our initial thought in adapting NetherLight's network architecture was the possibility to allow OpenFlow controlled connections between the PEs and the clients while maintaining the MPLS core network for steering the traffic between the PEs. The Juniper routers used by NetherLight (MX2008 and MX408) support OpenFlow version 1.3. In this scenario, we could have enabled OpenFlow on the Juniper interfaces where the clients are connected to, which would be configured, controlled and monitored by an SDN controller. Then, the MPLS protocol would be configured on the interface(s) facing the core network. This solution, though possible, is over-engineered. Adding an extra controller, an extra management system and monitoring just for some interfaces while maintaining current infrastructure for the interfaces facing the core is adding unnecessary complexity. NetherLight would have to maintain and control duplicate systems and work in parallel with two different technologies but without any added benefit that could not be achieved using OpenFlow or MPLS independently. Therefore, for implementing a solution using OpenFlow, a new core network is needed.

Without the current MPLS core network, both NetherLight locations need a new topology to connect. As current Juniper MX2008 do not support OpenFlow with multi-table (see section 6.2.1), new hardware is needed. Just one device per location will be considered following the directive of the NetherLight team. We show the proposed topology for NetherLight's peering service using OpenFlow in figure 9, with some clients as an example. As only two devices need to be connected, we propose a direct connection between them, using link aggregation to allow for resilience.

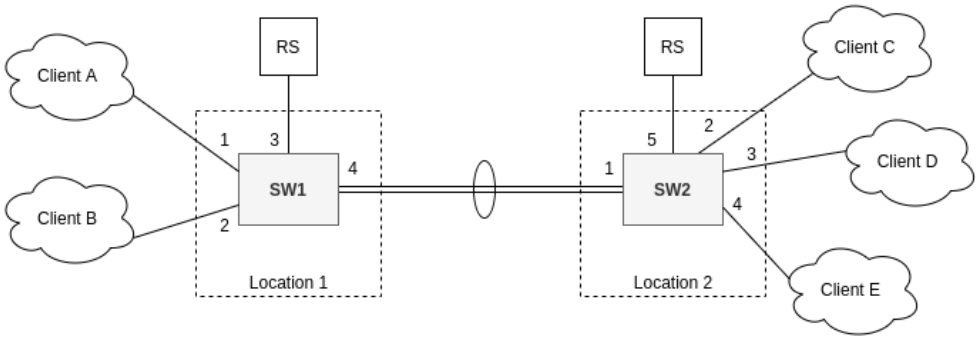

Figure 9: Proposed OpenFlow topology

The SDN paradigm for deploying a peering service has been used in three Internet Exchanges or SDXs (Software Defined eXchanges) in the entire world: Toulouse IXP (TouIX), NSPIXP-3 in Osaka and PIX-IE in Tokyo. All of them have been implemented following the Umbrella approach [32]. In Stringer, Pemberton, Fu, *et al.* [33] a distributed router using OpenFlow for exchanging traffic between the Research and Education Advanced Network of New Zealand (REANNZ) to the Wellington Internet Exchange (WIX) has been implemented. All these projects have as reference the work published by Gupta, Vanbever, Shahbaz, *et al.* [34] that introduces the concept of SDX and a prototype. The same authors propose an industrial-scale version in Gupta, MacDavid, Birkner, *et al.* [35], called iSDX. Siem Hermans and Jeroen Schutrup [36], former SNE students, investigated the possibility of converting AMS-IX to an SDX using iSDX, which concluded that was not possible due to scalability issues. The Umbrella approach is different from their approaches and proves that it is in fact scalable. The stability has also been

proven by the TouIX that has been running on OpenFlow since September 2015. Finally, there is the ENDEAVOUR project, that leverages some of the problems presented by former SDX publications [37].

6.2.1 Faucet Controller

Faucet controller[16] was chosen because it is based on Ryu. Ryu is a component-based software-defined networking framework written in Python that was used to develop the SDN controllers for TouIX, NSPIXP-3 and PIX-IE. Faucet is based on Ryu for production networks. It is light, compact and open source, built on the OpenFlow 1.3 standard. It uses the multi-table packet processing pipeline, which achieves greater scalability on the number of flow entries as it distributes flow entries among several tables. It also implements LACP (Link Aggregation Control Protocol), needed for the proposed topology. It can be installed in Ubuntu Linux, running as an application.
For a peering service, Faucet could be configured using the directives developed on the Umbrella project by Bruyere [2] but adapted to the NetherLight topology. Faucet requires hardware that supports OpenFlow version 1.3, with multi-table support. It has been tested in multiple vendors' hardware and virtual switches [38].

In figure 10, the usage of multi-tables and possible actions on each table in Faucet can be seen. This controller will allow fine-grained control of the traffic. We propose the creation of one VLAN, where clients can connect to using a specific VID. However, several VLANs can be created if NetherLight wants to create VLANs for specific groups of clients. The maximum number of possible VLANs is limited to 4096, as it uses the 802.1Q header. Using QinQ (802.1ad) is also supported. The first table in the flow pipeline will drop any packet that does not belong to the assigned VLAN on its incoming interface to eliminate VLAN hopping. Also, Faucet's default behaviour is to drop all packets that do not match any rule. Therefore, allowed EtherTypes can be defined per VLAN. A quarantine VLAN is not needed: the clients will not be able to introduce any unwanted traffic as the packets will be dropped before they enter the peering service. All the clients' MAC addresses should be known in advance in order to deploy the appropriate rules on the switches. IP addresses for the clients will be handed out by NetherLight and therefore also known in advance to prevent clients using the wrong IP and MAC address combination.

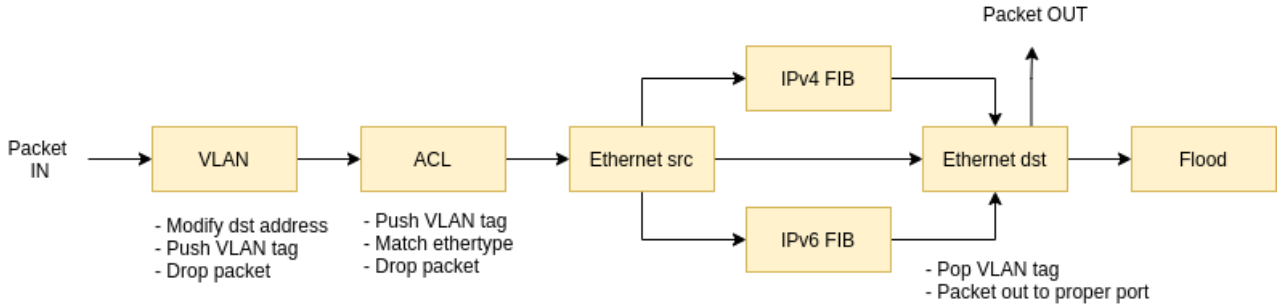

Figure 10: Flow table diagram with possible actions

The peering service can take full advantage of the ability of OpenFlow to modify the packet headers on matching a given rule [39]. VLAN tags can be changed on-the-fly, so if a client A uses the VLAN tag 100 for other services, it can be assigned an unused one 200 that will be used for the other services. Then, a rule can be introduced: on any packet from client A with tag 100 traffic will be retagged to 200 and vice versa.

Another key feature is the elimination of the location discovery mechanisms based on broadcast packets (i.e., ARP request). Using Umbrella [32] as a reference and adapting it to the proposed topology (figure 9) we can eliminate the ARP storm problems by translating broadcast to unicast traffic. This is done by replacing the MAC address of the broadcast packet to the destination. For example: if the client A is connected to switch 1, and wants to know an IP of a client C connected to switch 2, it will send an ARP request to the broadcast MAC address (ff:ff:ff:ff:ff:ff). When switch 1 receives this packet, the switch changes the broadcast address for the MAC address of client C and forwards it to the correct port. This is possible because the Faucet controller has a global vision of the network. In figure 11, we can see the protocols in the SDN enabled peering service.

When considering scalability on OpenFlow networks, the primary element to consider is the number of rules needed per switch and if the switch fabric can deal with them. In a peering service, the number of rules depends on the number of peering routers connected to the fabric. None of the current implementations of OpenFlow in peering services has more or the same amount of clients that NetherLight does. However, according to Bruyere [2], a system like we have described should scale without problems. In figure 12 Bruyere shows the average number of rules required for an OpenFlow switch with the number of clients present on different European exchange points. This amount of rules can be supported by most hardware. It can be derived from the figure that the proposed solution is usable with the current amount of clients in NetherLight, as well as scalable when new clients join.

[16] `https://faucet.nz/`

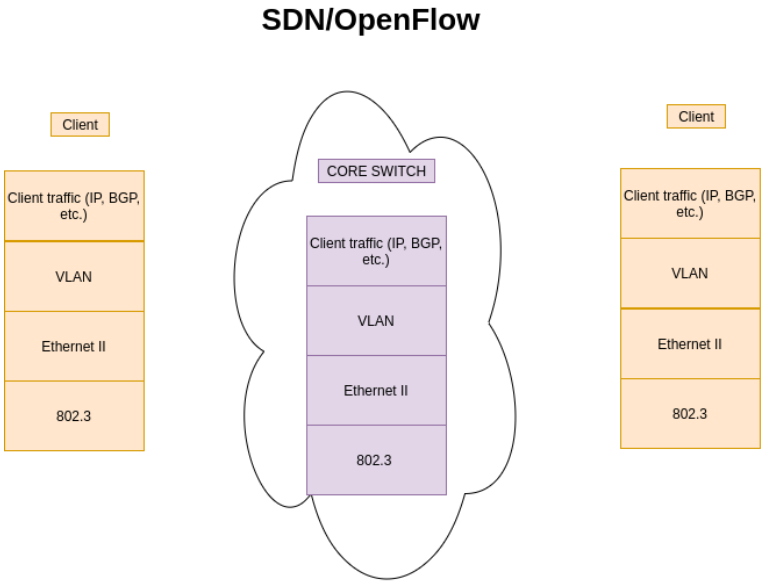

Figure 11: OpenFlow protocol stack

6.2.2 Monitoring and Management

For monitoring the described system, two main options can be implemented. The first option is Gauge. Faucet controls and exposes the forwarding switch state that monitors port and flows status via Prometheus so an open-source network monitoring system (NMS) (e.g. Grafana) can visualize it. Faucet can also be installed with another OpenFlow controller component, Gauge. Gauge uses OpenFlow to connect to the devices and monitors flow and port state. It can export into Prometheus, or InfluxDB or even a text log file. The complete list of metrics collected by Faucet and Gauge can be found in `https://docs.faucet.nz/_/downloads/en/stable/pdf/`. With this system, a visualisation tool needs to be installed to obtain a visual representation of the data collected by Gauge.

The second option, implemented as well in the EVPN solution, is sFlow. An OpenFlow enabled network is also compatible with sFlow technology if the hardware supports it (as major vendors do). An important benefit of sFlow is that it decouples monitoring from the control plane, so it does not use the controller or OpenFlow protocol to monitor the network. It is possible to change SDN controllers, routing and switching protocols, move between network operating systems, or build your control plane while maintaining the same level of visibility with sFlow. As mentioned in the EVPN solution, an sFlow collector and analyser are necessary to display all collected information.

The management of this network cannot be done using industrial solutions. Faucet has been implemented in production very few times and the Internet exchanges that use this controller have developed their own management tools. For the NetherLight peering service, two approaches can be considered. First, option is to integrate IXP Manager [17] with the Faucet controller. IXP Manager is an open-source full-stack management platform for Internet exchanges. A proof of concept of its integration with Faucet already exists in a virtualised environment, which according to their creators [18], will be made public soon (the timing is not clear yet). We were granted access to it as we wanted to have a clear view of its current development. Currently, this PoC works only for one specific topology, which we were able to recreate and add three clients. When adding the clients, IXP Manager allows to configure the IP and MAC of the client, and these information is added to the tables of the devices following the Faucet directives automatically.
IXP Manager is used for 92 IXPs around the world and has plenty of functionalities that NetherLight's peering service can use, e.g. provides a platform for clients to view their peerings, including managing route servers, looking glass features when using the BIRD daemon and more. The second option is that NetherLight develops its implementation of a management platform using Faucet's northbound API. This would fit perfectly to NetherLight requirements but with an extra effort associated with its development.

7 On- and off-boarding workflow

One of the requirements of NetherLight is the definition of the on-boarding and off-boarding processes for their clients. These processes include the steps to be taken, both from NetherLight and the client. The process of physically connecting to NetherLight

[17] `https://www.ixpmanager.org/`
[18] Marc Bruyere and Christoff Visser

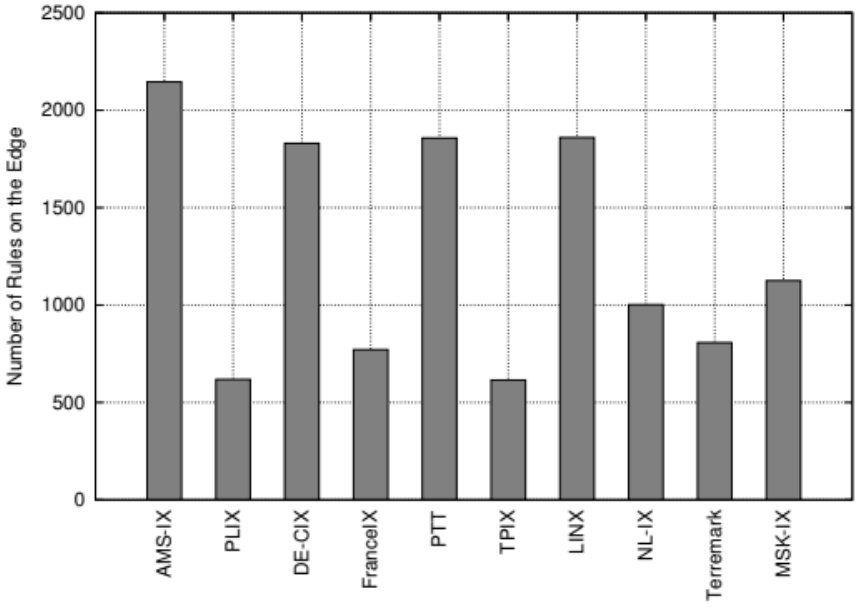

Figure 12: Average number of rules required. From Bruyere [2]

will not be described.

First, all the necessary information about the peering service should be available for the clients: locations, possible bandwidth, pricing, the presence or not of route servers and their available communities, allowed traffic on the peering service, on-boarding procedure, etc.

A client, when expressing his will to use NetherLight's peering service, should contact NetherLight with the location or locations he wants to connect. Also, he should communicate the desired bandwidth, the MAC address(es) that shall be used, ASN(s), contact information, and emergency technical contact. This can be done via email or using a web form. Supplying the clients with a standard template to fill in will ensure uniformity and completeness of the information.

When NetherLight receives the client's request to use the peering service, they should communicate to the client the VLAN tag necessary to access the peering service. If the client already has this tag in use, another one can be assigned, as in both solutions, EVPN and OpenFlow it can easily retag packets. Also, the client will be assigned public IP addresses (both IPv4 and IPv6) and instructions for properly configuring his connected port. Configuration templates would help the client to reduce configuration mistakes and avoid unwanted traffic on the peering service. These templates should be instantiated with client-specific information like IP addresses and VID.

For the EVPN solution, the client packets will be first directed to the quarantine EVI. After checking the configuration of the client, if it is correct, the client's traffic will be redirected to the main peering LAN. The changes between EVIs are transparent to the client. If the client has not configured the interface correctly, he will be asked to review and change the configuration until it is correct. For the SDN solution, a quarantine EVI or equivalent is not needed. After the client communicated the MAC address(es), the appropriate flow rules can be generated by the management tool. Once in the peering service, the client can peer with the route server to obtain BGP routes. Once this procedure is done, the client will be able to use the peering service.

In summary, the following information is required from a client to start the on-boarding:

- Desired bandwidth
- Location
- MAC Address(es)
- AS Number(s)

NetherLight will provide the following information to a client when on-boarding:

- VLAN ID
- IP address(es)
- ASN of the RS

- Configuration template

When a client communicates that it wants to quit using the peering service, its MAC address will be retired from ACLs and flow tables (for the SDN solution). The client will close his peering session with the route server, and his routes will not be announced any more. Then, his IP addresses will be free again to be reused by another client in the future. The client will be asked to stop using the peering service VLAN tag. Although it is not indispensable, NetherLight could announce that a client will not be using the service anymore to all other clients using a mailing list. This way, clients would know beforehand if any of their routes will be affected.

8 Comparison between solutions

A comparison of the solutions is given based on scalability, manageability, security, and implementation effort for NetherLight (NL). These metrics are taken directly from the requirements. The definition of the terms can be found in the requirements section 4. The plus and minus indicate how strongly they achieve those definitions from the requirements.

For scalability: that the solution can scale to at least 200 clients will mean that two pluses will be granted. If the solution is scalable between 100-200 clients it will receive a single plus. If the solutions is scalable from 50-100, it will receive a minus. And if the solutions is scalable to less then 50 clients it will receive a double minus.

For manageability to receive two pluses it needs to require minimal management effort from the NetherLight team once the solution is implemented. If some management effort is required from the NetherLight team to run the day to day operations the solution will receive a single plus. If a lot of effort is needed to run the day to day operation for the service it will receive a minus. And if it is a full-time job to manage the service then it will receive two minuses.

For security two pluses are given when clients are unable to perform prefix hijacks and successfully spoof MAC or IP addresses. If one of the spoofing actions if possible then the solutions will get one plus. If both spoofing actions are possible then one minus will granted. If both spoofing attacks and prefix hijacks are possible, two minuses are granted.

For ease of implementation for NetherLight two pluses are granted if no additional hardware is needed and no additional protocol needs to be implemented in the current infrastructure of NetherLight. If no additional hardware is needed but an additional protocol is introduced, then it will receive a single plus. If a significant change of the core network is needed e.g. switch from MPLS to IP, then it will receive a single minus. When new hardware is needed and a new protocol is added, the solution will receive two minuses.

	Scalability	Manageability	Security	Implementation (for NL)
MPLS-EVPN	++	+	++	+
VXLAN-EVPN	++	−	++	−
OpenFlow	++	++	++	−−

Table 1: Comparison of peering service solutions

8.1 MPLS-EVPN:

Scalability: ++
Does not have a hard limit on scaling as long as the hardware can cope with the demand. Hundreds of clients can be achieved. This is already proved by the current services at NetherLight for their current service. It can be used with a complex MPLS core network. Also, with multiple PoPs and a multi-node core network. All nodes in the provider network (NetherLight) need to support MPLS.

Manageability: +
When using an Orchestration & Automation platform, the service will be uniformly and easily manageable. Setup can be initially complex depending on the network size and creating the workflows in orchestration and automation systems. Users can join easily by configuring a VLAN, and NetherLight can put them in the peering EVI. Also, the ability to do traffic engineering or segment routing is one of the strengths of MPLS-EVPN. However, because the quarantine EVI is implemented in the solution, the manageability is only one plus for the reason that NetherLight needs to monitor the traffic in this EVI and contact the clients when unwanted traffic is sent by them. This leads to more management effort.

Security: ++
Integrated mechanisms to eliminate ARP storms by using ARP proxy and doing static MAC and IP address entries in the MAC-VRF. Loop prevention mechanisms by using a Designated Forwarder and Split Horizon for multihomed clients are implemented. Layer 2 ACLs need to be configured for dropping unwanted traffic and EtherTypes that are not allowed. When used

with a route server with RPKI and IRR, there is less chance of prefix hijacks.

Ease of implementation (For NL): +
MPLS and EVPN are already used therefore NetherLight only needs extra EVIs, and workflows in their Orchestration & Automation system for on-boarding and putting clients in quarantine. Also, RSs need to be created and setup. Overall, limited effort is needed from NetherLight for the implementation

8.2 VXLAN-EVPN:

Scalability: ++
VXLAN uses more bandwidth than MPLS when scaling up because it has more overhead. The nodes only need to support IP and thus the core can scale-out easier and cheaper than MPLS. VXLAN is capable of scaling out to hundreds of clients.

Manageability: -
VXLAN is easier to set up than MPLS and only requires support for the protocol on the PEs. No new management needed of the core except standard IP. However, it can not do traffic engineering or segment routing. For this solution, new workflows need to be created in the automation and orchestration systems to make sure a unified configuration is pushed to the devices of NetherLight. Because the quarantine EVI is implemented in the solution and the inability to do traffic engineering or segment routing is the manageability a minus. NetherLight needs to monitor the traffic in this EVI and contact the clients when unwanted traffic is sent by them. This leads to more management effort.

Security: ++
The security of this solution is the same as MPLS-EVPN. Integrated mechanisms to eliminate ARP storms by using ARP proxy and doing static MAC and IP address entries in the MAC-VRF. Loop prevention mechanisms by using a Designated Forwarder and Split Horizon for multihomed clients are implemented. Layer 2 ACLs need to be configured for dropping unwanted traffic and EtherTypes that are not allowed. When used with a route server with RPKI and IRR, there is less chance of prefix hijacks.

Ease of implementation (For NL): -
NL would need to switch over their core to IP. This can reduce cost because hardware only needs to support IP in the core and VXLAN on the edges. It is possible to switch over to IP for NL with current hardware. The only thing they will lose is segment routing, which is currently in use by NetherLight.

8.3 OpenFlow:

Scalability: ++
As seen in section 6.2 (*SDN*), scalability is not a problem because, theoretically, the number of flow rules should not increment beyond levels that are not supported by current hardware.

Manageability: ++
After integrating IXP Manager into Faucet management can be done easily. Flow rules are managed on the controller which distributes it to the switches. This makes it easy to configure the fabric uniformly. However, such integration needs to be done by NetherLight. A great benefit is once it is set up, there is no need to contact clients about unwanted traffic because OpenFlow can filter this. Thus less management effort is needed using this solution.

Security: ++
ARP storms are eliminated with Faucet implementation. ARP spoofing is not possible if MAC and IP addresses are statically configured. Switches will drop any traffic that does not match the rules. When combined with RS for prefix origin verification, prefix hijacking is mitigated.

Ease of implementation (For NL): –
Implementing the proposed SDN solution would require NetherLight to purchase, install and maintain new hardware. Also, IXP manager should be adapted to Faucet, and the faucet controller needs to be installed and setup.

9 Discussion

9.1 Generic services

The presence of route servers on the NetherLight peering service is not compulsory. However, because of the significant benefits and wide implementation on Internet exchanges [7], [10]–[12], we highly recommend to implement them regardless of the solution used. Like this, NetherLight can offer the clients the possibility to use the route server and to create their private bilateral peerings. Both options should be available in a peering service. Route servers should be implemented in a separate AS and connected to the peering service as any other client would be. This way, the clients can peer with the route servers as if they were creating a bilateral connection with any other client. Also, for redundancy, two route servers should be deployed with

their configurations and tables mirrored.

We recommend BIRD because of its wide implementation on IXPs and for its powerful filtering abilities. Applying fixed filtering and allowing the clients to use the defined BGP communities for setting their filter for the prefixes they announce and receive is binding if a route server is implemented. Otherwise, the route server could announce incorrect routes that would affect all clients. We recommend to implement, at least, the filtering and communities described in section 5.1.1 (*Filtering and Communities*) for ensuring a trustworthy service to the clients. SURFnet already has a system for performing RPKI validation. We do not see any reason why NetherLight could not use this existing service. However, if the access to it is not possible, there are multiple options that NetherLight could consider: Cloudflare's OctoRPKI [19], NLnetLabs Routinator [20], or RIPE NCC RPKI Validator 3 [21].

We also recommend NetherLight to use public IP addresses and at least a /23 subnet for IPv4 and /64 for IPv6 addresses, considering their scalability requirements. Also, to assign the addresses statically and allow just one IPv4 address and one IPv6 address per interface. All the security measures listed in section 5.3 *Security*, should be followed, as well as, comply with MANRS actions described in section 3.2 *Best practices on peering services*.

9.2 EVPN

There are two underlays for EVPN available, VXLAN and MPLS. We currently recommend NetherLight to use MPLS-EVPN because of the existence of the current MPLS core network. This will result in minimal extra setup and new knowledge that needs to be acquired by the NetherLight team and is, therefore, most feasible. Also, changing the core network to VXLAN would not gain any added value, even though it is possible with the current hardware. For new implementations that either have no network yet or an IP only network, VXLAN-EVPN is a very suitable option. In addition to using MPLS-EVPN, we advise using the current Cisco NSO system to do the configuration of the network devices. Workflows and configurations should be added so that a new peering service EVI is created and a quarantine EVI for checking the clients' configurations.

9.3 SDN/OpenFlow

SDN represents the most ground-breaking of both solutions: it requires the implementation of a new physical network and changes on the technology currently used by NetherLight. Faucet offers a list of supported hardware [38]. We would suggest considering the Cisco Catalyst 9000 Series switches with Cisco IOS XE 16.12.1c or later, as they support multi-table, large bandwidth and have a large amount of ports [40]. Other devices also have these characteristics, like Junipers MX10K, but they have not been tested with Faucet. It would be NetherLight's task to test if they are fully compatible.
Monitoring depends on the hardware, as not all devices support sFlow. However, we would still recommend the usage of Gauge. sFlow is based on sampling rates, and with Gauge, a complete picture of the network can be obtained. On management, we would recommend adapting the IXP Manager as there is no necessity of *reinventing the wheel* and it would entail an excellent improvement for the Faucet community, as a complete IXP management tool would become available.

This solution would provide significant benefits for NetherLight, as described in section 6.2. Common problems in peering services like ARP storms would be solved, as well as, any unwanted traffic would be rapidly dropped. However, it also presents some drawbacks. First of all, although it is estimated that large peering services are supported, no test in a real environment has been performed. NetherLight should perform this test. Also, setting up a management system would require investing resources into adapting IXP Manager. Furthermore, new hardware needs to be bought and installed. NetherLight will have to maintain two infrastructures, the current one, that provides the point-to-point and multipoint service and the peering service or study how to implement the former using OpenFlow.

We can say that currently an OpenFlow based peering service, although it is possible to set up, requires a significant effort to implement. However, because of less management effort, fine-grained control of traffic, and vendor independence, we think that it would be an excellent solution for NetherLight.

9.4 Requirements

Next, we discuss how the proposed solutions comply with the defined requirements.

A detailed explanation of what the service is and how it works.
In section 6.1.2 (*EVPN implementation*), we described how the peering service could be implemented using EVPN, as well as how to avoid unwanted traffic and broadcast problems. We also described its management and monitoring possibilities. In section 6.2 (*SDN*), we presented how the peering service can be implemented using the SDN paradigm, we propose a network topology and the elements needed for the solution. We explain how it deals with unwanted traffic and broadcast storms. Also,

[19] `https://github.com/cloudflare/cfrpki`
[20] `https://nlnetlabs.nl/projects/rpki/routinator/`
[21] `https://www.ripe.net/manage-ips-and-asns/resource-management/certification/tools-and-resources`

we explain its scalability capabilities and monitoring and managing possibilities. The generic services needed for both solutions are described in section 5 (*Generic Services*).

Clients set up BGP peerings with each other.
Both solutions create a layer 2 domain that allows clients to exchange BGP traffic over it. Clients can choose to either do bilateral peering or multilateral peering. Multilateral peering is enabled by having route servers in the network.

Clients connect to the service using either one or two VLAN tags towards NetherLight.
In both solutions, one or several VLANs can be created, and clients can connect to the VLAN using one tag. Also, if the client already has the VLAN tag in use, a different tag can be assigned to him, and NetherLight can re-tag his traffic to use the peering VLAN.

There is a clear on-boarding process.
The on-boarding process has been described in section 7 (*On- and off-boarding workflow*).

The service needs to be well-manageable.
For both solutions, management options have been presented. However, IXP Manager for SDN needs more development.

The service is secure and scalable.
Route servers verify the announced routes from clients with RPKI. This prevents prefix hijacking. Therefore we recommend all clients to use the route server, as in bilateral peerings NetherLight can not filter the prefixes. In both solutions, the MAC and IP addresses are statically configured, and rules have been implemented to prevent spoofing of clients. Therefore ARP spoofing is not possible.
We also suggest implementing a quarantine EVI for the EVPN solution to check a client's traffic first before it is put in the production peering service. Here the traffic can be checked for unwanted traffic like link-local protocols other than ARP and IPv6 ND. This should be reported back to the client, and if the traffic is clear, the client can be switched over to the peering service EVI. EVPN with both VXLAN and MPLS as underlay can scale to large, complex networks and hundreds of clients.
The SDN solution is secure, thanks to flow tables, any unwanted traffic is dropped. It also has been proven scalable, theoretically. However, more testing is needed before being used in a production environment.

At least one of the solutions from this research work can be implemented on the current NetherLight hardware (Juniper MX2008).
The EVPN solutions as described in section 6.1 can be implemented in on NetherLight's current hardware platform.

9.5 On- and off-boarding workflow

We suggest to NetherLight to have a public website where clients can consult their locations, possible bandwidth options, the presence/or not of route servers, which services offer the route servers (filtering, looking glass, etc.). Also, an explanation of the desired and forbidden traffic inside the peering service. Likewise, we would recommend NetherLight to develop a web form so clients can upload their information in a standard way. This web form can be on the web with all the info about the service. Also, this communication could be done via email.

9.6 Research questions

Finally, we end our discussion by answering our research question and sub-questions. We have proposed EVPN and OpenFlow as solutions that can facilitate a peering service for NetherLight. Both have been proven secure and manageable. They have been explained on section 6 (*Solutions*). For doing so, we have gathered the requirements of the NetherLight team, which we have exposed in section 4, (*Requirements*). These requirements include a description of how the proposed solutions should be adapted to NetherLight needs. We have also investigated the best practices and options for setting up a peering service, which can be found in section 3.2, (*Best practices on peering services*) and generic components in section 5, (*Generic Components*). The most common problems when setting a peering service have been identified and considered. We have advised NetherLight to use route servers as an extra service for their clients and filter routes to avoid the propagation of wrong routing information. Also, we listed the Ethernet types that should be allowed on the service, as well as, proposed solutions that avoid ARP storms. The IP space to be used has also been contemplated. How the protocols behave in the environment is described in the descriptions of the solutions in sections 6.1, (*EVPN*) and 6.2 (*SDN/OpenFlow*). We explained how EVPN works and proposed a quarantine VLAN to achieve a secure and uniform service. On the other side, Faucet has been proposed as the controller for the OpenFlow solution, programmed using the Umbrella rule set. For both solutions, we gave options on monitoring and manageability, as well as considered their scalability. Finally, we explained an on- and off-boarding procedure applicable to both solutions that will allow a uniform procedure and configurations for all clients, exposed in section 7, (*On- and off-boarding workflow*).

10 Future Work

As for future work, we propose the following:

- **First (small) implementation of MPLS-EVPN solution.** Because MPLS-EVPN is already present in the current NetherLight topology, a logical next step would be to implement the proposed MPLS-EVPN solution.
- **PoC of the OpenFlow solution.** Another logical next step is to create a PoC for the OpenFlow solution to test how the solution behaves and corroborate if it scales up to the number of clients expected by NetherLight.
- **Research the ability to use Umbrella rule set in other OpenFlow controllers.** Umbrella has only been implemented in production using Faucet. It would be interesting to determine if it is viable to use it on other controllers. This way, it would be possible to consider the advantages of disadvantages of different controllers when selecting one, specifically if a peering service is not the only service in the network. Also, it would be useful if an already existing OpenFlow network with a controller other than Faucet wanted to implement a peering service using Umbrella and its controller.

11 Conclusion

In this project, we have investigated, proposed, and compared several solutions that allow the creation of a state-of-the-art, manageable, scalable, and secure peering service for NetherLight. The requirements of NetherLight have been taken into account during the entire process.

NetherLight needs to consider different factors when building this service. A route server can be implemented to provide an extra service to the clients and avoid wrong BGP routing information and prefix hijacking. Also, the allowed protocols on the network have been considered and have been listed. ARP is one of the necessary protocols on the peering service, and the possibility of ARP storms has been considered and mitigated. The required IP address reservations are also provided

EVPN and OpenFlow solutions have been proposed to the NetherLight team. Right now, NetherLight can best create a peering service by adopting the first solution, MPLS-EVPN. It can be implemented using the current hardware, and management solution. It represents the option with the least effort.
OpenFlow is a more advanced solution because it offers less management effort, fine-grained control of traffic, and vendor independence. Implementing this solution would require more effort than implementing the MPLS-EVPN solution.
So in conclusion, we have researched and compared two solutions for creating a peering service for NetherLight with taking the best practices and pitfalls into account and can conclude that both options are viable, however, MPLS-EVPN requires less effort to implement, OpenFlow offers better manageability and vendor independency.

References

[1] Hill, A. van den, Vos, M. de, and Malenstein, G. van, *Netherlight*, 2016. [Online]. Available: https://meetings.internet2.edu/media/medialibrary/2016/05/17/20160517-Malenstein-NetherLight.pdf.

[2] Bruyere, M., "An outright open source approach for simple and pragmatic internet exchange", PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2016.

[3] MANRS, *About manrs*, Sep. 2019. [Online]. Available: https://www.manrs.org/about/.

[4] Jasinska, E., Hilliard, N., Raszuk, R., and Bakker, N., "Internet Exchange BGP Route Server", RFC Editor, RFC 7947, Sep. 2016, pp. 1–12. [Online]. Available: https://tools.ietf.org/html/rfc7947.

[5] Richter, P., Smaragdakis, G., Feldmann, A., Chatzis, N., Boettger, J., and Willinger, W., "Peering at peerings: On the role of ixp route servers", in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14, Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 31–44, ISBN: 9781450332132. DOI: 10.1145/2663716.2663757. [Online]. Available: https://doi.org/10.1145/2663716.2663757.

[6] Hilliard, N., Jasinka, E., Raszuk, R., and Bakker, N., "Internet Exchange BGP Route Server Operations", RFC Editor, RFC 7948, Sep. 2016, pp. 1–15. [Online]. Available: https://tools.ietf.org/html/rfc7948.

[7] Fenioux, A., *Route servers, features and security*, 2016. [Online]. Available: https://ripe72.ripe.net/presentations/97-RIPE72_05-16.pdf.

[8] euro-ix, *Ixp services dmz*, 2020. [Online]. Available: https://www.euro-ix.net/en/forixps/set-ixp/ixp-bcops/ixp-services-dmz/.

[9] Ondrej Filip. (Feb. 27, 2018). BIRD Routing Daemon, APRICOT 2018, [Online]. Available: https://www.slideshare.net/apnic/bird-routing-daemon.

[10] AMS-IX, *Ix route servers: Ams-ix amsterdam*, 2020. [Online]. Available: https://www.ams-ix.net/ams/documentation/ams-ix-route-servers.

[11] INEX, *Route servers*, 2020. [Online]. Available: https://www.inex.ie/technical/route-servers/.

[12] DE-CIX, *Cix madrid route server guide*, 2020. [Online]. Available: https://www.de-cix.net/en/locations/spain/madrid/routeserver-guide.

[13] Rekhter, Y., Moskowitz, R. G., Karrenberg, D., and Groot, G. J. de, " Address Allocation for Private Internets", RFC Editor, RFC 1918, Feb. 1996, pp. 1–9. [Online]. Available: https://tools.ietf.org/html/rfc1918.

[14] CYMRU, T., *Bogons via http*, 2020. [Online]. Available: https://team-cymru.com/community-services/bogon-reference/bogon-reference-http/.

[15] Durand, J., Pepelnjak, I., and Doering, G., " BGP Operations and Security", RFC Editor, RFC 7454, Feb. 2015, pp. 1–26. [Online]. Available: https://tools.ietf.org/html/rfc7454.

[16] Traina, P., Chandrasekeran, R., and Li, T., "BGP Communities Attribute", RFC Editor, RFC 1997, Aug. 1996, pp. 1–5. [Online]. Available: https://tools.ietf.org/html/rfc1997.

[17] Sangli, S. R., Tappan, D., and Rekhter, Y., "BGP Extended Communities Attribute", RFC Editor, RFC 4360, Feb. 2006, pp. 1–12. [Online]. Available: https://tools.ietf.org/html/rfc4360.

[18] Heitz, J., Snijders, J., Patel, K., Bagdonas, I., and Hilliard, N., "BGP Large Communities Attribute", RFC Editor, RFC 8092, Feb. 2017, pp. 1–. [Online]. Available: https://tools.ietf.org/html/rfc8092.

[19] Velde, G. V. de, Popoviciu, C., Chown, T., and Hahn, C., "IPv6 Unicast Address Assignment Considerations", RFC Editor, RFC 5375, Dec. 2008, pp. 1–34. [Online]. Available: https://tools.ietf.org/html/rfc5375.

[20] Gagliano, R., "IPv6 in IXPs", RFC Editor, RFC 5375, Aug. 2010, pp. 1–34. [Online]. Available: https://tools.ietf.org/html/rfc5963.

[21] Juniper, *Configuring firewall filters*, Mar. 2020. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/firewall-filter-qfx-series-cli.html.

[22] Sajassi, A., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and Henderickx, W., "BGP MPLS-Based Ethernet VPN", RFC Editor, RFC 7432, Feb. 2015, pp. 1–56. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7432.pdf.

[23] Quoitin, B., Pelsser, C., Swinnen, L., Bonaventure, O., and Uhlig, S., "Interdomain traffic engineering with bgp", *Comm. Mag.*, vol. 41, no. 5, pp. 122–128, May 2003, ISSN: 0163-6804. DOI: 10.1109/MCOM.2003.1200112. [Online]. Available: https://doi.org/10.1109/MCOM.2003.1200112.

[24] Li, T. and Conta, A., "MPLS Label Stack Encoding", RFC Editor, RFC 3032, Jan. 2001, pp. 1–23. [Online]. Available: https://tools.ietf.org/html/rfc3032.

[25] Iveson, S., *Tcp/ip over vxlan bandwidth overheads*, Mar. 2014. [Online]. Available: https://packetpushers.net/vxlan-udp-ip-ethernet-bandwidth-overheads/.

[26] Khatri, P., *Evpn. a tutorial*, [Online; accessed June 12, 2020. Slide 16], 2019. [Online]. Available: https://www.sanog.org/resources/sanog33/SANOG33_Tutorials-EVPN_Tutorial-Paresh_Khatri.pdf.

[27] Cisco, *Network services orchestrator network element drivers*, Jun. 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/network-services-orchestrator/datasheet-c78-734669.html.

[28] Zhu, A., *Sflow vs netflow vs snmp: What are the differences?*, Jul. 2018. [Online]. Available: http://www.cables-solutions.com/sflow-vs-netflow-vs-snmp-differences.html.

[29] sFlow, *About sflow*, 2020. [Online]. Available: https://sflow.org/about/index.php.

[30] (ONF), O. N. F., *Software-defined networking (sdn) definition*. [Online]. Available: https://www.opennetworking.org/sdn-definition/.

[31] Foundation, O. N., *Onf sdn evolution*, Sep. 2016.

[32] Bruyere, M., Antichi, G., Fernandes, E. L., Lapeyrade, R., Uhlig, S., Owezarski, P., Moore, A. W., and Castro, I., "Rethinking ixps' architecture in the age of sdn", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2667–2674, 2018.

[33] Stringer, J., Pemberton, D., Fu, Q., Lorier, C., Nelson, R., Bailey, J., Correa, C., and Esteve Rothenberg, C., "Cardigan: Sdn distributed routing fabric going live at an internet exchange", Jun. 2014, pp. 1–7. DOI: 10.1109/ISCC.2014.6912501.

[34] Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E., "Sdx: A software defined internet exchange", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2014.

[35] Gupta, A., MacDavid, R., Birkner, R., Canini, M., Feamster, N., Rexford, J., and Vanbever, L., "An industrial-scale software defined internet exchange point", in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 1–14.

[36] Hermans, S. and Schutrup, J., "On the feasibility of converting ams-ix to an industrial-scale software defined internet exchange point university of amsterdam system and network engineering", 2016.

[37] Antichi, G., Castro, I., Chiesa, M., Fernandes, E. L., Lapeyrade, R., Kopp, D., Han, J. H., Bruyere, M., Dietzel, C., Gusat, M., Moore, A. W., Owezarski, P., Uhlig, S., and Canini, M., "Endeavour: A scalable sdn architecture for real-world ixps", *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2553–2562, 2017.

[38] Faucet, *Vendor-specific documentation*. [Online]. Available: https://docs.faucet.nz/en/latest/vendors/index.html.

[39] Foundation, O. N., *OpenFlow Switch Specification*. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.2.pdf.

[40] Cisco, *Programmability Configuration Guide, Cisco IOS XE Gibraltar 16.12.x, OpenFlow chapter*. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1612/b_1612_programmability_cg/openflow.html.

Appendices

Appendix A: Contacted IXPs

1. AAIX. Klagenfurt, Austria. No web page available, information can be found at https://www.peeringdb.com/ix/377
2. AMS-IX. Amsterdam, The Netherlands. https://www.ams-ix.net/ams
3. BCIX - Berlin, Germany. https://www.bcix.de/bcix/
4. BNIX - Several locations, Belgium. https://www.bnix.net/
5. BREM-IX - Bremen, Germany. https://www.brem-ix.net/
6. CATNIX - Barcelona, Spain. https://www.catnix.net/
7. CIXP - CERN, Geneva, Switzerland. https://cixp.net/
8. DIX - Several locations, Denmark. https://dix.dk/
9. DE-CIX - Several locations. https://www.de-cix.net/
10. DO-IX - Dortmund, Germany. https://www.do-ix.net/locations/
11. ECIX - Several locations. https://www.ecix.net/
12. FICIX - Helsinki, Finland. https://www.ficix.fi/
13. France-IX - Several locations, France. https://www.franceix.net/en/
14. GE-CIX - Several locations. https://ge-cix.net/get-connected/location/
15. GRAX - Graz, Austria. https://www.grax.at/
16. GrenoblIX - Grenoble, France. https://www.rezopole.net/fr/ixp/grenoblix
17. IXLeeds - Leeds, UK. https://ixleeds.net/
18. LINX - London, UK. https://www.linx.net/
19. LyonIX - Lyon, France. https://www.rezopole.net/fr/ixp/lyonix
20. NDIX - Enschede, The Netherlands. https://www.ndix.net/
21. NetIX- Sofia, Bulgaria. https://www.netix.net/
22. NL-ix - The Netherlands. https://www.nl-ix.net/
23. LILLIX - Lille, France. https://www.lillix.fr/
24. Rheintal IX - Schaan, Liechtenstein. https://www.rheintal-ix.net/
25. SAIX - Salzburg, Austria. https://www.saix.at/
26. STHIX - Stockholm, Sweden. https://www.sthix.net/
27. SwissIX - Several locations, Switzerland https://www.swissix.ch/
28. TouIX - Toulouse, France. http://touix.net/en
29. TREX Regional Exchanges - Tampere, Finland. http://www.trex.fi/
30. VIX - Vienna, Austria https://www.vix.at/

Appendix B: Questions to IXPs

- What technique do you use to facilitate a L2 domain(LAN, VLAN, VXLAN, EVPN, anything else?)
- What IP addresses do you use inside the service (public or private. Ranges from clients or own?)
- What are the pitfalls we should take into account when creating an IP peering network? (ARP storm/spoofing, etc.)
- How do you make sure that the infrastructure is scalable? (Do you use route servers? When are they necessary?)
- We were thinking than with ACL it is possible to control what a client introduces to the network, why then is the quarantine VLAN needed?
- How do you control the BGP config and announcements of the clients, do you have one or several route servers on the quarantine VLAN or filtering is only done on the main route server on the production LAN?
- How do you limit the traffic types that a client introduces to the peering LAN?
- We were also wondering We thought that it could be done using ACL, but we found out that some IXP are using quarantine VLANS. What system you use and why? Are BGP announcements checked? Also, do you offer route servers to your clients?

Appendix C: Answers from the IXPs

AAIX:

Hi Mar and Arnold,

The AAIX is just a tiny regional IXP – based on Ethernet-VLAN – nothing spectacular here ...
Route servers are recommended when you have a number higher than 5 members ...
For IP ask RIPE for a dedicated IP range for IXP usage – the have some reserved for that purpose ...
on our AAIX side this is easy – we are a community driven and no-cost ixp –
limitation is to be avoided at each case
we try to push all members to go for 2 x 10 or 4 x 10 GE ports to never hit any limits – and: there is design wise no shaping feature existing – maximum speed without any blocking is the target
route server: is on the todo list – if configured then it will have (planned) proper filtering on irr and rpki base ...
layer2 filter systems are just rudimentary – we all (=members) know each other and all configurations are known by one of the admins – but improvement on that is on the todo list as well ...

AMS-IX:

Dear Mar, Arnold,

We use MPLS/VPLS. Although if we would build the platform from scratch right now we would use EVPN over MPLS and we are considering migrating to this. Btw MPLS and not VXlan because of the better possibilities to do traffic engineering.
Also, Public IPv4 and IPv6 address space.
ARP storm and spoofing are important, but the most important to take care of is loop prevention. You don't want a situation where a member creates a L2 loop next to the Peering platform. Typically this is prevented by limiting the MAC addresses on a member port to 1.
We scale by adding enough hardware :) and by using scalable protocols like MPLS as the underlying technology. Route servers do not help scale the peering platform. They help in scaling the management overhead of BGP sessions between peers. If you want to peer with all participants on an IXP (N) you need to establish N-1 BGP sessions. This can become a management overhead for the manager of a peering router. This is where a route servers helps.

CATNIX:

Hello Mar and Arnold,

The interconnection service is common to all CATNIX participants. It is based on the traffic exchange (peering) between them without restrictions and under a principle of free agreement. The service allows access to CATNIX connected networks with a single physical connection (or multiple added ports) to the infrastructure of level 2 based on Ethernet technology.
CATNIX assigns a public range of IP address
There is traffic that can be dangerous for CATNIX and its members. It is for this reason that CATNIX monitors the fulfilment of several conditions by members.
Ports between clients and CATNIX switches must be in access mode (trunked mode will not be allowed). Only one known MAC address per port (port security) is allowed. LLDP protocol is not accepted. BPDU protocol is not accepted. Multicast storm control is configured in the switch ports.
CATNIX offers two route servers to facilitate the exchange of routes to the IXP. Route servers simplify the exchange of routing information among members and speed up the interconnection process for new members.
Through a single BGP (Border Gateway Protocol) session with route servers, routes can be exchanged with all participating members, eliminating the need to establish direct peerings among members. Furthermore, it is also possible to use sessions with route servers as redundancy mechanisms for bilateral peering.
The Autonomous System (AS) of the CATNIX route servers is AS60082.
Peering sessions with route servers can be signalled by communities for each participant. The list of BGP communities allows CATNIX members to control the redistribution of their advertisements or to add content to their prefix routes.

Best regards

Rheintal IX:

Our Peering LAN is a stretched VLAN over three locations with public IP addresses from our assigned PI address range.
We limit the number of MAC addresses to one per port and fix the allowed MAC address to avoid spoofing. The number of our peers is low (around 20), so we do not have to deal with ARP storms.
We use route servers. But this is more a service to our peers, so that they don't need to have too many peering sessions.
To keep the infrastructure scalable, we have plans to switch from our spanning tree based L2 solution (where the control plane is based on flooding) to EVPN with a L3 underlay and a BGP based control plane, but not sure if we will put VXLAN, SRv6

or SR-MPLS on top.

TREX Regional Exchanges:

Hello,

We're using a simple VLAN + RSTP L2 domain right now, but we are planning to move to VXLAN or EVPN in the future.
We have public address space from RIPE NCC. They have special procedures for assigning address space to IXPs. Our IPv4 block pre-dates the special procedures, but our IPv6 block is from 2001:7f8::/32, which is the European IXP address block. The European IXP block for IPv4 going forward is 185.1.0.0/16.
There are lots of different threats, which mostly boil down to members 'clients' misconfigurations.
One common approach to securing the peering network (aka shared medium) is to not accept any unnecessary link protocols from member ports. These include all Spanning Tree Protocol variants, routing protocols (other than BGP), link test protocols (LLDP, CDP etc) and others (DHCP, GARP, GVRP, VTP, ...)
Another common measure is to restrict member ports to one MAC address. Sometimes members can accidentally loop the shared medium back to the shared medium, i.e. they echo all packets from the IXP back to the IXP, creating an endless storm of packets. If the member is only allowed to send packets with their own source MAC address, this will filter out all (or at least most) echoed packets. This also prevents certain kinds of spoofing.
Using default storm control measures provided by switch manufacturers will also help, but sometimes you can write better filters yourself.
The scalability of the core infrastructure is more to do with available link bandwidth. Monitoring network usage and planning upgrades before problems arise is relatively easy.
Route Servers are more of a value added service to make the lives of member networks easier. They are really necessary only after the IXP has over a hundred members, but introducing them at that point is difficult: Why would they configure peering with the route-server(s), if they already have sessions up with everyone they want to peer with?
I can be of some help. Firstly I would like to direct you to a much bigger resource of IXP information online: `https://www.euro-ix.net/` (also ixpdb and peeringdb)
Secondly, if you were to create an IXP in Amsterdam, your biggest challenges would be non-technical in nature. I can name at least four existing IXPs that already operate there: AMS-IX, Equinix, Asteroid and NL-IX. One of them is a contender for the world's biggest IXP and many of them have hundreds of connected members.
It is very difficult to create a new IXP next to an existing one, unless the existing one is bad somehow: They might be too greedy, or have constant technical problems, or they may have violated their members' trust.
The question potential members will ask is: "what good will connecting to your new IXP do to us, when we could connect to an existing one instead?" Peering with three members at one IXP is not as useful as peering with three hundred members at another. I call this the critical mass problem.
Even if connecting to the new IXP was free, the new member needs to pay for circuits within the city and a port in their router, bringing the total cost of connection to thousands or tens of thousands of Euros, depending on port speed. In fact, even the bigger IXPs sometimes struggle because economically it's cheaper to buy transit than to connect to the IXP.
Creating a new IXP far from existing IXPs may be easier. There are still a couple dozen third world countries that don't have an IXP at all. But even there they have non-technical challenges: There will be no existing culture of peering. Existing ISPs are used to just buying transit, and do not understand why they should exchange traffic with their competitors for free.
Third world countries tend to also have regulatory challenges: The law protects some existing monopoly, or all ISPs need some draconian license to operate a network. Electric power may be in short supply as well.

Appendix D: Requirements

R: A detailed explanation of what the service is and how it works.
The ultimate goal for NetherLight is to have a service in production that allows NetherLight customers to setup peerings with each other in a fast, manageable, predictable, scalable and reliable way. Preferably with minimal or no intervention from NetherLigth management/operations.

R: Customers setup BGP peerings with each other.

R: Customers connect to the service using either one or two VLAN tags towards NetherLight.
This is a must for the current implementation, at least for one scenario, if other scenarios are possible this requirement could be discarded.

R: There is a clear onboarding process:

- what does a customer do when starting to use this service?
- which parameters gets the customer from NetherLight?
- what is the policy for using the service, must be clearly stated and communicated.(And similarly there is a clear decommisioning process when customers stop using the service.)

R: A well-manageable service.

- customers are using the service in a uniform (specified!) way
- eliminating unwanted behaviour/configuration errors by customers as much as possible.
- as less strictly unnecessary (or additional) operational steps and procedures at NetherLight.
- the ability to check whether a customer is ready for using the service.

R: The service is secure and scalable.

R: At least one of the solutions from this research work can be implemented on the current NetherLight hardware (Juniper MX2008).

Other information: Number of clients: about 70 currently. Clients can have several peerings with separate other clients.