OpenFlow Based Multi-Domain VPN Prototype Architecture

Ronald van der Pol*, Marijke Kaat*,
Bart Gijsen†, Piotr Zuraniewski†
and Daniel Cabaca Romão‡

*SURFnet
Radboudkwartier 273
3511 CK Utrecht, The Netherlands
Email: ronald.vanderpol@surfnet.nl, marijke.kaat@surfnet.nl

†TNO
Brasserplein 2
2612 CT Delft, The Netherlands
Email: bart.gijsen@tno.nl, piotr.zuraniewski@tno.nl

‡University of Amsterdam
Science Park 904
1098 XH Amsterdam, The Netherlands
Email: d.f.romao@uva.nl

***Abstract*—This paper describes the architecture of a prototype for an OpenFlow based multi-domain VPN service, which is built in the Community Connection (CoCo) project. The prototype that is being developed will let end-users set up CoCo instances (VPNs) via an easy to use web portal, without needing the help of network administrators to do manual configurations of the network switches. The CoCo prototype handles the automatic setup and tear down of the instances by configuring the OpenFlow switches. Typical users are research communities that form a closed user group and want their e-science resources (servers, VMs, laptops, storage, instruments, etc.) interconnected, but reachable for their closed group only.**

I. Introduction

The CoCo project is one of the *open calls* funded by the European GN3plus project. It runs from October 2013 until March 2015. The prototype that is being built in the CoCo project consists of several parts. The data plane (forwarding plane) consists of OpenFlow switches. The switches in a domain are controlled by the OpenDaylight SDN controller. Each domain runs its own OpenDaylight controller. CoCo agents are an extension to the OpenDaylight controller and add specific CoCo functionality to the OpenDaylight controller.

The core of the network is based on MPLS label forwarding and is implemented over several domains. MPLS is only used as encapsulation. The forwarding paths are calculated by and provisioned from the OpenDaylight controllers. As such, we use no data plane label distribution protocols such as the Label Distribution Protocol (LDP [5]).

The CoCo agents use the BGP protocol to exchange VPN and end point information with each other. BGP is only used for exchanging information. BGP does not do any forwarding (FIB) manipulations. The forwarding tables in the OpenFlow switches are controlled via OpenDaylight. The first CoCo prototype supports L3 VPNs only. At a later stage the CoCo prototype can be extended to L2 VPN functionality.

II. CoCo Use Cases

The advent of Software Defined Networking is creating innovation opportunities for a wide range of use cases. The CoCo service will enable scientists from multiple organisations to dynamically create virtual, private networks for sharing services and facilities as if they were collaborating within a single network environment.

A particular use case demonstrating this innovative power of the CoCo service is the *DNA sequencer as a Service*. DNA sequencers are increasingly important instruments for scientists in the genomics science field. These sequencer instruments and the specific bioinformatics solutions required for the storage, processing and transport of their output are very expensive and get outdated relatively quickly, due to the current rapid developments. Therefore, research organizations can only justify such investments if the (re-)utilization of the sequencers and bioinformatics solutions is sufficiently high. The opportunity to strongly improve this return on investment by offering DNA sequencing as a Service, that can be consumed by scientists from multiple institutes, has been identified as a key innovation in the genomics research field.

Figure 1 presents an overview of a technical solution for a DNA sequencer as a Service. Currently, the automation of

DNA sequencing and processing is increasingly being applied via workflow management solutions, such as the Galaxy platform [6]. At their back-end such platforms will manage and interface to the resources that execute these processes. These resources include storage, processing and network connectivity. The CoCo service is a good candidate for providing management services for the connectivity resources. In particular, the CoCo service will provide the on-demand, multi-domain connectivity between the resources. CoCo will contribute to the ease-of-use by relieving the genomics scientists from the need to login on multiple, separate systems. Moreover, the CoCo service is being designed to be incorporated in future integrated resource management solutions.

III. CoCo Overall Architecture

The CoCo prototype will consist of multiple domains. Each domain represents an NREN (National Research and Education Network) . The implementation will use several testbeds (e.g. the SURFnet OpenFlow testbed and the GÉANT OpenFlow testbed), each representing a separate domain. Figure 1 shows the inter-domain architecture of CoCo. Each domain has an OpenFlow based infrastructure and each domain runs its own CoCo agent. These CoCo agents are extensions of the OpenDaylight SDN controller. The OpenFlow infrastructure consists of OpenFlow switches that have either a Provider Edge (PE) function or a Provider (P) function. The PE switches connect to either Customer Equipment (CE) via a UNI interface or to PE switches in other domains via a E-NNI interface.

The CoCo agents are responsible for topology discovery within a domain and do intra- and inter-domain path calculation. The intra-domain path calculation is based on the domain topology only. The inter-domain path calculation is based on BGP path information that is exchanged with neighbouring domains. In the inter-domain case each CoCo agent configures forwarding entries on the OpenFlow switches in its own domain that form a path between two PEs in that domain. This can be either two PEs directly connected to the source and destination CE, or it can be a PE that is connected to the E-NNI port to the inter-domain link that has been chosen by the BGP path selection process towards a CE in another domain. For simplicity, we start with one shortest path between each pair of PEs. At a later stage we can easily extend it with backup paths.

There is one centralised web portal for the CoCo service. End users login to this portal and they can then setup or tear down CoCo instances. CoCo instances are set up by choosing end sites from a list and entering prefix and port based VLAN information for each site. End users can join multiple CoCo instances simultaneously. The web portal distributes the prefix and VLAN information to the CoCo agents in the various domains.

IV. CoCo Data Plane Forwarding

The CoCo network core consists of Provider Edge (PE) and Provider (P) OpenFlow switches as shown in Figure 2. The P switches are internal core network switches. The PE switches connect to either end-sites (via UNI interfaces) or other domains (via E-NNI interfaces).

MPLS based forwarding is used in the core of the network in order to keep the forwarding tables small by aggregating all IP prefixes that are behind a PE OpenFlow switch. Two MPLS labels are used. The outer MPLS label is used to identify the PE to which a packet must be sent. The inner MPLS label is used to identify the CoCo instance (VPN). PE switches take care of encapsulating the user traffic received from Customer Edge (CE) equipment with the proper MPLS labels. When sending traffic from the backbone to the CE the PE removes the MPLS labels.

V. Encapsulation and decapsulation at the UNI interfaces

At the edges of the domain on the UNI interfaces towards the CE of the customer encapsulation and decapsulation takes place. We use VLAN based port services on the UNI interfaces. This means that traffic between CE and PE is VLAN tagged and the VLAN ID maps to a particular CoCo instance. The customer is responsible for putting traffic of nodes in the correct VLAN. The PE switches match on the VLAN ID and pop the VLAN header. Before forwarding the packet the PE switches add the two MPLS labels corresponding to the destination PE and CoCo instance. The CoCo agent that installs the flow forwarding rules on the PE switches needs to know what the destination PE for each IP prefix is. The CoCo agent learns this by running BGP and exchanging BGP/MPLS IP VPN information (RFC 4364 [3]). The CoCo agent also needs to know about the mapping between customer VLAN ID and CoCo instance and the IP prefixes that the customer uses in that CoCo instance. In the initial implementation of the CoCo prototype, the person adding a site to a CoCo instance configures this manually via the web portal.

VI. CoCo Control Plane Information Exchange

A CoCo agent has several tasks. One task is to control the OpenFlow switches in its domain by doing topology discovery and configuring flow forwarding rules on the switches. The other task is in the inter-domain control plane of CoCo. BGP is used to exchange information between the CoCo agents. This will be done similar to RFC 4364 [3] "BGP/MPLS IP Virtual Private Networks (VPNs)". There are BGP peering relationships between neighbour CoCo agents. The CoCo agents also work with the concept of transit. E.g., CoCo agent *a1* has a peering relation with CoCo agent *a2* only (see Figure 1). CoCo agent *a1* exchanges information with CoCo agents *a3* and *a4* via CoCo agent *a2*, which in this case acts as a transit agent.

For each PE in its domain, a CoCo agent sends the following information to its CoCo BGP peers:

- *VPN-IPv4 address family (12 bytes) (RFC 4364 [3])*
  The 12 bytes consist of an 8 byte Route Distinguisher (RD) and a 4 byte IPv4 address. An RD is encoded as a 2 byte Type and a 6 byte Value. We will use Type 2

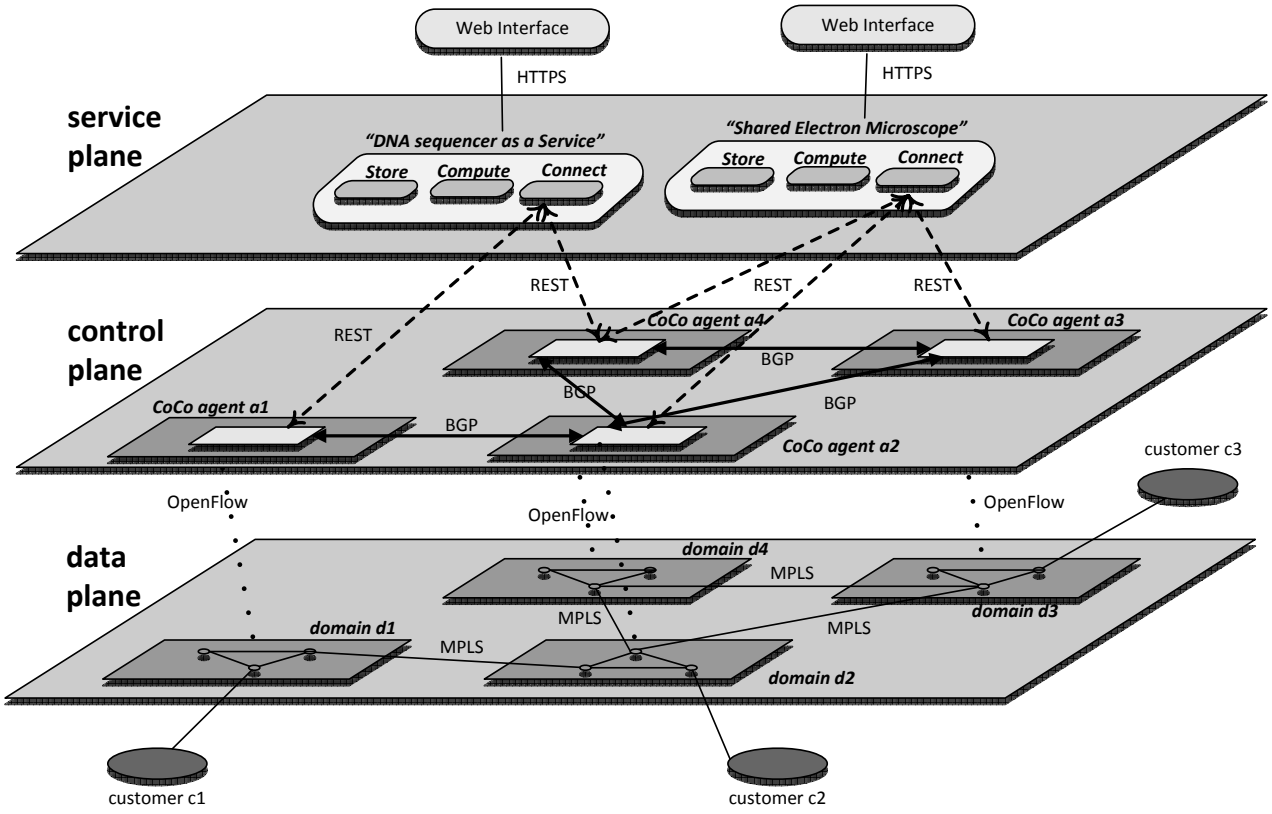

Fig. 1. CoCo Layers Architecture

RDs that consist of a 4 byte AS number followed by a 2 byte value. This value is managed by each domain, so by each CoCo agent. The CoCo agent manages a list of free values and assigns a unique value to each CoCo instance.

- *VPN-IPv6 address family (24 bytes) (RFC 4659 [4])*
  The 24 bytes consist of an 8 byte Route Distinguisher (RD) and a 16 byte IPv6 address. The RD will be the same value as for IPv4.
- *Next hop (VPN-IPv4 route with RD == 0)*
  The next hops in the route announcements should point to the PE that has the prefixes behind it. The CoCo agent assigns unique IPv4 addresses (from 10.0.0.0/24) to each PE in its domain to be used as next hop.
- *MPLS label to reach that PE (RFC 3107 [1])*
  This is done in the NLRI by using a AFI of VPN-IPv4 and a SAFI of 4. The NLRI is encoded as one or more triples of the form <length, label, prefix>. The length is in bits and includes prefix and label(s). Each label is encoded as 3 octets, where the high order 20 bits contain the label value, and the low order bit contains *Bottom of Stack*. The prefix field contains address prefixes followed by enough trailing bits to make the end of the field fall on an octet boundary.
- *CoCo instance identifier (2 bytes) encoded in Route Target (RFC 4360 [2])*
  A Route Target is sent via BGP Extended Communities (8 bytes) (RFC 4360 [2]) and is structured the same as a Route Distinguisher. We will use a Type 2 RD again with a 4 byte AS number and a 2 byte value. The value identifies the CoCo instance and will be used as inner MPLS label in the data plane for all traffic belonging to that CoCo instance.

VII. OpenDaylight

OpenDaylight [7] is a community-driven open source SDN controller framework hosted at the Linux Foundation [8]. The OpenDaylight project started in April 2013 and the first release was in February 2014. It is currently seen as one of the major open source SDN controllers. We have chosen to use OpenDaylight in the CoCo project because of this large and growing user and developer community.

OpenDaylight is mostly written in Java and uses OSGi [9] to dynamically load components (bundles). It supports several protocols towards network elements, including OpenFlow 1.0

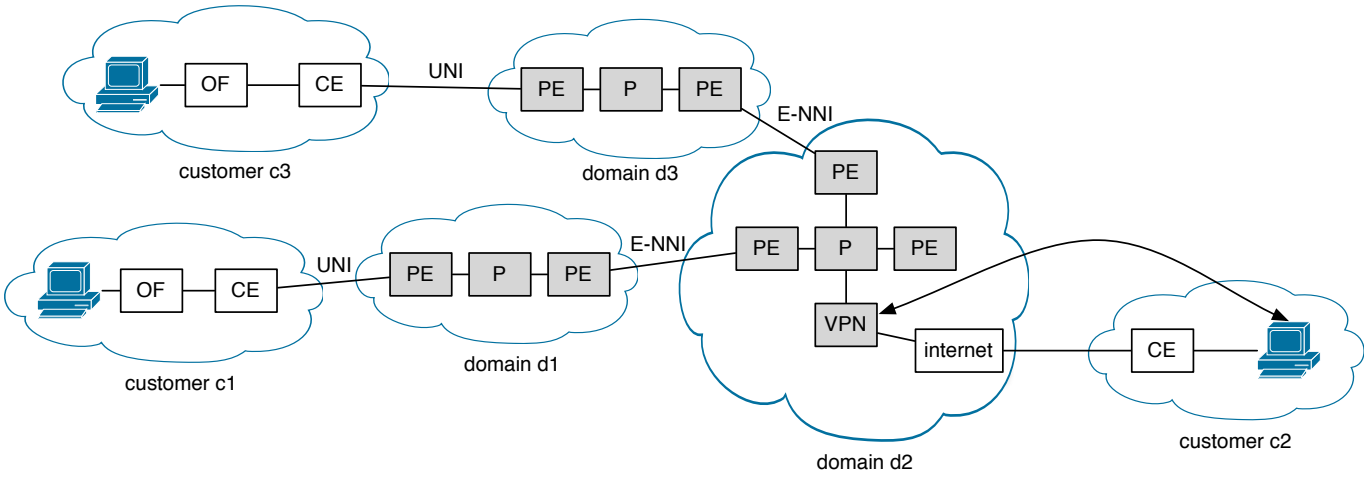

Fig. 2. Interdomain Data Plane Forwarding

and 1.3. Many components support REST APIs that can be used by SDN applications. Our initial implementation uses a REST API to configure forwarding entries in the OpenFlow switches. A later version of the prototype will be implemented as an OpenDaylight bundle so that it can use more of the internal OpenDaylight functionality, such as the topology manager.

VIII. L2 VPN CHALLENGES

The initial version of the CoCo prototype offers a L3 VPN service. Each site has one or more IPv4 and/or IPv6 prefixes and forwarding is based on IP address. A later version will extend this to also provide a L2 VPN service. However, there are some scalability challenges that need to be addressed, especially in the multi-domain case.

A L2 VPN service requires three major additional features:

- IP address assignment and avoiding duplicates.
- MAC learning (or an equivalent) is needed at the edges.
- The network must support broadcast for broadcast and unknown traffic.

In a L2 VPN service all nodes at all sites reside in the same L2 network and thus share the same IP prefix. There is no routing. Each node must get a unique IP address from that prefix.

One option is to use DHCP (or DHCPv6). This requires either a single server (with backup servers for redundancy) providing addresses from one big pool or multiple servers, each using a pool with a unique part of the total available address space. The latter option is difficult to manage, as it is usually hard to predict how many addresses are needed at each site.

For IPv6 there is the option to use SLAAC (Stateless Address Autoconfiguration). This requires a mechanism for ICMPv6 prefix announcement and discovery. However, this traffic does not need to traverse the backbone, as it is only needed at the edges. It is probably best to implement this at each site by using a proxy mechanism that intercepts discovery messages before they reach the backbone.

When an Ethernet frame coming from a CE device reaches a PE edge device there are two possibilities. When the destination MAC address is known, the frame can be forwarded on the proper interface. When the destination MAC address is not known, a L2 VPN service would usually broadcast that frame towards all destination PEs. This requires broadcast support in the backbone network. VPLS implements this with a full mesh of circuits between all PEs. This has clearly scalability issues. Recent work in the IETF L2VPN working group tries to solve this scalability problem with the EVPN specifications. With EVPN, MAC/IP address information is explicitly exchanged between PEs via the BGP protocol. By doing this there are no unknown MAC destinations anymore.

When a host in one site wants to communicate with a host in another site, it needs to know its IP address. This is done via the ARP and ND protocols. These messages are broadcast (multicast) and usually the destination node responds with a reply. In a L2 VPN service this requires broadcast (multicast) support in the backbone. Another option is to use proxy ARP (and ND). At the edge the ARP (ND) requests are intercepted and a proxy (that knows the destination IP address) replies to the sending node. The ARP (ND) messages never enter the backbone.

IX. COCO SERVICE VALIDATION

As mentioned before, the CoCo service is intended to be easy to use by its end-users, however, rolling out such a new service will require some initial efforts from the network administrators. Fortunately, there are possibilities to validate and tweak the CoCo service in a separated environment before deploying it in a production network. One of the options which requires almost no resources (except for the running PC) is to use Mininet [10]. Mininet is a software emulator allowing the creation of a virtual SDN network which can use either its internal OpenFlow controller or be instructed to

use an external one, e.g., OpenDaylight. Moreover, Mininet switches use OVSDB (Open vSwitch Database, [11]) which is recognized as a *de facto* management protocol standard for SDN [12].

The CoCo project will release some configuration and validation scripts which can be used to setup a Mininet-based experiment network for a CoCo service. For a quick start, a supplied predefined network topology can be utilized but using a custom topology (i.e., reflecting a production network where CoCo is actually to be deployed) is also entirely possible. An example of a command for running Mininet with a custom topology could be

```
# mn --custom /my/path/setup_topo.py --topo cocotopo
```

where `/my/path/` localizes a Python script which contains the instructions for adding the devices to be emulated (hosts, switches), their properties (like names or addresses) as well as connectivity information. Finally, `cocotopo` is a custom identifier of a given topology. Various other parameters can be supplied, for example if on IP address `1.2.3.4` we have an OpenDaylight controller listening on port `6633`, with OVSDB plugin enabled, running OpenFlow v. 1.3, the following options should be appended to the command above:

```
--switch ovsk,protocols=OpenFlow13 --controller
remote,ip=1.2.3.4,port=6633
```

The network created with Mininet allows for some performance validation to be conducted. However, due to the nature of Mininet (software emulator) there are certain limitations in, for example, attainable speed transfers, see [13] for the details. Nevertheless, Mininet can be regarded as a good (but not the only one, see for example [14]) option for experimenting with SDN and we plan to continue to use it in CoCo project, next to a physical testbed.

X. Summary

The Community Connection (CoCo) project is in the front-line of the OpenFlow and OpenDaylight developments. The prototype CoCo service that is being developed will facilitate a new generation of improved on-demand, user empowered and multi-domain connectivity services. In combination with recently emerging cloud services the CoCo service will meet the demand for innovative business services, such as a DNA sequencer as a Service in the eScience community. Although we encountered a number of unresolved issues when working with the fast developing technology of OpenFlow and OpenDaylight, we have illustrated how the innovative CoCo service is feasible and we implemented an initial prototype. In our following research we will further validate the CoCo prototype with testbed experiments and Mininet simulations. In addition, we will further explore the extension of the CoCo prototype to an on-demand, multi-domain L2 VPN service.

Acknowledgment

This paper has been produced with the financial assistance of the European Union. The contents of this document are the sole responsibility of SURFnet and TNO and can under no circumstances be regarded as reflecting the position of the European Union.

References

[1] Y. Rekhter and E. Rosen, *RFC 3107, Carrying Label Information in BGP-4*, 2001
[2] S. Sangli, D. Tappan and Y. Rekhter, *RFC 4360, BGP Extended Communities Attribute*, 2006
[3] E. Rosen and Y. Rekhter, *RFC 4364, BGP/MPLS IP Virtual Private Networks (VPNs)*, 2006
[4] J. De Clercq, D. Ooms, M. Carugi and F. Le Faucheur, *RFC 4659, BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*, 2006
[5] L. Andersson, I. Minei and B. Thomas, *LDP Specification*, 2007
[6] Galaxy Project http://galaxyproject.org/
[7] OpenDaylight http://www.opendaylight.org/
[8] OpenDaylight http://www.linuxfoundation.org
[9] OSGi (Open Services Gateway initiative) http://www.osgi.org/
[10] Mininet http://www.mininet.org/
[11] B. Pfaff and B. Davie, Ed., *RFC 7047, The Open vSwitch Database Management Protocol (OVSDB)*, 2013, http://tools.ietf.org/html/rfc7047
[12] https://wiki.opendaylight.org/view/OVSDB_Integration:Design
[13] What are Mininet's limitations? https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what-are-mininets-limitations
[14] ns-3 vns-3-dev documentation: OpenFlow switch support http://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html
[15] R. van der Pol, B.M.M. Gijsen, R.J. Strijkers, *Community Connection Service for eScience* https://tnc2014.terena.org/getfile/980