Community Connect (CoCo) 2.0: user initiated, on-demand, SDN-based, multi-domain VPN service. Technical Report

Date	03 March 2017
Author(s)	Piotr Zuraniewski, Michal Golinski, Borgert van der Kluit, Bart Gijsen, Marta Antosik, Simon Gunkel

Number of pages	20
Other parties involved	SURFnet

Contents

1 Introduction

This is the final technical report of the CoCo 2.0 project. It describes certain aspects of planning, execution and results of the project. It also serves as a “how-to” reference, giving detailed instructions about the development environment set-up as well as an example on instantiating a multi-domain topology in the *mininet* environment.

In the preceding GÉANT3Plus open calls project the initial version of the CoCo service prototype was developed and validated. The CoCo service enables end users to instantiate and tear-down VPNs and to modify the connected end points for the VPN. In the following Multi- domain (MD)-CoCo project the initially offered by CoCo prototype functionality was extended to be applicable in a multi-domain environment. The current project improved the functionality by enabling a process to invite other users to join a multi-domain VPN. In addition, it was demonstrated with an improved platform to show the CoCo 2.0 system possibilities.

2 Project Objectives and Related Work

In preceding GN3Plus “CoCo” project (November 2013 until March 2015) SURFnet and TNO designed, developed and validated an on-demand, user initiated Virtual Private Network (VPN) connectivity service in a single domain.In the following project Multi-domain CoCo (MD-CoCo April 2015 until December 2015) the available CoCo functionality was extended to operate in a multi-domain environment.

In current project CoCo 2.0 (2016) the process of establishing of the VPN was changed to enable adding new users by VPN owner by sending invitation with an email. In addition, more focus was put on independence of single domains by avoiding central control over multiple domains. Another goal was to integrate VPN Intent functionality present OpenDaylight controller into our system. Since intent framework allows descriptive rather than prescriptive formulation (“what” instead of “how”), using it can be viewed as moving a system to a higher abstraction level.

3 Implementation details and architecture adjustments

In this Section we give some more detailed explanation on implementation of selected features of CoCo as well as describe the differences between the original architecture and our implementation. Such adjustments were needed for various practical reasons such as lack of support in hardware or software for the selected features or bugs discovered in the software that CoCo depends on.

A detailed architecture description related to MD-CoCo can be found in [1]. In brief, MD-CoCo uses Multiprotocol BGP (MP-BGP) to exchange information about VPNs (i.e., CoCo instances) that span between multiple domains. Two MPLS labels were planned to be used: an inner label to identify a CoCo instance and an outer one to allow aggregation in the core network(s) that are crossed by the CoCo instance. In each domain a CoCo database is instantiated where information about participating sites is stored.

3.1 Sites

In single-domain CoCo a site i.e., the entity added to a VPN, was modelled as a host connected with a VLAN tagged interface to a Provider Edge (PE) switch. This VLAN tag reflected the fact that a service provider can offer its services to a customer via so-called Multi-service Port (MSP). In this setting each service (e.g., regular Internet connectivity, dedicated high-speed circuit, CoCo service) can be placed in a separate VLAN. In CoCo 2.0 the entity added to VPN is a subnet and a set of subnets constitute a site. Moreover, due to the fact that VPN Intent did not support VLAN handling, the tagging was not implemented and the UNI (connection between CE and PE) was realized using plain Ethernet. The CE device was modelled as a router, acting towards its LAN as an ARP proxy. Thanks to this solution only a very limited amount of MAC addresses (specifically: MACs of interfaces of routers connected to PE switches) have to be known to the provider and these are stored in the CoCo database. Unfortunately, VPN Intent did not support MAC re-writing. As a workaround we used dummy switches placed between CE and PE with only two functions:

i) passing all traffic from CE to PE;
ii) re-writing MAC addresses of the frames coming from PE to CE WAN interface MAC address and passing these frames towards CE.

3.2 Double MPLS labelling

In the original CoCo architecture, the setup from Figure 3-1 was proposed. Here traffic directed towards CoCo network is ingressing into the PE tagged with the Multi Service Port (MSP) VLAN. This tag is removed at the PE and two MPLS labels are added: first to identify the VPN trough the VPN MPLS tag and the second to mark (PE MPLS) the traffic directed to core traversal. Double MPLS enables aggregating single VPN flows between certain core edges.

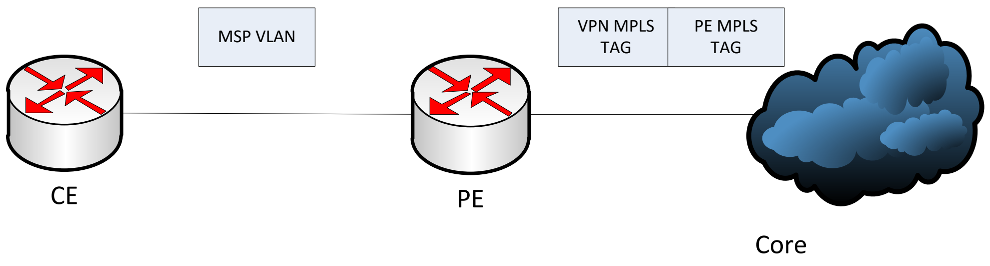


Figure 3-1 Original tagging

However, due to the double MPLS tagging feature missing in the current VPN Intent implementation (version Beryllium), only single MPLS label can be used. The temporary workaround could be to use VPN VLAN labelling to distinguish specific CoCo instances and MPLS for aggregation in the core. (see Figure 3-2). Unfortunately, VLANs handling is not implemented in the current Intent version Beryllium either.

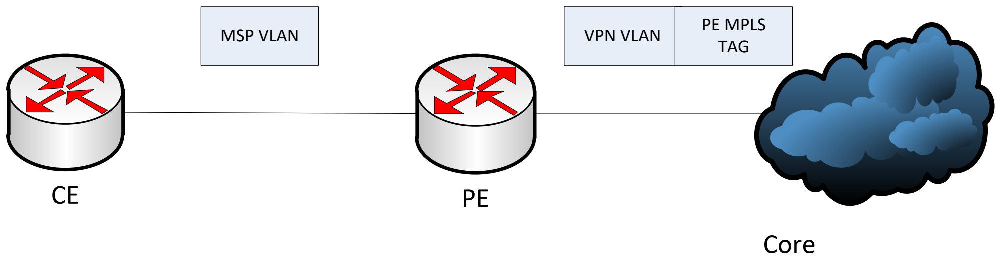


Figure 3-2 Proposed workaround for tagging

At the end, due to the missing features, each flow has its own MPLS label (see Figure 3-3) which increases a flow tables size significantly, so double MPLS labelling is recommended as soon as it is included in the Intent implementation.

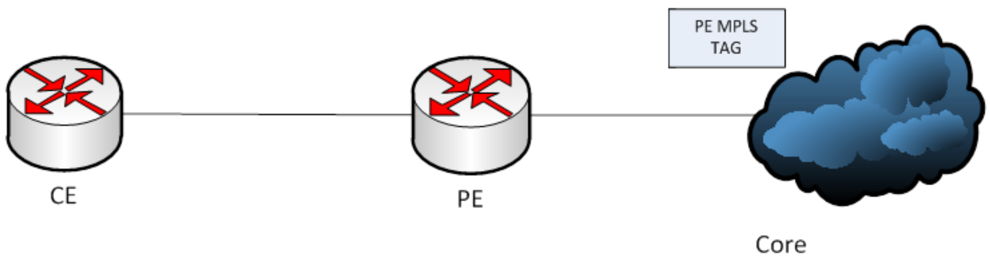


Figure 3-3 Implemented tagging

3.3 Crossing domains

A design choice we made concerns how to treat a subnet from a remote site participating in a VPN from a local domain point of view. We essentially treat it as a subnet from a virtual site in a local domain. In practice it means adding a virtual site to . This enables us to reuse almost all the flow construction logic developed for the single domain CoCo prototype.

In the current implementation we did not consider more complex topologies which include transit/multihoming. For these cases we foresee that routing information would be needed to assist CoCo agent in making correct decisions. For example, in dual-homing situation CoCo agent would have to select to which of the two ‘gateway’ switches a virtual site needs to be attached. Since we already run BGP (although only to send the information, not to make any routing) the selection of this protocols seems natural. Note that using ExaBGP alone is not sufficient then since ExaBGP does not maintain RIB so coupling it with routing engine such as Quagga would be needed.

3.4 BGP speakers and communication with and between them

We have decided to use MP-BGP [2] as a means to communicate VPN-related information between a customer and a service provider as well as between service providers. In each provider’s domain we have instantiated a stand-alone BGP speaker, namely *ExaBGP*. This BGP speaker establishes 2 types of communication channels:

1) BGP peering with its counterpart(s) in the other domain(s)
2) CoCo agent-to-BGP communication

We will now elaborate more on the functions and realization of these two communication channels mentioned above:

1) BGP peering (‘advertise route’) is used to send information about the subnets added to a given VPN which are available in a local domain of a given BGP speaker. This information is enriched with two BGP Extended Community attributes: a standard one being Route Target which carries a globally unique identification of a VPN and a custom, extended experimental transitive attribute to carry a security-related information (nonce).
   For deleting VPN ‘withdraw route’ message can be used. Right at this moment it is not implemented and VPN needs to be deleted manually (via portal) in all domains.
2) The CoCo Agent needs to retrieve information about the available prefixes originating outside its domain which came with BGP updates. In addition, it has to inform the provider’s BGP speaker that certain prefix has to be advertised towards peers in the other domains (or has to be withdrawn from advertisements). This communication channel is realized using a REST interface along with some custom-developed interfacing code.

Note that a third type of communication channel could also be used (currently not implemented), i.e.,

3) BGP peering with CE routers

3) Customer routers advertise available prefixes towards provider’s BGP speaker. All available prefixes which can potentially be added to some VPN can be advertised, along with a tag which can identify a prefix owner. All this information can be used to populate CoCo database. From the CoCo point of view we treat this information as given, since CoCo Agent does not interfere directly with the CE. Configuration of CE to advertise appropriate information can be treated as on-time/very infrequent effort. In the current implementation this information is manually entered into CoCo Agent database. We want to stress that regardless of the way to populate a database (manual or via BGP), joining or leaving VPN does not require any reconfiguration of a CE router.

Note that a BGP-related project is also executed within the ODL community: BGP LS PCEP, see [3]. It may be possible that at least part of the tasks mentioned above can be realized using the BGP LS PCEP module instead of ExaBGP, but we leave its assessment as a potential topic for further research.

4 Testbed environment

Our intention was to build two functionally and topologically identical testbeds. One being completely virtual, i.e., using virtual switches (*openvSwitch, mininet*) and the other using physical network switches present at TNO's and SURFnet's laboratories.

4.1 Topology T1

Topology T1 is a simple multi-domain topology and consists of three domains: SURFnet (SN), TNO-North (TN) and TNO-South (TS). An overview is presented in Figure 4-1. Having two domains located within the TNO domain and one at SURFnet allows for easier troubleshooting by first performing inter-domain tests locally at TNO and validating the CoCo multi-domain functionality via the TNO-SURFnet connection afterwards. Since, however, having three domains would require more architectural refinements and code development related to, for example transit or multi-homing, we have decided to focus on the two-domain case, keeping the TNO-North and TNO-South active for the time being. The details of the actually used topology are included in Figure 4-2
In the virtual testbed, we have assigned a separate virtual machine for each domain which hosts the following components (see also Section 4.2.1)

- *mininet* generated network, i.e., switches (OVS, linux bridge), BGP speakers (both CE's and provider's) and customer hosts; moreover, for the sake of tests, there are also "pingable hosts" connected to domain border switches
- *OpendayLight* controller instance controlling all switches within one domain
- *tomcat7* webserver instance hosting the domain's CoCo portal
- *MySQL* server instance hosting the domain's CoCo database

The aforementioned VMs are inter-connected via GRE tunnels. We have found out that *openVswitch* in version 2.4.0 is not capable to instantiate the switch in the kernel mode and handle frames containing both VLAN and MPLS tags correctly. Only user mode switches can do it, however, in this case we found another bug: the GRE tunnel does not function because encapsulation is not performed. Therefore we have decided to run all Provider Edge (PE) and Provider Core (PC) switches in user mode and add a "ghost" switch in the kernel mode with the only function to accept all the traffic coming into it from the domain and send it via GRE tunnel. These switches are marked with the small icons in Figure 4-2.

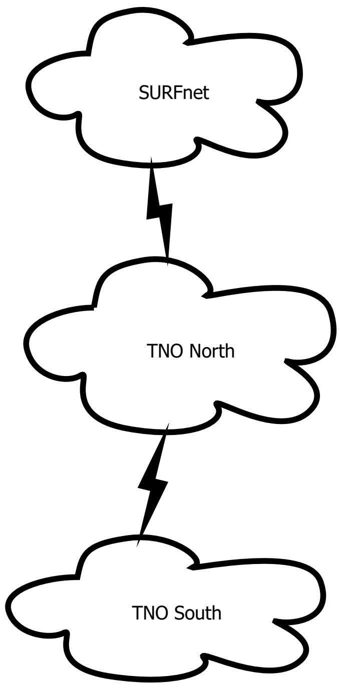


Figure 4-1: Topology T1 – overview

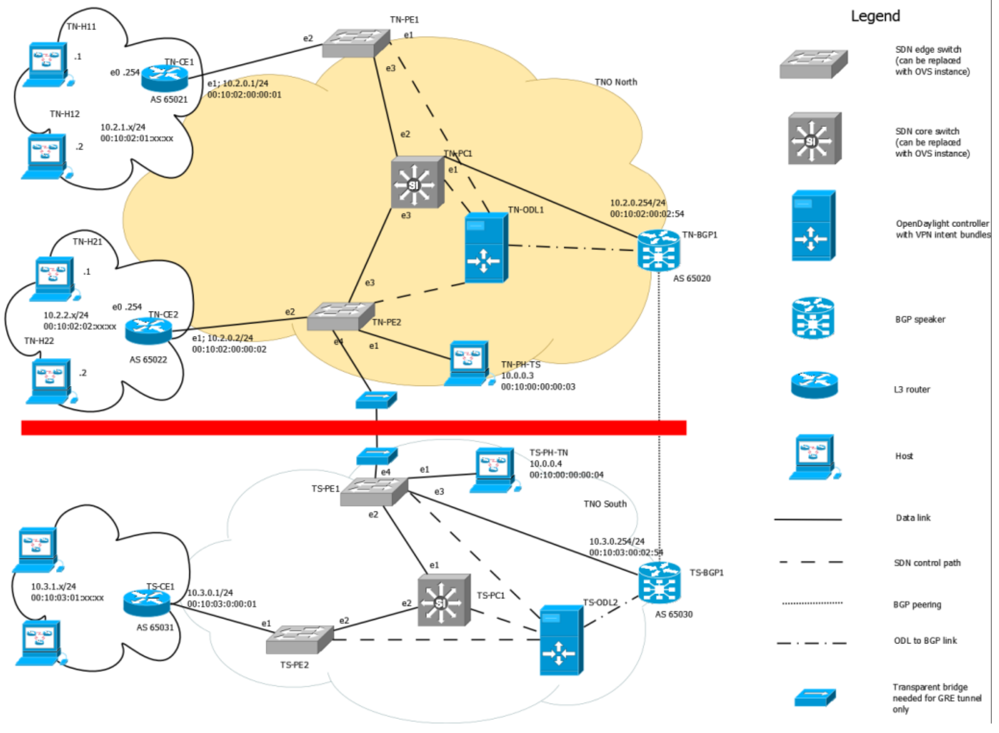


Figure 4-2: Topology T1 - details

4.1.1 Hardware testbed

In addition, the topology T1 was also tested with the hardware set up. The switches marked in the Figure 4-2 in grey has been replaced with physical equipment of PICA8. Unfortunately, both used models (P3922 and P3290) with the recent operating system PicOS2.7.3 have shown the same wrong behaviour while treating MPLS labelled packets. In the core, a labelled packet is properly recognized, but it is not forwarded correctly to the next hop. Due to that unresolved issue, it was impossible to continue testing with hardware set up which limits test to only virtual environment.

4.2 Steps to run CoCo

4.2.1 *Required software*

We used Ubuntu 14.04 LTS operating system. It is likely that CoCo agents will also run on the other operating systems, however, we have not tested that. We assume that the following software is already installed on a given machine. Note, that in ~/install_scripts directory (see below) there are shell scripts which may be helpful in installing most of the required packages.

No	Software	Version	Remarks
1	OpenDaylight	Berylium-SR2	Extra features are needed: • feature:install odl-vpn-service-intent • feature:install odl-dlux-all
2	Tomcat	7.0	
3	Eclipse	JavaEE	
4	Quagga	0.99.25-dev	main branch sent malformed vpnv4 updates; this forked version handles them correctly https://github.com/LabNConsulting/quagga-vnc/
5	Python	2.7.6	
6	mininet	2.2.0	
7	openVswitch	2.4.0	
8	MySQL	14.14	Distribution 5.5.53
9	ExaBGP	3.4.16	-

4.2.2 *Folders*

The folders listed below should be cloned from git repository. They host the following content

No	Name	Files hosted	Remarks/source
1	CoCo/demo_invitation	topology initiation and initial configuration (including GRE tunnels, initial OVS flows), wrapper functions and setting up Linux bridges	https://github.com/CommunityConnect/CoCo/tree/master/demo_invitation
2	CoCo/demo_	exabgp routers	

	invitation /bgp_configs	initial configuration files	https://github.com/CommunityConnect/CoCo/tree/master/demo_invitation/bgp_configs
3	CoCo/install_scripts	bash scripts helpful in installing required software	https://github.com/CommunityConnect/CoCo/tree/invitations/install_scripts
5	CoCo/cocoportal	CoCo portal and agent files	https://github.com/CommunityConnect/CoCo/tree/master/cocoportal

4.2.3 *Running the CoCo service (in mininet)*

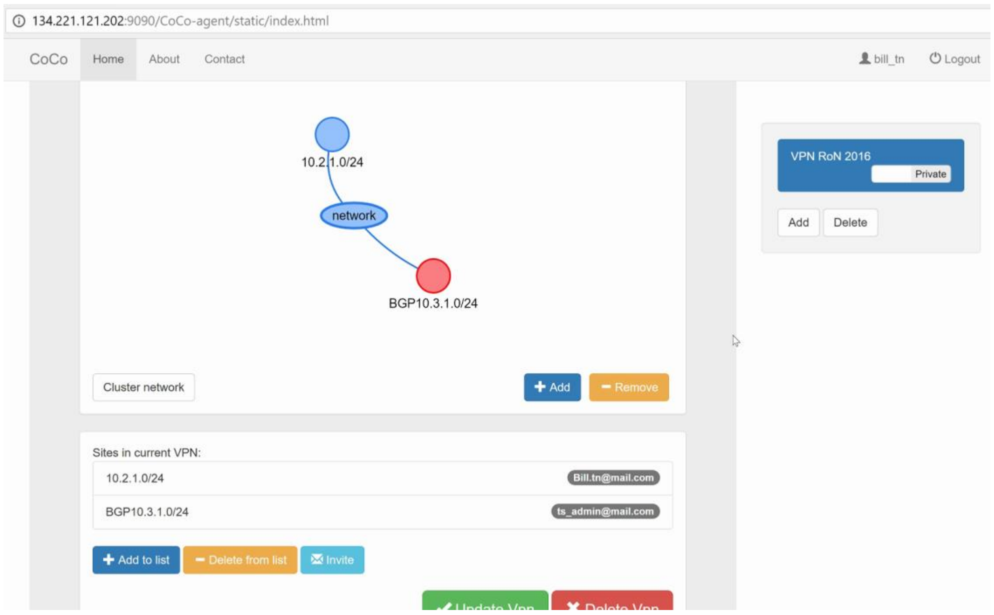
The following steps are needed to start the CoCo service:

1) start *OpenDaylight* controller with correct features installed, see 4.2.1
   A. typically: ~/ distribution-karaf-0.4.2-Berylium-SR2/bin/karaf

2) start *mininet* with selected topology (currently: TNO-North or TNO-South)
   A. TNO-North: ~/CoCo/demo_invitation/mntn.sh
   B. TNO-South: ~/ CoCo/demo_invitation/mnts.sh
   C. Inside the script it is indicated which domain is initiated (ts or tn) and in which mode:
      i. full – the full domain is run at that instance
      ii. all – all components besides switches marked in grey in the Figure 4.2 are run (it is a preparation for hardware set-up)
      iii. ce1/ce2/bgp – a single component in run

3) start CoCo portal (see Appendix)
   A. Start from eclipse
      i. make sure that tomcat is not running if server is run from Eclipse (use *sudo tomcat7 service stop* command if needed)
      ii. navigate to http://localhost:8080/CoCo-agent/static/index.html
   B. Start on server
      i. Deploy portal on server (see Appendix)
      ii. Start server: *sudo tomcat7 service start*
      iii. navigate to http://{serverip}:9090/CoCo-agent/static/index.html

4) Add selected sites to a given VPN (see Figure 4-3) ; recall, that for inter-domain communication sites have to be added to the gateway switch and this action has to be performed in each domain (see Section 3.3)



Sites in current VPN:

10.2.1.0/24	Bill.tn@mail.com
BGP10.3.1.0/24	ts_admin@mail.com

Figure 4-3: CoCo portal sample page

4.2.4 *Verification*

Connectivity (or lack of it) between the hosts having an address in the range which is added to a VPN can be verified by opening the terminal window of a given host and pinging the far-end host. For example, if sites behind routers TN-CE1 and TS-CE1 were added to the same VPN it means hosts in LANs 10.2.1.0/24 (TN-CE1) and 10.3.1.0/24 (TN-CE1) should have a connectivity, but none of them should reach 10.2.2.0/24 (TN-CE2). In order to verify this we issue the following command in the *mininet* prompt at TNO-North:

- `xterm tn-h11 tn-h21`

Next, in the hosts' terminal windows we issue a ping command trying to reach host ts-h11

- `ping 10.3.1.1`

Ping should be successful from tn-h11 and fail from tn-h21.

5 Next steps

5.1 BGP related software

As mentioned in Section 3.4, a BGP peering between domains needs to be established.

A mandatory feature to be supported by the BGP speaker is

1. Ability to send/receive arbitrary Extended Community attributes

Optional features to be supported by the BGP speaker are

2. Multiprotocol Extensions for BGP-4 (RFC 4760, [2])
3. BGP/MPLS IP Virtual Private Networks (RFC 4364, [4])
4. Multi label tagging for MPLS
5. Usage of multiple VPNs and associated VRF

The optional features were not used in the current implementation. We, however, foresee that for the more complex scenarios (e.g., interfacing with traditional, non-SDN based BGP/MPLS IP VPNs) some additional functionality of BGP may be required.

One of the most popular software router implementation – Quagga – did not support arbitrary Extended Community attributes. We have decided to select ExaBGP. Bear in mind, however, that ExaBGP only speaks BGP but (as opposed to Quagga) does not maintain Routing Information Base. In the current implementation it is not a problem, however, as discussed in Section 3.3, for more complex topologies where actually routing may be needed, other solution then ExaBGP (alone) will be needed. That means either replacing ExaBGP with the engine which can do both routing and handle arbitrary Extended Communities or coupling ExaBGP with the routing engine (e.g., Quagga).
Note, that in current implementation we have still used Quagga on CE routers. Again, as mentioned in Section 3.3, in the current implementation, there is no BGP peering established between CE and ExaBGP instance so CE merely acts as a default gateway and proxy ARP for its LAN.

Below there is a short list of alternative open source BGP-related software. Except for Quagga and ExaBGP we have not tried any of these in our prototype so any remarks about (lack of) support of a given function should be taken with care.

ExaBGP,
https://github.com/Exa-Networks/exabgp/wiki The Swiss army knife for BGP. Supports RFC 4760, [2] and arbitrary Extended Community attributes.

Quagga
http://www.nongnu.org/quagga/
Version from the main branch sent malformed vpnv4 updates; this forked version handles them correctly
https://github.com/LabNConsulting/quagga-vnc/

BIRD,

http://bird.network.cz/
https://en.wikipedia.org/wiki/Bird_Internet_routing_daemon,
According to the documentation it supports BGP, but does not support MP-BGP (RFC 4364, [4]). There exists, however, a custom version of BIRD which claims support for VPNv4, see, http://bird.mpls.in/.

OpenBGPD,
http://www.openbsd.org/
https://en.wikipedia.org/wiki/OpenBGPD
RFC 4760, [2] and (RFC 4364, [4]) referred in the manual http://www.openbsd.org/cgi-bin/man.cgi/OpenBSD-current/man8/bgpd.8?query=bgpd
Operating system is OpenBSD, FreeBSD

XORP, https://en.wikipedia.org/wiki/XORP, supports BGP, but MP is unknown
According to a user question on the forum it can support up to 7 VRF's but it is not mentioned in their Wiki. Last release is dated January 2012

VyOS, see https://en.wikipedia.org/wiki/VyOS, a fork of Vyatta. It is linux based. Last version is dated August 2015, It does not support MPLS.

BGP module in Opendaylight [3]

5.2 Connection between testbeds

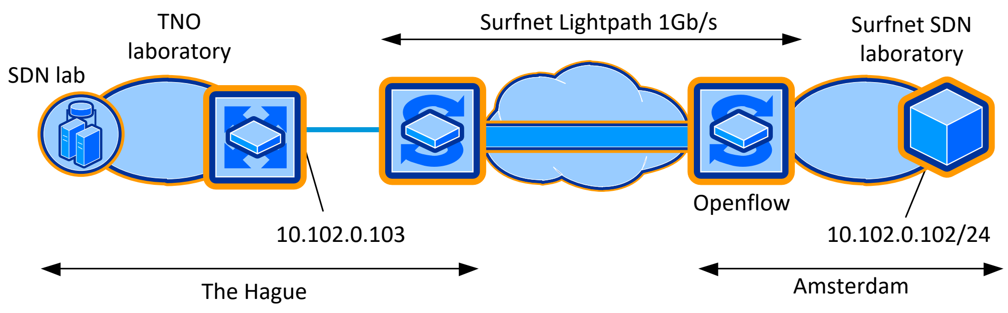


Figure 5-1

Due to safety, continuity and quality reasons, SURFnet and TNO decided to have a direct connection between both Labs. This connection is not shared with the operational/production link between TNO and SURFnet. The lightpath connection is terminated on the shared (POP) switch on the TNO location and from this switch connected to the TNO labs environment using its own interface and (optical) internal infrastructure.
On the 24th of September '15 the physical connection was completed. However, due to the problems with the hardware testbed, the SURFnet domain has not been used in the testing.

5.3 Proposed improvements

There are number of ways for enhancing the current prototype. Some of the possible enhancements are:

- Migrate prototype to the physical testbed after resolving the issue with incorrectly handle of MPLS labels
- Improve architecture and implementation to handle more complex topologies and cases such as transit, multi-homing, interoperability with traditional ("non-SDN") domains
- Introduce measurement-based quality and security assurance
- Verify applicability of programmable data plane for functions like client authentication

6 Appendix: Install and Run CoCo Portal

This is the explanation on how to create CoCo development environment.

6.1 Initial configuration

1. Clone the repository:
   a) git clone https://github.com/CommunityConnect/CoCo b)
2. Import project to Eclipse MARS:
   a) File/Import/Maven/Existing Maven Projects,
   b) choose "cocoportal" folder within the repository,
   c) a project called CoCo-agent should be created on clicking Finish,
   d) If you are getting "log cannot be resolved" errors, that means that lombok is not installed with Eclipse. (It is installed in Eclipse Mars on the server). Lombok is used for certain annotations in source code (makes source code cleaner). If this is the case, install lombok (https://projectlombok.org/),
   e) add tomcat7 server configuration to Eclipse (tab servers, Create new server -> Download and install). You may need to wait a couple of seconds before you will be able to click next or finish, because Tomcat is downloading,
   f) add CoCo-agent configuration to the web server.
   g) configure:
      cocoportal\src\net\geant\coco\agent\portal\props\config.properties
      cocoportal\src\net\geant\coco\agent\portal\props\jdbc.properties (edit ip of MySQL server with database, database name and credentials)

6.2 Configuration of the Tomcat server

1. Edit servers/server.xml file if needed (ports for Tomcat).
2. Run the server. The portal should be accessible under http://localhost:9090/CoCo-agent/static/index.html (change localhost and ip address if you run Tomcat remotely).

6.3 Running Tomcat Server Remotely

1. Configure Tomcat
   a. Server needs to be configured to for Management Application Access
   b. Helpful links:
   - https://www.dontpanicblog.co.uk/2014/06/25/deploying-to-tomcat-7-with-maven/
   - https://www.mkyong.com/maven/how-to-deploy-maven-based-war-file-to-tomcat/
2. Configure Eclipse
   a. Setup configuration files per server
      i. Directory is net.geant.coco.agent.portal.props
      ii. config.{name}.properties
      iii. jdbc.{name}.properties

   b. Setup Run configuration
      i. Has to be maven build
      ii. Base directory: ${project_loc:CoCo-agent}
      iii. Goals: tomcat7:redeploy
      iv. Set the following parameters
      v. tomcatIpPort = {server ip:admin port}
      vi. deployName = {name}
3. Deploy WAR container to server
   a. Simply run the maven build configuration
   b. Check console output
   c. Common errors:
      i. Server not accessible and needs configuration of manager console
      ii. Server is crashed and needs reboot.

7 Bibliography

[1] R. v. d. Pol, B. Gijsen, P. Zuraniewski, D. F. C. Romão and M. Kaat, “Assessment of SDN technology for an easy-to-use VPN service,” *Future Generation Computer Systems,* 2015.

[2] T. Bates, R. Chandra, D. Katz and Y. Rekhter, “Multiprotocol Extensions for BGP-4,” [Online]. Available: https://tools.ietf.org/html/rfc4760.

[3] [Online]. Available: https://wiki.opendaylight.org/view/BGP_LS_PCEP:Main.

[4] E. Rosen and Y. Rekhter, “BGP/MPLS IP Virtual Private Networks (VPNs),” [Online]. Available: https://tools.ietf.org/html/rfc4364.

[5] [Online]. Available: https://wiki.opendaylight.org/view/Network_Intent_Composition:Main.

[6] [Online]. Available: https://wiki.opendaylight.org/view/VPNService:Main.