31-03-2015

Open Call Deliverable OCA-DS1.1
Final Report (CoCo)

Open Call Deliverable OCA-DS1.1

Grant Agreement No.:	605243
Activity:	NA1
Task Item:	10
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	SURFnet
Document Code:	GN3PLUS14-1287-75
Authors:	Bart Gijsen, Marijke Kaat, Ronald van der Pol, Piotr Zuraniewski

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract
The aim of the CoCo project was to develop and validate a proof of concept for a novel on-demand connectivity service for research communities and education communities. In the CoCo project the CoCo architecture was developed and a prototype was implemented and validated successfully. The interaction with potential users resulted in use cases that illustrate the market demand for the CoCo service. This final report of the CoCo project provides an overview of the investigated use cases, the developed CoCo architecture and prototype and its successful validation. As guidance for further reading the report contains references to relevant deliverables from the CoCo project.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The advent of new networking technologies and Software Defined Networking (SDN) in particular, brings new opportunities for collaborative research within reach. A specifically promising aspect is the opportunity to create connectivity solutions in a more flexible way than with current Internet technology. In the Community Connection (CoCo) project we have built a service with cutting edge SDN technology and demonstrated how it empowers users in multiple domains to (de)activate multi-point Layer 3 Virtual Private Networks (L3-VPNs) on-demand.

In several interactions with stakeholders from the eScience community, a number of use cases were proposed and discussed. One of them, the "DNA sequencer as a Service" use case, was elaborated in more detail and discussed with the Wageningen University (WUR)/Plant Research International. The potential to facilitate higher utilization of the DNA sequencers by providing it 'as a service' to a larger community is an essential business driver for such expensive scientific instruments that get outdated relatively quickly. The business opportunity for this service illustrates the demand for flexible and easy-to-use connectivity services, such as the CoCo service in the eScience community.

The developed CoCo prototype demonstrates that a service that can fulfil this need has become within reach. The CoCo web portal lets end-users set up CoCo instances (multipoint L3-VPNs) within several seconds, without needing the help of network administrators to do manual configurations (which can take hours or days even) of the network switches. The developed CoCo software uses the OpenDaylight [ODL] controller interface to handle the automatic setup and tear down of the instances that the controller configures on the OpenFlow switches.

The CoCo architecture has been designed to operate this connectivity service both in a single-domain and in a multi-domain environment. We were successful in building a single-domain CoCo prototype. For the current state of the art of SDN technology we learned that our original objective to implement the CoCo architecture in a multi-domain context was too ambitious. The design of the architecture and the implementation of the prototype took more effort than we had anticipated in our project planning. We started with designing a Layer 2 VPN (L2-VPN) service, but concluded that there were many scalability and operational challenges, especially in the multi-domain case. After analysing existing work (RFCs, etc.) we concluded that a L3-VPN service was a more feasible solution. The state of the art of the SDN technology that we built on required additional work in order to make it sufficiently mature for our prototype. At the start of our project the MPLS support of our OpenFlow switches lacked some features that the CoCo architecture relies on and we ran into several bugs. This required frequent interaction with the switch supplier. Subsequently, several updates of the OpenDaylight controller that became available in the Helium release (available by the end of September 2014) solved some issues related to ease of configuration and interaction with the controller, bringing also more structured and complete documentation. In addition to these

contributions to the supplier and the OpendayLight community we managed to develop the CoCo prototype, implemented in SURFnet's SDN testbed.

The developed prototype functionality has been tested successfully. An automated test environment was developed for performing user-level experiments in the CoCo prototype testbed. The results from these experiments show that end-users can easily (de)activate CoCo instances and that CoCo instance activation times will be perfectly acceptable from a user's point of view.

In addition to the experimental validation in the SDN testbed a Mininet-based simulation environment was created. Mininet emulates the OpenFlow switches and includes an OpenDaylight controller in the CoCo testbed and it can run the developed CoCo code on top of that. In this way, Mininet simulations can be used for scalability validation by extrapolating the testbed configuration into a setting with many more OpenFlow switches and / or CoCo connected sites, for example. The calibration results that compare the behaviour of the prototype CoCo service in the testbed to the Mininet simulation look very promising.

The significant progress that we made in the CoCo project sheds light on new challenges. A first challenge is to complete the CoCo prototype to a multi-domain context for which it is designed. Other challenges include (a) the extension of the CoCo service to instantiate L2-VPNs, (b) to improve the security of the service (e.g. federated authentication, role-based authorisation) and (c) supervisor controls on the CoCo service for network administrators. In a follow-up of this GN3plus Open Call project, the CoCo project partners SURFnet and TNO have planned to expand and improve the CoCo prototype implementation and validate it in a multi-domain setting. With these expansions the project partners will explore further use cases enabled by the CoCo service. With the progress made in the CoCo project, we demonstrated how obstacles for collaborative research that are present in legacy network solutions can be alleviated by exploiting emerging SDN technology.

The results from the CoCo project have been published extensively. During the project we gave presentations and demonstrations at the TERENA Networking Conference (TNC2014) and the Super Computing (SC14) conference, amongst others. We also got in touch with potential users, suppliers and other communities that were working on similar concepts. We succeeded in collaborating with Ericsson in the OpenDaylight *VPN Service* project, where a BGP based inter-domain solution is being developed that is quite similar to the CoCo architecture. With these results and exposure the CoCo project has brought new perspective to collaborative research in the eScience community.

1 Introduction

1.1 Document purpose

The CoCo project is a project in one of the open calls funded by the European GN3plus project [CoCo]. It ran from October 2013 until March 2015. The CoCo service is a proof of concept intended for application in the research and education communities connected through participating NREN (National Research and Education Network) networks and the GÉANT network.

In the course of the project a number of milestones and deliverables were produced and the purpose of this final report is to provide an overview of these results. Further, the achieved results are compared to the originally planned results and the gap between them is explained in the following sections. As such this final report also serves the purpose of evaluating the project results.

1.2 Background

The advent of Software Defined Networking (SDN) is creating innovation opportunities for a wide range of use cases. In eScience research the importance of networked services and facilities such as medical and genome databases, scientific instruments, visualisation facilities, storage and cloud computing, is increasing. However, due to security and privacy issues, services and facilities may not always be exposed via the public Internet. On the current Internet secure communication does not come out of the box. It requires configuration of security technology by network administrators and in many cases by users. Reducing the required user actions has been a long term endeavour for Internet researchers and significant steps have been taken. For example, in most intranet environments users already feel as if secure communication comes out of the box and with HTTPS related technology secure client-server communication on the Internet has become child's play. And a little more skilled user will be able to apply file encryption for securely sending its content to his peers. However, here is where current, user-friendly secure communication stops. More generic secure communication technology such as Virtual Private Networks (VPNs), that are often point-to-point, involves manual processes at one or more Network Operation Centers (NOCs). This lack in communication technology leaves a variety of interesting applications out of scope. And the pain is felt in particular in the eScience community, where collaboration between experts from multiple domains will boost the progress of their research. The CoCo service creates a new type of user-friendly, secure, multi-domain and point-to-multipoint connectivity services that will open a new range of business opportunities, such as illustrated in the example presented in Section 2.

1.3 The CoCo service

The CoCo service is a multi-domain on-demand VPN service based on SDN technology that will enable scientists from multiple organisations to dynamically create virtual, private networks for sharing services and facilities. **Error! Reference source not found.** illustrates a CoCo instance that connects endpoints, for example laptop's or other devices used by scientist or scientific instruments, from multiple domains.

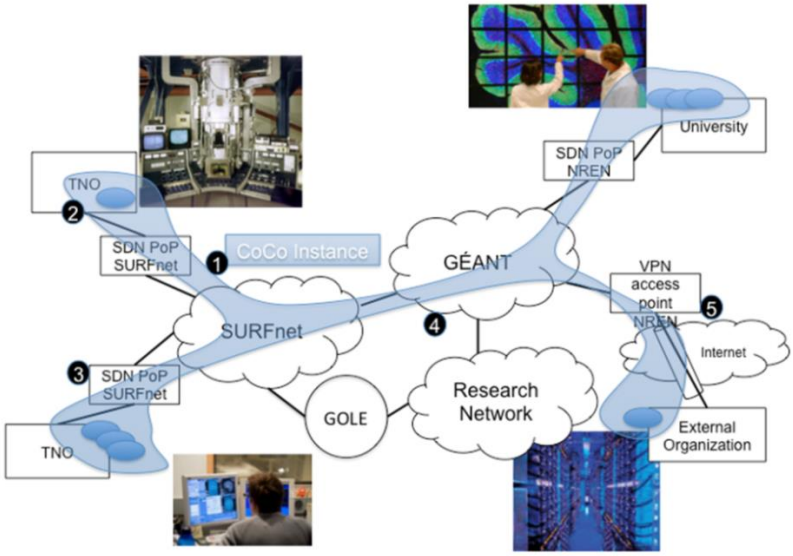

Figure 1.1: Topological overview of a CoCo instance connecting endpoints in multiple domains

After a one-time initial general set-up of the CoCo service by the network administrator end-users will be able to set up and manage CoCo instances via an easy to use web portal, without having to rely on further manual intervention of the NOCs. Using REST APIs applications can also be developed that can automatically set-up and tear down CoCo instances. The CoCo architecture is described in Section 3.

The CoCo prototype is developed on SURFnet's OpenFlow testbed. The testbed has an OpenFlow based infrastructure, controlled by an OpenDaylight controller. In addition, a web portal is developed via which the user can (de)activate CoCo instances and add or remove sites from an existing CoCo instance, for example. This web portal uses the REST interface of the OpenDaylight controller for effectuating these user instructions. The prototype is developed using existing state-of-the-art open source SDN frameworks and open standards are used where possible. Details about the developed prototype are described in Section 4.

Given the user-friendliness objective of the CoCo service the validation was done at user-level. This was achieved by creating an automated test environment with AutoHotKeys and several bash scripts. AutoHotKeys was used to emulate interaction with the CoCo portal from a user's browser. The bash scripts were used to parallelise simultaneous connectivity measurements from multiple VMs that are connected by a CoCo instance. The latter is needed to verify whether the activation of a CoCo instances indeed achieves connectivity between the VMs and to be able to measure the time that activation of

a CoCo instance takes. By having this test environment fully automated, we were able to test a large variety of setups for the CoCo prototype. The details and test results are described in section 5.

Finally, in Section 6 concluding remarks about the CoCo project are presented as well as the anticipated follow-up.

2 Use cases

An overview of the approach, the results and observations from the work on use cases in the CoCo project (Work Package 1) is presented in Table 3.1.

Section	Description of Contents
Approach	In order to explore the demand in the eScience community related to the CoCo service a workshop was organized, with eight participants representing eight academic institutes in The Netherlands. During the interactive workshop, held on January 21st 2014, a number of use cases were suggested and discussed. The "DNA sequencer as a Service" use case was selected to be elaborated in more detail, in interaction with the Wageningen University (WUR)/Plant Research International. The business driver for this use case was investigated in collaboration with the WUR / Plant Research. The reported use case activities were offered for review to members of IUAC (International User Advisory Committee). With their comments the use case deliverable was finalized at the beginning of 2015.

Results	The results of the Use Cases WP are documented in two consecutive milestone reports:
	
	• MS1.1.1 Initial use case milestone ⇔ CoCo use cases deliverable v1-0
	  ○ Including the Use case workshop results
	• MS1.1.2 Intermediate use case milestone ⇔ CoCo use cases deliverable v1-1
	  ○ An extension of the initial deliverable with an elaborate description of the "DNA sequencer as a Service" use case, including the business rational
	  ○ At the beginning of 2015 this deliverable was further updated with review comments by IUAC

Observations	The use case interactions with stakeholders in the eScience community showed that there is a need for on-demand and user initiated virtual overlay networks on top of multi-domain connectivity infrastructures. CoCo is a well-positioned option to fulfil this demand and the open source & open community approach of the CoCo project is encouraged.
Recommendations	Based on the feedback from the eScience community it is recommended to pay attention to the following aspects, when further developing the CoCo service: • the feasibility and effort needed by network administrators to install a CoCo agent on OpenFlow switches and to apply appropriate network configurations, • authentication and confidentiality requirements will need to be addressed in more detail for the CoCo service, • for realistic eScience use cases the feasibility of integrating the CoCo service with on-demand data storage and compute services needs to be verified.

Table 2.1: CoCo use cases overview

2.1 Use case workshop

The results from the use case workshop are reported in detail in MS1.1.1. In this report, we summarize the conclusion of the use case workshop.

During the workshop the participants from the eScience community provided a number of suggestions, constraints and requests for use cases. Not all of the requests are in the scope of the CoCo service. For example, specific BigData solutions and the demand for combining data storage, compute and connectivity resources is beyond the scope of the CoCo project.

Added value of the CoCo prototype service is foreseen in terms of improvement of current multi-domain connectivity services regarding:

- On-demand connectivity service and enabling composition with on-demand data and compute services;
- User initiated connectivity service instantiation;
- Reusability of connectivity service solutions;
- Supporting scalable, broadband (and aggregation of) communication services;
- The solution should be affordable (exploiting infrastructure sharing / overlay).

2.2 DNA sequencer as a Service use case

In MS1.1.2 the discussion with the Wageningen University (WUR)/Plant Research International is reported.

DNA sequencer as a Service and the role of CoCo

A schematic overview of the DNA sequencer as a Service is shown in Figure 2.1. The overview shows how a genome scientist can interact with a DNA sequencer as a Service portal to set-up a CoCo instance in order to get access to the sequencer and/or stored results and processing capacity[1]. Via the portal the user can for example log-in to the portal and initiate a DNA sequencing run (step 1 in Figure 2.1). To this end, the DNA sequencing as a Service "plug-in" implements a script that can instruct the CoCo portal to (de)activate and modify CoCo instances via the REST interface to the CoCo agent in its domain (step 2). For activating the instances in other domains the CoCo portal is designed to interact with the portals in the other domains. The CoCo agent in each domain translates these instructions into forwarding rules on the local OpenFlow switches. More details about the CoCo service are given in the following sections.

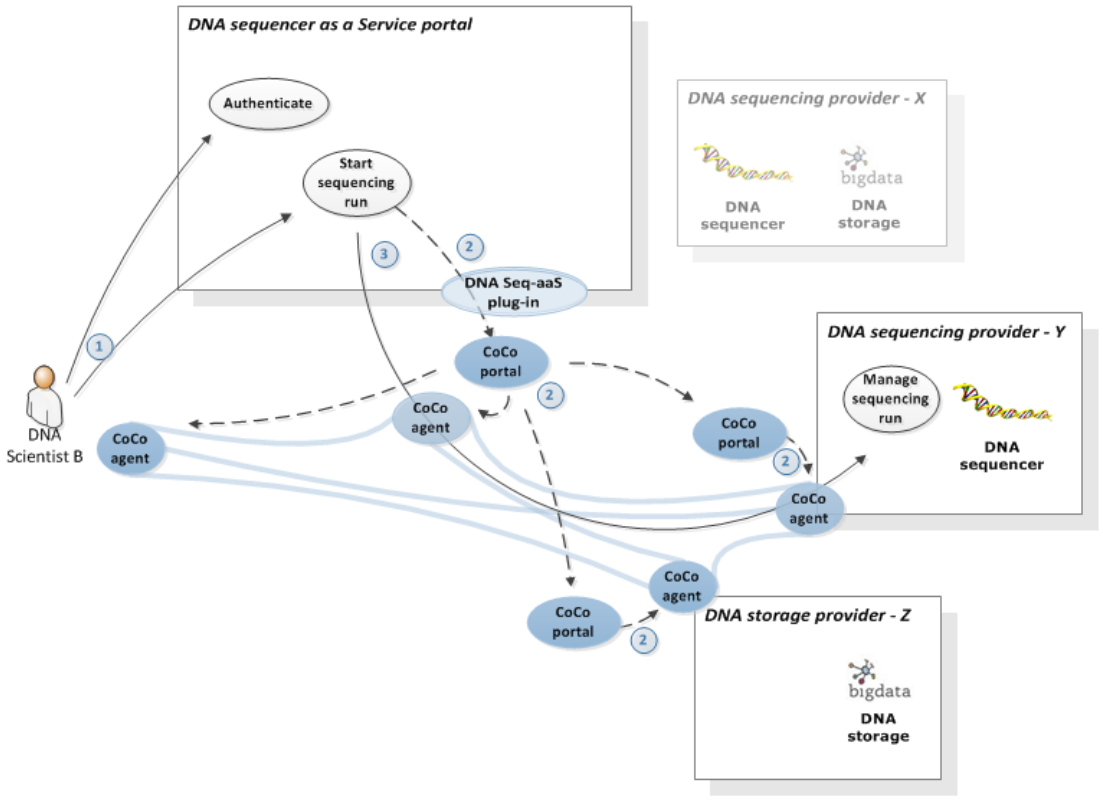

Figure 2.1: CoCo support for DNA sequencer as a Service

Once the CoCo instance is instantiated this on-demand, multi-domain virtual private LAN can be used to forward the "Start sequencing run" command to the DNA sequencer (step 3 in Figure 2.1). The CoCo instance can be used for subsequent user interactions, until the user deactivates the CoCo instance.

Business perspective

The equipment used for DNA sequencing and bioinformatics is expensive and gets outdated relatively quickly, due to current rapid developments. For example, the investment cost for upgrading the WUR's HiSeq sequencer is approximately 600,000 Euro. Due to the fast technological progress of DNA

[1] Note that these steps can only be performed after network administrators of the respective domains have installed the CoCo service for the DNA sequencer as a Service. More details can be found in the MS1.1.2 deliverable.

sequencers this investment should be exploited within a three year time frame. Investments in the required computing software and equipment are also high and their applicability is, to some extent, dedicated to the purpose of genomics research. For research organisations the overall investment costs are high and can only be justified if the (re-)utilization of the sequencers and bioinformatics solutions is sufficiently high. The opportunity to offer DNA sequencing as a Service that can be consumed by scientists from multiple institutes can strongly improve this return on investment. The CoCo service enables easy-to-use connectivity functions that are required for creating a DNA Sequencer as a Service.

This elaborated DNA sequencer as a Service use case (that is more extensively described in report MS1.1.2) clarifies how the CoCo service can be used as the connectivity pillar in such a collaborative eScience service.

3 Architecture

Section	Description of Contents
Approach	The architecture and design for the Community Connection (CoCo) service was designed based on functional requirements and suggestions from the Use Case Workshop and recent developments on OpenFlow and open standards and protocols where possible.
Results	The results of the architecture, design & development WP are documented in: • MS2.1.1 – CoCo System Architecture (version 1.4)
Observations	During the work on the design of the CoCo architecture we found many, mostly scaling, issues with Layer 2 VPNs. For a L2-VPN the issues regarding the need for MAC learning, support for broadcast and multicast, are not easy to solve. We decided to first focus on L3-VPNs for the CoCo service.
Recommendations	For authentication and authorization of the CoCo web interface we looked at a federated authentication infrastructure, but incorporating it in the design was not straightforward and did not have the highest priority. Further development of the CoCo infrastructure is needed to enhance the security of the service and ease of use. Also different roles, authorization methods and usage control and authentication of possibly different types of users need to be explored and developed within the CoCo service environment. The possibilities of and consequences for the (campus) network infrastructure at customers' sites to enable the use of the CoCo service need to be further discussed with candidate users. Different aspects of enabling the use of CoCo in production networks need to be further discussed with network administrators and operators. The demands and possible solutions for a L2-VPN service should be further explored. Possibly recent work in the IETF on EVPN [RFC7432] can help solve scalability issues with L2-VPNs.

Table 3.1: CoCo architecture work overview

3.1 CoCo system architecture

The CoCo system architecture is described in document MS2.1.1. It describes the different parts of the architecture and their interworking. It gives definitions of the separate elements that are part of the CoCo service. The design choices that were made are elaborated and the concepts on which the CoCo prototype is based are explained. In this report a short description of the overall architecture is given.

The CoCo service will consist of multiple domains. Each domain represents an NREN. Figure 3.1 shows the inter-domain architecture and the different layers defined in the CoCo system. The data plane (forwarding plane) consists of OpenFlow switches. The switches in one domain are controlled by an OpenDaylight SDN controller and each domain runs its own CoCo agent. A CoCo agent is an extension to the OpenDaylight controller and adds specific CoCo functionality to the OpenDaylight controller. 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. The CoCo agents use the BGP protocol to exchange VPN and end point information with each other. 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.

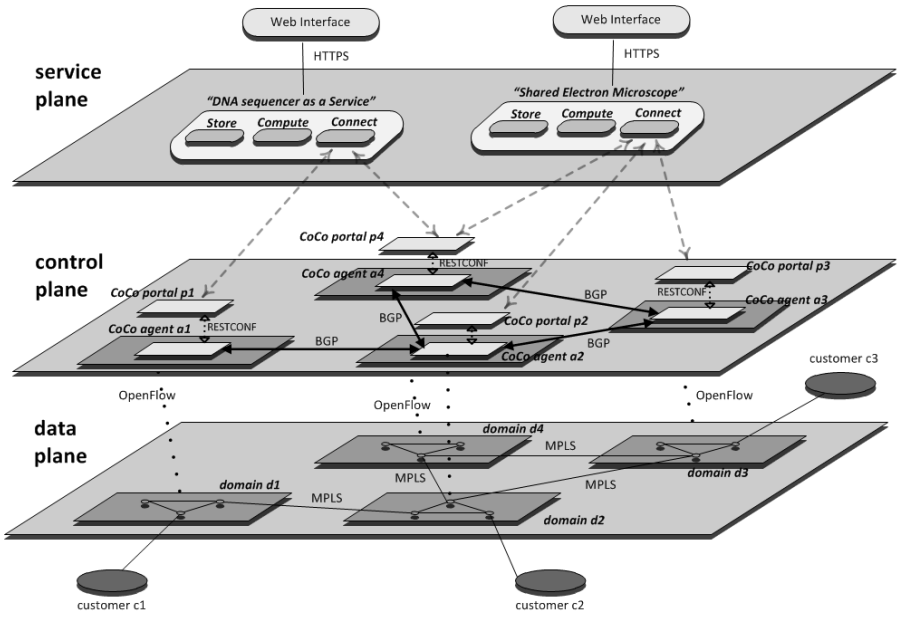

Figure 3.1: CoCo layered architecture for multi-domain L3 VPNs

In the control plane of each domain there is a single CoCo portal. End users can login to this portal and they can 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. In Figure 3.1 the service plane is also shown for the DNA sequencer as a Service as described in the previous chapter.

3.1.1 CoCo data plane forwarding and information exchange

The CoCo networks core 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 an E-NNI interface, as shown in Figure 3.2. The P switches are internal core network switches. The terminology and concepts used are specified in RFC 4026, Provider Provisioned Virtual Private Network Services [RFC4026].

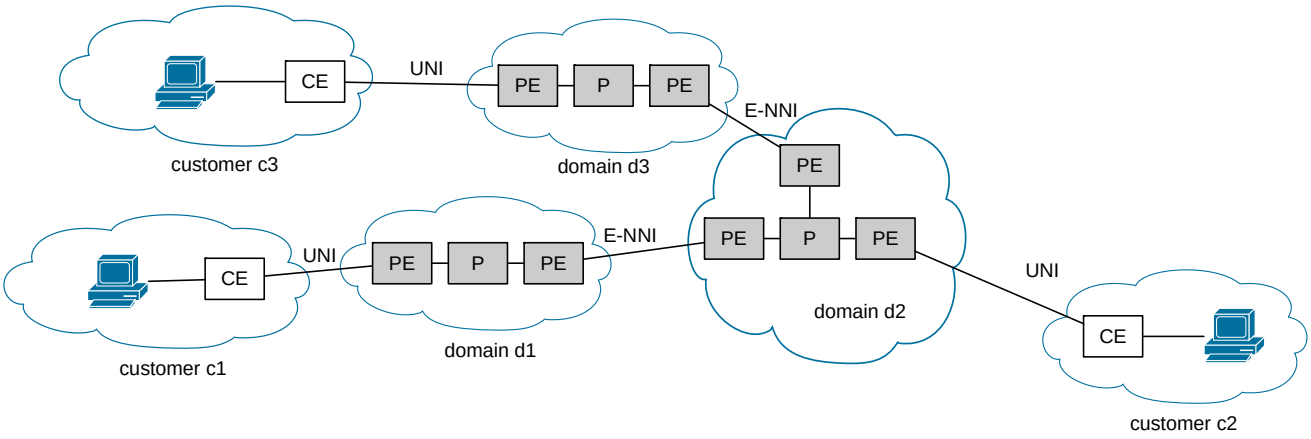

Figure 3.2: CoCo Inter-domain forwarding

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 one PE OpenFlow switch. Two MPLS labels are used. The outer MPLS label is used to identify the PE switch to which a packet must be sent. The inner MPLS label is used to identify the CoCo instance (VPN). PE switches take care of tagging the user traffic received from Customer Edge (CE) equipment with the proper MPLS labels. At the edges of the domain on the UNI interfaces towards the CE of the customer encapsulation and decapsulation takes place.

During development of the CoCo prototype it appeared that currently not all OpenFlow switch software supports the MPLS functionality required for CoCo. Therefore, VLAN based port services are used on the UNI interfaces. This means that traffic between the CE and the PE switches 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, and add the two MPLS labels corresponding to the destination PE and CoCo instance. The CoCo agent learns all destination PEs for all IP prefixes by running BGP and exchanging BGP/MPLS IP VPN information as described in RFC 4364 BGP/MPLS IP Virtual Private Networks (VPNs) [RFC4364]. The CoCo agent must also know the mapping between the customer VLAN ID and the CoCo instance and the IP prefixes used for each instance. In the implemented prototype version the users must configure this manually via the web portal.

4 Prototype

This section describes the implementation of the single domain prototype of CoCo. Table 4.1 presents an overview of the approach, the results and observations from the work on the CoCo prototype in WP 2.

Section	Description of Contents
Approach	The single domain prototype as specified in the CoCo architecture document was implemented. For the current state of the art of SDN technology the implementation of the prototype took more effort than we had anticipated, and was therefore not realised. We chose OpenDaylight as framework to communicate with the OpenFlow switches in the testbed. OpenDaylight is written in Java and has an external REST interface. Therefore we implemented the CoCo prototype as a Java web application that acts as a REST client for OpenDaylight.
Results	The CoCo prototype was implemented and demonstrated in November 2014 at SC14 in New Orleans in the GÉANT and Dutch Research Consortium booths.
Observations	CoCo is one of the first open wide area network SDN applications. As a result there is not much existing support code (in e.g. OpenDaylight) for the features that are needed in CoCo.
Recommendations	The CoCo implementation would be simpler if the SDN controller would provide functionalities like MPLS paths. It would be useful to work with the SDN controller communities (e.g. OpenDaylight and/or ONOS) to get that sort of basic functionality supported in the controller. Enhance the prototype with multi-domain support and make it interoperable with traditional BGP/MPLS VPN setups, like the MDVPN service in GÉANT.

Table 4.1: CoCo prototype overview

The implemented CoCo prototype offers a single domain L3-VPN service. It has a web interface (implemented with Apache Tomcat) that can be used by end-users to setup and tear down these VPNs. User authentication is implemented on the CoCo portal via a username / password. Implementing support for distinguishing user groups / roles did not have priority, but can be appended to the authentication scheme.

In the implementation we used existing software and frameworks as much as possible. When the project started at the end of 2013 the OpenDaylight project was rapidly gaining support from the community, especially the developers community. We started to investigate if OpenDaylight could be the base for our software. Together with GN3plus JRA2T1 we evaluated several controllers and decided in March 2014 to use OpenDaylight. The first version of OpenDaylight (hydrogen) was released in February 2014. We decided to start implementing CoCo as an external application and use the REST interface of OpenDaylight to communicate with it.

CoCo is implemented as a Java web application. It uses the OpenDaylight REST interface to retrieve the topology from the network and to send flow forwarding rules to the switches. Figure 4.1 shows the CoCo software architecture. The CoCo agent and the OpenDaylight controller run on different servers. OpenDaylight is configured as OpenFlow controller on all the OpenFlow switches in the testbed.

We use the OpenDaylight OpenFlow plugin to communicate with the OpenFlow switches. OpenDaylight uses the OpenFlow packet-out message to send Link Layer Discovery Protocol (LLDP) packets on all interfaces of the OpenFlow switches. There are flow forwarding entries on all of the OpenFlow switches to send LLDP packets to the controller. OpenDaylight receives these LLDP packets as packet-in messages. These messages contain information about the OpenFlow switch and port on which the LLDP packet was received. OpenDaylight combines this information with the information in the LLDP packet that indicates which device and port sent that LLDP packet. OpenDaylight uses this information to build a full topology database.

The CoCo *RestClient* retrieves the topology information by requesting the restconf/operational/opendaylight-inventory path. It uses JSON format for this. It also sends flow forwarding entries to OpenDaylight so that they can be installed on the OpenFlow switches. This is done via the restconf/config/opendaylight-inventory path. The *org.sun.jersey* package is used to implement the REST client.

The *Topology* class uses the *org.json.simple* Java package to parse JSON data. The Java *org.jgrapht* package is used as internal representation of the topology. Jgrapht is a Java graph package that offers a Dijkstra shortest path algorithm that we use to setup MPLS paths through the OpenFlow network. The *Topology* class uses the *Flow* class to store flow forwarding information and to generate the corresponding JSON format for OpenDaylight. The *CoCoVPN* class is used to store information about the configured VPNs.

The CoCo agent is written as Java web application and uses the Spring Model-View-Controller software pattern. It uses one controller, the *PortalController* that hands of requests for VPN changes to *VpnService.* These requests can be creating a new (empty) VPN, deleting an existing VPN and adding sites to and deleting sites from an existing VPN. These changes are stored in a MySQL database that contains information about all existing VPNs, information about all sites and information about which site is part of which VPN.

The GUI consists of two JSP files. The *portal.jsp* page is the main page and shows an overview of the network topology and information about existing VPNs. The *updatevpn.jsp* page is used to create a new VPN, to delete an existing VPN or to add sites to or delete sites from an existing VPN. These requests are handled by *doupdatevpn* in the *PortalController*.

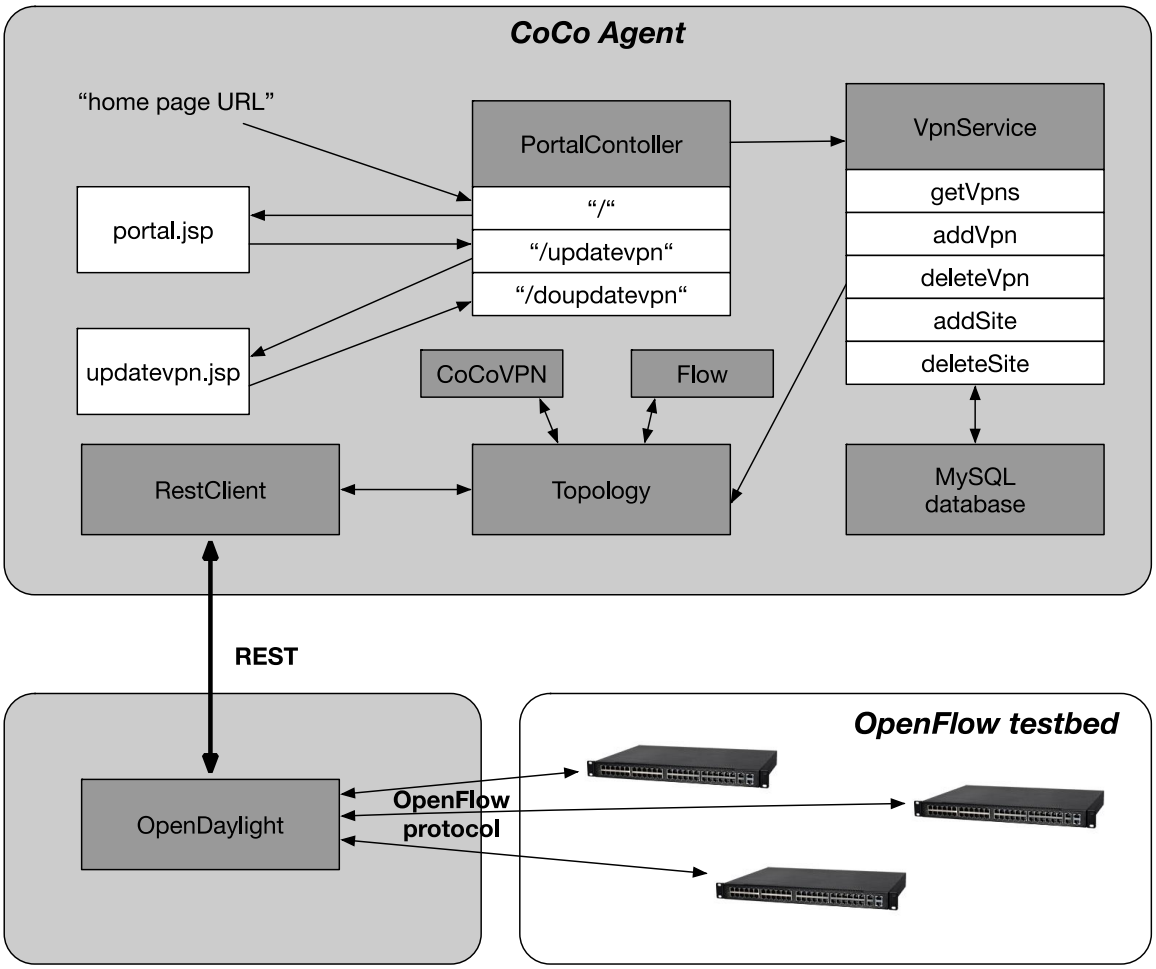

Figure 4.1: CoCo Software Architecture

5 Validation

Table 5.1 presents an overview of the approach, the results and observations from the work on experimental validation in the CoCo project (Work Package 3).

Section	Description of Contents
Approach	The first activity in the experimental validation WP was a thorough design, planning and setup of the demonstration and experiment environment, focussed on the basic CoCo functions in the single domain prototype. An automated test environment was developed for performing user-level experiments with the CoCo prototype in SURFnet's testbed. The automation of the test environment strongly improves the testability of the current and future releases of the CoCo service. In order to create more flexible validation options we also implemented a Mininet-based simulation environment along-side the testbed. Mininet emulates the OpenFlow switches and was loaded with the same OpenDaylight controller as in the CoCo testbed and it can run the developed CoCo code on top of that. In the final stage we validated the developed CoCo prototype, using the automated test environment.
Results	The results of the Experimental Validation WP are documented in two consecutive milestone reports: • MS3.1.1 Test plan for the CoCo service • MS3.1.2 Testplan and report ○ Main test conclusion: the developed prototype functionality and usability performance has been tested successfully
Observations	Execution of the testplan was postponed for several months, due to challenges in realising the CoCo prototype. Also, the testbed experiments were performed for

	the implemented prototype functionality, which are a subset of the planned test cases.
Recommendations	• For future, extended releases of the prototype the scalability and security test cases in the delivered testplan should be tested. For the scalability tests Mininet can be used. • After completion of future release(s) of the CoCo prototype it is recommended to reuse the developed testing tools and environment (for most test cases this would merely require recording of augmented user clicking sequences with AutoHotkey).

Table 5.1: CoCo experimental validation overview

5.1 Testplan and set-up

The first activity in the experimental validation WP was a thorough design, planning and setup of the demonstration and experiment environment. For this activity the functional requirements of the use cases (WP1) and the CoCo architecture (WP2) were used as a starting point. In the delivered testplan (MS3.1.1) the following test cases were described:

1. Set-up & configuration of CoCo agent
    a) network domain subscription to CoCo service
    b) CoCo agent installation by network administrator
    c) subscription of user to CoCo agent
2. User initiation of CoCo instance(s)
    a) user creates a CoCo instance
    b) user joins a CoCo instance
    c) user disconnects from a CoCo instance
    d) user requests list of CoCo instances
    e) deleting CoCo instance
3. Using CoCo instances
    a) user sends broadcast traffic via a CoCo instance
    b) user sends broadband traffic
4. Security and Stability of CoCo instances
    a) intra-domain, data plane security and stability
    b) control / data plane security and stability
    c) intra-domain, control plane security and stability
    d) inter-domain, data plane security and stability
    e) inter-domain, control plane security and stability

The focus of experimental validation was to test the basic CoCo functions and usability of these functions in the single domain prototype. In fact, all the test cases that could be run for the developed CoCo prototype (see previous section) were executed. In terms of the four categories above this means that the set-up & configuration (category 1) test cases were implicitly tested by the manual set-up in the testbed, while test cases for user initiation of CoCo instances were tested more extensively (category 2). Simple tests for verifying that traffic can be sent over activated CoCo instances (category 3) have been executed, as explained in the next subsection. Although these were not the planned L2-broadcast or broadband test cases 3a and 3b, since these were not applicable for the developed prototype. This also holds for the security and stability test cases (category 4) that made no sense to execute, yet.

5.1.1 Automated test environment

Given the user-friendliness objective of the CoCo service the validation was done at user-level. An automated test environment was developed for performing user-level experiments with the CoCo prototype in SURFnet's testbed. The software tool AutoHotKeys[2] was used to emulate interaction with the CoCo portal from a user's browser[3]. AutoHotKeys is open source software that enables recording and replaying of user browsing clicks and facilitates augmentation of such scripted sessions. By recording several browsing sessions on the CoCo portal we emulated user interactions such as the activation and the deactivation of CoCo instances. We used this scripting facility to invoke measurements within the emulated browsing sessions. For example, after emulating a user that activates a CoCo session we invoked a script that verifies that the CoCo instance indeed establishes connectivity between the VM that were selected by the user to be connected. The script was also programmed to measure the elapsed time between the user submitting a CoCo instance activation and the moment that the instance actually provided connectivity.

Note, that the automation of the test environment strongly improves the testability of the CoCo service. In fact, once new functionality is added to the CoCo prototype the developed testing tools and set-up can easily be reused. The generation of test output for the new functionality would merely require recording of augmented user clicking sequences with AutoHotkey. This is an explicit benefit of the portal-oriented design of the CoCo service and the developed testing tools.

5.1.2 Mininet

Having an environment that simulates the testbed brings additional validation opportunities. For example, scalability testing is not very sensible given the limited number of OpenFlow switches in the testbed, while simulated switches can be extrapolated to larger numbers. Also, having access to simulated OpenFlow switches is very helpful for the initial tweaking of physical testbed switches, a priori to (manual) configuration on the physical switches themselves. A pre-requisite for achieving these benefits is that the simulated CoCo service should resemble the testbed environment as closely as possible. To explore this approach we created a Mininet-based simulation environment along-side the testbed in the course of the project. Mininet (we have used version 2.2.0) mimics the OpenFlow

[2] See www.autohotkey.com
[3] In our tests we used Internet Explorer, but the test can be repeated with other browsers as well.

switches and can be loaded with the same OpenDaylight controller as in the CoCo testbed and it can run the developed CoCo code on top of that.

5.2 Test results

The results from the execution of the automated test cases is reported in deliverable MS3.1.2. The concluding test result matrix is presented in Table 5.2. As explained in subsection 5.1 not all of the planned test cases were executed. The status of the executed tests is indicated in the column titled "Tested?". The outcome of the test and a brief explanation is given in the last two columns.

	Tested?	Test result	Remark
1a. subscription to CoCo service	*Indirectly*	*Pass*	Installation of CoCo service was done and verified manually, by set-up in the testbed
1b. CoCo agent installation	*Indirectly*	*Pass*	
1c. subscription to CoCo agent	*Indirectly*	*Pass*	
2a. user creates CoCo instance	*Yes*	*Pass*	Functionally ok and the instantiation times meet the usability objective
2b. user joins CoCo instance	*No*	-	
2c. user disconnects from CoCo instance	*No*	-	
2d. requesting list of CoCo instances	*Yes*	*Pass*	Visible via the CoCo portal
2e.deleting CoCo instance	*Partial*	*Yes*	Not all combinations for deleting CoCo instance were tested (yet)
3a. sending multicast traffic	*No*	-	
3b. sending broadband traffic	*Indirectly*	*Pass*	Sending traffic is successful; not tested with broadband traffic

Table 5.2: CoCo validation result matrix

The matrix indicates that the functionality that could be tested for the developed prototype works as expected. In particular, it is shown that end-users can easily (de)activate CoCo instances and view the status of CoCo instances via the portal. As a part of the automated test sequence it was also verified that connectivity between the VM's in a CoCo instance can indeed reach each other and no more VM's than these. The functionality to join and disconnect user sites from a CoCo instance and to de-activate individual CoCo instances (as opposed to de-activating all CoCo instances at once) were implemented in the prototype after the completion of the experiments. Therefore, these functions were not tested, yet.

In addition, the test results indicate that the CoCo instance activation times will be perfectly acceptable from a user's point of view. For example, the elapsed time for a session including the steps to open a browser, deleting existing CoCo instances (and testing the effect), selecting sites and activating the CoCo instance (also including some artificially introduced user clicking time) took approximately 13s. Excluding all steps except for the actual CoCo instance activation (from pressing 'create CoCo instance' until the instance can transport traffic), the elapsed time is roughly 1 second.

Mininet simulations were used for scalability validation by extrapolating the testbed configuration into a settings with many more OpenFlow switches and / or CoCo connected sites, for example. The calibration results that compare the behaviour of the prototype CoCo service in the testbed to the Mininet simulation look very promising.

6 Conclusions

The Community Connection (CoCo) project was conducted in the frontline 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.

Building on emerging OpenFlow and OpenDaylight technology involves risks, such as missing features that still need to be developed and bugs that need to be fixed. For the unresolved issues that we encountered in our project we found the required solutions. This resulted in the implementation and successful validation of the innovative CoCo prototype that illustrates the feasibility of a user initiated, multi-domain L3-VPN service.

In the course of the CoCo project several challenges were brought to the surface. A first challenge is to complete the CoCo prototype to a multi-domain context for which it is designed. Other challenges include the extension of the CoCo service to instantiate L2-VPNs, to improve the security of the service (e.g. federated authentication, role-based authorisation) and supervisor controls on the CoCo service for network administrators.

The project partners SURFnet and TNO will continue to explore SDN technology updates and implement them in the CoCo service. The follow-up actions will include:

- SURFnet is promoting follow-up of some of the CoCo concepts in GN4 phase 1 JRA2.
- Joint work by Ericsson and SURFnet (based on the CoCo service) on a OpenDaylight VPN Service project has been accepted for inclusion in the OpenDaylight Lithium release (due June 2015).

SURFnet and TNO are discussing to continue the CoCo project as internally funded project and have reserved budget for this.

References

[CoCo]	http://www.geant.net/opencall/SDN/Pages/CoCo.aspx
[ODL]	http:// www.opendaylight.org
[MS1.1.1]	Initial use case milestone, https://intranet.geant.net/JRA0/CoCo/Shared Documents/Milestones and deliverables/CoCo use cases deliverable v1-0.docx, version 1.0
[**MS1.1.2]**	Intermediate use case milestone, https://intranet.geant.net/JRA0/CoCo/Shared Documents/Milestones and deliverables/CoCo intermediate use cases-deliverable v1-1.docx, version 1.1
[MS2.1.1]	CoCo System Architecture, https://intranet.geant.net/JRA0/CoCo/Shared Documents/Milestones and deliverables/CoCo-arch.docx, version 1.4
[**MS3.1.1]**	Test plan for the CoCo service, https://intranet.geant.net/JRA0/CoCo/Shared Documents/Milestones and deliverables/CoCo experiments deliverable v1-0.docx, version 1.0
[MS3.1.2]	Testplan and report, https://intranet.geant.net/JRA0/CoCo/Shared Documents/Milestones and deliverables/CoCo experiments deliverable v1-1.docx, version 1.1
[RFC4206]	http://tools.ietf.org/html/rfc4026 L. Andersen and T. Madsen, ”RFC 4026, Provider Provisioned Virtual Private Network Services”, 2005.
[RFC4364]	http://tools.ietf.org/html/rfc4364 E. Rosen, Y. Rekhter, “RFC 4364, BGP/MPLS IP Virtual Private Networks (VPNs)”, 2006.

Glossary

BGP	Border Gateway Protocol
CE	Customer Edge (also Customer Equipment)
CoCo	Community Connection
E-NNI	External Network to Network Interface
JSON	JavaScript Object Notation
LLDP	Link Layer Discovery Protocol
MPLS	Multiprotocol Label Switching
NOC	Network Operation Center
NREN	National Research and Education Network
ODL	OpenDaylight
P	Provider
PE	Provider Edge
REST	Representational State Transfer
RFC	Request For Comments
SDN	Software Defined Networking
SQL	Structured Query Language
UNI	User Network Interface
VM	Virtual Machine
VPN	Virtual Private Network
WUR	Wageningen University