Master's thesis

2019

Janita Marielle Johansen

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Janita Marielle Johansen

Access Management in Backbones of SDN

May 2019

UNINETT

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

UNINETT

Access Management in Backbones of SDN

Janita Marielle Johansen

Master of Science in Communication Technology

Submission date:	May 2019
Supervisor:	Otto Wittner, IIK NTNU
Co-supervisor:	Arne Øslebø, UNINETT

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Title:	Access Management in Backbones of SDN
Student:	Janita Marielle Johansen

Problem description:

In most networks today it is important to restrict access to resources, so that only those allowed can access the resource. When then number of users, resources and networks increase, managing this access control can be quite cumbersome and involve much administrative work. Software Defined Networking (SDN) brings many opportunities and possible solutions to access control schemes. Already there are many studies concerning the different ways to use SDN features to improve access control management or just general network management. Among these is a way to use the hosts/users identity in combination with an authentication service. This way the controller of the domain would know who wants to access the different resources and can make an access control decision based on their identity. However, what if the host/user moves to another domain with a different controller? Can inter-domain cooperation between controllers ensure that users retain access to the relevant resources?

This thesis investigates how SDN may improve access control management in network domains, compared to today's solutions. In particular the research focus will be on how access control management in a SDN network can work not only within one domain, but between different domains.

Responsible professor:	Otto Wittner, IIK NTNU
Supervisor:	Arne Øslebø, UNINETT

Abstract

Access control within Software Defined Networking (SDN) have already been researched, however, none of the previous research has considered several domains and cooperating access control between them. This thesis looks at such a scenario, where a user leaves it's home network - be it the user's work or university network - and connects to a partner institution's network. In this thesis the goal is for this user to be able to regain access to a resource at it's home network while the user itself is away from the network. Such a possibility could improve cooperation and collaboration between educational institutes, research facilities, or partner workplaces. While a solution to this may be Virtual Private Networks (VPN), this thesis takes it a step further and sees if it is possible to be connected to resources both at the home network and the visiting network at the same time. In this, VPN is found lacking.

In order to enable the user getting access to these resources, this thesis proposes a data model containing structured values. This data model will decide the content of messages sent between the network's controllers, such that a user may regain access. The values for the data model will be discussed, and the model itself written in the YANG language. After this the model will be validated and tested in order to make sure that the values chosen are sufficient. This validation will be executed in an emulation in Mininet, and messages made following the data model is sent between the controllers. The result from this validation shows that the user regains access to the resource in it's home network, while still retaining it's access to resources in the visiting network.

Sammendrag

Selv om aksesskontroll innenfor Software Defined Networking (SDN) allerede har blitt forsket på, har ingen av de tidligere undersøkelsens vurdert nettverk med flere domener og samarbeidende aksesskontroll mellom dem. Denne oppgaven ser på et slikt scenario, hvor en bruker forlater sitt hjemmenettverk - enten det er arbeids- eller skolenettverk - og kobler seg til nettverket hos en partner-institusjon. Målet med denne oppgaven er at brukeren skal kunne få tilgang til en ressurs som befinner seg i hjemmenettverket, mens brukeren selv er borte fra nettverket. En slik mulighet kan forberede samarbeidet, tilknytning og samspill mellom utdanningsinstitusjoner, forskningsinstitusjoner og forskjellige arbeidsplasser. Selv om en løsning på dette kan være Virtual Private Networks (VPN), tar denne oppgaven det et skritt videre og ser om det er mulig å være koblet til ressurser både i hjemmenettverket og besøks-nettverket samtidig. Her vil VPN være mangelfull.

For å muliggjøre at brukeren får tilgang til disse ressursene, foreslår denne oppgaven en datamodell som inneholder strukturerte verdier. Denne datamodellen bestemmer innholdet i meldinger som sendes mellom kontrollerne i nettverket, slik at en bruker kan få tilgang igjen til ressurser ved hjemmenettverket. Verdiene i datamodellen vil bli diskutert, og selve modellen er skrevet i språket YANG. Etter dette vil valideringen og testingen av modellen bekrefte at de valgte verdiene er tilstrekkelige. Denne valideringen vil bli utført i en emulering i Mininet, og meldinger laget basert på datamodellen sendes mellom kontrollerne. Resultatet av denne valideringen viser at brukeren etterhvert får tilgang til ressursen i hjemmenettverket, mens det fortsatt bevarer tilganger til ressurser på besøks-nettverket.

Preface

This thesis marks the conclusion of my five years at the Master's program in Communication Technology at Norwegian University of Science and Technology (NTNU). The research was carried out in the spring semester in 2019.

The goal of this thesis is to investigate whether access control can be improved, by using SDN features to make inter-domain access control possible.

I would like to thank my supervisors, Otto Wittner and Arne Øslebø, who have provided valuable suggestions and feedback throughout the whole process of writing this thesis. A great deal of thanks also has to be given to my friends and family for their input, advice and support this semester, and an extra thanks to Trine Østensen for proof-reading this thesis.

Lastly, to all the amazing people I've met in my five years at NTNU, thank you for making these some of the best years in my life.

Janita Marielle Johansen
Trondheim, May 2019

Contents

List of Figures

List of Tables

List of Acronyms

AA Authenticcation and Authorization.

AAM Authentication and Authorization Module.

ACL Access Control List.

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

ARP Address Resolution Protocol.

AS Authentication Server.

BGP Border Gateway Protocol.

CSV Comma-Separated Values.

DISCO DIstributed SDN COntrol plane.

EAP Extensible Authenitcation Protocol.

EAPOL EAP Over LAN.

GDPR General Data Protection Regulation.

HTTPS HyperText Transfer Protocol Secure.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IP Internet Protocol.

ISP Internet Service Provider.

LAN Local Area Network.

LDAP Lightweight Directory Access Protocol.

MAC Media Access Control.

NAT Network Address Translation.

NTNU Norwegian University of Science and Technology.

QoS Quality of Service.

RADIUS Remote Authentication Dial-In User Service.

RFC Request For Comments.

SDN Software Defined Networking.

SLA Service Level Agreement.

SPOF Single Point of Failure.

SQL Structured Query Language.

TCP Transmission Control Protocol.

UML Unified Modeling Language.

VLAN Virtual LAN.

VM Virtual Machine.

VPN Virtual Private Networks.

WAN Wide Area Network.

WE West-East.

XML Extensible Markup Language.

List of Descriptions

802.1x	802.1x authentication involves three parties: a supplicant S, an authenticator A, and an authentication server AS. The EAP data sent between the parties is typically encapsulated in EAPOL between the S and A and RADIUS between the A and AS.
ACL	An Access Control List (ACL) can be used for many different purposes such as filtering traffic on an interface, used in a dialer list to identify interesting traffic, used on policy based routing to make a routing decision, etc [6]. It is an implementation of a type of logic, and is also stateless [7].
AMQP	Advanced Message Queueing Protocol (AMQP) is an application layer protocol for message-oriented middleware, provides message orienting, queuing, routing both point-to-point and publish/subscribe, reliability and security [8].
ARP	Address Resolution Protocol (ARP) is a protocol that discovers the mapping between a hosts IP address and its MAC address, or more generally between internet and link layer addresses [9]. This mapping is put into a table which contains a timeout for the mapped pair.

BGP	The Border Gateway Protocol (BGP) is a standardized protocol used for exchanging routing and reachability information between autonomous systems on the internet (external BGP) or within autonomous systems (internal BGP) [10].
Firewall	Firewalls are a type of device that can examine traffic passing through a part of the network and make decisions about what to let through and what to block [6]. Firewalls also does statefull inspection and can be able to use deep packet inspection to track application-layer behaviour [11].
Hostapd	Host Access Point Daemon (hostapd) is a user space software access point capable of turning normal network interface cards into access points and authentication servers [12].
Identity-based Access Control	Access control based on the identity of the user or the users attributes, where access to different resources are assigned based on that user [13].
JSON	JSON, or JavaScript Object Notation, is a lightweight data-interchange format. It is human readable, and easy for machines to parse and generate [14, 15].
NETCONF	The Network Configuratin Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices [16]. The network operations are realized as remote procedure calls, and uses XML and YANG for the configuration data.

REST	REpresentational State Transfer (REST) is an architectural style for providing standards between computer systems on the web, in order to ease communication between them [17]. Such systems are stateless and separate between client and server [17].
RESTCONF	The RESTCONF protocol provides a RESTful programmatic interface for accessing data defined in YANG, using the concepts defined in NETCONF [18].
RFCs	Request For Comments (RFCs) are a document series that contains technical and organizational descriptions of methods, behaviours, research or innovations applicable to the working of the Internet and connected systems [19].
YANG	YANG is a language used for data modeling, and is the language used in the NETCONF and RESTCONF protocols. It models configurations and state data, and is produced and maintained by the IETF [20].

Chapter 1 Introduction

In every network, and especially in academic networks, it is important to restrict access to some or all resources. Resources may contain sensitive or confidential data, have conditional licenses, provide important services, etc. Independent of the resource, it is only meant to be accesses by certain users within the institutions. In order to enable this, access control mechanisms and schemes are implemented. The mechanisms generally used to enforce access control in today's networks are firewalls or ACLs which blocks traffic to the resource which does not match the set criteria for traffic allowed through to the resource. The traffic can be blocked or allowed based on the source and destination IP address, the different ports, protocols and so on depending on the technology used.

In some educational networks the access control for the resources are implemented by giving different groups of users IP addresses in certain ranges of IP address. For example if the students have IP addresses within one range, and the employees IP addresses within another range. This way the access control enforcers need only check whether the source IP address is within a specified range. However, in order for this to work, the users needs to be connected to the institutions network - often restricted to a physical area(s) - in order to get a correct IP address. For users outside the network their IP address will not get through to the resources. Today this can be solved by using Virtual Private Networks (VPN) which encapsulate the traffic sent from the users and send it through a tunnel to the institutions network so that the traffic looks as if it originates from within the network. When the VPN encapsulates the traffic the IP address is one in the specified range of the institution. This way the user can get access to the resources as long as it is connected to the VPN. However, VPN has it's drawbacks.

1.1 Motivation

While VPN solves a problem appearing with remote users, it can have some drawbacks. Among them being the possible loss of performance [21, 22], and difficulty to scale [21]. This makes using VPN not an optimal solution for educational or institutional networks which needs both performance and possibility to scale. In addition, VPN tunnels are unpopular among network administrators as any errors happening are generally hard to troubleshoot. Leading to more time and work being done to fix a mistake, than should be necessary.

This thesis focuses on enabling users to travel to different networks and still retaining access of resources from their home institutions. However, the user should also at the same time have access to the resources at the visiting network that are open to guests or visitors from the user's home institution. When using VPN this is difficult - if at all possible - to accomplish. While connected via VPN to one institutions the users IP address will be in that institutions specified range. This IP address will not be allowed access to other institutions resources, not even if you are within their physical network. Possible solutions to this could be to turn on and off the VPN connections to the different institutions and use the resources in order, or try to configure the VPN connection so that only certain parts of the internet traffic goes through the VPN connection. These solutions however, are inefficient and difficult to scale.

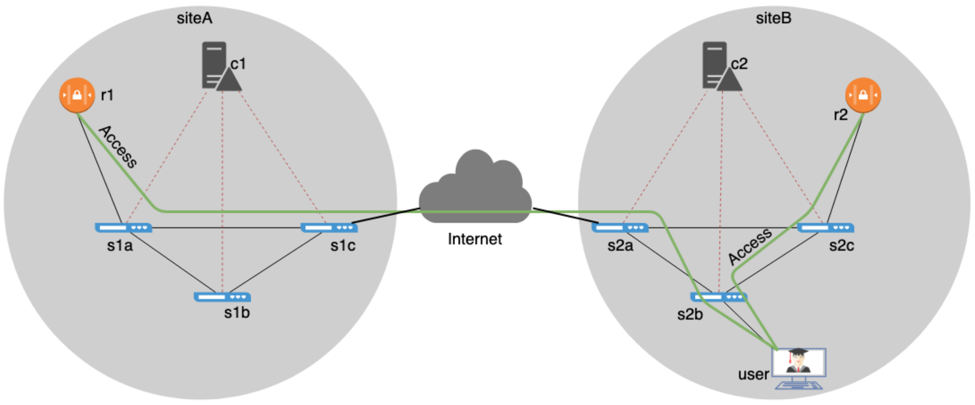


Figure 1.1: Two network domains, or sites. Each with their own network and a controller that manages the network and all access control. A user with home network at siteA moves to siteB's network. The goal it for the user to have access to both resource `r1` at siteA and resource `r2` at siteB simultaneous. For a legend to the Figure, see Figure A.1 in Appendix A.

With a scenario similar to the one shown in Figure 1.1 the user can have access to it's home resources wherever it is as long as the visiting network supports this thesis' work. This especially is an advantage in situations where VPN is not possible to use, for example when using resources at the visiting networks. This master seeks to improve the access control by adding to it. This could improve the possibility for cooperation and collaboration between institutions. Especially in educational and research institutions this is an advantage. An increased ease of collaboration can increase the innovation and the research community can flourish. Better cooperation between schools can lead to an improved learning outcome for students at the schools as exchange, of both information, students, and teachers, becomes easier and gets a lower threshold.

1.2 Using Software Defined Networking

With the emergence of Software Defined Networking (SDN), that will be described in Section 2.1.1, comes improved ways of managing network access control, handle data and generally managing networks. In addition, SDN use the OpenFlow protocol, which work on most types of hardware, making SDN a promising solution for network administrators with several kinds and brands of hardware devices. There are already several ways to use SDN to improve network management, both using and improving existing technologies and creating new ways of doing tasks. Among these are those using 802.1x, an IEEE standard which provides an authentication mechanism, to provide identity based access control in SDN networks and those creating solutions for inter-domain communication using both standard technologies and creating new ones.

Studies done on using SDN within access control management have been carried out before, and among them the ones focusing on identity-based access control are the ones most relevant to this thesis. One of the articles mentions using eduroam[1] as an application to base the access control on, as eduroam is used across the world [23] and every user has it's own username within it's domain. A common thread for all the studies done on access control within SDN is that it has been carried out within a single network, with a single controller. This thesis wishes to look at multiple domains, and so a focus has been to look at how communication between domains would work within SDN. To date, there has been some studies regarding inter-controller communications - or east-west communication, using both mainstream, long standardized technologies and creating their own new solutions. However, all of these studies only consider transmitting network and reachability information between the controllers.

[1]https://www.eduroam.org/

This thesis will build upon the features that SDN make possible and research how access control between controllers could work. It will add to the access control already available and improve it by adding features only possible by using SDN.

1.3 Objectives

The main objective of this thesis is formulated in the following research question:

> *Can a novel inter-domain access control scheme built upon Software Defined Networking improve how institutions manage user access to online content and online resources at partner institutions, compared to today's solution?*

Which will look at whether a new access control scheme can improve the access control between partner institutions, while built upon Software Defined Networking features. The improvement this thesis will attempt is a way of combining identity-based access control and multi-domain communication so that users can get access to resources no matter which domain they are in physically. The access should be decided based on the user's identity within it's home network. The thesis will look at what information must be shared or sent between different institutions, in order to retain access to resources.

To help answer the research question above the two following knowledge questions have been formulated:

1. What information has to be transmitted?
2. How can the information be transmitted?

The first knowledge question is answered by looking at scenarios and possible situations and considering all the values that may be needed and why. Then a data model is made and written in the YANG language, and validated in a Mininet emulation in order to see whether the model and it's values would be capable of making the user regain access to resources.

The latter knowledge question is answered in Chapter 2 Background and Related Works where a literary review is made to find possible answers to this question. While none of the current research use their solutions to inter-controller communication the way this thesis would like to do, it would most likely be possible to modify the solutions the research community offer to this thesis' problem.

1.4 Scope

In this thesis the focus will be on access control between two SDN domains. Users that belong to different domains should have the possibility to still have access to access protected resources at their home domains while out roaming, given that the roaming domains implements this thesis' solution.

Mostly, this thesis will be about what information has to be transmitted between the domains, and somewhat go into different ways to implement the communication. However, the focus will be on finding a standardized way to structure information communicated between the domains rather than a standardized way of implementation.

The scenarios used for validation and testing will be limited in scope, both due to the inherent ability of networks to be highly complex, and in order to create limits for this master thesis so that the work be achievable in a timely fashion. In order to limit the scope of the thesis, the rest of the report follow these points:

- Rather than set up an authentication scheme with authentication server and functional use of authentication protocols, the action of the user communication with the authentication server is mocked.
- The resources are internal within the network.
- The solution needs not be efficient, fault-tolerant or secure.

This thesis' contribution is the data model produced in Chapter 5 Research & Results. The emulation in Mininet is only made in order to validate and test the model.

1.5 Outline

Chapter 2 Background and Related Works will show the most relevant of the related works, and present shortly the technologies used or mentioned in this master thesis. Note that the glossary may also be helpful should you miss definitions for acronyms or terms.

Chapter 3 Methodology describes the methods used in this thesis. Some formalization is presented, followed with a presentation of how the master followed the method.

Chapter 4 Scenario describes the scenario used in the master thesis. This scenario shows a basic interaction between two domains, where the user has access to resources at both domains. The scenario will be used in the validation of the YANG data model.

Chapter 5 Research & Results will describe in detail the consideration in choosing the values for the data model, making the YANG model, a description of the emulation environment, and the actual emulation itself. During this chapter the final YANG model is also presented, as well as the results of the emulation.

Chapter 6 Discussion will discuss the results compared to the research goal and the related works.

Chapter 7 Conclusion will present a conclusion to the thesis.

Chapter 2

Background and Related Works

In order to get some background knowledge of this thesis, this chapter presents relevant theory and technology. It also presents relevant articles within two areas within SDN: identity-based access control and inter-domain communication. Note also, that the List of Description on page XVII presents descriptions of most technology mentioned throughout this thesis.

2.1 Theory and Technology

2.1.1 Software Defined Networking

In Software Defined Networking the network architecture is divided into three "layers" or "planes". These are the application layer, the control layer, and the infrastructure layer[1] as shown in Figure 2.1. The infrastructure layer, also called the data plane or forwarding plane, contains all the forwarding hardware such as switches and routers [24]. This hardware is controlled and managed by the controller, which resides in the control layer or control plane. The controller is a logical entity that receives instructions from the application layer and translates and relays them to the infrastructure layer. The controller can also extract topology and statistical information from the network and relay this to the application layer. The applications within the application layer uses the relayed information from the controller and can be many types of applications. It can for example include networking management, business applications, or analytic applications built to recognize suspicious network activity [1] In order to relay information between the three layers, there are northbound and southbound Application Programming Interfaces (APIs). Also called north/south communication. Between the control and forwarding layers is the southbound API, where the standard is the OpenFlow protocol. A survey on OpenFlow can be found here [25]. The northbound API refers to the API used between the control and application plane, however there is no standardized protocol for this API yet. While north/south communication is between

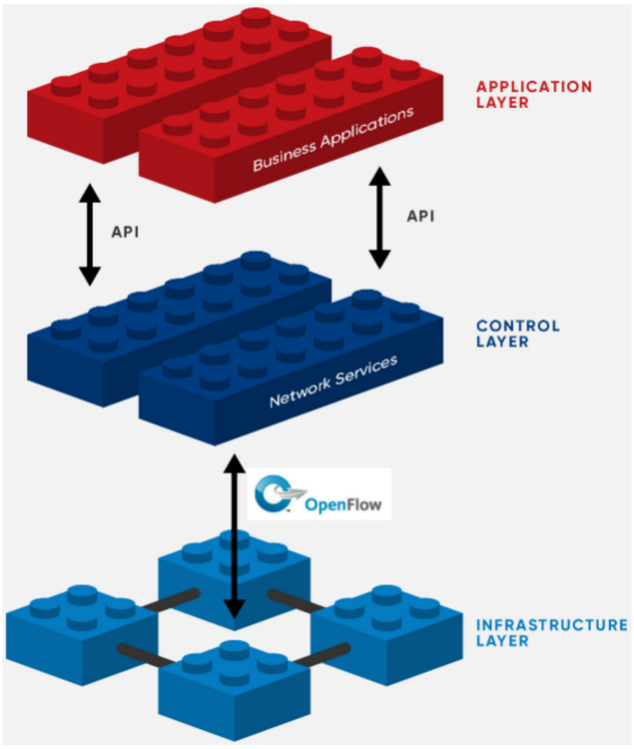


Figure 2.1: Figure showing the different layers or planes in Software Defined Networking, Section 2.1.1. Taken from [1]. Note that the bottom layer is also called data plane and forwarding layer.

the controller and either of the other layers, east-west communication is between different controllers at the control plane.

SDN has become popular mainly due to the dramatically simplification of network management and easy use of network resources in order to enable innovation and evolution [24]. Before SDN and programmable network the network devices where usually black boxes which exacerbated the challenges network operators and administrators already faced [24]. In addition to this the benefits ranges from "centralized control, simplified algorithms, commoditizing network hardware, eliminating middleboxes, to enabling the design and deployment of third-party 'apps'"[24].

On a more practical note, is the use of flow rules to manage access control and routing in the network. Each switch can have multiple flow tables with flow rules. When a packet enters the switch these flow rules and tables are checked one by one, to try to find a match for the incoming packet. The order of these rules are dependent on the priority of the flow rule, and often the very last rule matches all packets. Often this rule's action is to simply send the packet up to the controller in an PACKET_IN so the controller can decide the proper action. The controller is the one that creates the flow rules in the first place and sends them down to the switches, by using a FLOW_MOD packet. This is the way that the controller can manage and enforce the access control in a network. It may also, instead of downloading flow rules to the switches, simply let the switches send all the packets up to the controller so that the controller actively routes the incoming packets. This is done by sending a PACKET_OUT to the switch with a matching action.

2.1.2 Today's Solutions

Traditional networks generally use firewalls or ACLs to manage the access control within the network. These both contains rules and criteria for traffic allowed or not allowed, but firewalls are generally preferred as they offer more features than ACLs [6, 7, 11]. Hereafter, the access control mechanisms are simply referred to as firewall. Networks often consists of forwarding hardware from several manufacturers, each of which has it's own way of implementing firewalls. Because of this it can be hard to manage access control within networks. One of the solutions to manage such networks, and the one that UNINETT[1] uses, is Firewall Builder[2]. A similar solutions to Firewall Builder is Capirca[3], which seems to have the same functionalities and limitations as Firewall Builder [26]. There are other commodity solutions that handle complex and large networks, but these are often expensive and only works on certain hardware. For example, Cisco and Juniper have respectively the Cisco Adaptive Security Device Manager[4] and Junos Space Security Director[5]. However, these only work on the manufacturers' own hardware. There are also some open-source solutions, but again these only cover one type of hardware, often linux-based.

While the tools mentioned above works within the network, it has no features to automatically give access to hosts from outside the network. This would have to be done manually by the network administrator, which would then close the access or change it when the host changes IP address or disconnects. Firewall Builder and Capirca also do not have a built-in solution for handling NAT access control. Some network operators using Firewall Builder have separate NATs for each group of access control, i.e. employees have one and students another. So that access control is based on the IP addresses the group's NAT uses.

[1]https://www.uninett.no/
[2]http://fwbuilder.sourceforge.net/
[3]https://github.com/google/capirca
[4]https://www.cisco.com/c/en/us/products/security/adaptive-security-device-manager/index.html
[5]http://www.networkscreen.com/JS-Security-Director.asp

The closest it gets to an intra-domain access in mainstream networks today, it using VPN when in remote networks. VPN extends the private network across public networks. It generally works by encapsulating the traffic and tunnel it to the destination. At the destination it is stripped of the encapsulation and eventual encryption is decrypted. By using encapsulation and a tunnel, the traffic from the remote domain looks as if it originates from the home, or destination, network. This way a remote user can get access to a closed resource at it's home network. However as we have seen in Section 1.1 Motivation VPN has some drawbacks. Among these are the loss of performance, difficulty to scale, and the encapsulation of all traffic sent which makes access to resources at the visiting network impossible.

2.1.3 NAT

Network Address Translation (NAT) was designed to be a short-term solution for the depletion of IP addresses [27, 28]. Despite it being a short-term solution it is still used in networks today. In NAT the local addresses within the internal network are mapped to a unique global address. The local addresses need only be unique within the internal network. Traffic sent from hosts within the network gets it's header changed when passing through the NAT router, where the source IP address and possibly the port number is changed. When the packets return the NAT router changes the header back to the original. This mapping between addresses is either done 1:1, or 1:m with port numbers ensuring uniqueness within the internal network. [27, 28]

2.1.4 Information and Data models

Information models are models showing the conceptual representation of the environment: the managed objects and the relationship between them. Generally it only shows as much details as the designers need, with no information regarding underlying implementations. For information regarding implementation, protocols and sufficient details, data models are used. These contain information for the implementors on how to implement the model. The data modelling language YANG describes how data is supposed to look. It is a way to structure data and is produced and maintained by Internet Engineering Task Force (IETF). The YANG language is used to describe data sent over network management protocols such as NETCONF and RESTCONF, and this thesis has used their way of structuring data. Some of the benefits of YANG data models are that they are human readable and extensible, so that the model can be easily customized for complex or custom solutions.

2.2 Related works: Identity-based Access Control

With regards of using user/device binding for access control in SDN systems, this has been done before [29, 30, 2]. Some focus on making the system compatible with traditional network structures while some work on purely SDN networks. There has also been attempts to make these access control systems secure in different ways.

2.2.1 SDN-driven Authentication and Access Control System

Dangovas and Kuliesus [29, 30] has seen the possibility to strengthen the network security by binding network appliances and it's ports to the names and user machines using the network. In a traditional network this is not possible or at least difficult to realize, so they too have looked at SDN as a potential solution. The article [29] describes using the pre-existing authentication and account infrastructure as well as part of the network hardware to create the tight binding between user/device to the topology in the network.

The architecture of SDN allows for smart control and inspection of data packets which are sent to the controller from any switch. The article [29] proposes to track and authenticate the access layer switches and then bind users to the appropriate switch/port and effectively control their traffic policy. In an enterprise model there would be SDN switches in the access control layers, and thereafter slowly adding SDN switches to the upper layers as well in order to eventually empower the security such that the system can construct a fully-managed topology with distinguished and predictive flows for all users and services.

The authentication process was implemented as a web-based authentication interface which created an authentication request to the RADIUS server and forwards the response to the controller following 802.1x procedures. The controller then uses the response type in order to create flow entries for the switch. After authentication, the access control then follows the following steps: 1- does the user belong to a group that is allowed to reach the destination, 2- does the destination MAC address have an entry in the active users database. If the destination is local to the switch the controller provides a rule to forward traffic through interface at the same switch. If not then the traffic is destined for the default gateway or simply must be switched by a multi-protocol switch in the upper layer of the topology. Entries are deleted when either the port changed it's state, or the traffic counter decrements to zero.

The article [29] focuses on the security. They use hashed key databases on a Floodlight controller to store the active users, the user right policy and the data flow counter databases. The most important of these are the database of active users where the MAC address where used as the primary key value, in addition to the IP address, attachment switch ID and the interface that receives packets from the client.

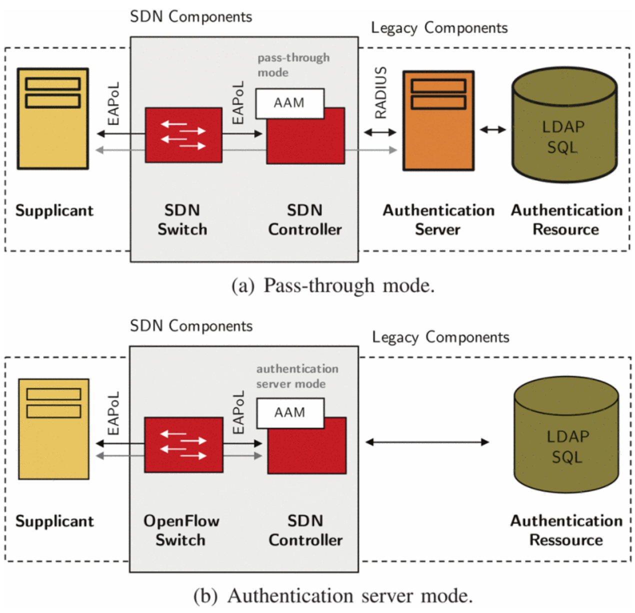


Figure 2.2: Showing the two modes of authentication presented in [2], Section 2.2.2. Taken from [2].

2.2.2 Establishing a session database for SDN using 802.1X and multiple authentication resources

This report [2] argues that while there has been several attempts using 802.1x with SDN, none has leveraged the increased flexibility of SDN. In particular that there is no common best practice to perform Authenticcation and Authorization (AA) using 802.1x for SDN. It [2] claims address mapping and web front-ends are most widely used for AA in SDN. The MAC addresses are either used as identities or they are mapped to identities. The identities are then used to fetch authorization data from a RADIUS server or an LDAP server. A limitation to this is that this identification does not verify the identity claimed by the host. An attacker can easily get a hold of a MAC address and impersonate hosts to obtain access to the network.

The Authentication and Authorization Module (AAM) is an AA module that serves as an application for a SDN controller. This will serve as an authenticator module passing EAPOL and RADIUS messages between the supplicant and authentication server (Pass-through mode (a) in Figure 2.2). It will translate the authorization data coming from the RADIUS Access Accept message into corresponding SDN rules to apply on the SDN edge switch. The article [2] also proposes an alternative solution (Authentication server mode (b) in Figure 2.2) where they don't rely on RADIUS or Diameter protocols. Rather the AAM module in the controller interacts directly with the LDAP or SQL database,or a CSV file, in order to check the AA. It essentially acts both as the authenticator and the authentication server. The article [2] also mentions the possible use of Virtual LAN (VLAN) tags, hostapd, and eduroam[6] in SDN access control.

As an improvement for providing confidentiality about the identity on the intermediate nodes, most EAP types support the concept of outer and inner identities. The outer identities is transmitted in the initialization in plain text and only serves as a forwarding hint within a distributed RADIUS infrastructure to find the RADIUS server in the home organization of the user. The actual identity of the user is transmitted within an encrypted tunnel between supplicant and authentication server. In this solution they use the outer identity in order to determine the appropriate AA resource.

This solution [2], similar to the article in Section 2.2.1, uses either port-down events or periodically checks to see whether the session is still active in the session database.

Article Name	**Technology**	**Description**
SDN-driven Authentication and Access Control System [29]	RADIUS 802.1x	Layer of switches at the bottom that catches RADIUS responses and implements the corresponding flow rules.
Establishing a session database for SDN using 802.1X and multiple authentication resources [2]	RADIUS 802.1x	Controller either acts as authenticator or authentication server. Implements flow rules based on the authentication response.

Table 2.1: Overview of the related works within identity-based access control, Sections 2.2.1 and 2.2.2. They both use the same technology with differences in implementation.

2.2.3 Limitations

The former articles both authenticate the hosts within the network based on their identity to some extent. They both also have solutions possible to implement using pre-existing authentication methods like 802.1x, as shown in Table 2.1. While the first article [29] simply looks at the type of return message the host gets from

[6]https://www.eduroam.org/

the authentication server, the second article [2] also proposes a solution where the controller directly interacts with the LDAP or database containing the authentication information.

While this would work well for intra-domain access control, it does not consider a way to work for multiple domains. This could possibly be done by expanding on the second article [2] and the database containing all authentication information. However, this would mean that all domains wanting to cooperate in this way would have to configure their 802.1x implementation to the one specified in [2]. Rather, if it would be possible to use an authentication scheme that is universally used, mainstream, then the solution would be easier to implement. Such as the mentioned eduroam, which is used globally in educational institutions [23].

2.3 Related works: Inter-controller communication

There is done quite a bit of study on the subject of east-west, or inter-controller, communication. However, most works within inter-controller communication has focused on communication between controllers in a distributed control network [31, 32, 33]. That is, several controllers within one domain. Distributed networks often improve robustness and fault-tolerance within a network as there are no single point of failure. These solutions are not relevant for this thesis.

There are however, some works describing implementations of inter-controller communication between multiple domains [34, 3]. Among these are the ones discussed below which includes a description of the implementation and workings of the solutions, but there are also some documents describing a concept[35, 36, 37] without mentioning how it could be implemented. A common thread among all these research documents are that they only focus on routing and network information being sent between the controllers.

2.3.1 A west-east bridge based SDN inter-domain testbed

This article [34] proposes a West-East (WE) Bridge mechanism to enable different SDN administrative domains to peer and cooperate. It extends the centralized control model of SDN to account for inter-domain traffic. In the testbed the feasibility of the WE-Bridge is proven by introducing two inter-domain applications: fine-granularity inter-domain routing and end-to-end QoS. The article [34] argues that while BGP has already been used to connect SDN and SDN networks as well as SDN and IP networks/domains, it still has some shortcomings. Some of which are lack of support for QoS routing, limited policy expressiveness, lacking path diversity, and lacking support for extended SDN routing-functions like fine-grain routing and local transit policies. This is the motivation behind making the WE-bridge.

When making the WE-Bridge the article [34] took into account what information should be exchanged among domains, how to exchange such information in high performance, and the northbound interface for providing such information to the applications above. Regarding what information should be exchanged, in this case network topology, this is based on the domains security levels. There are different levels of network topology view abstraction the controller can share, explained in detail in the article. The least of which being simply the links leading into the controllers domain. This virtual network information is then sent between the controllers in JSON format.

In essence the WE-bridge is a way to communicate the network topology between controllers in different domains. The WE-bridge communicates between the controllers, but also to the above-laying application via a northbound API. This way the application can use the network topology of several domains in it's application. In this way the WE-bridge promotes innovation within the SDN community.

2.3.2 DISCO: Distributed multi-domain SDN controllers

Designed for Wide Area Networks (WANs) and overlay networks, the DIstributed SDN COntrol plane (DISCO) [3] provides a channel between controllers of different domains where they can share aggregated network-wide information, and hence support end-to-end service, much like [34]. The article [3] argues that centralized SDN controller networks represents a Single Point of Failure, while the then recent proposals of a distributed SDN control plane generates large amounts of data in order to synchronize the distributed controllers. The article's [3] answer to these problems is DISCO.

The DISCO controller is composed of two parts as shown in Figure 2.3. These are an intra-domain part which contains the main functionalities of the controller, and an inter-domain part which manages the east-west communication with other DISCO controllers. The inter-domain part discovers neighboring controllers and maintain a distributed publish/subscribe communication channel. The channel builds upon the Advanced Message Queuing Protocol (AMQP) which is used both to publish information and request actions from other controllers. It should support both group and direct communications in order to exchange status information and request actions. Each controller uses this channel to exchange network-wide information with other controllers. By using this channel each controller can obtain reachability information about it's neighbor domains such that it can build a view of the inter-domain networks. Consequently it has capabilities to perform routing, path reservation, and manage Service Level Agreements.

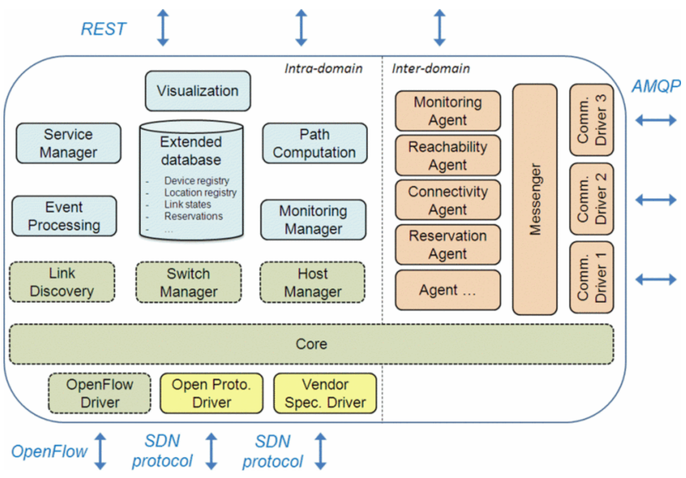


Figure 2.3: Figure showing the architecture of the DISCO controller, Section 2.3.2. Taken from [3], fig. 1.

2.3.3 Others

ONOS ICONA [38] is a peer-to-peer approach for WANs, where the load of the control plane is divided between controllers. It wishes to enable programmable networks to span multiple clusters of controllers within multiple domains. Similar to the solution presented in [34] this solution uses certain abstraction on the network view depending on who's receiving the information. ICONA gets the local topology from the ONOS core and exposes it to remote clusters, and reacts to network events that could reflect a change in the topology. Likewise, it gets topologies and updates from other controllers. ICONA can use both BGP and REST. It uses REST in a client/server peer-to-peer architecture between clusters in order to implement east-west communication, and an extension of BGP to offload the communication system to an external router and as a pure transport protocol for data exchange.

Another article [39] also mentions an inter-controller communication that uses REST APIs at the controllers. While it does not propose an implementation for this, the idea has merit as the model in this thesis is written in YANG, exactly the language used to define RESTCONF and NETCONF.

Also OpenDaylight (ODL) has proposed a solution for inter-controller communication. The ODL-SDNi app [40] uses BGP update messages to communicate between controllers, and sends network capabilities. Other solutions [41, 42] also proposes to use BGPs to ensure inter-connection between domains. These solutions exchanges BGP reachability information over a TCP connection.

Article Name	**Technology**	**Description**
A west-east bridge based SDN inter-domain testbed	WE-bridge	Sends topology and network information between controllers and northbound up to applications.
DISCO: Distributed multi-domain SDN controllers	DISCO AMQP	Controllers maintains channel to communicate with other controllers. Publish information or request actions. Sends aggregated network information.
ICONA: A Peer-to-Peer Approach for Software Defined Wide Area Networks Using ONOS	BGP REST	Sends abstracted topology and updates between controllers.
Various articles and sources	BGP	Exchange BGP reachability information over a TCP connection.

Table 2.2: Overview of the related works within inter-controller communication, Sections 2.3.1, 2.3.2 and 2.3.3.

2.3.4 Limitations

All the articles above proposes a way for controllers to communicate with each other. The solutions are different, and some articles go more in-depth of the implementation of the solution than others.

It seems that some of the articles tends to focus on using this communication between controllers where the domains they control are very similar, or even a part of the same ISP network or WAN. In that while they are controllers of separate domains, they cooperate to provide services to the users. Almost an extension of the idea of distributed controllers. This can be a limitation should the solution not work when the domains have different policies regarding routing or cooperation with other controllers. Some of the solutions also relies on a channel always being open, which can be a limitation if the channel is left unused for long periods of time while the controller still has to use resources to keep it open.

In addition, the existing research only consider using the east-west communication to send network topology, network capabilities, and routing information in general. In essence: information that helps the controllers choose the best and quickest way to forward packets from one place to another.

2.3.5 How can the information be transmitted?

This last section has described multiple ways of implementing inter-controller communication. The technology used range from BGP and REST to AMQP and the original WE-bridge. ICONA [38] even offers two ways of exchanging traffic between the controllers. In answer to the question, there is many ways of transmitting the information between controllers. One can create a WE-bridge [34] and transfer data to any other controller also implementing a WE-bridge. DISCO [3] and AMQP is also a possibility regarding how to exchange information between controllers. The same is using BGP [40, 41, 42] and REST [38, 39]. Among the different ways of implementing east-west communication, using the REST APIs may be the ones this thesis recommends the most. Partly due to it's ease of use, being long standardized and mainstream, and partly due to RESTCONF being a RESTful way to extract data from YANG based on NETCONF. The thought being that a similar solution could be made to extract data from the YANG model made in this thesis.

2.4 Summary

There has been several attempts at using identity-based access control within an SDN network [29, 2]. However, the attempts does not take into account inter-domain access control, of using inter-domain communication in order to further access control.

While the articles above has designed a way to send data between controllers in different domains, none of them has considered transmitting anything other that network data between domains.

This thesis differs from the related works in several ways. It attempts to work further on the limitations described above. In essence this thesis differs from the related works and state-of-the-art in that it combines identity-based access control and inter-domain communication in order to further the access control in the domains. This will be done by looking mainly at what information will be needed to be transmitted between the controllers. How to structure this data will also be important, as it decides how the controllers read and understand the data sent.

Forward in the thesis it is assumed that an identity-based access control scheme is used in both communicating networks, and that this scheme builds upon eduroam or a similar application of 802.1x that uses usernames per domain to identify users.

Chapter 3

Methodology

In this master thesis, different methods are used to solve different problems. In order to solve the main research problem, knowledge questions are selected. These knowledge question are made in order to help solving the design problem, or research question. The different research methods used for the different tasks are shown in Figure 3.1.

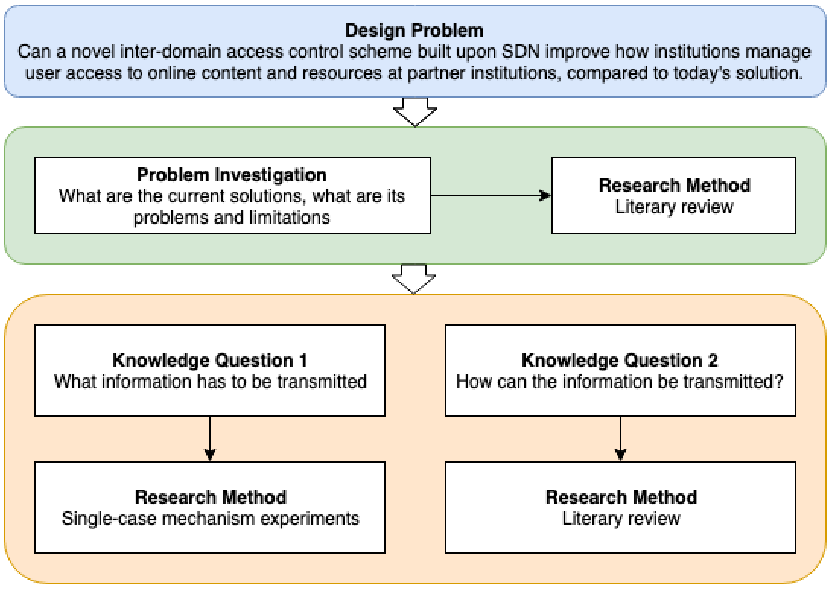

Design Problem
Can a novel inter-domain access control scheme built upon SDN improve how institutions manage user access to online content and resources at partner institutions, compared to today's solution.
Problem Investigation
What are the current solutions, what are its problems and limitations
Research Method
Literary review
Knowledge Question 1
What information has to be transmitted
Knowledge Question 2
How can the information be transmitted?
Research Method
Single-case mechanism experiments
Research Method
Literary review

Figure 3.1: The methodological structure of this master thesis. Inspired by Figure from [4] p. 220. The design problem is investigated by doing a literary review. This in order to investigate the problem context, but also to find knowledge questions that can help solve the problem.

3.1 Objectives and main tasks

The objective of this thesis was to see if it was possible to improve the access control between different SDN domains. By looking at relevant works, it was concluded that while information had been sent between SDN domains before, it was almost always network or topology information. This thesis will therefore see if it is possible to add on to the access control already existing in SDN domains, by sending information between them. This way the users visiting other SDN networks may retain their access to resources located behind firewalls against outside hosts.

The research question formalized in Chapter 1 Introduction was:

Can a novel inter-domain access control scheme built upon Software Defined Networking improve how institutions manage user access to online content and online resources at partner institutions, compared to today's solution?

and the two extracted knowledge questions where:

What information has to be transmitted?

and :

How can the information be transmitted?

Within the first knowledge question the main tasks are to decide which values are needed, make a data model in YANG, and then validating the model in a emulation.

3.2 Literature review

An important part of the work done in this master thesis has been to locate and analyze the available documents, research articles and other literature. There are several reasons to do a literature review, one of them is to form an objective summary of the literature and relevant research in the topic being studied [43]. It also helps form the basis for future research in the area, and can be helpful in defining further goals and research questions [43].

3.2.1 The literature review process

The following steps are inspired by Cronin et al. [43], which presents a step-by-step approach to undertake a literature review.

Selecting a review topic This was done early in the research process in collaboration with the supervisor and responsible professor of this thesis. It is important to choose a topic that has enough information available, but also ensure that the topic is not too broad.

Searching the literature In order to find relevant literature, Google's online academic database Google Scholar[1] was used in combination with keyword searches. Aware that keyword searches can be somewhat hit-and-miss, several different ways of phrasing was used. In addition, using the reference list of already gathered literature was especially helpful as the literature had short summaries regarding relevant works within the reference list. In addition to using Google Scholar to find relevant literature, Google's search engine was sometimes used.

Gathering, reading and analyzing the literature This began with reading the summary or abstract of the literature in order to decide the relevance of the literature. This way the irrelevant literature could be discarded and the remaining literature sorted into categories within the topic. While reading the literature, personal comments and informal keyword summaries was written down so that it would be easy to return to the article at a later time and remember it's contents. The analyzing of the literature also helped refine the goal and research questions.

Writing the review This can be found in Chapter 2, which provides a summary of the relevant works. The categories found during the former step helps with the structure of the review.

References References can be found at the end of this thesis. The references where collected during the literature search using Google Scholar's ready-made citations. In addition to this the author found a blog entry by Gray Chen [44] especially helpful in producing references for RFCs.

3.2.2 Reliability of a literature review

In order to evaluate the reliability and validity of a literature review, McNabb [45] formulates three fundamental purposes of literature review. These purposes are: 1) show the audience that the author is familiar with contributions already done to the research field, 2) help identify the key issues and gaps in the research area, and 3) assists the audience in comprehending the principles and theories that have been used in the study [45]. Literature reviews that fulfill these purposes are valid and reliable. McNabb's three purposes was taken into account when doing the literature review in this thesis.

[1]https://scholar.google.no/

3.3 Using Desing Science

In answering the research question "What information has to be transmitted?", this thesis has based the methodology on single-case mechanism experiments [4] from design science.

3.3.1 Single-case mechanism experiments

One of four research methods presented by Wieringa in [4], *single-case mechanism experiments* is based on experimenting with individual cases "in order to learn which phenomena can be produces by which mechanism [4]". These cases can be technical or social systems, or models of these systems [4]. This method is often tested in the laboratory and is useful for testing new technology, evaluating a implementation and investigating problems without effecting the real world.

In this thesis, the case is a system like the one described in Scenario A in Chapter 4, with a YANG data model describing what is sent between the two controllers. When experimenting with this thesis's case, one can change the values in the model

The following steps are inspired by Wieringa's [4] single-case mechanism experiments method, and presents the steps taken in order to execute the research.

Research Context

In this step the knowledge and any improvement goals are identified and the current knowledge is assessed.

In this master this method is used to solve the knowledge question: "What information has to be transmitted?". Therefore the knowledge goal is to find a valid and reliable answer for the question. The answer or prototype will be in the form of a data model, and the validity of the model will be tested in an emulated environment. A motivation behind the master is to improve upon the current solution with VPN, but also to improve upon cooperation and partnership between institutions. This is the improvement goal of the method. The data model produced will add to the access control between controllers. The current knowledge is assessed in Chapter 2 Background and Related Works, and the research will be done within a limited scope.

Research Problem

This step focuses on the frameworks used and further knowledge questions. Among the possible knowledge questions are: 1) What effects are produced by the interaction between the prototype and the emulated context, and why? 2) What happens if the prototype architecture is changed, and why? 3) What happens if the context is changed.

The data model will mostly be written by looking at the scenario described in Chapter 4 Scenario and considering different situations that may happen and what values needs to be present in the model in order to deal with those situations. The considerations and the reasoning for these are written in Section 5.1. The YANG language is used by the IETF and is generally accepted as a standard for data models. Additionally, while the data model will be written in YANG, the model will be validated/tested by running an emulation in Mininet with controllers running POX. These technologies are all accepted/mainstream and have been used before in educational or research projects.

To answer question 1), when the data model is validated in Mininet it is added to a general POX controller which runs an access control within it's network based on learning switches. The added functionality adds to the access control, and affects which user gets access to different resources. The effect of the interaction will be like that described in Chapter 4 and the Sequence Diagrams in Appendix B.

Regarding 2), changes to the data model changes which information is sent between the controllers, and it affects the interaction between them.

The data model is made to be minimal, standard, and easy to build upon. This is so that specific projects or companies may make their own model containing all information their solution needs. For this reason the data model is built based on Scenario A, and some general situations. Where this to change like in 3), then the data model might have had to be expanded. The emulation of the model will also be according to Scenario A and this is a minimal scenario where the environment is rather idealized. In the real world the network would most likely be more complex, larger, and outside disturbance could affect the result.

Inference Design and Validation

Inferences is reaching a conclusion on the basis of results and reasoning. In general this concerns why the conclusion reached can be concluded, and why the result is valid.

When making the data model, the values chosen are done so as they were deemed needed for the different foreseeable situations that the model may be used for. Concluding that these values are needed are based on reasoning within the situation, and looking at various sources for some similarity. Sequence diagrams are a useful tool in order to see whether there appears a need for values not yet thought of. If the conclusions made when choosing values where wrong, these would be made clear when the model is validated in the emulation. Should the model work in the emulation done in this thesis, the conclusion can quite safely be that the model contains the needed values. Because only the situation described in the scenario is validated and tested, there is some doubt as of whether the model would work in other situations. However, this is outside the scope of this thesis and can be considered part of the future works.

While an emulation often lacks the disturbances and complexity real-world testing has, it is still an approximate picture of a real-world system. In addition, the disturbances of a real-world system would most likely only affect the communication between the controllers and not the actual logic and handling of the messages by the controllers. When the prototype works in the emulation, one can (quite confidently) assume that it will work in the real-world. Perhaps with some implementation changes to fit the more complex system.

Research Execution

This step simply discusses what happened. Whether something happened, not being according to plan.

A detailed description of the making of the data model as well as the emulation setup is described in Chapter 5 Research & Results. The results of the method can also be read in this chapter.

Shortly, the method used for answering the first knowledge question (Objectives on page 4) is as follows:

- **Design**: Create a scenario the model should work in. Consider relevant considerations and other situations the model should manage. Add necessary values to YANG data model as they become needed.
- **Modeling**: Create YANG data model based on the values. Create sequence diagrams to image the model in practice. Does the model contain all necessary information for the foreseeable situations in the scenario and beyond. If not, go back to design.
- **Validation**: Test and validate the model in a emulated environment. Can the controllers give access based on information from YANG model?

The steps are inspired by the engineering cycle (Wieringa [4] p. 27-28) and is illustrated in Figure 3.2. Before these steps is the problem investigation, however this is done before the main steps are started, see Figure 3.1. The previous steps in the single-case mechanism experiments are designed to create a research method and

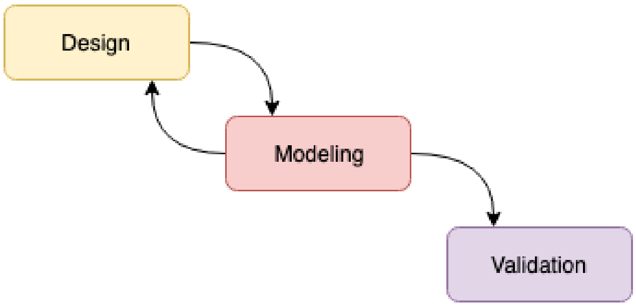


Figure 3.2: Main method steps within Research Execution. Inspired by the engineering cycle presented by Wieringa [4] at page 27-28.

ensure valid results and conclusions through the research. The Design step in the method is rather straight forward and is where the artifact or product is designed. The Modeling step involves creating the artifact and somewhat evaluating if the artifact designed would treat the problem (in essence, improve access control). The Validation step is done in order to treat the problem with the designed artifact, in this thesis this being testing and validating the model in emulated SDN partner networks. The end result of the method is evaluated in the next step, and evaluates how successful the designed artifact solves the problem.

Research Context

This step is based on whether or not the method answers the knowledge and improvements goals made in the first steps. And if it did not, whether there was any potential contribution.

This is answered and discussed in Chapter 6 Discussion and Chapter 7 Conclusion.

3.3.2 Method choice and reliability of research method

Design science is a way to solve design problems, often by cutting it down to concrete knowledge questions. Design science then iterates over solving design problems and answering knowledge questions. In this thesis there is only one iteration, but in larger projects over longer time several iterations may be needed as issues are discovered and possible improvements made obvious. This parent research method was chosen as the research question was more like a design problem, rather than a traditional research/knowledge question. Within the design problem, knowledge questions are extracted in order to solve the design problem, as shown in Figure 3.1. Single-case mechanism experiments as a research method was chosen for it's suitability to the project.

The reliability of the research method chosen and design science in general is based on it's focus on the reliability and validity throughout the process. It requires inferences and other conclusions to be reliable and valid. For single case research methods in general, Willis [46] arguments that while they do possess limitations, any research method involves necessary trade-offs. That the inherent weakness on any one method can be offset by situating them within a broader mixed-method research strategy. Willis [46] also arguments that there are several advantages to single case study analyses, among them the empirically-rich, context-specific, holistic accounts that they offer, and their contribution to theory-building and theory-testing [46].

Chapter 4 Scenario

4.1 Scenario A

The main scenario, scenario A, concerns a user from siteA currently within siteBs network regaining access to resource `r1` which resides within siteA's network. Both networks are based on Software Defined Networking.

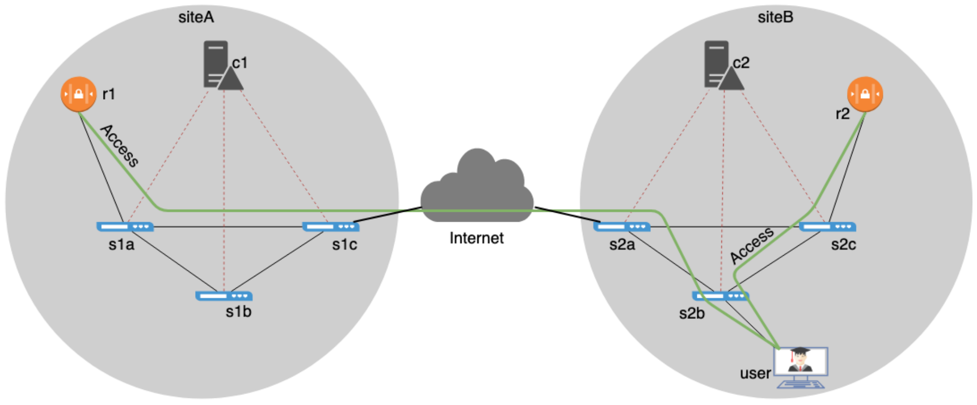


Figure 4.1: Scenario A: `user` has access to both `r2` at visiting network (Network B in siteB), and `r1` from it's home network (Network A in siteA). See Figure A.1 in Appendix A for legend.

At the beginning of the scenario, the `user` has just moved from `siteA`, it's home network, to `siteB`. Both sites has a controller that controls the network, and manages all policies and logic. The resources `r1` and `r2` are internal resources within their respective domain or site, as shown in Figure 4.1. Both resources have firewalls which blocks access for all users not within the domain. This means that the `user` will not have access to `r1` after it has moved to `siteB`. This is where this thesis proposes a solution. This thesis wishes to find a way for the `user` to have access to both `r1` and `r2` at the same time.

The data model made and described in Chapter 5 will be validated and tested based on this scenario. The scenario is quite minimal in order to have a simple proof of concept in a timely fashion. While having a more complex scenario would be closer to a real world scenario, it could risk the focus being shifted from proving the concept to implementing a real world scenario.

Chapter 5 Research & Results

What information has to be transmitted?

This it one of the knowledge questions to be answered in this master thesis. Taken from the research question, this focuses on what information is needed in order to improve the access control between controllers. In order to answer this question, a data model is first made. This model will be written in YANG. First, the considerations in making the model will be shown, as well as the final values chosen for the final YANG data model. Following this is a description of the emulation done in order to validate and test the model.

5.1 Considerations

There are many considerations when making this data model. In order to make the model as standard and over all usable in as many situations as possible, the foreseeable situations it may be used in is considered. As a result of this, there are many questions that need to be answered and discussed. This is what follows in the sections below.

5.1.1 Proactive, reactive, or a hybrid

One thing to consider when making this model is when the controllers should exchange the information. Should it happen proactively immediately when the user connects to the internet (Eduroam) such that when the user tries using the resources at it's home institution/network it is already set up in both networks? Or should it happen reactively when the user tries connecting to the resource at the home institution/network, so that either network need not execute unnecessary configurations? Perhaps some kind of hybrid is the best solution. The different models are described in detail below, and summarized in Table 5.1.

Proactive

The proactive model mainly includes configuring access in both networks when the user connects to the internet. In this model the user already has access to all resources it has access to at the home institute, when it tries to connect to them. This results in the shortest response time possible between the two networks. In addition the model can ensure that the configuration happens only once per instance, and the controllers need not do anything more until the user leaves the network. At this moment they will have to clean up by closing down and reversing the access previously configured for the user.

The biggest downside of the proactive model is the likely large number of unnecessary configurations - one per user for all it's resources, per controller - and the following cleanup the controllers have to do, when re-configuring the network to be the way it was before the user got access. This leading to useless usage of processing power possibly leading to lower general performance. Without having any specific data, it is likely that the majority of users accessing the internet at the guest network B will not access any resources configured for them. Users walking by an access point may have both laptops and mobiles that connect automatically to Wi-Fi, and not intend on using the network. This is becoming more and more likely as Eduroam - for example - is only extending further across Norway these past years. Further, if they do access a resource, it is likely not more than one or two. Ending in the controller having done many unnecessary configurations.

Reactive

In the reactive model, the controllers only exchange information when the user tries to connect to the resource. The controller in network B would then send the necessary information about the attempt to the controller in network A - the home network. The controller in network A would then configure only what is necessary for the user to reach the resource it tried to connect to. This way there is no unnecessary configurations done by either controller, and the user still gets to access the resources it wants to use.

With the reactive model the responsiveness of the network would be worse than the proactive model when the user tries to connect to a resource. However, with the general speed of the internet today this would not be a tremendous drawback. The same conclusion could be made for the amount of messages being sent between the controllers being larger as there is one exchange per resource. Something else to take into considerations with the purely reactive model is the question of how the controller in network B knows the user is trying to reach a resource at network A. Depending on the solution to this, described in Section 5.1.3, this can result in some cases of hit and miss where unnecessary messages are created and sent.

ID	Type	Description
1	Proactive	cB sends message to cA when user connects. cA configures access within networks, and sends message to cB how to do the same. cB then configures access.
2	Hybrid	cB sends message to cA when user connects. cA sends lists of resource information user has access to. cB listenes to edge switch after connection tries to these resources. After attempt by user - cB sends message to cA after which both configures access.
3	Hybrid NAT	cB sends message to cA when user connects, with the information the cB currently has. cB then sends message when user attempts to access resource outside B network, with information about users IP and port mapping. Mapping is sent with certain intervals.
4	Reactive	cB waits until user attempts connecting to resource at home network, then sends message to cA. cA configures access within network, and sends message to cB how to do the same. cB then configures access.

Table 5.1: Overview and explanation of different models types. Proactive, reactive and two hybrid models as described in Section 5.1.1. Sequence diagrams of each model can be found in Appendix B (B.1-B.4).

Hybrid NAT

A third option is a kind of hybrid that takes NAT into consideration as it would have an effect of the information needed to be sent between the controllers. With NAT, one needs to have the port number as well as the IP address the NAT uses for traffic sent from the user. If the controller at network A only had the IP address, it could allow access to everyone behind the NAT router. However, the controller at network B may not have the port number before the user sends traffic to the application outside network B. With the hybrid NAT model the controller at network B sends all the information it already has to the controller at network A, when the user connects to network B. This would generally be username, domain and IP address. The controller at network A may use this information to do pre-configurations actions to minimize computer resources needed later on, or simply be aware that the user is in network B. When the user then tries to access a resource, the controller at network B will know which port number the user traffic has. The port number may be dependent on what type of application the resource is. The controller at network B after knowing the port of the user sends this to the controller at network A along with information regarding the resource the user tried accessing. The controller at network A then finishes configurations in order to manage access.

The pros and cons with this hybrid approach depends on whether the model leans towards the proactive or reactive model. There is a choice between configuring all resources when the controller at network A has all the necessary information, or to configure each resource when the user tries accessing it. With this comes the downsides of either response time or unnecessary computer resources. It may or may not send information from controller at network A regarding the resources the user has access to, as a response to the messages from controller in network B. The number of messages depends on this, as well as the memory occupied in the edge switch in network B. In addition comes the possibility that the port number can change often. If the application or resource the user connects to changes port, then the mapping existing in both controllers must be updated immediately. This dynamicity can lead to many messages being sent and processing powers being used.

Hybrid - Proactive lean

A hybrid model with a slightly proactive approach might be the best model. While limiting the unnecessary configurations happening at network A, this model also seeks to solve the problem with reactive models regarding how to know when the user is attempting to access a resource at network A, Section 5.1.3. Similar to the proactive approach the controller at network B sends a message when the user first connects to the network. However, instead of configuring for the access control in network A, the controller at network A instead sends a list of resources the user is allowed access to in network A as well as their known IP addresses, port numbers and other relevant information. This way, the controller at network B can screen/filter traffic leaving the user matching any of these resources and notify the controller at network A about the attempted access when it happens.

A possible downside to this solution is the database the controller in network B must contain in order to have overview of the resources each user in it's network has access to. In addition the switches in network B may have larger flow tables for rules detecting traffic to resources, while the user remains in the network. In order to see how the interaction between the controllers could be in this model, see Figure 5.1.

Conclusion

When debating the pros and cons between these models the variables that popped up where the number of configurations done by the controllers, the responsiveness

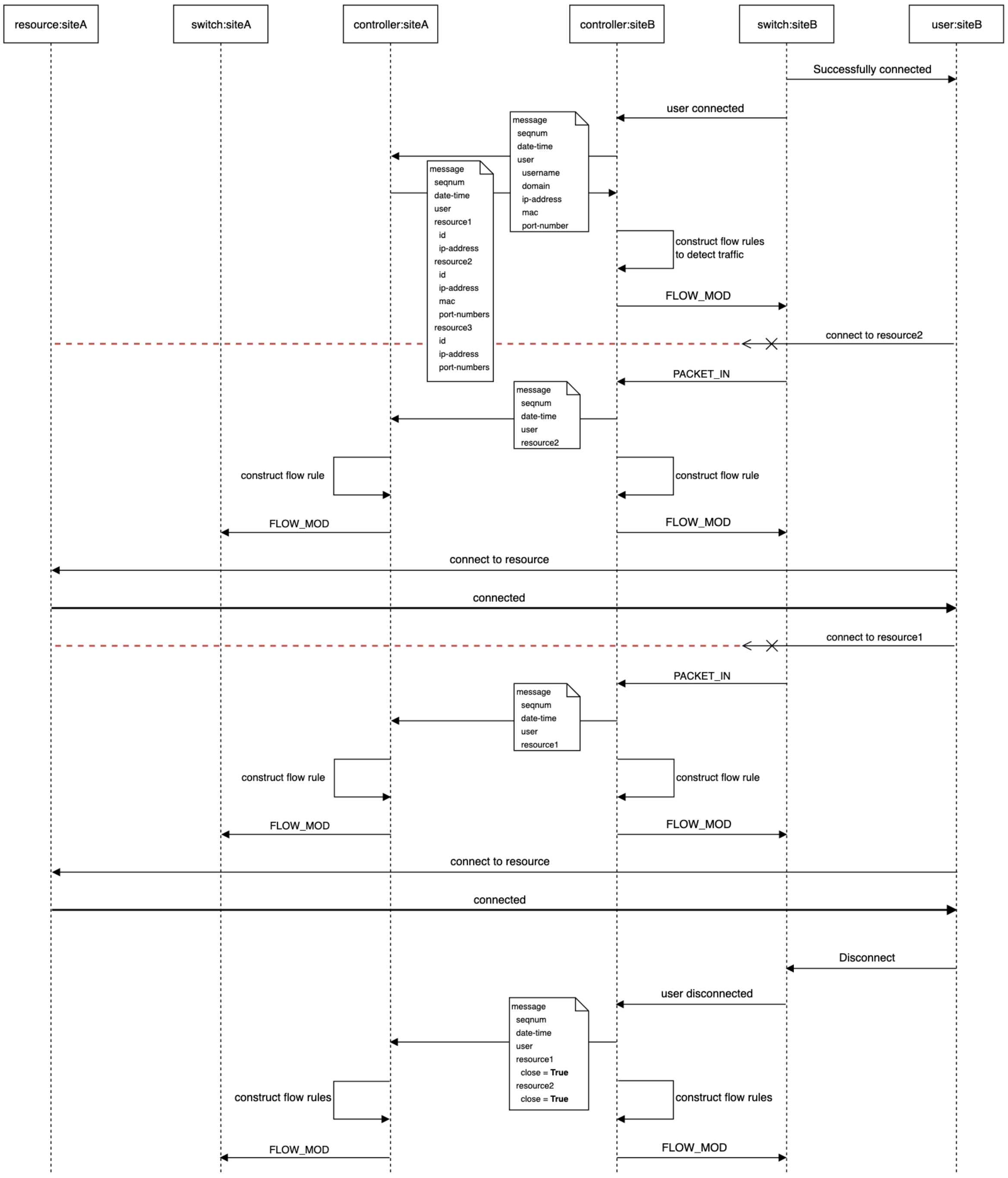


Figure 5.1: Proposal of a way to use the YANG model in a hybrid model (Table 5.1).

felt by the user, as well as the number of unnecessary configurations done by the home network.

Based on the drawbacks and benefits of the different models, in this thesis the authors chose to use the hybrid model with a proactive lean in the research further and the emulation. Note that while this may be the thesis' decision, the desire is that the final YANG model can still be used by the other models. Sequence diagrams with proposals on how all the models can use the final YANG data model is shown in Appendix B (B.1-B.4). Notice also that the sequence diagram for the model is shown in Figure 5.1 and described in the next section.

5.1.2 Detailed explanation of sequence diagram in Figure 5.1

The figure shows a sequence diagram with basis in the hybrid model described above. The first thing that happens is that the user connects to the network. Assuming the authentication and authorization scheme is 802.1x this is done with an Access-Accept message from the authentication server. The switch:siteB, either the one connected to the user or the edge switch, then notifies the controller:siteB of the user. This notification should contain either a PACKET_IN or the controller:siteB should have another way of getting the packet contents. This is necessary, because the controller:siteB needs to know the username and domain of the user. The controller:siteB uses this to find out whether the network's policy allows users from that domain to get access to remote sources. In this thesis, it is assumed that there is no such limitations. The controller:siteB then gathers all information it has about the user, and organizes it into a message in the format of the YANG model. The controller:siteB knows the username and domain from before and can get different values about the user - often at least IP address - from some kind of host detection or topology information. This message, msg1, is then sent to the controller:siteA in the other network. It is assumed that the controller:siteB knows how to transfer the message, or at least know how to get that information.

When the controller:siteA get the message it extract the information from the message. It uses the username and domain from the message in order to find which resources and services the user has access to, whether this be through an access database, user database, communicating with the authentication server, or something else entirely. The controller:siteA may also use the remaining information in the message to check against blacklisted IPs, prepare configurations, etc. After the controller:siteA has gathered all information about the resources the user has access to, this is organized in a similar message, msg2, which is sent back to the controller:siteB at the other network.

The controller:siteB then uses the information regarding the resources in order to make flow rules to detect the user attempting to access one or several of them. When the user does this, the switch:siteB notifies the controller:siteB. The controller:siteB at this point sends along information about the user and the resource(s) it attempts to access in a new message, msg3, using the YANG model structure. If the controller:siteB has any kind of restrictions on outgoing or returning traffic, it would then configure new flow rules to allow access for the user to the resource(s). When the controller:siteA gets the message regarding the user and the resource, it may configures new flow rules accepting traffic from the user to the resource(s). In the emulation later, most traffic goes through the controllers and as such the logic for accepting traffic is done within the controllers and not as flow rules. After all these configurations and flow rules are made, the user should have access to the resource.

Should the user need access to another resource at a later time, the controller:siteB would again send a message, new msg3, containing information about the user and the new resource it attempts to access. The controllers would then again configure access to allow access to the new resource. This would go on, until the user leaves the network. This may be detected in several ways (see Background and Related Works) and when the controller:siteB gets the notification it sends a new message. This message contains information on the user and the resources it has sent, as well as one or several closing arguments which tells the controller:siteA to close down all access. After this the controllers re-configures the flow rules and access so that the user no longer has access to any resources.

5.1.3 Detecting Resource Access Attempt

The proactive and hybrid models will receive a list of resources the user has access to from the controller at network A. This contains some information in order to identify the resource, presumably an IP address and possibly in addition one or several ports the resource is reached at. The controller at network B can in these models use this information to detect that the user is attempting to access a resource from the list it got from the controller at network A. The controller can either choose to make flow table rules at the switches that sends a PACKET_UP towards the controller when the user tries accessing any of the IP addresses, or if it already has a similar setup in it's edge switch it might be better to just send any traffic coming from the user up to the controller so that it can check against the list of resources. Either way, in this way it is quite easy for the controller at network B to detect that the user attempts to access a resource.

While the hybrid NAT model may follow the proactive way to get the list of resources from the controller at network A, the reactive model should be purely reactive and not contact the controller at network A before the user actually attempts access to a resource. However, this brings the question on how the controller will do so? Solutions may be somewhat try and fail based with a cached solution. Where the controller makes rules for the edge switch in the network that detect outward traffic towards ports not web-based, or several failed tries from the user to the same IP address, or knows the address is within network As address prefix, or some other detection schemes. After the controller receives information about the user attempting to access something that is likely a resource at network A, through PACKET_IN or statistics from the switch, the controller sends a message to the controller in network A regarding the user and possible resource. The controller at network A will then either answer the controller at network B with the necessary information and start the configuration in their own network, or simply send a return message telling it it does not recognize the resource. This way the controller at network B can cache whether the IP address is a resource at network A or not, and contain a cache of this for the domains it inter-communicates with.

5.1.4 Discontinuing Access

Whether the controller at network A configures access to all resources at once or one at a time, it still has to shut all access down when the user leaves network B. If not, a new user not supposed to get access to the resources may be allowed access due to having gotten the same IP address as the user that left. Therefore the flow rules installed at the switches in the network must be deleted or changed. However for this to be done, the controller at network A as well as the controller at network B must know when the user leaves. Depending on how much control the controller at network A should have, this could be done in two ways. In both cases the controller at network B has to know when the user leaves. This can be done by the switch regularly checking whether the user is still connected. The controller at network B then knows when the user leaves and can discontinue the access at network B. The controller at network A can either regularly get insurances that the user is still in the network and have an own timeout deadline, or can get one message at the end when the user has left network B.

5.1.5 Open Edge Switch

The edge switches at networks can either be open or closed to some degree. This in whether or not the switches stop certain traffic going in or out. Usually in the cases when the switches are closed, this is for traffic with unusual/non-mainstream port numbers. The switch being open or closed matters when discussing whether both controllers has to do configurations, or just the controller containing the resource. This affects the necessary information sent between the controllers in the different messages.

5.1.6 Two-way Traffic

Some scenarios may have resources where the traffic goes both ways, so that the switches in both networks have to open up to more IP and port mappings. Such two-way traffic is mostly used on communication and chat resources, and remains a small part of the general traffic. Due to web-traffic being the most common resources, and only being one-way this thesis will focus on this kind of traffic. However, this can be an interesting issue for future research.

5.1.7 Network Address Translation

While NAT has already been discussed, it is worth considering the possible information needed in the YANG model to support a NAT based model. IP addresses and port mapping has already been discussed earlier in Section 2.1.3, but the rate of mapping changes is also important. This is dependent on the type of resource the user is connected to, as some resources may have a dynamic port and change ports often and the NAT mapping has to change to keep up with the resource. When the NAT mapping changes, both controllers has to be made aware. This is because when the port mapping changes, the user is no longer behind the IP and port combination. However, a new user might have been assigned the combination and can get access to resources it should not be able to access. Should the port-change change often and quickly, the controllers has to keep up with the changes and send the update to the other network's controller. There has to be some kind of update or keep-alive, and the controllers needs to be able to see how long ago the message with information came.

5.2 Model Values

The model should contain values that are necessary for the most common scenarios. Like other YANG models [47], this can easily be extended to cover any vendor-specific configurations. One also has to consider that the traffic between controllers may go both ways: from visiting network controller to home network controller in order to tell the home network that a user from your domain is within it's domain or that the user is trying to reach a resource; from the home network controller to the visiting network controller either in case the visiting network must open traffic to a certain port range or IP address or if it needs to be aware of the resources the user is allowed to access.

For both directions, it is important to know which `user` the interaction is concerning. For this, we need at least `username` and `domain` in order to uniquely identify the user, and to know which controller to contact regarding the users access. In addition the `IP-address` of the user may be needed so the home network controller can open the network for traffic from that IP address, and if needed configure access to the

```
module: intercontroller-access-control
  +--rw message
     +--rw seqnum          uint32
     +--rw date-time       yang:date-and-time
     +--rw user
     |  +--rw username                      string
     |  +--rw domain                        string
     |  +--rw (ip-address)
     |  |  +--:(ipv4-address)
     |  |  |  +--rw ipv4-address?           inet:ipv4-address
     |  |  +--:(ipv6-address)
     |  |  |  +--rw ipv6-address?           inet:ipv6-address
     |  |  +--:(ipv4-prefix)
     |  |  |  +--rw ipv4-address-prefix?    inet:ipv4-prefix
     |  |  +--:(ipv6-prefix)
     |  |     +--rw ipv6-address-prefix?    inet:ipv6-prefix
     |  +--rw mac?                          yang:mac-address
     |  +--rw port-number?                  inet:port-number
     |  +--rw close-all?                    boolean
     +--rw resources* [id]
        +--rw id                            string
        +--rw (ip-address)
        |  +--:(ipv4-address)
        |  |  +--rw ipv4-address?           inet:ipv4-address
        |  +--:(ipv6-address)
        |  |  +--rw ipv6-address?           inet:ipv6-address
        |  +--:(ipv4-prefix)
        |  |  +--rw ipv4-address-prefix?    inet:ipv4-prefix
        |  +--:(ipv6-prefix)
        |     +--rw ipv6-address-prefix?    inet:ipv6-prefix
        +--rw mac?                          yang:mac-address
        +--rw port-numbers*                 inet:port-number
        +--rw close?                        boolean
```

Figure 5.2: The YANG model, shown in tree format for readability. For information about the tree diagram syntax, see Section 5.2 or [5].

resource within the network. This IP address can be in several formats to cover most network situations: `IPv4-address`, `IPv4-prefix`, `IPv6-address`, and `IPv6-prefix`. The `mac` address and `port-number` can also be used to configure access for the user.

The messages also need to consider the `resources`. Whether the message is from the home network controller to the visiting network controller and containing a list of all resources the user has access to, or going in the other direction to inform the home network controller of the user attempting to access a resource. Each `resource` has an `id` to identify them, and this value could also be used to distinguish between categories of resources dependant on the implementation of the model. The `resource` also has several other values in order to recognize when the user attempts to access the resource or to configure access at both networks. These values are similar to the ones describing the user: `IPv4-address`, `IPv4-prefix`, `IPv6-address`, `IPv6-prefix`, and `mac` address. It also has the possibility to list all `port-numbers` related to the resource.

In addition to the values characterizing the users or resources, there are the `date-time` which is the full date and time of when the message was sent, and the `seqnum` which is the sequence number of the message. Both of these are used to be able to ensure no mix-up in messages, or that missing messages are able to be requested again. Also, the timestamp can be useful in NAT or other dynamic scenarios where the controllers needs to send several messages to update the state. Furthermore, the messages also contain the booleans `close` and `close-all` values which tells the controllers to either close all accepted connections and configurations made for that specific user, or to close down access to that specific resource.

The final model can be seen in tree form in Code 5.2 at page 38. In addition, the YANG code for the model is uploaded to the master thesis' Github page[1]. Note that the message separates the user and resource information, while date-time and sequence number is mandatory for every message. Note also that in the tree format the question mark `?` behind the value names are not mandatory, and that the asterisk `*` means the value is a list and the value in brackets `[]` behind it is the key for the list if it has any. The text a couple `tab blocks` behind the value is the type of the value. While `string` and `uint32` are built-in, the types starting with `inet` and `yang` are defined in RFC 6991 [48].

5.3 Making the YANG model

When making the model it was first written in YANG, then validated by Pyang[2] and tested by Pyang's PyangBind[3] plugin. Regarding writing the model in YANG there was several RFCs documents that were useful. Among these where RFC6991 [48] which contained many types needed in the model, and RFC6020 [20] which describes the YANG language and provides examples on how to use it.

While programming the model in YANG, Pyang was used to validate the model as it was. PyangBind was used to test the functionality of the model underway. In addition, Pyang has the functionality to create not only tree models, but also UML diagrams and XML skeletons of the YANG model. These can be seen in Figures C.1, C.3 and C.2, respectively, in Appendices.

[1]https://github.com/janitamjohansen/intercontroller-access-control
[2]https://github.com/mbj4668/pyang
[3]https://github.com/robshakir/pyangbind

5.4 Validation of the model

This emulation is done in order to create a proof of concept, and to implement and validate the YANG model and it's values. The validation is based on the scenario described in Chapter 4 Scenario and the hybrid model described earlier in this chapter, see Sections 5.1.1 and 5.1.2. The sequence diagram this emulation is based on is the one shown in Figure 5.1, and it shows the expected interaction between objects in the hybrid model. All files used in the emulation can be found at this master thesis's github page[4], as well as a step-by-step guide to running the emulation.

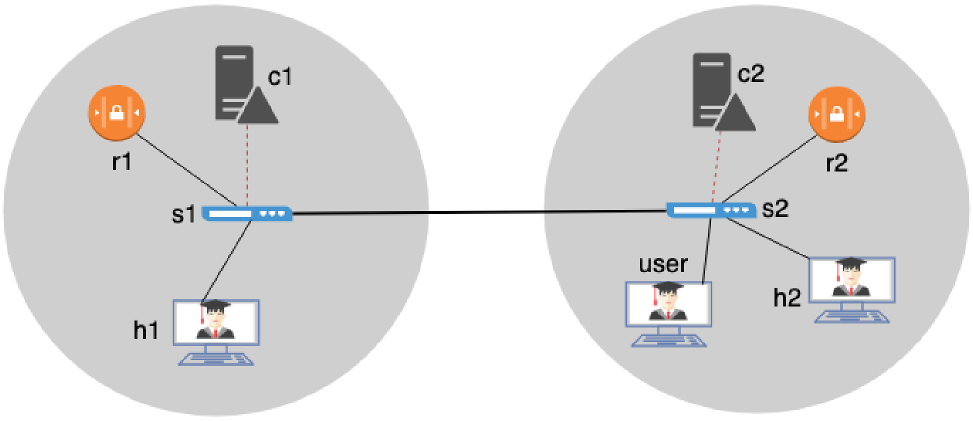


Figure 5.3: Showing the topology used in the emulation. The two controllers are running POX, and the switches are Open vSwitch. See Figure A.1 for legend to the figure.

Technology

The validation is emulated in Mininet[5] with POX[6] running on the two controllers. The switches are Open vSwitches. Mininet is an easy-to-use emulator which creates a network of virtual hosts, switches, controllers, and links. It was designed to be a tool for education and research and therefore seemed the perfect emulator for this thesis. The documentation available for simple use of Mininet is also freely available. Mininet is compatible with POX and can connect the virtual controllers within Mininet to remote controllers which runs on another machine. POX is a python-based controller which is much used in the research community. It's popularity has decreased in the last years, but the documentation available is still current. POX also has less steep learning-curve than other controllers, something which was considered when choosing controllers. For further research, a different controller might have been chosen as

[4]https://github.com/janitamjohansen/intercontroller-access-control
[5]http://mininet.org/
[6]https://github.com/noxrepo/pox

there are some controllers with more support both for inter-controller communication in complex domains and from the research community. For this thesis' validation through emulation, POX was concluded to be sufficient.

The POX controllers communicate with their switch by using OpenFlow (1.0), as the standard for southbound API. As for the communication between the two controllers, this was done simple by using HTTP POST request messages. The controllers listen to ports, and send POST messages containing the YANG messages in JSON format to each other by using the other's IP addresses and port numbers, as well as a POX stock component providing a webservice.

Topology

The topology used in the emulation is similar to the one shown in Scenario A, with some differences. See Figure 5.3 for an illustration of the topology. The topology is made in Mininet by running the custom topology setup with the custom.py file shown in Figure C.4 in Appendix C. The file is also available on this master thesis's github page. The validation's scenario has, unlike Scenario A, only one switch in each network. This is due to some problems early on in the making of the emulation, and considering that the number of switches has little effect with the end result of this emulation. It also has a direct line between `s1` and `s2` instead of going though `"internet"` or a router, and two hosts, `h1` and `h2`, representing other user within the network. These hosts should get access to the resources within their own network, but not the ones outside their network. In this scenario both switches are open for all traffic except traffic from outside the network with destination to the network's resources. This can, however, easily be changed by filtering further in the code should one wish to have a more closed network. Note that while traffic towards resources are blocked, the resources are allowed access to the hosts and users.

The code and limitations

The code running on the controllers started out as POX's `l2_learning.py`[7] code for forwarding packets on the layer two plane, so that traffic not included in the validation of the model were routed normally. Other POX stock components used in this emulation are: `log.level`[8] and `samples.pretty_log`[9] in order to debug the code during writing, `webcore`[10] in order to start a webserver within the POX process, and an altered `openflow.webservice`[11] to handle all incoming POST requests sent to the controllers.

[7]https://github.com/noxrepo/pox/blob/eel/pox/forwarding/l2_learning.py
[8]https://github.com/noxrepo/pox/blob/eel/pox/log/level.py
[9]https://github.com/noxrepo/pox/blob/eel/pox/samples/pretty_log.py
[10]https://github.com/noxrepo/pox/blob/eel/pox/web/webcore.py
[11]https://github.com/noxrepo/pox/blob/eel/pox/openflow/webservice.py

The code created is not meant to be optimal and elegant, and the solutions done in the code are meant to make it work with no regard as to whether it is the most elegant solution.

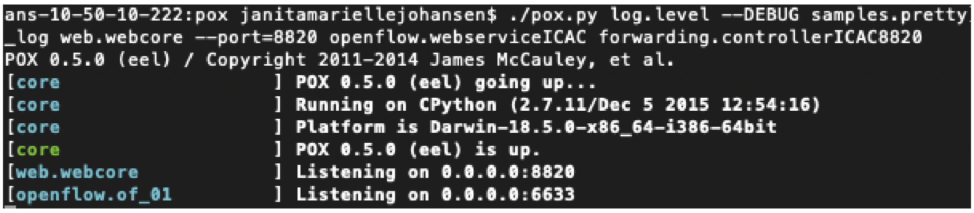
The closing or shutting down of access after the user leaves the network has not been simulated in this emulation. The YANG data model does however employ several ways of closing - the use of which are shown in the sequence diagrams in Appendices. Where the discontinuing of the access to be simulated, the controllers had only to check incoming messages whether the `close` or `close-all` fields were `true` or not. Had they been true, the home controller had to remove the access it previously had configured/added. In the code this could be done my removing the elements in the list of allowed connections.

The messages sent between the controllers are made ahead of time, and fetched by the controllers when sending the message. In addition, the IP addresses sent and used are in IPv4 format. This is mostly due to the default values of Mininet. MAC addresses were also added to both the messages, but port numbers where not, as with such a minimalist scenario that even just one of the IP and MAC addresses is enough to uniquely identify a resource and configure their access. Note that this means that the flows or traffic coming from the hosts in the network can be identified by either address, which is clear in the code where the addresses used differs to show that both are possible to use. However, for allowing access to the resource and detecting the user, the user's IP address is used. Note also that traffic trying to reach the resources are blocked or allowed by the controller in the network, rather than a separate firewall application.

Interaction

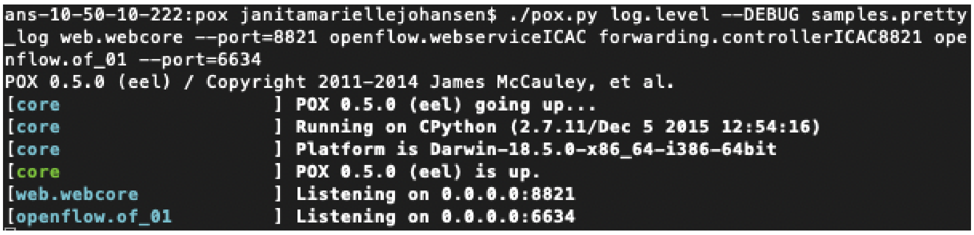
The scenario coded in the emulation is the one described in Scenario A. In this scenario, the user at siteB should get access to both resource `r1` at siteA and `r2` at siteB. For this reason the two controllers are running different codes. One serves as the visiting network controller, `c2` in Figure 5.3, and informs the home network of the user's arrival and attempts to reach the resource. The other, `c1` in Figure 5.3, serves as the home network controller and informs the visiting network of the user's resources at their home network. Outside the simulation this could go both ways by simply implementing both the controllers' logic.

The controller acting as the visiting network controller has to pay attention to when the user connects to the network, or as it is done in the emulation: the user tries contacting the authentication server. When this happens, the controller sends message 1 `msg1` (Figure C.5) that contains information about the user. In this emulation it is assumed that the controller knows how to reach the controller at the home network, and that it can extract which domain the user comes from - by for



```
ans-10-50-10-222:pox janitamariellejohansen$ ./pox.py log.level --DEBUG samples.pretty
_log web.webcore --port=8820 openflow.webserviceICAC forwarding.controllerICAC8820
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
[core                    ] POX 0.5.0 (eel) going up...
[core                    ] Running on CPython (2.7.11/Dec 5 2015 12:54:16)
[core                    ] Platform is Darwin-18.5.0-x86_64-i386-64bit
[core                    ] POX 0.5.0 (eel) is up.
[web.webcore             ] Listening on 0.0.0.0:8820
[openflow.of_01          ] Listening on 0.0.0.0:6633
```

(a) Controller `c1`. OpenFlow `c1 <-> s1` communication listens at port 6633 and webcore `c1 <-> c2` at port 8820.



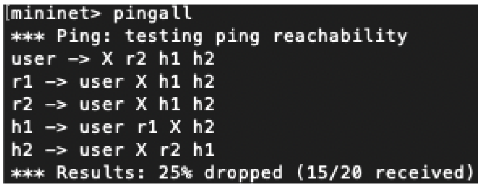
```
ans-10-50-10-222:pox janitamariellejohansen$ ./pox.py log.level --DEBUG samples.pretty
_log web.webcore --port=8821 openflow.webserviceICAC forwarding.controllerICAC8821 ope
nflow.of_01 --port=6634
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
[core                    ] POX 0.5.0 (eel) going up...
[core                    ] Running on CPython (2.7.11/Dec 5 2015 12:54:16)
[core                    ] Platform is Darwin-18.5.0-x86_64-i386-64bit
[core                    ] POX 0.5.0 (eel) is up.
[web.webcore             ] Listening on 0.0.0.0:8821
[openflow.of_01          ] Listening on 0.0.0.0:6634
```

(b) Controller `c2`. OpenFlow `c2 <-> s2` communication listens at port 6634 and webcore `c2 <-> c1` at port 8821.

Figure 5.4: Both controllers are instantiated by running the commands at the top of the figures. Figure (a) have no port parameters for OpenFlow switch-controller communicating due to port 6633 being the default value.

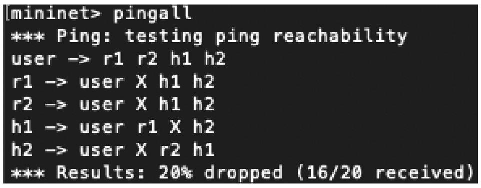
example looking at the authentication messages. After this the other controller will send message 2 `msg2` (Figure C.6) which will add all resources the user has access to in siteA to a resource list. This list will be checked every time the user sends traffic to check whether the user tries to reach a resource outside the network. Note that this is the same as if flow rules were made to match packets with destination address equal to one in the resource list, only that now the packets go through the controller instead. When the user tries to access a resource, the controller sends message 3 `msg3` (Figure C.7) which informs the other controller that the user is attempting contacting this specific resource. The other controller then changes the access control so that the user can communicate with the resource.

The controller acting as the home network controller has to wait for message 1 `msg1` from the other controller. It then sends message 2 `msg2` back. When this is done, it waits for message 3 `msg3` which signifies that the user tried contacting one of the resources. `msg3` contains the IP address of both the user and the resource it has tried to contact, and the controller uses these to make a pair in the list for allowed connections. This list is checked when traffic tries reaching resources. After this is done, the user will be able to send traffic to the resource.



```
mininet> pingall
*** Ping: testing ping reachability
user -> X r2 h1 h2
r1 -> user X h1 h2
r2 -> user X h1 h2
h1 -> user r1 X h2
h2 -> user X r2 h1
*** Results: 25% dropped (15/20 received)
```

(a) Pingall before user has access



```
mininet> pingall
*** Ping: testing ping reachability
user -> r1 r2 h1 h2
r1 -> user X h1 h2
r2 -> user X h1 h2
h1 -> user r1 X h2
h2 -> user X r2 h1
*** Results: 20% dropped (16/20 received)
```

(b) Pingall after user has access

Figure 5.5: The two figures show the result of two pingall commands at different times - before and after the user gets access to r1. The tables in the figures show which hosts a host can reach, by showing either the hosts' name or a simple X to signify a dropped packet. For example we see that the user in (a) can reach r2, h1 and h2, but that the ping sent to r1 was dropped. The last line shows the percentage of packets dropped.

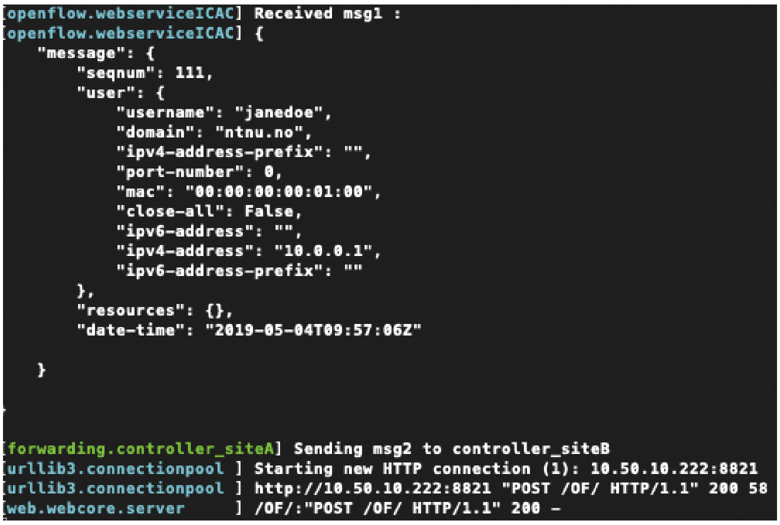
5.4.1 Emulating

Mininet runs on a virtual machine, but can be reached via ssh from a terminal. The two controllers are both running locally, listening to different ports - one for OpenFlow communication with the switch and one for web-communication between the controllers. As shown in Figure 5.4 `c1` listens to port 6633 and port 8820, while `c2` listens to port 6634 and 8821. The custom topology created for Mininet creates controllers that run remotely, in this case locally on the machine running the virtual machine with Mininet, on the port mentioned specific to that controller. The topology is run in Mininet, and the network is set up.

The emulation and validation of the model in this thesis was executed on a MacBook Pro running macOS Mojave version 10.14.4. The Mininet VM image used was Mininet 2.2.2 on Ubuntu 14.04 LTS - 64 bit fetched from the Mininet project's webpage[12], and the VM ran in Oracle's VirtualBox 6.0.4. The Mininet VM and VirtualBox was set up according to the steps described in [49].

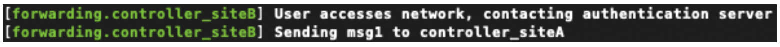
In order to mock traffic and test the access each host has to each other, the `pingall` command is used. This is a function in Mininet that sends ping packets between all hosts in the network, and shows the result both in a table form and in a summary of packets dropped and received. Figure 5.5a shows the result of the `pingall` before any message is sent between the controllers. From this it is clear that no host that is not within the network has access to the network's resource, but that they are free communicate with each other. Note also that ECHO REPLY packets are generally allowed past. This is the reason the Figure 5.5 shows the resources having successful pings to hosts that do not have the same result the other way.

[12]https://github.com/mininet/mininet/wiki/Mininet-VM-Images



```
[openflow.webserviceICAC] Received msg1 :
[openflow.webserviceICAC] {
   "message": {
       "seqnum": 111,
       "user": {
           "username": "janedoe",
           "domain": "ntnu.no",
           "ipv4-address-prefix": "",
           "port-number": 0,
           "mac": "00:00:00:00:01:00",
           "close-all": False,
           "ipv6-address": "",
           "ipv4-address": "10.0.0.1",
           "ipv6-address-prefix": ""
       },
       "resources": {},
       "date-time": "2019-05-04T09:57:06Z"

   }


[forwarding.controller_siteA] Sending msg2 to controller_siteB
[urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8821
[urllib3.connectionpool ] http://10.50.10.222:8821 "POST /OF/ HTTP/1.1" 200 58
[web.webcore.server      ] /OF/:"POST /OF/ HTTP/1.1" 200 -
```

(a) Controller at SiteA



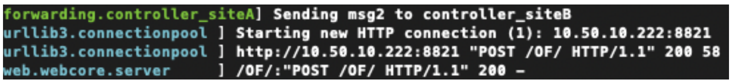
```
[forwarding.controller_siteB] User accesses network, contacting authentication server
[forwarding.controller_siteB] Sending msg1 to controller_siteA
```

(b) Controller at SiteB

Figure 5.6: Step 1. The output of both controllers, running locally on terminals. The user at siteB tries contacting the authentication server, which triggers the controller at siteB to send msg1 to the controller at siteA. Shown in the bottom of (a) is the controller at siteA reacting to msg1 by sending msg2.

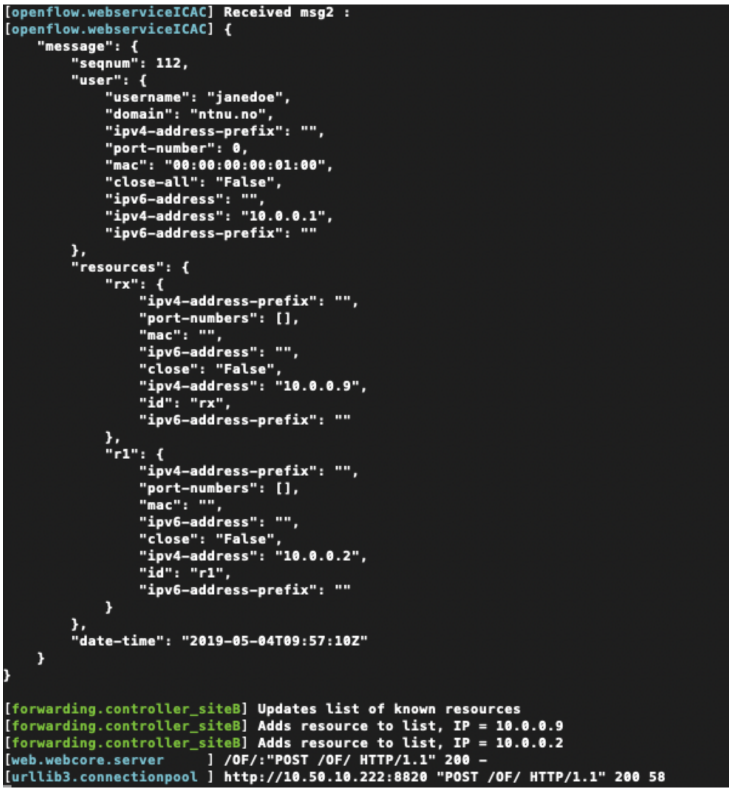
In order to start the interaction between the controllers, the `user` has to *connect to the network*. From the `pingall` shown in Figure 5.5a it is clear that the `user` already has connected to the network, but in order to mock the `user` communicating with an Authentication Server (AS) the controller checks whether the `user` tries contacting an IP address reserved for the "AS". Earlier versions had the `user` not being able to send or receive traffic before it contacted the AS, however the resulting `pingall` was misleading as `msg3` goes between the controllers while the `user` tries pinging r1 (the first node the user pings) and the configuration at c1 is so swift that the ping is successful.

When the `user` communicates with the AS the controller at siteB sends `msg1` to the controller at siteA, as shown in Figure 5.6. This triggers the controller at siteA to send `msg2` to the controller at siteB, as shown in Figure 5.7, and the controller at siteB then updates it's list of known resources the `user` has access to at siteA. After the `msg1` and `msg2` has been sent between the controller as a result of the user communication with the AS, the controller `c2` waits for the `user` to contact a



```
forwarding.controller_siteA] Sending msg2 to controller_siteB
urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8821
urllib3.connectionpool ] http://10.50.10.222:8821 "POST /OF/ HTTP/1.1" 200 58
web.webcore.server     ] /OF/:"POST /OF/ HTTP/1.1" 200 -
```

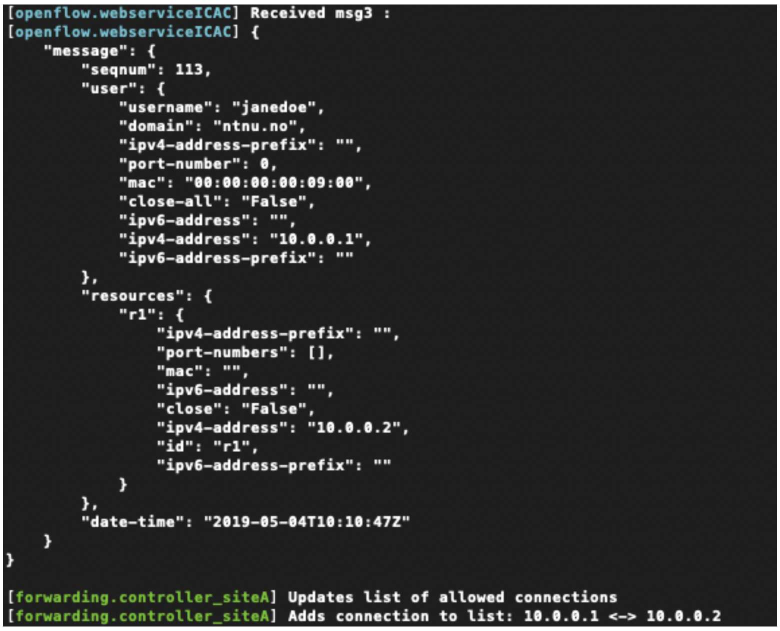
(a) Controller at SiteA



```
[openflow.webserviceICAC] Received msg2 :
[openflow.webserviceICAC] {
    "message": {
        "seqnum": 112,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:01:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "rx": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.9",
                "id": "rx",
                "ipv6-address-prefix": ""
            },
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T09:57:10Z"
    }
}

[forwarding.controller_siteB] Updates list of known resources
[forwarding.controller_siteB] Adds resource to list, IP = 10.0.0.9
[forwarding.controller_siteB] Adds resource to list, IP = 10.0.0.2
[web.webcore.server     ] /OF/:"POST /OF/ HTTP/1.1" 200 -
[urllib3.connectionpool ] http://10.50.10.222:8820 "POST /OF/ HTTP/1.1" 200 58
```

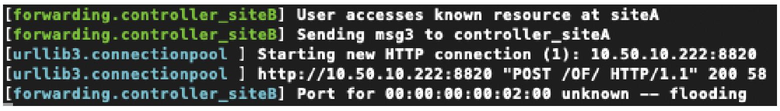
(b) Controller at SiteB

Figure 5.7: Step 2. The output of both controllers. The msg1 sent from siteB triggers the controller at siteA to send msg2, see Figure 5.6a, to the controller at siteB. The controller at siteB then updates the list of known resources, based on the contents of msg2.



```
[openflow.webserviceICAC] Received msg3 :
[openflow.webserviceICAC] {
    "message": {
        "seqnum": 113,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:09:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T10:10:47Z"
    }
}

[forwarding.controller_siteA] Updates list of allowed connections
[forwarding.controller_siteA] Adds connection to list: 10.0.0.1 <-> 10.0.0.2
```

(a) Controller at SiteA

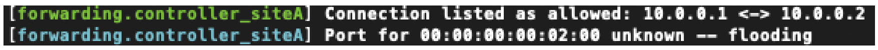


```
[forwarding.controller_siteB] User accesses known resource at siteA
[forwarding.controller_siteB] Sending msg3 to controller_siteA
[urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8820
[urllib3.connectionpool ] http://10.50.10.222:8820 "POST /OF/ HTTP/1.1" 200 58
[forwarding.controller_siteB] Port for 00:00:00:00:02:00 unknown -- flooding
```

(b) Controller at SiteB

Figure 5.8: Step 3. The output of both controllers. The controller at siteB recognized the user attempting to access a known resource, and sends msg3 to the controller at siteA. The controller at siteA then adds the connection between the user and the accessed resource to the list of allowed connections.

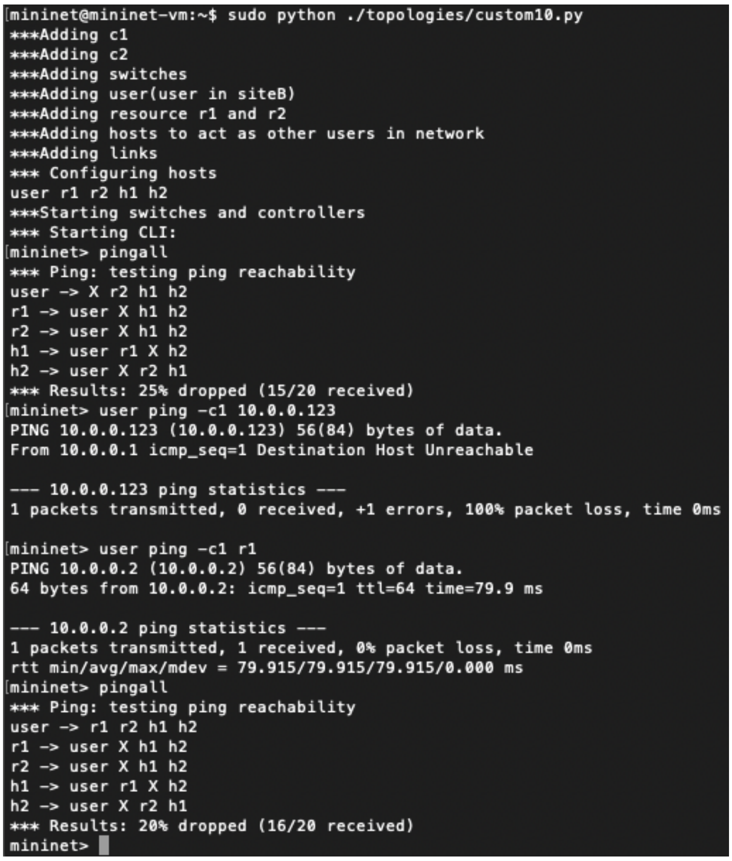
resource at network A. In this scenario there is only one resource at each controllers network, but we see in `msg2` in Figure C.6 that the message sent to `c2` can contain several resources and the controller `c2` checks whether the `user` tries accessing either of the resources from this message. When the `user` attempts accessing the resource, the `msg3` is sent to c1 and it configures access to the resource, as shown in Figure 5.8. Figure 5.5b shows the result of the `pingall` executed after `msg3` is sent. By looking at the upper left corner in the ping success table in the figures in Figure 5.5 it is clear to see that the `user` goes from not having access to `r1`, to having access after the messages are sent. This can also be seen in Figure 5.9, which shows the user is now allowed to launch a connection to the resource r1. See Figures C.8 and C.9 in Appendix C for screenshots of the message sequence as seen from the terminals running the controllers.



```
[forwarding.controller_siteA] Connection listed as allowed: 10.0.0.1 <-> 10.0.0.2
[forwarding.controller_siteA] Port for 00:00:00:00:02:00 unknown -- flooding
```

Figure 5.9: The output of controller at siteA when user tried connecting to resource r1 after the messages has been transmitted between the controllers. It is clear that the user `<->` r1 connection is among the list of allowed connections, and the packet from the user is then flooded.

Figure 5.10 shows the commands run on the terminal in Mininet in order to act out the emulation. First, the file containing custom topology, see Figure C.4, is executed and it creates a network with virtual hosts, switches, links and remote controllers. It also opens Mininet's CLI. Thereafter, a `pingall` is done as shown in Figure 5.5a. The `user` pings it's AS and not surprisingly the ping is unsuccessful. This is due to the IP address set aside as the AS is not belonging to a host in the emulating networks. When the `user` then tries pinging `r1`, the message `msg3` is sent and the access configuration done so fast that the ping is successful. The `pingall` done in the end is the same as shown in Figure 5.5b.



```
mininet@mininet-vm:~$ sudo python ./topologies/custom10.py
***Adding c1
***Adding c2
***Adding switches
***Adding user(user in siteB)
***Adding resource r1 and r2
***Adding hosts to act as other users in network
***Adding links
*** Configuring hosts
user r1 r2 h1 h2
***Starting switches and controllers
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
user -> X r2 h1 h2
r1 -> user X h1 h2
r2 -> user X h1 h2
h1 -> user r1 X h2
h2 -> user X r2 h1
*** Results: 25% dropped (15/20 received)
mininet> user ping -c1 10.0.0.123
PING 10.0.0.123 (10.0.0.123) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.123 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

mininet> user ping -c1 r1
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=79.9 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 79.915/79.915/79.915/0.000 ms
mininet> pingall
*** Ping: testing ping reachability
user -> r1 r2 h1 h2
r1 -> user X h1 h2
r2 -> user X h1 h2
h1 -> user r1 X h2
h2 -> user X r2 h1
*** Results: 20% dropped (16/20 received)
mininet> 
```

Figure 5.10: Shows the actions taken from the terminal in Mininet during the emulation.

Chapter 6 Discussion

This chapter discusses the results given in Chapter 5 and evaluates the model based on whether it answered the research question presented in Chapter 1 and if the solution is a good one. First, a recap of the results and what it means. Then, to the model and if it answers the research question. At last, the limitations of the model is commented on.

6.1 The results

Last chapter presented the results and research done to get the results, but what do they actually show? The final YANG data model shown in Figure 5.2 presents the values - or information - sent between the controllers, and the structure these are ordered in. These values should enable controllers to configure flow entries, or forwarding rules, such that the specific user gets access to only the specific resource. The structure helps the controllers know how to structure the information in the messages they send, and how to read and interpret the messages they receive. However, does the model actually enable the controllers to configure rules so that users get access to resources?

In order to validate this, an emulation was done in Mininet. As mentioned in the Methodology chapter, the result of the emulation can be considered as valid due to the emulation being close to a real-life system, and reacting in similar ways. However, the system is closed off from outside disturbances and effects which could possibly affect the outcome of the validation. The most likely part of the emulation to be affected would probably be the communication between the controllers, and not how the controllers themselves use the messages. While this is an important part of making the emulation possible, it is not the part of the system being validated. That is the model and the controllers being able to use the model to configure access for users. Due to this, the emulation can be considered a valid way of validating and testing the model.

```
module: intercontroller-access-control
  +--rw message
     +--rw seqnum        uint32
     +--rw date-time     yang:date-and-time
     +--rw user
     |  +--rw username                     string
     |  +--rw domain                       string
     |  +--rw (ip-address)
     |  |  +--:(ipv4-address)
     |  |  |  +--rw ipv4-address?          inet:ipv4-address
     |  |  +--:(ipv6-address)
     |  |  |  +--rw ipv6-address?          inet:ipv6-address
     |  |  +--:(ipv4-prefix)
     |  |  |  +--rw ipv4-address-prefix?   inet:ipv4-prefix
     |  |  +--:(ipv6-prefix)
     |  |     +--rw ipv6-address-prefix?   inet:ipv6-prefix
     |  +--rw mac?                         yang:mac-address
     |  +--rw port-number?                 inet:port-number
     |  +--rw close-all?                   boolean
     +--rw resources* [id]
        +--rw id                           string
        +--rw (ip-address)
        |  +--:(ipv4-address)
        |  |  +--rw ipv4-address?          inet:ipv4-address
        |  +--:(ipv6-address)
        |  |  +--rw ipv6-address?          inet:ipv6-address
        |  +--:(ipv4-prefix)
        |  |  +--rw ipv4-address-prefix?   inet:ipv4-prefix
        |  +--:(ipv6-prefix)
        |     +--rw ipv6-address-prefix?   inet:ipv6-prefix
        +--rw mac?                         yang:mac-address
        +--rw port-numbers*                inet:port-number
        +--rw close?                       boolean
```

Figure 6.1: The final data model of inter-controller access control. Written in YANG for structure, and containing the necessary values.

In the emulation the YANG model was used to structure the messages sent between the controllers, and also in order to interpret the messages received. This is clear in the code[1], when the users' and the resources' IP addresses are extracted from the messages. As an end result of the emulation, the pingalls (Figure 5.5) done in the command line shows that the user' access to the resource has changed. The user did not have access before the messages was sent, but it did after. The result of the pingalls also shows that the user still has access to resource `r2` even when it has access to resource `r1`.

Another point of discussion, could be whether the values chosen are indeed the values needed in the model. The model is made to be used in most foreseeable scenarios and manner, while only being validated for the scenario described earlier

[1]https://github.com/janitamjohansen/intercontroller-access-control

in this thesis (Chapter 4) due to this thesis' constraints. Therefore, there is valid doubt as for whether this model is valid for other scenarios. This is a case for further research, to test and validate the model in several scenarios and see whether and how it could be improved based on the results on those tests. On the other hand, as mentioned earlier in this thesis, the data model created is made for use in the most general cases. This so that commercial or specialized cases can extend the model as a standard to fit their own needs and features. This much like other data models made in YANG [47].

6.2 The model

The main research question was first mentioned in Section 1.3 Objectives in Chapter 1 Introduction.

> *Can a novel inter-domain access control scheme built upon Software Defined Networking improve how institutions manage user access to online content and online resources at partner institutions, compared to today's solution?*

In order to help answer this research question, two knowledge questions have been formulated and answered. The first knowledge question, *What information has to be transmitted?*, has been answered in Chapter 5 and resulted in a YANG data model showing values and the structure of these values needed to configure access for the user in most foreseeable situations. While the second knowledge question, *How can the information be transmitted?*, was answered in Chapter 2 in Section 2.3.5 by looking at relevant works. The result of the latter was that there existed several ways of transmitting information between controllers. From using the controller's REST APIs or the universal BGP, to the DISCO controller and the WE-bridge. Even though the solutions mentioned only transfer network and reachability information, it is assumed that they can easily send inter-controller access control messages.

First, lets take a look at the last part of the research question:

> *compared to today's solution?*

As seen in Section 2.1.2 in Chapter 2 Background and Related Works, today the solution for managing access control is generally by using firewalls. Firewalls are generally static and local on the forwarding hardware. Often in large networks, there is a firewall manager software that creates the rules and installs the rules onto the forwarding hardware. Without this type of software, the network administrator would have to change the access lists to fit the hardware manually and update each of the devices. Two of the solutions for a firewall manager software are, as mentioned in Section 2.1.2, Firewall Builder and Capirca. While there are similar solutions, these two stands out in that they can install access list rules for firewalls on most of the forwarding hardware used today while others only work in commodity hardware. While Firewall Builder and Capirca has many functions when making and managing firewalls, it does not support changing access list rules dynamically when users wish to access resources. Should the user get access while it is in another network, the network administrator would have to get the users current IP address, change the rules to allow the IP address, and reload the access list rules to the firewalls on the hardware. Then, when the user leave the network, the opposite would have to happen quite immediately so that the unwanted hosts are kept out. This is not a solution for this problem, as the administrative work is immense, due to the security risks should the rules remain active for longer than they should, or that it simply takes too long to install the new rules for the user to make use of them.

However, looking at it in another light it might not be impossible to use Capirca or Firewall Builder for an inter-domain access control should it support use of this thesis' data model. Perhaps following a proactive model, described in Section 5.1.1, in which all rules allowing the user access to resources are allowed in the start when the user connects. The firewall manager, whether Capirca or Firewall Builder, would have to be able to read the data model messages and translate them to rules allowing access in the firewall. This interface would have to be automated, should the solution be efficient. The manager would then have to download/rewrite the access lists without disturbing the current flow of traffic going through the firewall and network. This would then be undone when the user disconnects, which can be possible by using idle and timeout in 802.1x or looking at the ARP tables to see how long since the MAC has sent traffic. The main part of this is that the managers has to be able to understand and parse the messages, no matter the implementation. This adds on to the idea that the emulation in this thesis is only to validate the YANG data model, rather than find an efficient implementation.

Another limitation of both Capirca and Firewall Builder is the hardship in handling NAT solutions. While Capirca [26] looks unable to support NAT at all, Firewall Builder is able to provide access control based on groups if each group has it's own NAT. A solution that circumvents NAT entirely. NAT, as seen in Chapter 5 under Considerations, has been recognized as a scenario the data model can be used for, and the values tries to reflect that. However, this is not validated or tested, and errors or missing values may appear during testing. This is something to consider working further on in the future.

Another of today's solutions mentioned in the early chapters of this thesis is the use of VPN. VPN allows remote users access to resources at their home network, by making it appear as if the users traffic originates from within the network. While VPN is a great solution for remote users, it does have some drawbacks. This thesis wishes to improve the access control, and extend it further than what VPN makes possible. So that the user gets access to both resources at it's home network and the network it currently resides in simultaneously. This is where the VPN stops to be a solution. Another implementation of this thesis' solution could be using the inter-domain access control to simultaneously use resources at multiple institutions, for example in a research context. Often, different institutions have different and maybe unique resources, so easily getting access to all of them at the same time could bring lots of possibilities. However, while one of the mentioned drawbacks of using VPN was scaling, this is not tested in this thesis. The scenario emulated only contains one resource each at the domains, and only one user that regains access.

Next, lets focus on whether the YANG data model has answered the research question. The research question mentions improving on how institutions manage access control at partner institutions. While improvement can mean many things, in this master thesis the focus has been on adding to the access control. While before the access control between institutions was limited to using VPN, now there is a possibility for users to get access to resources at it's home network - the visiting network's partner institution - while connected to the visiting network. As a drawback of this, this assumes the partner institutions have implemented this thesis' solution and have security policies that allow users from other institutions to get access. In this way it is possible to conclude that the thesis answers the research question, and that it does indeed improve the access control. On the other hand, partner institutions may decide that they do not have use for this thesis' solution and VPN works well enough for them, and from this stance this thesis' YANG data model has not improved the access control.

Regarding the security policy mentioned in the last paragraph, the institutions may not allow opening up access from outside the network, despite it being a user with affiliation to the institution, due to security constraints. It may also go the other way, the institution may have a closed edge switch, and traffic to some resources are not allowed no matter if the resources belongs to a partner institution. These security issues that come from opening for the users access can be something to consider for future works, like how to make the access secure, and counteract hackers exploiting weaknesses in the communication between the controllers.

Furthermore, another point to consider is whether the solution is likely to be used. Is it worth implementing a large network, does it come with enough benefits? Mainly, this thesis' model would be useful for educational network with high cooperation. It might even be used between different networks and campuses, within the same university or organization. This begins to sound similar to physically distributed domains, but as we saw in Background and Related Works there are several ways of implementing east-west communication within those kinds of networks.

All in all, the model can be said to have improved the access control between partner institutions. It adds functionality previously not there, and has several instances where it is an improvement from access control today, some simply because it achieves access control inter-domain.

6.3 The limitations

From the beginning, this thesis has tried finding a solution for inter-controller access control. This, as a rather obvious consequence, means that the model has been made for communication between two fully SDN networks and assumes there is a controller with full view and control of it's network. While it does not mean that it cannot be used for hybrids, between SDN and legacy networks with some kind of support, or other combinations, it means that this thesis' model has not made any considerations to these scenarios or limitations that may appear between non SDN networks.

Another limitation regarding this solution is that the controllers need to implement logic to interpret, send, and generally handle messages containing values structured following the YANG data model. However, in a programmable network, this may not be as big of an effort as it would in legacy networks. In addition, it is assumed that some sort of east-west communication is implemented. It may be as easy as a web server listening for an HTTP POST message or more complicated like the ones listed in Section 2.3.5 as an answer to the knowledge question *How can the information be transmitted?* Depending on the east-west communication, it also needs to know how to reach the different controllers of the different domains, or at least know how to get that information.

An assumption made in this thesis and in the emulation of the model as well, is that the networks employ some sort of well-known identity-based access control scheme. An example of this is eduroam. Here the user logs on to a network with a username, domain and password. It is also assumed that the controllers are somehow able to get information from the messages sent between the user and the authentication server. Examples of this can be seen in the two articles [29, 2] mentioned in Chapter 2. Another assumption is that the controller can check the user's access - which resources and services is it allowed to use - by the user's username and eventually also it's domain. This in order to know which resources to eventually open for the user.

Something important to remember when evaluating the model, the emulation, and in general this thesis, is that it has a limited scope, as seen in Section 1.4

Scope. This thesis does not consider the security of the model, the connection, or the communication between the controllers. Neither the security policy a controller nor network may have. This is all considerations it is possible to continue in further works. In addition to security, the emulation neither take into consideration efficiency or fault-tolerance. Should the emulation be run with commands not in the scenario or with commands run in different orders, the code does not check for this. While some common scenarios are certainly thought of when making the code of the emulation, edge cases are not considered relevant to the validation of the model. Only the main scenario, scenario A from Chapter 4, is relevant. This also bring up a mentioned limitation of the model in that it is only tested for one scenario. While several scenarios and situations where considered and though through, there is no proof or guarantee that the model would work for several different situations.

Chapter 7

Conclusion

This thesis has looked at access control in SDN networks, and tried to improve it. More specifically, it has looked at access control between partner institutions. While solutions for managing and making access decisions exists inside legacy networks, there is no functionality for extending access control to different institutions. However, with the emergence of SDN technology, this might be made possible. As a way of facilitating this, this thesis has contributed a data model written in YANG, containing the structured values needed to give a user access to resources at partner institutions. Specifically, the user gets access to resources at the home university, while the user itself finds itself connected to the partner institution's network. While the access control could very well go the other way, this will have to be researched in future works.

A design science method was used to create the data model, and it was validated and tested using an emulation in Mininet. The emulation showed the user regaining access to the resource after messages had been sent between the controllers. These messages are structured after the YANG data model, and contains necessary data for the controllers to provide and configure access. Of course there are limitations and assumptions and these are described in Chapters 5 and 6.

This thesis has also looked into how the messages could be sent between the controllers. While none has sent access control information between controllers before, there has been attempts and solutions for sending network and reachability information between controllers. These east-west communication, or inter-controller solutions vary from using already existing technology like using REST APIs and BGP, to entirely new solutions like the WE-bridge or DISCO controller. Some of these solutions could be the way to send the inter-controller access control messages between the controllers.

7.1 Future work

As mentioned before throughout this thesis, security is not considered. Future works within security could focus on several points. Among these is the security of the communication channel between the controllers. Considerations must be taken for integrity, authentication and confidentiality. Integrity, because if not then attackers could change the message to present an IP address not belonging to the user - or possible the resource as well. An attacker could exploit this to gain access to resources within a network. Authentication, because the controller has to be sure the message comes from the controller it says it's from. If not an attacker could easily spoof a message to gain access. Certificates could be used to manage authentication between controllers. Confidentiality, due to the potential sensitive information send in the messages. Especially since GDPR came into effect in 2018, has this been important. Attackers should not be able to get it's hand on sensitive information, and host information, username and domain can be considered this.

In addition to the communication channel between the controllers, the user may also have to authenticate itself, so the controller can be sure the user is who it says it is with the information it gives. Also, the database or other storage containing all the users' access could contain sensitive information, and has to be protected. Mentioned in Section 5.1.3 was caching data in order to know when the user attempts to access a resource. This will have to be stored safely, should it be allowed. Whether information should be allowed stored or not is part of the security policy of the controllers. This security policy is also something to consider in future works, as it may provide limitations for using the data model.

While this thesis provides a validation and testing of the data model in a scenario where the user regains access to it's resources located at the home network, while it itself is connected to a partner institutions network, the opposite scenario has not been tested. While the model contains the necessary values to configure access, there would in addition be some kind of agreement between the controllers that users are allowed to access specific resources. This agreement has not been mentioned in this thesis, and so must be worked further on.

Additionally, as mentioned earlier in the thesis, using the data model with a NAT scenario has not been tested. While the model has considered NAT in it's making, further research in the shape of testing and validation is needed before one can confidently say this data model works with NAT solutions. Moreover, further testing in general of different scenarios and situations can be considered for further work as only one scenario is validated and tested in this thesis. As a part of this further testing, scaling in particular should be tested as it is an important part of the possibilities the inter-domain access control brings.

Finally, the scope of this thesis and a limitation in making the emulation, is that much is simulated and assumed instead of actually implemented. An identity-based access control within the SDN networks is assumed, but not implemented. Other limitations and assumptions is mentioned both when describing the emulation and in the last chapter. Further works would implement all parts of the solution in a real-life environment in order to be decidedly sure that the data model works as it should.

References

[1] SDX central, "Understanding the SDN Architecture - SDN Control Plane & SDN Data Plane." https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/. Accessed: 2019-04-22.

[2] F. Hauser, M. Schmidt, and M. Menth, "Establishing a session database for SDN using 802.1 X and multiple authentication resources," in *IEEE ICC 2017 SAC Symposium SDN & NFV Track*, pp. 1–7, 2017.

[3] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–4, IEEE, 2014.

[4] R. J. Wieringa, "Design science methodology for information systems and software engineering," 2014.

[5] M. Bjorklund and L. Berger, "YANG Tree Diagrams," BCP 215, RFC Editor, March 2018.

[6] Richard Burts, "Difference between ACL and firewall." https://community.cisco.com/t5/switching/difference-between-acl-and-firewall/td-p/449204, 2005. Accessed: 2019-05-24.

[7] IG, "What Is The Difference Between ACL And Firewall? Definations & Examples." https://www.gigxp.com/difference-between-acl-and-firewall/. Accessed: 2019-05-24.

[8] J. O'Hara, "Toward a commodity enterprise middleware," *Queue*, vol. 5, no. 4, pp. 48–55, 2007.

[9] D. C. Plummer, "Ethernet address resolution protocol: Or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware," STD 37, RFC Editor, November 1982. http://www.rfc-editor.org/rfc/rfc826.txt.

[10] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," RFC 4271, RFC Editor, January 2006. http://www.rfc-editor.org/rfc/rfc4271.txt.

[11] ipwithease, “Difference between ACL on Router and Firewall.” https://ipwithease.com/difference-between-acl-on-router-and-firewall/, 2018. Accessed: 2019-05-24.

[12] Gentoo Foundation, “Hostapd.” https://wiki.gentoo.org/wiki/Hostapd, January 2019. Accessed: 2019-05-15.

[13] National Institute of Standards and Technology, “Glossary -> identity-based access control.” https://csrc.nist.gov/glossary/term/identity_based-access-control. Accessed: 2019-05-13.

[14] “Introducing JSON.” https://www.json.org/. Accessed: 2019-05-24.

[15] D. Crockford, “The application/json media type for javascript object notation (json),” RFC 4627, RFC Editor, July 2006. http://www.rfc-editor.org/rfc/rfc4627.txt.

[16] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network configuration protocol (netconf),” RFC 6241, RFC Editor, June 2011. http://www.rfc-editor.org/rfc/rfc6241.txt.

[17] Codecademy, “What is REST?.” https://www.codecademy.com/articles/what-is-rest. Accessed: 2019-05-13.

[18] A. Bierman, M. Bjorklund, and K. Watsen, “Restconf protocol,” RFC 8040, RFC Editor, January 2017.

[19] Internet Engineering Task Force, “Internet Standards -> RFCs.” https://www.ietf.org/standards/rfcs/. Accessed: 2019-05-13.

[20] M. Bjorklund, “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” RFC 6020, RFC Editor, October 2010. http://www.rfc-editor.org/rfc/rfc6020.txt.

[21] T. Berger, “Analysis of current VPN technologies,” in *First International Conference on Availability, Reliability and Security (ARES’06)*, pp. 8–pp, IEEE, 2006.

[22] Jay H Simmons, “5 Disadvantages of VPN That You Should Know Before Using It.” https://www.vpncrew.com/5-disadvantages-of-vpn-that-you-should-know-before-using-it/. Accessed: 2019-04-19.

[23] eduroam, “Where can I eduroam?.” https://www.eduroam.org/where/. Accessed: 2019-04-19.

[24] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[25] A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using openflow: A survey,” *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[26] GoogleCodeExporter, “General NAT support as an action in the framework.” https://github.com/google/capirca/issues/31, August 2016. Accessed: 2019-05-08.

[27] K. Egevang and P. Francis, “RFC 1631 The IP Network Address Translator (NAT),” 1994.

[28] P. Srisuresh and K. Egevang, “Traditional IP Network Address Translator (Traditional NAT).” RFC 3022 (Informational), January 2001.

[29] V. Dangovas and F. Kuliesius, “SDN-driven authentication and access control system,” in *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC2014)*, pp. 20–23, The Society of Digital Information and Wireless Communication, 2014.

[30] F. Kuliesius and V. Dangovas, “SDN enhanced campus network authentication and access control system,” in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*, pp. 894–899, IEEE, 2016.

[31] A. S. Muqaddas, A. Bianco, P. Giaccone, and G. Maier, “Inter-controller traffic in ONOS clusters for SDN networks,” in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2016.

[32] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, “Towards an elastic distributed SDN controller,” *ACM SIGCOMM computer communication review*, vol. 43, no. 4, pp. 7–12, 2013.

[33] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. I. NEC, H. I. NEC, T. H. NEC, and S. Shenker, “Onix: a distributed control platform for large-scale production networks,” in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, (Berkeley, CA, USA), pp. 351–364, 2010.

[34] P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu, and Y. Lin, “A west-east bridge based SDN inter-domain testbed,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 190–197, 2015.

[35] H. Yin, H. Xie, T. Tsou, D. R. Lopez, P. A. Aranda, and R. Sidi, “SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains,” Internet-Draft draft-yin-sdn-sdni-00, Internet Engineering Task Force, June 2012. Work in Progress.

[36] H. Xie, T. Tsou, D. R. Lopez, and H. Yin, “Use Cases for ALTO with Software Defined Networks,” Internet-Draft draft-xie-alto-sdn-use-cases-01, Internet Engineering Task Force, June 2012. Work in Progress.

[37] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, “Forwarding and Control Element Separation (ForCES) Protocol Specification,” RFC 5810, RFC Editor, March 2010. http://www.rfc-editor.org/rfc/rfc5810.txt.

[38] M. Gerola, F. Lucrezia, M. Santuari, E. Salvadori, P. L. Ventre, S. Salsano, and M. Campanella, “ICONA: A Peer-to-Peer Approach for Software Defined Wide Area Networks Using ONOS,” in *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*, pp. 37–42, Oct 2016.

[39] C. Stevan, G. Erwan, and P. Yohan, “Inter-controller communication in switching SDN architecture.” https://wiki.minet.net/_media/wiki/sdn/inter-controller_communication_in_switching_sdn_architecture.pdf. Accessed: 2018-11-15.

[40] OpenDaylight.org, “ODL-SDNiApp:Developer Guide for Beryllium Release.” https://wiki.opendaylight.org/view/ODL-SDNiApp:Developer_Guide_for_Beryllium_Release. Accessed: 2019-05-09.

[41] R. Bennesby, P. Fonseca, E. Mota, and A. Passito, “An inter-as routing component for software-defined networks,” in *2012 IEEE Network Operations and Management Symposium*, pp. 138–145, IEEE, 2012.

[42] F. X. Wibowo and M. A. Gregory, “Software Defined Networking properties in multi-domain networks,” in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 95–100, IEEE, 2016.

[43] P. Cronin, F. Ryan, and M. Coughlan, “Undertaking a literature review: A step-by-step approach,” *British journal of nursing (Mark Allen Publishing)*, vol. 17, pp. 38–43, 01 2008.

[44] G. Chen, “Bibtex Entries for IETG RFCs and Internet-Drafts.” http://notesofaprogrammer.blogspot.com/2014/11/bibtex-entries-for-ietf-rfcs-and.html, November 2014. Accessed: 2019-05-13.

[45] D. E. McNabb, “Research methods for political science: Quantitative and qualitative methods,” 2008. ME Sharpe, 2nd edition.

[46] Ben Willis, “The advantages and limitations of single case study analysis.” https://www.e-ir.info/2014/07/05/the-advantages-and-limitations-of-single-case-study-analysis/, January 2013. Accessed: 2019-05-29.

[47] M. Jethanandani, S. Agarwal, L. Huang, and D. Blair, “Network Access Control List (ACL) YANG Data Model,” Internet-Draft draft-ietf-netmod-acl-model-21, Internet Engineering Task Force, Nov. 2018. Work in Progress.

[48] J. Schoenwaelder, “Common YANG Data Types,” RFC 6991, RFC Editor, July 2013.

[49] B. Linkleter, “Set up the Mininet network simulator.” http://www.brianlinkletter.com/set-up-mininet/, September 2013. Accessed: 2019-05-08.

Appendix A
Legend for Figures

Shows the legend for the Figures displaying topology in this thesis.

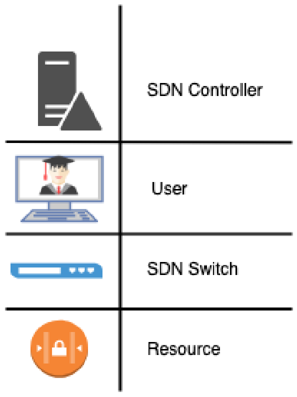


Figure A.1: Legend for topology figures used in throughout the thesis.

Appendix B
Sequence Diagrams

Shows the four sequence diagrams mentioned in Chapter 5 Research & Results. The four diagrams correlates to the four models presented in Table 5.1 regarding proactive, reactive or hybrid models.

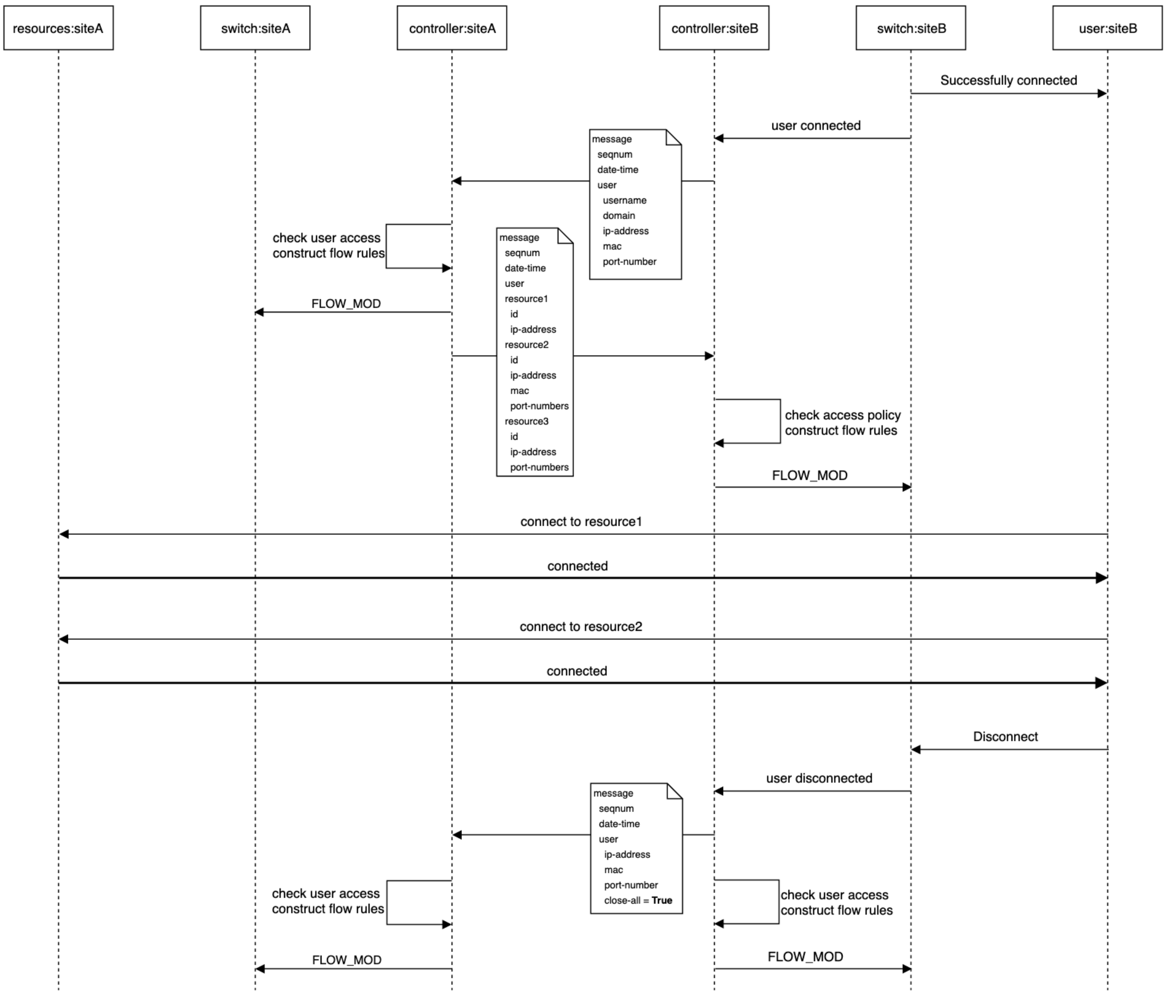


Figure B.1: Sequence Diagram Model 1 - The Proactive Model. Proposal of a way to use the YANG model in a proactive model (Table 5.1).

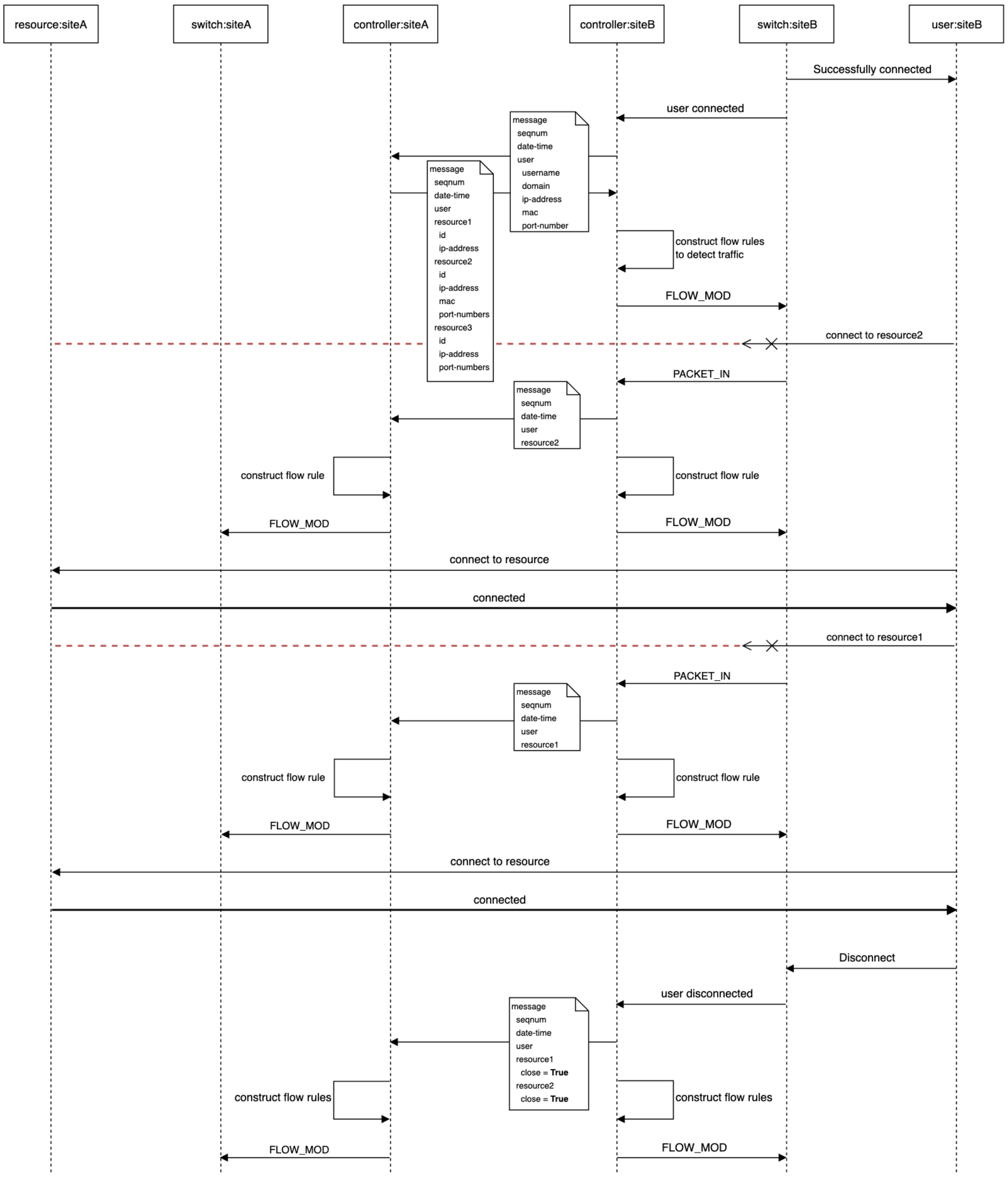


Figure B.2: Sequence Diagram Model 2 - The Hybrid Model. Proposal of a way to use the YANG model in a hybrid model (Table 5.1).

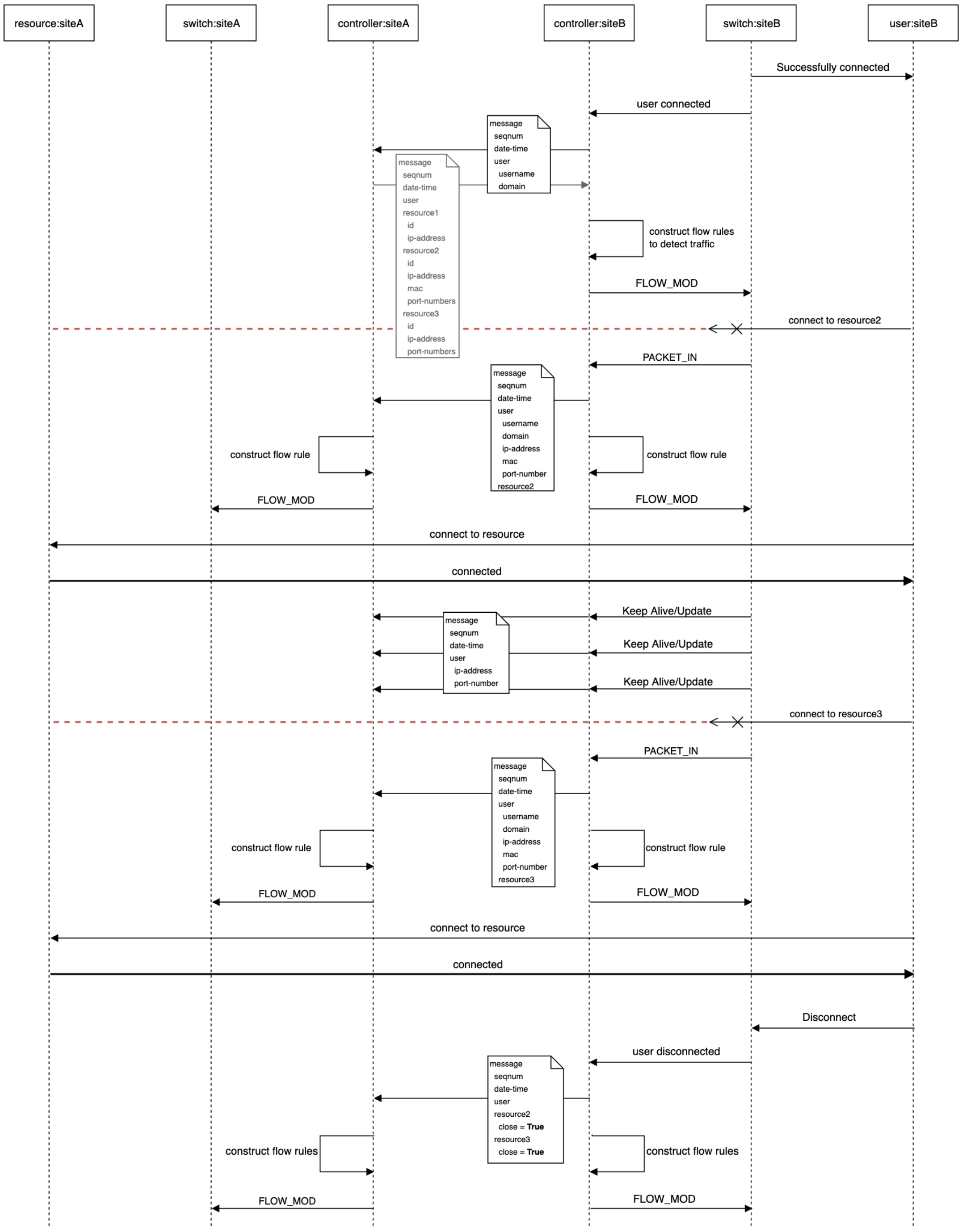

resource:siteA
switch:siteA
controller:siteA
controller:siteB
switch:siteB
user:siteB
Successfully connected
user connected
message seqnum date-time user username domain
message seqnum date-time user resource1 id ip-address resource2 id ip-address mac port-numbers resource3 id ip-address port-numbers
construct flow rules to detect traffic
FLOW_MOD
connect to resource2
PACKET_IN
message seqnum date-time user username domain ip-address mac port-number resource2
construct flow rule
construct flow rule
FLOW_MOD
FLOW_MOD
connect to resource
connected
Keep Alive/Update
Keep Alive/Update
Keep Alive/Update
message seqnum date-time user ip-address port-number
connect to resource3
PACKET_IN
message seqnum date-time user username domain ip-address mac port-number resource3
construct flow rule
construct flow rule
FLOW_MOD
FLOW_MOD
connect to resource
connected
Disconnect
user disconnected
message seqnum date-time user resource2 close = True resource3 close = True
construct flow rules
construct flow rules
FLOW_MOD
FLOW_MOD

Figure B.3: Sequence Diagram Model 3 - The Hybrid NAT Model. Proposal of a way to use the YANG model in a hybrid NAT model (Table 5.1).

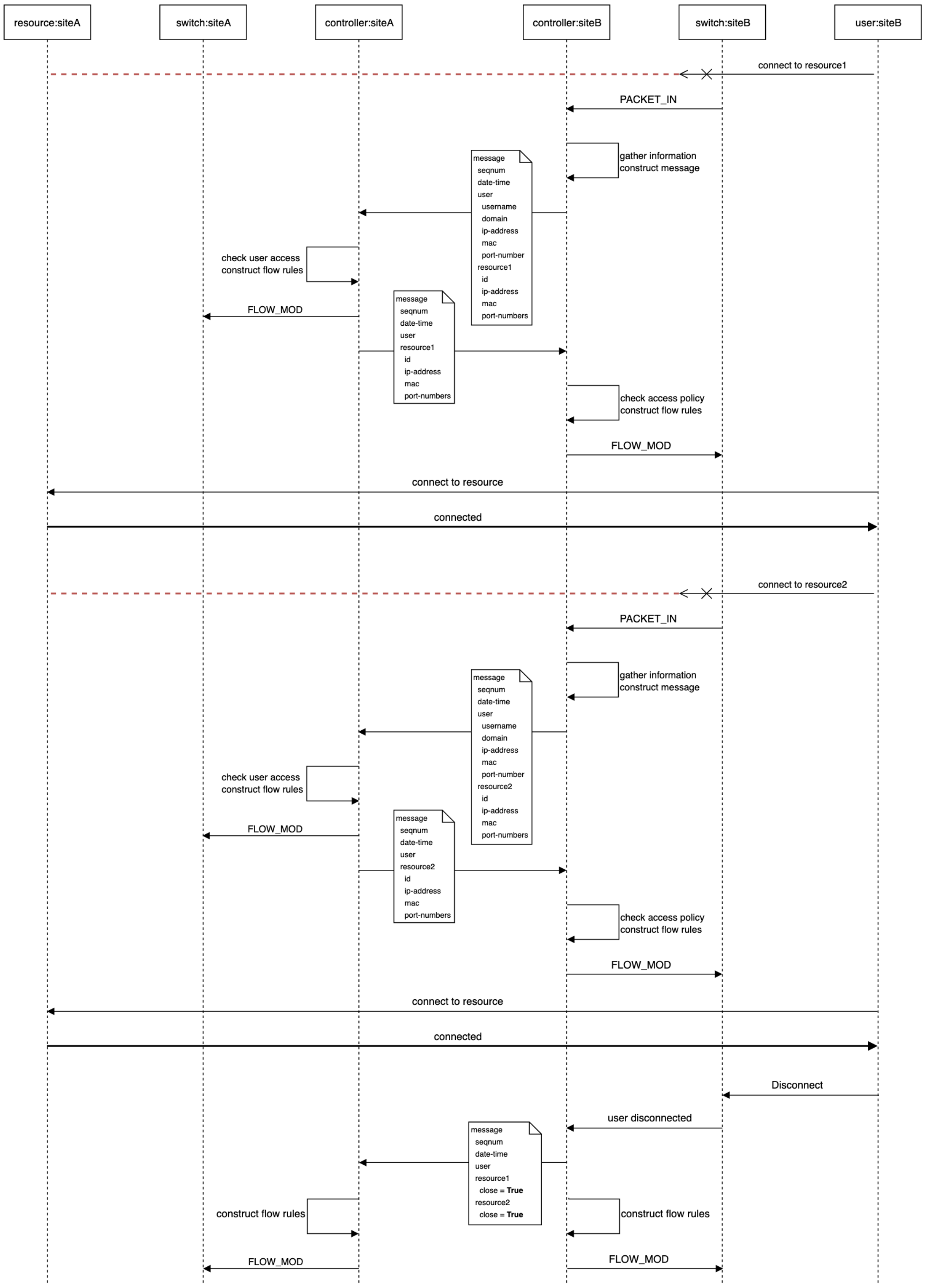
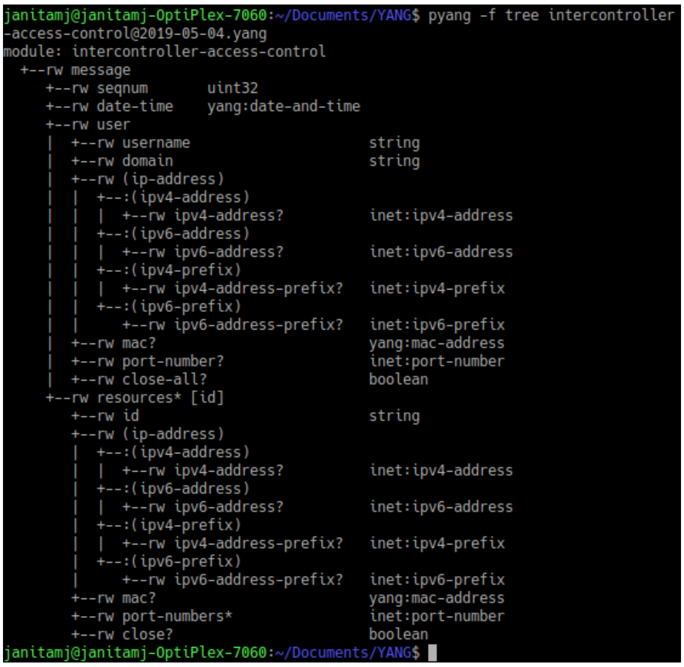


Figure B.4: Sequence Diagram Model 4 - The Reactive Model. Proposal of a way to use the YANG model in a reactive model (Table 5.1).

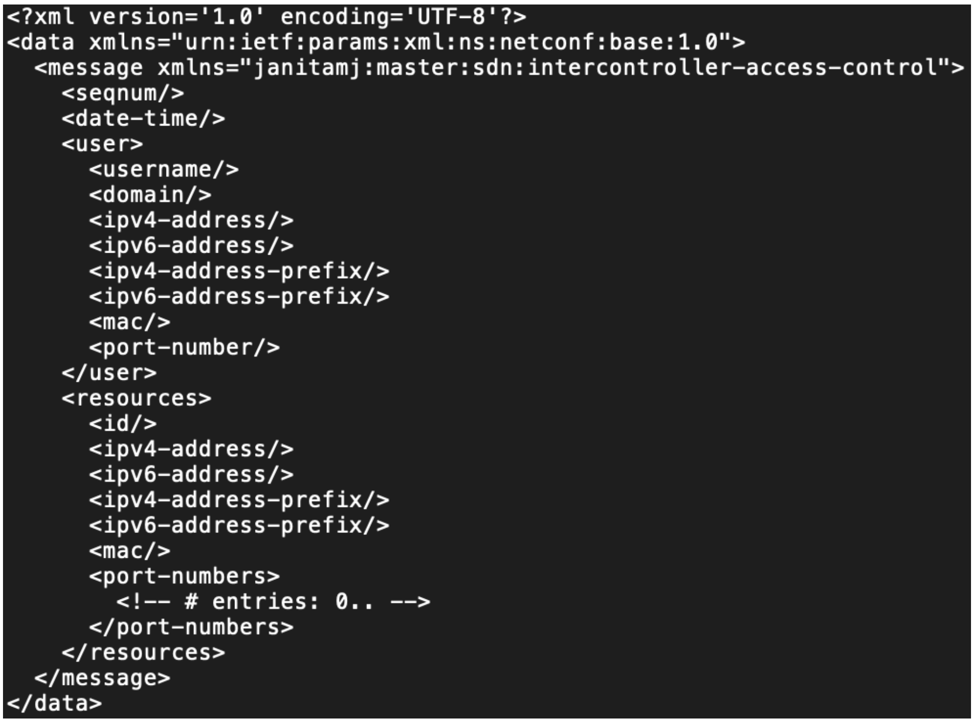
Appendix C Research & Results

Shows the screenshots and diagrams made when making the YANG model. In addition, the messages for the simulation are at the end of the section.



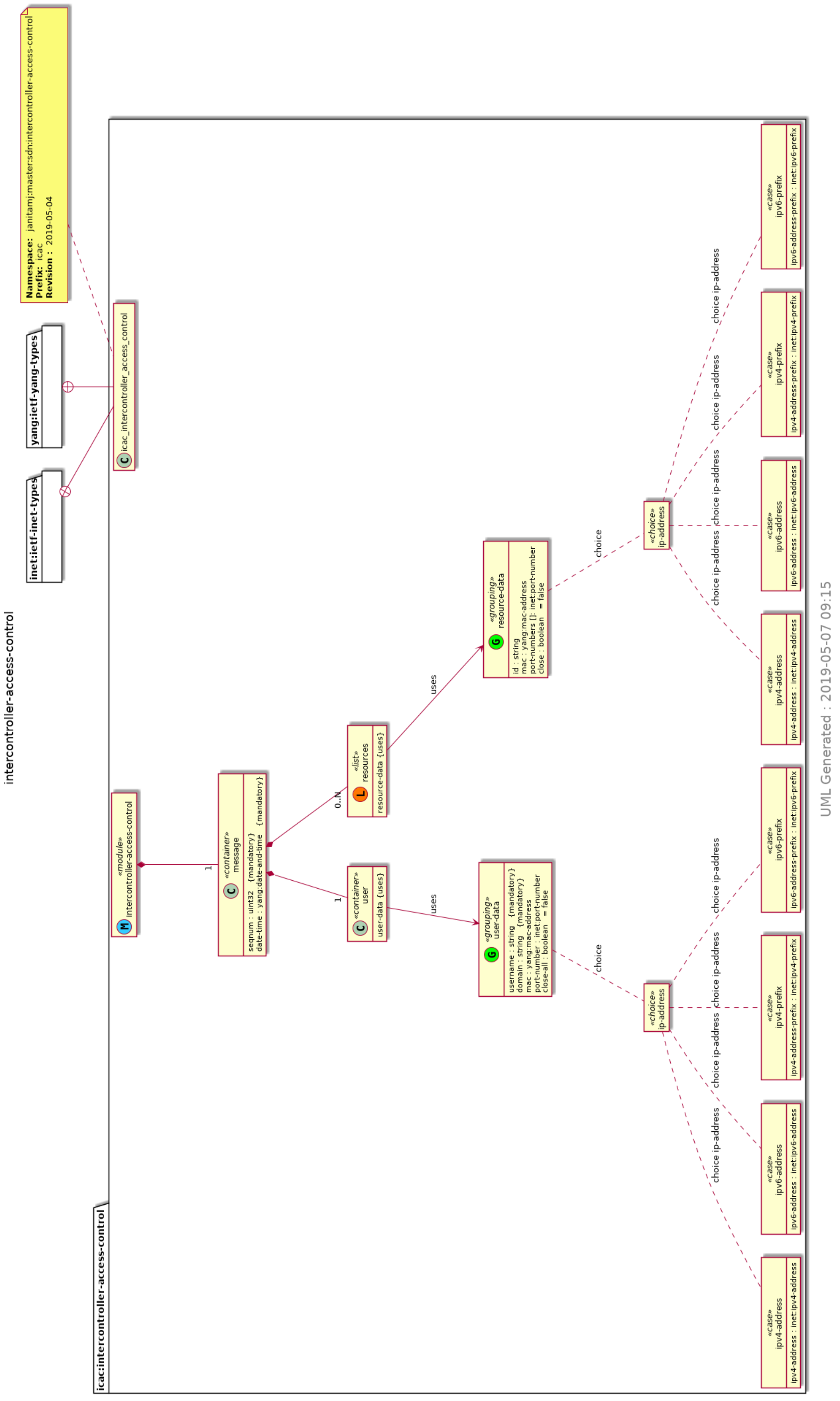
```
janitamj@janitamj-OptiPlex-7060:~/Documents/YANG$ pyang -f tree intercontroller
-access-control@2019-05-04.yang
module: intercontroller-access-control
  +--rw message
     +--rw seqnum       uint32
     +--rw date-time    yang:date-and-time
     +--rw user
     |  +--rw username                      string
     |  +--rw domain                        string
     |  +--rw (ip-address)
     |  |  +--:(ipv4-address)
     |  |  |  +--rw ipv4-address?           inet:ipv4-address
     |  |  +--:(ipv6-address)
     |  |  |  +--rw ipv6-address?           inet:ipv6-address
     |  |  +--:(ipv4-prefix)
     |  |  |  +--rw ipv4-address-prefix?    inet:ipv4-prefix
     |  |  +--:(ipv6-prefix)
     |  |     +--rw ipv6-address-prefix?    inet:ipv6-prefix
     |  +--rw mac?                          yang:mac-address
     |  +--rw port-number?                  inet:port-number
     |  +--rw close-all?                    boolean
     +--rw resources* [id]
        +--rw id                            string
        +--rw (ip-address)
        |  +--:(ipv4-address)
        |  |  +--rw ipv4-address?           inet:ipv4-address
        |  +--:(ipv6-address)
        |  |  +--rw ipv6-address?           inet:ipv6-address
        |  +--:(ipv4-prefix)
        |  |  +--rw ipv4-address-prefix?    inet:ipv4-prefix
        |  +--:(ipv6-prefix)
        |     +--rw ipv6-address-prefix?    inet:ipv6-prefix
        +--rw mac?                          yang:mac-address
        +--rw port-numbers*                 inet:port-number
        +--rw close?                        boolean
janitamj@janitamj-OptiPlex-7060:~/Documents/YANG$
```

Figure C.1: Command to make tree presentation of the final YANG model. Made with the Pyang tool.



```xml
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <message xmlns="janitamj:master:sdn:intercontroller-access-control">
    <seqnum/>
    <date-time/>
    <user>
      <username/>
      <domain/>
      <ipv4-address/>
      <ipv6-address/>
      <ipv4-address-prefix/>
      <ipv6-address-prefix/>
      <mac/>
      <port-number/>
    </user>
    <resources>
      <id/>
      <ipv4-address/>
      <ipv6-address/>
      <ipv4-address-prefix/>
      <ipv6-address-prefix/>
      <mac/>
      <port-numbers>
        <!-- # entries: 0.. -->
      </port-numbers>
    </resources>
  </message>
</data>
```

Figure C.2: XML skeleton of YANG model. Made with the Pyang tool.



intercontroller-access-control
inet:ietf-inet-types
yang:ietf-yang-types
Namespace: janitamj:master:sdn:intercontroller-access-control
Prefix: icac
Revision : 2019-05-04
icac_intercontroller_access_control
icac:intercontroller-access-control
«module» intercontroller-access-control
1
«container» message
seqnum : uint32 {mandatory}
date-time : yang:date-and-time {mandatory}
1
«container» user
user-data {uses}
0..N
«list» resources
resource-data {uses}
uses
«grouping» user-data
username : string {mandatory}
domain : string {mandatory}
mac : yang:mac-address
port-number : inet:port-number
close-all : boolean = false
«grouping» resource-data
id : string
mac : yang:mac-address
port-numbers []: inet:port-number
close : boolean = false
choice
«choice» ip-address
choice ip-address
«case» ipv4-address
ipv4-address : inet:ipv4-address
«case» ipv6-address
ipv6-address : inet:ipv6-address
«case» ipv4-prefix
ipv4-address-prefix : inet:ipv4-prefix
«case» ipv6-prefix
ipv6-address-prefix : inet:ipv6-prefix
UML Generated : 2019-05-07 09:15

Figure C.3: UML diagram of the YANG mode. Produced by the Pyang tool.

```
#!usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info


def MyNetwork():
    net= Mininet(topo=None, build=False)

    info("***Adding c1 \n")
    c1 = net.addController(name='c1',controller=RemoteController,ip='
        10.50.10.222',port=6633)

    info("***Adding c2 \n")
    c2 = net.addController(name='c2',controller=RemoteController,ip='
        10.50.10.222',port=6634)


    info("***Adding switches \n")
    s1=net.addSwitch('s1')
    s2=net.addSwitch('s2')


    info("***Adding user(user in siteB) \n")
    user=net.addHost('user', mac='00:00:00:00:01:00', ip='10.0.0.1')

    info("***Adding resource r1 and r2 \n")
    r1=net.addHost('r1', mac='00:00:00:00:02:00', ip='10.0.0.2')
    r2=net.addHost('r2', mac='00:00:00:00:03:00', ip='10.0.0.3')

    info("***Adding hosts to act as other users in network \n")
    h1=net.addHost('h1', mac='00:00:00:00:04:00', ip='10.0.0.4')
    h2=net.addHost('h2', mac='00:00:00:00:05:00', ip='10.0.0.5')

    info("***Adding links \n")
    net.addLink(s1,s2)
    net.addLink(r1,s1)
    net.addLink(h1,s1)
    net.addLink(user,s2)
    net.addLink(h2,s2)
    net.addLink(r2,s2)

    net.build()

    info("***Starting switches and controllers \n")
    net.get('s1').start([c1])
    net.get('s2').start([c2])

    c1.start()
    c2.start()

    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    MyNetwork()
```

Figure C.4: `custom.py` file, creates a custom topology in Mininet.

```
{
    "message": {
        "seqnum": 111,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:01:00",
            "close-all": False,
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {},
        "date-time": "2019-05-04T09:57:06Z"
    }
}
```

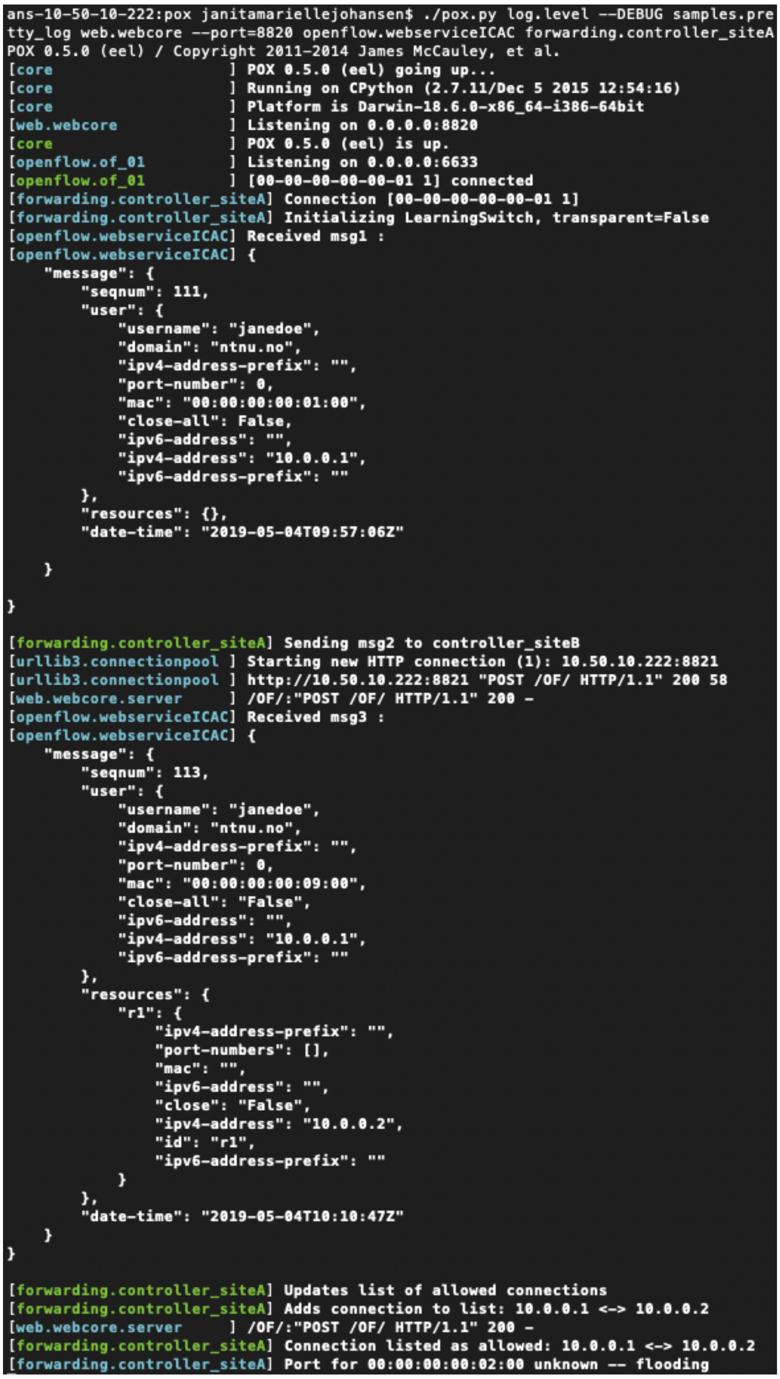
Figure C.5: Message 1, msg1, sent from controller c1 to controller c2 in simulation. Contains information about the user connected to c2's network.

```
{
    "message": {
        "seqnum": 112,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:01:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "rx": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.9",
                "id": "rx",
                "ipv6-address-prefix": ""
            },
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T09:57:10Z"
    }
}
```

Figure C.6: Message 2, msg2, sent from controller c2 to controller c1 in simulation. Contains information on all resources the user has access to in c1's network.

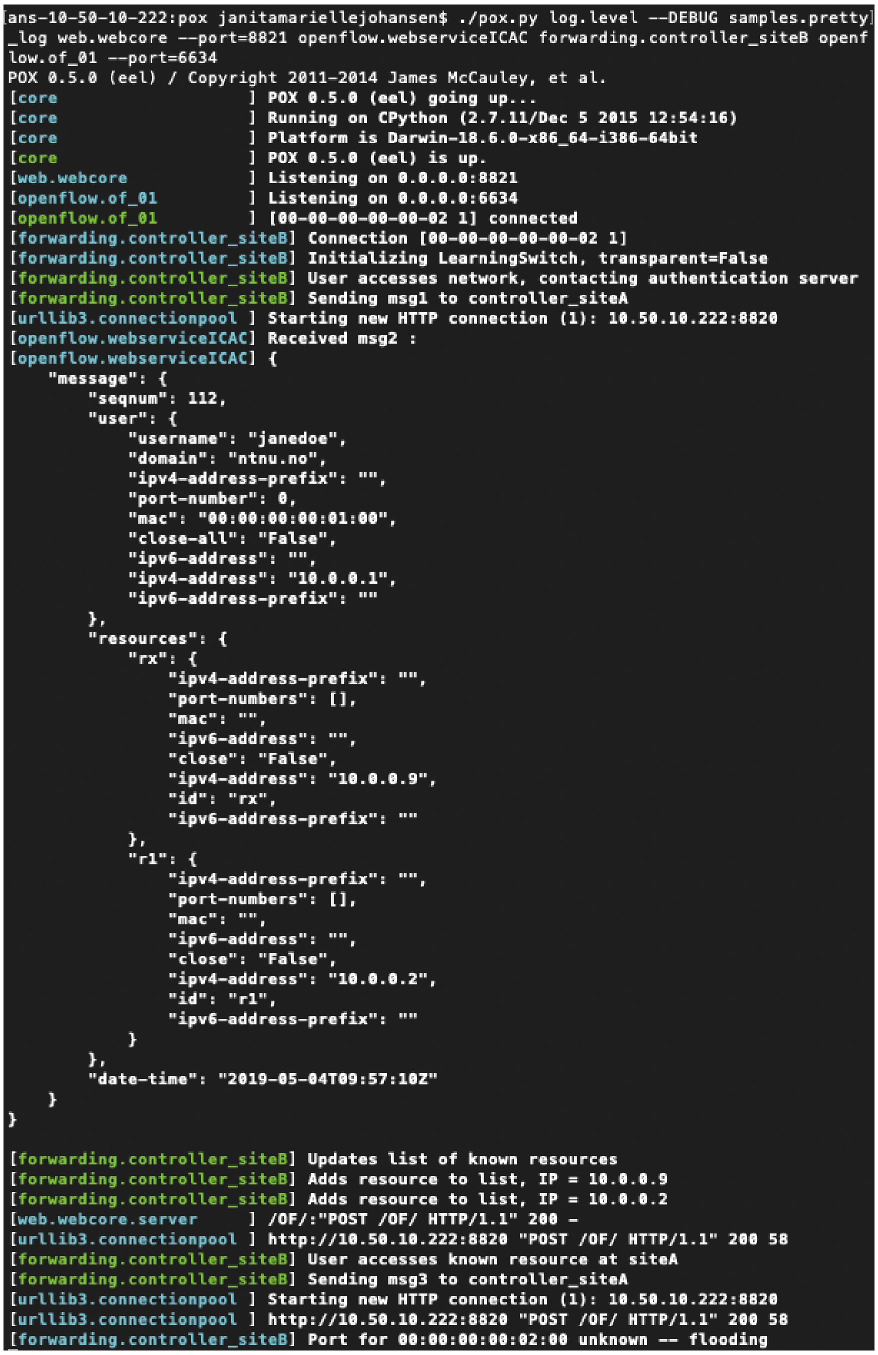
```
{
    "message": {
        "seqnum": 113,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:09:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T10:10:47Z"
    }
}
```

Figure C.7: Message 3, msg3, sent from controller c1 to controller c2 in simulation. Contains information on the resource the user attempts to access.



```
ans-10-50-10-222:pox janitamariellejohansen$ ./pox.py log.level --DEBUG samples.pre
tty_log web.webcore --port=8820 openflow.webserviceICAC forwarding.controller_siteA
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
[core                      ] POX 0.5.0 (eel) going up...
[core                      ] Running on CPython (2.7.11/Dec 5 2015 12:54:16)
[core                      ] Platform is Darwin-18.6.0-x86_64-i386-64bit
[web.webcore               ] Listening on 0.0.0.0:8820
[core                      ] POX 0.5.0 (eel) is up.
[openflow.of_01            ] Listening on 0.0.0.0:6633
[openflow.of_01            ] [00-00-00-00-00-01 1] connected
[forwarding.controller_siteA] Connection [00-00-00-00-00-01 1]
[forwarding.controller_siteA] Initializing LearningSwitch, transparent=False
[openflow.webserviceICAC] Received msg1 :
[openflow.webserviceICAC] {
    "message": {
        "seqnum": 111,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:01:00",
            "close-all": False,
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {},
        "date-time": "2019-05-04T09:57:06Z"

    }

}

[forwarding.controller_siteA] Sending msg2 to controller_siteB
[urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8821
[urllib3.connectionpool ] http://10.50.10.222:8821 "POST /OF/ HTTP/1.1" 200 58
[web.webcore.server     ] /OF/:"POST /OF/ HTTP/1.1" 200 -
[openflow.webserviceICAC] Received msg3 :
[openflow.webserviceICAC] {
    "message": {
        "seqnum": 113,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:09:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T10:10:47Z"
    }
}

[forwarding.controller_siteA] Updates list of allowed connections
[forwarding.controller_siteA] Adds connection to list: 10.0.0.1 <-> 10.0.0.2
[web.webcore.server     ] /OF/:"POST /OF/ HTTP/1.1" 200 -
[forwarding.controller_siteA] Connection listed as allowed: 10.0.0.1 <-> 10.0.0.2
[forwarding.controller_siteA] Port for 00:00:00:00:02:00 unknown -- flooding
```

Figure C.8: Screenshot of the controller at siteA's terminal during the message sequence between the controllers at siteA and siteB, where msg1, msg2, and msg3 is sent. At the end we see that the user gets access and the packet is flooded.



```
ans-10-50-10-222:pox janitamariellejohansen$ ./pox.py log.level --DEBUG samples.pretty
_log web.webcore --port=8821 openflow.webserviceICAC forwarding.controller_siteB openf
low.of_01 --port=6634
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
[core                      ] POX 0.5.0 (eel) going up...
[core                      ] Running on CPython (2.7.11/Dec 5 2015 12:54:16)
[core                      ] Platform is Darwin-18.6.0-x86_64-i386-64bit
[core                      ] POX 0.5.0 (eel) is up.
[web.webcore               ] Listening on 0.0.0.0:8821
[openflow.of_01            ] Listening on 0.0.0.0:6634
[openflow.of_01            ] [00-00-00-00-00-02 1] connected
[forwarding.controller_siteB] Connection [00-00-00-00-00-02 1]
[forwarding.controller_siteB] Initializing LearningSwitch, transparent=False
[forwarding.controller_siteB] User accesses network, contacting authentication server
[forwarding.controller_siteB] Sending msg1 to controller_siteA
[urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8820
[openflow.webserviceICAC] Received msg2 :
[openflow.webserviceICAC] {
    "message": {
        "seqnum": 112,
        "user": {
            "username": "janedoe",
            "domain": "ntnu.no",
            "ipv4-address-prefix": "",
            "port-number": 0,
            "mac": "00:00:00:00:01:00",
            "close-all": "False",
            "ipv6-address": "",
            "ipv4-address": "10.0.0.1",
            "ipv6-address-prefix": ""
        },
        "resources": {
            "rx": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.9",
                "id": "rx",
                "ipv6-address-prefix": ""
            },
            "r1": {
                "ipv4-address-prefix": "",
                "port-numbers": [],
                "mac": "",
                "ipv6-address": "",
                "close": "False",
                "ipv4-address": "10.0.0.2",
                "id": "r1",
                "ipv6-address-prefix": ""
            }
        },
        "date-time": "2019-05-04T09:57:10Z"
    }
}

[forwarding.controller_siteB] Updates list of known resources
[forwarding.controller_siteB] Adds resource to list, IP = 10.0.0.9
[forwarding.controller_siteB] Adds resource to list, IP = 10.0.0.2
[web.webcore.server      ] /OF/:"POST /OF/ HTTP/1.1" 200 -
[urllib3.connectionpool ] http://10.50.10.222:8820 "POST /OF/ HTTP/1.1" 200 58
[forwarding.controller_siteB] User accesses known resource at siteA
[forwarding.controller_siteB] Sending msg3 to controller_siteA
[urllib3.connectionpool ] Starting new HTTP connection (1): 10.50.10.222:8820
[urllib3.connectionpool ] http://10.50.10.222:8820 "POST /OF/ HTTP/1.1" 200 58
[forwarding.controller_siteB] Port for 00:00:00:00:02:00 unknown -- flooding
```

Figure C.9: Screenshot of the controller at siteB's terminal during the message sequence between the controllers at siteA and siteB, where msg1, msg2, and msg3 is sent. At the end we see the user's ping packet is flooded.