VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SDN ŘÍZENÉ POMOCÍ IDENTITY UŽIVATELŮ
SDN CONTROLLED ACCORDING TO USER IDENTITY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MARTIN HOLKOVIČ
AUTHOR

VEDOUCÍ PRÁCE Ing. LIBOR POLČÁK
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce je propojení systému pro správu dynamických identit vyvíjený v rámci projektu Sec6Net s řízením SDN sítě. Pro řízení sítě je použitý kontroler Pyretic, který umožňuje tvorbu aplikací pomocí match-action pravidel. Bylo navrženo rozhraní mezi systémem pro správu dynamických identit a kontrolerem Pyretic, které bylo následně implementováno a integrováno do obou systémů. Byly vytvořeny různe případy užití týkající se bezpečnosti, směrování a účtování pro ověření správnosti konceptu. Tyto případy byly následně implementovány jako aplikace nad kontrolerem Pyretic. Všechny aplikace byly dle možností testovány v síťové laboratoři. Hlavním přínosem této práce je zjednodušení správy počítačových sítí a zároveň poskytnutí nových možností správcům těchto sítí a v konečném důsledku i jejich uživatelům.

Abstract

The aim of this work is to connect dynamic identity management system developed under the project Sec6Net with a control of SDN network. The controller Pyretic is used for network control, which allows application development by using the match-action rules. Interface between the identity management system and controller Pyretic is designed and implemented in both systems. To prove the concept, selected use cases related to security, routing and accounting are created. The use cases are implemented as applications for Pyretic controller. All programs were tested in networking laboratory according to the possibilities. The main contribution of this work is to simplify and improve the management of computer networks while providing new capabilities to administrators of these networks and ultimately their users.

Klíčová slova

SDN, dynamická identita, SIMS, Pyretic, OpenFlow, firewall, směrování, účtování

Keywords

SDN, dynamic identity, SIMS, Pyretic, OpenFlow, firewall, routing, accounting

Citace

Martin Holkovič: SDN Controlled According to User Identity, diplomová práce, Brno, FIT VUT v Brně, 2015

SDN Controlled According to User Identity

Prohlášení

Čestně prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Libora Polčáka a za pomoci uvedené literatury.

....
Martin Holkovič
May 26, 2015

Poděkování

Děkuji Ing. Liborovi Polčákovi, vedoucímu diplomové práce, za odborné vedení a pomoc při vypracovávání této práce.

Contents

Chapter 1

Introduction

Computer networks are an inseparable part of computer systems. Computer systems use networks to communicate and exchange information between each other. Similarly to computer systems, computer networks are constantly evolving. Every year, many new features and network protocols emerge. One of the most significant innovations in recent years is the specification of a new computer network architecture.

The new architecture, called Software Defined Networking (SDN) [12], primarily aims at accelerating innovation in computer networks while providing advantages such as lower price of networking devices. SDN architecture improves network management by separating control logic of switch from switch hardware, using a standardized configuration protocol OpenFlow [13] and by providing the global view of the network to the applications. Thanks to the better network management, it is possible to create more complex network solutions. The chapter 2 describes the new SDN network architecture, lists the advantages of the architecture and compares the new architecture with the current architecture.

One approach to improve network management and to create new features is using the knowledge of user identity [14]. If the network application is able to use the identity of a user, it is possible to configure such an application by high-level policy. Instead of creating rules for specific IP addresses, the rules are specified to a concrete user or group of users. For example, the network firewall can be configured to block all packets from user Alice instead of blocking all packets from IP address *172.16.28.172*.

The chapter 3 describes programming applications for SDN networks. Concept of SDN controller is introduced with description of controller interfaces. In this work, applications are developed for Pyretic controller.

The chapter 4 explains the meaning of the identity in the computer networks, ways how is the knowledge of the identities obtained and introduces the system SIMS [16].

Connecting an identity management system to a SDN controller is described in chapter 5. Chapter explains modifications, which are necessary to allow an application on SDN controller use user identities. Second part of the chapter describes use cases for providing the concept of connecting SIMS to Pyretic.

Chapters 6 describes implementation of connection between SIMS and Pyretic, and implementation of use cases. In chapter 7 use cases are tested and evaluated.

Chapter 2

Software-Defined Networking

For more than ten years, computer networks are using the same architecture without any important change [11]. During the same time, huge innovation were done in the computer systems. Because the network architecture is still the same, the architecture does not reflect needs of the current networks [8]. For example, the speed of innovation is relatively slow [1]. The network architecture called Software-Defined Networking (SDN) is a response to the several gaps of the current network architecture. This chapter describes the new SDN architecture, evolution and the benefits of the architecture.

2.1 Current networking architecture

Each network device consists of Forwarding information base (FIB) and Routing information base (RIB) as shows figure 2.1. FIB is hardware that forwards packets from and to network, provides access control, packet buffering, packet marking, shaping and scheduling. RIB is an operational unit of the switches that manages FIB and possibly processes packets. Current devices are responsible for maintaining its own FIB and RIB, which are highly integrated in current architecture. This high integration of FIB and RIB was designed in the early Internet architecture. The reason for this architecture was very unreliable hardware and architecture designers wanted to minimize source of the errors [3].

Integrated hardware and software results in high complexity of the switches, which makes switches more expensive. With this architecture, upgrading the network with a new feature is not possible without touching the physical hardware [4]. In the past, a lot of improvements were not accepted, not because they were bad but because they were not implementable [5].

Independently operating network devices are causing unecessary execution of activities on multiple devices. To achieve global view of the network, network devices must communicate with many other devices using various protocols. Achieving the global view of the network by communication between individual devices does not allow to create a complex view of the network. Without complex overview of the network state, it is not easy to do complex things in the networks [6].

Complexity of the devices with the current architecture increases complexity of the management. Another reason why is management of the devices so complex is that the current network architecture is not designed with focus on the network management but for lowering the barriers of applications deploying [7]. To operate and maintain a network, network operators are implementing complex high-level network policies with low-level vendor-specific configuration [22]. The low-level vendor-specific configuration of the devices is done by pushing a lot of commands individually on each device [11]. Pushing a lot of commands on each device leads to misconfigurations of the devices if the changes in the configurations are frequent and complex [23].

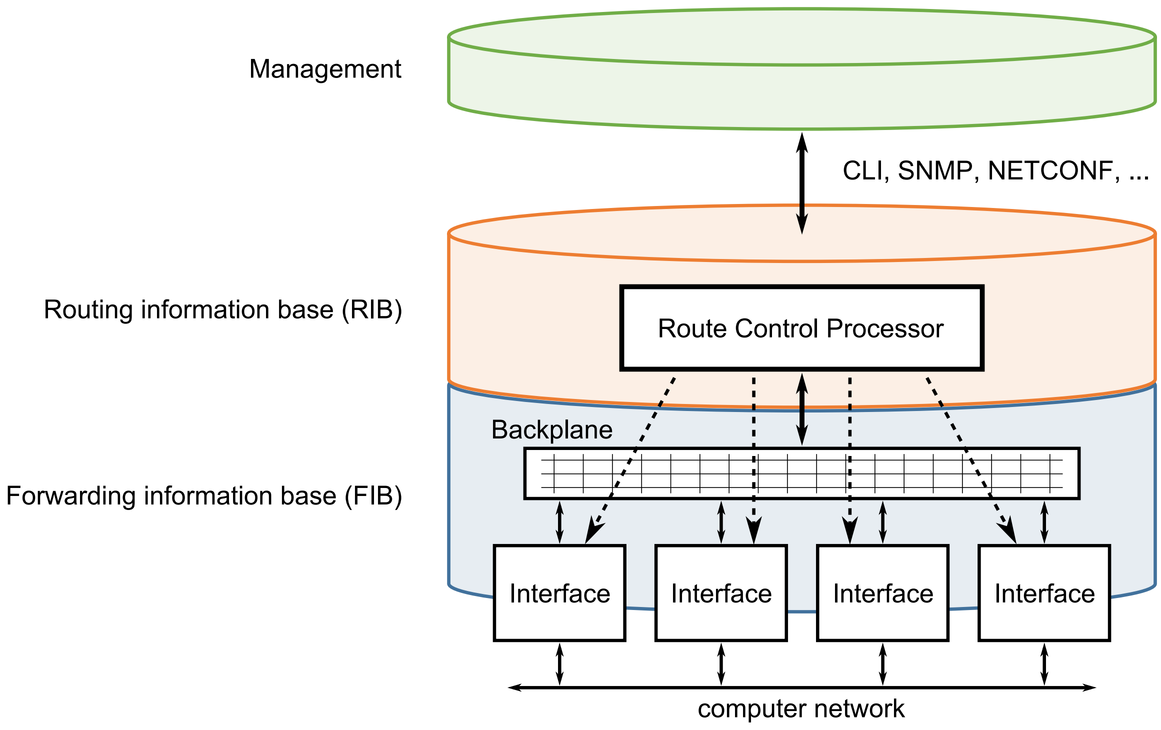


Figure 2.1: Switch consists of FIB, which is responsible for forwarding packets to and from network via network interfaces interconnected by backplane. Packets directed for switch itself are transfered to processor (RIB) over backplane. RIB is also responsible for FIB configuration. The whole switch is managed by various management protocols.

Current architecture is missing an universal way of configuring network devices. Command line interface (CLI) commands are not standardized and even commands on multiple devices from same vendor are not absolutelly the same. Protocol SNMP uses standardized tree of manageable elements to unite device configuration of different vendors. Problem with SNMP is that most of the manageable elements are read-only, what limits the SNMP to a very specific field [9]. The solution to the many read-only elements of SNMP is protocol NETCONF. When comparing the SNMP with NETCONF, NETCONF does not define a standardized database of manageable elements. In most cases, network devices maps NETCONF commands directly into existing CLI commands [10].

Several problems of the current architecture lead to the slow innovation [1, 21]:

- dependence on vendors and the IETF;
- control logic is tied to hardware;
- missing the complex network-wide view of the network;
- insufficient scalability of configuration testing.

2.2 What is Software-Defined Networking

The key feature of the architecture is the separation of the RIB (in SDN called control plane) from FIB (in SDN called data plane) on the switch. The data plane configuration is transfered by using a standardized protocol between control and data plane. By abstracting hardware details inside the data plane, control plane creates configuration without the necessity of hardware details knowledge. Figure 2.2 shows switch architecture with separed control plane.

Another key feature of SDN architecture is capability to move control plane to another device in the network. After the planes separation, control plane is implemented as a common software capable to be run on commodity hardware. This allows control plane to be placed on two different locations:

- **On a switch** - switch in addition to a specialized hardware includes a common hardware capable of running control plane;
- **Out of a switch** - control plane is placed on a different physical device (typically server), which remotely configures the data plane.

Physical relocation of control plane allows centralization of control planes from multiple devices. Multiple switches are managed by one device which brings new view of the network to the configuration logic. Figure 2.2 shows architecture of SDN switches with control plane on the same and different physical device. Centralization of the control plane is however not the primary objective of SDN architecture.

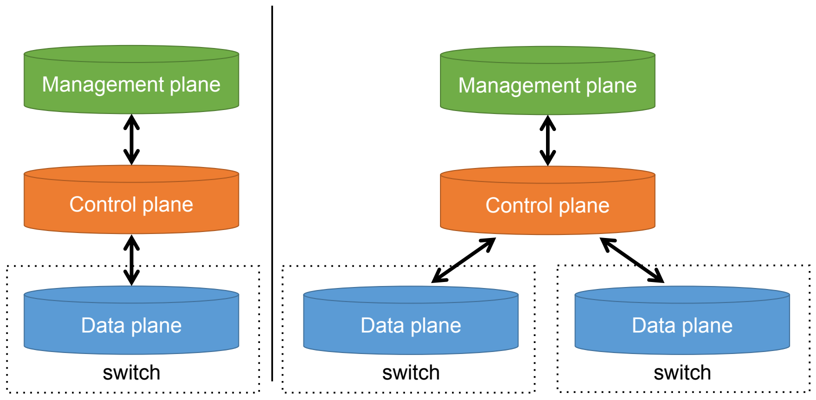


Figure 2.2: Switch on the left displays architecture with logically separated control plane, which is physically on the same device. Middle and right switch display architecture with logically and physically separated control control plane. Both control planes are configured from Management plane by whatever way.

It is very important to realize that SDN is just a new architecture of network devices which does not increase function capability of devices. SDN architecture also does not solve the problems tied to transfered/switched data, such as problems witch TCP/IP stack. Benefits of the new architecture come from new method of managing network devices.

2.2.1 Benefits of the Software-Defined Networking

Separation of control plane from data plane reduces complexity of devices which lowers their price. Separation of software and hardware development removes dependencies on vendors and IETF and allows faster innovation. Planes are developed independently as hardware and software. It is possible to connect external applications with control logic of network.

With standardized interface and abstracted hardware details, devices with multiple platforms and from multiple vendors are configured equally. Unified configuration simplifies automatization of configuration. By abstracting hardware details, pipeline processing of packets is replaced by bits manipulation. In pipeline processing, order of switch features execution is fully wired into hardware without possibility to change. Bit manipulation processing allows customization of features execution order and allows manage network by flows.

With more effective control logic of devices, it is possible to reduce number of devices. Functionality of some devices can be integrated into switch configurations, e.g. load balancer or firewall. It is possible to save money on electricity by shutting down redundant switches during low load (e.g. at night).

Switches are possible to divide into multiple slices, where each slice use different control plane. Administrator is able to test new features inside production network by creating new slice on switches. For example, administrator testing new routing algorithm creates new slice which will be used only by selected users for verification of the routing. Another benefit is gradual deploying new features by gradual shifting traffic from one slice to another.

SDN allows better virtualization of networks. New architecture allow us to work with switches as with pool of resources. The whole network is possible to represent as one big virtual switch. When the whole network is represented as one switch, managing applications do not have to take into account topology details. Network virtualization also allows physical topology change without necessity of changing network configuration.

2.2.2 Challenges with Software-Defined Networking

New network architecture with separed control plane brings new challanges to the network:

- **Scalability** - Since not every switch is capable of running control plane localy, control planes are deployed on servers. After deploying number of switches into network, one server will not be able to sustain loads and another server is deployed. To manage multiple servers as one, replica algorithms are used between the servers. Using replica algorithms increase response time to events. The aim is to increase number of switches which are managed by single server and to minimize server response time.
- **Reliability** - It is necessary to ensure correct operational state under failure . What happens if connection to control plane is corrupted? By separating the planes, the correct behavior of switch is no longer guaranteed. Should the switch uses its last configuration or stop its functionality? It is also necessary to secure full operation if one server fails. Is it enough to reconnect switches to new server without any additional information? One way to ensure availability is by using server replicas with proper coordination algorithms.
- **Consistency** - What happens if control replica does not have the same state? Can the replica server with outdated state provide correct functionality? Is it possible to duplicate all server inputs for replica server to ensure the same state?
- **Security** - Can the attacker reach the controler? Is control plane or the whole network safe if attacker gains control over one network switch? What if attacker deploys his or her own rogue control plane to the network? How is control plane protected from DDoS attack?

2.2.3 Evolution of Software-Defined Networking

The idea of the SDN network architecture is not as recent as it seems. During the last century, several architectures was created. All attempts have encountered two major problems [1]. There was no killer application which would take advantage of another architecture. Killer environment for the SDN is the data center and the cloud which did not exist in the past. Second problem of all atempts was hardware support for a new type of an architecture. Every device was using ASICs, whereas now devices uses TCAMs, FPGAs and NPUs. SDN architecture utilizes the concept of multiple predecessors:

- **Network Control Point (NCP)** - Introduced in 1981 by AT&T, separates the control plane from the data plane for voice network.
- **Active Networks** - Output of the DARPA project from 1994, where packets carry data and actions that are executed on network devices. Switches support the set of permitted actions, which administrator combines to produce the desired behavior of the network.
- **Routing Control Platform (RCP)** - Introduced in 2004, uses separated control plane and BGP protocol for configuration of the data plane. Control plane is centralized for the entire network.
- **OpenFlow** - Introduced in 2008, protocol OpenFlow allows the configuration of the flow tables on network devices, requiring no special hardware modification.

SDN was initially deployed only in the data center and on WAN links. New SDN networks start to be deployed in Internet Exchange Points, campus networks, wireless networks and home networks.

2.3 Transition to the new architecture

From abstract view of a network, switches are resources which use packets as an interface between each other. As long as the interfaces between resources remain the same, it is possible to replace one resource (switch with one architecture) with an absolutelly different resource (switch with another architecture). Because SDN switches are receiving and sending packets in the same way as the current architecture switches, it is possible to deploy SDN switches to network with current architecture switches. Problem with merging current and SDN switches is that applications on current switches are not compatible with SDN architecture. Solution to this problem is implementing applications for new architecture or combine multiple architecture on one device.

The first solution to the incompatibility of applications is to create new applications for SDN architecture. First way of creating new applications is to take application from old architecture and architecture dependent code replace with new code. Reprogramming applications for new architecure is straightforward, new application is capable of communicating with old application but application does not utilize benefits from new architecture. Programming a totaly new applications by utilizing benefits from new architecture is second way of creating new applications. Disadvantage is that current applications are deployed for years and are well tested in production networks.

The second solution to the incompatibility of applications is using multiple architectures on the same device at the same time. Device with combined architecture is capable of running well tested current architecture applications alongside with applications from new architecture which utilize benefits of new architecture. After receiving data from the network, switch has to choose which architecture processes received data. Two ways of architecture selection exists:

- **Ships in the night** - after receiving new data on switch, architecture is chosen based on port on which data were received or VLAN tag of the data;
- **Integrated** - all received data are processed by new architecture. Inside the new architecture, it is possible to choose which data will be sent to old architecture, which leads to more granular choosing of architecture.

Chapter 3

Programming SDN networks

This chapter is about programming applications on a control plane of the SDN architecture. One way of programming applications for SDN is directly on interface between data plane and control plane, which is called southbound interface. An example of southbound interface is OpenFlow which is described in this section. It is possible to create applications directly over OpenFlow protocol, but it not very practical.

To simplify and improve programming of applications for SDN network, a concept of interlayer is used. A interlayer is placed between data plane and applications nad it is called controller. Interface between controller and applications is called northbound interface. Nowadays, several northbound interfaces exist and every controller can use a different one. Controller used in this work is the Pyretic controller, which uses its own proprietary northbound interface. The Pyretic controller with its interface is described in this chapter.

3.1 Southbound interface

Southbound interface is the interface between data plane and control plane which has two main purposes. The first purpose is the definition of communication protocol between data plane and control plane. Even if control plane is placed on a different device, control plane is still communicating with data plane. A communication protocol is used for configuring data plane, sending packets from switch to control plane and collecting information from switches.

The second purpose of southbound interface is the definition of configuration protocol inside a data plane. Configuration protocol defines the form of configuration which switch receives from a control plane. From the definition of SDN, configuration protocol is not a hardware specific and uses abstraction of hardware details. One of the most known southbound interface is OpenFlow.

3.1.1 OpenFlow

OpenFlow [13, 24] is an open standard of southbound interface protocol developed by the Open Networking Foundation (ONF). A switch with OpenFlow support is called OpenFlow switch and by definition of OpenFlow consists of two main components:

- **Flow table** - performs a packet lookup and forwarding;
- **Secure channel** - creates secure connection with a control plane.

The OpenFlow protocol configures flow table on OpenFlow switches which are connected to a control plane over a secure channel. The architecture of OpenFlow is displayed in Figure 3.1.

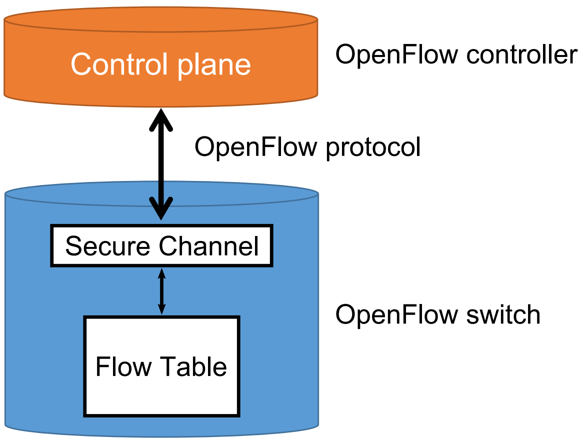
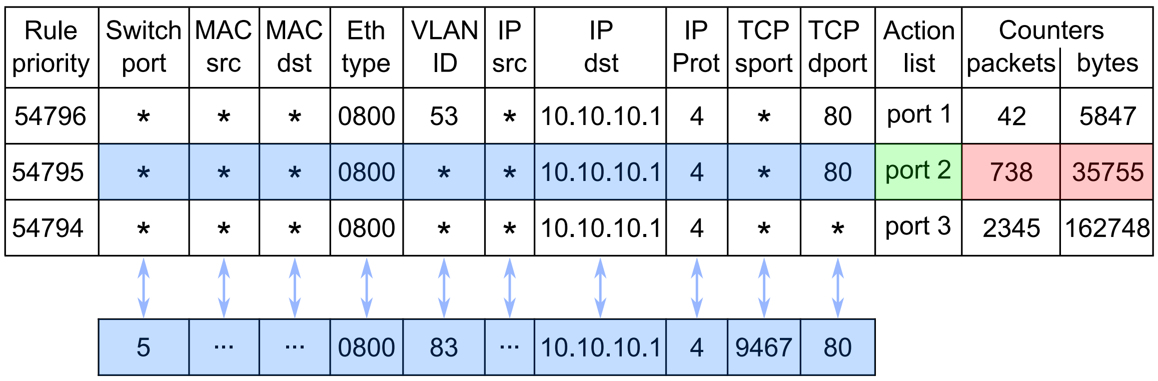


Figure 3.1: OpenFlow defines OpenFlow switch which consists of a flow table responsible for packet forwarding and a secure channel responsible for a connection with a control plane. Switch communicates with the control plane by OpenFlow protocol which is used for flow table configuration.

A flow table from an OpenFlow switch is the key part of the whole OpenFlow protocol. Forwarding decisions with modifications to forwarded packets are made in flow table which is configured by control plane. Figure 3.1 shows an example of the flow table with some fields omitted. Flow table consists of rules which define modifications to packets that meet certain criteria. After receiving a packet on switch interface, the switch process rules in a fixed order (defined by priority numbers of rules) and stops at the first rule where the packet meets the criteria. Based on the modifications defined in the rule, the switch modifies the received packet, updates the rule (explained in a later section) and sends the packet to a set of interfaces. Each rule in flow table consists of:

- **Priority** - a number which specifies the order in which the rules are processed. Switch starts with rule with the highest number and looks for rule step by step to rule with lower number;
- **Header fields** - 12-tuple of headers: Ingress Port, Ether source, Ether dst, Ether type, VLAN id, VLAN priority, IP src, IP dst, IP proto, IP ToS bits, TCP/UDP src port, TCP/UDP dst port. Every field has to contain a specific value or ANY, which matches any value. It is also possible to specify IP addresses with a netmask;
- **Action list** - a list with zero or more actions. There are two types of actions. The first type sets output ports to which the processed packet is sent. One packet can be sent to multiple output interfaces. OpenFlow also specifies special virtual ports, such as port to data plane, port to process packet by old architecture or port to send packet to all interfaces except the incoming interface. The second type of action is modification of header fields of packet. For example it is possible to rewrite IP address, rewrite the transport protocol port or add VLAN tag;

- **Counters** - every rule contains three counters. Two counters for counting amount of packets and bytes that was processed with concrete rule and one counter to measure how long is the rule installed.



Rule priority	Switch port	MAC src	MAC dst	Eth type	VLAN ID	IP src	IP dst	IP Prot	TCP sport	TCP dport	Action list	Counters packets	bytes
54796	*	*	*	0800	53	*	10.10.10.1	4	*	80	port 1	42	5847
54795	*	*	*	0800	*	*	10.10.10.1	4	*	80	port 2	738	35755
54794	*	*	*	0800	*	*	10.10.10.1	4	*	*	port 3	2345	162748
	5	···	···	0800	83	···	10.10.10.1	4	9467	80			

Figure 3.2: When looking for a rule that will process the received packet, the table starts to look starting with rules with higher priority numbers. All fields in the rule have to match with packet header fields to apply actions to the packet. Figure does not show VLAN priority and IP ToS bits fields. Rule with priority 54795 is the first rule that matched the whole processed packet. Action list containts only the output interface to which the packet should be sent. All header fields of a processed packet remains unmodified. When the packet is matched by any rule, counters of the matching rule are updated.

There are two modes of installing rules to a flow table. Reactive mode and proactive mode. With reactive mode, after a switch connects to a control plane, only a rule for sending all packets to control plane is installed. During processing of new packet, packet is sent to the control plane which decides what to do with packet and control plane also installs new rules to the flow table. Sending packets to control plane increase load of control plane and response time to forward new packets. In the proactive mode, all rules are installed immediately after switch connects to control plane. Therefore it is not necessary to send any packet from switch to control plane. Best way of installing rules is to combine both methods. After connecting switch to control plane, control plane installs rules so switch knows how to forward most of the packets. However, some packets switch still sends to control plane which increase capability of control plane.

Full operation of OpenFlow switch is displayed on Figure 3.3:

1. When a new packet is received from a network, the switch checks flow table for a first matching rule. If no rule is fount, packet is dropped. Otherwise counters of rule are updated and action list is used for next process;

2. After finding action list for packet, the packet is modified. Part of the packet modification sets an output ports. If no output is set, packet is dropped;

3. After packet header fields are modified, packet is send to output ports. In addition to physical interface, connection to control plane is also considered as output port which can be used for packet forwarding;

4. Based on packets received from switches or external events, control plane is capable of sending packets into network. Control plane can send a packet to any switch which sends the a packet to interfaces specified by control plane.

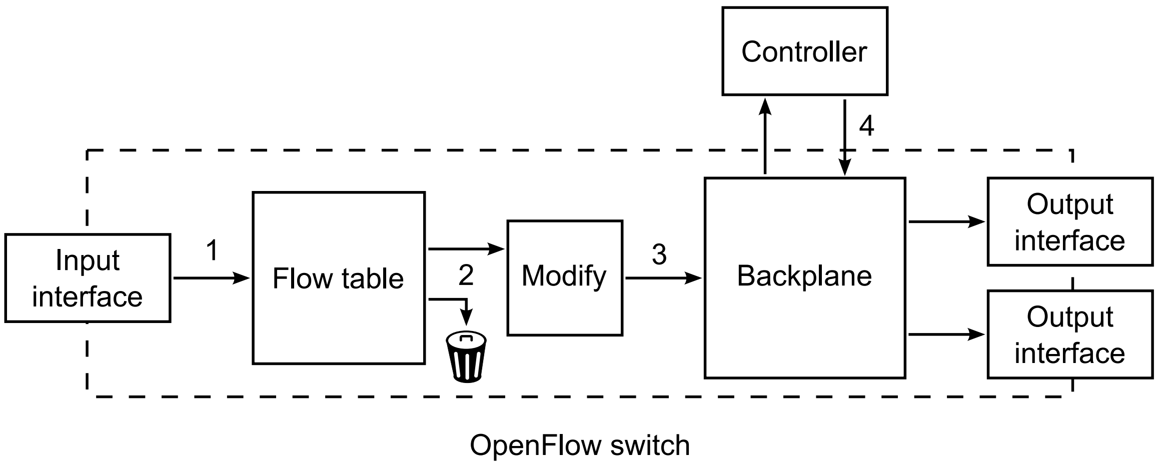


Figure 3.3: Operation of OpenFlow switch.

This section describes OpenFlow in version 1.0, because that version is used in next section. However, several new versions of OpenFlow protocol were released which bring several key enhancements. OpenFlow versions from 1.1 to 1.5 defines using of multiple flow tables, tunnel ports, MPLS support, IPv6 support, optical links support and many other new things.

3.2 Pyretic controller

Programming applications directly over a southbound interface is not simple. Even a hardware details are abstracted, programming is still very low level. For example, during the installation of new rules to a flow table, application ensures correct order of installed rules and communication with switches. Typical application based on southbound interface is one large monolitic application with hard expansion and reuseability. To ease applications development, concept named controller is introduced. A controller directly communicates with switches, process requests from application to switch and abstract several low-level details. OpenFlow controller is any controller which uses OpenFlow protocol as southbound interface. Architecture of SDN network (and specifically control plane) with and without controller is showed on Figure 3.4.

In the left part of the Figure, architecture of a control plane without a controller is showed. The control plane consists of one large monolithic application executing all necessary functions. In the right part of the Figure, architecture of a control plane with a controller is showed. Control plane is defined by modular architecture with several modular parts. A controller abstracts low-level information from the network and configures connected switches based on the actually running applications. In addition to data plane, both architectures communicates with management plane which is used for configuration of SDN applications. SDN architecture does not specify this interface. Example of interface

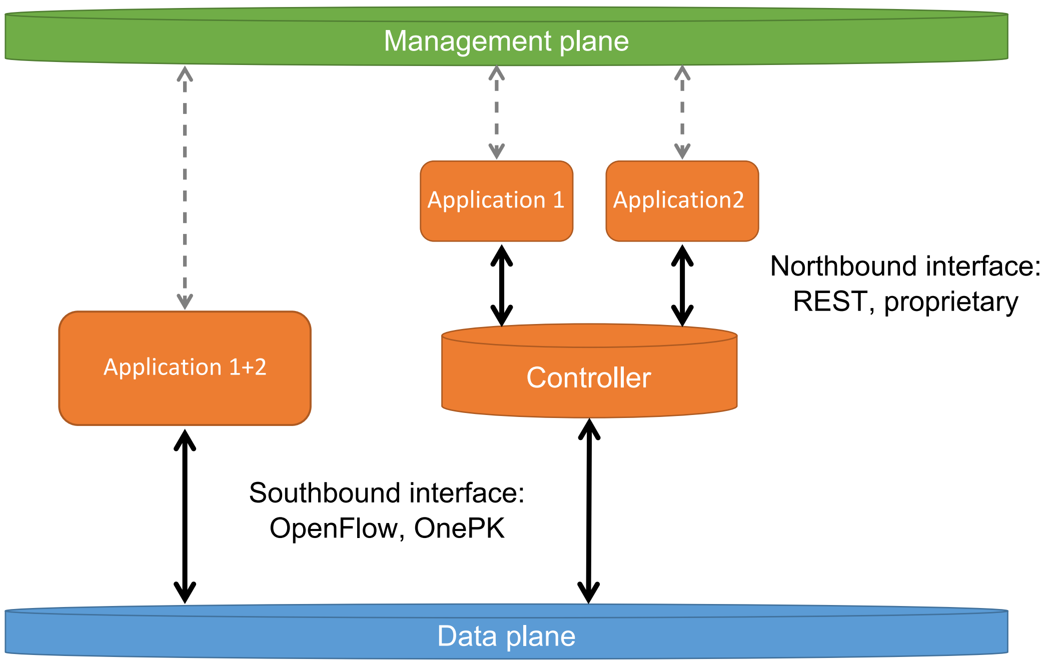

Management plane
Application 1
Application2
Northbound interface: REST, proprietary
Application 1+2
Controller
Southbound interface: OpenFlow, OnePK
Data plane

Figure 3.4: Architecture of control plane with and without controller.

between control plane and management plane is command line interface and locally stored configuration files.

A controller used in this work is called POX with Pyretic extension (also can be referred as Pyretic controller). The Pyretic extension is designed to work with any OpenFlow controller which supports OpenFlow 1.0 protocol. From the point of this work, POX is not important as Pyretic which solves most of the problems with applications programming. The biggest difference between the Pyretic controller and rest of controllers is way how the applications are programmed. The Pyretic uses proprietary interface to applications called northbound interface. Pyretic northbound interface uses high level representation of network configuration which allows easily reuseability of Pyretic applications.

3.2.1 Policies

The Pyretic uses term policy for definition of application. A policy is a set of rules which define what actions are done to processed packets. Basic representation of a policy rule can be represented as IF match THEN action, where:

- **match** - consists of OpenFlow table header fields and defines packets for which the policy is applied. In the case that match is not defined, policy is used for every processed packet.
- **action** - defines how a processed packet is processed and changed during the policy processing. Actions have the same capabilities as actions from OpenFlow table.

For easier packet manipulation, action list of policy rule is supplemented by three special values. Action identity, which leaves processing packet unchanged, action drop which drops packet on a switch and action to send a packet to controller. Action for sending packets to a controller modifies output port for a processing packet to virtual port to the controller. When a controller receives packet from a switch, controller saves the packet to buckets. Bucket is a data structure where packets from switches are saved and bucket also contains reference link to Pyretic policies. When the new packet is saved to a bucket, controller informs policies which are using that bucket that new packet has come.

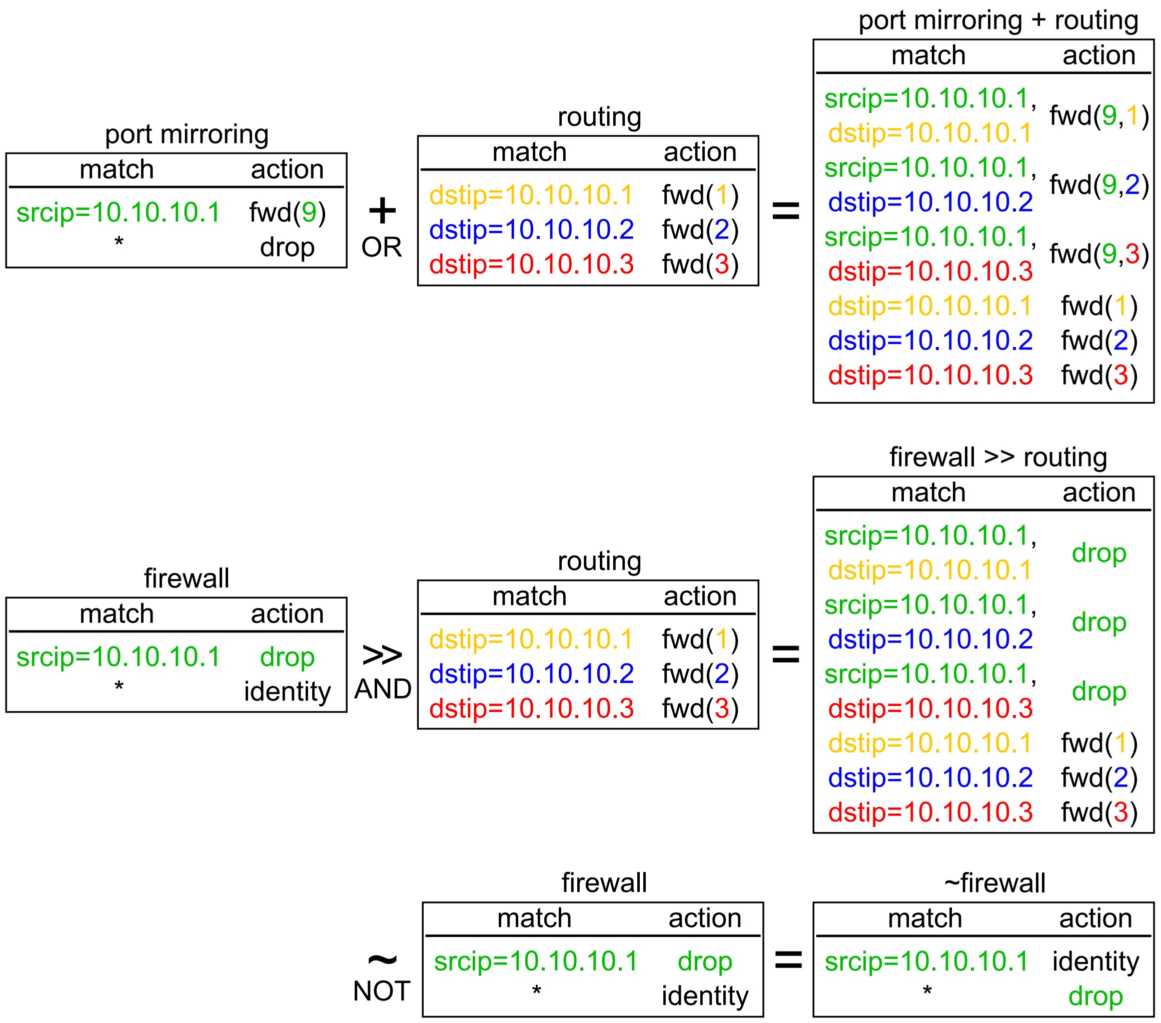
For creating a complex policies, Pyretic defines three operations for working with policies. Example of all three operations is showed in Figure 3.5. Operations are defined as:

- **negation** - Operation of negation is possible to apply only for policy rules with action identity or drop. By negation a policy rule with action identity, an action is replaced with drop value and vice versa.
- **sequential processing** - For better understanding, it is also possible to name the operation as AND function. Function AND merges two small policies to one big policy. By executing new created policy on a input packet, packet is modified in the same way as when the packet is processed by both policies sequentially.
- **parallel processing** - For better understanding, it is also possible to name the operation as OR function. Function OR merges two small policies to one big policy. By executing new created policy on a input packet, packet is modified in the same way as when the packet is duplicated and sent to two policies for parallel processing. Each policy modifies the packet in different way and expect that modification to the packet is successful. For example, one policy modifies source IP address of a packet and another policy modify source MAC address. After execution both policies in parallel, both IP address and MAC address are modified. Function OR has limit, that parallel policies can not modify the same field of a processed packet.

Pyretic extends number of header fields used for packet matching and packet modification with a concept called virtual header. Virtual header is encoded an stored inside other header fields that packets contain. In the actual version of the Pyretic, virtual headers are saved inside VLAN header (fields VLAN ID and VLAN PCP). When a policy modify a virtual header of a packet, value of virtual header is encoded to a number which is saved in binary form to VLAN header fields. After encoding a virtual header to the VLAN header, switches which process incoming packets uses value decoded inside a VLAN header for decoding virtual header value.

Architecture of controller POX with Pyretic extension is showed in Figure 3.6 and consist of:

- **POX** - OpenFlow controller which is responsible for direct communication with OpenFlow switches;
- **Backend** - Backend of the Pyretic which is connected to any OpenFlow controller. Actual version of Pyretic support only version OpenFlow 1.0.
- **Network topology** - Module monitores messages from OpenFlow switches about topology changes and contain representation of a network. When a new topology change is detected, internal represetnation is updated and each application policy is notified about change.



port mirroring

match	action
srcip=10.10.10.1	fwd(9)
*	drop

+ OR

routing

match	action
dstip=10.10.10.1	fwd(1)
dstip=10.10.10.2	fwd(2)
dstip=10.10.10.3	fwd(3)

=

port mirroring + routing

match	action
srcip=10.10.10.1, dstip=10.10.10.1	fwd(9,1)
srcip=10.10.10.1, dstip=10.10.10.2	fwd(9,2)
srcip=10.10.10.1, dstip=10.10.10.3	fwd(9,3)
dstip=10.10.10.1	fwd(1)
dstip=10.10.10.2	fwd(2)
dstip=10.10.10.3	fwd(3)

firewall

match	action
srcip=10.10.10.1	drop
*	identity

>> AND

routing

match	action
dstip=10.10.10.1	fwd(1)
dstip=10.10.10.2	fwd(2)
dstip=10.10.10.3	fwd(3)

=

firewall >> routing

match	action
srcip=10.10.10.1, dstip=10.10.10.1	drop
srcip=10.10.10.1, dstip=10.10.10.2	drop
srcip=10.10.10.1, dstip=10.10.10.3	drop
dstip=10.10.10.1	fwd(1)
dstip=10.10.10.2	fwd(2)
dstip=10.10.10.3	fwd(3)

~ NOT

firewall

match	action
srcip=10.10.10.1	drop
*	identity

=

~firewall

match	action
srcip=10.10.10.1	identity
*	drop

Figure 3.5: Three operations for Pyretic policies are defined. Operation OR and AND combines two policies which are processed sequentially or parallel to one policy with the same output. Operation NOT negates actions of one policy.

- **Buckets** - Data structues which contains packets from switches and links to application which ara buckets using.
- **Applications** - Applications which create configuration of a whole network. Each application is independent and capable of read network topology information and packets from buckets.
- **Policy compilation** - Policies from applications are combined based on the policy operators which defines relations between policies. After creating an one policy that describes the whole network configuration, policy is compiled to OpenFlow rules which are installed in switches.
- **Runtime** - Core of the Pyretic, that control packet processing and policy updates.

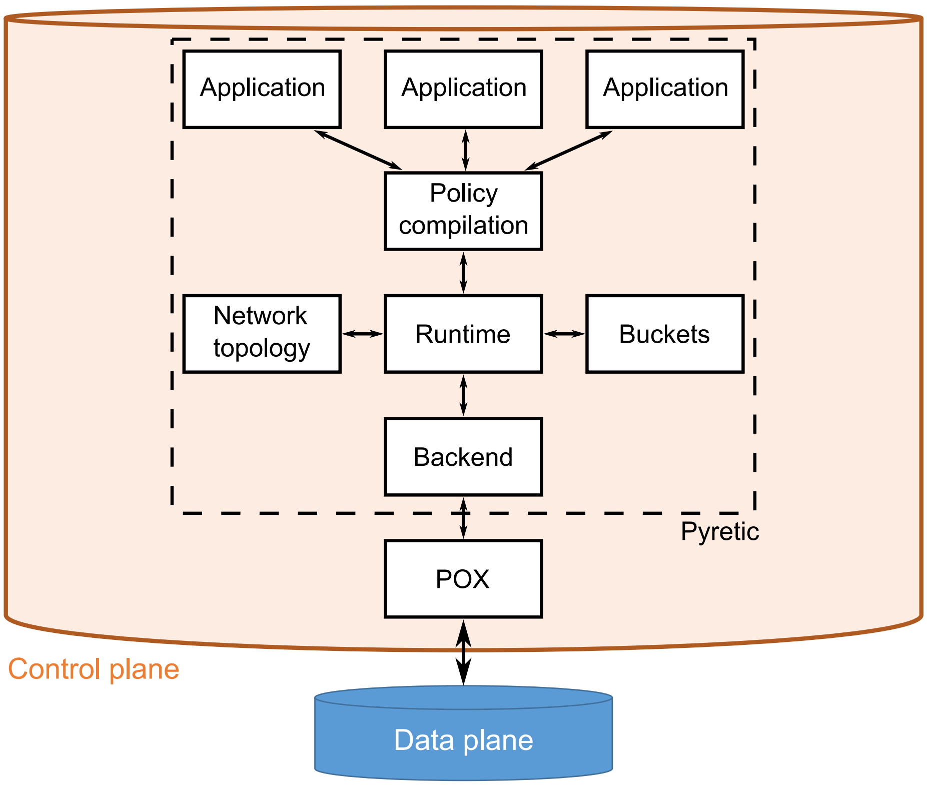


Figure 3.6: Architecture of the POX controller with Pyretic extension.

3.3 Northbound API

Interface between controller and applications is not standardized as is the case of South interface (OpenFlow). The Pyretic uses custom proprietary interface in the form of Python class. Application defined by Python class consists of method for calculating a network policy. Python class can also consists of method for processing input packets and methond for processing topology change information.

In addition to application definitions, it is also necessary to define relations between applications. Relations are defined sequential and parallel operators as is showed in Figure 3.7. Policies from each application are combined together by usigng policy operators and compiled into OpenFlow cofiguration. OpenFlow cofiguration is sent to OpenFlow switches to update switch flow tables.

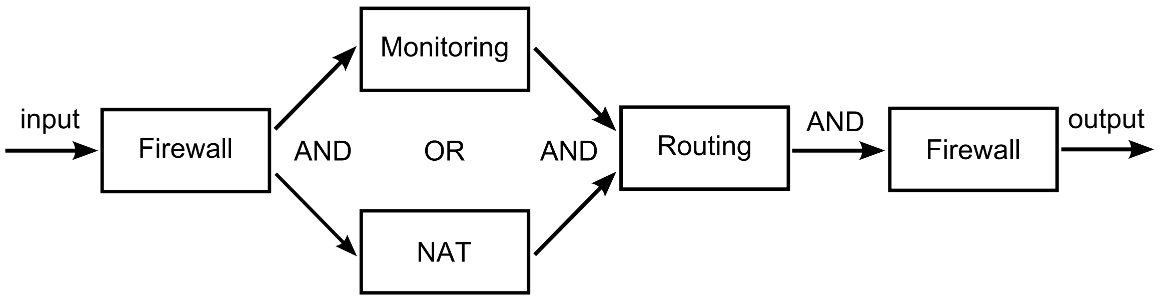


Figure 3.7: After creating policies for Pyretic, it is also necessary to define relations between policies by using policy operators.

Chapter 4

Dynamic identity

In order to use knowledge of the identity of users in network management, it is necessary to first detect user identity. Detection of identity is gathered by analyzing the use of identifiers which identify the user or device within a specific context. In computer networks, identifiers are used by network protocols and applications. The analysis of identifiers is aimed to detect correct identifier values, beginning and end of time when the identifiers are used by users or devices, and links between detected identifiers. Content of the chapter is defition of terms identitity and identifier, description of multiple ways to detect identifiers, method of linking identifiers and description of one identity management system.

4.1 Identity

According to the terminology about privacy defined by Pfitzmann and Hansen [14], an identity is a set of attribute values which distinguishes one subject from all other subjects within any set of subjects. The identity of an individual subject includes many partial identities. Each partial identity consists of a union of attribute values and represents the subject in a specific context or role. The subject can be regarded as human being, a legal person or a device. In this work it is assumed that device is part of the identity of one person (person using device is called user) at the time.

The identity of the person can be expressed by identity of devices which the person uses. User of device can use the device for several roles. Each role of user is called partial identity. Partial identity can be assigned to device and user of that device. On every device used be user can the user be part/act in several roles, each is referred to as a partial identity of the device and the user. Figure 4.1 displays identity of person which uses two devices, notebook and smartphone. Notebook has obtained IP addresses via DHCP protocol and SLAAC protocol. IP addresses are used by SIP for VoIP communication and by HTTP for authentication of user. At the same time the same user is using a smartphone with IP address assigned from DHCP. User has sent e-mail via SMTP protocol from smartphone and as well as with laptop, he used HTTP protocol for user authentication.

4.2 Identifiers

An identifier is an attribute unique for a subject or group of subjects. Identifiers have usually a form of a number, a name or a bit string [14]. In the context of computer

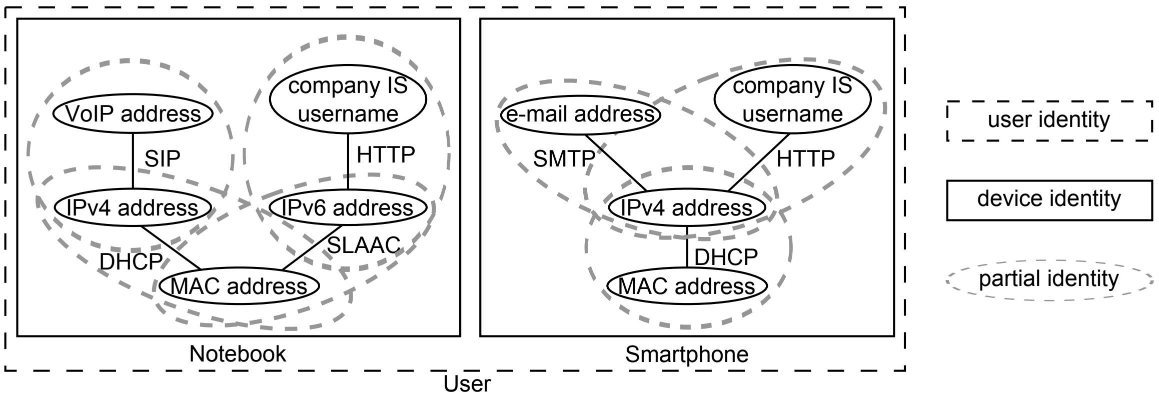


Figure 4.1: An example of an identity of a person consisting of partial identities connected to a notebook and a smartphone, which the person uses. Both devices participates in multiple roles. Each role has a different partial identity with different set of identifiers.

networks, identifiers are used by network services for identification and to distinguish their participants (users and devices). Identifiers are divided into two groups (see Figure 4.2):

- user identifiers - e-mail address, PPP login, SIP username, ICQ number and others,
- device identifiers - IP address, MAC address, DUID value and others.

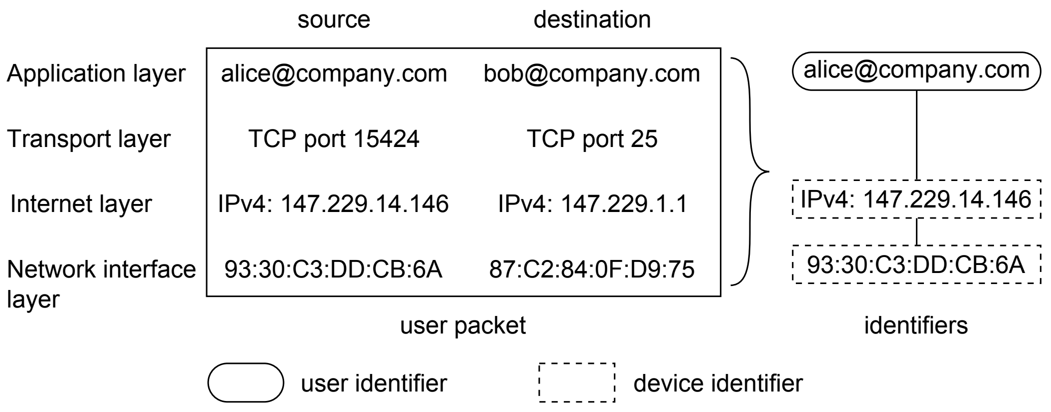


Figure 4.2: Packet sent by user contains identifiers belonging to device and one user identifier. By linking both types of identifiers it is possible to obtain information which is later used for detection of user identity from device identifiers.

Despite the fact that management of computer network is based on user identities, knowledge of user identifiers is not sufficient. Most of packets sent to network does not contain user identifiers. By analysing packets without user identifiers it is not possible to determine which user identity packets belong. For this reason it is necessary to have knowledge of device identities which are linked with identities of users. Figure 4.2 displays packet which is part of e-mail communication for sending message from address “alice@company.com” to address “bob@company.com”. In addition to e-mail addresses, by analyzing this packet it is possible to detect IP address of user with e-mail address “alice@company.com”. Next packets sent from detected source IP address belongs to the same user.

Identifiers, even if they are unique, do not require having the same value over time. Some applications and protocols, do not assign identifiers to a person or a device forever. Applications have a pool of identifiers and assign identifiers only temporary. Temporary assigned identifiers are called dynamic identifiers and identities which consist of dynamic identifiers are called dynamic identities. With dynamic identifiers, one can not never assume that if he or she saw a person using a attribute in the past that the same person is also using the same identifier at that time.

4.3 Identity management

Nowadays, there is not an universal way to gather the knowledge of user identities. Applications by themselves have only the knowledge of partial identities and usually they do not have the interface to combine the identities together. The knowledge of all identities can be obtained by gathering all partial identities from applications and by combining partial identities together. In this work, the knowledge of identities gathered by combining the identifiers is called identity database.

4.3.1 Gathering the partial identities

To reveal a partial identity, it is necessary to obtain identifiers from which the identity is comprised. In addition to obtaining identifier values, gathering of identifiers is also aimed to detect beginning and end of identifier validity together with context in which the identifier is used. Knowledge of identifier duration validity is necessary to the fact that not all of identifiers are associated with the user or device permanently. With the context of identifiers it is possible to determine set of identifiers which are part of the same partial identity.

The following list describes several ways to gather information about identifiers. Each of the methods has certain advantages and disadvantages which make every method appropriate for different type identifiers and environment in which they are used. For example, a distributed dynamic assignment of IP addresses has different specifics as authentication by login name on server.

- **Static configuration** - An administrator of the network manually creates identity database from the information obtained in the networking application and service configurations.
  - **advantages** - When the identity database is correctly maintained, it is the most accurate solution which does not require changes to the newtwork applications or network infrastructure.
  - **disadvantages** - The database must be frequently and adequately updated otherwise the number of errors will make this method unusable. It is not always possible to predict how the identifiers are to be assigned.
- **Log information** - An probe application is deployed into the network. The probe application periodically inspects log files from relevant network applications and collect identifiers.

  - **advantages** - Most of network applications are already using log files and it is not necessary to modify the network applications. It is possible to verify detected identifiers by backward inspection of the log files, e.g. during an incident investigation.
  - **disadvantages** - When an network application does not log every desired information to log files, it is necessary to customize the network application. Modifications of network application creates an additional cost for future updates of the network application. Whenever the format a file changes, it is also necessary to modify the parsing algorithm of probe application.

- **Network traffic** - An probe application for inspecting network traffic is deployed into the network. The probe application captures and analyses network traffic of network applications which contains user identifiers.
  - **advantages** - The analysis of the network traffic does not requires any modification of network applications. It is possible to update network application automatically without any additional action of the administrator if the communication protocol remains the same.
  - **disadvantages** - To analyse all the network traffic, the probe application has to analyse traffic from many places at the same time. Missing packets and wrong ordered packets are source of errors. Encrypted communication is a challenge since the probe application needs decryption keys to analyse packets.

- **Application programming interface (API)** - This method is similar to gathering information from the log file. However instead of parsing log files created by the network applications, network applications send messages to the analysing probe application.
  - **advantages** - By customizing the additional information sent by the API, the accuracy of this method is increased. If the information sent to the API is stored, it is possible to backward verify the identity database.
  - **disadvantages** - It is not very common that network applications contains API to advertise information about active identifiers. Introduction of API to an existing network applications is generally more cumbersome than a log file analysis.

4.3.2 Linking the identifiers

After detection of identifiers (which is described in the previous Section 4.3), identifiers are sent to a central point where are linked with every identifier from the same partial identity. Gradually as detected identifiers are linked, complex view of the network is created. This complex view is used to express partial identities, identities of users and devices.

An example is the identifier of the login user name to an information system, which is associated with the IP address from which the user subscribes.

Partial identities are linked by common identifiers to create a view of the person or device identity. The linkage allows to match communication of one partial identity by an identifier from a different partial identity. The combination of the identifiers must fullfil some requirements [17] but it is out of scope of this work. Figure 4.3 displays combination of two partial identities which belong to the same person.

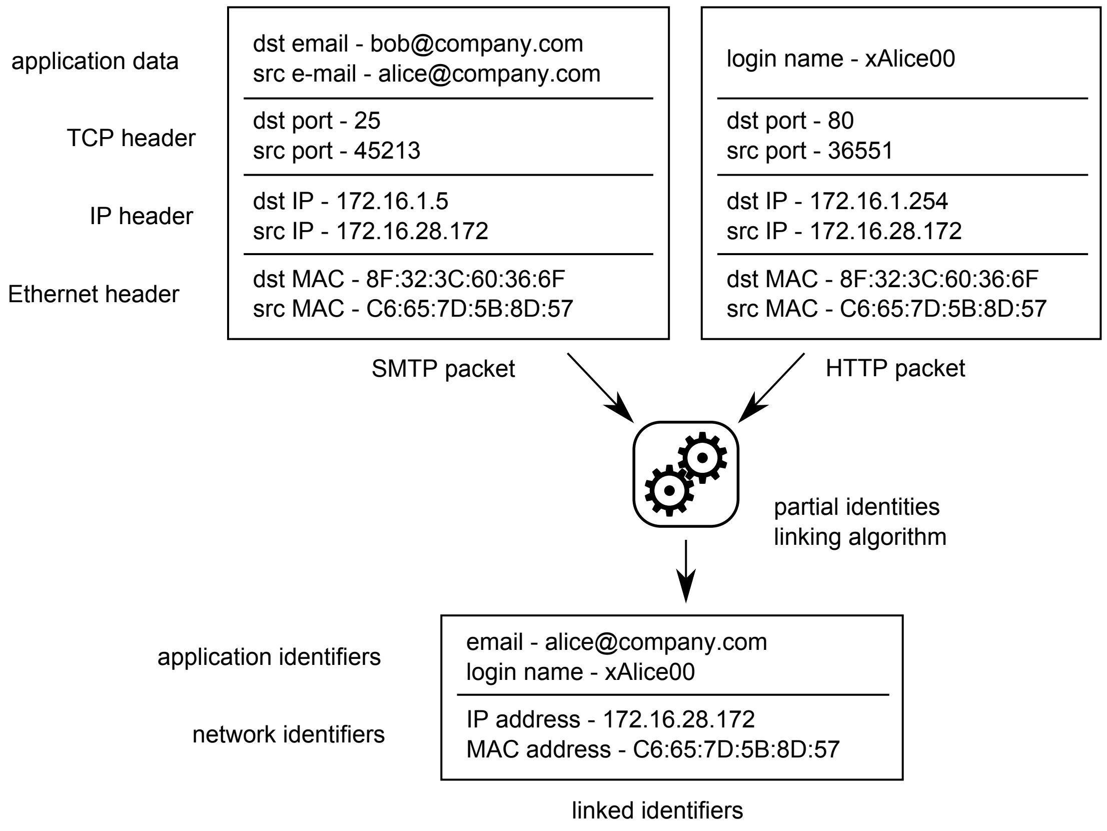


Figure 4.3: The computer with the MAC address C6:65:7D:5B:8D:57 and IPv4 address 147.229.8.53 is used only by one user. The user uses SMTP service to send an email from address alice@company.com and HTTP service to authenticate on web server as user xAlice00. Each service represents a different partial identity and contains a different set of identifiers. Identifiers from both services are linked together to create a union of the identifiers which belong to the same user.

4.4 Sec6Net Identity Management System

An example of a system for managing user identities is Sec6Net Identity Management System (SIMS). SIMS is developed at the Brno University of Technology, Faculty of Information Technology as a part of the project "Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet" (Sec6Net) with the identification code of VG20102015022. Sec6Net project (including system SIMS) focuses on the lawful interceptions which creates several limitations that are described in Chapter 5.

Architecture of the SIMS system is shown at Figure 4.4 and is based on ETSI standards (mostly ETSI TR 102 528 [18]). The system consists of two main parts: Administration Function (AF) and Intercept Related Information Internal Interception Function (IRI-IIF)

- **Administration Function (AF)** - Administrative function handles requirements for monitoring identities which are defined via identifiers (eg. identity with IPv4 147.229.14.146). Based on the requests, the AF configures IRI-IIF.

- **Intercept Related Information Internal Interception Function (IRI-IIF)** -

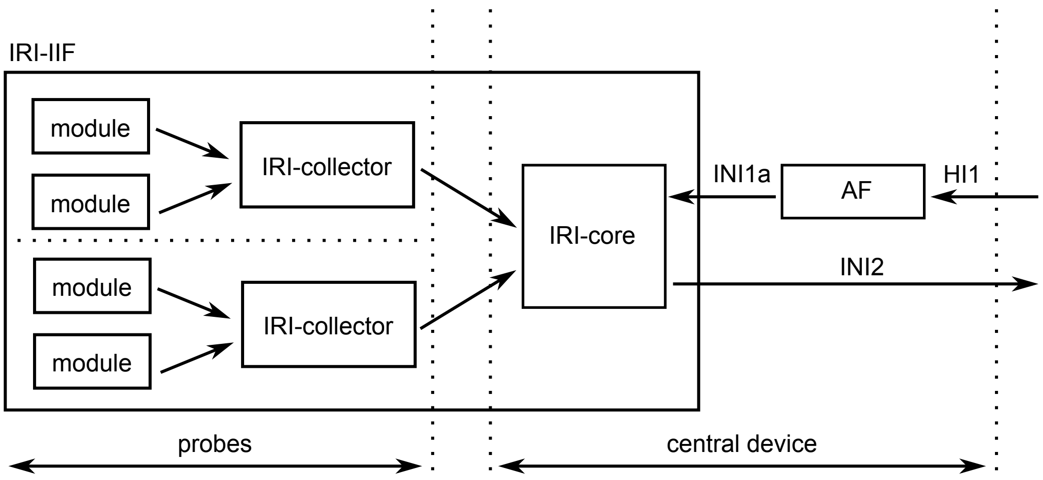


Figure 4.4: System SIMS consists of modules, IRI-collectors and IRI-Core, which together form a block IRI-IIF and take care for the management of user identities. The second part is the block AF, which configures which user identities should be sent by IRI-IIF for further processing.

> The role of the IRI-IIF is identity management based on detected identifiers. Monitored identifiers are sent for further processing.

Standards ETSI also describe AF and IRI-IIF interfaces:

- **HI1** - Monitoring requirements for selected identities are sent via HI1 interface. Each request includes: monitored identifier, level of monitoring (which identifiers are sent by IRI-IIF), begin and end time of monitoring.
- **INI1a** - The interface between the blocks IRI-IIF and AF by which AF announces IRI-IIF list of monitored identifiers.
- **INI2** - Output interface of the IRI-IIF and whole system SIMS. After detection of a change in the monitored identity, IRI-IIF sends related identifiers to the external system for further processing via the interface.

Under the project Sec6Net [15] the IRI-IIF is divided to IRI-Core, IRI-collector and modules:

- **module** - Modules detect identifiers from different sources, as described in chapter 4.3.1. After detection of identifiers, modules send identifiers to IRI-collector which is located on the same device.
- **IRI-collector** - IRI-collector receives identifiers from several modules and forward them to IRI-Core.
- **IRI-Core** - IRI-Core is responsible for linking identifiers, which are received from IRI-Collectors. IRI-Core configured by the AF sends identifiers for further processing.

The operation of system starts with an analysis of input data by modules. Modules are maintaining state information for each partial identity. After detection of new event, module sends a message with detected identifiers and with description of the event. IRI-Collector marks received data from modules with identification string and sends data to IRI-Core. IRI-Core add or remove identifiers from its identifier database based on type of event received. After receiving message from IRI-Collectors, IRI-Core checks whether the message contains an identifier or one of the identifier is linked with the identifier for which monitor request was received. If some identifier is monitored, according to level of monitoring set of identifiers associated with monitored identifier is send to INI2 interface. An example of system operation is shown in Figure 4.5.

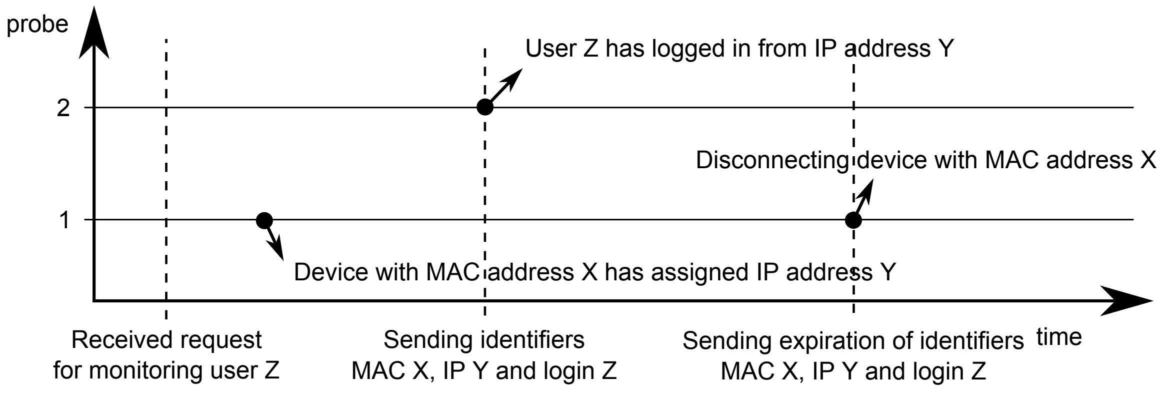

probe
2
1
User Z has logged in from IP address Y
Disconnecting device with MAC address X
Device with MAC address X has assigned IP address Y
Received request for monitoring user Z
Sending identifiers MAC X, IP Y and login Z
Sending expiration of identifiers MAC X, IP Y and login Z
time

Figure 4.5: Example of operation system SIMS with user Z and two probes in network. After the launch of system SIMS and the connecting of both probes to SIMS, monitoring request for user Z is received. After some time probe no. 1 detects assignment of IP address and after a while probe no. 2 detects authentication of user Z. By linking identifiers from both events, message with all identifiers is sent out of system SIMS. After some time probe no. 1 detects expiration of IP address and based on that information, all links with identifier are removed and identifiers are sent out of system SIMS.

System SIMS uses graph representation of received identifiers and links between identifiers. Node of the graph represents one identifier and edge between two nodes represents identifiers which belong to the same partial identity. Level of monitoring in the graph representation is represented by graph algorithm which take into account type of identifiers and distance to selected identifiers [17]. Issues connected with levels of monitoring are beyond the scope of this work, however for imagination maximum level of monitoring is explained. Maximum level of monitoring includes all identifiers from which exists path to the monitored identifier. The Figure 4.6 shows an example of a graph which represents identifiers along with maximum monitoring level of selected identifier.

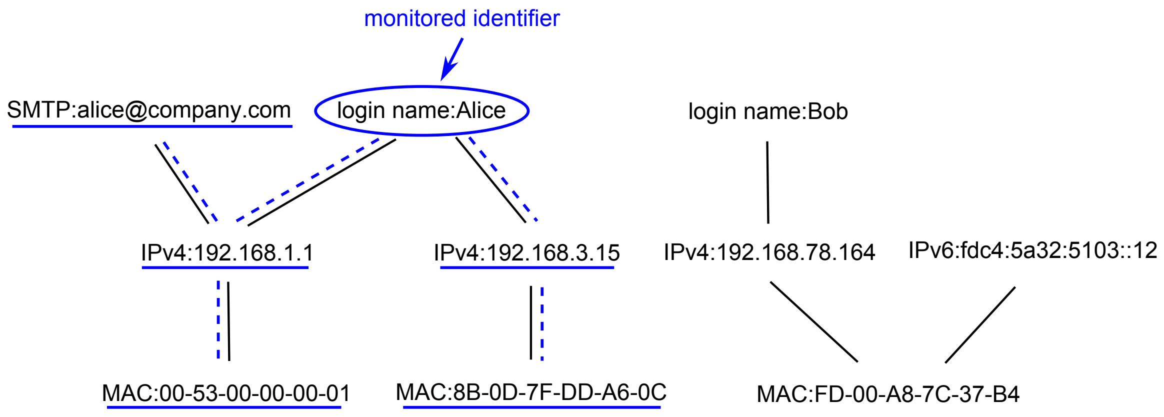


Figure 4.6: A visual representation of the detected identifiers and connections between themselves. Identifier *login name:Alice* is monitored with the maximum level of monitoring. All identifiers that belongs to the monitoring range are underlined and the leading edges are marked with the dashed lines.

From the point of deployment view, system SIMS is divided to central device and probes. Probes consist of modules and IRI-collector, and are deployed in multiple locations on the network. The central device containing IRI-Core and AF is placed only in one place in the network. Figure 4.7 shows deployment of system SIMS into a computer network with one probe for analyzing network traffic and one probe integrated inside the application through the API.

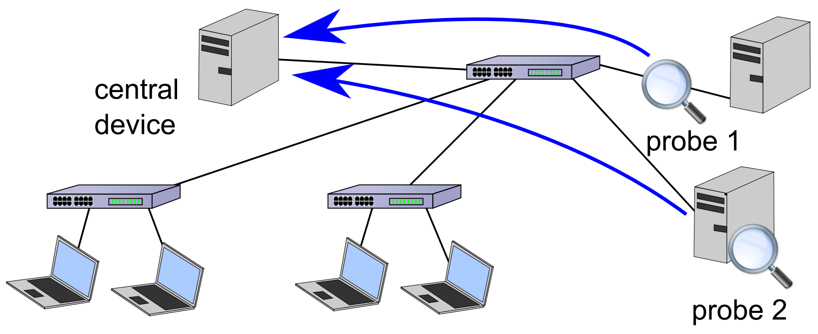


Figure 4.7: Deployment schema of system SIMS into a compute network. The probe no. 1 contains module for analyzing network traffic that flows through the line to the server. On the probe no. 2 is module integrated with the application by using the API. Both modules are sending detected identifiers over the network to central device.

Chapter 5

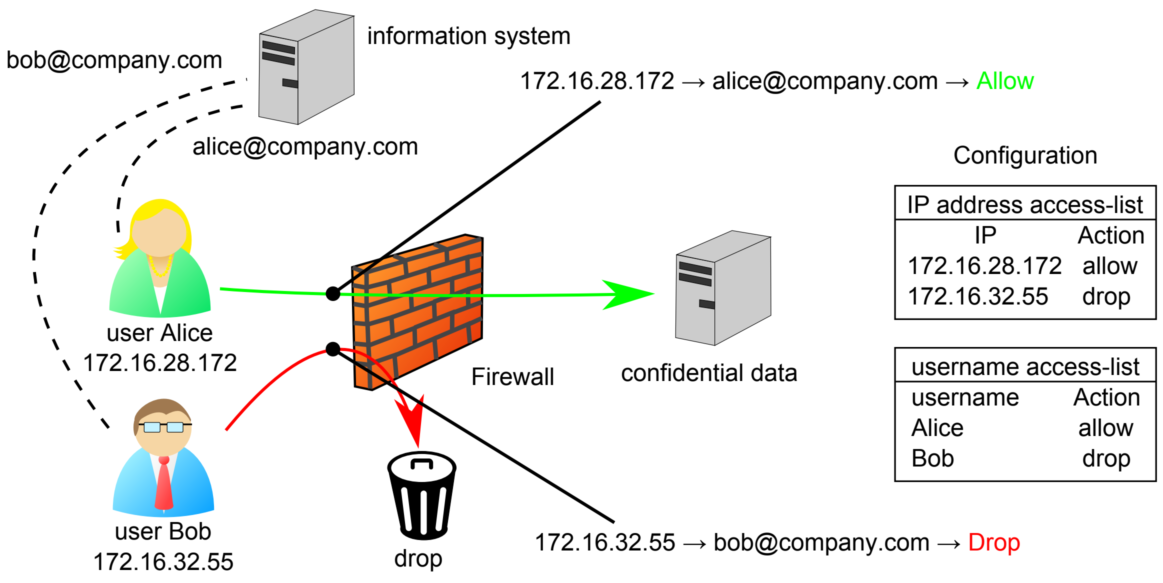
Design

In todays networks, configuration of network devices (switches) is based on devices connected to a network. For example, configuration of firewall is defined by a list of rules which consists of IP addresses that are or are not allowed to communicate. Disadvantage of this approach is that a device configuration consists of network policies defined for end users instead for devices. To create a device configuration from a network policy, administrator has to know which user or network service (which is not used by ani user) has assigned which IP address. If the mapping between users or network services and IP addresses has changed, it is necessary to update configuration of devices. Manual update of configuration is not a efficient option for very dynamic networks where users are allowed to use new devices (eg. smartphones) anytime.

With knowledge of user identities from identity management system inside a SDN controller, it is possible to extend capabilities of a network. By using the knowledge of user identities, it is possible to use identifiers as part of a controller configuration. For example, instead of configuring firewall with list of IP addresses which are blocked, list of e-mail addresses is used. Only devices with a user identity which does not contains any e-mail address from list of blocked e-mail addresses are allowed to communicate.

Figure 5.1 shows a difference between configuration of the firewall by using a device identifier (IP address) and by using a user identifiers (login name). Both configurations limit access to confidential data for user *Alice* only. Traffic from user *Bob* has to be dropped on the firewall. With the configuration by using IP addresses, administrator of a network has to ensure that users are always using the same IP addresses. Otherwise, it is possible that user *Bob* reaches confidential data or user *Alice* do not. Configuration based on usernames requires to know identity of users. Before connecting to the confidential data, user has to authenticate on company information system with login username. Based on the authentication information, IP addresses of devices are linked with username of device users and converted to configuration for firewall. Only device with authenticated user with username *Alice* is allowed to access the confidential data.

This work is aimed on connecting system SIMS with Pyretic controller. Before creating applications based on user identities, it is necessary to modify SIMS, Pyretic controller and create interface between both systems. Content of this chapter is aimed on required changes and later in chapter, several use cases are demonstrated.



IP address access-list

IP	Action
172.16.28.172	allow
172.16.32.55	drop

username access-list

username	Action
Alice	allow
Bob	drop

Figure 5.1: A company firewall is used to protect confidential data by droping packets from unauthorized users. Two examples of configuration are displayed on the right side of the Figure. The first configuration is based on IP addresses and the second on usernames.

5.1 SIMS

SIMS manages user identity database even without running SDN controller. It is assumed that system SIMS is running all the time. When a new SDN controller is connected to a network, controller needs to know what is the actual state of network. However, actual version of SIMS does not have interface for requesting actual network state. One way of creating interface for receiving actual network state is with interlayer between controller and SIMS. Interlayer runs on a device with SIMS and processes all data sent to HI2 interface by SIMS. Interlayer creates a network state of the network from HI2 messages and is able to send actual network state to a new connected controller.

With SIMS, it is not possible to monitor every user identifier e-mail. This limitation exists due to architecture of SIMS which is aimed for lawful interceptions. If a SDN controller wants to receive all data about user identifiers which are tied to some user identifier (such as e-mail address), controller has to create a monitoring request for each unique value of e-mail address. The Pyretic controller should know all unique values of identifier and immediately, when the controlles starts to run, controller creates monitoring request from each unique value.

Another challange with monitoring requests is that each monitoring request has to have unique name. Using identical name of the request and indetifier value as solution is not sufficient, because the name of request has to be unique globally even for expired requests. History of requests is saved on SIMS and after restarting SDN controller, history is not cleared. Solution uses interlayer between SIMS and controller. Interlayer receives requests for monitoring and remembers the whole history of requests. When a new request for identifier monitoring is received and identifier is not yet monitored, interlayer sends new monitoring request to SIMS where interlayer creates a unique monitoring request name. Unique monitoring request name is created by concatenating value of monitored identifier with revision number of local history of identifiers.

SIMS receives messages from modules and based on the monitored identifiers, new messages to HI2 interface are sent. Level of the monitoring is the highest level, which means any identifier with path to monitored identifier (inside graph representation of the network) belongs to monitoring range. Login username to web information system is used for authentication of users and for creating monitoring requests. List of supported identifiers by SIMS has to contain login username and also a probe for detecting login username inside a network. One way of possible detection of login usernames is to create probe inside a web application for user authentication.

To express location of user inside a SDN network, new identifier for SDN location is defined. Location of users inside a SDN network consists of unique ID of switch (called datapath ID) and ID of network interface. SDN controller contains a module for SIMS which detects location of users and send data to SIMS.

5.2 Controller

The controller uses virtual header field to mark each packet with group name of the user. A new virtual header field named group is created. Before a new virtual header can be used, it is required to define all possible header values. Similar to monitoring requests, immediately after controller is running, list of all user group values is detected and used for definition of all possible values.

By using virtual headers, controller is appending VLAN tag to packets to determine virtual headers value. VLAN should be used only between switches and not on links to end devices. Unfortunatelly, Pyretic does not remove VLAN tags on links to end devices, therefore a modification of Pyretic runtime is used. After policies are combined and ready for compilation to for OpenFlow rules, VLAN tags on links to end devices are removed.

By deleting VLAN tag on links with end devices, new challange is created. In OpenFlow 1.0, it is not possible to send one packet to multiple output interface with different action list. Which means, switch is not able to send packet to link with end device and to link to another switch. Action lists differs in VLAN tags, where packet sent on link to another switch wants to remain VLAN tag and packet sent on link to end device requires removing VLAN tag.

The best solution is to use higher version of OpenFlow, where the concept of multiple OpenFlow tables solves this challage. Unfortunatelly Pyretic supports only version 1.0. Therefore it is necessary to have this limitation in mind during creation of applications for Pyretic. For example, application which forwards packets on switches by using broadcast is not allowed to use.

Another limitation of Pyretic is that Pyretic does not easily allow to detect what exactly changed during the topology change notification. Applications receive only object representing a new topology. If the application needs to know what exactly changed, it is necessary to compare old version of network topology with a new one. Easier and more general solution is to customize Pyretic runtime. Network topology representation is extended with list of topology change notifications. When the Pyretic receives new topology change from OpenFlow switch, type of change is appended to a list of last received events. Because the list of last received events is part of the network topology representation, applications are easily able to find out what happend in the network.

For easier transfer of states between multiple Pyretic applications, a new shared variable is used. The variable contains information about all connected end devices inside a network.

For each device, MAC address, IP address, location, username and user group name is stored. MAC address, IP address and location are detected by module which analyse ARP messages. Module which analyse ARP messages is also SIMS module which sends detected data to SIMS.

5.3 Identity interface

Interface is used for transfering monitoring requests, transfering initial identity database and receiving identity database changes. Transfering monitoring requests and initial identity database is triggered by starting the Pyretic controller. Identity database changes are triggered by messages from SIMS modules which generate HI2 messages for controller.

Pyretic controller does not have interface for processing external events. A new interface for processing events from SIMS is created. Interface is not connected directly to the SIMS but to the interlayer which solves several challanges as described in last sections. After receiving data from SIMS to controller it is necessary to process received data to network configuration.

At first, received data are saved into shared variable used for sharing data between multiple applications. When the user is authenticated, SIMS sends MAC address, IP address, location and username of the user to controller. Every Pyretic application that uses user identity inside its configuration uses this variable to detect user identity of connected devices. Secondly, controller enforce policy update of relevant applications.

5.4 Use cases

Rest of this chapter describes several use cases which aim to show benefits of managing network by using knowledge of user identites. Use cases are aimed on a midsize company and covers various areas of computer networks. Each use case works with groups of users instead of working with users individually. Is assumed that multiple users inside a company network share the same network policy. From the point of optimalization, it is easier and more clear to work with much smaller amount of policy rules. Even if some user requires specific policy just for him or her, it is possible to create a group only for one user.

5.4.1 Resource firewall

Resource firewall filters packets sent to a predefined resource. The term resource represents a network service which is defined with IP address, protocol type and protocol number. Firewall knows which resources can be used by which group of users. The functionality of the firewall is distributed among all network switches in network and ensure that packets that should be dropped are dropped before the packets reach the resource.

Example

The company has a general purpose server that runs two services:

- **HTTP** - port 80, the web contains a wiki page, every employee has access to this service
- **FTP** - port 21, repository includes personal data of the employees, only company's management has access to this service

Resource	**Parameters**	**Groups**
HTTP	192.168.1.1:TCP:80	*
FTP	192.168.1.1:TCP:21	management

Table 5.1: Configuration of example from Figure 5.2.

The situation is shown in Figure 5.2.

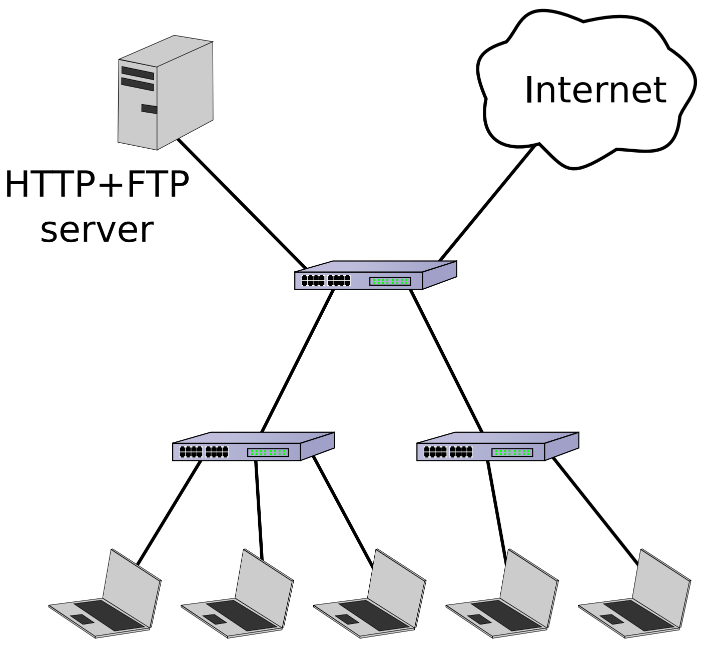


Figure 5.2: Figure shows server with multiple services deployed in the company network.

Employees are divided into a regular employees and management. Administrator wants to configure the network that both the regular employees and management have access to wiki pages of the company however only management has access to the FTP service. Configuration of use case is shown in table 5.1. Configuration consists of resource definitions and list of user groups which are allowed to access the resource. User group * represents any user group.

5.4.2 User groups firewall

Resource firewall filters packets sent between two groups of users. Every group of users has different security restrictions of communication with another user groups. Firewall is responsible for filtering packets between two groups which should not be able to communicate between themselves. The functionality of the firewall is distributed among all network switches in network.

Example

The company employees are divided into a regular employees and management. After connecting a new device to a network, identity of a device is unknown. Every unknown

Source user group	**management**	**regular employees**	**default**
management	x		
regular employees		x	
default			

Table 5.2: Configuration of example from figure 5.3.

device belongs to user group "default". The situation is shown in Figure 5.3.

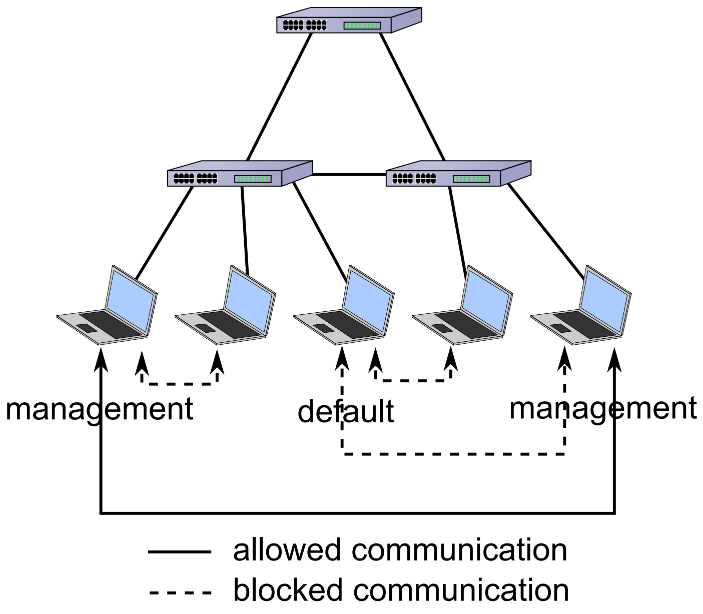


Figure 5.3: Figure shows network with three groups of users.

Employees from the group of regular employees or management should be able to communicate between themselves. Communication between users from different groups and communication between users from group default is not allowed. Configuration of use case is shown in table 5.2. Each row represents one source of a packet and column represents destination of the packet. Letter X represents that group of users from the row where letter X is placed is allowed to communicate with group of users from the column.

5.4.3 Routing

In addition to the destination IP address, SDN networks are able to route network traffic based on the source MAC address, source IP address, etc. Source based routing allows distribution of the network traffic to the same destination over multiple routes. With knowledge of user identities, network traffic can also be routed based on senders identity. This routing use case aims for limiting which users can use which links for forowarding packets.

Example

The company is connected to the Internet via two lines:

Link	**Parameters**	**Condition**
1-3	management	2-3
2-3	development	1-3
1-3	guest	never

Table 5.3: Configuration of example from figure 5.4.

- Optical cable
- Wi-Fi connection

The company wants to prefer optical cable to access the Internet for selected users. The policy of the company requires that the traffic from the management of the company is routed over the optical cable and other traffic routed over Wi-Fi connection. In the case of a link failure, all traffic will be routed to the other link. The example is shown in Figure 5.4.

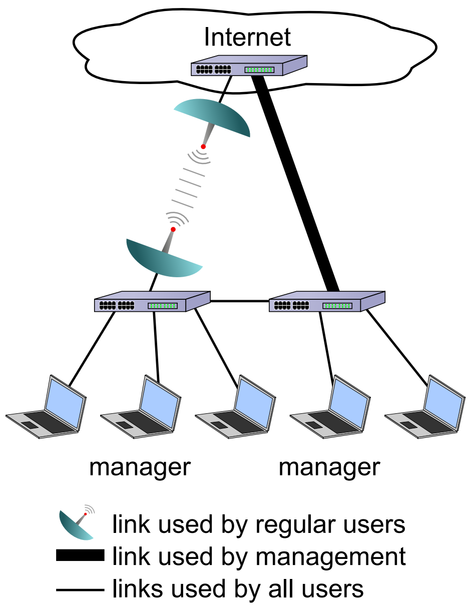


Figure 5.4: The example shows different routing policies. Management of the company uses routing policy with the optical connection to the Internet. Other users use routing policy with the Wi-Fi connection.

5.4.4 Accounting

The term accounting in this use case means the monitoring amount of data transmitted to a resource. Without knowledge of user identities, amount of data is counted per a device identifier, such as IP address. Use case aims for calculating amount of transfered data per user or per user group. With the knowledge of the user identities, it is possible to aggregate usage of multiple devices in the case that the user is using multiple devices.

Resource	**Parameters**	**Groups**
HTTP	192.168.1.1:TCP:80	regular employees

Table 5.4: Configuration of example from figure 5.5.

Example

The company wants to monitor amount of data transferred to HTTP server by regular employees. For every user which belongs to a monitored user group is created a unique counter. The example is shown in Figure 5.5.

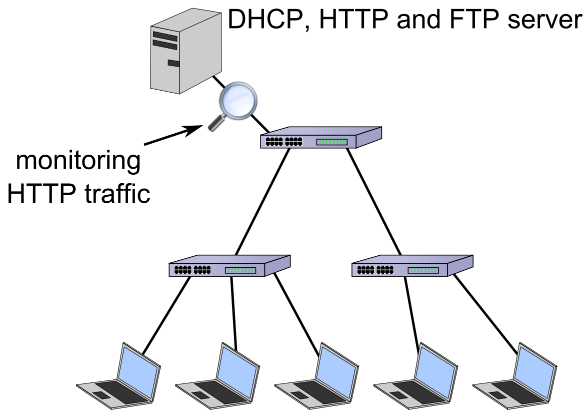


Figure 5.5: Network topology with a HTTP server witch is monitored.

Chapter 6

Implementation

This chapter describes implementation of parts required for using identity knowledge inside SDN network control.

6.1 Web Information System

This work uses HTTP server with company web information system (WIS) to gather identity of users. From the point of network management, identity of a network user is determined from username under which is the user authenticated. A simple WIS was implemented in PHP programming language. It is possible to authenticate on WIS with four usernames: alice, bob, charlie and guest (all lowercase letters). To log in, password is not required.

Source of WIS contains SIMS module for detection of new autheticated users. When a new user is authenticated, module sends message with an IP address and user login name to SIMS. If a user logs out from WIS, module send message about cancelling the link between IP address and WIS login name. After one user has logged out, a new user is able to authenticate on the same device.

6.2 Sec6Net Identity Management System

The SIMS does not require any modification of source code. Only required modification is added to the list of supported identifiers. List of supported identifiers on SIMS is extended with two new identifiers:

- **SDN location** - describes location of device inside a SDN network. Value of identifier consists of ID of switch (called datapath ID) and ID of network interface. Example of identifier - “00:00:00:00:00:01-3”.
- **WIS login** - user login name to web information system. Identifier is in form of a string. Example - “admin”.

Username	**Group**
alice	management
bob	development

Table 6.1: Example of configuration file users.csv for assigning groups values to users.

6.3 Controller

Virtual header definition

Before Pyretic controller starts to compile policies from applications, virtual header fields are defined. Definition of virtual field consists of name, list of all possible values and data type. My solution uses virtual field named “group” with values of data type “string”. List of all possible “group” values is obtained from configuration file `users.csv`. Table 6.1 shows example of file `users.csv`. List of unique group of users is gathered by processing config file line by line.

Removing virtual headers

When the packet with virtual header is forwarded towards end device, virtual header is not removed automatically. It is not guaranteed that end device processes packet with virtual header encoded in VLAN tag. Therefore, VLAN tags from packets forwarded to end devices are removed manually. Instead of modifying every Pyretic application to remove VLAN tags, runtime code of Pyretic is modified.

After the compilation of application policies, new procedure is executed. New procedure compares list of all ports to end users with action list of compiled fields. When a action list contains action for forwarding packet to port with end user, VLAN tag is removed. VLAN tag is removed by setting VLAN ID and VLAN PCP to 0. List of all ports to end users is obtained from network topology information stored inside the Pyretic controller.

Topology change list

Python class which represents network topology is extended with list of topology change events. When a change in topology is detected, description of the change is saved to topology change list. Three types of topology changes are saved:

- New switch is connected or connection to some switch is lost;
- New port is connected or some port got disconnected;
- Link to another switch is detected or lost.

Monitoring ARP messages

ARP packets from network are forwarded to controller for packet inspection. From source IP address and MAC address of ARP packet, controller knows location and addresses of end devices. MAC address, IP address and location is stored to global variable available to all applications. Variable is implented as dictionary, where key is the location and value is IP and MAC address. After ARP packets are inspected, packets are sent back to network.

If packet is ARP request, packet is sent to all end devices inside network. If packet is ARP reply, packet is sent to location of desination device, which location is found in global variable.

Application for ARP inspection is also a SIMS module which sends identifiers. When the application detects new device on a new port than MAC address, IP address and location is send to SIMS. In the case that newly detected device is already detected on a different location, message that cancel old location is sent to SIMS.

Source code 6.1 shows policy for forwarding packets to application. For every port to which it is possible to connect end device, rule is created. Rule match location of the port and ARP protocol. Action of rule is “identity”, which means packets are not modified and are send to next sequent policy. Each rule has next sequent policy packetBucket. Structure packetBucket receives packets from network.

```
    policy = None
    for i in range(1, len(portList)):
        policy = policy + match(ethtype=ARP_TYPE, switch=portList[i]["switch"
    ], inport=portList[i]["port"]) >> identity
    policy = policy >> packetBucket

```

Listing 6.1: Generating policy for forwarding ARP packets to application.

External events

Pyretic policy is updated by calling “update_policy” method. Pyretic runtime calls “update_policy” method of applications after controller receives packet from network or after topology changes. To enable update policy from external event, such as new user identity detection, it is necessary to add new way of calling “update_policy”. Solution is that application creates new thread, which is receiving external events. When an external event is received, thread calls “update_policy” which updates network policy.

Packet marking

Packets are marked with virtual header field “group”. When identity of a device user is not known, packets are marked with group “default”. Source code 6.2 shows generating policy for marking packets with virtual header group. In addition to detected devices which are marked with detected group value, unknown devices from each port are marked with default value.

```
    policy = modify(group="default")
    for device in port["devices"]:
        policy = if_(match(match(switch=port["switch"], inport=port["port"]),
     srcmac=device["mac"]), modify(group=device["group"]), policy)

```

Listing 6.2: Marking packets with virtual field group.

6.4 Interface between SIMS and Pyretic

Applications are not connecting to SIMS directly. Script called interface script is placed on the device where the SIMS is running. Interface script is responsible for creating new monitoring requests and for transmiting data from SIMS to controller.

Monitoring requests

Application that is responsible for communicating with SIMS and marking packet is called identityManagement. When identityManagement application connects to SIMS via interface script, application sends list of all users from `users.csv` file to SIMS. Interface script is responsible for filtering users on which monitoring request already exists. For rest of the users, interface script creates monitoring request message and send message to SIMS.

Identity database

After identityManagement application sent list of users to SIMS, application is waiting for actual identity database from SIMS. Interface script creates identity database from messages received from SIMS. Identity database consists of IP address, MAC address and location of detected users. Whenever new identityManagement application connects to interface script, interface script sends the whole list of users to identityManagement application. After transmitting the whole identity database, only new detected changes are sent to controller.

Receiving event from SIMS

When interface script sends information about new detected user, identifiers are saved to global variable which is accessible to all Pyretic applications. After received changes are reflecled in identityManagement application, update of policy is triggered. Every other application on controller which uses identity information for application functionality needs to be updated too. IdentityManagement application contains links to "update_policy" methods from other applications which are also triggered.

6.5 Use cases

Implementation of applications consist in using information of device users. Each application is capable of detect location of any device and user of that device.

6.5.1 Resource firewall

Resource firewall creates set of rules for blocking selected users access to the resource. Resource is a network service deployed inside a network with IP address, L4 protocol type and protocol number. Each resource has list of users which should be able to use the resource (network service). Application for each resource detects a list of users which are not allowed to access the resource. Every blocked user which is not allowed to communicated is located in network. Switch to which the blocked user is connected is found. On switch there is installed a rule that blocks traffic from blocked user to resource as shown in code 6.3.

```
 if_(match(switch=port["switch"],port=port["port"],group=device["group"],
dstip=resource["IP"],protocol=l4protocol,dstport=resource["port"]),ethtype
=IP_TYPE), drop, policy)

```

Listing 6.3: Generating rules for blocking unallowed users to communicate with resource.

6.5.2 User groups firewall

User groups firewall blocks traffic between two user groups. Application found all end device ports and determines which user groups are not able to communicate with that ports. On every switch with an end device port, set of rules is created to block incoming packets from blocked groups. Code 6.4 shows creation of blocking rules between user groups.

```
    policy = if_(match(switch=int(port["switch"]),outport=int(port["port"]),
   group=blockedGroup), drop, policy)

```

Listing 6.4: Generating rules for blocking unallowed communication from groups to port.

6.5.3 Routing

Application for routing calculates best path for every used IP address inside a network. Application uses Dijkstra algorithm [19] and cost of the link is defined as `1000000000 / Mbps`. Application uses knowledge of user identities for blocking selected links for selected users. Link that is not alowed to be used by group of users is marked with a high cost of link. For every device and switch inside a network, the shortest path between switch and device is calculated. Shortest path is executed on each user group separatelly. When the cost of path for some group is too high, path is considered as unaccessible. When the link is considered as accessible, the rule for user group, destination IP and switch which sets the output interface is created. Otherwise, blocking rule is created as shown in code 6.5.

```
    policy = if_(match(switch=switch,dstip=IPAddr(device["ip"]),ethtype=
   IP_TYPE,group=userGroup), fwd(outputPort), policy) #forwarding rule
    policy = if_(match(switch=switch,dstip=IPAddr(device["ip"]),ethtype=
   IP_TYPE,group=userGroup), drop, policy) #droping rule for unaccessible
   paths

```

Listing 6.5: Generating rules for forwarding droping packets.

6.5.4 Accounting

Application creates rules for each monitored resource and resource that should be monitored. For each resource and user group, two rules are created. One rule is matching incoming and another rule is matching outcoming traffic from service. Every rule sends packet to a bucket with resource name as parameter. Once per 10 seconds, controller is gathering statistics of generated rules. Statistics are saved to output CSV files. Code 6.6 shows generating rules for accounting.

```
    filter = match(switch=port["switch"],inport=port["port"],srcip=IPAddr(
   resourceParam["IP"]),protocol=l4protocol,srcport=resourceParam["port"],
   group=userGroup,ethtype=IP_TYPE)
    filter = filter + match(switch=port["switch"],outport=port["port"],dstip=
   IPAddr(resourceParam["IP"]),protocol=l4protocol,srcport=resourceParam["
   port"],group=userGroup,ethtype=IP_TYPE)
    policy + ( filter >> (bucket(resourceName) + identity) )

```

Listing 6.6: Generating rules for monitoring amount of traffic to resource.

Chapter 7

Testing use cases

Created use cases was tested on network created in school laboratory (C304). Network switches were simulated with software Open vSwitch. Open vSwitch is OpenFlow switch able to run on most of the linux based operating systems. This work uses Ubuntu 14.04 to run Open vSwitch.

Each use case was tested on the same network. Figure 7.1 shows testing topology. Testing topology consists of:

- **Users** - four users are connected to network;
- **User groups** - each new user is assigned to user group "default users". After user is authenticated, user belongs to management or developent user group;
- **Switches** - three Open vSwitch are deployed. Switch ID (and datapath ID) is manually configured to value from one to three;
- **Servers** - two servers with three different network services are deployed;

To change user group of an user, user has to authenticate on WIS. WIS allows users to authenticate by usernames: alice (management group), bob (development group) and charlie (development).

7.1 Resource firewall

Testing of resource firewall application aimed at droping packets for resources that are not allowed to reach. Configuration of application is shown in Table 7.1. It is necessary to allow every user to communicate with DHCP server and WIS server. Without access to DHCP server, client has to manually assign address. By blocking any group to communicate with WIS, users of that group will not be able to log out or change login name. Testing of application consisted of several steps:

- new device connects to network;
- which resources can the device use? DHCP server and WIS;
- person on device is authenticated on WIS as user alice (management group);
- which resources can the device use? all of them;

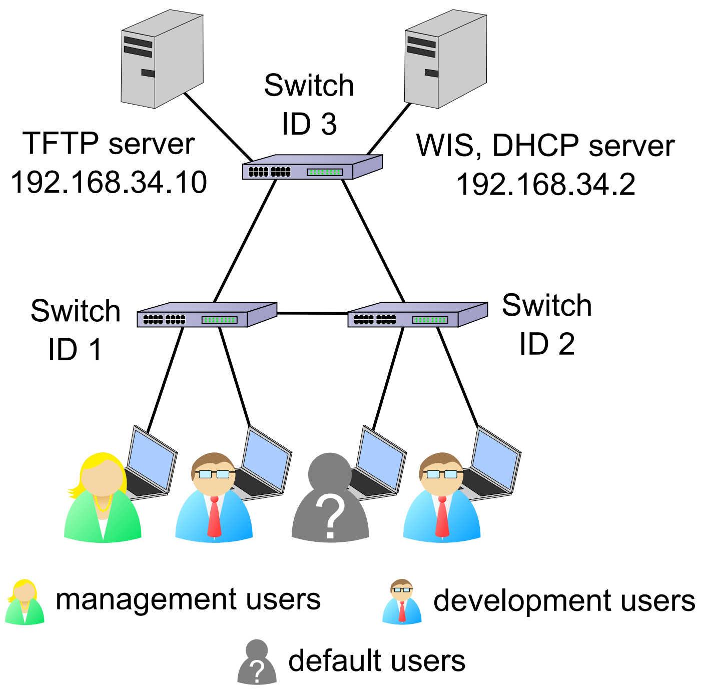


Figure 7.1: Testing topology for validating functionality of use cases. Two servers with three services are deployed to the network together with four users.

Resource	**Parameters**	**Groups**
TFTP	192.168.34.10:UDP:69	management, development
DHCP	192.168.34.2:UDP:67	*
WIS	192.168.34.2:TCP:80	*

Table 7.1: Configuration of resource firewall

- another device connects to network;
- person on second device is authenticated on WIS with same username alice (management group);
- which resources can both the devices use? all of them;
- user logs out from WIS on first connected device;
- which resources can the first device use? DHCP server and WIS.

Results measured from test are compared with expected results defined by configuration file. All the results match.

7.2 User groups firewall

Application blocks communication between users of selected groups. Configuration of application is shown in Table 7.2. It is not necessary that users from the same group are allowed

Source user group	**management**	**development**	**default**
management	x	x	
development	x	x	x
default		x	

Table 7.2: Configuration of group firewall

to communite between each other. By typing letter x in row with group name development and column management, traffic from group development to group management is allowed. Testing of application consisted of several steps:

- device n. 1 connects to network;
- device n. 2 connects to network;
- can device n. 1 (default) communicate with device n. 2(default)? No;
- person on device n. 1 is authenticated on WIS as user bob (development group);
- can device n. 1(development) communicate with device n. 2(default)? Yes;
- person on device n. 2 is authenticated on WIS as user alice (management group);
- can device n. 1(development) communicate with device n. 2(management)? Yes;
- user of device n. 1(development) logs out from WIS;
- can device n. 1(default) communicate with device n. 2(management)? No.

Results measured from test are compared with expected results defined by configuration file. All the results match.

7.3 Routing

Configuration of application is shown in Table 7.3. Testing of application consisted of several steps:

- packet capture runs on all switches;
- device n. 1 connects to network on switch 1;
- which path uses device n. 1(default) for communication with TFTP server? switch1 - switch3;
- person on device n. 1 is authenticated on WIS as user alice (management group);
- which path uses device n. 1(management) for communication with TFTP server? switch1 - switch2 - switch3;
- device n. 2 connects to network on switch 2;
- which path uses device n. 2(default) for communication with TFTP server? switch2 - switch1 - switch3;

Link	Parameters	Condition
1-3	management	never
2-3	development, default	never

Table 7.3: Configuration of routing

- person on device n. 2 is authenticated on WIS as user bob (development group);
- which path uses device n. 2(development) for communication with TFTP server? switch1 - switch2 - switch3;
- user on device n. 2 logs out from WIS and is again authenticated as user alice (management group);
- which path uses device n. 2(management) for communication with TFTP server? switch2 - switch3;

Results captured from test are compared with expected results defined by configuration file. Packets in direction from user device to TFTP server are transferred over correct links. However, traffic in opposite direction is not. Packets from TFTP server to device on switch 1 is always transferred by path switch3-switch1 and packets from TFTP server to device on switch 2 is always transferred by path switch3 - switch1 - switch2. The reason of this behavior is that routing decisions are made on identity of source devices. Because TFTP server is not authenticated on WIS, it belongs to group default.

7.4 Accounting

Application measures amount of data transfered to and from TFTP server by users of group development. Configuration of application is shown in Table 7.4. Testing of application consisted of several steps:

- how much traffic is transferred by development group to or from TFTP resource? 0 bytes;
- new device connects to network;
- user on device (default) updates a file to TFTP server;
- how much traffic is transferred by development group to or from TFTP resource? 0 bytes;
- person on device is authenticated on WIS as user alice (management group);
- user on device (management) updates a file to TFTP server;
- how much traffic is transferred by development group to or from TFTP resource? 0 bytes;
- user logs out from WIS and is again authenticated as user charlie (development group);
- user on device (development) updates a file to TFTP server;

Resource	**Parameters**	**Groups**
TFTP	192.168.34.10:UDP:69	development

Table 7.4: Configuration of accounting

- how much traffic is transferred by development group to or from TFTP resource? 72 bytes.

Results measured from test are compared with expected results defined by configuration file. Results with 0 bytes match and last result with 72 bytes do not. User uploads file with size about 4MB and only 72 bytes is measured. Analysis of communication between device and TFTP server shows that file is not transfered to UDP port 69 where the service is running. Only few packets are sent to UDP port 69 and rest of them use dynamic port numbers.

Chapter 8

Conclusion

The objective of this work is to extend possibilities of network control logic. My solution combines identity management system with control login of the network. Interface between the two systems was designed and implemented. Based on the information from identity management system, network control logic allows applications to work with identities of users.

Before designing the interconnection between systems, it was necessary to study a new architecture of the networks and user identities. Biggest different part of the new architecture is in separation of control plane from data plane. Two ways of programming SDN applications was described. My solution implements applications over controller called Pyretic. Second part of the theory was aimed at user identities, how to detect identity of users inside networks and identity management system (SIMS) was described.

Based on the theory, an interface SIMS and Pyretic controller is designed and implemented. Problem with SIMS is that architecture is designed for lawful interceptions. Created interface solves several SIMS limitations.

Several use cases for demonstrating benefits of created solution are designed. The applications use the knowledge of user identities and adjusts the configuration of computer networks by using the knowledge of the user identities. Use cases were implemented and tested in school laboratory.

Bibliography

[1] FEAMSTER, Nick. Software Defined Networking. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002

[2] APPENZELLER, Guido. Software Defined Networking: Interview with Guido Appenzeller. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/91

[3] CLARK, David. Software Defined Networking: Interview with David Clark. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/25

[4] KOPONEN, Teemu. Software Defined Networking: Interview with Teemu Koponen. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/35

[5] MCKEOWN, Nick. Software Defined Networking: Interview with Nick McKeown. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/79

[6] CASADO, Martin. Software Defined Networking: Interview with Martin Casado. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/15

[7] REXFORD, Jennifer. Software Defined Networking: Interview with Jennifer Rexford. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/63

[8] NADEAU, Thomas D and GRAY, Ken. *SDN: software defined networks.* Cambridge: O'Reilly, 2013, xxvii, 352 s. : grafy. ISBN 9781449342302.

[9] PEPELNJAK, Ivan. Why is RESTful API better than SNMP? ÂŤ ipSpace.net by @ioshints *Blog.ipspace.net* [online]. [cit. 2015-05-23]. Available from: http://blog.ipspace.net/2012/08/why-is-restful-api-better-than-snmp.html

[10] PEPELNJAK, Ivan. Why is RESTful API better than SNMP? ÂŤ ipSpace.net by @ioshints *Blog.ipspace.net* [online]. [cit. 2015-05-23]. Available from: http://blog.ipspace.net/2012/06/netconf-expect-on-steroids.html

[11] APPENZELLER, Guido. Software Defined Networking: Interview with Guido Appenzeller. *Coursera* [online]. [cit. 2015-01-01]. Available from: https://class.coursera.org/sdn-002/lecture/91

[12] MCKEOWN, Nick. Software-defined networking. *INFOCOM keynote talk* (2009).

[13] MCKEOWN, Nick, ANDERSON Tom, BALAKRISHNAN Hari, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review.* ACM, 2008, vol. 38, issue 2, s. 69-74.

[14] PFITZMANN, Andreas, and HANSEN Marit. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. (2010): 34.

[15] MARTÍNEK, Tomáš, KRAMOLIŠ Petr, HOLKOVIČ Martin and POLČÁK Libor. Dynamická identifikace uživatelú v prostředí sítí IPv4 a IPv6. FIT-TR-2012-006, Brno: Fakulta informačních technologií VUT v Brně, 2012.

[16] POLČÁK Libor, MARTÍNEK Tomáš, HRANICKÝ Radek, et al. Zákonné odposlechy v moderních sítích. FIT-TR-2014-007, Brno: Fakulta informačních technologií VUT v Brně, 2012.

[17] POLČÁK Libor, HRANICKÝ Radek, and MARTÍNEK Tomáš. On Identities in Modern Networks. *Journal of Digital Forensics, Security and Law 9.2* (2014): 9-22.

[18] European Telecommunications Standards Institute: *ETSI TR 102 528: Lawful Interception (LI); Interception domain Architecture for IP networks.* 10 2006, version 1.1.1.

[19] SKIENA, Steven. Dijkstra's Algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley* (1990): 225-227.

[20] Jazyková poradna ÚJČ AV ČR, v. v. i. Česko *Internetová jazyková příručka* [online]. [cit. 2015-05-24]. Available from: http://prirucka.ujc.cas.cz/?id=725

[21] SHERWOOD, Rob, et al. Can the production network be the testbed?. *OSDI.* Vol. 10. 2010.

[22] KIM, Hyojoon and FEAMSTER Nick. Improving network management with software defined networking. *Communications Magazine, IEEE.* USA: IEEE, 2013, vol. 51, issue 2, s. 114-119.

[23] KIM, Hyojoon, et al. The evolution of network configuration: a tale of two campuses. *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference.* ACM, 2011.

[24] OpenFlow Switch Consortium. OpenFlow Switch Specification Version 1.0.0. (2009).