Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Ústav aplikované informatiky

Centralizovaná správa aktivních prvků s využitím Software Defined Networking (SDN) a technologie OpenFlow, včetně zobrazení grafické topologie

Diplomová práce

Bc. Jan Tůma

Vedoucí práce: Ing. Rudolf Vohnout PhD.

České Budějovice 2016

Bibliografické údaje

Tůma, J., 2016: Centralizovaná správa aktivních prvků s využitím Software Defined Networking (SDN) a technologie OpenFlow, včetně zobrazení grafické topologie. [Centralized management of active devices using Software Defined Networking and OpenFlow technology, also with a topology graph. Mgr. Thesis, in Czech.] – 64 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

NÁZEV:

Centralizovaná správa aktivních prvků s využitím Software Defined Networking (SDN) a technologie OpenFlow, včetně zobrazení grafické topologie.

ABSTRAKT:

Náplní této diplomové práce je vytvoření nadstavbové webové grafické aplikace, využívající Software Defined Networking Controller a protokol OpenFlow pro základní management a vykreslení topologie aktivních prvků, popis hlavních aspektů technologie OpenFlow a optimalizace vybraného SDN Controlleru pro plnou podporu aktivních prvků v multi-vendor prostředí.

KLÍČOVÁ SLOVA:

Software Defined Networking (SDN), OpenFlow, Centralizovaná správa, Topologie, GUI, Kontrolér

TITLE:

Centralized management of active devices using Software Defined Networking and the OpenFlow technology, also with a topology graph.

SUMMARY:

The aim of this master thesis is to create a web-based graphical user interface which communicates with a Software Defined Controller and the OpenFlow protocol to provide a basic network management and a topology graph, then to describe main aspects of the OpenFlow technology and to optimalize the chosen SDN Controller for full support in a multi-vendor enviroment.

KEYWORDS:

Software Defined Networking (SDN), OpenFlow, Central management, Topology, GUI, Controller

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 1. 12. 2016 Bc. Jan Tůma

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce panu Ing. Rudolfu Vohnoutovi PhD. za cenné rady při tvorbě této práce a zapůjčení aktivního prvku pro fyzické testování. Dále bych rád poděkoval své rodině a přátelům za podporu.

Obsah

1. Úvod

V průběhu času se pravděpodobně většině správců sítě stane, že síť začne být relativně nepřehledným shlukem přepínačů a směrovačů, kdy každý tento jednotlivý aktivní prvek je nakonfigurován samostatně. Zajištění veškerého nastavení tak, aby spolu všechna zařízení mohla komunikovat, se tak může stát poměrně velkým problémem, vzhledem k tomu, že každý aktivní prvek je samostatná jednotka konfigurovaná izolovaně od ostatních a nemající o ostatních prvcích v síti v podstatě žádné informace.

Tímto narůstají nároky na samotného správce, který musí mít o síti dokonalý přehled, aby věděl co a kde má nastavit, aby dosáhl kýženého výsledku. Může tak docházet ke skrytým chybám v konfiguraci jednotlivých prvků a tím např. i k poteciálně bezpečnostním problémům v nastavení, které mohou vyústit v její úspěšné napadení útočníkem. Navíc se pak správa sítě stává poměrně závislou na konkrétním člověku, což může vyvolat problémy různého typu – od jeho nepřítomnosti až po odchod, kdy se pak síť stává v podstatě bezprizorní.

Přirozeně tak vyvstala otázka, jak lze takovýmto situacím předcházet, jak změnit pro velké sítě jejich nedostačující a náročnou správu. V současné době tak dochází k velkému rozmachu tzv. Softwarově definovaných sítí SDN[1], které od základu mění koncept správy sítě. Namísto izolovaných jednotek je tak síť řízena centrálním kontrolérem, který ví o všech připojených zařízeních a nastavuje sám aktivní prvky podle zadaných parametrů. Odpadá tak nutnost uchovávání plánků sítě a konfigurací jednotlivých aktivních prvků, o vše se stará kontrolér, který ví o veškerých nastaveních konkrétních prvků. Spojení tak probíhá poze ve směrech správce - kontroler = aktivní prveky, nikoli jako je tomu u klasických sítí správce - aktivní prvky.

Pokud má tedy být v síti provedena konfigurační změna, je tato zadána kontroléru, který se již sám postará o provedení změn na samotných aktivních prvcích. Jedná se tedy o centralizaci řízení sítě, což odbourává veškeré nastíněné problémy. Z výše popsaných důvodů se tímto způsobem řízení zabývá již mnoho společností, vyrábějících aktivní prvky a které se tak aktivně podílejí na nových standardech. Jde např. o Cisco, BigSwitch Networks, HP, Dell, NEC, Juniper a další.

[1] viz. Seznam použitých zkratek – SDN

V této práci je nejprve nastíněn konkrétní řešený problém a jsou přiblíženy využité technologie, případně jejich alternativy. Dále je popsáno navržené řešení a samotná implementace, včetně dostupných funkcionalit, které byly implementovány. Výstupem práce je nadstavbová webová grafická aplikace umožňující pohodlné ovládání vybraného SDN kontroléru a tím celé jím řízené sítě, včetně přehledného grafického zobrazení topologie spravované sítě. Dále pak návod na zprovoznění kontroléru s implementovaným grafickým rozhraním, jeho instalaci.

1.1 Definice problému

Hlavním podnětem ke vzniku této diplomové práce je skutečnost, kdy většina volně dostupných (Open Source) SDN kontrolérů neobsahuje žádné, nebo jen velmi strohé grafické uživatelské rozhraní. Tyto kontroléry je tak nutné ovládat jen pomocí REST[2] rozhraní, což je nejen velmi nepohodlné, ale i náchylné k chybám, a jedná se o relativně pomalý způsob obsluhy, pokud má tato být prováděna ručně. Vyvstává tak potřeba zjednodušit a zrychlit manipulaci s kontrolérem a zajistit zpětnou vazbu pro uživatele, k čemuž se nejlépe hodí grafické rozhraní, které na rozdíl od příkazové řádky nevyžaduje nutnost znalosti příkazů, jejich syntaxi, strukturu zpráv zasílaných kontroléru atp. Další výhodou grafického rozhraní je možnost přehledně zobrazit topologii sítě, aby na ní měl uživatel ucelený pohled jako na celek.

1.2 Cíle

1.2.1 Hlavní cíl

Hlavním cílem této práce je vytvoření nadstavbové webové grafické aplikace umožňující základní ovládání SDN kontroléru a vykreslení topologie dané, kontrolérem spravované sítě. S touto aplikací bude možné vzdáleně plně ovládat vybraný kontrolér, provádět konfigurační změny v síti a zobrazovat topologii připojené sítě, se kterou kontrolér pracuje a umožnit tak ucelený pohled na ni. Důvodem je stav, kdy většina zdarma dostupných kontrolérů neposkytuje grafické rozhraní, umožňující vzdálené ovládání kontroléru a připojené sítě, kterou spravuje.

[2]viz. Seznam použitých zkratek – REST

1.2.2 Vedlejší cíle

Vedlejšími cíli této práce je stručný popis technologie OpenFlow, pomocí které je zajištěna komunikace mezi kontrolérem a připojenými aktivními prvky, především jejími základními aspekty, a to z důvodu, že se jedná o široce využívaný protokol pro tento typ komunikace. Dále případná optimalizace vybraného SDN kontroléru pro bezproblémovou práci v multi-vendor prostředí, kdy spolupráce mezi aktivními prvky různých výrobců a kontrolérem nemusí být naprosto bezproblémová a kontrolér může vyžadovat provedení určitých optimalizačních nastavení, např. při prvotním připojení aktivního prvku ke kontroléru, kdy dochází k vyjednání verze protokolu, která bude následně použita pro komunikaci.

2. Metodika

Při přípravě této diplomové práce bylo nejprve nutné seznámit se s pojmem SDN obecně, jaký je vlastně koncept a v čem se odlišuje od stávajících postupů, dále s technologií OpenFlow, která je ve většině případů využita pro komunikaci mezi kontrolérem a aktivními prvky připojené sítě a dostupnými SDN kontroléry, které tuto síť řídí, konfigurují a sbírají o ní informace. Následně provést na základě multikriteriální analýzy porovnání těchto kontrolérů a vybrat na základě zvolených hodnotících kritérií nejvhodnější kontrolér pro tuto práci. Hodnotící kritéria zohledňují jak funkčnost daných kontrolérů, tak jejich vhodnost pro tuto práci.

Při navrhování grafického rozhraní, pomocí kterého je kontrolér ovládán a probíhá pomocí něj veškerá konfigurace sítě, byly zohledněny dostupné funkce vybraného vítězného kontroléru. Základní snahou bylo, aby byla naprostá většina dostupných funkcí ovladatelná z tohoto rozhraní, bez nutnosti zasahovat do chodu kontroléru z příkazové řádky nebo pomocí konfiguračních souborů. Správa kontroléru je tak kompletně možná za pomoci tohoto GUI[3].

Vzhledem ke zvolenému kontroléru probíhalo psaní backendové části v jazyce Java verze 7, na němž je kontrolér postaven. Pro frontendovou část byla vybrána technologie HTML 5 a Javascript (ECMAScript 5) za použití frameworku Angular.js 1, a to z důvodu nezávislosti GUI na koncovém zařízení uživatele, kdy k ovládání stačí internetový prohlížeč a není nutná instalace žádných dalších částí ani doplňků do prohlížeče, což by bylo nutné např. při použití technologie Adobe Flash, který navíc nemá ani tak širokou základnu podporovaných zařízení, kdy operační systém Android tuto technologii nepodporuje.

[3]viz. Seznam použitých zkratek – GUI

3. Přehled současného stavu

V současné době je většina Open Source kontrolérů vzdáleně ovládána za pomoci příkazové řádky, resp. pomocí REST rozhraní. Jedná se o relativně nepohodlný a pomalý způsob, který je sice velmi jednoduchý a snadno modifikovatelný, nicméně se nelze obejít bez dokumentace k danému kontroléru, ve kterém jsou uvedeny postupy pro práci s kontrolérem, formáty zpráv zasílané kontroléru a další potřebné náležitosti, např. HTTP metody. Zároveň je nutné veškeré konfigurační objekty psát ručně.

Jeho ovládání je tak velmi nepohodlné, náchylné k chybám, pracné a časové náročné. Možnost ovládat kontrolér pomocí GUI je tak velmi užitečná, a tak většina kontrolérů obsahuje i základní GUI, nicméně to je ve většině případů nepoužitelné pro jeho správu a umožňuje jen základní přehled o jeho stavu, popř. základní vykreslení topologie připojené sítě, bez jakýchkoli dalších podrobností a možnosti úprav.

Naopak komerční kontroléry jsou povětšinou již velmi propracované ve všech ohledech a jejich ovládání pomocí GUI je naprosto běžné, jedná se o standard. Mezi nejznámější patří např. HP VAN[4] SDN kontrolér, Cisco APIC[5] a VMware NSX Controller. Nicméně přirozenou nevýhodou takovýchto komerčních řešení je jejich cena, a to nejen pro komerční použití, ale i pro nekomerční, kdy většina umožňuje bezplatné používání jen po omezený časový úsek, určený na vyzkoušení kontroléru a jeho vlastností.

3.1 Definice problému

Základním problémem většiny bezplatně (Open Source) dostupných kontrolérů je tedy nedostatečné grafické rozhraní, umožňující jejich přehledné a intuitivní ovládání. Zároveň by toto GUI mělo poskytovat informace o připojných zařízeních a vykreslovat přehlednou topologii připojené sítě, která jinak může být obtížně zjistitelná. Přitom kontrolér má o všech potřebných vazbách přehled a vykreslení topologie je tak jen otázkou získání těchto vazeb z kontroléru a jejich vhodného

[4]viz. Seznam použitých zkratek – VAN
[5]viz. Seznam použitých zkratek – APIC

převedení do grafické podoby, která je, narozdíl od nepřehledné změti hodnot a parametrů uvedených v tabulkách, snadno srozumitelná i pro méně pokročilého uživatele.

Skutečně plně použitelná GUI, umožňující kompletní správu kontroléru, se začínají objevovat až nyní, a to bohužel jen u některých kontrolérů, které začaly být masivně podporovány velkými síťovými výrobci (ONOS, OpenDayLight). Přitom jiné kontroléry jsou minimálně stejně kvalitní, co se týče práce s připojenými prvky, či snadnosti jejich instalace a používání.

Tato práce si tak klade za cíl vytvořit plnohodnotné grafické rozhraní pro ovládání vhodného vybraného kontroléru, který toto nenabízí, ale jinak má všechny potřebné kvality a toto ho tak může v určitých ohledech znevýhodňovat, i když umí např. lépe pracovat s připojenou sítí a získávat z ní více informací. GUI by mělo být schopno nejen vykreslovat topologii připojené sítě, ale umožnit její kompletní správu včetně nastavení kontroléru a dodat tak kontroléru potřebnou uživatelskou přívětivost a snadnou využitelnost dostupných funkcí, které nabízejí a které by jinak byly obtížně využitelné, např. z důvodu složitosti konfigurace, či použití.

4. Výběr SDN kontroléru

Při výběru vhodného SDN kontroléru pro tuto práci bylo zohledněno několik kritérií, důležitých jak v rámci využití kontroléru jako takového, tak pro tuto práci, za které každý kontrolér mohl získat 1 až 5 bodů. Výsledný součet byl pak získán vynásobením počtu získaných bodů v konkrétní části vahou příslušného kritéria, která je uvedena v závorce za ním v jejich přehledu níže. Čím více bodů kontrolér získal, tím lépe.

Kritéria pro výběr kontroléru:

1. **Bezplatné použití jak v nekomerční, tak komerční sféře (5)** - jedná se o velmi důležité kritérium, z toho důvodu, aby mohl být kontrolér bez jakéhokoli zpoplatnění nasazen i v komerční sféře. Kontrolér mohl získat 0 - 1 bod.

2. **Rozšiřitelnost (3)** – možnost kontrolér rozšířit o další moduly a implementovat tak další funkčnost. Funkce kontroléru tak nejsou omezeny jen jejich tvůrci. Kontrolér získal bod za každé využitelné rozhraní.

3. **Vzdálené ovládání pomocí REST rozhraní (4)** – komunikace s tzv. North-bound rozhraním kontroléru za pomoci RESTu. Jedná se o jednoduché ovládání kontroléru přes protokol HTTP a jeho metod, kdy není nutné zasahovat přímo do zdrojového kódu a lze tak snadno vytvářet nadstavbové aplikace např. v internetovém prohlížeči, který je schopen pomocí Javascriptu s tímto rozhraním komunikovat. Rozhraní je tak nezávislé na jazyce, ve kterém je psán kontrolér a není tak nutné komunikovat s kontrolérem přes rozhraní daného jazyka. Kontrolér mohl získat 0 - 1 bod, pokud je rozhraní přítomno.

4. **Podporované verze OpenFlow protokolu (5)** – významné kritérium, zohledňující s jakou škálou funkcionalit a aktivních prvků je schopen kontrolér pracovat. Kontrolér získal bod za každou podporovanou verzi protokolu.

5. **Dostupnost dokumentace (4)** – jak dobře je popsán zdrojový kód kontroléru, především jeho rozhraní, určená pro komunikaci s ním a množství

dostupných návodů na jeho ovládání, popř. jejich aktuálnost, která je rovněž velmi podstatná. Kontrolér mohl získat bod za dokumentaci, pokud je dostupná a jak je velké množství návodů již od samotných tvůrců (maximálně 3 body).

6. **Pokračující vývoj (2)** – zda je kontrolér nadále vyvíjen a nehrozí tak akutně jeho zastarání. Kontrolér získal bod, pokud v aktuálním roce (2016) vyšla nová verze kontroléru.

7. **Jazyk, ve kterém je kontrolér napsán (2)** – z důvodu snadné čitelnosti a případné upravitelnosti zdrojového kódu kontroléru. Kontrolér dostal 2 body za Javu, ostatní jazyky po 1 bodu.

8. **Ovládání pomocí GUI (5)** – taktéž velmi důležité kritérium, rozhodující o snadnosti ovládání kontroléru a zároveň o jeho vhodnosti pro tuto práci, která má za cíl toto GUI vytvořit. Kontrolér nezískal žádný bod, pokud již GUI v základu má a není tak vhodným kandidátem a naopak 1 bod, pokud ovládání pomocí GUI není v základu možné nebo je jen velmi omezené.

Do porovnání byly zařazeny tyto kontroléry, které lze považovat za nejdůležitější: [6] [7]

- **OpenDaylight** (`https://www.opendaylight.org`)
- **OpenContrail** (`http://www.opencontrail.org`)
- **Floodlight** (`http://www.projectfloodlight.org`)
- **Ryu** (`https://osrg.github.io/ryu`)
- **ONOS** (`http://onosproject.org`)

[6] Five must-know open source SDN controllers. *TechTarget* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `http://searchsdn.techtarget.com/news/2240225732/Five-must-know-open-source-SDN-controllers/`.

[7] SDN Controller Comparison Part 2: Open Source SDN Controllers. *SDxCentral* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/`.

Na základě vyhodnocení multikriteriální analýzy z tabulky 4.1 byl vybrán kontrolér Floodlight (logo obrázek 4.1), který je vyvíjen za přispění firmy BigSwitch Networks[8] a je tak pravděpodobně zajištěn jeho kontinuální budoucí vývoj. Tento kontrolér splňuje všechna jmenovaná kritéria a získal nejvyšší počet bodů v rámci analýzy. Kontrolér má v základu jednoduché GUI (obrázek 4.2), které však vykreslí jen jednoduchou topologii a zobrazí naprosto základní informace, ale neumožňuje žádným způsobem ovlivňovat připojenou síť ani jeho chod. Je tak vhodným kandidátem pro tuto práci, kdy vytvoření GUI velmi zkvalitní práci s jeho rozhraním a umožní tak jeho pohodlné, přehledné a rychlé ovládání.

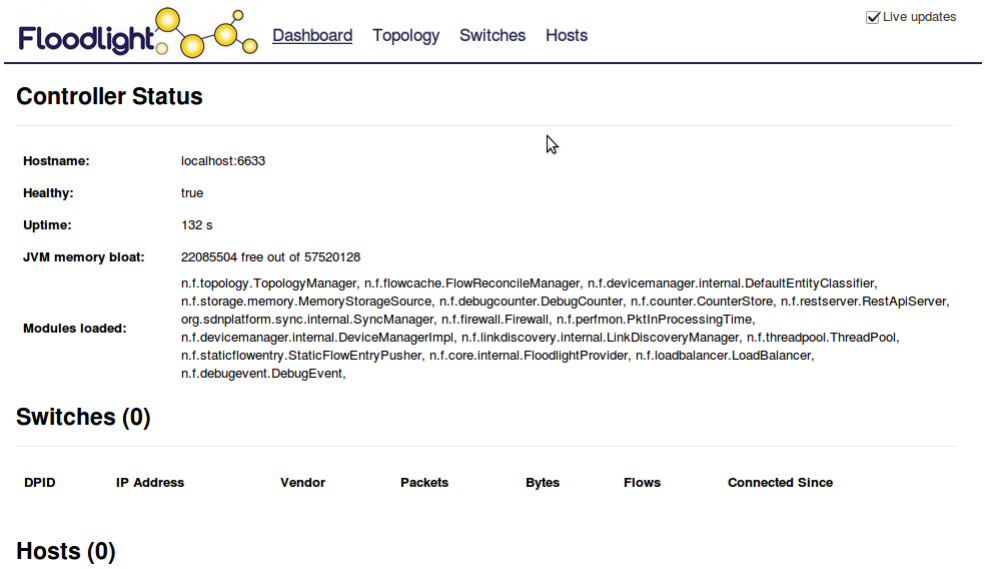
Kontrolér	**č. 1**	**č. 2**	**č. 3**	**č. 4**	**č. 5**	**č. 6**	**č. 7**	**č. 8**	**Výsledek**
OpenDaylight	5x1	3x2	4x1	5x5	4x3	2x1	2x2	5x0	58
OpenContrail	5x1	3x1	4x1	5x5	4x2	2x1	2x1	5x0	49
Floodlight	5x1	3x2	4x1	5x5	4x3	2x1	2x2	5x1	63
Ryu	5x1	3x2	4x1	5x5	4x3	2x1	2x1	5x1	61
ONOS	5x1	3x2	4x1	5x5	4x4	2x1	2x2	5x0	62

Tabulka 4.1: Porovnávací tabulka kontrolérů

Obrázek 4.1: Floodlight logo.[9]

[8]What is a Floodlight Controller? *SDxCentral* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/`.

[9]Floodlight logo [obrázek]. *It's that time of year again* [online]. 2012 [cit. 2016-09-10]. Dostupné z: `http://www.bigswitch.com/blog/2012/12/13/its-that-time-of-year-again`.



Obrázek 4.2: Výchozí Floodlight GUI.

Pro tento kontrolér již sice existují grafické aplikace, které by měly umožňovat jeho ovládání, resp. ovládání připojené sítě, nicméně tyto jsou téměř nepoužitelné, protože jsou určeny buď pro starší verze kontroléru před verzí 1, kdy se zásadním způsobem měnilo REST rozhraní a tudíž nejsou schopné s ním nadále komunikovat, nebo jsou nespustitelné podle návodů, zveřejněných jejich tvůrci.

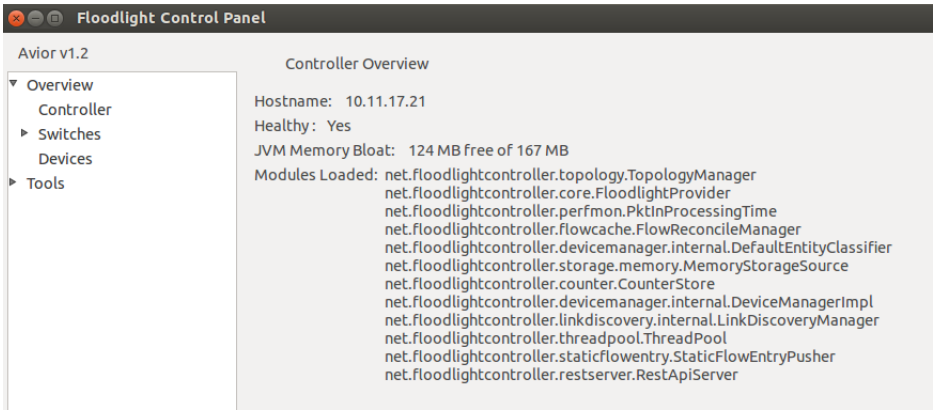
Jedná se o aplikace Marist[10] (obrázek 4.3) a Avior 2.0[11] (obrázek 4.4). Aplikace Marist by měla být univerzální pro jakýkoli SDN kontrolér a Avior by měl plně podporovat kontrolér Floodlight a OpenDaylight. Oba projekty se navíc bohužel zdají být neudržované a pro Floodlight verze 1.2 jsou nepoužitelné z důvodu napojení na staré rozhraní, nebo nefunkční instalace.

Marist je navíc jen desktopovou aplikací a žádné z těchto GUI neimplementuje přihlašování uživatelů a jedná se tak jen o pomocné aplikace, které však nejsou plně integrovány s kontrolérem a snaží se být univerzální pro více kontrolérů. Nicméně z tohoto důvodu jim chybí některé specifické funkce daného kontroléru, hlavně co se týče správy jeho samotného, kdy toto při své univerzálnosti ani nemo-

[10] OpenFlow at Marist. *Marist SDN Lab* [online]. 2013 [cit. 2016-08-10]. Dostupné z: `http://openflow.marist.edu/avior`.

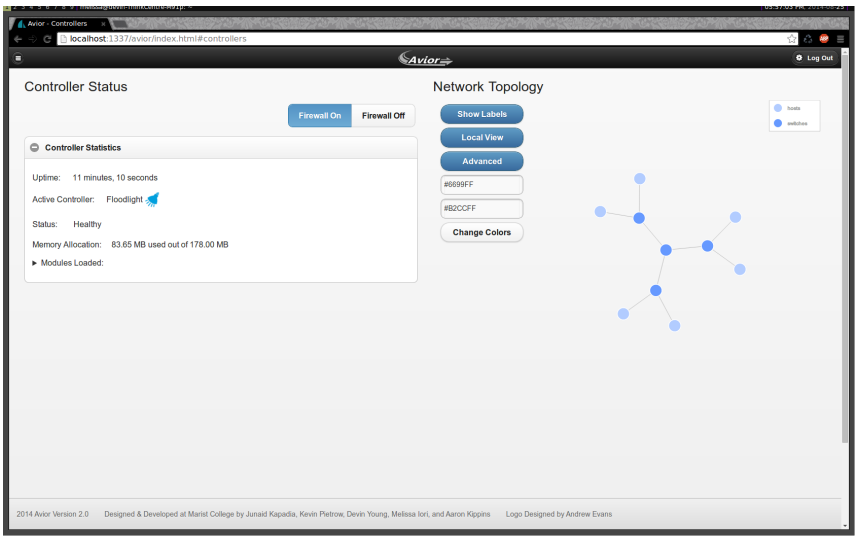
[11] 1PhoenixM/avior-service. *GitHub* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `https://github.com/1PhoenixM/avior-service`.

[12] Controller Overview [obrázek]. *OpenFlow at Marist, Gwtting started* [online]. 2013 [cit. 2016-09-10]. Dostupné z: `http://openflow.marist.edu/avior/gettingstarted`.



Obrázek 4.3: Ukázka Marist GUI.[12]

hou nabídnout z důvodu, že každý kontrolér má jiný typ konfigurace a obsáhnout tyto odlišnosti pro všchny by bylo velmi náročné a obtížně udržitelné. Navíc by aplikace musely být více integrovány s daným kontrolérem a tato omezení jsou tak daní za jejich univerzálnost.



Obrázek 4.4: Ukázka Avior GUI.[13]

Další nevýhodou aplikace Marist je, vzhledem k její podobě, nepřenositelnost mezi operačními systémy a nemožnost spustit ji např. na mobilním zařízení.

[13]Controller Status [obrázek]. *Avior 2.0* [online]. 2014 [cit. 2016-09-10]. Dostupné z: `https://raw.githubusercontent.com/1PhoenixM/avior-service/master/assets/images/Avior.png`.

Správce sítě tak přichází o možnost připojit se ke kontroléru v terénu a je odkázán na počítač s vhodným běhovým prostředím. Tuto nevýhodu částečně odbourává Avior, nicméně i ten je v konečném důsledku nepoužitelný a to kvůli chybějící autentizaci vůči kontroléru.

Nedá se tedy bohužel říci, že by tyto aplikace byly vhodné pro produkční práci s kontrolérem a zde vyvstává možnost plně integrovaného GUI, které běží přímo na daném kontroléru, je jeho součástí, stejně jako jednoduché GUI, dodávané s kontrolérem, avšak plně jej ovládá a obsahuje autentizaci uživatele, díky které je možné kontrolér zabezpečit před neoprávněným přístupem a manipulací s jeho prostředky.

5. Přehled využitých technologií

Tato kapitola shrnuje veškeré použité technologie, programy a knihovny použité při psaní této diplomové práce a rozebírá obecné koncepty, případně uvádí jejich možné alternativy.

Dále rozebrané technologie a programy:

- **SDN**
- **OpenFlow**
- **Mininet**
- **Floodlight kontrolér**

Stručný výčet programů a knihoven:

- **Java Service Wrapper Community Edition 3.5.27** – Servisní služba, umožňující spouštění aplikací psaných v jazyce Java (.jar archiv) jako služby na operačním systému. Stará se o spuštění aplikace po jeho startu, obnovení při pádu i logování do souborů. Pomocí konfiguračních souborů lze nastavit nepřeberné možnosti, včetně parametrů při spouštění, parametrů virtuálního stroje atp. Je využit pro spouštění kontroléru, kdy tento to sám o sobě neumí a nebylo by tak možné jeho pohodlné ovládání pomocí GUI a to především kvůli potřebě jeho restartu při změně konfigurace. Drobnou nevýhodou tohoto softwaru je, že je zdarma jen pro nekomerční použití, nicméně jeho schopnostmi to plně vynahrazuje. Dostupný z: `https://wrapper.tanukisoftware.com`.
- **Fabric.js 1.6.2** – Javascriptová knihovna, použitá pro vykreslování grafu topologie prvků a obrázků s přepínači, jeho porty, ve správě aktivních prvků. Umožňuje pohodlné kreslení na canvas HTML objekt a to přímo z Javascriptového kódu, navíc umožňuje i přesun objektů a zachytávání jejich událostí. Je zdarma dostupná na `http://fabricjs.com`.
- **AngularJS 1.5.0** – Javascriptový frontendový framework, použitý pro psaní frontendové části. Jedná se o komplexní framework, pomocí kterého je

řízena celá aplikace. Jeho hlavní výhodou je oboustranný data binding, kdy změny hodnot proměnných, provedené v kódu, jsou okamžitě přenášeny do grafických prvků, které vidí uživatel, i opačným způsobem. Velmi se tak zjednodušuje práce s uživatelskými vstupy, ale i se zobrazováním dat z REST rozhraní, kdy odpadá nutnost jeho aktualizace a tím aktualizace dat na více místech. Je zdarma dostupný na `https://angularjs.org`.

- **jBcrypt 0.3m** – Java knihovna, použitá pro šifrování hesel uživatelů. Používá BlowFish algoritmus a sůl je již přímo součástí vygenerovaného hashe, není tedy nutné ji ukládat zvlášť. Jedná se o oblíbený způsob práce s hesly a je ve výchozím nastavení použit např. i u Spring frameworku. Knihovna je zdarma dostupná na `https://search.maven.org/#artifactdetails%7Corg.mindrot%7Cjbcrypt%7C0.3m%7Cjar`.

- **Miniedit 2.2.0.1** – Grafický program, napsaný v Pythonu, určený na jednoduchou tvorbu a správu virtuálních sítí, postavených pro virtuální prostředí Mininet. Jedná se přímo o součást Mininetu a je případně zdarma dostupný na: `https://github.com/mininet/mininet/blob/master/examples/miniedit.py`.

- **Maven 4.0.0** – Jedná se o komplexní buildovací nástroj, používaný převážně pro Java aplikace, umožňující komplexní sestavení aplikací i obsahující balíčkovací systém, pomocí kterého jsou do projektu jednoduše začleňovány externí knihovny, které tak není nutné jednotlivě stahovat a přidávat do projektu. Je spravován Apachem viz. `https://maven.apache.org`.

- **IntelliJ IDEA 15.0.6** – Kompletní vývojové prosředí, které zvládá nejen jazyk Java, ale i HTML a Javascript včetně frameworků. Jeho použití tak velmi usnadňuje práci při vytváření jak backendové, tak frontendové části. Jedná se o placený produkt, nicméně je možné získat zdarma studentskou licenci pro nekomerční použití. Dostupný na: `https://www.jetbrains.com/idea`.

5.1 SDN

Jedná se o nový koncept centralizovaného řízení sítě za pomoci centrálního kontroléru. Odpadá tak konfigurování každého jednotlivého aktivního prvku v síti samostatně a izolovaně od ostatních. Ta je na místo toho řízena tímto kontrolérem. Není tak nutné při změně konfigurovat odděleně několik prvků, namísto toho se změna provede centrálně na kontroléru, který se již sám postará o provedení změn na konkrétních prvcích. Na jednom místě tak lze mít přehled o kompletní konfiguraci sítě a jejích prostředcích včetně vzájemných vazeb.

Mezi hlavní výhody patří[14]:

- dynamičnost
- úspora prostředků
- přizpůsobitelnost
- konfigurovatelnost

Koncept SDN má 3 vrstvy[15] (obrázek 5.1):

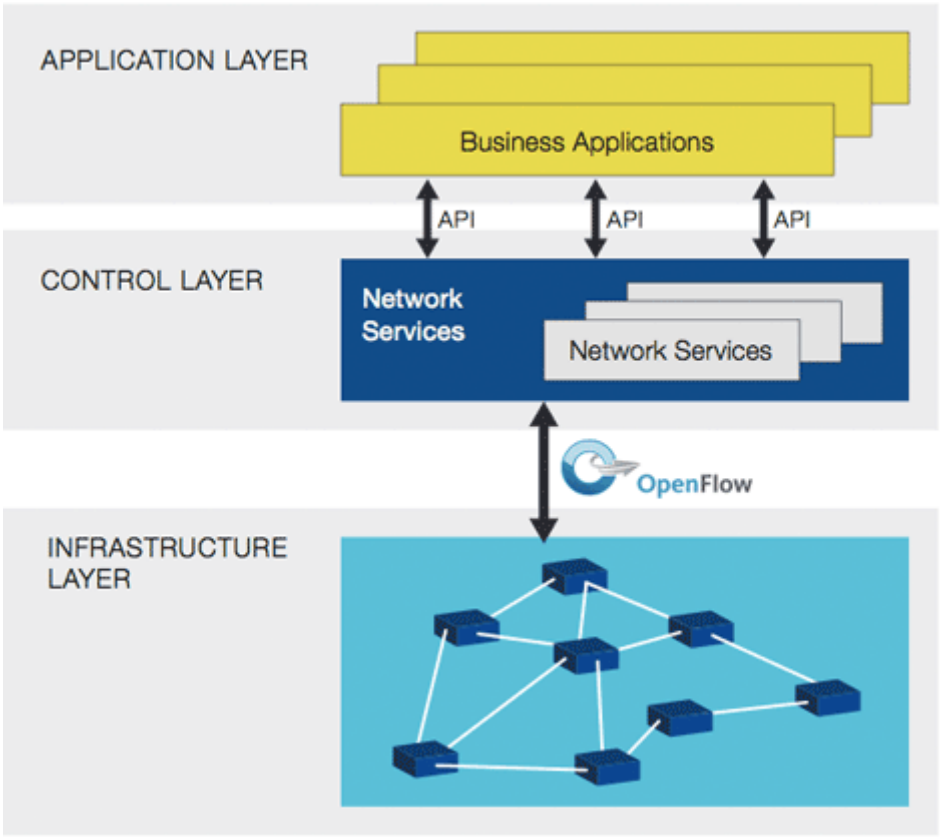
- Aplikační – vrstva, se kterou pracuje uživatel kontroléru za pomoci aplikací k tomu určených. Aplikace komunikují s kontrolérem za pomoci API [17] rozhraní, např. REST. Jedná se o tzv. Northbound rozhraní a kontrolér jich může mít více.
- Řídící – jedná se o jádro kontroléru, sbírající požadavky od aplikací a provádějící tyto požadavky na fyzických aktivních prvcích.
- Fyzickou – aktivní prvky v síti, řízené pomocí kontroléru. S ním komunikují nejčastěji protokolem OpenFlow. Jedná se o tzv. Southbound rozhraní a standardně je jedno.

[14] Software-Defined Networking (SDN) Definition. *Open Networking Foundation* [online]. 2016 [cit. 2016-08-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/sdn-definition`.

[15] Understanding the SDN Architecture. *SDxCentral* [online]. 2016 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture`.

[16] SDN layers [obrázek]. *Software-Defined Networking (SDN) Definition* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/sdn-definition`.

[17] viz. Seznam použitých zkratek – API



Obrázek 5.1: Vrstvy SDN.[16]

5.2 OpenFlow

OpenFlow (logo obrázek 5.2) je jedním z prvních protokolů, umožňujících komunikaci mezi aktivním prvkem sítě a SDN kontrolérem, který jej plně ovládá. Základ protokolu byl navržen na Stanfordské unvirzitě v roce 2008[18]. Základním chováním protokolu je, že pokud přijde paket, který není dosud známý, je předán kontroléru ke zpracování. Ten rozhodne, co se má s daným paketem provést a aktivní prvek si založí tento záznam do své tabulky. Když tedy později přijde stejný paket, resp. paket, který má stejné sledované znaky, ví již, co s ním provést a kontroléru se již nedotazuje, aby nedocházelo ke zbytečným zpožděním v komunikaci. Nicméně toto výchozí chování lze samozřejmě modifikovat, kdy některé pakety mohou být na kontrolér zasílány, např. neustále i za cenu vyšší režie v síti.

[18]What is OpenFlow? Definition and how it relates to SDN. *SDxCentral* [online]. 2016 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/what-is-openflow/`.

Obrázek 5.2: OpenFlow logo.[19]

Aktuální verze protokolu je č. 1.5, přičemž vždy po připojení aktivního prvku ke kontroléru dojde k vyjednání nejvyšší oboustranně podporované verze protokolu. Kontrolér tedy např. zvládá verze 1.0, 1.3 a 1.5, ale aktivní prvek jen 1.0 a 1.3. Dojde tedy ke komunikaci ve verzi 1.3. Z toho vyplývají i možnosti, co vše lze s daným prvkem provést, kdy každá verze protokolu má přidány určité možnosti.

Protokol umožňuje vkládání 3 typů záznamů do aktivního prvku (obrázek 5.3):

- Pravidla – určují, jakých paketů se bude určená akce týkat
- Akce – co se s paketem provede
- Statistiky – statistiky o zpracovaných paketech a síťových zdrojích

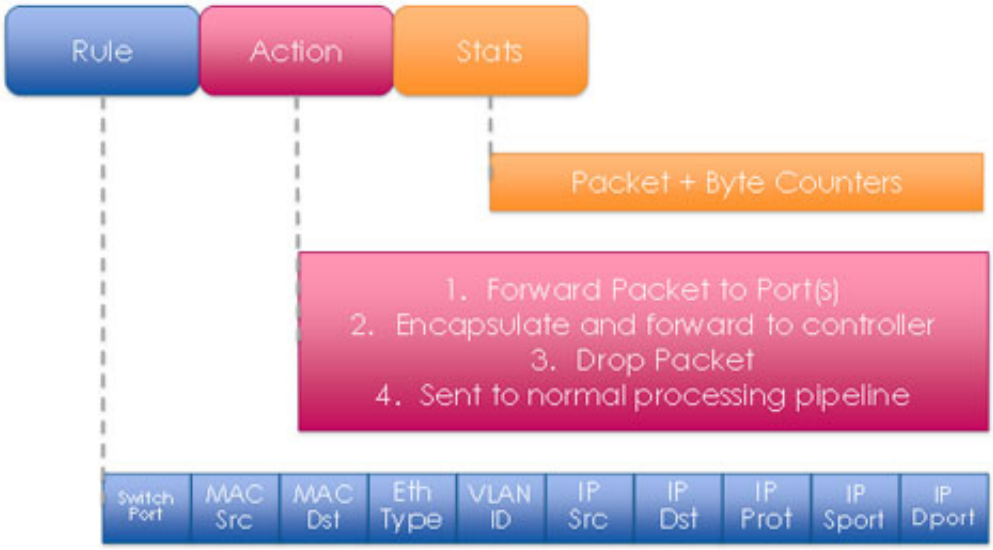
Vkládání záznamů do tabulek může v základu probíhat dvěma způsoby[21]:

- Reaktivně – po připojení prvku ke kontroléru je tabulka prázdná a postupně se plní záznamy, tj. na začátku se prvek ptá kontroléru na každý paket. Nevýhodou jsou větší doby odezvy na začátku, než se síť naučí provoz. Naopak výhodou je nezahlcení sítě ihned po jejím spuštění.

[19] OpenFlow [obrázek]. *Open Networking Foundation* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/openflow`.

[20] Flow-Table Entries That Can Be Manipulated in an OF Switch [obrázek]. *SDxCentral* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/what-is-openflow/`.

[21] SALISBURY, Brent. OpenFlow: Proactive vs Reactive Flows. *NetworkStatic* [online]. 2013 [cit. 2016-08-10]. Dostupné z: `http://networkstatic.net/openflow-proactive-vs-reactive-flows/`.



Obrázek 5.3: Typy OpenFlow záznamů.[20]

- Proaktivně – po připojení prvku ke kontroléru dojde k naplnění jeho tabulky. Při zpracování paketu tak již ví, co s ním provést. Zde dochází k prvotní obsáhlé komunikaci mezi kontrolérem a řízeným prvkem, nicméně pak již ke komunikaci povětšinou nedochází.

Případně může být aktivní i jejich kombinace, kdy dojde po připojení prvku ke kontroléru k určitému naplnění pravidly, ne však kompletnímu a zbytek se prvek doučí v průběhu, pokud vůbec bude záznamy potřebovat.

5.3 Mininet

Mininet je virtuální emulační software, umožňující provoz kompletních virtuálních sítí na jednom počítači a to včetně plnohodnotných koncových zařízení, přepínačů, směrovačů a jejich spojů.

V základu se ovládá pomocí příkazové řádky, je však možné i ovládání pomocí GUI, které je jeho součástí. Vzhledem k emulaci komplexní sítě lze tento program s výhodou využít pro práci v laboratorním prostředí, kde umožňuje testování, aniž jsou takto rozsáhlé sítě fyzicky dostupné.

V této práci je Mininet využit pro vývoj GUI a jeho otestování, umožňuje totiž i připojení externího SDN kontroléru a tak tvorbu SDN sítí. Je tak umožněn

pohodlný a především dostupný vývoj, kdy bez použití virtualizace by pro účely této práce byly takovéto rozsáhlé sítě obtížně dostupné.

5.4 Floodlight kontrolér

Floodlight byl původně vyvíjen firmou Big Switch Networks v rámci projektu OpenDaylight[8]. Je založen na kontroléru Beacon[22], což je jeden z původních SDN kontrolérů psaný v Javě, nicméně dnes již zastaralý. V této práci je použita verze 1.2.

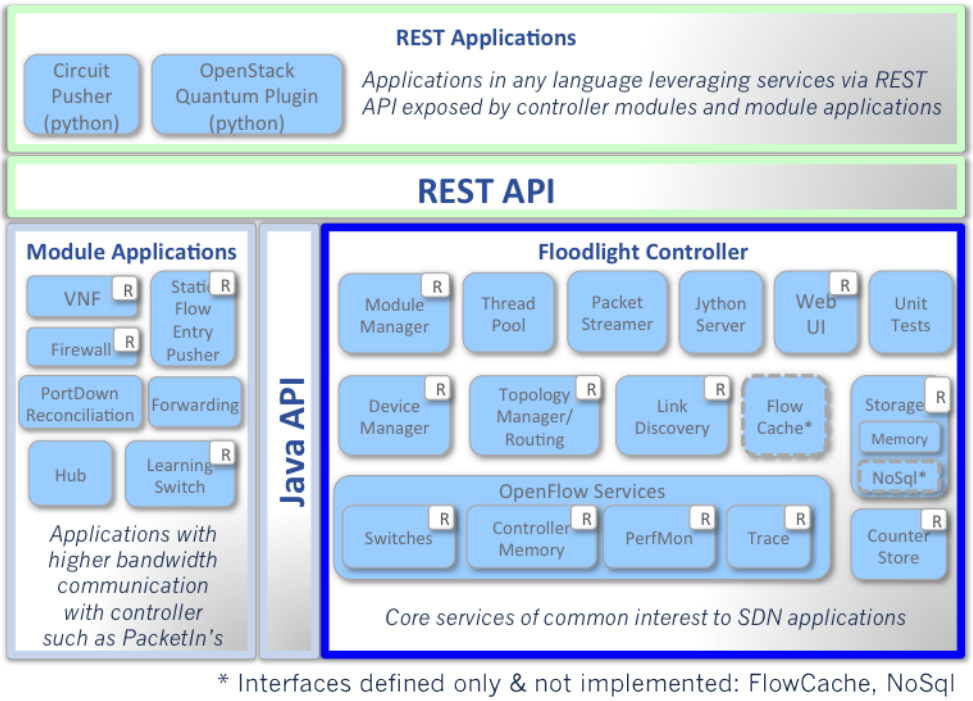
Má již v základu poměrně obsáhlou škálu funkcí, které lze navíc doplňovat pomocí 2 aplikačních rozhraní o další. Prvním rozhraním je REST, které je pomalejší a je využitelné pro aplikace, které nepotřebují okamžitou odezvu. Je tudíž vhodné např. pro GUI s přehledem kontroléru atp. Druhým rozhraním je Java API, které poskytuje odezvu v reálném čase a je tak vhodné pro moduly, starající se přímo o zpracování síťových informací, kdy je nežádoucí odezva nepřípustná. Kontrolér je tedy modulární a umožňuje pomocí konfiguračních souborů zapojení jen potřebných modulů. To se bohužel neobejde bez restartu kontroléru.

Výhodou tohoto kontroléru je vzhledem k jeho oblíbenosti poměrně obsáhlá dokumentace a aktivní komunita, tudíž případné řešení problémů je většinou dostupné. Dokumentace je dostupná na `https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+Documentation`.

Kontrolér samotný se skládá z několika funkčních celků, modulů (obrázek 5.4). Na vrcholu leží aplikace, komunikující přes Northbound API, které je v tomto případě řešeno pomocí REST rozhraní, jak již bylo zmíněno. Ze standardně dodaných aplikací v rámci kontroléru využívá tohoto rozhraní CircuitPusher, který vytváří cesty v síti mezi koncovými body. Dále aplikací, komunikujících pomocí Java API, které je rychlejší a je tedy využito pro aplikace, které vyžadují rychlejší odezvu a jsou náročnější, co se datového přenosu týče a toto rozhraní využívají i moduly, dodávané přímo s kontrolérem, jako je např. modul LearningSwitch,

[22] SDN Series Part Five: Floodlight, an OpenFlow Controller. RAO, Sridhar *The New Stack* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `http://thenewstack.io/sdn-series-part-v-floodlight`.

[23] Architecture [obrázek]. *Architecture - Floodlight Controller - Project Floodlight* [online]. 2012 [cit. 2016-09-10]. Dostupné z: `https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Architecture`.



Obrázek 5.4: Architektura Floodlight kontroléru.[23]

který zajišťuje chování připojených prvků jako klasických přepínačů, učících se z již proběhlého provozu.

Oba aplikační bloky komunikují pomocí svých rozhraní s jádrem kontroléru. To se stará o samotnou funkci, práci se sítí a jejími prvky. Moduly dostupné pomocí REST rozhraní jsou označeny písmenem R v bílém čtverci a jak je z tohoto přehledu vidět, perzistentní úložiště ještě není implementováno, má jen definováno rozhraní, což přináší určité nevýhody, které jsou později v této práci rozebrány.

6. Návrh řešení

Výsledná aplikace, vytvořená v rámci této práce, byla navržena jako platformně nezávislá, kdy kontrolér přijímá pomocí REST rozhraní požadavky uživatele, přičemž tyto požadavky jsou uživatelem vytvářeny za pomoci webového grafického rozhraní, které je vykreslováno ve standardních webových prohlížečích.

Navržené grafické rozhraní bylo tedy implementováno jako webová signle-page aplikace a to za pomoci frameworku AngularJS. Jedná se o standardní HTML 5 stránku, ovládanou pomocí Javascriptu, se vzhledem definovaným pomocí CSS stylů. Toto grafické rozhraní komunikuje s kontrolérem za pomoci REST rozhraní a předává tak požadavky na kontrolér, serverovou část, která se stará o samotné řízení připojené sítě.

Jako hlavní výhodu takovéto aplikace, oproti desktopové, běžící na lokálním přístupovém zařízení uživatele, lze označit snadnou přenositelnost mezi operačními systémy. Takto je aplikace dostupná z jakéhokoli zařízení, které je schopné otevřít webový prohlížeč. Navíc je aplikace dostupná odkudkoliv a celkové nároky na přístupové zařízení jsou minimální. Není potřeba doinstalovávat žádná virtuální běhová prostředí ani knihovny.

Aplikace je plně integrována do kontroléru, je jeho nedílnou součástí a nahrazuje i originální GUI, dodávané s kontrolérem. Díky této plné integraci je zabezpečeno, že aplikace může s kontrolérem pracovat v plném rozsahu a to včetně konfiguračních změn. Navíc není nutné zajišťovat její běh na externím webovém serveru, ale běží okamžitě po spuštění kontroléru na jeho integrovaném Restlet serveru a je tak vždy dostupná, pokud je kontrolér v provozu, za předpokladu, že je v konfiguraci kontroléru aktivní příslušný modul, zpřístupňující příslušné rozhraní.

Problémem, který bylo nutno při tvorbě GUI vyřešit bylo, že při změně konfigurace je nutné kontrolér restartovat. Vzhledem k tomu, že se jedná o standardní Java aplikaci (.jar), nikoli o webovou aplikaci (.war), která by běžela v aplikačním kontejneru, jako je např. Apache Tomcat, bylo nutné zajistit, aby se po restartu kontrolér opětovně spustil. Toho bylo docíleno za pomoci Java service wrapperu, který kontrolér spouští jako službu a sleduje jeho stav. Při jeho restartu je tak schopen ho opětovně spustit a to samé při jeho pádu, ať už z jakéhokoli důvodu. Navíc se stará i o aplikační logy.

Určitým problémem tohoto kontroléru je, že při restartu neuchovává v žádném perzistentním úložišti již nastavenou konfiguraci. Tato vlastnost by se však v budoucnu měla změnit a ani nyní to není až takový problém, vzhledem k tomu, že aktivní prvky jsou schopny si samy držet zadané záznamy, pokud není v konfiguraci kontroléru nastavena volba, která při iniciačním vyjednávání s aktivním prvkem po jeho připojení ke kontroléru smaže obsah jeho tabulek. V opačném případě veškerá již provedená konfigurace zmizí a je nutné začít od začátku.

Aby měl uživatel kontroléru možnost přesně vidět, co s kontrolérem pomocí GUI provádí, je v navrženém řešení možnost zapnutí si zobrazování sestavených JSON[24] objektů, které budou odeslány na kontrolér a bude pomocí nich provedena konfigurační změna. Má tak dokonalý přehled, co provádí a zároveň i možnost provádět stejnou konfiguraci ručně, pomocí REST rozhraní, či konzole integrované do GUI nebo si dané příkazy ukládat a poté např. pomocí skriptu provést naprosto stejnou konfiguraci. Tato navržená funkcionalita tak může velmi zpříjemnit opakovanou práci s kontrolérem i posloužit pro výukové účely.

Je tedy možné poprvé kontrolér nakonfigurovat ručně, pomocí GUI, se zaznamenáváním jednotlivých objektů a jejich snadným sestavováním pomocí rozhraní a následně si vytvořit např. Bash skript, který pomocí programu CURL provede všechny změny a obnoví tak výchozí stav kontroléru. Toto řešení tak lze využít i po restartu kontroléru a obnově jeho nastavení.

Návrh GUI počítá i s integrovanou konzolí, pomocí které je možné ovládat kontrolér na přímo, a to výběrem URL, HTTP metody a dat ve formě JSON objektu, která se mají poslat. Jedná se o zjednodušení práce s REST rozhraním, kdy je možné takto s kontrolérem pracovat na přímo i bez dalšího externího programu. Lze takto ovládat i možnosti, které v budoucnu přibudou v kontroléru a které tudíž nejsou nyní v GUI implementované.

6.1 Autentizace a autorizace

Důležitým prvkem v návrhu je oblast autentizace a autorizace uživatele. Jedná se o podstatnou oblast, která zamezuje případné neoprávněné manipulaci s kontrolérem a jeho prostředky. Zároveň zahrnuje i oblast přístupových záznamů, kdy je z nich pak zřejmé, kdo, co a kdy provedl za změnu a je za ni tedy odpovědný.

[24] viz. Seznam použitých zkratek – JSON

V základu bohužel kontrolér neposkytuje žádné možnosti autentizace uživatele a tím pádem ani autorizaci operací v systému. Bylo tedy nutné do návrhu úprav kontroléru tyto dvě oblasti začlenit a zaznamenávat do přístupových záznamů aplikace autentizační i autorizační údaje uživatelů.

Kontrolér tedy nyní obsahuje SQLite databázi s uživateli a jejich oprávněními, pomocí které je zajištěna schopnost kontroléru pracovat s autentizačními a autorizačními mechanismy, resp. ověřovat poskytované údaje. Dále tato databáze obsahuje texty pro nápovědu u nastavení kontroléru. SQLite databáze byla zvolena díky své přenositelnosti (databáze je v jednom souboru) a tomu, že není nutné instalovat databázový server, který by v tomto případě byl pravděpodobně zbytečný, když databáze je velmi malá a využívaná v podstatě jen na autentizaci/autorizaci. Zároveň by se tím zbytečně zkomplikovala instalace kontroléru, která takto zůstává velmi jednoduchou a kompaktní.

Začlenění proběhlo na backendové části, ve které je do modulu obsluhujícího REST rozhraní doimplementována autentizace. Dále je vytvořen modul, starající se o zajištění dostatečných oprávnění pro každý zdroj, poskytovaný daným rozhraním. Tato oprávnění jsou zároveň přenášena do frontendové části, ve které jsou přihlášenému uživateli zobrazovány pouze nabídky a možnosti, na které má oprávnění[25]. Při používání GUI se tedy ani nestane, že by se uživatel snažil provést akci, na kterou oprávnění nemá, nicméně se o to může pokusit pomocí integrované konzole nebo externím programem. Ale vzhledem k tomu, že autorizace operace se provádí na kontroléru, dojde pouze k vrácení HTTP stavového kódu 401 – Unauthorized.

[25]HENGEVELD, Gert. Techniques for authentication in AngularJS applications. *Opinionated AngularJS* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `https://medium.com/opinionated-angularjs/techniques-for-authentication-in-angularjs-applications-7bbf0346acec#.dnqkj0c0h`.

7. Implementace

Tato kapitola obsahuje podrobnější informace o implementovaných částech vytvořeného GUI. Je rozdělena na část backendovou a frontendovou, přičemž každá obsahuje příslušné implementační detaily a vlastnosti. Kompletní zdrojové kódy jsou součástí příloh této práce. Aby kontrolér s integrovaným GUI správně fungoval, je nutné dodržet postup uvedený v manuálu na zprovoznění. Tento manuál je také součástí příloh.

7.1 Backend

Na serverové části byly doimplementovány části, obsluhující vytváření cest v síti za pomocí skriptu CircuitPusher.py, který je součástí kontroléru, dále správa uživatelů, včetně databáze, jejíž schéma je taktéž součástí příloh, modul umožňující změnu konfigurace kontroléru pomocí jeho konfiguračních souborů, autentizace a autorizace a další pomocné třídy. Všechny implementované části jsou součástí balíku net.jantus.almarsdn, jehož class diagram se nachází v přílohách i s obrázkem, znázorňujícím adresářovou strukturu, až na jednu vyjímku a tou je úprava třídy RestApiServer z balíku net.floodlightcontroller.restserver, která se stará o vynucení autentizace na všech dostupných URL kontroléru. Všechna implementovaná rozšíření REST rozhraní jsou dostupná na URL /almarsdn/....

7.1.1 CircuitPusher

Pomocí třídy CircuitPusherResource z balíku net.jantus.almarsdn.controller.web je obsluhován skript CircuitPusher_Auth.py psaný v jazyce Python. Kontrolér inicializuje externí proces, spouštějící tento skript s tím, že zachytává jeho výstupy pro vyhodnocení úspěšnosti spuštění a provedení operací. Na základě výstupu je tak uživateli do GUI zaslán výsledek zadané operace, jeho požadavku na vytvoření cesty v síti mezi koncovými body.

Pomocí vytvořeného GUI tak lze velmi jednoduše ovládat tento skript a vytvářet tak cesty sítí od jednoho zařízení k jinému. Není tedy nutné spouštět tento skript z příkazové řádky a zadávat potřebné parametry. Jediným omezením je, že oba koncové body musí být v době spuštění skriptu kontroléru známy.

Vzhledem k tomu, že skript je dodáván zároveň s kontrolérem, není připraven na dodatečnou autentizaci na REST rozhraní. Tu tak bylo nutné do skriptu doimplementovat, aby mohl s kontrolérem komunikovat. Upravený skript se nachází ve složce resources/apps/circuitpusher/circuitpusher_auth.py. Vytvořené cesty jsou ukládány do souboru circuits.json ve stejné složce, nicméně tento soubor je při restartu kontroléru mazán, a to z již zmíněného důvodu, kdy si kontrolér nepamatuje po restartu již vytvořené Flow záznamy, ale v tomto souboru cesty zůstávají. Pro uživatele v GUI to následně vypadá, jako že cesty existují z důvodu, že jsou z tohoto souboru načítány, nicméně není tomu tak, kvůli chybějícím Flow záznamům.

7.1.2 Uživatelé

O přidávání, odebírání nebo editaci uživatelů se starají především třídy UserResource, UsersIdResource a UsersResource, vše z balíku net.jantus.almarsdn.controller.web. První jmenovaná je využita pro situaci, kdy uživatel upravuje vlastní údaje. Druhá vyřizuje požadavky uživatelů, majících oprávnění editovat jiné uživatele. Poslední se stará o získání seznamu všech uživatelů a zakládání nových.

7.1.3 Konfigurace kontroléru

Hlavní část změny konfigurace kontroléru je implementována ve třídě SettingsIO z balíku net.jantus.almarsdn.controller.io. Ta zajišťuje zápis do konfiguračních souborů kontroléru a je tak možné její pomocí ovlivňovat konkrétní nastavení kontroléru i jaké moduly budou při příštím restartu načteny. Zároveň s konfigurací zapisuje i kdo a kdy provedl poslední změny, lze tak snadno zjistit, kdo byl za poslední změny zodpovědný a způsobil např. pád kontroléru. Zápis probíhá do souboru almarsdn_floodlight.properties a je abecedně řazen pro usnadnění případné ruční editace, pokud např. uživatel editací tohoto souboru způsobí pád kontroléru a není tak již možné toto chybné nastavení opravit z GUI.

Restartování kontroléru zajišťuje třída RestartResource z balíku net.jantus .almarsdn.controller.web. Ta pomocí dalšího vlákna s odloženým spuštěním naplánuje restart kontroléru. Spuštění pomocí časovače je nutné, jinak by již nebyl

kontrolér schopen odeslat odpověď, že požadavek na restartování přijal a vykonat nezbytné akce před samotným restartem. Třída zároveň pro GUI zajišťuje zjištění stavu restartu kontroléru, tj., zda již byl úspěšně restartován a je aktivní.

7.1.4 Autentizace a autorizace

Základ spočívá ve vynucení autentizace uživatele ve třídě RestApiServer pomocí HTTP Basic autentizace. Toho je docíleno za pomoci třídy ChallengeAuthenticator z balíku org.restlet.security. Ta se stará o získání autentizačních údajů z požadavku na server, pokud nějaké obsahuje. Dále jsou přidány filtry pro autentizaci a autorizaci a až následně je požadavek předán hlavnímu routeru kontroléru ke konečnému zpracování příslušnou třídou.

Autentizační filtr zjišťuje, zda má daný uživatel vůbec přístup k zdrojům serveru. Jako vedlejší úkol zajišťuje i přesměrování na výchozí stránku aplikace při zadání neplatné URL. Navíc obsahuje přemostění pro složku almarsdn/web, která obsahuje statické soubory používané frontendem a pro přístup k nim není vyžadována autentizace. Toto přesměrování se provádí i pokud autentizace není schválena a to z důvodu dotazu prohlížeče např. na ikonu.

Po úspěšném průchodu autentizačním filtrem následuje autorizační filtr, starající se o zjištění, zda má autentizovaný uživatel potřebná oprávnění pro požadovaný zdroj. Tento je identifikován na základě požadované URL. Pokud uživatel nemá dostatečné oprávnění, dojde k zalogování jeho činosti.

O samotném schválení/neschválení přístupu a vyhodnocení oprávnění se stará třída AuthProvider z balíku net.jantus.almarsdn.authentication. Její instance zjišťuje autentizační a autorizační údaje z přidružené SQLite databáze. Vyhodnocení autorizace operace probíhá postupně pro každou URL a pokud není shoda nalezena, je automaticky zamítnuta.

Přidávání dalších uživatelských oprávnění se provádí ve třídě Permisson z balíku net.jantus.almarsdn.db.model. Jedná se o výčet aktuálně používaných oprávnění. Při zadávání nových je třeba dodržet stávající pořadí, protože záznamy v databázi již jsou podle stávajícího pořadí a došlo by tak k přeházení oprávnění, což není pravděpodobně vhodné. Nová oprávnění by tak měla být vždy přidávána na konec. Zároveň je nutné doplnit tato oprávnění i do příslušného souboru frontendové části.

Oprávnění typu VIEW jsou využívána na frontendové části a rozhodují o částech GUI, které přihlášený uživatel uvidí. Druhým typem oprávnění je CHANGE, které využívá backend pro vyhodnocení oprávněnosti požadavku. Speciálním typem je RESTART, umožňující restartování kontroléru.

7.1.5 Pomocné třídy

Jedná se především o třídy, vytvářející a parsující JSON objekty na instance tříd, starající se o zápis do databáze a konfiguračních souborů v požadovaném formátu, převádějící uživatelská oprávnění, validující uživatelské vstupy a zpřístupnění všech implementovaných rozšíření REST rozhraní kontroléru.

Speciálním případem je třída Statistics z balíku net.jantus.almarsdn.controller. Ta se stará o pamatování si, zda je na kontroléru zapnut sběr statatistických dat, kdy kontrolér sám o sobě není schopen toto rozpoznat.

Třídy z balíku net.jantus.almarsdn.controller.web se starají o vyřizování požadavků na specifické URL. Základním routerem pro tyto třídy je třída Controller z balíku net.jantus.almarsdn.controller, která definuje implementované cesty. Samotné třídy pak jen provádějí vyhodnocení požadavku, zda odpovídá požadované HTTP metodě (např. GET, POST) a jeho případné zpracování.

Posledními nezmíněnými balíky tříd jsou balíky net.jantus.almarsdn.model a net.jantus.almarsdn.db. První zmiňovaný obsahuje modelové třídy, využívané pro komunikaci pomocí REST rozhraní. Jsou podle nich tedy vytvářeny a parsovány JSON objekty. Druhý balík obsahuje modelové třídy využívané databází. Jsou do nich tedy plněna data z databáze a odpovídají tak schématu tabulek v ní uložených.

7.2 Frontend

Celá frontendová část byla napsána od základu v Javascriptovém frameworku AngularJS. Vytvořená aplikace umožňuje základní obsluhu kontroléru pomocí REST rozhraní, definovaného ve verzi kontroléru 1.2 a doimplementovaných částí, popsaných v backendové části.

Aplikace se dělí na moduly a komponenty, které jsou pomocí Angularu načteny. Moduly obsahují kontroléry, HTML šablony a CSS soubory definující vzhled.

Zajišťují celkový chod aplikace a nacházejí se v příslušných podsložkách kořenové složky app. Komponenty jsou sdílené pro celou aplikaci a kontroléry je využívají jako podpůrné služby. Samotné kontroléry zajišťují navigaci mezi stránkami v rámci aplikace, přičemž se jedná o tzv. single-page aplikaci, jak již bylo zmíněno. V přílohách práce se nachází vyobrazení adresářové struktury aplikace.

Po otevření aplikace je uživatel, pokud již není přihlášen, přesměrován na přihlašovací formulář. Aplikace nabízí dva typy přihlášení a to trvalé pomocí LocalStorage a dočasné využívající SessionStorage. V obou případech je tedy pro přihlášení využito lokálního úložiště, poskytovaného prohlížečem. To je využíváno i pro uchování dalších nastavení uživatele u obou typů přihlášení, např. z důvodu obnovení stránky aplikace prohlížečem, kdy jinak by byl uživatel odhlášen, pokud by bylo přihlášení implementováno jen v Javascriptovém kódu bez perzistentního uložení. Po úspěšné autentizaci jsou v aplikaci načtena ze serveru oprávnění konkrétního uživatele a ta je tomuto přizpůsobena případným skrytím nabídek a možností. Každý uživatel si může měnit své heslo a nastavovat požadované preference pro jeho účet, např., zda chce zobrazovat sestavované JSON objekty, či hodnoty časových intervalů, obnovujících data.

7.2.1 Moduly

Základním modulem je app/app, který sdružuje všechny moduly i komponenty, používané v aplikaci. Jsou v něm definovány obslužné funkce pro vyvolávané události patřící celé aplikaci a to především související s autentizací uživatele. Dále modul obsahuje sdílené aplikační konstanty, obsahující definovaná práva a URL na REST rozhraní serveru, kontrolér zajišťující autentizaci a autorizaci uživatele, kontrolér pro navigaci v rámci aplikace a ostatní součásti, které jsou obtížně zařaditelné, např. obrázky a funkce na zobrazení sestavených JSON objektů.

Struktura ostatních modulů, umístěných ve složce app, přesně odpovídá vytvořenému GUI a konkrétní modul tak vždy zajišťuje obsluhu konkrétní stránky aplikace. Pro stránku Home je to tedy modul ve složce app/home, pro stránku ACL modul app/management/acl atd.

Každý kontrolér ve složce app vždy obsahuje definici, s jakými uživatelskými právy lze zobrazit jeho obsah, přičemž u vnořených kontrolérů je tato definice vždy u základního kontroléru, tj. pro Management je definice v app/management.js.

Kontrola oprávnění na úrovni kontroléru zajišťuje aplikaci i před přímým zadáním URL kontroléru do prohlížeče. V opačném případě by jinak bylo možné toto omezení velmi jednoduše obejít, protože pouhé nezobrazení nabídky zajišťuje autentizační kontrolér, který však oprávnění fakticky nevynucuje.

7.2.2 Komponenty

Komponenty jsou ve složce app/components a obsahují především služby, zajišťující komunikaci se serverovým REST rozhraním. Kontrolér se tedy dotáže příslušné služby na požadovaná data. Ta je získá ze serveru a předá je kontroléru, který o ně žádal. Obdobně při požadavku na změnu předá kontrolér tento požadavek službě, která ho provede a vrátí kontroléru výsledek operace, popř. aktualizovaná data.

Speciálními případy jsou utilsService, který sdružuje pomocné funkce a nekomunikuje tak se serverovou částí, version, zobrazující verzi aplikace, timeService, starající se o převod času, switchParserService, upravující formát portů na připojených aktivních prvcích, storageService, zapisující uživatelská data do úložiště prohlížeče a pathParserService, nahrazující proměnné části URL, které jsou definovány v aplikačních konstantách.

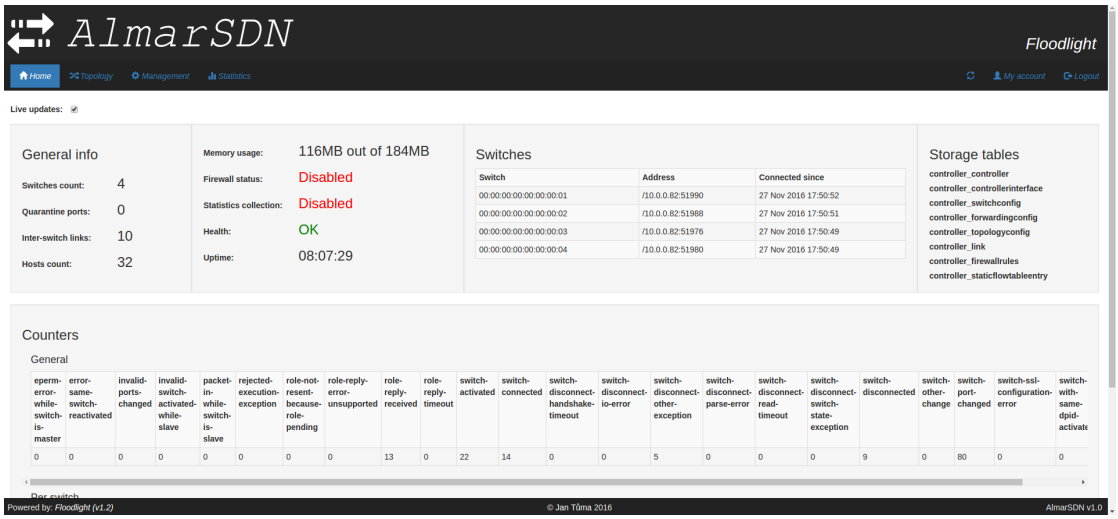
8. Možné využití

Kontrolér s integrovaným GUI lze využít jak k testovacím, tak produkčním účelům. V obou případech by GUI mělo zjednodušit a zpřehlednit práci s kontrolérem a to zejména méně pokročilým uživatelům, kterým výrazně usnadní zadávání požadovaných operací a nastavení. Nicméně i pokročilým uživatelům má co nabídnout a to zejména ucelený přehled o kontroléru a jím spravované síti.

Další výhodou je zabezpečení kontroléru pomocí autentizace a autorizace, která ho chrání před neoprávněným přístupem a nakládáním s jeho prostředky, umožnění vzdálené změny jeho konfigurace, jeho celkovou správu a to vše z jednoho místa. Není tak nutné vyhledávat jednotlivé konfigurační soubory a v nich požadovaná nastavení.

8.1 Ukázka použití

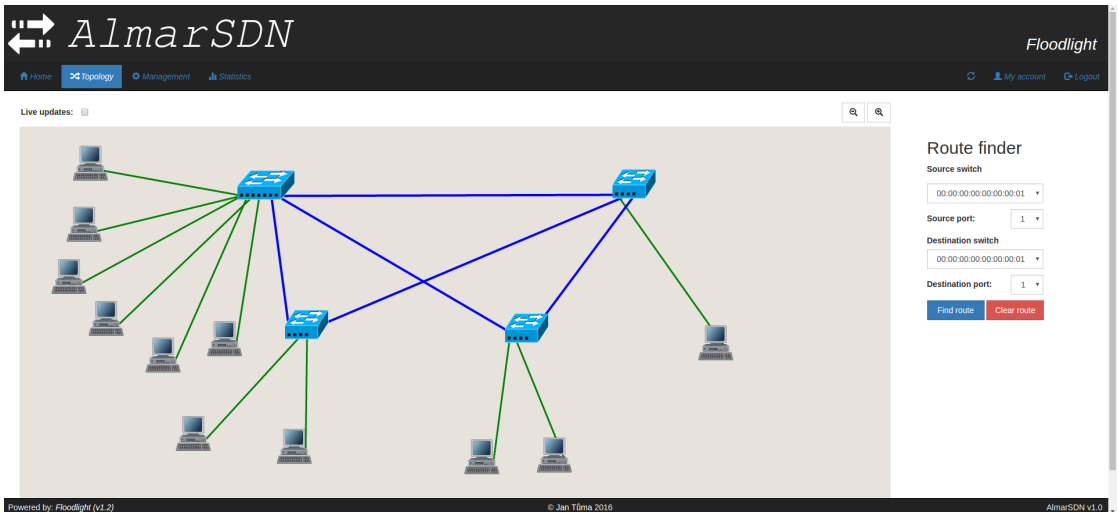
Následující obrázky ukazují, jak vypadá vytvořená grafická aplikace pro práci s kontrolérem a část možností této aplikace. Kompletní přehled o dostupných možnostech je součástí příloh této práce.



Obrázek 8.1: Základní obrazovka aplikace.

Obrázek 8.1 zobrazuje základní obrazovku aplikace, kterou uživatel uvidí po úspěšném přihlášení, přičemž horní obslužné menu zůstává vždy stejné i během přepínání karet. Je zde základní přehled o stavu kontroléru, připojených přepínačích, počtu spojení, využití paměti, stavu integrovaného Firewallu, zda probíhá sběr statistických dat a další základní informace, jako jsou počty přenesených paketů, chyby atp. V pravém horním rohu se nachází jméno autentizovaného uživatele a prvky umožňující správu jeho účtu, obnovení aktuální stránky a odhlášení z aplikace. Na stránce je možné zapnutí aktualizací v reálném čase a mít tak aktuální přehled o kontroléru bez obnovování stránky.

Na konkrétních stránkách dochází pouze k dočasné aktivaci/deaktivaci tohoto nastavení, přičemž trvale lze toto nastavení ovlivnit v nastavení účtu uživatele, kde se nachází i konfigurace časového rozpětí, tedy jak často jsou aktualizace prováděny. Zde je vhodné volit větší hodnoty, aby nedocházelo ke zbytečnému zatěžování kontroléru, kdy tyto aktualizace vyžadují poměrně velké množství dotazů na jeho REST rozhraní.



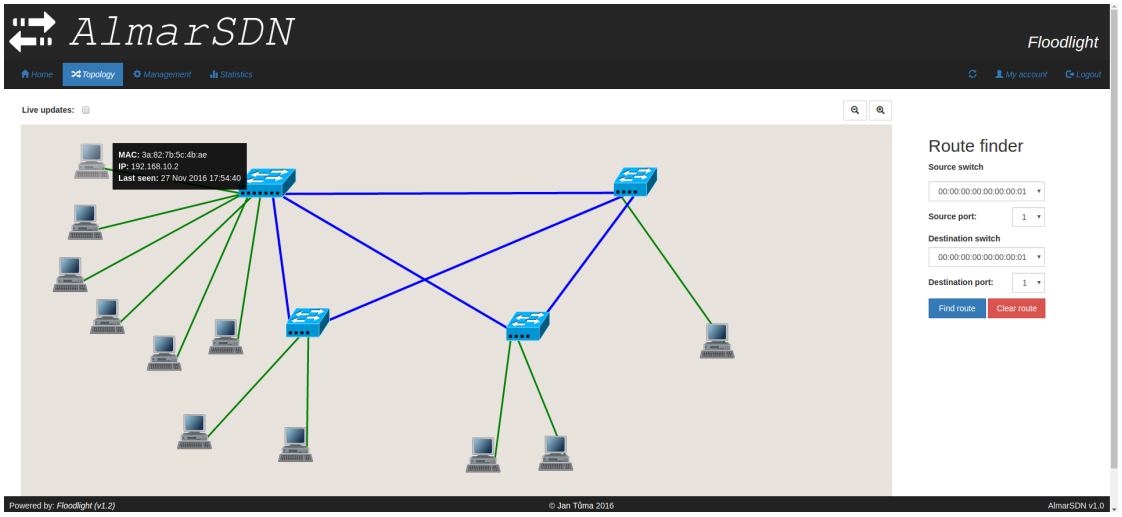
Obrázek 8.2: Topologie sítě.

Další kartou v nabídce je karta, zobrazující topologii připojené sítě, jak lze vidět na obrázku 8.2. Obsahuje kompletní topologii sítě tak, jak ji vidí kontrolér. Koncová zařízení jsou připojena pomocí zelených spojů a spoje mezi aktivními prvky jsou znázorněny modrou barvou. Konkrétní spoj vede vždy do příslušného

portu přepínače, tudíž je již přímo z vykreslené topologie vidět, kam je které zařízení připojeno, do jakého portu. Zobrazená zařízení se dají libovolně přesouvat pomocí myši a v horní části se nacházejí ovládací prvky pro přiblížení a oddálení zobrazené topologie.

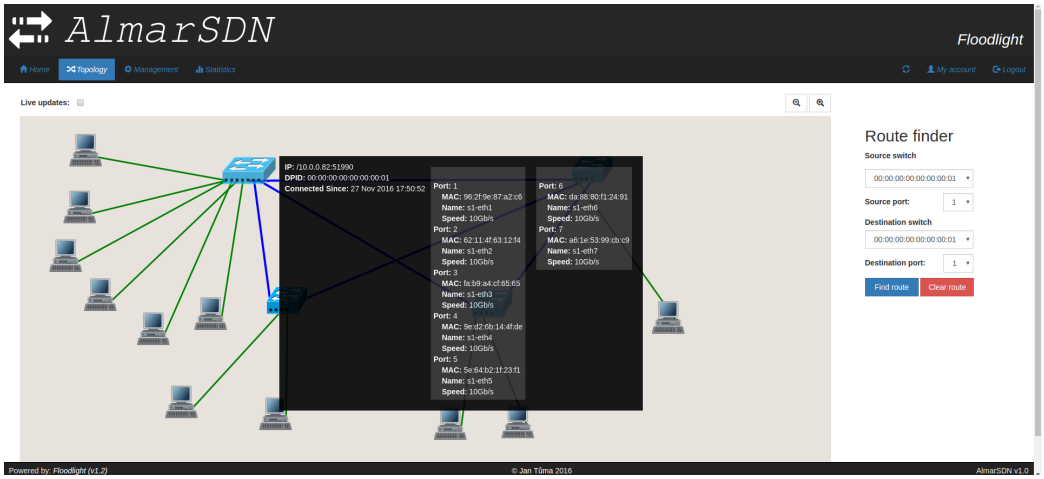
Topologii lze také aktualizovat v reálném čase, nicméně je zde určitá nevýhoda, kdy dojde k překreslení topologie a uživatelovo rozmístění je tak přepsáno do výchozího stavu. Zároveň dojde k načtení nových dat do modulu pro hlednání cesty sítí. Aktualizace je zde tak pravděpodobně vhodější provádět ručně.

Tato konkrétnní topologie je zobrazením testovacího scénáře č.3, kdy kontrolér neví o přepínači, nepodporujícím OpenFlow protokol, a všechna koncová zařízení do něj připojená se tak tváří, jako by byla připojena do jednoho portu levého horního přepínače. Nepřipojený přepínač tak sice není vykreslen, ale i tak je zřejmé, že na daném portu musí nějaké obdobné zařízení být připojené. Jen není jasné, kolik jich může být.



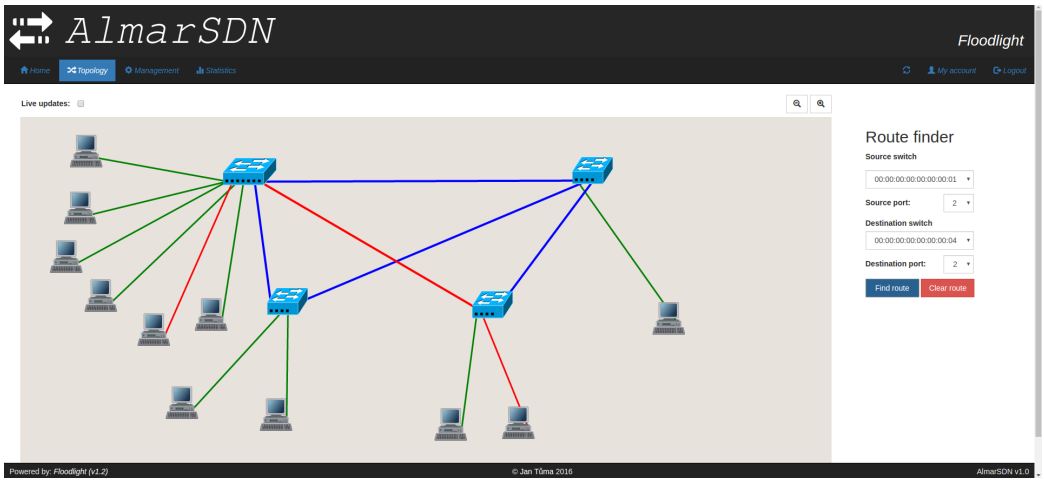
Obrázek 8.3: Topologie sítě – koncové zařízení.

Nastavením kurzoru na příslušné koncové zařízení (obrázek 8.3) lze zobrazit konkrétní informace o připojeném zařízení. Jedná se o MAC adresu, IP adresu a kdy bylo zařízení naposledy aktivní. Tyto informace má kontrolér o každém připojeném zařízení a je tak z jednoho místa dobře vidět konkrétní informace, bez složitého hledání.



Obrázek 8.4: Topologie sítě – přepínač.

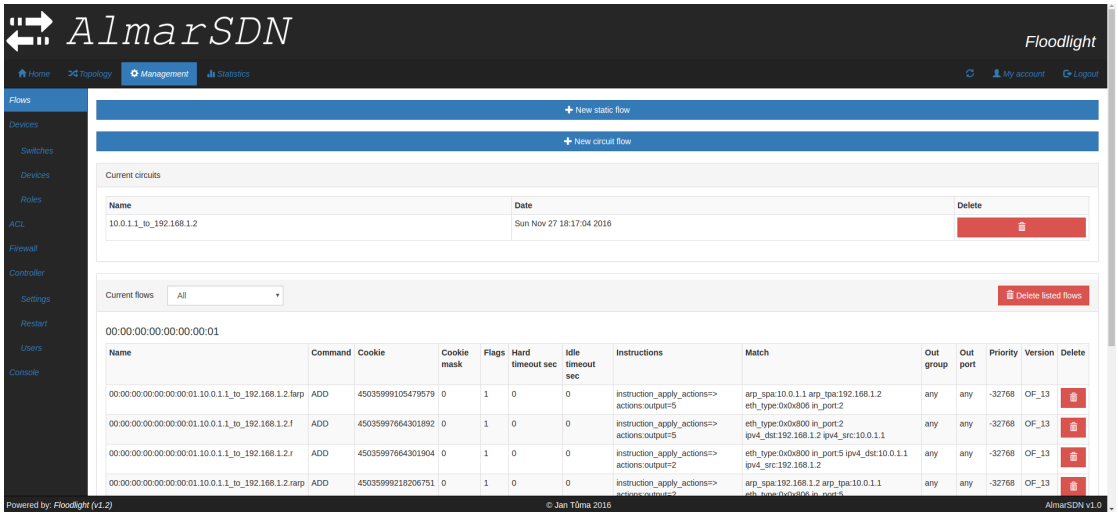
Na dalším obrázku 8.4 je vidět, jak vypadají informace o připojeném přepínači. Tento dialog obsahuje daleko více informací oproti koncovému zařízení, a to informace nejen o samotném přepínači, jako je jeho IP adresa, označení v rámci OpenFlow protokolu a od kdy je připojen ke kontroléru, ale i o jeho jednotlivých portech. V tomto výpisu jsou záměrně vynechány lokální porty a porty, sloužící pro spojení s kontrolérem, které by vykreslenou topologii znepřehlednily. U konrétního portu přepínače je vždy uvedena jeho MAC adresa, název portu a jeho rychlost.



Obrázek 8.5: Topologie sítě – hledání cesty.

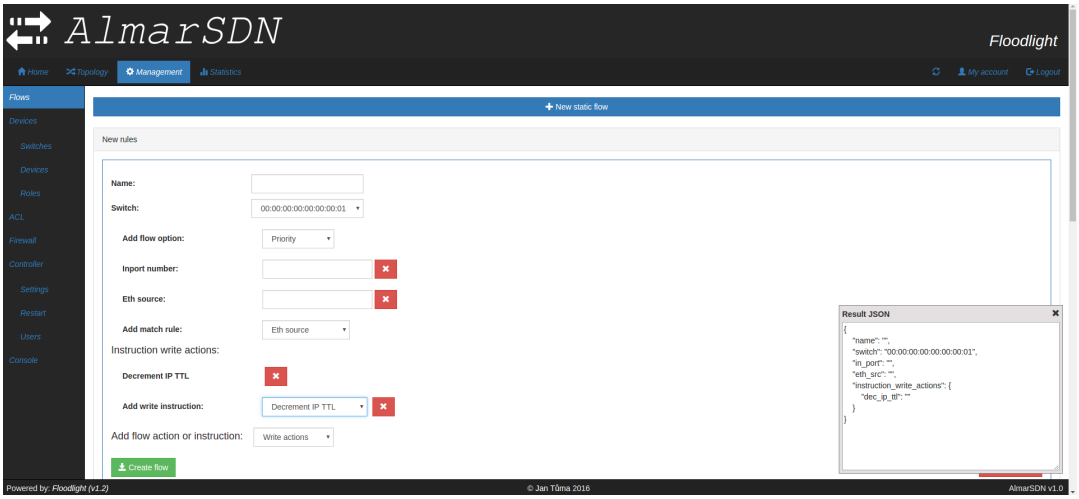
V pravé části karty, zobrazující topologii sítě, se nachází vyhledávání cesty sítí, tedy kudy jsou směrovány pakety z určitého portu přepínače na jiný port stejného, či jiného přepínače. Po vybrání počátečního a koncového bodu cesty a kliknutí na vyhledání cesty, je tato znázorněna pomocí červeného zvýraznění (obrázek 8.5). Lze tak snadno zjistit, kudy pro dané body probíhá komunikace. Toho lze s výhodou využít při potřebě změny konfigurace, kdy je naprosto jasné, kde je potřeba změnu provést.

Další a nejobsáhlejší kartou v hlavní nabídce je karta pro správu kontroléru (obrázek 8.6), jejímž základním bodem je správa Flow záznamů kontroléru a vytváření cest v síti pomocí skriptu CircuitPusher. Je zde kompletní přehled o vytvořených Flow záznamech a cest, přičemž záznamy lze filtrovat pro konkrétní přepínač. Zároveň se zde provádí případné odstraňování záznamů i vytvořených cest.



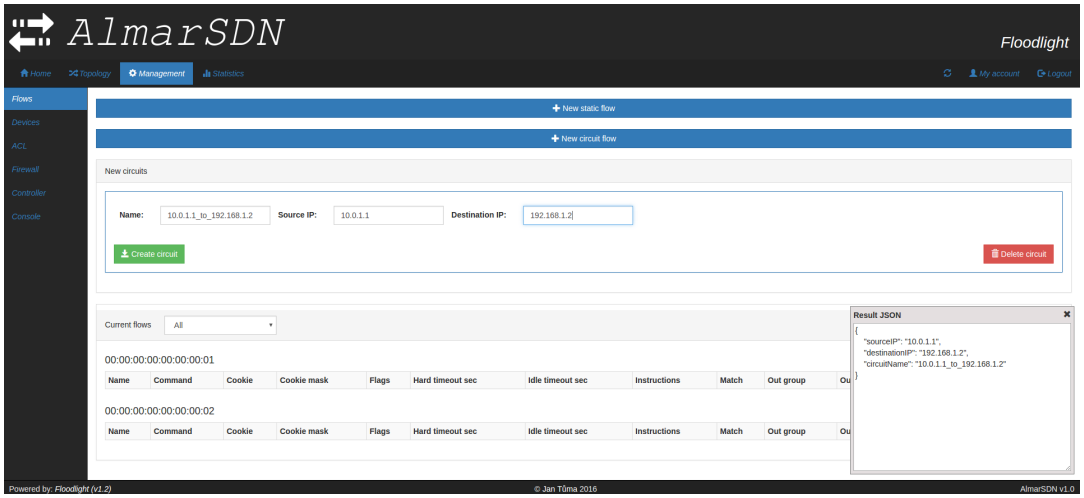
Obrázek 8.6: Flow záznamy.

V přehledu okruhů lze vidět jeho název a kdy byl vytvořen. U Flow záznamů jsou informace mnohem podrobnější, a to, jakého přepínače se týkají, jaké je konkrétní nastavení daného pravidla, tedy na co se vztahuje a co se má s paketem provést. GUI nabízí i možnost odstranění všech záznamů najednou a to buď pro konkrétní přepínač, nebo naprosto všech.



Obrázek 8.7: Přidání Flow záznamu.

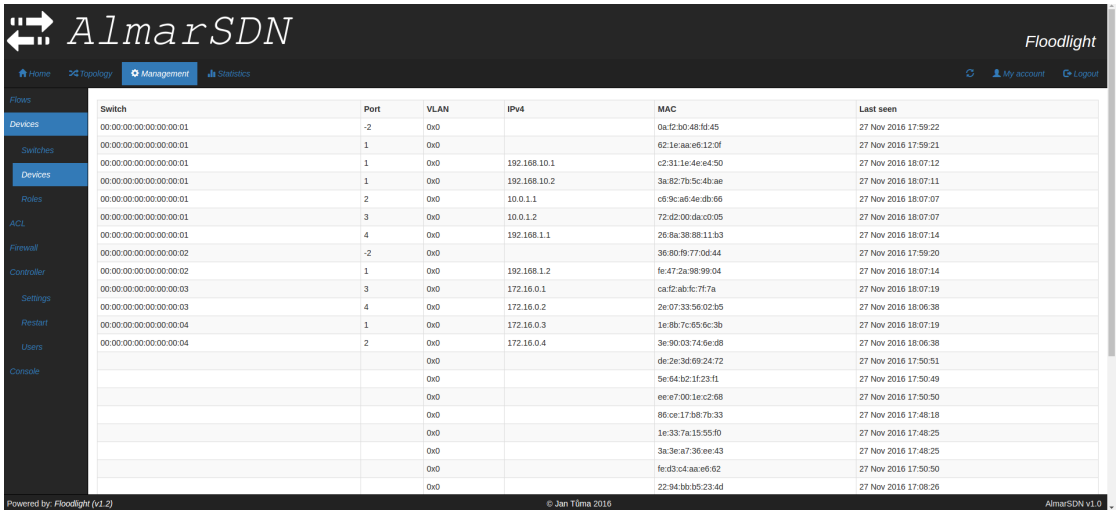
V horní části se nacházejí nabídky pro přidání nových Flow záznamů (obrázek 8.7). Je zde velké množství nastavení, která lze jednotlivě konfigurovat a výsledek, který bude odeslán na kontrolér, může být zobrazen v pravé dolní části jako JSON objekt. Nastavení lze odebírat/přidávat pouhým klikáním a je tak na první pohled vidět, co vše je možné nastavit, přičemž škála je velmi široká od portu, až po velmi pokročilé hodnoty.



Obrázek 8.8: Vytvoření cesty.

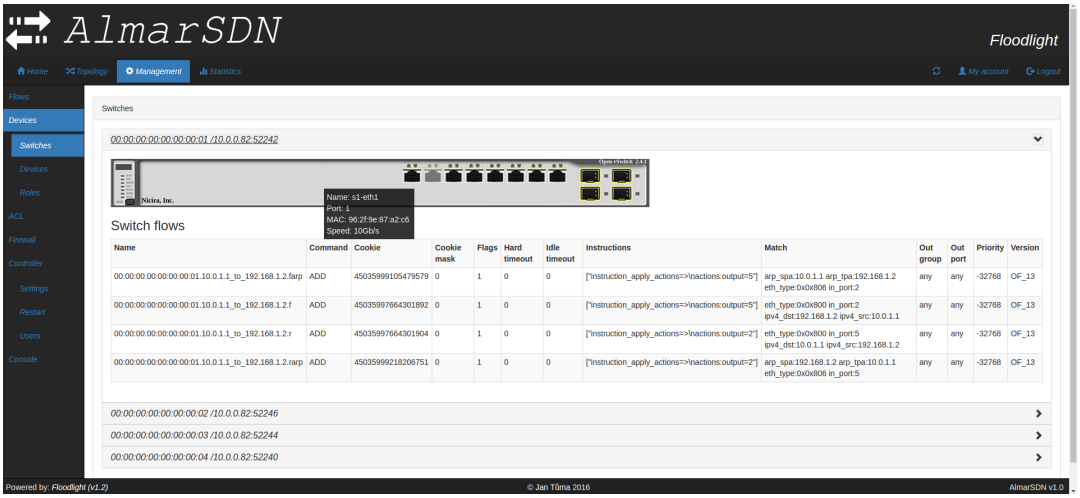
Naproti tomu nabídka pro tvorbu cesty sítí (obrázek 8.8) je výrazně jednodušší, kdy stačí zadat jednoslovný název vytvářené cesty a zdrojovou a cílovou IP adresu. O vše ostatní se postará kontrolér, který vytvoří potřebné Flow záznamy na příslušných přepínačích a umožní tak komunikaci mezi jednotlivými body sítě.

Vybráním příslušné podnabídky lze vyvolat další části pro ovládání kontroléru a zobrazení přehledů. Na obrázku 8.9 jsou zobrazena veškerá ke kontroléru připojená zařízení. Jedná se tedy o stručný přehled, díky kterému lze v textové podobě vidět, které koncové zařízení je připojeno do jakého portu přepínače a další informace o tomto zařízení, jako je IP adresa, MAC adresa, která nicméně nemusí být přímo zařízení, pokud je např. v cestě připojen klasický přepínač, jako je tomu u testovacího scénáře č.3 a kdy bylo zařízení naposled aktivní.



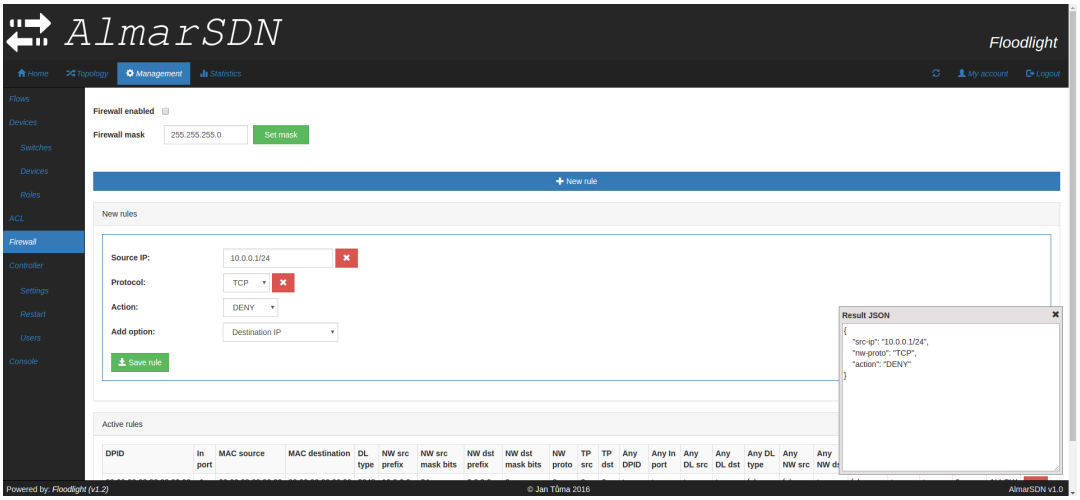
Obrázek 8.9: Přehled připojených zařízení.

Další podnabídkou je karta, zobrazující jednotlivé přepínače, připojené ke kontroléru (obrázek 8.10), včetně jejich grafického znázornění.Jednoduchým výběrem konkrétního přepínače je zobrazena jeho symbolická grafická reprezentace, kdy nastavením kurzoru na jeho porty lze vyvolat informační okénko s konkrétními informacemi o daném portu. Jedná se o jeho název, číslo, MAC adresu a rzchlost. Dále tato ilustrace obsahuje název výrobce, konkrétní model a software daného prvku.



Obrázek 8.10: Informace o přepínači.

Ve spodní tabulce se nachází Flow záznamy na vybraném konkrétním prvku, nicméně zde není možná jejich úprava a jedná se tedy jen o přehled aktivních záznamů pro konkrétní prvek. Obdobně lze tento přehled získat pomocí filtrace v kartě, určené pro práci s Flow záznamy.



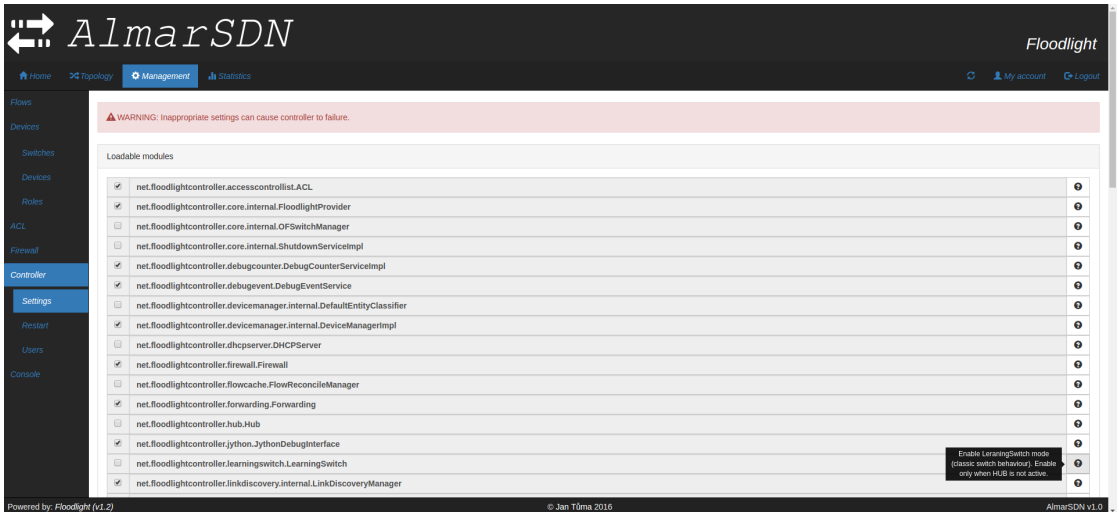
Obrázek 8.11: Nastavení Firewallu.

Karta s nabídkou nastavení Firewallu (obrázek 8.11) obsahuje veškeré nastavení, týkající se tohoto modulu, včetně jeho aktivace a přidávání konkrétních

pravidel, která jsou následně kontrolérem aplikována. Stejně jako na kartě pro Flow záznamy, lze i zde konfigurovat jednotlivá nastavení, popř. přidávat další dostupná z nabídky a výsledný JSON objekt může být zobrazen v přídavném boxu.

Ve spodní části jsou zobrazena již nakonfigurovaná pravidla, která jsou v případě aktivace Firewallu kontrolérem uplatňována. Zároveň lze zde konkrétní pravidla i odstraňovat.

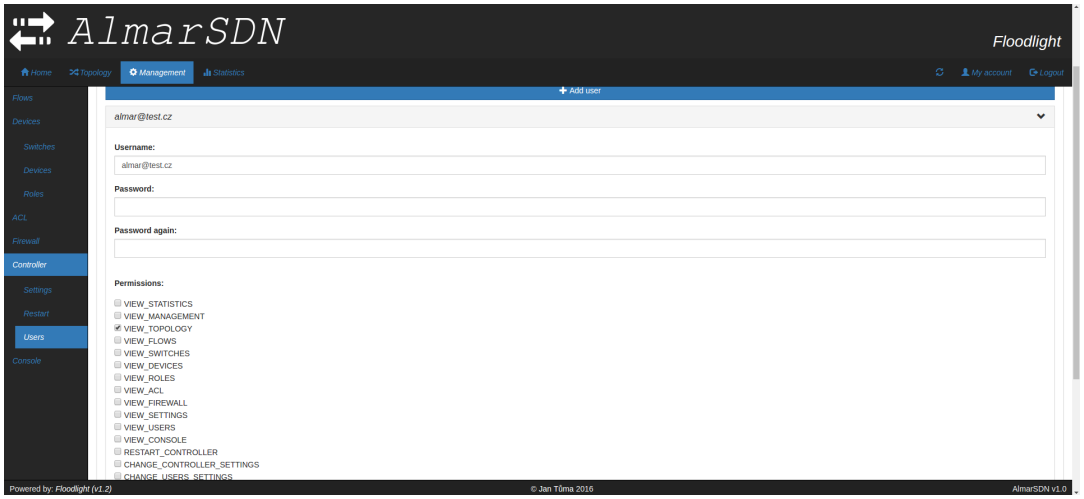
Nastavení samotného kontroléru je v nabídece Controller (obrázek 8.12). Zde lze nastavovat, které moduly budou při dalším startu načteny a veškerá další nastavení dostupná v konfiguračních souborech. Aktivace konkrétního modulu se provádí jeho vybráním a v pravé části se nachází nápověda, k čemu daný modul slouží a co jeho případná aktivace/deaktivace ovlivní. Ve spodní části této nabídky se nachází konrétní nastavení, tj. např., na jakém portu bude kontrolér naslouchat, kde bude dostupné REST rozhraní a další. Zároveň lze z příslušné podnabídky vyvolat i restartování kontroléru a tak změny okamžitě aktivovat.



Obrázek 8.12: Nastavení kontroléru.

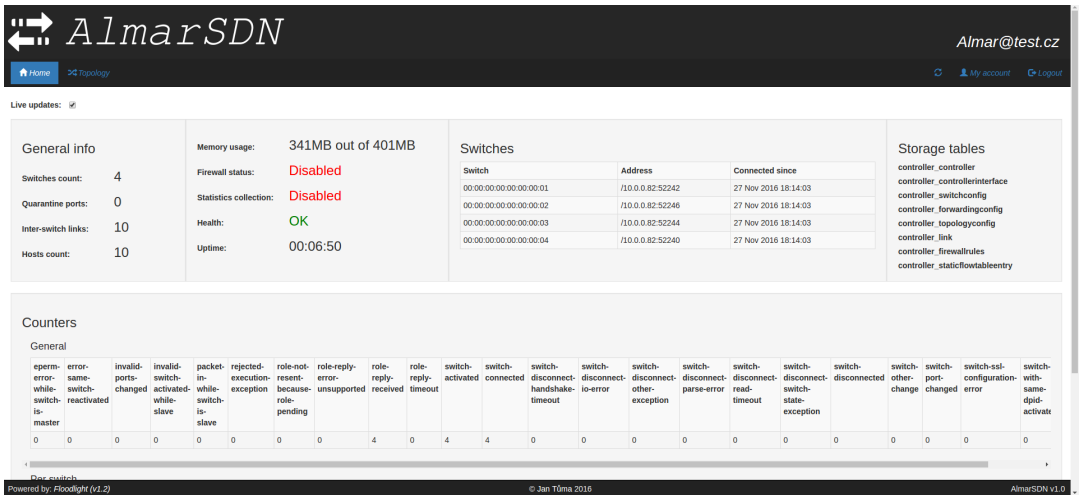
V podnabídce Users (obrázek 8.13) lze kompletně spravovat uživatele, mající přístup ke kontroléru. Provádí se zde editace stávajících, vytváření nových i jejich odstraňování a zejména editace uživatelových práv.

Úpravou uživatelových práv lze docílit velké přizpůsobitelnosti výsledné aplikace, což je vidět na obrázku 8.14, kdy daný uživatel může zobrazit jen domovskou



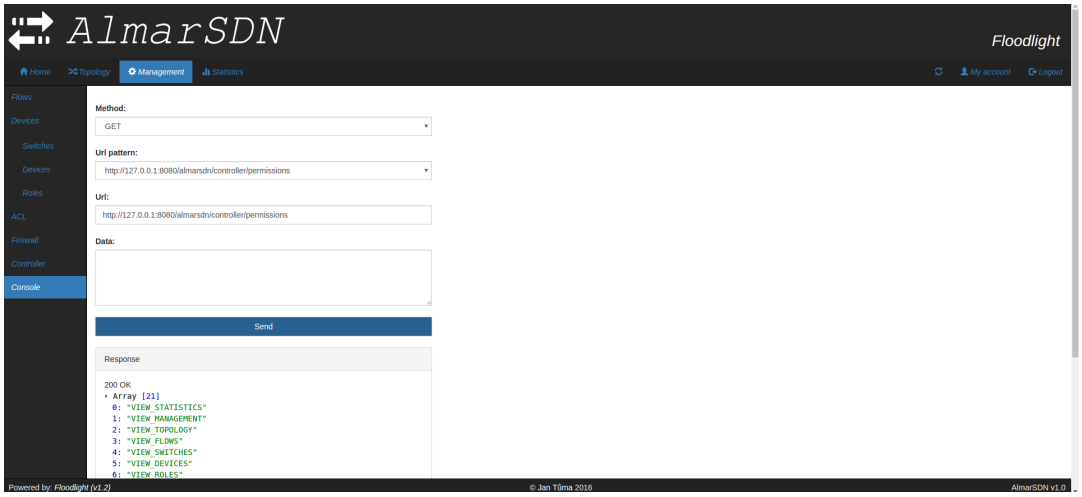
Obrázek 8.13: Správa uživatelů.

stránku aplikace a topologii sítě. Lze tak snadno ovlivňovat, který uživatel má mít k čemu přístup a to s velmi jemnými nuancemi, kdy lze ovlivnit i to, zda uživatel smí určitou fuknkci jen vidět, nebo má právo i na její editaci.



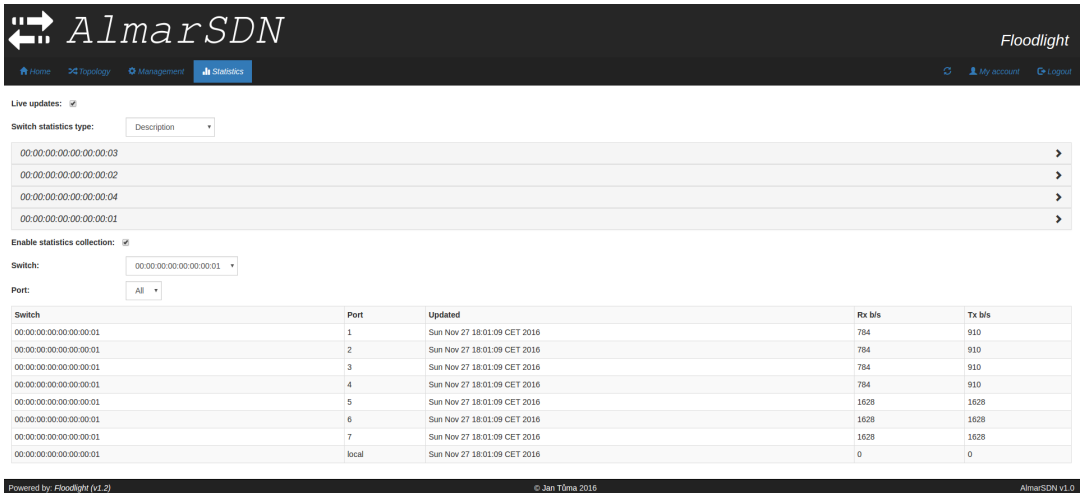
Obrázek 8.14: Upravené rozhraní uživatele s omezenými právy.

Poslední nabídkou z hlavní nabídky Management je Console (obrázek 8.15). Ta umožňuje práci s kontrolérem pomocí jeho REST rozhraní, kdy je ale již vyřešena autentizace s oprávněními právě přihlášeného uživatele. Lze tak jednoduše vybírat



Obrázek 8.15: Integrovaná konzole.

dostupné URL ze seznamu, určit požadovanou HTTP metodu a poslat případná data. Odpověď kontroléru je pak přehledně pomocí grafického znázornění JSON objektu zobrazena ve spodní části. Lze tak snadno získávat surová data přímo z kontroléru, ale bez použití nástrojů třetích stran, u kterých je nutné dodatečně řešit autentizaci a dohledávat konkrétní URL, která mohou být dotazována.



Obrázek 8.16: Statistiky kontroléru.

Poslední hlavní kartou (obrázek 8.16) je karta Statistics, která zobrazuje dostupné statistiky o kontroléru i informace o jeho součástech a připojených zařízeních. Výběrem požadovaného typu dat jsou tato načtena a zobrazena pod výběrovým boxem. Dále se zde provádí aktivace sběru dalších statistických dat, která jsou zobrazena v tabulce ve spodní části. Tato data lze následně filtrovat podle konkrétního aktivního prvku. I tato karta nabízí aktualizaci dat v reálném čase, kterou lze v horní části aktivovat/deaktivovat.

9. Testování

Testování probíhalo zejména ve virtuálním prostředí, které lze však označit vzhledem k potřebám této práce za plně dostatečné. Jak již bylo zmíněno, jako virtuální prostředí bylo použito emulační prostředí Mininet a pro fyzické testování směrovač TP-LINK TL-WR1043ND v2 s upraveným operačním systémem OpenWRT.

Obecně byly testy soustředěny především na funkčnost práce vytvořeného GUI s kontrolérem. Byla testována reakce kontroléru a GUI na chybné uživatelské vstupy, dále, zda je kontrolér dostatečně zabezpečen a není možný neoprávněný přístup k němu a neautorizovaná manipulace s jeho prostředky. Dále se testy zaměřovaly na práci kontroléru se sítí jako takovou.

Zvláštní pozornost byla věnována testování doimplementovaných částí na serverové části, kdy zvláště část autentizační/autorizační je kritickou a při jejím selhání by nebyla možnost s kontrolérem dále komunikovat. To zahrnovalo i testování přístupových práv uživatelů ke zdrojům serveru. Značnou pozornost vyžadovala i část, starající se o vytváření cest sítí, která komunikuje s již hotovým skriptem v jiném programovacím jazyce a musí tedy analyzovat jeho výstupy na standardní a chybový výstup a část starající se o změnu konfiguračních souborů, která taktéž při selhání může způsobit pád kontroléru.

Grafická část aplikace byla testována zejména s ohledem na uživatelskou přívětivost, snadnost práce a schopnost správně komunikovat s kontrolérem a poskytovat uživateli odpovídající zpětnou vazbu. Poměrně kritickou částí bylo i testování uživatelských oprávnění v aplikaci, aby uživatelům s nízkými oprávněními nebyly vůbec přístupny funkce, ke kterým nemají mít přístup a nemohli toto omezení obcházet např. pomocí přepisování URL adres v prohlížeči, což by bylo možné pokud by byly možnosti pouze skryté.

Testování obou implementovaných částí proběhlo úspěšně a nebyly zjištěny žádné chyby, které by znemožňovaly práci s kontrolérem pomocí tohoto GUI, či dokonce způsobily jeho pád.

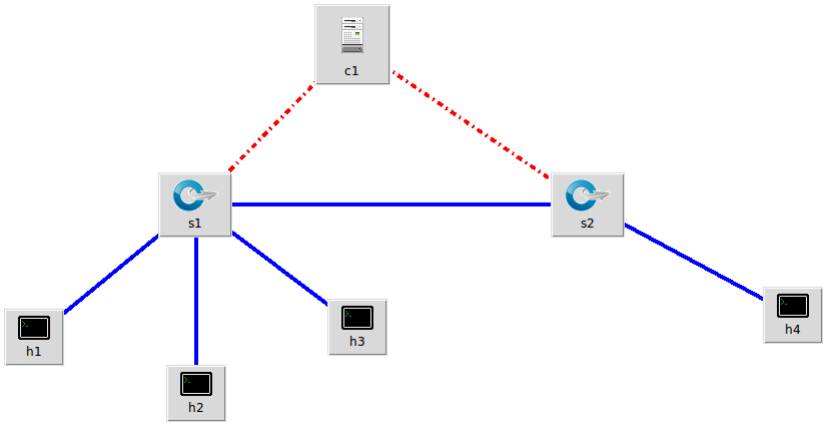
9.1 Virtuální prostředí

Pro testování ve virtuálním prostředí byly vytvořeny 3 testovací scénáře, přičemž každý měl prověřit schopnost vytvořeného GUI pracovat s danou spravovanou topologií, potažmo kontrolérem obecně.

Při testech byly vytvářeny cesty jak za pomoci standardních Flow záznamů, tak pomocí aplikace CircuitPusher. Cesty byly vytvářeny i mezi jednotlivými sítěmi a byla nastavována ACL a Firewall pravidla.

9.1.1 Testovací scénář č.1

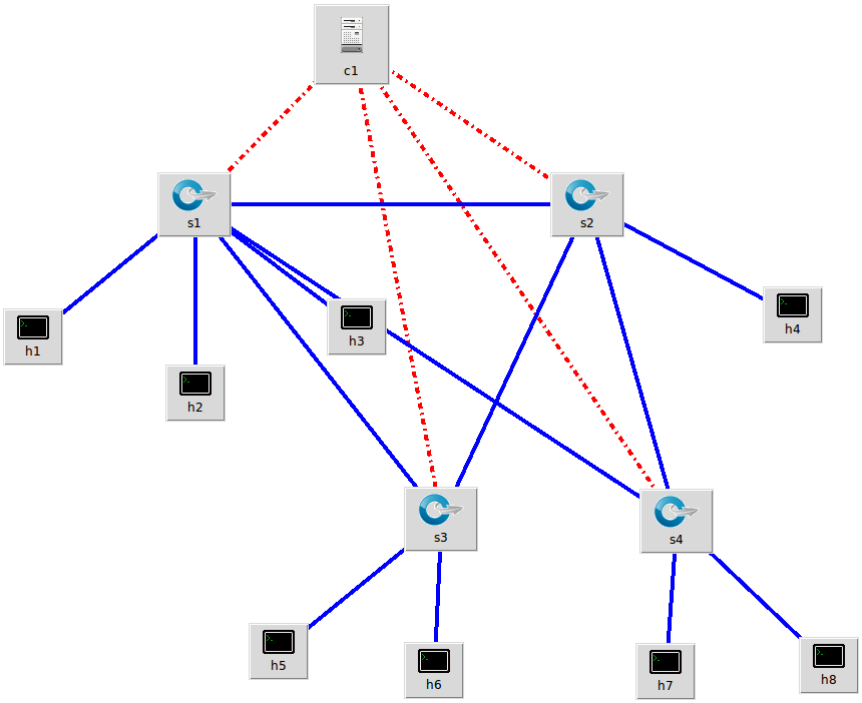
Testovací scénář č.1 (obrázek 9.1) obsahuje dvě podsítě a to 10.0.1.0/24 (h1 a h2) a 192.168.1.0/24 (h3 a h4). Jedná se o základní testovací scénář.



Obrázek 9.1: Testovací scénář č.1.

9.1.2 Testovací scénář č.2

Testovací scénář č.2 (obrázek 9.2) vychází ze scénáře č.1 a obsahuje tři podsítě a to 10.0.1.0/24 (h1 a h2), 192.168.1.0/24 (h3 a h4) a 172.16.0.0/24 (h5, h6, h7 a h8). Jedná se o složitější testovací scénář, kdy jsou navíc spoje mezi přepínači duplicitní.



Obrázek 9.2: Testovací scénář č.2.

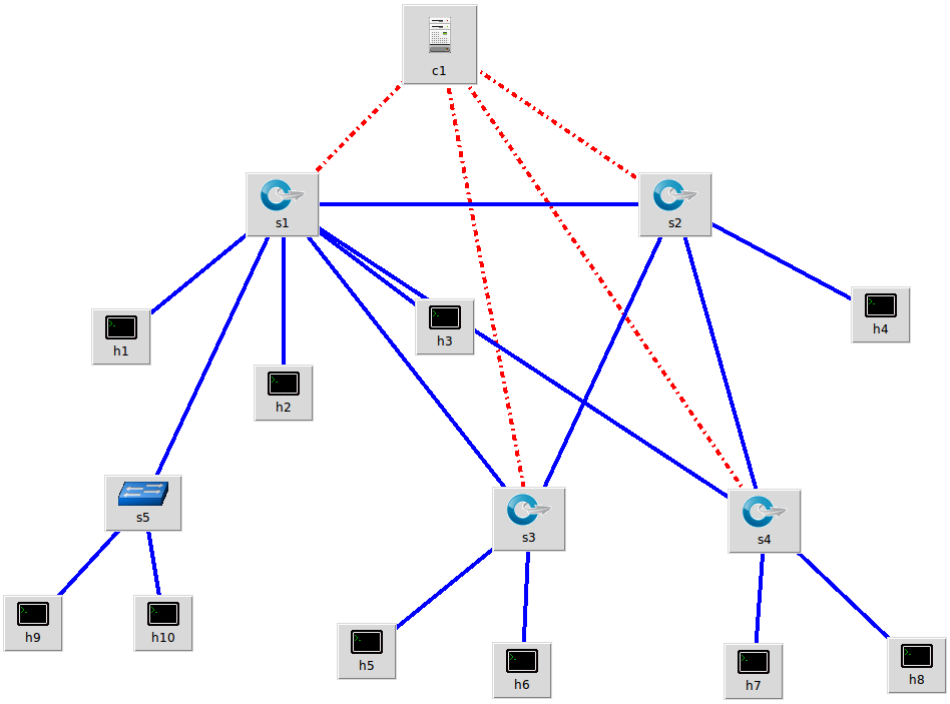
9.1.3 Testovací scénář č.3

Testovací scénář č.3 (obrázek 9.3) vychází ze scénáře č.2, je však navíc obohacen o externí síť 192.168.10.0/24 (h9 a h10), která není součástí sítě řízené kontrolérem, ale je připojena pomocí standardního přepínače.

Ve všech testovacích scénářích fungoval kontrolér dle očekávání a nebyly zjištěny žádné problémy. Lze tak tyto testy označit jako úspěšné.

9.2 Multi-vendor prostředí

Testování kontroléru ve fyzickém prostředí probíhalo na směrovači TP-LINK TL-WR1043ND v2 (obrázek 4.1) s upraveným systémem OpenWRT, který byl již použit v bakalářské práci pana Bc. Nebáznivého, který prováděl obsáhlé testo-



Obrázek 9.3: Testovací scénář č.3.

vání Floodlight kontroléru v multi-vendor prostředí[26] a jeho poznatky lze tak využít i pro testování v této práci. Směrovač má 5 portů, přičemž po připojení ke kontroléru a případnému dalšímu prvku zbývají pro koncová zařízení porty 3, což bohužel není mnoho.

Z výsledků jeho práce vyplývá[26], že tento směrovač neumožňuje současné provozování různých verzí protokolu OpenFlow a pro účely této práce byla tedy vybrána verze 1.3.

S touto verzí fungoval směrovač dle očekávání a komunikoval s kontrolérem. Nicméně dle výsledků pana Bc. Nebáznivého může při připojení více různorodých přepínačů ke kontroléru docházet k chybám a je pak nutné provést určitá opatření[26], tak, aby vše fungovalo.

[26]NEBÁZNIVÝ, Petr. *Cross-platform Software Defined Networking controller v multi-vendor prostředí.* p. 33, 46 – 47, 49 – 51, 53

[27]TP-LINK TL-WR1043ND [obrázek]. *Bezdrátový gigabitový N router 450 Mbit/s* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `http://cz.tp-link.com/products/details/cat-9_TL-WR1043ND.html`.

Obrázek 9.4: TP-LINK TL-WR1043ND v2.[27]

Dá se však říci, že po nezbytných úpravách je tento kontrolér schopen práce v multi-vendor prostředí a je plně funkční i na poměrně levném a běžném zařízení, jako je směrovač TP-LINK TL-WR1043ND v2 a naskýtá se tak možnost vytvoření poměrně levné centrálně řízené sítě, což může být relativně zajímavé vzhledem k cenám přepínačů, které mají protokol OpenFlow implementován standardně již od výrobce.

10. Diskuse a otevřené otázky

V rámci vylepšení práce s kontrolérem by bylo vhodné doplnit striktnější validaci vstupů na frontendové části, pomocí které by se odlehčil kontrolér a zamezilo by se zadávání nepatřičných hodnot do daných vstupních polí.

Další velmi užitečné rozšíření stávající aplikace by bylo doplnění kontroléru o perzistentní úložiště, za pomocí kterého by se dosáhlo toho, že by kontrolér zachovával nastavenou konfiguraci i po jeho restartování. Toto vylepšení by dle zveřejněné architektury (obrázek 5.4) měl v budoucnu obsahovat již samotný kontrolér, nicméně implementovaná funkčnost vytváření cest za pomoci python skriptu CircuitPusher by zřejmě touto cestou podporována nebyla. Lze pravděpodobně uvažovat o dvou možných cestách, jak potřebné funkčnosti zavedení cest po spuštění kontroléru dosáhnout.

První možností je uchovávat vytvářené cesty a po startu kontroléru je všechny opětovně zavést. Tento způsob bohužel, alespoň ve stávající verzi, naráží na fakt, že koncové body v síti, pro které je cesta vytvářena, musí být v době tvorby cesty kontroléru známy. Pokud tato podmínka neplatí, cesta se nevytvoří. Nelze tak tento způsob příliš doporučit.

Pravděpodobně lepší variantou uchovávání vytvořených cest po restartu kontroléru za pomocí tohoto skriptu by bylo zavádění již jen konkrétních Flow záznamů do tabulek přepínačů, tak jak je daný skript vytvořil při zadávání. Samotný seznam cest je již nyní uchováván v samostatném souboru a informace o nich jsou tedy již perzistentně uloženy. O uchování samotných Flow záznamů by se pravděpodobně již měla postarat budoucí verze implementovaného perzistentního úložiště. Nemělo by tedy být nutné tyto záznamy dodatečně ukládat, což by bylo poměrně náročné. V budoucí verzi kontroléru by tedy mělo jít o v podstatě již implementovaný systém, jedinou nutnou úpravou by bylo odstranění funkce, mazající soubor, obsahující údaje o cestách z disku při restartu kontroléru. To je však velmi jednoduchá úprava a lze tak tento způsob doporučit, pokud bude perzistentní úložiště skutečně ukládat veškeré aktivní flow záznamy na přepínačích.

Dále, aby nedocházelo k nekonzistencím mezi vytvořenými cestami tímto skriptem a Flow záznamy na přepínačích, bylo by pravděpodobně vhodné zamezit možnosti jejich jednotlivých odstraňování z příslušných Flow tabulek konkrétních přepínačů a povolit tak jen jejich správu pomocí GUI pro cesty v síti. Vzhledem

k tomu, že vytvořené Flow záznamy mají vždy určitý prefix, mělo by se jednat o poměrně snadný úkon, kdy by stačilo např. dané Flow záznamy s příslušným prefixem nezobrazovat (což není úplně vhodné), resp. zamezit jejich editaci, určitým způsobem je odlišit.

Jako velmi podstatnou část v případě využití navrženého řešení lze označit zabezpečení komunikace mezi kontrolérem a REST rozhraním pomocí šifrování přes protokol HTTPS, aby nebylo možné získávat přihlašovací údaje z hlavičky Authorization Basic autentizace protokolu HTTP. Nabízí se řešení provozovat REST rozhraní kontroléru přímo na HTTPS, kdy tento tuto možnost umožňuje, nebo umístit kontrolér za reverzní proxy server, který zajistí potřebné šifrování a REST rozhraní kontroléru již provozovat jen na lokální adrese, kde nehrozí neoprávněný přístup k údajům. K tomuto lze využít i např. webový server Apache, který má možnost fungovat jako reverzní proxy, starající se o šifrovanou komunikaci a následné předání požadavku dále.

11. Závěr

Hlavním cílem této diplomové práce bylo vytvoření nadstavbové webové grafické aplikace, umožňující základní ovládání SDN kontroléru a vykreslení topologie dané sítě. K dosažení tohoto cíle bylo nutné provést analýzu stávajících SDN kontrolérů a jejich grafického rozhraní. Následně vybrat vhodného kandidáta pro tuto práci a pro něj GUI vytvořit. Výsledkem této práce je tedy webové grafické rozhraní, pomocí kterého je možné kompletně ovládat kontrolér a zobrazovat topologii připojené sítě.

Hlavní výhodou ovládání kontroléru pomocí GUI by měla být jednoduchost takovéhoto ovládání, menší náchylnost ke konfiguračním chybám a celkově menší nároky na nezbytné znalosti obsluhy. Jako další výhodu použití tohoto GUI lze označit strukturovaný seznam dostupných funkcí kontroléru, a to pomocí menu grafického rozhraní, kdy je v podstatě okamžitě vidět, co vše lze na kontroléru nastavit a jak.

Dalšími cíli této práce bylo nastínit technologii OpenFlow, její základní aspekty a vyzkoušet funkčnost kontroléru, což bylo provedeno jak pomocí virtuálních sítí, tak na fyzickém směrovači. Navíc vzhledem k již obsáhlé práci na toto téma lze tento kontrolér doporučit, jelikož jako hlavní nevýhoda byla označena složitost konfigurace, která ovšem díky vyvinutému grafickému rozhraní odpadá[26].

Grafické rozhraní bylo za použití několika testovacích scénářů důkladně otestováno a vše je plně funkční. Za úvahu by stálo implementování některých navržených rozšíření, pokud se tak nestane ze strany tvůrců tohoto kontroléru. Všechny vytýčené cíle této diplomové práce se tak dají označit za splněné.

Seznam použité literatury

- 1PhoenixM/avior-service. *GitHub* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `https://github.com/1PhoenixM/avior-service`.
- Architecture [obrázek]. *Architecture - Floodlight Controller - Project Floodlight* [online]. 2012 [cit. 2016-09-10]. Dostupné z: `https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Architecture`.
- Controller Overview [obrázek]. *OpenFlow at Marist, Gwtting started* [online]. 2013 [cit. 2016-09-10]. Dostupné z: `http://openflow.marist.edu/avior/gettingstarted`.
- Controller Status [obrázek]. *Avior 2.0* [online]. 2014 [cit. 2016-09-10]. Dostupné z: `https://raw.githubusercontent.com/1PhoenixM/avior-service/master/assets/images/Avior.png`.
- Five must-know open source SDN controllers. *TechTarget* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `http://searchsdn.techtarget.com/news/2240225732/Five-must-know-open-source-SDN-controllers/`.
- Floodlight logo [obrázek]. *It's that time of year again* [online]. 2012 [cit. 2016-09-10]. Dostupné z: `http://www.bigswitch.com/blog/2012/12/13/its-that-time-of-year-again`.
- Flow-Table Entries That Can Be Manipulated in an OF Switch [obrázek]. *SDxCentral* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/what-is-openflow/`.
- HENGEVELD, Gert. Techniques for authentication in AngularJS applications. *Opinionated AngularJS* [online]. 2014 [cit. 2016-08-10]. Dostupné z: `https://medium.com/opinionated-angularjs/techniques-for-authentication-in-angularjs-applications-7bbf0.346acec#.dnqkj0c0h`.
- NEBÁZNIVÝ, Petr. *Cross-platform Software Defined Networking controller v multi-vendor prostředí.* 2016 [cit. 2016-09-05].

- OpenFlow [obrázek]. *Open Networking Foundation* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/openflow`.
- OpenFlow at Marist. *Marist SDN Lab* [online]. 2013 [cit. 2016-08-10]. Dostupné z: `http://openflow.marist.edu/avior`.
- SALISBURY, Brent. OpenFlow: Proactive vs Reactive Flows. *NetworkStatic* [online]. 2013 [cit. 2016-08-10]. Dostupné z: `http://networkstatic.net/openflow-proactive-vs-reactive-flows/`.
- SDN Controller Comparison Part 2: Open Source SDN Controllers. *SDxCentral* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/`.
- SDN layers [obrázek]. *Software-Defined Networking (SDN) Definition* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/sdn-definition`.
- SDN Series Part Five: Floodlight, an OpenFlow Controller. RAO, Sridhar *The New Stack* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `http://thenewstack.io/sdn-series-part-v-floodlight`.
- Software-Defined Networking (SDN) Definition. *Open Networking Foundation* [online]. 2016 [cit. 2016-08-10]. Dostupné z: `https://www.opennetworking.org/sdn-resources/sdn-definition`.
- TP-LINK TL-WR1043ND [obrázek]. *Bezdrátový gigabitový N router 450 Mbit/s* [online]. 2016 [cit. 2016-09-10]. Dostupné z: `http://cz.tp-link.com/products/details/cat-9_TL-WR1043ND.html`.
- Understanding the SDN Architecture. *SDxCentral* [online]. 2016 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture`.
- What is a Floodlight Controller? *SDxCentral* [online]. 2015 [cit. 2016-08-10]. Dostupné z: `https://www.sdxcentral.com/sdn/`

definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/.

- What is OpenFlow? Definition and how it relates to SDN. *SDxCentral* [online]. 2016 [cit. 2016-08-10]. Dostupné z: https://www.sdxcentral.com/sdn/definitions/what-is-openflow/.

Seznam tabulek

Seznam obrázků

Seznam použitých zkratek

API – Application Programming Interface 17

APIC – Application Policy Infrastructure Controller 7

GUI – Graphical User Interface 6

JSON – JavaScript Object Notation 24

REST – Representational State Transfer 4

SDN – Software Defined Networking 3

VAN – Virtual Application Networks 7

Přílohy

A Zdrojové kódy

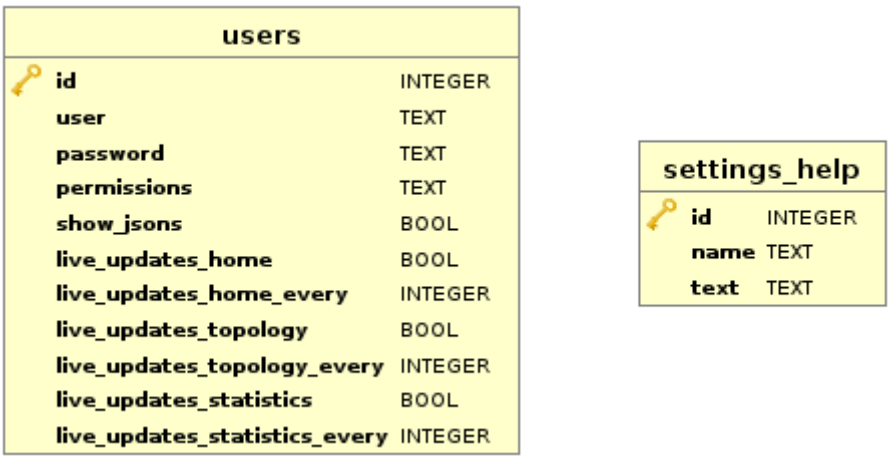
Všechny zdrojové kódy jsou, vzhledem k jejich rozsahu a formě, umístěny na CD nosiči v elektronické podobě.

- **Zdrojové kódy** – jsou rozděleny na backendovou a frontendovou část:
  - *backend* – složka zdrojové kódy/backend
  - *frontend* – složka zdrojové kódy/frontend
- **Kompletní balíček spustitelný jako služba** – složka aplikace

B Class diagram

Class diagram backendové části je umístěn, vzhledem k jeho velikosti, na CD v souboru class.png.

C Schéma databáze



D Testovací topologie

Všechny virtuální testovací topologie použité v této práci a zmíněné v kapitole Testování jsou umístěny na CD ve složce testy.

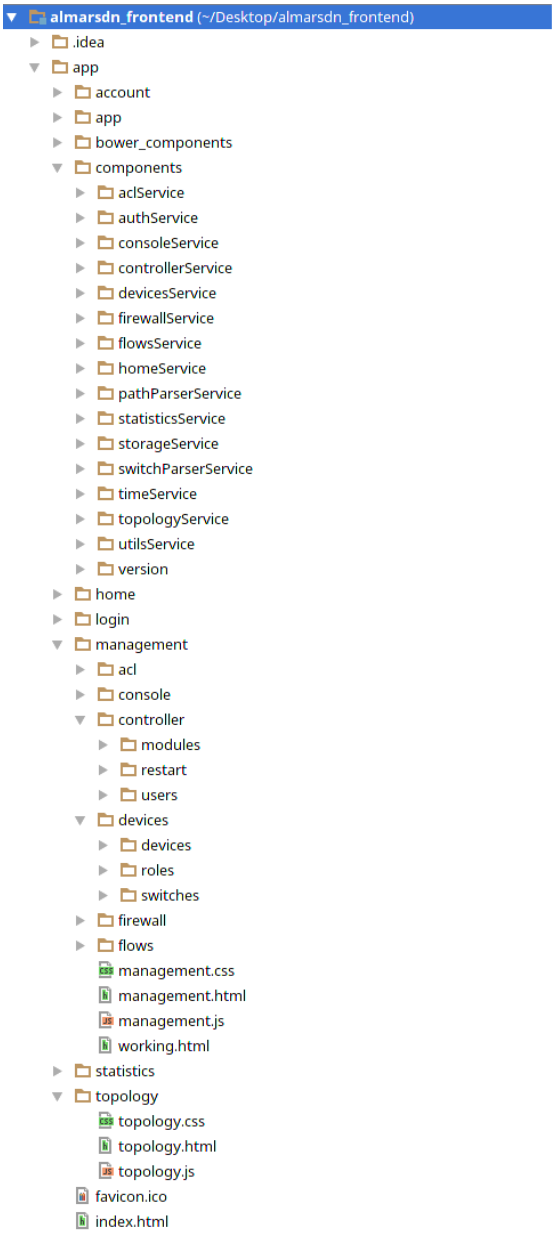
E Návod na zprovoznění

Kontrolér lze velmi jednoduše zprovoznit jako službu, která zajistí jeho bezproblémové ovládání, včetně restartu a to za pomoci následujících příkazů, nebo spustit jen jako aplikaci, která se po zavření konzole ukončí:

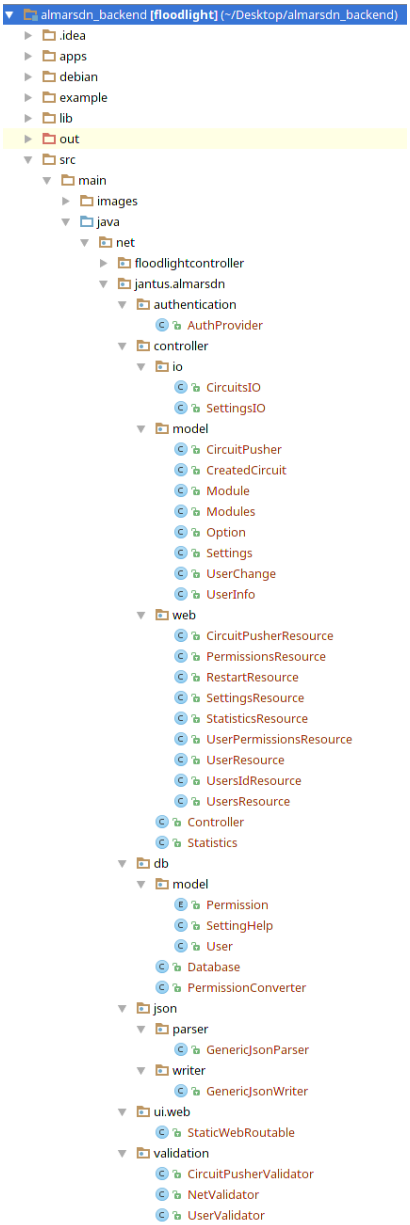
- *sudo mkdir /var/lib/floodlight*
- *sudo chmod 777 /var/lib/floodlight*
- *sudo mv aplikace /opt/almarsdn*
- *sudo chown -R root /opt/almarsdn*
- *sudo /opt/almarsdn/bin/almarsdn.sh install*
- *sudo service almarsdn start*

Pro spuštění jen v konzoli je třeba nahradit poslední dva příkazy tímto příkazem: *sudo /opt/almarsdn/bin/almarsdn.sh console*. Kontrolér je následně ve výchozím stavu dostupný na adrese `http://localhost:8080/almarsdn` a administrátorský účet má uživatelské jméno *floodlight* a heslo rovněž. Logy aplikace se pak nacházejí ve složce /opt/almarsdn/logs. Testování proběhlo na operačním systému Ubuntu 16.0.4.1 LTS 64-bit s OpenJDK-JRE Javou verze 7.

F Struktura projektu - frontend



G Struktura projektu - backend



H Přehled implementovaných funkcionalit

Zde je uveden kompletní souhrn implementovaných funcionalit na frontendové části.

H.1 Stránka HOME

- dočasné zapnutí/vypnutí aktualizací v reálném čase
- základní informace o kontroléru (počet přepínačů, hostů, linek, karanténních portů, využití paměti, stav kontroléru, stav firewallu, stav sběru přenosových statistik, doba běhu kontroléru)
- údaje o připojených přepínačích
- seznam flow tabulek

H.2 Stránka TOPOLOGY

- dočasné zapnutí/vypnutí aktualizací v reálném čase
- zobrazení topologie připojené sítě, včetně informací o daných zařízeních
- hledání cesty v síti a její zobrazení v grafu topologie

H.3 Stránka MANAGEMENT

Karta Flows

- přehled aktuálních Flow záznamů pro každý přepínač
- přehled aktuálních CircuitFlows
- možnost odebrání Flow/CircuitFlow záznamů
- možnost přidání Flows/CircuitFlow záznamů

Karta Devices

- **Podkarta Switches**
  - informace o připojených přepínačích a jejich Flow záznamech
- **Podkarta Devices**
  - informace o připojených koncových zařízeních
- **Podkarta Roles**
  - role připojených přepínačů, možnost změny

Karta ACL

- přehled aktivních ACL pravidel
- mazání a tvorba ACL pravidel

Karta Firewall

- přehled aktivních Firewall pravidel
- mazání a tvorba Firewall pravidel
- zapnutí/vypnutí Firewallu
- nastavení masky Firewallu

Karta Controller

- **Podkarta Settings**
  - přehled aktivních/aktivovatelných modulů včetně nápovědy, editovatelné
  - přehled nastavení kontroléru, editovatelné
- **Podkarta Restart**
  - restartování kontroléru

- **Podkarta Users**
  - přidání/úprava/smazání uživatelů, oprávněných k práci s kontrolérem a jejich práv

Karta Console

- práce s kontrolérem pomocí JSON zpráv a REST API v jednodušší podobě

H.4 Stránka STATISTICS

- dočasné zapnutí/vypnutí aktualizací v reálném čase
- prohlížení dostupných statistik
- zapnutí/vypnutí sběru přenosových statistik a jejich prohlížení

H.5 Stránka MY ACCOUNT

- zapnutí/vypnutí aktualizací v reálném čase, zvlášť pro stránky HOME, TOPOLOGY a STATISTICS
- nastavení intervalu těchto aktualizací
- zapnutí/vypnutí zobrazování sestavených JSON objektů, posílaných na kontrolér