VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. Vojtěch Piska

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VÝVOJ APLIKACÍ PRO SOFTWAROVĚ DEFINOVANÉ SÍTĚ
APPLICATIONS DEVELOPMENT FOR SOFTWARE DEFINED NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Vojtěch Piska

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Vladislav Škorpil, CSc.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Vojtěch Piska ***ID:*** 136039
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Vývoj aplikací pro softwarově definované sítě

POKYNY PRO VYPRACOVÁNÍ:

Softwarově definované sítě (SDN) jsou moderním trendem v telekomunikacích. Použijte obraz operačního systému (Allinone SDN App Development Starter VM) pro verzi 64bit nebo 32bit pro Virtual Box. Tento systém obsahuje soubor řešení, jako jsou kontrolery, přepínače a nástroje pro vývoj a testování SDN sítí. Detailně prozkoumejte a popište danou problematiku SDN a vytvořte na základě získaných znalostí postup pro vývoj vlastních aplikací v rámci obsažených nástrojů SDN. Uvažujte zejména využití nástrojů MININET a RYU. Vytvořte také soubor minimálně tří počítačových cvičení včetně vzorových protokolů.

DOPORUČENÁ LITERATURA:

[1] SDN Hub. http://sdnhub.org/

[2] RYU Controller Tutorial. http://sdnhub.org/tutorials/ryu/

[3] PUŽMANOVÁ,R. Moderní komunikační sítě A-Z. Computer Press, Brno 2007.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: doc. Ing. Vladislav Škorpil, CSc.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

V této diplomové práci je popsán nový typ síťové architektury také znám jako softwarově definované sítě. V první části práce jsou vysvětleny a popsány úlohy jednotlivých vrstev architektury. Jsou zde také zhodnoceny výhody a nevýhody těchto sítí. V další části je rozebrán protokol OpenFlow, který dovoluje kontroléru komunikovat s podřízenými hardwarovými prvky. V poslední části je pak návrh laboratorních úloh, které demonstrují technologii SDN.

KLÍČOVÁ SLOVA

softwarově definovaná síť, OpenFlow, SDN, kontrolér, Mininet

ABSTRACT

In this diploma thesis is described new network architecture also known as software defined networks. In first part of work are explained and described tasks of individual architecture layers. Work includes discussion about advantages and disadvantages of these networks. In next part is described OpenFlow protocol which allows to controller communicate with underlying hardware devices. Last part contains proposal of laboratory excercises which demonstrate SDN technology.

KEYWORDS

Software Defined Network, OpenFlow, SDN, Controller, Mininet

PISKA, Vojtěch *Vývoj aplikací pro softwarově definované sítě.*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 103 s. Vedoucí práce byl doc. Ing. Vladislav Škorpil, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vývoj aplikací pro softwarově definované sítě.“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Vladislavu Škorpilovi, CSc. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

V dnešním rychle a dynamicky se rozvíjejícím světě patří informační odvětví k těm nejrychleji rostoucím vůbec. Přesto se určité základní principy informačních technologií, konkrétně ty, týkající se datových sítí až na pár výjimek a drobnějších inovací příliš nezměnily.

V dnešní používané síťové architektuře tak stále převládá hierarchický model se stromovou typologií obsahující decentralizované síťové prvky, jako jsou přepínače či směrovače s vlastní inteligencí. Tento přístup již však přestává stačit potřebám organizací, které potřebují pružně a hlavně rychle zavádět do provozu nové aplikace a služby. Jako možné řešení této situace se nabízí technologie softwarově definovaných sítí - SDN (Software Defined Network), která je označována jako další revoluční krok v oblasti síťové architektury.

Tato práce se zabývá rozborem nové architektury softwarově definovaných sítí. Jsou zde zmíněny výhody a nevýhody ve srovnání s tradičním pojetím síťové architektury. Jednotlivé části SDN jsou zde představeny a podrobně rozebrány. Dále se tato práce zaobírá otevřeným protokolem OpenFlow, který slouží ke komunikaci mezi kontrolérem, jako centrálním prvkem celé sítě a podřízenými síťovými prvky. Nachází se zde i část s informacemi týkajících se problematiky vývoje aplikací pro tyto softwarově definované sítě se zaměřením na kontrolér RYU. V další části jsou navrhnuty tři laboratorní úlohy na demonstrování schopností SDN.

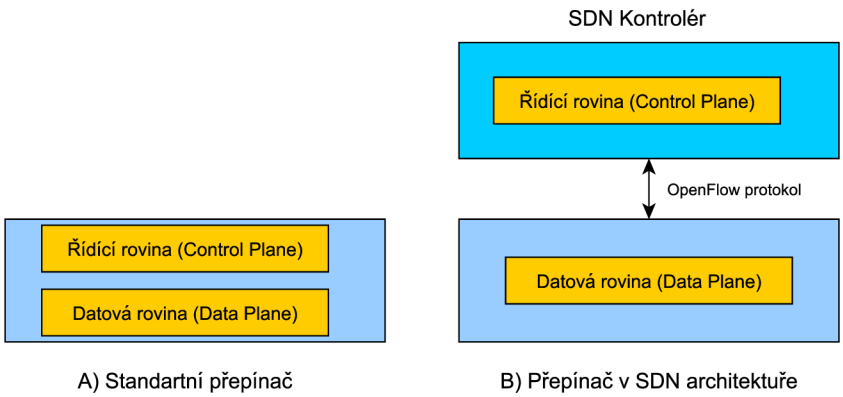
1 SOFTWAROVĚ DEFINOVANÉ SÍTĚ

1.1 Architektura softwarově definovaných sítí

Se současným trendem rychlého zavádění nových služeb a aplikací do provozu, již začínají být tradiční koncepce síťové architektury v mnoha ohledech značně limitující. Víceméně statická hierarchická stromová struktura současných sítí, obsahující prvky jako jsou směrovače či přepínače s vlastním decentralizovaným řízením kladnou velké nároky na celkovou flexibilitu, udržitelnost a především pak správu celé sítě.[11]

Tradiční přepínače a směrovače jsou často proprietárními systémy různých výrobců a jejich nasazování či následný rozvoj síťové infrastruktury se může stát obtížným. Jako jeden z největších problémů se jeví těsná vazba mezi mezi řídící (Control) a datovou rovinou (Data Plane), což má za následek, že rozhodnutí o průchodu dat v síti se musí vykonávat v modulech každého síťového prvku. V takovém to prostředí není zavedení nového protokolu či spuštění nové služby jednoduché a je nutné, aby bylo zavedeno napříč celou infrastrukturu. [11],[6]

Základní myšlenkou konceptu softwarově definovaných sítí je oddělení řídící roviny od datové (Control and Data Plane), kdy je tato řídící rovina přímo programovatelná. Toto oddělení řídící roviny a její přesunutí do dedikovaného zařízení (viz obr. 1.1), které bylo jinak pevně vázáno v každém jednom síťovém prvku nyní umožňuje jistou abstrakci a vytvořit tak globální pohled na obraz sítě. Tato vlastnost dovoluje spravovat zařízení různých výrobců skrze jedno unifikované rozhraní a proměnit tak síť složenou z proprietárních systémů mnoha výrobců do jednoho otevřeného.



Obr. 1.1: Srovnání přepínačů

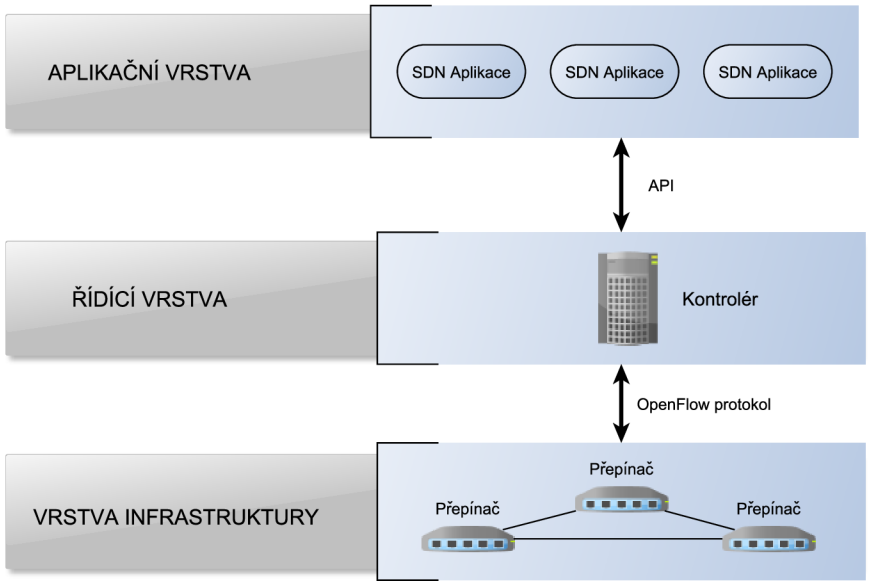
Řídící rovina - Control Plane

- Logika, která rozhoduje o tom jak zacházet s daným datovým tokem.
- Zahrnuje funkce pro správu, konfiguraci a výměnu směrovacích informací.
- Zajišťuje výměnu informací o typologii s ostatními zařízeními.
- Vytváří směrovací tabulku na základě směrovacích protokolů jako jsou (RIP[1], OSPF[2] či BGP[3])
- Funguje jako signalizace v síti.[6]

Datová rovina - Data Plane

- Také známá jako předávací rovina.
- Její úkol je předávat datové jednotky ze vstupního rozhraní na výstupní na základě informací získaných z řídící roviny.[6]

Vlastní architektura softwarově definovaných sítí se skládá ze tří vrstev. Jedná se o vrstvu infrastruktury, řídící vrstvu a vrstvu aplikační jak je nastíněno na obr. 1.2. Každá z těchto vrstev má svůj vlastní účel, přičemž tyto vrstvy mezi sebou obousměrně komunikují.



Obr. 1.2: Vrstvový model SDN sítě

[1]RIP - Routing Information Protocol
[2]OSPF - Open Shortest Path First
[3]BGP - Border Gateway Protocol

1.1.1 Vrstva infrastruktury

Vrstva infrastruktury je základním prvkem každé sítě, ať už se jedná o softwarově definovanou či síť s tradičním přístupem. Zahrnuje v sobě hardwarové prvky jako jsou přepínače či směrovače, které si dokáží mezi sebou vyměňovat datové jednotky. V tradičních sítích se nevyměňují pouze datové jednotky obsahující uživatelské informace, ale také informace o protokolech sloužící k řízení a správě sítě jako jsou pakety s informacemi o směrovacích protokolech (RIP, OSPF, BGP), ICMP[4] zprávy zjišťující dostupnost stanic v síti a podobně. Jednotlivé implementace těchto protokolů se mohou u různých výrobců síťových prvků drobně lišit což omezuje případnou flexibilitu, rozšiřitelnost a škálovatelnost sítě. Přímo úměrně se pak zvyšují nároky na obsluhu a údržbu těchto zařízení.

Řídící rovina, která by byla tradičně obsažena v této vrstvě infrastruktury a vázaná společně s datovou rovinou v jednom síťovém prvku (přepínač, směrovač) je zde oddělena a realizována softwarově ve zvláštním prvku nazývajícím se kontrolér, který pracuje na řídící vrstvě SDN architektury. Síťové uzly se tak stanou méně inteligentními a závislé na instrukcích přicházející z kontroléru. S kontrolérem pak komunikují jednotlivé síťové prvky skrze TCP[5] komunikační kanál, jehož obsah je zpravidla šifrován technologií TLS[6]. Tento komunikační kanál mezi kontrolérem a podřízeným síťovým prvkem se označuje jako takzvané „jižní rozhraní“. Protokol, kterým tyto podřízení prvky s kontrolérem komunikují se nazývá OpenFlow a je jedním ze základních prvků SDN sítí. Více o tomto protokolu bude zmíněno později v kapitole 2.[11],[12]

1.1.2 Řídící vrstva

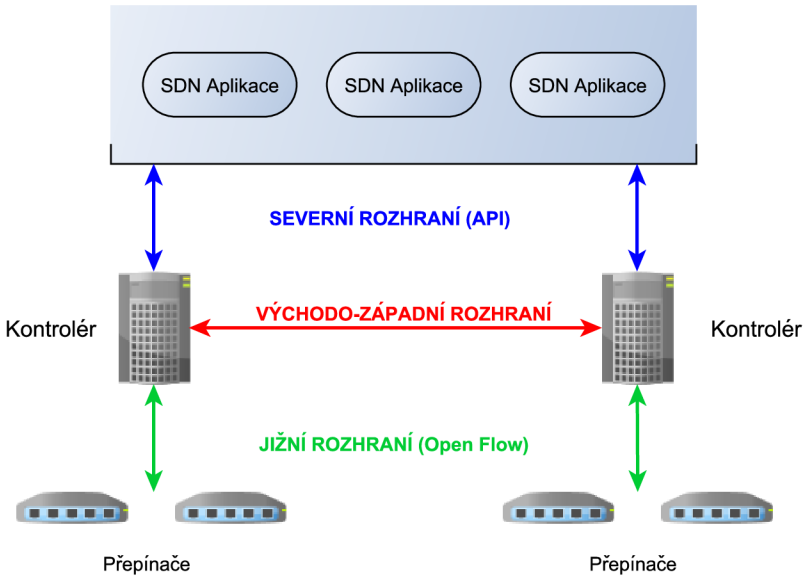
Řídící vrstva je prostřední vrstvou architektury SDN a zprostředkovává komunikaci na jedné straně mezi hardwarovými prvky infrastrukturní vrstvy a na straně druhé mezi aplikacemi bežící v aplikační vrstvě. Je tak klíčovou komponentou, která odlišuje softwarově definované sítě od sítí s tradičním přístupem.

Nejdůležitějším prvkem řídící vrstvy je kontrolér. V tomto prvku se soustřeďuje veškerá inteligence sítě. Kontrolér, který je realizován softwarově udržuje přehled nad logickou typologií sítě a má globální pohled na jednotlivá zařízení, která jsou k němu připojena. V důsledku toho se celá síť může navenek jevit jako jeden logický přepínač. Software kontroléru může běžet na obyčejném počítači či serveru. Daleko častěji se pro něj ovšem používají dedikovaná zařízení obr. 4.3, která jsou pro tyto účely daleko lépe optimalizována.

[4]ICMP - Internet Control Message Protocol
[5]TCP - Transmission Control Protocol
[6]TLS - Transport Layer Security

Obr. 1.3: Kontrolér ONEC-A-600 pro SDN

Kontrolér přebírá funkci řídící roviny (Control Plane), ze zařízeních ležící pod ním. Rozhoduje tak, jakým způsobem bude s datovými toky nakládat na základě aplikací, které běží nad ním v aplikační rovině. Kontrolér tak pracuje v několika rovinách viz obr. 1.4. Zpracovává požadavky přicházející od síťových prvků, přes takzvané „jižní rozhraní“, komunikuje s aplikacemi, které říkají kontroléru jakým způsobem by měl podřízenou síť řídit přes takzvané „severní rozhraní“. Jelikož má kontrolér svoji nezastupitelnou funkci, může se jevit jako velmi zranitelná část architektury v případě jeho selhání. Proto je zde na místě uvažovat nad jistým způsobem redundance a o možnosti přidat do sítě další kontrolér, který by v případě selhání přebral funkci hlavního kontroléru. Ten by v tomto případě zajišťoval komunikaci zařízení s aplikacemi. Proto je zde definováno ještě takzvané „východo-západní rozhraní“, kterým komunikují kontroléry mezi sebou navzájem.[11][5]



Obr. 1.4: Komunikační rozhranní SDN kontroléru

1.1.3 Aplikační vrstva

Aplikační vrstva je nejvyšší a nejinteligentnější vrstvou celé SDN architektury. Zahrnuje v sobě 4. – 7. vrstvu referenčního ISO/OSI[7] modelu. Aplikační vrstvu v kontextu softwarově definovaných sítí tvoří aplikace. Tyto aplikace jsou programy napsané ve vyšších programovacích jazycích jako jsou Python, JAVA či C++, které explicitně a přímo komunikují s prostředky v síti a ovlivňují její chování. Využívají přitom abstraktního pohledu na síť, který jim poskytuje kontrolér. Toto se děje skrze takzvané „severní rozhraní“, kterým tyto aplikace komunikují s kontrolérem a kontrolér pak skrze „jižní rozhraní“ řídí činnost jednotlivých hardwarových prvků v síti jako jsou přepínače či směrovače.[8],[12]

Aplikace pro softwarově definované sítě mohou mít rozličné funkce a na základě toho mohou:

- Monitorovat a spravovat síťový provoz
- Konfigurovat jednotlivá zařízení
- Vyžadovat a stanovovat přístupové politiky v síti
- Provádět vyvažování (tzv. Load Balancing)
- Zajišťovat požadovanou kvalitu služeb QoS[8]

1.2 Výhody softwarově definovaných sítí

Softwarově definované sítě založené na protokolu OpenFlow přináší mnohé výhody. Většina z nich je zapříčiněna oddělením řídící roviny od roviny datové a tím získat nový pohled na architekturu tradičních sítí. Díky tomu dnes umožňují SDN dynamicky a rychle zavádět nové aplikace, zjednodušovat správu zařízení a ještě lépe se přizpůsobovat potřebám trhu.

Mezi výhody těchto softwarově definovaných sítí patří:

- **Centralizovaná správa sítě:** SDN kontrolér nám dovoluje spravovat, kterékoliv zařízení, které je k němu připojeno a podporuje protokol OpenFlow, přičemž nezávisí na druhu zařízení či jeho výrobci. Díky tomu tak můžeme z jednoho centrálního místa spravovat konfigurace síťových prvků, kontrolovat a upravovat jejich činnost. Tím se práce s jednotlivými zařízeními může značně zefektivnit a také poskytnout menší prostor pro chybu při správě zařízení.

[7] ISO/OSI - International Standards Organization / Open System Interconnection
[8] QoS - Quality of Service

- **Snížení nároků na síťová zařízeni:** V tradiční síťové architektuře je inteligence ve smyslu algoritmů rozhodujících o nejlepší cestě sítí implementována přímo do jednotlivých zařízení rozprostřených v celé síti. To sice snižuje náchylnost na vznik chyby v jediném bodu, avšak tyto zařízení jsou zahlcovány množstvím protokolů, které jsou potřeba k vyjednávání, udržovaní, správě sousedství mezi zařízeními, výměně konzistentních směrovacích informací a celkově prodlužuje čas nutný ke konvergenci sítě.
  Oproti tomu v SDN se tato celá rozhodovací logika odehrává v centralizované části kontroléru, který pak jen podřízeným síťovým zařízením určuje komu a jakou cestou daná data předat. Toto ulehčuje výkon samostatným zařízením, u kterých odpadá nutnost složitými výpočty a vyměňováním těchto zpráv mezi zařízeními tyto informace pracně získávat.
- **Preciznější řízení sítě:** SDN sítě umožňují síťovým administrátorům mnohem lépe aplikovat politiky na základě uživatelů a jejich zařízení.
- **Zvýšení bezpečnosti sítě:** SDN má globální přehled nad děním v síti. Umožňuje zavést automatizované techniky v případě zjištění nestandardního chování. V případě přemístění, přidání či odebrání koncového zařízení, služby či aplikace odpadá nutnost individuálně překonfigurovat zařízení či upravovat jednotlivé záznamy v ACL[9]jednotlivých prvcích.
- **Rozšiřitelnost:** Jelikož je SDN softwarovou záležitostí, je tak snadné rozšířit možnosti stávající sítě skrze API[10], které kontroléry nabízejí (tzv. severní rozhraní viz obr. 1.4). Toto dovoluje tvorbu nových aplikací pro již stávající kontrolér.
- **Větší míra inovací:** Potřeby organizací pružně zavádět a testovat nové obchodní politiky jde ruku v ruce s technologií SDN, která umožňuje síťovým administrátorům spouštět a testovat nové služby v síti, aniž by se to jakkoliv dotklo produkční sítě.
- **Snížení ceny:** Nasazení architektury SDN může přinést novou funkcionalitu stávajícím síťovým zařízením, které budou přijímat instrukce z kontroléru za předpokladu, že podporují OpenFlow protokol. Mohou být nasazeny levnější a méně inteligentní prvky, jelikož je veškerá inteligence soustředěna v SDN kontroléru.[11],[13],[15]

[9]ACL - Access Control List
[10]API - Application Programming Interface

1.3 Nevýhody softwarově definovaných sítí

Jakožto každá nová technologie má kromě svých kladů také své zápory, s kterými je nutno počítat. Ani SDN nejsou výjimkou. Nejpalčivější problémy současných softwarově definovaných sítí se dají přisuzovat jejich rané vývojové fázi, kde nejsou ještě všechny problémy zcela vyřešeny.

Jedny z možných problémů softwarově definovaných sítí mohou být:

- **Různé softwarové implementace kontrolérů a aplikací:** Jedním z problémů je velký počet různých implementací softwarových kontrolérů a k nim přidružených aplikací. V současné době existují kolem desíti různých řešení, které jsou vyvíjeny různými výrobci. Některé z nich jsou jsou vyvíjeny jako opensource projekty s otevřeným zdrojovým kódem, jiné mají naopak kód skrytý a prodávají se jako samostatná komerční řešení. Velkou nevýhodou je pak vzájemná nekompatibilita mezi aplikacemi různých kontrolérů. Není totiž možné provozovat jednu aplikaci vytvořenou v jazyce Python pro jeden kontrolér a tu stejnou aplikaci nasazovat na kontrolér podporující aplikace psané v jazyce JAVA.
- **Problém centralizace:** Jelikož je v softwarově definovaných sítích inteligence jednotlivých síťových přenesena do kontroléru, který řídící rovinu centralizuje a rozhoduje tak o chování celé sítě, jsou nároky na spolehlivost tohoto prvku kritické. V případě jeho selhání, tak musí byt implementovány mechanismy, které tuto vzniklou situaci vyřeší.[3],[6]
- **Počet SDN kontrolérů:** Jak již bylo zmíněno v předešlém bodě, SND kontrolér je nejnáchylnější prvek sítě a v případě jeho nefunkčnosti může dojít k zhroucení celé sítě. Proto je na místě uvažovat o redundanci kontrolérů v síti, aby v případě selhání primárního kontroléru mohl síť dále obsluhovat sekundární. Tady ovšem vyvstává otázka jaký je optimální počet kontrolérů v síti, jelikož vytížení kontroléru je proměnlivé. Vynaložené náklady na provoz více SDN kontrolérů pak mohou být značně neefektivní.[7]
- **Motivace nasazovat softwarově definované sítě:** Asi jedním z největších problémů bránící rozšíření softwarově definovaných sítí je malá motivace současné sítě tradiční architektury transformovat do podoby softwarově definovaných sítí. Zejména pak nechuť dramaticky zasahovat do již fungující infrastruktury a neschopnost docenit výhod, které SDN přináší. Na druhou stranu se zvyšuje podíl společností, kteří přinášené výhody dokáží ocenit a při stavbě sítě se orientují tímto směrem. Jedná se pak zejména o datová centra, poskytovatelé internetu či větší firmy.

2 OPENFLOW PROTOKOL

2.1 Úvod do OpenFlow protokolu

OpenFlow protokol je základním kamenem softwarově definovaných sítí. Jedná se o standardizovaný otevřený protokol, který vznikl v roce 2006 jako univerzitní projekt na Stanfordské univerzitě v Kalifornii. Následně pak v roce 2011 byla ustanovena organizace Open Networking Foundation, která převzala záštitu nad vývojem dalších verzí OpenFlow protokolu. ONF[1] aktuálně sdružuje více než 50 firem, které se na vývoji podílí. Mezi nejznámější přispěvatelé patří společnosti jako Google, Facebook, Microsoft, Cisco, HP či Citrix. Hlavní myšlenkou při vývoji tohoto protokolu byla snaha překlenout uzavřené systémy výrobců síťového hardwaru a vytvořit sadu základní instrukcí, skrze které by tato zařízení mohla komunikovat s kontrolérem nezávisle na výrobci.[2]

Koncepce OpenFlow

Na začátek je potřeba si uvědomit, že samotný pojem SDN není to samé jako OpenFlow. SDN označuje celou myšlenku programovatelné softwarově definované sítě, zatímco OpenFlow je protokol, též označovaný jako takzvané „jižní rozhraní“ viz obr. 1.4, který komunikuje mezi *vrstvou infrastruktury* (jednotlivá síťová zařízení) a *řídící vrstvou*, která je charakterizovaná kontrolérem. Důležitým požadavkem je, aby jednotlivá zařízení podporovaly OpenFlow protokol.

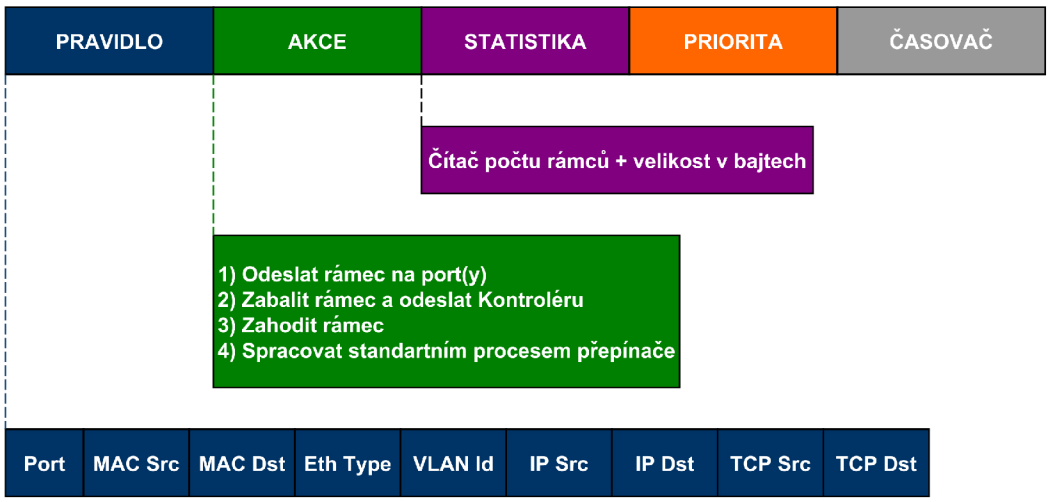
V klasickém přepínači je rychlé přepínání rámců (Data Plane) a vysokoúrovňové rozhodování o směrování (Control Plane) přítomné v jednom a tom samém zařízení. Oproti tomu OpenFlow přepínač odděluje funkce *řídící* a *datové roviny*, které jsou jinak pevně vázány v přepínači a přesouvá tuto řídící rovinu do kontroléru, který je řídícím prvek celé sítě. Kontrolér rozhoduje o zacházení s jednotlivými rámci na základě informací, která získává od *aplikační vrstvy*. Tyto instrukce pak skrze OF[2] předává přepínači a ten podle toho již sám upraví informace v *tabulce toků* své datové roviny.[11]

Podstatou celé funkčnosti této architektury je úprava *tabulek toků* přepínače z jednoho centrálního místa, které má povědomí o celkové situaci v síti. Kontrolér může záznamy v tabulkách toků jednotlivých přepínačů upravovat, přidávat či mazat. To vše na základě instrukcí, které přebírá z *aplikací*, které běží na kontroléru.

[1] ONF - Open Networking Foundation
[2] OF - OpenFlow

2.2 OpenFlow tabulky

V datové rovině každého přepínače se nachází OpenFlow tabulky. Tyto tabulky obsahují záznamy o tom, jakým způsobem přijaté rámce obsloužit. Přepínač obsahuje minimálně jednu takovou to tabulku, ale může jich být i více. Obsah těchto tabulek je spravován kontrolérem, ke kterému přepínač náleží. Strukturu OpenFlow tabulky můžete vidět na obr. 2.1.



Obr. 2.1: Struktura OpenFlow tabulky

Význam jednotlivých polí uvedených na obr. 2.1:

- **Pravidlo:** Při příchodu rámce zařízení zjišťuje informace obsažené v hlavičce datové jednotky a určuje, zda odpovídá některá část hlavičky stanovenému záznamu v tabulce. Prvek může být porovnáván na základě 1., 2., 3. či 4. vrstvy referenčního ISO/OSI modelu. V případě, že je toto pole *Pravidlo* v tabulce prázdné, pak provede akci pro všechny příchozí rámce.
- **Akce:** Toto pole specifikuje akce, které se mají provést v případě, že je nalezená shoda. Prvek může rámec odeslat na příslušný port, odeslat kontroléru, zpracovat standardním způsobem pomocí lokální řídící roviny popřípadě rámec zahodit. Je možnost zřetězit více akcí. Například odeslat rámec na daný port a navíc jej odeslat kontroléru.
- **Statistika:** V tomto poli se inkrementuje čítač rámců obsloužených dle daného pravidla a celková velikost těchto rámců v bajtech.
- **Priorita:** V tomto poli je uvedená priorita jednotlivých záznamů tabulky toků. V případě, že přijatý rámec vyhovuje více než jednomu pravidlu je zpracován podle záznamu s vyšší prioritou.

- **Časovač:** Záznam v tabulce toků může být časově omezen. V případě, že časovač u některého záznamu vyprší, je tento záznam z tabulky toků vymazán.

Zpracování datové jednotky

Na obr. 2.2 můžeme vidět jakým způsobem jsou příchozí datové jednotky v OpenFlow prvcích zpracovávány.

Zpracování příchozí datové jednotky začíná tím, že je z ní extrahována hlavička, která je následně porovnávána s jednotlivými záznamy v tabulce toků odshora dolů. Zpracování začíná v *Tabulce 0*. Pokud je zde nalezena shoda s některým záznamem v této tabulce, tak se zvýší hodnota čítače u tohoto záznamu a dané datové jednotce je přiřazena odpovídající akce. V případě, že není potřeba tuto datovou jednotku dále zpracovávat je následně jí přiřazená akce vykonána.

Jiná situace nastává, když je datové jednotce přiřazená akce na zpracování v jiné tabulce. Pak je datová jednotka odeslána na zpracování buď přímo do určité tabulky například *Tabulka 5* nebo je v postupném iteračním procesu testována. Zde je opět hledána shoda s jednotlivými záznamy tabulky toků. V případě, že není shoda nalezena je tato datová jednotka v závislosti na konfiguraci zahozena či odeslána do kontroléru pomocí zprávy *Packet-In*.[10]

Obr. 2.2: Zpracování datové jednotky OpenFlow prvkem

Režimy správy tabulek toků

Kontrolér může instalovat tabulky toků do jednotlivých přepínačů ve třech režimech. Každý z nich má svá daná specifika. Zvolený režim závisí na preferencích síťových administrátorů a konkrétním typu sítě.

Tyto režimy jsou:

- **Reaktivní režim:** Přepínač v tomto režimu funguje tím způsobem, že OpenFlow tabulky jsou ze začátku prázdné a neobsahují žádné záznamy o tom jak s daným rámcem zacházet. Při přijetí rámce pak přepínač vytvoří zprávu typu *Packet-In*, ve které se dotáže kontroléru jak s tímto rámcem zacházet. Ten mu požadované informace nejen zašle, ale zajistí naprogramování OpenFlow tabulky tak, že pro další příchozí rámec stejného typu již bude přepínač vědět jak s rámcem zacházet a nebude se již muset znovu dotazovat kontroléru. Tento režim dává kontroléru dobrý přehled o dění v síti. Mírnou nevýhodou může být vyšší zpoždění v případě rozsáhlejších sítí.
- **Proaktivní režim:** Tento režim funguje tak, že kontrolér dopředu nastaví všechny záznamy v OpenFlow tabulce pro všechny možné situace na základě svých informací. OpenFlow tabulka tak při příchodu rámce již ví, jak s daným rámcem naložit.
- **Hybridní režim:** Tento režim umožňuje předchozí zmiňované techniky kombinovat a dokonce využít i využít lokální *řídící rovinu* přepínače a to třeba v případě, že OpenFlow tabulka neobsahuje žádný záznam, který by odpovídal přijatému rámci. Přepínač se tak může rozhodnout, že jeho zpracování zajistí klasickým způsobem.[14]

2.3 Typy zpráv

V OpenFlow protokolu se můžeme setkat se třemi typy zpráv. Jedná se o zprávy *controller-to-switch*, *asymetrické* a *symetrické zprávy*. Každá z nich má ještě několik podtypů zpráv. Controller-to-switch zprávy jsou iniciovány kontrolérem a slouží k správě či zjišťování informací o stavu přepínače. Asymetrické zprávy jsou naopak iniciovány jednotlivými přepínači. Jejich účel je aktualizování síťových informací a jejich zasílání kontroléru. Symetrické zprávy oproti tomu mohou iniciovat obě dvě strany bez předchozím požadavků na jejich odeslání. [10]

2.3.1 Controller to Switch zprávy

Jak již bylo zmíněno dříve jedná se o zprávy, které jsou odesílány kontroléry směrem k přepínačům. Tyto zprávy jsou ve většině případů typu dotaz-odpověď. Pomocí nich může kontrolér zjišťovat stav podřízených zařízení popřípadě upravovat jejich stávající konfiguraci kdykoliv je potřeba.

Typy controller-to-switch zpráv:

- **Features:** Odesláním této zprávy může kontrolér požadovat identifikaci přepínače a informace o jeho základních schopnostech. Přepínač pak musí reagovat na zprávu odpovědí *Features reply*, ve které specifikuje svoji identitu a vyžadované základní informace. Výměna těchto zpráv běžně probíhá při vytváření OpenFlow komunikačního kanálu mezi kontrolérem a přepínačem.
- **Configuration:** Kontrolér je schopen pomocí těchto zpráv nastavovat a dotazovat se na určité parametry přepínače. V tomto případě pak přepínač kontroléru na jeho dotaz odpoví.
- **Modify-State:** *Modify-State* zprávy jsou posílány kontrolérem a spravují stav přepínače. Těmito zprávami je možno přidávat, mazat či modifikovat záznamy v tabulkách toků jednotlivých zařízení, popřípadě nastavovat chování jejich portů.
- **Read-State:** Tyto zprávy jsou používány kontrolérem ke sběru rozličných informací z připojených přepínačů. Takovými informacemi mohou být například aktuální informace o konfiguraci zařízení, informace o jeho schopnostech či shromažďované statistiky.
- **Packet-Out:** Zprávy tohoto typu jsou používány k reakci na příchozí zprávy typu *Packet-In*, které se odesílá přepínač do kontroléru v případě, že neví jak s přijatým rámcem v přepínači naložit. Zpráva *Packet-Out* pak musí obsahovat celý rámec, který přišel kontroléru ve zprávě *Packet-In* popřípadě jeho identifikátor odkazující na rámec uložený v paměti přepínače. Obsahem těchto *Packet-Out* zpráv je pak posloupnost akcí, které mají být aplikovány na dotazovaný rámec ve specifikovaném pořadí. Prázdný obsah rámce pak znamená pokyn k zahození rámce.
- **Barrier:** Žádosti či odpovědi na zprávy typu *Barrier* jsou používány kontrolérem pro přijímání notifikací o dokončených operacích.
- **Role-Request:** Tyto typy zpráv jsou používány především v případě, že je přepínač připojen k více kontrolérům a je třeba odlišit informace z jednotlivých kontrolérů. Kontrolér pak skrze tyto zprávy nastavuje svou roli a identifikuje OpenFlow komunikační kanály.
- **Asynchronous-Configuration:** Zprávy tohoto typu slouží kontrolérům k na-

stavené filtrování asynchronních zpráv, které chce přijímat na svém OpenFlow komunikačním kanálu. Toto může být užitečné v případě připojení přepínače k více SDN kontrolérům.[10],[16]

2.3.2 Asynchronní zprávy

Na rozdíl od zpráv typu *controller-to-switch*, které jsou iniciovány kontrolérem, tak *asynchronní zprávy* v tomto případě odesílá přepínač. V těchto zprávách se dotazuje kontroléru jak zacházet s danými rámci, informuje kontrolér o stavu svých portů a podobně.

Typy asynchronních zpráv:

- **Packet-in:** Tato zpráva je slouží k přenesení kontroly rámce z přepínače do kontroléru.
  Zpráva může být generována ve dvou případech a to:
  - Když přepínač obdrží rámec a ve své tabulce toků nenajde shodu záhlaví rámce s žádným pravidlem. Pak tento rámec odešle pro kontrolu kontroléru, který určí jak se má s daným rámcem zacházet. Kontrolér nato vygeneruje zprávu typu *Packet-Out* a zašle ji zpět přepínači.
  - Když je přepínač implicitně přednastaven, aby veškeré rámce, které příjme byly odesílány na kontrolér, který o jejich řízení rozhodne. Načež opět vygeneruje zprávu *Packet-Out* s informacemi jak s rámcem naložit.

  V případě, že přepínač disponuje dodatečnou pamětí, nemusí se kontroléru odesílat celý rámec. Přepínač si uloží rámec do paměti a vytvoří *Packet-In* zprávu, s identifikátorem odkazujícím na daný rámec v paměti. V této zprávě ze tak přepošle jen hlavička rámce, podle níž kontrolér určí jak s rámcem naložit. Následně pak vygeneruje *Packet-Out* zprávu s informací jak s rámcem naložit společně s identifikátorem obsaženým v *Packet-In* zprávě. Pomocí tohoto identifikátoru se pak vybere adekvátní rámec z paměti přepínače a ten je pak obsloužen na základě informací obsažených v *Packet-Out* zprávě vygenerované kontrolérem.
- **Flow-Removed:** Informuje kontrolér o odebrání záznamu z tabulky toků. Tyto zprávy jsou generovány jako odpověď v případě, že kontrolér přikáže přepínači nějaký záznam z této tabulky vymazat, popřípadě když vyprší platnost záznamu v tabulce toků. Informuje tímto přepínač a ten může v případě potřeby vygenerovat novou zprávu *Modify-State*, která bude obsahovat nový záznam pro vložení do tabulky toků.
- **Port-Status:** Zpráva informující kontrolér o stavech jednotlivých portů přepínače. Odesílá zprávy obsahující konfiguraci portů a informaci o tom, když

nějaký z nich přejde do režimu „Down“.

- **Role-Status:** V této zprávě odesílá informace o své roli popřípadě o její změně. Jsou odesílány například v případě, že je přepínač připojen k více kontrolérům a ty během své existence změní své role.
- **Controller-Status:** Informuje kontrolér o změnách stavu OpenFlow kanálu, který je mezi přepínačem a kontrolérem. Tato informace může být užitečná při řešení chyb v případě zjištění, že tento kanál mezi nimi nefunguje standardně.[10],[16]

2.3.3 Synchronní zprávy

Zprávy tohoto typu mohou být odeslány jak přepínači tak kontroléry. Zprávy toho typu se dají označovat jako servisní. Oznamují protistranám stav kanálu mezi nimi, popřípadě pak vzniklé problémy.

Typy synchronních zpráv:

- **Hello:** *Hello* zprávy se vyměňují mezi kontrolérem a přepínačem při vytváření spojení mezi nimi.
- **Echo:** Zprávy *Echo* jsou typu dotaz/odpověď a jsou posílány oběma stranami, přičemž na každý dotaz musí být vygenerovaná odpověď. Hlavní účel těchto zpráv je ověřovat, zda je spojení mezi kontrolérem a přepínačem aktivní. Mohou být, ale také využity pro měření zpoždění či šířky pásma.
- **Error:** Chybové zprávy jsou používány k oznámení protistraně o vzniklém problému. Nejčastěji jsou používány přepínači na oznamování problémů kontroléru.
- **Experimenter:** Tyto zprávy v současné době nemají zatím stanoven přesný účel. Jsou tak vesměs používány pro testovací účely. Do budoucna se počítá s jejich implementací ve vyšších verzích OpenFlow protokolu.[10],[16]

2.4 Verze protokolu

Jelikož OpenFlow vznikl jako univerzitní projekt, jeho počáteční verze nebyly řádně standardizovány. To vše se ovšem změnilo jakmile nad celým projektem převzala záštitu organizace Open Networking Foundation.

- **Verze 1.0** Verze 1.0 byla vydána v prosinci 2009. V této verzi byla definovaný role kontroléru, podřízených prvků a základní typy zpráv pomocí níž komunikovaly. Příchozí datové jednotky mohly být s tabulkou toků porovnávány na základě 1.–4. vrstvy referenčního ISO/OSI modelu. Prvky v této verzi obsahovaly pouze jednu tabulku toků. Tato verze obsahovala taky základní typy čítačů a statistik, které dávaly kontroléru přehled o dění v síti.

- **Verze 1.1** Tato verze byla vydána v únoru roku 2011. Největší novinkou oproti verzi předešlé je podpora více tabulek toků. Prvek tak v sobě může mít tabulky specializované na konkrétní vrstvy ISO/OSI modelu. První tabulka tak například může datové jednotky porovnávat na základě informací z 2. vrstvy, jiná tabulka může porovnávat informace v hlavičce týkající se 3. vrstvy. Další novinkou je podpora práce s MPLS[3]značkami. Zařízení tak mohou hledat shody v tabulkách toků i na základě těchto značek, popřípadě s nimi pak pracovat. Mohou je přidávat, odebírat, přeznačovat a podobně.
- **Verze 1.2** Ještě téhož roku 2011 byla v prosinci vydána verze 1.2. Ta přinesla podporu nových protokolů zejména pak IPv6[4]. Datová jednotka mohla být takto porovnávána podle nových polí. Další velkou novinkou byla podpora propojení OpenFlow prvku k více než jednomu kontroléru. Tímto se vyřešil problém chyby jediného bodu, který by v případě selhání kontroléru vedl k problému v celé síti. Nyní může v síti operovat více kontrolérů. Jeden jako primární, ostatní jako podřízené, které ale v případě chyby hlavního kontroléru mohou převzít jeho funkci.
- **Verze 1.3** Další verze s označením 1.3 přišla v dubnu 2012. Tato verze přinesla drobná vylepšení protokolu IPv6 jako jsou rozšířené hlavičky, podporu ESP[5] či podporu hop-by-hop pole v rozšířené hlavičce IPv6 paketu. Přibyla také podpora tunelování. OpenFlow ve verzi 1.3 je zatím nejvíce podporován výrobci zařízení.
- **Verze 1.4** Další verze přišla na scénu v srpnu roku 2013. Zahrnovala podporu pro optické porty a OpenFlow mohlo být tedy nasazeno i do vysokorychlostních sítí. Další významnou novinkou bylo pak sdružování zpráv, které do té doby posílal kontrolér jednotlivě. Jednou z dalších novinek pak bylo funkce na upozornění kontroléru, že dochází kapacita v tabulce toků ještě před jejím vyčerpáním. V předešlých verzích se tak stávalo, že se kontrolér pokoušel vložit do tabulky nový záznam, který se zde již nevešel a prvky tak na tuto událost reagovaly až po vzniklé chybě.
- **Verze 1.5** V tuto chvíli nejnovější OpenFlow specifikace, která byla publikována v prosinci 2014. Zaměřila se především na opravu chyb verzí předešlých, ale přibyly i nové funkcionality. Porovnávání datových jednotek na základě TCP příznaků, sdružované zprávy, které se objevily ve specifikaci 1.4 dostaly možnost odesílání na základě časového naplánování. Přibyly i nové možnosti měření statistik.[1],[10]

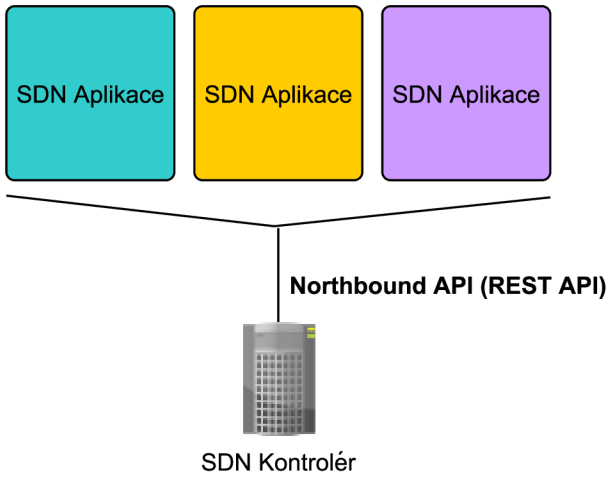
[3]MPLS - Multiprotocol Label Switching
[4]IPv6 - Internet Protocol version 6
[5]ESP - Encrypted Security Payload

Vývoj OpenFlow a jeho standardizace dosahuje velmi svižného tempa. S každou další vydanou specifikací přibývají nové funkce a podpora nových protokolů. Bohužel se implementace těchto nových standardizací tak rychle neprojevuje u výrobců zařízení, kteří z větší části podporují verze 1.0 a 1.3.

3 VÝVOJ APLIKACÍ PRO SDN SÍTĚ

Jak již bylo naznačeno v předešlých kapitolách aplikační vrstva softwarově definovaných sítí je stěžejním prvkem této infrastruktury. Zahrnuje v sobě 4. - 7. vrstvu referenčního ISO/OSI modelu. Co tvoří aplikační vrstvu SDN jsou aplikace.

Aplikace jsou spuštěny na kontroléru a komunikují s ním přes takzvané „severní rozhraní“ (Northbound) viz obr. 3.1. Skrze toto API mohou získávat informace o celkové stavu sítě a ke kontroléru připojeným síťovým zařízením, které tyto informace zasílají kontroléru skrze „jižní rozhraní“ (SouthBound) protokolem OpenFlow. Na základě těchto informací mohou aplikace nejen shromažďovat statistky a informace o dění v sítí, ale mohou skrze toto API programovat a řídit její chování.



Obr. 3.1: Rozhraní mezi SDN aplikacemi a kontrolérem

Při vývoji aplikací pro SDN kontroléry je největší překážkou různé softwarové implementace kontrolérů. V současné době existuje okolo desíti různých řešení, které jsou vyvíjeny buď jako volně šiřitelné opensource projekty, či naopak jako samostatná komerční řešení. Různé kontroléry totiž definují svá vlastní API pro programování aplikací a může se stát, že konkrétní API, které aplikace vyžaduje není pro daný kontrolér k dispozici nebo je definováno odlišně.[15]

Vývoj SDN aplikací je také do jisté míry znesnadněn nekompatibilitou mezi aplikacemi pro různé kontroléry. Není totiž možné provozovat jeden typ aplikace psanou v jazyce Python na kontroléru, který používá aplikace psané v jazyce JAVA.

Kontroléry v zásadě nabízí dva způsoby programování aplikací.

REST API

Externí forma využívá REST[1] API rozhraní, tedy webových volání, které lze realizovat z externích aplikací. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům. Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty).

Ideální je pro úkoly, které nevyžadují reakci sítě v reálném čase ani asynchronní události. Příklady zahrnují proaktivní programování toků (pravidel pro provoz), získávání informací (statistiky, koncové stanice, linky, topologie), vkládání služeb, statické rozdělování zátěže či řízení kvality služby.[4]

Charakteristiky REST rozhraní:

- Všechny zdroje mají svůj jedinečný identifikátor URI[2]
- REST implementuje čtyři základní metody, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.

V případě, že tedy budeme potřebovat zjistit a do naší aplikace implementovat výpis tabulky toků z přepínače `s1` můžeme to udělat skrze REST volání.

```
$ curl -X GET http://127.0.0.1:6363/stats/flow/1
```

Kde `GET` značí, že chceme informace ze zdroje získat, `http://127.0.0.1:6363/stats/flow/1` pak symbolizuje URI adresu skládající se z IP adresy a portu, na kterém je dosažitelný kontrolér, část `stats/flow` pak definuje danou API metodu kontroléru, který je následován parametrem `1`, který značí, že požadujeme informace z přepínače `s1`.

Odpověď na toto REST volání pak bude níže zobrazené informace ve struktuře JSON[3]. Jak je možné vidět, informace obsahují informace o záznamech v tabulce toků přepínače `s1`.

[1] REST - Representational State Transfer
[2] URI - Uniform Resource Identifier
[3] JSON - JavaScript Object Notation

```
{
  "1": [
    {
      "table_id": 0,
      "duration_sec": 2,
      "priority": 100,
      "idle_timeout": 0,
      "hard_timeout": 0,
      "flags": 1,
      "cookie": 1,
      "packet_count": 40,
      "byte_count": 4080,
      "match": {
        "in_port": 1
      },
      "actions": [
        "OUTPUT:2"
      ]
    }
  ]
}
```

V případě, že bychom chtěli místo zobrazení tabulky toků, data externí aplikace přidat, můžeme využít následujícího REST volání, díky kterému kontrolér upraví tabulku toků přepínače `s1` tak, že cokoliv příjde na vstupní port 1, bude zpracováno instrukcemi v následující tabulce číslo 1 (první tabulka má číslo 0).

```
$ curl -X POST -d '{
    "dpid": 1,
    "priority": 22222,
    "match":{
        "in_port":1
    },
    "actions":[
        {
            "type":"GOTO_TABLE",
            "table_id": 1
        }
    ]
 }' http://127.0.0.1:6363/stats/flowentry/add
```

Nativní programování

Druhý typ programování je nativní a přináší možnost vytvoření modulu architektury kontroléru, který je možné za běhu nahrát do přímo do něj. Vyžaduje programování v binárním jazyce, který je s danou architekturou kontroléru kompatibilní.

V tomto případě budeme hovořit o kontroléru RYU, který je programován v jazyce Python. Zahrnuje všechny možnosti zmíněné výše (skrze REST API), dále pak umožňuje rozšířit REST API o další aplikačně specifická volání a reakce v reálném čase na události sítě včetně asynchronních situací. Aplikace tak může reagovat na zprávy `Packet_In`, generovat `FlowMod` či `Packet_Out` zprávy.[17]

Struktura SDN kontroléru RYU

- `app/` - Obsahuje soubor aplikací, které běží na kontroléru.
- `base/` - Obsahuje základní třídu pro běh RYU aplikací RyuApp v souboru `app_manager.py`.
- `controller/` - Obsahuje soubory nutné k obsluhování OpenFlow instrukcí jako jsou datové jednotky z přepínačů, generování toků, obsluha událostí či shromažďování statistik.
- `lib/` - Obsahuje kolekci knihoven, které jsou nutné k zpracování, dekódování záhlaví různých protokolů.
- `ofproto/` - Obsahuje informace specifické k zpracování rozličných verzí OpenFlow protokolu (verze 1.0, 1.2, 1.3 a 1.4)
- `topology/` - Obsahuje funkce, kterými kontrolér získává informace o topologii sítě (porty, linky).

Tvorba nové aplikace pro kontrolér RYU spočívá ve vytvoření podtřídy hlavní třídy RyuApp a nezbytné logiky pro zpracovávání síťových událostí. Je proto potřeba vytvořit nový soubor s příponou `*.py` například `Aplikace.py`, do které je třeba doplnit výše zmíněné náležitosti viz obsah souboru níže.

```
from ryu.base import app_manager

class Aplikace(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(Aplikace, self).__init__(*args, **kwargs)
```

Takto vytvořený soubor je již validní aplikací pro kontrolér RYU, přestože ještě neobsahuje žádnou logiku pro obsluhování přicházejících datových jednotek z připo-

jených přepínačů. Dalším krokem je tedy umožnit aplikaci přijímat tyto OpenFlow zprávy. K tomu nám poslouží implementace třídy `EventOFPPacketIn`.

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
```

První argument volá funkci pokaždé, když je přijata zpráva typu `packet_in`. Druhý argument pak informuje o stavu přepínače. Jakákoliv informace o přepínači a verzi protokolu pak může být získána následujícím kódem:

```
msg = ev.msg       # Objekt reprezentující strukturu zprávy packet_in
datapath = msg.datapath    # Identifikátor přepínače
ofproto = datapath.ofproto # Vrací verzi OpenFlow protokolu
```

Jakmile je tok datových jednotek realizován je možné informace z nich dekódovat. Toto umožníme importováním knihoven, které se nacházejí v adresáři `/ryu/lib`.

```
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
```

Následně je již možné z datových jednotek extrahovat informace z jejich záhlaví.

```
pkt = packet.Packet(msg.data)
eth = pkt.get_protocol(ethernet.ethernet)


dst = eth.dst # Cílová adresa rámce
src = eth.src # Zdrojová adresa
```

Nyní je již možné zpracovávat příchozí zprávy `Packet_In` a extrahovat z nich informace. Zbývá ještě vytvořit možnost generovat odpovědi na tyto zprávy `Packet_Out`. Kromě vytvoření odpovědi na příchozí `Packet_In` můžeme generovat zprávu typu `Flow_Mod`, která vloží do tabulky přepínače požadovaný záznam, aby věděl jak příští stejnou datovou jednotku zpracovat. Nejdříve je potřeba vytvořit pravidlo shody, kde `in_port` a `eth_dst` jsou extrahovány z `Packet_In` zprávy.[17]

```
msg = ev.msg
in_port = msg.match['in_port'] # Vstupní port


pkt = packet.Packet(msg.data)
eth = pkt.get_protocol(ethernet.ethernet)
dst = eth.dst  # Cílová linková adresa
match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
```

Následně pak vytvořit akci a instrukci načež vygenerovat a odeslat zprávu `Flow_Mod` díky ní bude toto pravidlo přidáno do tabulky toků na přepínači.

```
actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD)]
# Vytvoření akce

inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
# Aplikování instrukce

mod = parser.OFPFlowMod(datapath=datapath, priority=0, match=match,
 instructions=inst)
datapath.send_msg(mod)
# Vygenerování a odeslání Flow_Mod zprávy do přepínače
```

Tímto způsobem představeným výše je možné vyvíjet jednoduché aplikace pro SDN kontrolér RYU. Důležité je vždy zajistit příjem `Packet_In` zpráv, které odesílají přepínače kontroléru a následně na ně reagovat zprávami `Packet_Out` a `Flow_Mod`, které upraví tabulky toků v jednotlivých přepínačích.

4 NÁVRH LABORATORNÍCH ÚLOH

Součástí této práce je návrh a příprava laboratorních úloh pro demonstraci problematiky softwarově definovaných sítí. V rámci této práce byly připraveny celkem tři laboratorní úlohy, ve kterých studenti mohou dobře pochopit klíčové vlastnosti a možnosti této nové síťové architektury. Následně si je pak v těchto úlohách prakticky vyzkoušet.

Laboratorní úlohy jsou koncipovány tak, aby je bylo možné vypracovat během standardní laboratorní výuky o délce 2x 45 minut. Laboratorní úlohy jsou rozděleny na teoretickou a praktickou část.

V teoretické části se nachází informace k tématu laboratorní úlohy, které mají za cíl usnadnit a osvětlit následující úkoly v praktické části.

Praktická část je vždy rozdělena na dva úkoly, přičemž první z nich obsahuje postup řešení, kde se studenti pozvolna seznámí s řešením dané problematiky. Druhý úkol pak zpracovávají studenti samostatně již bez návodu, přičemž při řešení vychází z již předešlého úkolu či znalostí nabytých v předešlých cvičeních. Vyjímkou je pak první laboratorní úloha, kde se studenti v obou úkolech praktické části řídí postupem řešení. Tato úloha je brána jako seznamovací.

Na konci každé laboratorní úlohy studenti vypracují laboratorní protokol, v němž budou zodpovězeny kontrolní otázky ke každé úloze.

Součástí laboratorních úloh je pak dokumentace pro vyučujícího obsahující odpovědi na kontrolní otázky a vzorové konfigurační soubory s návodem.

4.1 Shrnutí laboratorní úlohy č. 1 – Úvod do softwarově definovaných sítí

V rámci první laboratorní úlohy budou mít studenti možnost seznámit se s koncepcí architektury softwarově definovaných sítí. V teoretické části jsou vysvětleny jednotlivé části vrstvového modelu architektury mezi níž patří vrstva infrastruktury, řídící a aplikační vrstva, které jsou dány do kontrastu s tradiční síťovou architekturou. Následně je v této části rozebrána úloha kontroléru, jakožto klíčového prvku. Neméně je zde kladen důraz na OpenFlow protokol, který tvoří komunikační kanál mezi kontrolérem a jemu podřízenými prvky. Je zde vysvětlena problematika OpenFlow zpráv, tabulek toků a samotného zpracování datových jednotek v zařízení.

V praktické části této laboratorní úlohy si studenti vyzkouší práci práci se síťovým emulátorem Mininet, který slouží k testování softwarově definovaných sítí.

První část je zaměřena na seznámení se s tímto prostředím a vytvořením jednoduché topologie, kde bude demonstrován proaktivní režim kontroléru. Studenti budou mít příležitost seznámit se s tabulkou toků a jejími úpravami.

Druhá část je pak zaměřena na demonstraci reaktivního režimu kontroléru a zachytáváním OpenFlow zpráv v protokolovém analyzátoru Wireshark. Studenti zde budou mít praktický pohled na dění v síti a nahlédnou podrobněji do struktury vyměňovaných informací.

Studenti po absolvování této laboratorní úlohy budou umět vysvětlit podstatu architektury softwarově definovaných sítí, znát její klíčové komponenty včetně protokolu OpenFlow. Získají také praktickou zkušenost se síťovým emulátorem Mininet, který bude využitý i v úlohách následujících. Na závěr laboratorní úlohy je připraveno několik kontrolních otázek, které prověří úroveň pochopení demonstrované látky. Součástí této laboratorní úlohy je pak část pro vyučujícího, kde nalezne soubor správných odpovědí na kontrolní otázky.

4.2 Laboratorní úloha č. 1 – Úvod do softwarově definovaných sítí

4.2.1 Cíl

Cílem této laboratorní úlohy by mělo být nahlédnutí do problematiky softwarově definovaných sítí – SDN[1]. Seznámíme se s jejich architekturou, protokolem OpenFlow a také se síťovým emulátorem Mininet, ve kterém budou všechny naše simulace prováděny.

4.2.2 Vybavení pracoviště

- 1x Stolní počítač s nainstalovaným virtualizačním softwarem VMware
- 1x Obraz disku s All-in-one SDN App Development Starter VM
- 1x Návod s laboratorní úlohou

4.2.3 Úkoly

1. Seznamte se s pojmem softwarově definovaná síť.
2. Nastudujte roli kontroléru a protokolu OpenFlow.
3. Seznamte se síťovým emulátorem Mininet a prozkoumejte jeho možnosti.

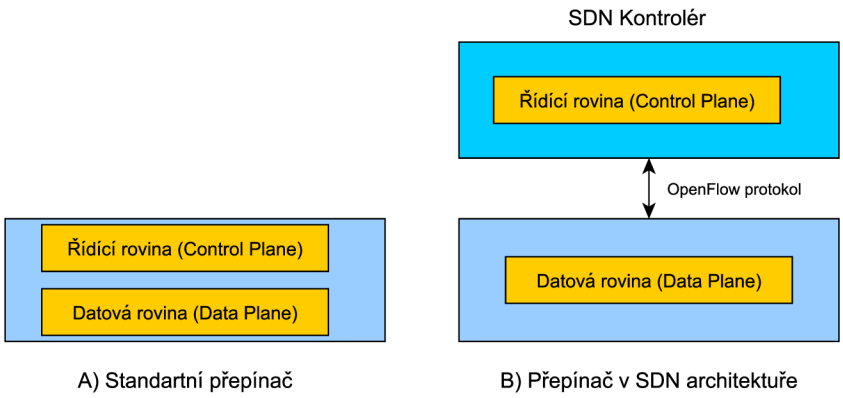
[1]SDN - Software Defined Networks

4. Dle zadání vytvořte SDN topologii a analyzujte pomocí analyzátoru Wireshark OpenFlow zprávy.
5. Odpovězte na kontrolní otázky.

4.2.4 Teoretický úvod

Softwarově definované sítě jsou relativně novým pojmem co se týče síťových technologií. Jejich vznik se datuje do roku 2009, ale k jejich velkému rozmachu dochází až v průběhu posledních několika let.

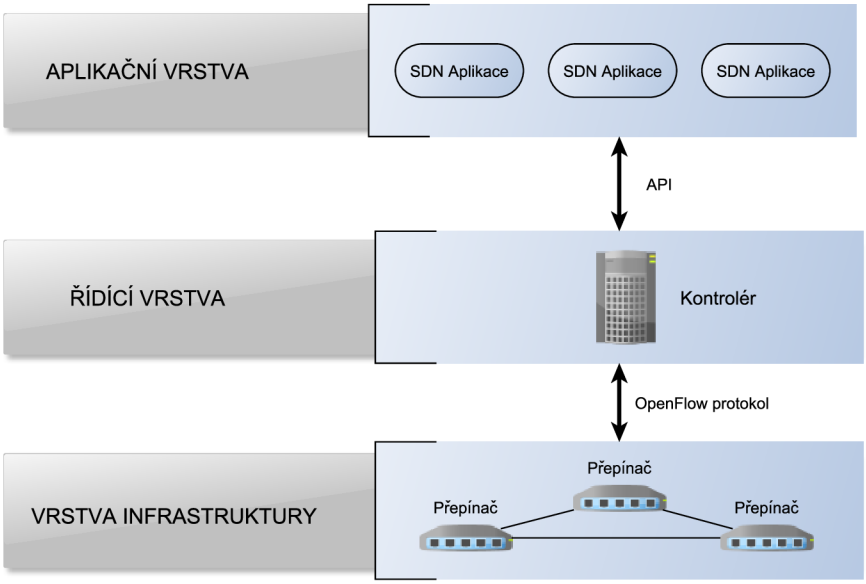
Nejzásadnějším rozdílem oproti tradičním síťovým architekturám je přenesení řídících funkcí z jednotlivých hardwarových prvků do specializovaného zařízení též známého jako kontrolér. V běžných sítích obsahují hardwarové prvky dvě hlavní části. Jedná se *řídící rovinu* (Control Plane), která je zodpovědná za všechny algoritmické výpočty co se týče směrovacích informací, navazování sousedství, rozhodování o tom, jakým způsobem bude datová jednotka obsloužena a *datovou rovinu* (Data Plane), která na základě rozhodnutí řídící roviny předá data z jednoho fyzického portu na druhý viz obr.4.1. Z distribuované architektury, kdy každý jeden prvek hledal optimální cestu v síti zvlášť pomocí různých protokolů, se tak stala architektura centralizovaná, kdy tyto vyšší řídící funkce v sobě sdružuje kontrolér, který má tak velmi dobrý přehled o dění v síti.[1], [3]



Obr. 4.1: Srovnání přepínačů

Architektura SDN je rozdělena do tří vrstev jak je možno vidět na obr.4.2

- **Vrstva infrastruktury:** Tato vrstva obsahuje tradiční hardwarové prvky jako jsou přepínače či směrovače. Tyto prvky musí podporovat protokol OpenFlow, kterým komunikují s kontrolérem. Komunikace mezi kontrolérem a podřízenými prvky bývá označováno jako takzvaný „jižní rozhraní“ neboli „Southbound“.



Obr. 4.2: Vrstvový model SDN sítě

- **Řídící vrstva:** V této vrstvě se nachází kontrolér, který je centrálním prvkem sítě udržuje si přehled o stavu sítě, provádí abstrakci pro nad ním bežící aplikace, řídí činnost všech podřízených zařízení.
- **Aplikační vrstva:** Obsahuje aplikace, které běží na kontroléru. Takovými aplikacemi může být například firewall, load-balancer, monitor stavu sítě a podobně. Aplikační vrstva díky kontroléru zprostředkovaný pohled na síť a může jednotlivé prvky skrze kontrolér programovat. Aplikační vrstva komunikuje s kontrolérem skrze takzvané „severní rozhraní“ neboli „Northbound“.

Kontrolér

Kontrolér řídí činnost jednotlivých hardwarových prvků pomocí správy jejich *tabulek toků* viz obr. 4.4. Každý prvek má minimálně jednu takovou tabulku, která obsahuje informace o tom, jak s příchozími datovými jednotkami naložit. Komunikace mezi zařízeními probíhá pomocí OpenFlow zpráv, skrze které může kontrolér do tabulek toků záznamy přidávat, odebírat či měnit. Skrze tyto zprávy může získávat informace o stavech jednotlivých zařízení či dostávat statistiky o přenesených datových jednotkách.[3]

OpenFlow protokol podporuje tři typy zpráv:

- **Controller-to-Switch:** Zprávy tohoto typu vytváří kontrolér a slouží k přímému ovládání nebo k zjištění stavu připojených prvků.
- **Asynchronní:** Tento typ zpráv vytvářejí podřízené prvky a informují v nich

přepínač o změnách jejich stavu či o vzniklých událostech.

- **Synchronní:** Zprávy jsou vytvářeny oběma stranami, jak kontrolérem tak jednotlivými prvky bez předchozích požadavků.

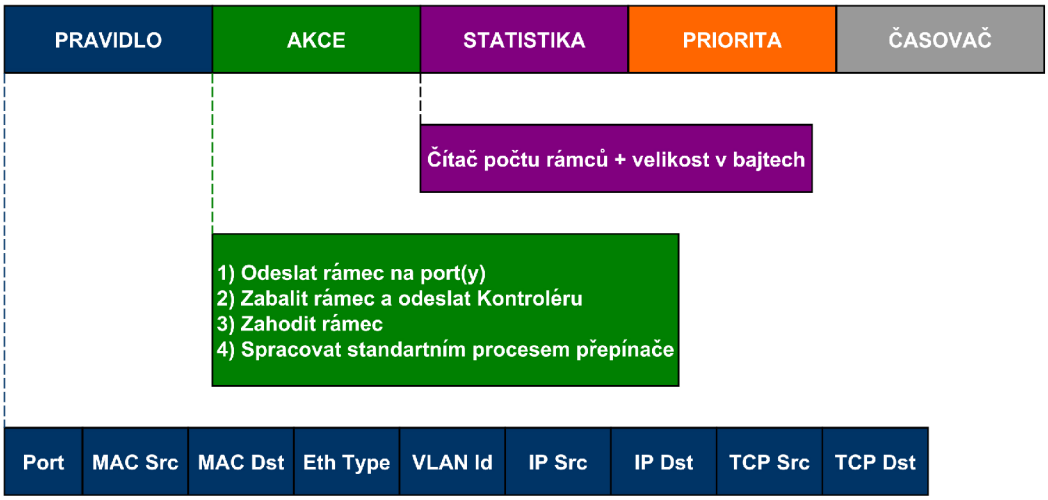
Kontrolér může fungovat v několika režimech:

- **Reaktivní režim:** Přepínač v tomto režimu funguje tím způsobem, že OpenFlow tabulky jsou ze začátku prázdné a neobsahují žádné záznamy o tom jak s daným rámcem zacházet. Při přijetí rámce pak přepínač vytvoří zprávu typu *Packet-In*, ve které se dotáže kontroléru jak s tímto rámcem zacházet. Ten mu požadované informace nejen zašle, ale zajistí naprogramování OpenFlow tabulky tak, že pro další příchozí rámec stejného typu již bude přepínač vědět jak s rámcem zacházet a nebude se již muset znovu dotazovat kontroléru. Tento režim dává kontroléru dobrý přehled o dění v síti. Mírnou nevýhodou může být vyšší zpoždění v případě rozsáhlejších sítí.
- **Proaktivní režim:** Tento režim funguje tak, že kontrolér dopředu nastaví všechny záznamy v OpenFlow tabulce pro všechny možné situace na základě svých informací. OpenFlow tabulka tak při příchodu rámce již ví jak s daným rámcem naložit.
- **Hybridní režim:** Tento režim umožňuje předchozí zmiňované techniky kombinovat a dokonce využít i využít lokální *řídíci rovinu* přepínače a to třeba v případě, že OpenFlow tabulka neobsahuje žádný záznam, který by odpovídal přijatému rámci. Přepínač se tak může rozhodnout, že jeho zpracování zajistí klasickým způsobem.[4]

Obr. 4.3: Hardwarový SDN kontrolér ONEC-A-600

Zpracování datové jednotky

Hlavička příchozí datové jednotky je porovnávána s jednotlivými záznamy v tabulce toků krok po kroku od shora dolů. V případě, že je nalezená shoda se záznamem v tabulce, je provedená stanovená akce. V opačném případě je datová jednotka poslána pro kontrolou kontroléru (`Packet-In` zpráva) nebo může být v rámci nastavení zahozena.



Obr. 4.4: Struktura OpenFlow tabulky

Význam jednotlivých polí:

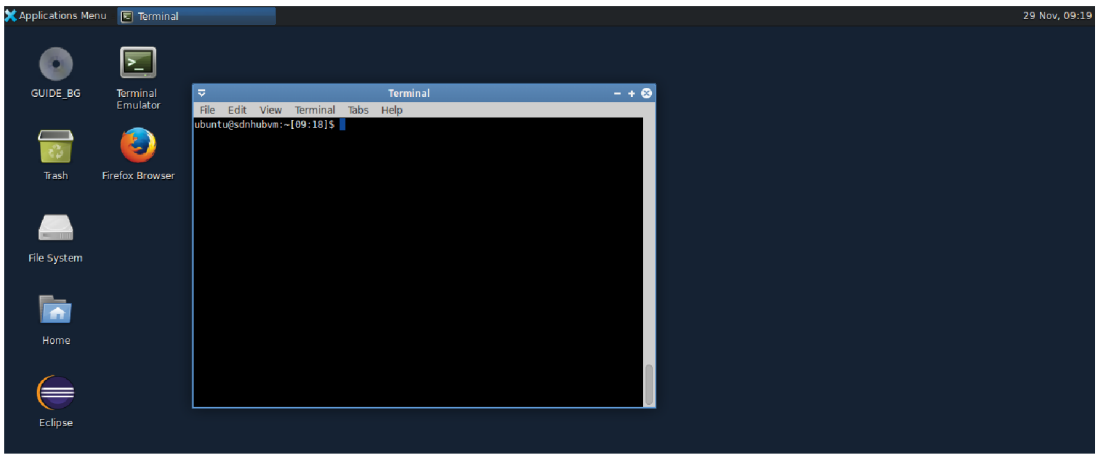
- **Pravidlo:** Při příchodu rámce zařízení prohledává jeho hlavičku a určuje, zda odpovídá některá část hlavičky stanovenému záznamu v tabulce. Prvek může být porovnáván na základě 1.,2.,3. či 4. vrstvy referenčního ISO/OSI modelu. V případě, že je toto pole *Pravidlo* v tabulce prázdné, pak provede akci pro všechny příchozí rámce.
- **Akce:** Toto pole specifikuje akci, která se má provést v případě, že je nalezená shoda. Prvek může rámec odeslat na příslušný port, odeslat kontroléru, zpracovat standardním způsobem pomocí lokální řídící roviny popřípadě rámec zahodit. Je možnost zřetězit více akcí. Například odeslat rámec na daný port a navíc jej odeslat kontroléru.
- **Statistika:** V tomto poli se inkrementuje čítač rámců obsloužených dle daného pravidla a celková velikost těchto rámců v bajtech.
- **Priorita:** V tomto poli je uvedená priorita jednotlivých záznamů tabulky toků. V případě, že přijatý rámec vyhovuje více než jednomu pravidlu je zpracován podle záznamu s vyšší prioritou.
- **Časovač:** Záznam v tabulce toků může být časově omezen. V případě, že časovač u některého záznamu vyprší, je tento záznam z tabulky toků vymazán.[1]

4.2.5 Postup řešení

Úkol č. 3

V této části laboratorní úlohy se seznámíte se síťovým emulátorem Mininet, ve kterém bude probíhat testování softwarově definovaných sítí. Vytvoříte si jednoduchou topologii a vyzkoušíte základní příkazy tohoto emulačního nástroje.

1. Z webové stránky vmware.com stáhněte virtualizační software *VMware Workstation Player*. Stažený program následně nainstalujte.
2. Ze stránek sdnhub.org stáhněte obraz linuxové distribuce s předintalovanými nástroji pro vývoj a testování softwarově definovaných sítí *All-in-one SDN App Development Starter VM*. Je doporučeno použít 64-bitovou verzi.
3. Poklikáním otevřete stažený soubor *SDN tutorial VM 64bit.ova*. Zobrazí se nabídka na importování do programu *VMware Workstation Player*. Odsouhlaste objevující se dialogové okna a spusťte import, který bude trvat několik minut v závislosti na výkonu počítače.
4. Po dokončení importu se začne spouštět linuxová distribuce a přivítá nás úvodní obrazovka plochy.

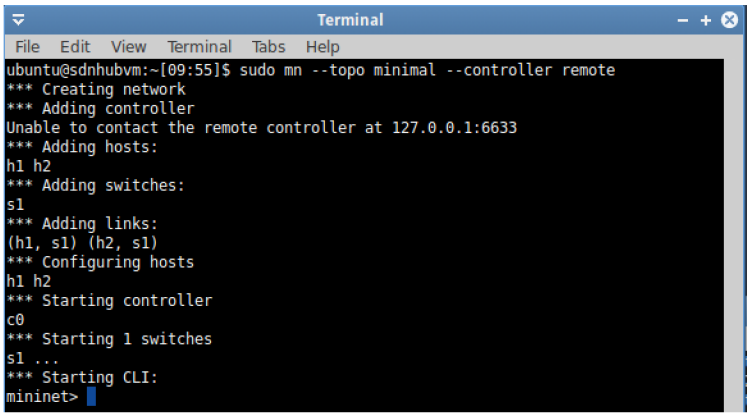


Obr. 4.5: Uvodní obrazovka linuxové distribuce

5. Nyní je potřeba skrze terminálové okno spustit prostředí Mininetu a vytvořit naši první jednoduchou topologii.
   Příkazem `sudo mn --topo minimal --controller remote` spustíme Mininet s úvodní topologií.
   - `sudo mn` - spouští emulátor Mininet
   - `--topo minimal` - tento atribut vytvoří minimální topologii s dvěma hosty `h1` a `h2` a přepínačem `s1`
   - `--controller remote` - přidá do topologie SDN kontrolér

Pro zobrazení všech možností Mininetu můžeme zobrazit nápovědu příkazem `sudo mn help`.

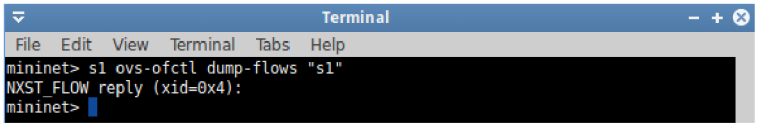
6. Provedením příkazu se nám vytvořila topologie, kdy jsou hosti `h1,h2` připojeni k přepínači `s1` a ten ke kontroléru `c0`.



Obr. 4.6: Vytvořená topologie

7. Vyzkoušejte následující sadu příkazů:
   - `mininet> nodes` - vypíše jednotlivé uzly sítě
   - `mininet> links` - vypíše spojení mezi jednotlivými prvky
   - `mininet> xterm h1 h2` - spustí terminálové okna jednotlivých stanic
   - `mininet> dump` - zobrazí jednotlivá zařízení s jejich informacemi
   - `mininet> h1 ifconfig` - zobrazí jednotlivá síťová rozhraní dané stanice
   - `mininet> exit` - pozor tento příkaz ukončí Mininet, vytvořená topologie bude ztracena
   - `mininet> help` - zobrazí nápovědu s jednotlivými příkazy Mininetu
8. V tomto kroku zkusíme poslat zprávu `icmp echo` také známou jako `ping` ze stanice `h1` na stanici `h2`. Použijeme k tomu příkaz `mininet> h1 ping h2`. Popřípadě můžeme použít příkaz `mininet> pingall`, který odešle `ping` mezi všemi stanicemi. Běh tohoto příkazu můžete zastavit stiskem kláves `CTRL+C`.

   Bohužel jak uvidíte, příkaz `ping` mezi stanicemi nefunguje. Dokázali byste říci proč?
   Zkuste si proto zobrazit tabulku toků přepínače `s1`. Použijte příkaz `mininet> s1 ovs-ofctl dump-flows "s1"`. Tímto příkazem zjistíme, že se v dané tabulce toků nenalézá žádný záznam. Jde o to, že kontrolér operuje v takzvaném **proaktivním režimu**. To znamená, že přepínač má tabulku toků prázdnou a skrze kontrolér je potřeba tabulku toků nejprve naplnit a odeslat do přepínače.
9. Nyní budeme potřebovat vytvořit záznam do tabulky toků. Použijeme příkaz `mininet> s1 ovs-ofctl add-flow "s1" in_port=1,actions=output:2`

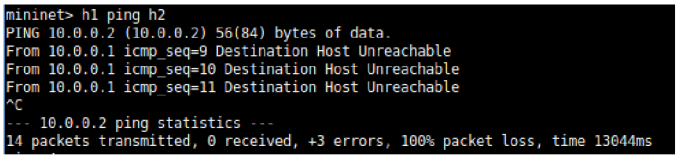


Obr. 4.7: Obsah tabulky toků přepínače S1

- `add-flow` - definujeme, že chceme přidat záznam do tabulky toků
- `in_port=1,actions=output:2` - tímto říkáme, že vše co příjde na port 1 přepínače `s1` odešleme na port 2

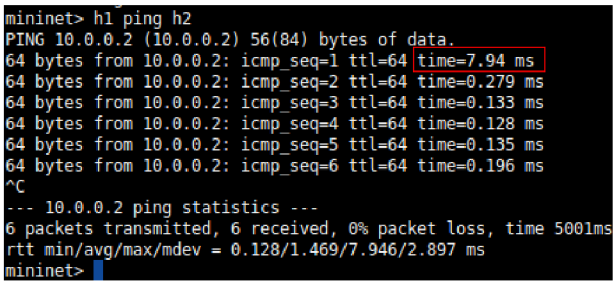
10. Záznam jsme přidali, nyní znovu zkusme ověřit vzájemnou viditelnost stanic `mininet> h1 ping h2`. Jak vidíme `ping` mezi stanicemi se opět nezdařil. Běh tohoto příkazu můžete zastavit stiskem kláves `CTRL+C`.

    Příkaz `ping` se skládá z *žádosti* a *odpovědi*. Přepínač tedy přepošle ping ze stanice `h1` stanici `h2`. Zpátky už ale odpověď stanici `h1` nedorazí. Přepínač `s1` má totiž ve své tabulce toků záznam jen pro cestu jedním směrem.



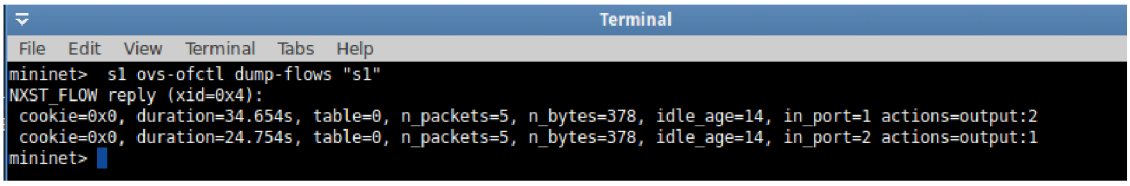
Obr. 4.8: Neúspěšný ping stanic H1 a H2

11. Do tabulky toků na přepínači proto přidáme záznam, který zajistí aby odpověď na příkaz `ping` dorazila zpět na stanici `h1`. V Mininetu proto zadáme příkaz `mininet> s1 ovs-ofctl add-flow "s1"in_port=2,actions=output:1`
12. Nyní vyzkoušíme znovu `ping` mezi stanicemi. Jak nyní můžeme vidět, komunikace mezi stanicemi již probíhá bez problémů.



Obr. 4.9: Úspěšný ping mezi stanicemi H1 a H2

13. Nyní se ještě můžeme podívat jak vypadá tabulka toků přepínače `s1`. Zobrazíme ji příkazem `mininet> s1 ovs-ofctl dump-flows "s1"`. V tabulce vidíme dva záznamy. Jeden pro směr z `h1` na `h2` a druhý pro opačný směr. Můžeme si všimnout struktury tabulky.



Obr. 4.10: Tabulka toků přepínače S1

- `duration` - vyjadřuje jak dlouho už daný záznam v tabulce toků je
- `table` - číslo tabulky, první tabulka má číslo 0
- `n_packets` - celkový počet datových jednotek, které byly zpracovány dle toho záznamu tabulky toků
- `n_bytes` - počítadlo celkové velikosti bajtů datových jednotek, které byly přiřazené tomuto záznamu tabulky toků
- `in_port` - vstupní fyzický port přepínače
- `actions` - definuje co se má s danou datovou jednotkou stát, zde konkrétně parametr `output` určuje výstupní fyzický port přepínače

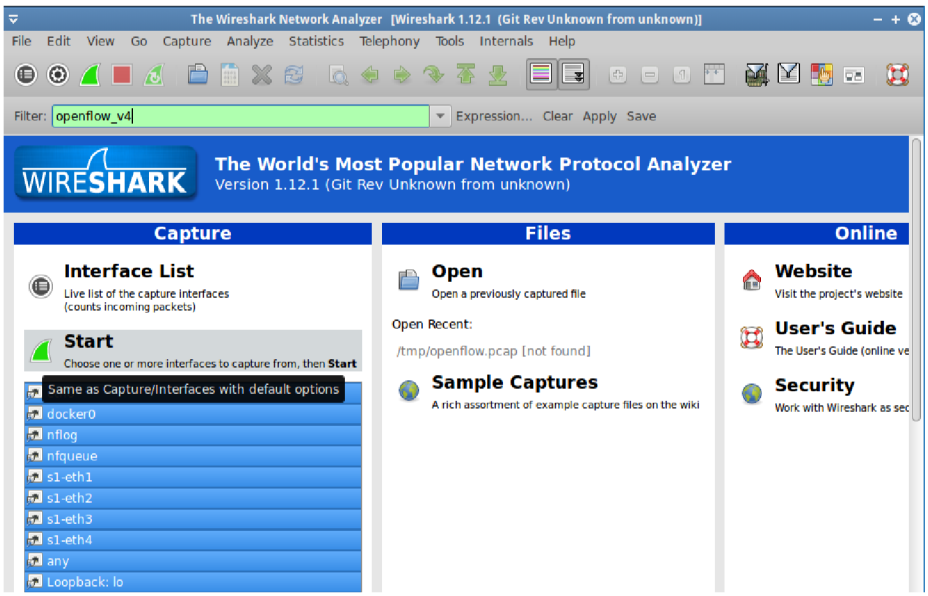
14. Posledním krokem bude ukončení Mininetu a smazání jeho dočasných souborů. Ukončení provedeme příkazem `mininet> exit`. Vymazání jeho dočasných souborů pak již v linuxovém terminálu příkazem `sudo mn -c`. Kde atribut `-c` znamená clean.

Úkol č. 4

V této části laboratorní úlohy si ukážeme jak funguje reaktivní režim SDN kontroléru. Zaměříme se také na odchytávání OpenFlow zpráv pomocí protokolového analyzátoru Wireshark.

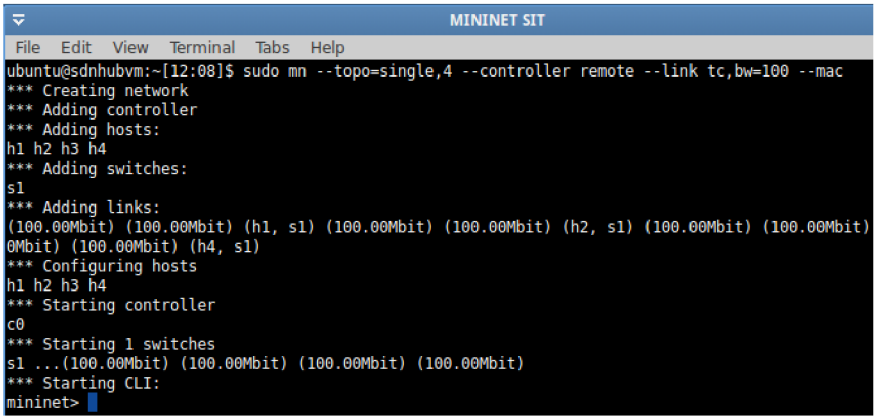
1. Otevřeme terminálové okno, ve kterém si spustíme protokolový analyzátor sítě Wireshark. Ten spustíme příkazem `sudo wireshark &`. V něm vybereme pro zachytávání všechna rozhraní, aplikujeme filtr `openflow_v4` a spustíme zachytávání.
2. Nyní otevřeme nové terminálové okno a vytvořte topologii, kde bude jeden přepínač `s1` a stanice `h1, h2, h3 a h4`. Použijeme k tomu příkaz `sudo mn --topo=single,4 --controller remote --link tc,bw=100 --mac`

- --topo=single,4 - vytvoří jeden přepínač a k němu 4 hosty



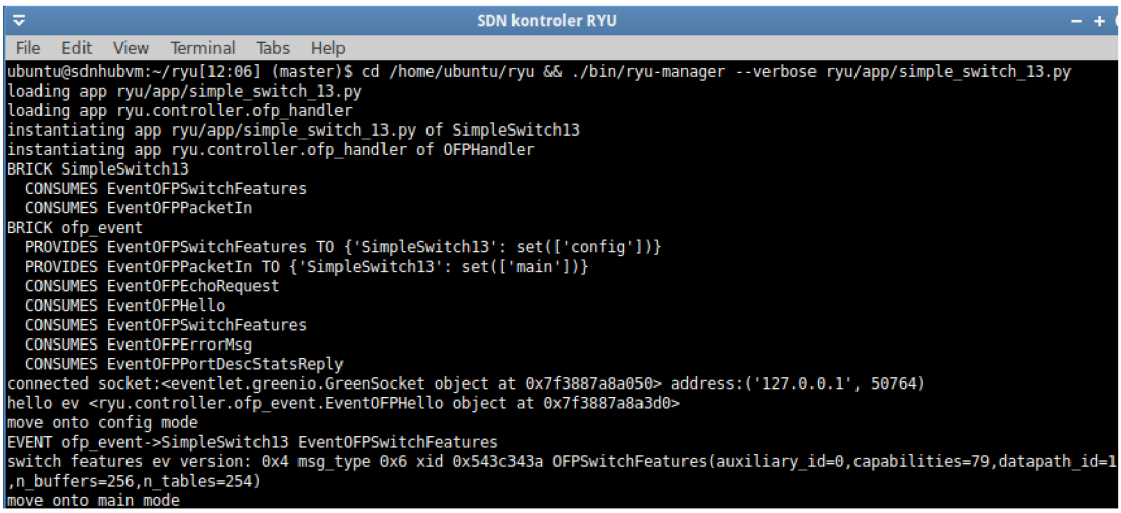
Obr. 4.11: Nastavení programu Wireshark

- --controller remote - definuje, že bude použitý vzdálený kontrolér (jeho samotné spuštění inicializujeme v následujícím kroku)
- --link tc,bw=100 - vytvoří mezi zařízeními linky s rychlostí 100 Mbit/s
- --mac - MAC adresy hostů budou odpovídat jejich IP adresám (h1, IP: 10.0.0.1, MAC:00:00:00:00:00:01)



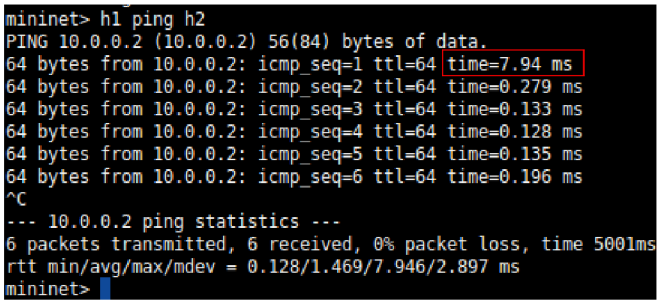
Obr. 4.12: Vytvořená topologie sítě

3. V třetím terminálovém okně inicializujeme spuštění SDN kontroléru. V tomto případě použijeme kontrolér RYU. Otevřeme si proto paralelně nové terminálové okno, ve kterém spustíme následující příkaz: `cd /home/ubuntu/ryu && ./bin/ryu-manager --verbose ryu/app/simple_switch_13.py`
4. Nyní se vrátíme zpět do terminálového okna Mininetu, kde máme vytvořenou naši topologii. Spustíme příkaz `ping` mezi stanicemi `h1` a `h2`. Všimněte si, že



Obr. 4.13: Inicializizace SDN kontroléru RYU

byl tento příkaz již úspěšný na rozdíl od pokusu v úkolu č. 3.

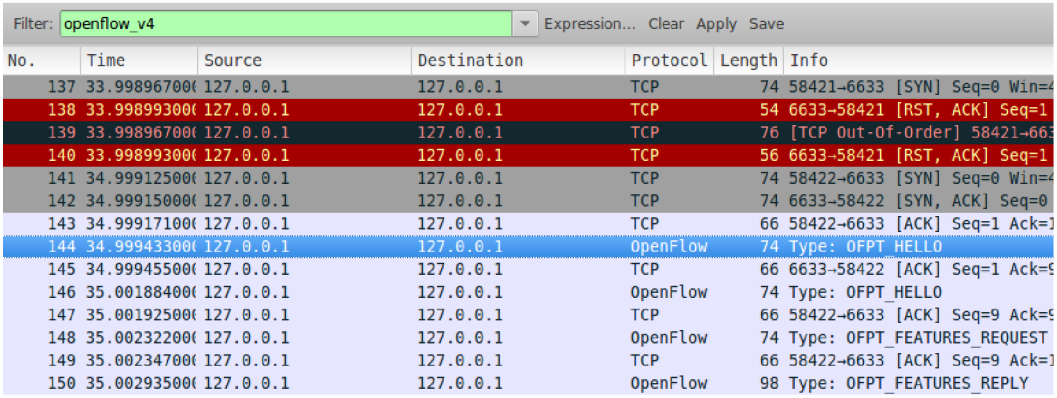


Obr. 4.14: Úspěšný ping mezi stanicemi H1 a H2

Všimněte se delší časové odezvy u první `icmp` zprávy. Je způsobena tím, že kontrolér nyní pracuje v **reaktivním režimu**. To znamená, že se přepínač `s1` dotazuje na všechny datové jednotky, které nemá ve své tabulce toků kontroléru.

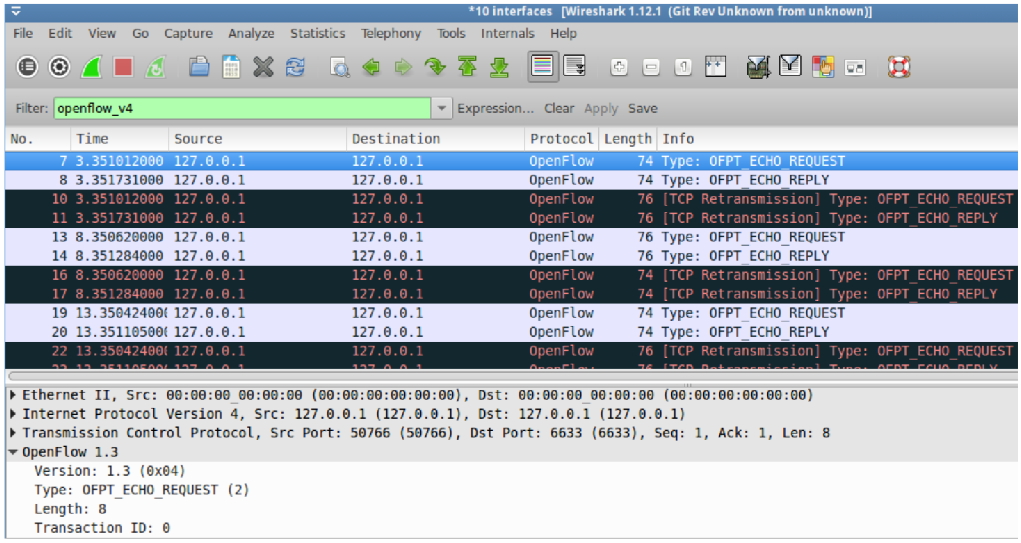
5. Vrátíme se do programu Wireshark, kde zastavíme zachytávání komunikace a blíže prozkoumáme co se v naší síti odehrálo. Prozkoumejte strukturu následujících typů zpráv `OFPT_HELLO`, `OFPT_FEATURES_REQUEST` a `OFPT_FEATURES_REPLY`.

   - `OFPT_HELLO` - pomocí těchto zpráv se navazuje spojení mezi kontrolérem a přepínačem
   - `OFPT_FEATURES` - kontrolér se dotazuje touto zprávou na schopnosti přepínače
   - `OFPT_FEATURES_REPLY` - přepínač v této zprávě zasílá kontroléru informace o svých schopnostech



Obr. 4.15: Výměna úvodních zpráv mezi kontrolérem a přepínačem

Na obr. 4.16 můžeme vidět příklad **synchronních zpráv**. Tyto zprávy jsou typu dotaz/odpověď (`OFPT_ECHO_REQUEST`/ `OFPT_ECHO_REPLY`). Zařízení pomocí nich kontrolují svoji dostupnost.



Obr. 4.16: Kontrola spojení mezi kontrolérem a přepínačem

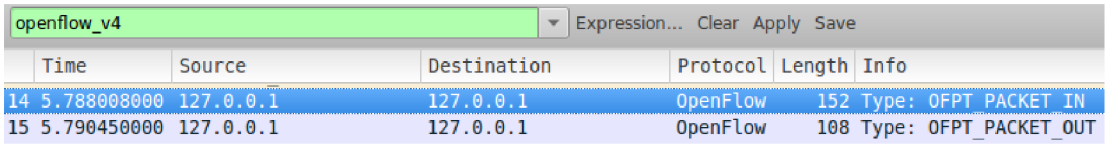
6. Nyní se zaměříme na zprávy typu `Packet-In`, `Packet-Out` a `Flow-Mod`. Vyhledejte tyto zprávy a analyzujte jejich strukturu.

   `Packet-In` zprávy - jsou odesílány přepínačem do kontroléru pro identifikaci neznámé datové jednotky, pro níž není záznam v tabulce toků

   `Paket-Out` zprávy - jsou odesílány jako reakce na `Packet-In`, v nichž specifikuje jak se má s datovou jednotkou naložit

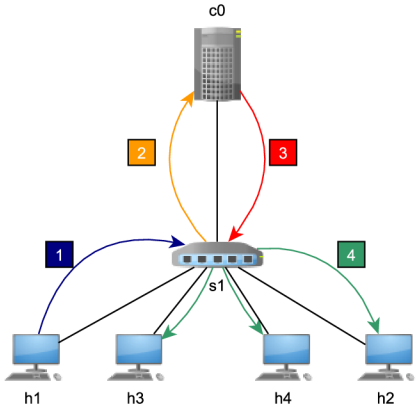
   `Flow-Mod` zprávy - tyto zprávy provádějí úpravy tabulky toků přepínače a jsou zasílány kontrolérem

Na obr. 4.17 můžeme vidět zachycenou komunikaci mezi kontrolérem a přepínačem. Na obr. 4.18 je znázorněno jak probíhalo odesílání zprávy `icmp echo`



Obr. 4.17: Zachycené OpenFlow zprávy

`request` mezi stanicí `h1` a `h2`.



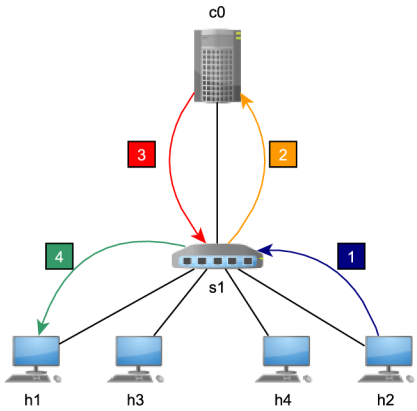
Obr. 4.18: icmp echo request ze stanice H1 na H2

1) Stanice `h1` posílá `icmp echo request` s cílovou adresou `10.0.0.2` přepínači `s1`.
2) Přepínač `s1` ve své tabulce toků nemá žádný záznam o této adrese, a proto vytvoří zprávu typu `Packet-In` s hledanou IP adresou a odešle ji kontroléru `c0`.
3) Kontrolér informace o této adrese nemá. Vygeneruje tedy zprávu `Packet-Out`, ve které nařídí akcí `OFPP_FLOOD` přepínači `s1`, aby tuto zprávu odeslal na všechny své porty, kromě toho odkud mu zpráva přišla.
4) Stanice `h2` dostává zprávu `icmp echo request`.

Zpráva tedy úspěšně dorazila úspěšně až ke stanici `h2`. Je třeba si uvědomit, že aby byl příkaz `ping` kompletní, je nutné stanici `h1` odeslat odpověď zprávou `icmp echo reply`. Schéma komunikace můžeme vidět na obr. 4.19.

1) Stanice `h2` posílá `icmp echo reply` s cílovou adresou `10.0.0.1` přepínači `s1`.

2) Přepínač `s1` ve své tabulce toků nemá žádný záznam o této adrese, a proto vytvoří zprávu typu `Packet-In` s hledanou IP adresou a odešle ji kontroléru `c0`.
3) Kontrolér má nyní informace jak o adrese stanice `h1` a `h2`. Vygeneruje tedy zprávu typu `OFTP_FLOW_MOD`, jejíž funkce je přidat do tabulky toků přepínače `s1` záznam o tomto spojení. Při dalším pokusu o komunikaci již nebude potřeba kontaktovat znovu kontrolér a komunikace mezi stanicemi `h1` a `h2` již bude probíhat výhradně přes přepínač.
4) Stanice `h1` dostává zprávu `icmp echo reply`. Komunikace mezi nimi byla úspěšná.

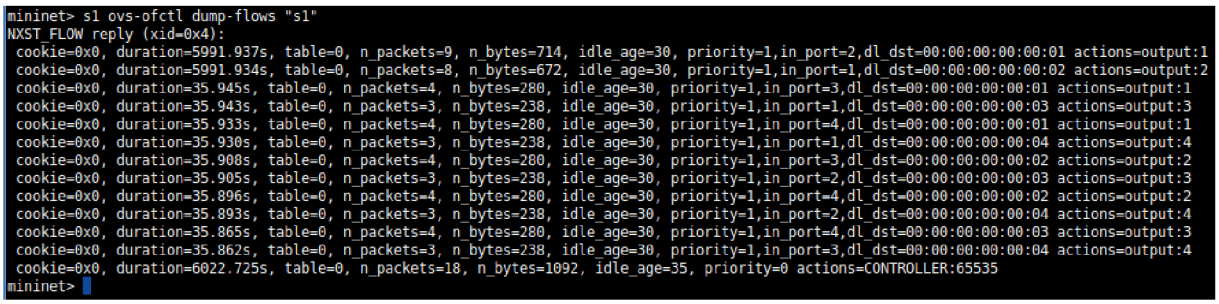


Obr. 4.19: icmp echo reply ze stanice H2 na H1

7. Nyní znovu spusťte analyzátor Wireshark a vyzkoušejte komunikaci mezi zbývajícími stanicemi pomocí příkazu `ping` nebo využijte `pingall`, který zahájí komunikace mezi všemi uzly sítě. Ověřte přítomnost nových OpenFlow zpráv a analyzujte jejich obsah.
8. Zajímavé bude nyní zkontrolovat obsah tabulky toků na přepínači `s1`. To provedete příkazem `mininet> s1 ovs-ofctl dump-flows "s1"`. Výsledek by pak měl vypadat jako na obr. 4.20.
9. Posledním krokem bude opuštění Mininetu příkazem `mininet> exit` a v terminálovém okně následně zadat příkaz `sudo mn -c`, který smaže jeho dočasně vytvořené soubory. Následně můžeme celé virtualizované prostředí ukončit kliknutím na křížek okna VMware.

4.2.6 Kontrolní otázky

1. Popište jednotlivé vrstvy architektu SDN a stručně popište jejich funkci.
2. Jaký je rozdíl mezi zprávami `Packet-In` a `Packet-Out`.



Obr. 4.20: Tabulka toků přepínače S1

3. Vysvětlete rozdíl mezi *proaktivním*, *reaktivním* a *hybridním* režimem kontroléru.
4. Co je tzv. *Northbound* a *Southbound*.

4.2.7 Seznam zkratek

IP Internet Protocol
ISO/OSI International Standards Organization / Open Systen Interconnection
MAC Media Access Control
SDN Software Defined Network

4.2.8 Literatura

[1] OpenNetworking.org, *Software defined networking: The New Norm for Networks.* [online]. [cit. 17. 10. 2015]. Dostupné z URL: <`https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`>.

[2] Mininet.org, *Mininet Walkthrough* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <`http://mininet.org/walkthrough/`>.

[3] OpenNetworking.org, *OpenFlow Switch Specification v.1.5.0* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <`https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf`>.

[4] SALISBURY, Brent. *OpenFlow: Proactive vs Reactive Flows* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <`http://networkstatic.net/openflow-proactive-vs-reactive-flows/`>.

4.3 Shrnutí laboratorní úlohy č. 2 – Vícenásobné tabulky toků a Access Control Management v síti

V této druhé laboratorní úloze zaměřené na architekturu softwarově definovaných sítí se studenti v rámci teoretické části seznámí se způsobem zpracování datových jednotek v přepínači. Dozví se rozdíly mezi zpracování pomocí jedné a více tabulek toků. Do povědomí se jim dostanou pole, dle kterých můžou hledat shodu se záznamy v tabulce a také jaké akce s datovými jednotkami mohou vykonávat.

V praktické části pak na ně čekají dva úkoly. První z nich je zaměřen na práci v síťovém emulátoru Mininet, kde si vytvoří zadanou topologii a podívají se na to, jak jde prostřednictvím konfigurace tabulek toků zabezpečit v síti Access Control Management. Vyzkouší si jak dávkově vkládat záznamy do tabulky toků přepínače a jak je s nimi možno zacházet.

Druhá část je pak zaměřena na samostatnou práci v tomto emulátoru. Studenti opět vytvoří zadanou topologii sítě a nastaví v ní deklarované přístupové práva k službám a zdrojům v síti.

Absolvováním této laboratorní úlohy budou umět studenti vysvětlit jakým způsobem se datová jednotka v přepínači zpracovává. Dále pak jaký je rozdíl mezi zpracováním v jedné tabulce toků oproti zpracování skrze více tabulek. Dostanou hlubší náhled do struktury jednotlivých záznamů těchto tabulek a budou sami schopni skrze ně ovlivňovat chování sítě. V závěru laboratorní úlohy je připraveno několik kontrolních otázek, které prověří úroveň pochopení demonstrované látky. Součástí této laboratorní úlohy je pak část pro vyučujícího, kde nalezne soubor správných odpovědí na kontrolní otázky a zdrojové soubory pro samostatný úkol č. 4.

4.4 Laboratorní úloha č. 2 – Vícenásobné tabulky toků a Access Control Management v síti

4.4.1 Cíl

Cílem této laboratorní úlohy by mělo být nahlédnutí do problematiky zpracování datových jednotek v OpenFlow přepínači v rámci více tabulek toků. Dále pak využití přepínače jako Access Control prvku, který stanovuje pravidla pro přístup k jednotlivým službám a uzlům v rámci sítě.

4.4.2 Vybavení pracoviště

- 1x Stolní počítač s nainstalovaným virtualizačním softwarem VMware
- 1x Obraz disku s All-in-one SDN App Development Starter VM
- 1x Návod s laboratorní úlohou

4.4.3 Úkoly

1. Seznamte se způsobem zpracování datových jednotek v OpenFlow přepínači.
2. Nastudujte možnosti klasifikace datových jednotek a možnosti jejich zpracování, které OpenFlow nabízí.
3. Vytvořte zadanou topologii sítě a ověřte její funkčnost.
4. Podle zadání vytvořte topologii sítě, nastavte Access Control pravidla a ověřte jejich funkčnost.
5. Odpovězte na kontrolní otázky.

4.4.4 Teoretický úvod

V předchozí laboratorní úloze č. 1 jste se seznámili s tím, že v OpenFlow přepínači existuje takzvaná tabulka toků. Ta mohla být v přepínači pouze jedna a byla označována jako `table 0`. Toto se o ovšem změnilo s příchodem OpenFlow specifikace 1.1, jejímž hlavním přínosem byla podpora více těchto tabulek toků v jednom přepínači. Jakmile je nalezena shoda s nejvyšší prioritou, je provedena odpovídající akce a datová jednotka obsloužena. Kontrolér tedy správou této tabulky ovlivňuje, jak bude s příchozími datovými jednotkami zacházeno. Hledání shody a vykonávání příslušných akcích se v tomto případě děje v rámci jedné tabulky toků. Toto řešení je přímé a jednoduché. Velkou nevýhodou je malá škálovatelnost, velké množství záznamů v jedné tabulce pro vytvoření jediné komplexnější funkcionality.[2]

Jediná tabulka toků
Se objevila, již v první specifikaci OpenFlow protokolu. Tato tabulka toků obsahuje množinu záznamů, které jsou do přepínače vkládány kontrolérem skrze OpenFlow zprávy. Jednotlivé záznamy obsahují pole jako jsou shoda, akce a priorita. Přepínač při příchodu kontroluje záhlaví datové jednotky a testuje je na shodu s jednotlivými záznamy tabulky toků. Jakmile je shoda nalezena, datové jednotce je přiřazena akce a podle ní se také zpracuje. Akce přiřazované datové jednotce mohou být rozličného typu.[2],[3]

Datová jednotka může být:

- předána na daný fyzický port

- přeposlána do kontroléru
- zahozena
- může jí byt změněno záhlaví (změna cílové IP adresy, cílového portu, přidání/odstranění VLAN identifikátoru a podobně)

Porovnávání datových jednotek s tabulkou toků na základě:

- **Fyzická vrstva:** Vstupní port
- **Linková vrstva:** MAC adresa, VLAN ID, Ethernet Type
- **Síťová vrstva:** IPv4/IPv6, ToS, DSCP, TTL, ARP, ICMP
- **Transportní vrstva:** TCP/UDP

Pro lepší představu, jak konkrétně takový záznam v tabulce toků vypadá si ukážeme jeden příklad, který si rozebereme (uvedený záznam je na jednom řádku).

```
table=0,priority=100,duration=140s,idle_age=118,n_packets=4,n_bytes=280,
nw_src=10.0.0.1,tcp_dst=80,actions=mod_nw_src=20.0.0.1,output:2
```

Pravidlo: Pro všechny příchozí datové jednotky se zdrojovou IP adresou 10.0.0.1 a cílový TCP portem 80.
Akce: Zdrojová adresa 10.0.0.1 bude změněna na 20.0.0.1 a datová jednotka bude odeslána z 2. fyzického rozhraní přepínače.
Statistika: Počet odeslaných datových jednotek, dle tohoto pravidla jsou 4 o celkové velikosti 280 bajtů (`n_packets=4,n_bytes=280`).
Priorita: Je nastavena na hodnotu 100. V případě, že by zde byl záznam, který by odpovidal stejným pravidlům ohledně zdrojové IP adresy a cílového portu, tak se datová jednotka obslouží dle záznamu s vyšší hodnotou této priority.
Čítače: V poli `duration` můžeme vidět, že tento záznam je v tabulce toků již 140 sekund. Pole `idle_age` naproti tomu ukazuje čas za jak dlouho bude záznam z tabulky toků odstraněn v případě, že neproběhne žádná shoda s příchozí datovou jednotkou.

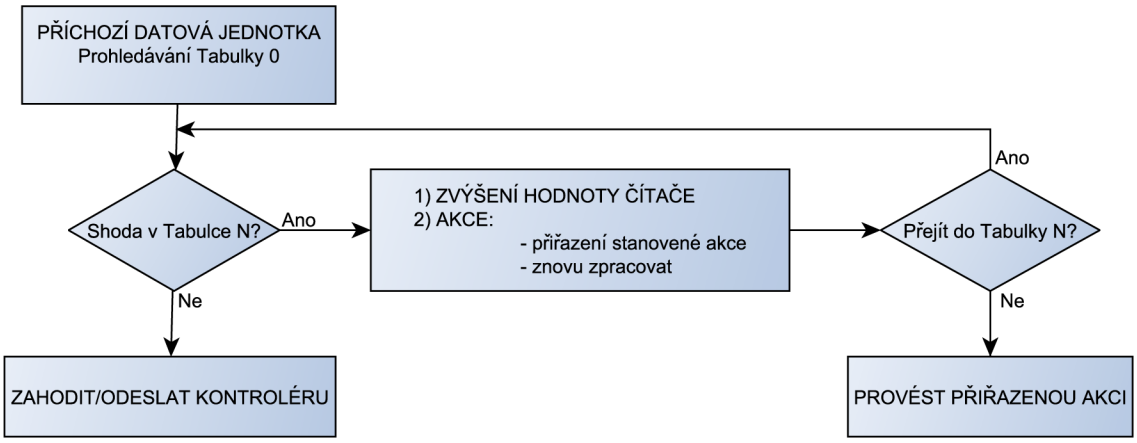
Více tabulek toků

Podporu více tabulek toků přinesla OpenFlow specifikace 1.1. OpenFlow přepínače v její implementaci podporují až 255 takových tabulek. Tabulky jsou číslovány popořadě přičemž první má hodnotu 0. Prohledávání začíná tabulkou 0 a pokračuje do dalších tabulek v případě potřeby. Toto rozdělení dovoluje zpřehlednění záznamu v těchto tabulkách a rozdělit například službu Access Control do samostatné tabulky,

zatímco pravidla pro NAT udržovat tabulce jiné. Struktura tabulek toků zůstala stejná jako v případě jediné OpenFlow tabulky. Co zde ale přibylo je **nový typ akce**. Tento nový typ akce dovoluje přesunout rozhodnutí o zpracování datové jednotky do tabulky jiné, ve které se provede porovnávání vůči jednotlivým záznamům této tabulky znovu. Celý proces zpracování datové jednotky skrze více tabulek je uveden na obr. 4.21 Záznam může poslat datovou jednotkou vždy jen do tabulky s vyšším identifikátorem (například z `tabulky 0` do `tabulky 1`), aby se předešlo zacyklení při přeskakování napříč tabulkami. Jestliže není v žádné z tabulek nalezena patřičná shoda, je datová jednotka v závislosti na nastavení odeslána kontroléru či zahozena. [2],[3] Záznam odkazující na zpracování v jiné tabulce pak v takovém případě může vypadat takto:

```
table=0, nw_src=10.0.0.1, tcp_dst=80, actions=resubmit(,3)
```

Zpracování datových jednotek s danou zdrojovou IP adresou 10.0.0.1 a TCP portem 80 bude přesunuto z tabulky 0 do tabulky 3, kde se provede nové porovnávání a hledání shody se záznamy této tabulky.

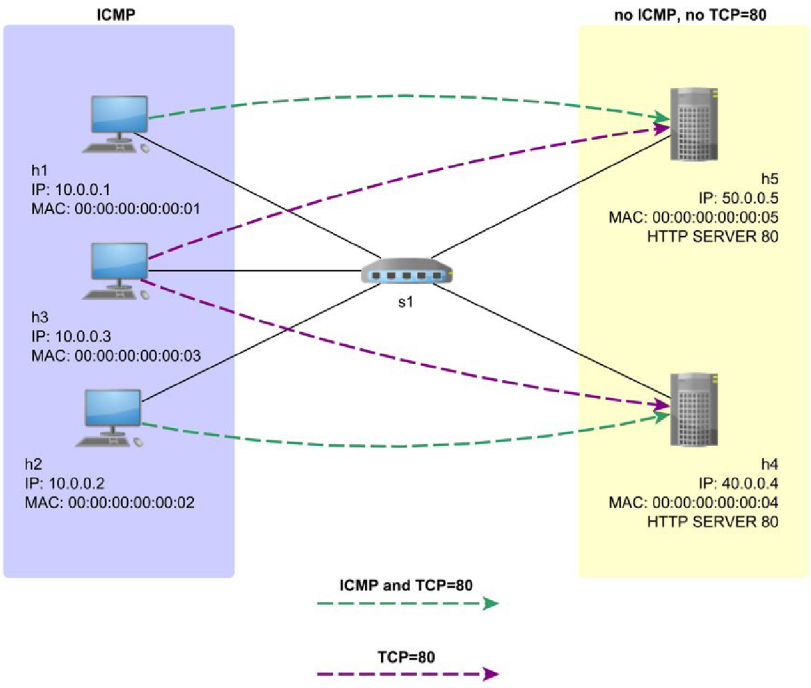


Obr. 4.21: Zpracování datové jednotky ve více tabulkách toků

4.4.5 Postup řešení

Úkol č. 3

V této části laboratorní úlohy si vyzkoušíme zpracovávání datových jednotek ve více tabulkách toků a ukážeme si, jak se pomocí jejich záznamů dá nastavit Access Control Management sítě. Naším úkolem bude vytvořit topologii uvedenou na obr. 4.22.

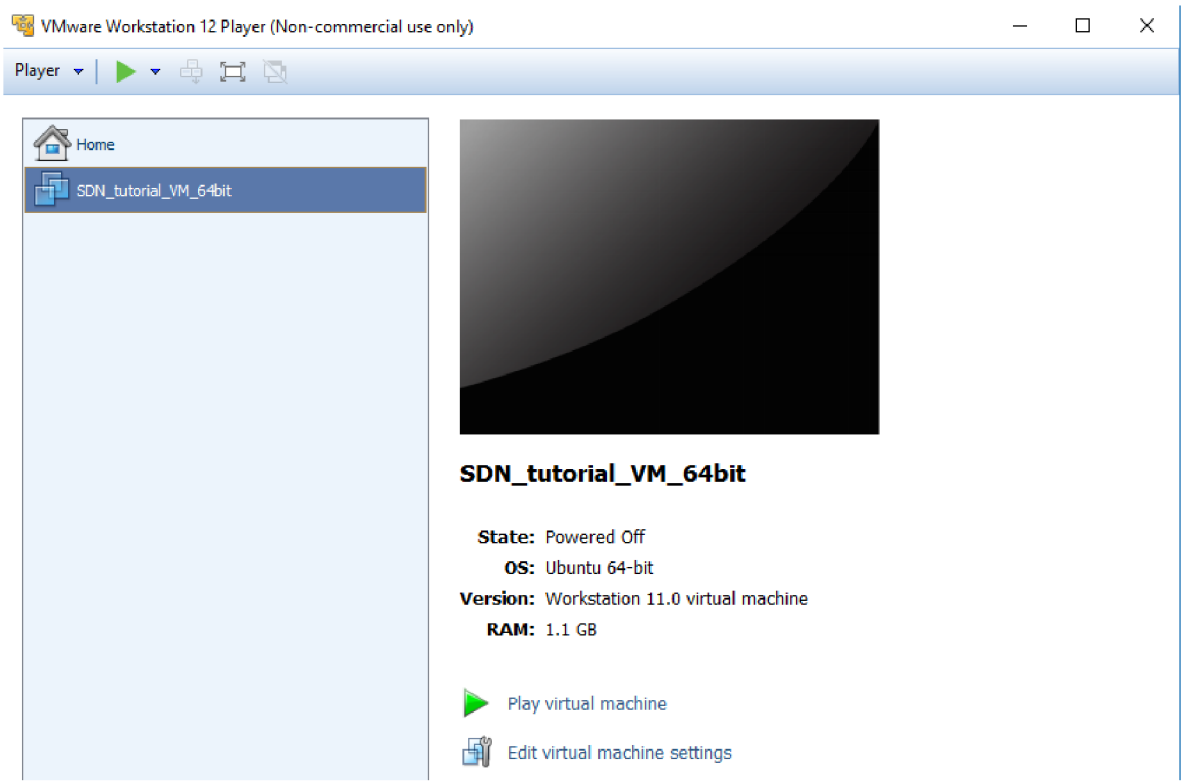


Obr. 4.22: Schéma topologie s vyznačenou bezpečností politikou

V této topologii budou figurovat přepínač `s1`, uživatelské stanice `h1, h2, h3` a dva webové servery `h4, h5`. Naším úkolem bude nastavit přístupová pravidla na jednotlivé stanice a služby v síti podle následujícího zadání a využít při tom dvou tabulek toků.

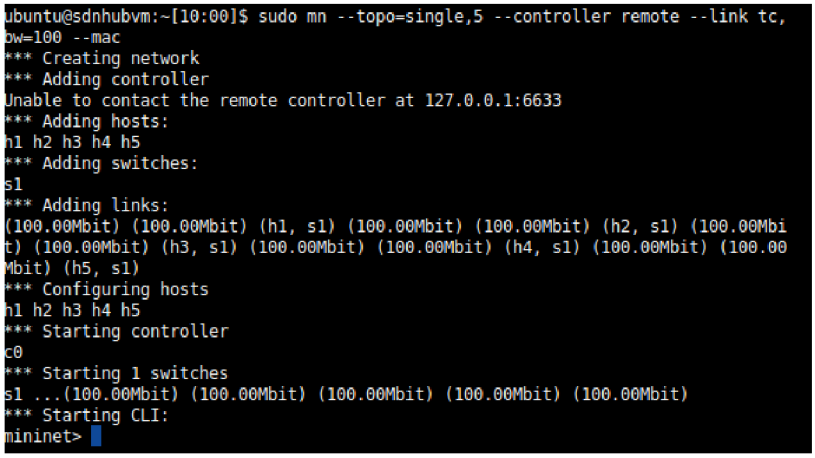
- hosté `h1, h2, h3` budou mezi sebou navzájem schopni komunikovat pomocí `icmp` zpráv
- webové servery `h4, h5` mezi sebou nebudou moci komunikovat skrze `icmp` ani navazovat `tcp` spojení
- host `h1` bude moci komunikovat skrze `icmp` a `tcp` se serverem `h5` na portu 80
- host `h2` bude moci komunikovat skrze `icmp` a `tcp` se serverem `h4` na portu 80
- host `h3` bude moci komunikovat s oběma servery `h4, h5` pouze skrze `tcp` na portu 80

1. Spustíme program *VMware Workstation Player* a spustíme virtuální stroj `SDN_tutorial_VM_64bit` a vyčkáme, než systém nastartuje do linuxového prostředí.



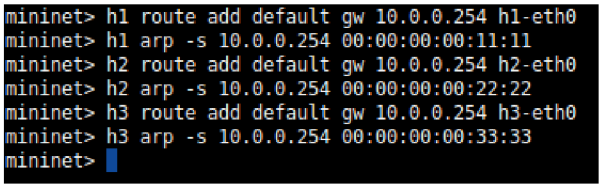
Obr. 4.23: VMware Work Station

2. Otevřeme si nové terminálové okno a vytvoříme si naši topologii dle obrázku 4.22. Použijeme k tomu příkaz `sudo mn --topo=single,5 --controller remote --link tc,bw=100 --mac`



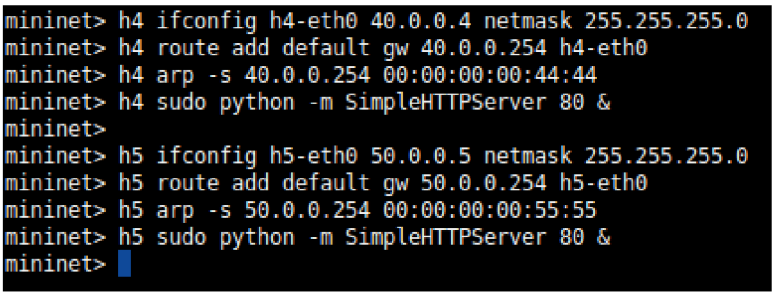
Obr. 4.24: Úspěšné vytvoření topologie sítě

3. Nyní je třeba nakonfigurovat jednotlivým stanicím `h1, h2, h3` dodatečné informace o výchozí bráně a nastavíme ji statický `ARP` záznam. V terminálovém oknu Mininetu postupně zadáme následující příkazy.



Obr. 4.25: Dodatečné nastavení hostů

4. Stejnou konfiguraci provedeme i pro servery `h4, h5`, kterým navíc změníme jejich defaultní adresy, které jsou z rozsahu 10.0.0.0/24 a spustíme na nich webové servery na portu 80.



Obr. 4.26: Dodatečné nastavení serverů a spuštění služeb

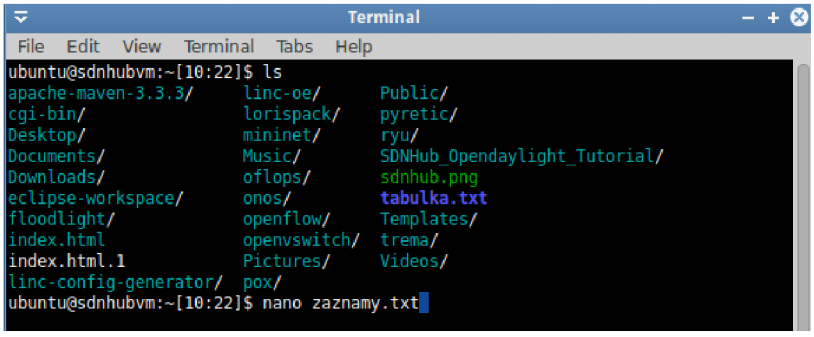
5. Nyní ověříme jak vypadá naše tabulka toků na přepínači `s1`. Příkazem `s1 ovs-ofctl dump-flows "s1"` bychom měli zjistit, že naše tabulka toků je prázdná.
6. Tabulka toků je prázdná, a proto začneme přidávat jednotlivé záznamy tak, abychom dostali stanového chování v síti. To jest splnit veškeré požadavky, deklarované v zadání.
7. V laboratorní úloze č. 1 – Úvod do softwarově definovaných sítí jsme se již setkali s vkládáním záznamů do tabulky toků. Zde jsme vkládali jednotlivé záznamy příkazem `s1 ovs-ofctl add-flows "s1"`, ve kterém jsme následně deklarovali dané pravidlo shody, akci, prioritu a podobně jakožto můžeme vidět na příkladu níže. V tomto pravidle definujeme jednoduché přepínaní mezi fyzickým portem 1 a 2.

   ```
   s1 ovs-ofctl add-flows "s1"table=0,priority=100,in_port=1,
   actions=output:2
   ```

   Toto vkládání je, ale značně neefektivní a my využijeme pro následující práci vkládání záznamů ze souboru. Syntaxe je přitom následující:

   ```
   s1 ovs-ofctl add-flows "s1" jmeno_souboru.txt
   ```

   Do tohoto textového souboru pak budeme vkládat jednotlivé záznamy pro tabulku toků. Každý záznam na zvláštní řádek.
8. Otevřeme si druhé terminálové okno a ujistíme se, že jsme v adresáři `/home/ubuntu` případně do něj přejdeme. Zde si pak vytvoříme textový dokument, do kterého budeme vkládat jednotlivé záznamy pro naši tabulku toků. Můžeme využít příkaz `nano jmeno_souboru.txt`. My si vytvoříme textový soubor s názvem `zaznamy.txt`.



Obr. 4.27: Terminálové okno

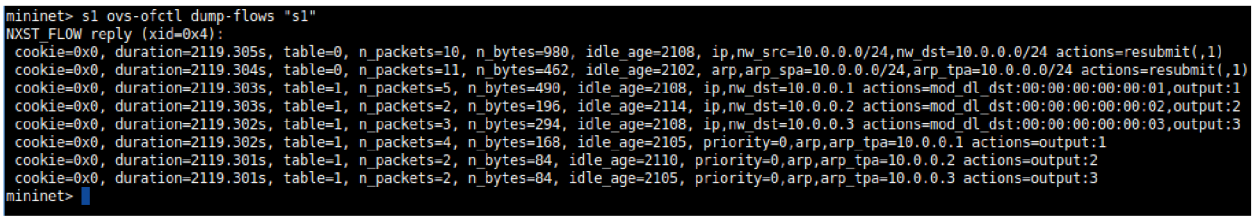
9. Otevře se nám okno textového editoru a nyní nám již nic nebrání vkládat naše záznamy. Vytvoříme si zde záznamy pro dvě tabulky toků.

   - `tabulka 0:` bude obsahovat všechny Access Control pravidla
   - `tabulka 1:` bude obsahovat směrovací a routovací informace pro dosažení jednotlivých hostů a serverů

10. Nejdříve se zaměříme na zprovoznění `icmp` komunikace v síti 10.0.0.0/24, to jest mezi hosty `h1, h2` a `h3`. Do otevřeného okna textového editoru vložíme následující záznamy.

```
#tabulka 0 - Access Control Management
table=0,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=resubmit(,1)
table=0,arp,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=resubmit(,1)

#tabulka 1 - Forwarding/Routing
table=1,ip,nw_dst=10.0.0.1,actions=mod_dl_dst=00:00:00:00:00:01,output:1
table=1,ip,nw_dst=10.0.0.2,actions=mod_dl_dst=00:00:00:00:00:02,output:2
table=1,ip,nw_dst=10.0.0.3,actions=mod_dl_dst=00:00:00:00:00:03,output:3
table=1,priority=0,arp,nw_dst=10.0.0.1,actions=output:1
table=1,priority=0,arp,nw_dst=10.0.0.2,actions=output:2
table=1,priority=0,arp,nw_dst=10.0.0.3,actions=output:3
```

    První dva záznamy v `tabulce 0` nám říkají, že jakýkoliv IP a ARP provoz ze sítě 10.0.0.0/24 do téže sítě bude zpracován podle instrukcí v `tabulce 1`. Vidíme zde návěstí `action=resubmit(,1)`, které nás do ní odkazuje.
    V `tabulce 1` pak vidíme, již konkrétní přepnutí provozu na port, podle cílové IP adresy. Pro každého hosta je zde záznam pro IP a ARP.
11. Nyní textový soubor uložíme `CTRL + O` a přesuneme se do terminálového okna Mininetu.
12. Příkazem z kroku vložíme naše záznamy do směrovače `s1`. V našem případě tedy `s1 ovs-ofctl add-flows "s1" zaznamy.txt`. V tomto případě bychom neměli dostat žádné chybové hlášení.
13. Zkontrolujeme nyní úspěšné vložení záznamů do tabulek toků viz obr. 4.28.
14. Nyní by mělo být vše připraveno. Zkontrolujte funkčnost `icmp` zpráv mezi stanicemi `h1, h2, h3`. Komunikace by měla být úspěšná.



Obr. 4.28: Tabulky toků na přepínači

15. Nyní do našich tabulek přidáme záznamy o serverech `h4` a `h5`. Otevřeme si znovu náš soubor `zaznamy.txt` a rozšíříme záznamy tabulek o tyto stanice. Všimněte si posledního záznamu v `tabulce 0`. Má nastavenu akci `drop`. Tudíž všechny datové jednotky, které nenaleznou shodu v této tabulce toků budou zahozena.

```
#tabulka 0 - Access Control

#povoleni vzajemne komunikace v 10.0.0.0/24 siti
table=0,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,action=resubmit(,1)
table=0,arp,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,action=resubmit(,1)

table=0,ip,nw_src=40.0.0.4,actions=resubmit(,1)
table=0,ip,nw_src=50.0.0.5,actions=resubmit(,1)
table=0,priority=0,actions=drop

#tabulka 1 - Forwarding/Routing
table=1,ip,nw_dst=10.0.0.1,actions=mod_dl_dst=00:00:00:00:00:01,output:1
table=1,ip,nw_dst=10.0.0.2,actions=mod_dl_dst=00:00:00:00:00:02,output:2
table=1,ip,nw_dst=10.0.0.3,actions=mod_dl_dst=00:00:00:00:00:03,output:3
table=1,ip,nw_dst=40.0.0.4,actions=mod_dl_dst=00:00:00:00:00:04,output:4
table=1,ip,nw_dst=50.0.0.5,actions=mod_dl_dst=00:00:00:00:00:05,output:5

table=1,priority=0,arp,nw_dst=10.0.0.1,actions=output:1
table=1,priority=0,arp,nw_dst=10.0.0.2,actions=output:2
table=1,priority=0,arp,nw_dst=10.0.0.3,actions=output:3
table=1,priority=0,arp,nw_dst=40.0.0.4,actions=output:4
table=1,priority=0,arp,nw_dst=50.0.0.5,actions=output:5
```

Obr. 4.29: Záznamy o stanicích H4 a H5

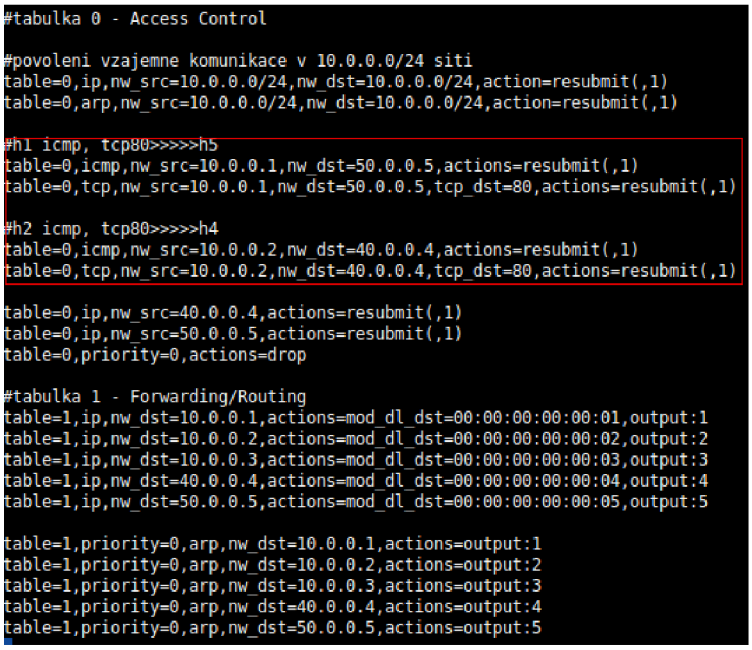
16. **Důležité i pro ostatní kroky!** Aby se nově přidané záznamy do souboru `zaznamy.txt` projevily pokaždé i v tabulkách toků na přepínači `s1`. Budeme muset použít pokaždé dvojici příkazů. Nejdříve pro vyčištění stávající tabulky a následně pro přidání nové ze souboru `zaznamy.txt`!

```
s1 ovs-ofctl del-flows "s1"
s1 ovs-ofctl add-flows "s1" zaznamy.txt
```

17. Nyní nastavíme stanici `h1`, aby dle zadání mohla komunikovat pouze se serverem `h5` a to skrze `icmp` a `tcp` na portu 80. Do textového souboru s našimi záznamy přidáme do `tabulky 0` následující záznamy. První z nich povoluje `icmp`, další pak `tcp` na portu 80.

```
#h1 icmp, tcp80»»»h5
table=0,icmp,nw_src=10.0.0.1,nw_dst=50.0.0.5,actions=resubmit(,1)
table=0,tcp,nw_src=10.0.0.1,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)
```

18. Obdobně nastavíme komunikaci mezi `h2` a `h4`.
19. Výsledný soubor by měl vypadat takto viz obr. 4.30



Obr. 4.30: Povolení komunikace na servery H4 a H5

20. Nyní vyzkoušíme komunikaci mezi `h1-h5` a `h2-h4`. Měl by fungovat příkaz `ping` mezi nimi. TCP spojení demonstrujeme tím, že se ze hostitelských stanic připojíme pomocí příkazu `h1 curl h5`. Obdobně pak pro `h2 curl h5`. Na webovém serveru těchto serverů běží skript, který vypíše obsah adresáře `/home/ubuntu`. Popřípadě si můžete vytvořit vlastní `*.html` soubor, který do tohoto adresáře umístíte a měl by se pak po zadání příkazu `h1 curl h5` zobrazit.

```
mininet> h2 ping h4
PING 40.0.0.4 (40.0.0.4) 56(84) bytes of data.
64 bytes from 40.0.0.4: icmp_seq=1 ttl=64 time=0.796 ms
64 bytes from 40.0.0.4: icmp_seq=2 ttl=64 time=0.132 ms
^C
--- 40.0.0.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.132/0.464/0.796/0.332 ms
mininet> h2 curl h4
<html>
<head>
        <title>WEB SERVER</title>
</head>
<style>
h1 {
        color:red;
        font-size:30px;
}
</style>
<body>
<h1>Ahoj, uspesne jste se dostali az na webovy server!</h1>
</body>
</html>
mininet>
```

Obr. 4.31: Ověření icmp, tcp komunikace mezi H2 a H4

21. Dalším úkolem je povolit pouze `icmp` komunikaci mezi `h3` a servery `h4, h5`. Do našeho konfiguračního souboru přidáme příkazy:

    ```
    #h3 tcp80»»»h4,h5
    table=0,tcp,nw_src=10.0.0.3,nw_dst=40.0.0.4,tcp_dst=80,actions=resubmit(,1)
    table=0,tcp,nw_src=10.0.0.3,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)
    ```

    Znovu pak stávající konfiguraci tabulek toků na přepínači smažeme a znovu přidáme záznamy z konfiguračního souboru. Nyní můžeme otestovat správnost naší konfigurace provedením následujících příkazů:
    `h3 curl h4`
    `h3 curl h5`
    Můžeme vyzkoušet i příkaz `h3 ping h4` či `h3 ping h5`. Ty ovšem nebudou fungovat, protože tuto službu mezi stanicemi nemáme povolenou.
22. V tomto kroku vyzkoušejte `ping` mezi stanicemi `h4` a `h5`. Jak vidíte příkaz funguje bez problému, stejně jako `h4 curl h5`.

23. Toto chování máme dle zadání zakázat, proto přidáme do tabulky omezující záznamy.

```
#h4»»«««h5 no ping, no tcp80
table=0,icmp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=drop
table=0,tcp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=drop
```

24. Znovu ověříme mezi těmito stanicemi `ping` a `tcp` spojení. Nyní tyto služby nefungují, což je v pořádku.
25. Na závěr ještě v Mininetu spustíme příkaz `pingall`, kterým zkontrolujeme konektivitu mezi všemi stanicemi. Pokud výsledek vypadá jako na obrázku4.32, vše jsme nastavili správně.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 X h5
h2 -> h1 h3 h4 X
h3 -> h1 h2 X X
h4 -> X h2 X X
h5 -> h1 X X X
*** Results: 50% dropped (10/20 received)
mininet>
```

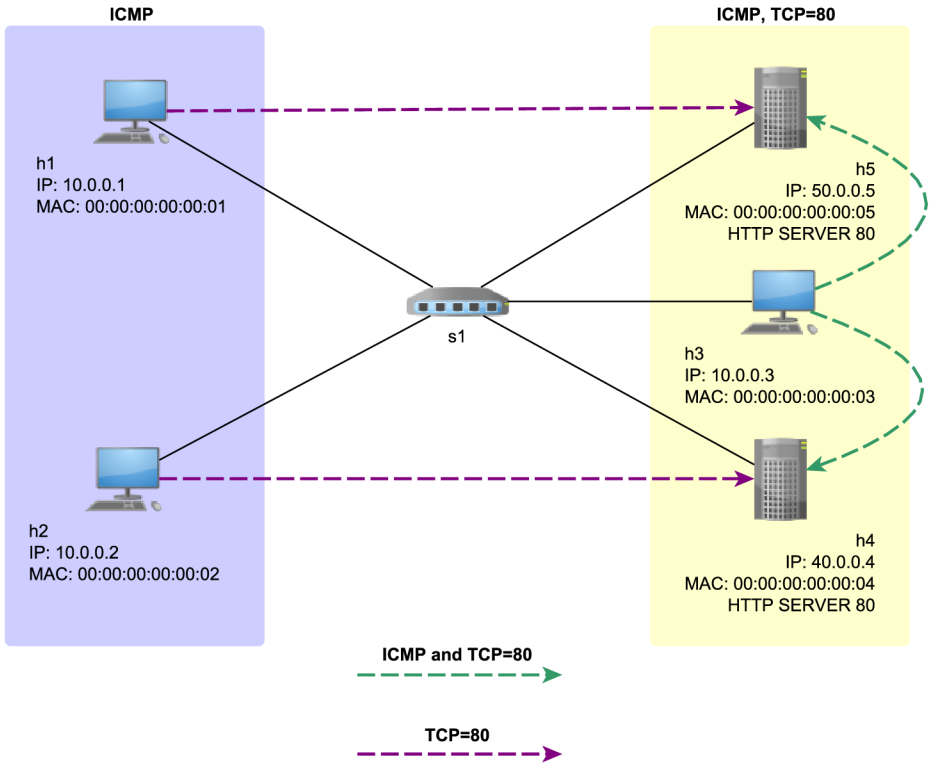
Obr. 4.32: Ověření konektivity mezi všemi stanicemi

```
#tabulka 0 - Access Control
table=0,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,action=resubmit(,1)
table=0,arp,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,action=resubmit(,1)

table=0,icmp,nw_src=10.0.0.1,nw_dst=50.0.0.5,actions=resubmit(,1)
table=0,tcp,nw_src=10.0.0.1,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)

table=0,icmp,nw_src=10.0.0.2,nw_dst=40.0.0.4,actions=resubmit(,1)
table=0,tcp,nw_src=10.0.0.2,nw_dst=40.0.0.4,tcp_dst=80,actions=resubmit(,1)

table=0,tcp,nw_src=10.0.0.3,nw_dst=40.0.0.4,tcp_dst=80,actions=resubmit(,1)
table=0,tcp,nw_src=10.0.0.3,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)

table=0,icmp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=drop
table=0,tcp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=drop

table=0,ip,nw_src=40.0.0.4,actions=resubmit(,1)
table=0,ip,nw_src=50.0.0.5,actions=resubmit(,1)
table=0,priority=0,actions=drop

#tabulka 1 - Forwarding/Routing
table=1,ip,nw_dst=10.0.0.1,actions=mod_dl_dst=00:00:00:00:00:01,output:1
table=1,ip,nw_dst=10.0.0.2,actions=mod_dl_dst=00:00:00:00:00:02,output:2
table=1,ip,nw_dst=10.0.0.3,actions=mod_dl_dst=00:00:00:00:00:03,output:3
table=1,ip,nw_dst=40.0.0.4,actions=mod_dl_dst=00:00:00:00:00:04,output:4
table=1,ip,nw_dst=50.0.0.5,actions=mod_dl_dst=00:00:00:00:00:05,output:5

table=1,priority=0,arp,nw_dst=10.0.0.1,actions=output:1
table=1,priority=0,arp,nw_dst=10.0.0.2,actions=output:2
table=1,priority=0,arp,nw_dst=10.0.0.3,actions=output:3
table=1,priority=0,arp,nw_dst=40.0.0.4,actions=output:4
table=1,priority=0,arp,nw_dst=50.0.0.5,actions=output:5
```

Obr. 4.33: Záznamy v tabulkách toků přepínače S1

4.4.6 Úkol č. 4

V této části laboratorní úlohy budete mít za úkol vytvořit následující topologii a nastavit v ní následující Access Control Management pravidla.



Obr. 4.34: Topologiie samostatného úkolu

- hosté `h1, h2` budou mezi sebou navzájem schopni komunikovat pomocí `icmp` zpráv
- hosté `h1, h2` nebudou moci komunikovat skrze `icmp` se stanicí `h3`
- webové servery `h4, h5` mezi sebou budou moci komunikovat skrze `icmp` a navazovat `tcp` spojení
- host `h1` bude moci komunikovat skrze `tcp` se serverem `h5` na portu 80
- host `h2` bude moci komunikovat skrze `tcp` se serverem `h4` na portu 80
- host `h3` bude moci komunikovat s oběma servery `h4, h5` skrze `icmp` a `tcp` na portu 80

Při řešení toho úkolu se můžete inspirovat úkolem předešlým, kde naleznete vše co potřebujete. Jakmile budete mít úkol hotový a úspěšně nastaveny všechny Access Control pravidla, tak se přihlaste a nechte zkontrolovat vyučujícím.

4.4.7 Kontrolní otázky

1. Popište zpracování datové jednotky při příchodu do OpenFlow přepínače.
2. Jaký je rozdíl mezi zpracováním v rámci jedné tabulky toků a mezi více tabulkami.
3. Jaký smysl má nastavovat Access Control Management pravidla v síti.

4.4.8 Seznam zkratek

ARP Address Resolution Protocol
DSCP Differentiated Service Code Point
ICMP Internet Control Message Protocol
IP Internet Protocol
IPv4 Internet Protocol version 4
IPv6 Internet Protocol version 6
MAC Media Access Control
NAT Network Address Translation
TCP Transport Control Protocol
ToS Type Of Service
TTL Time To Live
UDP User Datagram Protocol
VLAN Virtual Local Area Network
VLAN ID Virtual Local Area Network Identificatior

4.4.9 Literatura

[1] OpenNetworking.org, *OpenFlow Switch Specification v.1.5.0* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <`https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf`>.

[2] OpenNetworking.org, *The Benefits of Multiple Flow Tables and TTPs* [online]. [cit. 25. 11. 2015]. Dostupné z URL: <`https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_Multiple_Flow_Tables_and_TTPs.pdf`>.

[3] OpenvSwitchManual.org, *Open vSwitch Manual* [online]. [cit. 20. 11. 2015]. Dostupné z URL: <`http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt`>.

[4] Mininet.org, *Mininet Walkthrough* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <http://mininet.org/walkthrough/>.

4.5 Shrnutí laboratorní úlohy č. 3 – Tvorba vlastní topologie a Policy-based Routing

V třetí laboratorní se studenti zaměří na tvorbu vlastních topologií pro emulátor Mininet a seznámí se s pojmem Policy-based Routing.

V teoretické části se seznámí s tím, jakým způsobem je možné si v Mininetu vlastní topologii sítě navrhnout a implementovat. Okrajově se seznámí s programovacím jazykem Python. Představen zde i bude pojem Policy-based Routing, který si následně v praktické části samostatně vyzkouší implementovat do zadané topologie sítě.

V praktické částí studenty čekají dva praktické úkoly. V prvním z nich si studenti vytvoří zadanou topologii jednoduché sítě, kterou následně spustí v Mininetu. Druhá praktická úloha pak spočívá v samostatné práci v Mininetu. Studenti dostanou zadání topologie a požadavky, jakým způsobem bude síť fungovat. Jedním z bodů je implementace Policy-based Routingu a uměle tak ovlivnit datový tok tak, aby procházel stanovenými cestami. Budou přitom muset využít znalostí z předešlého úkolu, ale i předešlých laboratorních úloh. Tato úloha je koncipována jako opakovací a studenti v ní budou muset použít znalostí a důvtipu z úloh předešlých.

Po absolvování této laboratorní úlohy budou studenti schopni tvořit vlastní topologie pro emulátor Mininet a nebudou se muset omezovat na ty, které jsou zde přítomny již v základu. Seznámí se se základy jazyka Python, kterým se toto prostředí ovládá. Studenti budou schopni vysvětlit pojem Policy-based Routing, popsat jeho specifika a jeho vysvětlit přednosti jeho nasazení což si v praktické části i vyzkouší. Na závěr úlohy je připraveno i několik kontrolních otázek vztahující se k řešené problematice.

4.6 Laboratorní úloha č. 3 – Tvorba vlastní topologie a Policy-based Routing

4.6.1 Cíl

V této laboratorní úloze se podíváme jak vytvářet vlastní topologie v prostředí Mininet, kde se okrajově seznámíme se skriptovacím jazykem Python. V další části si

pak vytvoříme zadanou topologii a nastavíme politiky v síti dle zadaných požadavků. Nově se zde setkáme s pojmem Policy-based Routing, který je v SDN sítích hojně využíván. Použijeme přitom již nabyté znalosti z předchozích úloh.

4.6.2 Vybavení pracoviště

- 1x Stolní počítač s nainstalovaným virtualizačním softwarem VMware
- 1x Obraz disku s All-in-one SDN App Development Starter VM
- 1x Návod s laboratorní úlohou

4.6.3 Úkoly

1. Seznamte se s teoretickým úvodem k této úloze, kde se zaměřte na to jakým způsobem tvořit vlastní topologie a na informace týkající se Policy-based Routingu.
2. Vytvořte zadanou topologii sítě a nastavte její chování podle zadání.
3. Samostatně vytvořte zadanou topologii, nastavte její chování podle zadání. Následně pak demonstrujte její chování vyučujícímu.
4. Odpovězte na kontrolní otázky.

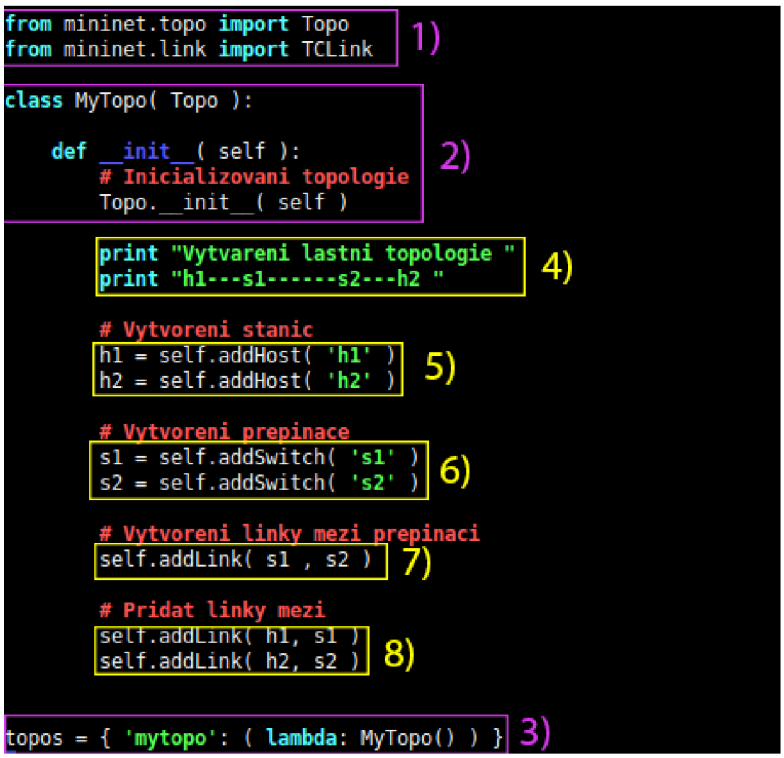
4.6.4 Teoretický úvod

Tvorba vlastních topologií pro emulátor Mininet

V předešlých dvou laboratorních úlohách jsme se seznámili s prostředím síťového emulátoru Mininet, pomocí něhož jsme byli schopni demonstrovat vlastnosti a možnosti SDN sítí. Využívali jsme k tomu především topologie, kterými Mininet disponuje již v základu. Nicméně pro pokročilejší tvorbu síťových topologií a především pak pro uspokojení specifičtějších nároků na ně můžeme narazit na to, že nám již základní topologie v Mininetu obsažené přestanou dostačovat.

Pro tyto případy můžeme využít jazyk Python a knihovny, kterými Mininet disponuje a vybudovat takovou topologii jakou budeme potřebovat. Nejdříve ze všeho je třeba si vytvořit soubor s příponou `*.py`, do něhož budeme vkládat informace o naší topologii tj. informace o přepínačích, stanicích, linkách mezi nimi a podobně. Do nově vytvořeného souboru budeme muset umístit následující nezbytné deklarace viz obr. 4.35, které si blíže představíme.

1. Těmito úvodními řádky naimportujeme knihovny Mininetu `Topo`, která obsahuje objekty jako jsou stanice, přepínače a linky pro jejich propojení. Knihovna `TCLink` pak obsahuje linky na propojování uzlů sítě s obsáhlejšími parametry než ty, které obsahuje knihovna `Topo`.



Obr. 4.35: Soubor s vlastní topologií sítě

2. Další nezbytnou částí je pak inicializace hlavní funkce `def _init_( self ):`, za níž už budeme psát náš vlastní kód k tvorbě topologie.
3. Poslední nezbytnou částí, je pak uzavření třídy `MyTopo`, do které vkládáme náš kód topologie.

   Nyní se již můžeme plně soustředit na tvorbu topologie.
4. Příkazem `print ""` můžeme do konzolového okna mininetu vypsat nějaký textový řetězec či hodnotu proměnné.

5. V této části vytváříme dvě stanice `h1, h2` pomocí metody `addHost()`. Funkce `addHost` může mít více vstupních parametrů jako jsou IP adresa, MAC adresa či síťová maska.

   ```
   h1 = self.addHost('h1',ip='10.1.0.1',mac='00:1:00:00:00:01',prefixLen='24')
   ```

6. Obdobným způsobem pak můžeme vytvářet přepínače metodou `addSwitch()`.
7. Další částí je pak propojení vytvořených uzlů mezi sebou. V tomto bodu propojujeme přepínače `s1` a `s2` mezi sebou metodou `addLink()`, jejíž dva povinné parametry jsou názvy protistran.
8. V této části propojujeme jednotlivé stanice s přepínači. Opět k tomu využíváme metody `addLink()`. Vstupními argumenty této funkce však nemusí být pouze názvy protistran. Importováním knihovny `TCLink` můžeme lince nastavovat rozličné QoS parametry jako jsou zpoždění `delay`, velikost fronty `max_queue_size` či ztrátovost v procentech `loss`.

   ```
   self.addLink(s1, s2, bw=10, delay='5ms', max_queue_size=1000, loss=10)
   ```

   Pro podrobnější informace o knihovnách a API Mininetu můžete navštívit stránku s dokumentací na adrese http://mininet.org/api/hierarchy.html

Nyní už stačí jen výsledný soubor s topologií uložit (přípona `*.py`) a spustit v Mininetu. To provedeme následujícím příkazem.

```
sudo mn --custom nazev-souboru-s-topologii.py --topo=mytopo --arp --link=tc

#sudo mn .... spouští Mininet
#--custom soubor.py --topo=mytopo .... informuje minine o spuštění
 vlastní topologie, příkaz je následován jménem souboru
```

Tímto způsobem je možné v prostředí Mininetu vytvářet a testovat vlastní topologie sítí a neomezovat se pouze na ty, které obsahuje. V tomto úvodu, jsme si ukázali pouze základní informace nutné k vypracování tohoto laboratorního cvičení.

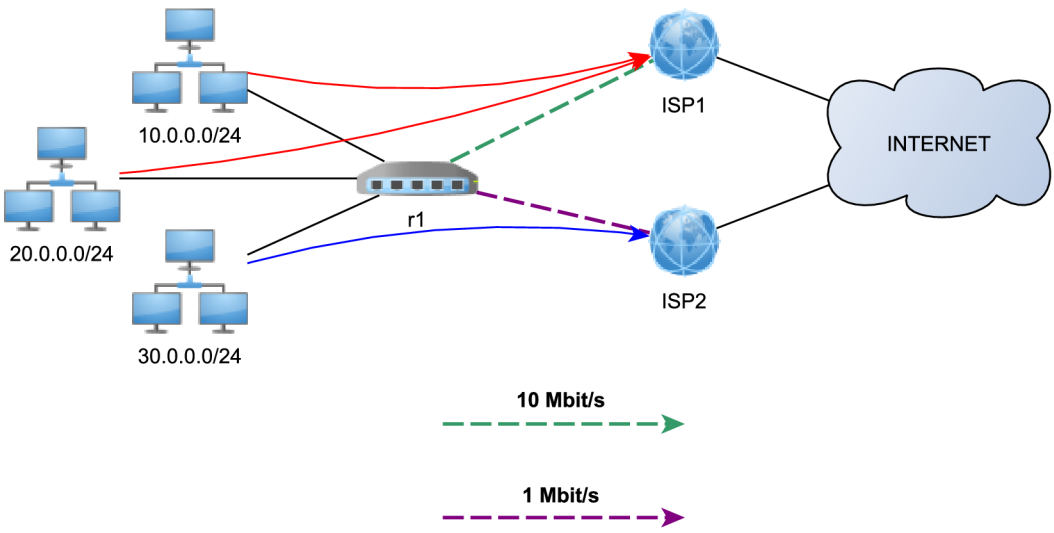
Policy-based Routing

Jedná se o techniku používanou k vytvoření směrovacích rozhodnutí na základě pravidel, která stanoví administrátor sítě. Směrovače klasicky posílají pakety do jejich cílové destinace na základě informací v jejich směrovacích tabulkách. Použitím Policy-based Routingu (PBR) je možné realizovat koncepci, která výběrově určuje jakou cestou bude daný paket odeslán.

Výhody Policy-based Routingu:

- **Source-Based Transit Provider Selection** - Organizace můžou používat PBR k směrování provozu pocházejícího z různých skupin uživatelů, prostřednictvím různých internetových propojení, přes tzv. policy routery.
- **QoS** - Rozlišením provozu, nastavením hodnot priority nebo typu služby (Type of Service - TOS) v hlavičkách IP paketů mohou být pro jednotlivé služby vybrány trasy s požadovanými charakteristikami QoS.
- **Snižování nákladů** - Organizace můžou dosáhnout snížení nákladů odesíláním určitých dat skrze cenově levnější sdílené trasy zatímco kritická data mohou odesílat dražším dedikovaným spojem.
- **Rozdělení zátěže** - Správci sítí mohou realizovat koncepci distribuce provozu mezi rozmanitými trasami založenými na provozních charakteristikách. Mohou tak například zátěž distribuovat v poměru 50:50 mezi dvěma trasami.

Realizace techniky PBR je v SDN sítích hojně využívány a nároky na její implementaci nejsou veliké. Příchozí datová jednotka je v přepínači testována na shodu v tabulce toků a následně vykonána akce. V tomto případě tedy odeslat datovou jednotku ven skrze daný port. Můžeme tak snadno směrovat provoz do určitých cílových sítí podle zadaných pravidel a politik.



Obr. 4.36: Policy-based Routing

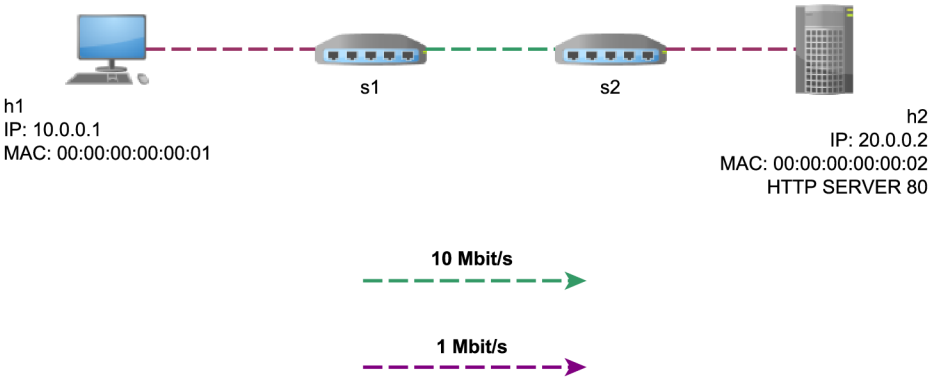
- Sítě `10.0.0.0/24` a `20.0.0.0/24` se do internetu dostanou skrze poskytovatele ISP1 díky 10 Mbit lince.

- Síť `30.0.0.0/24` se do něj dostane skrze poskytovatele ISP2 a pomalejší 1 Mbit linku.

4.6.5 Postup řešení

Úkol č. 2

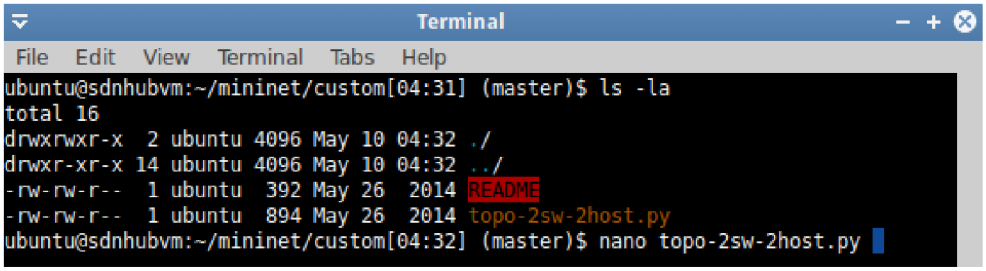
V tomto úkolu si vytvoříme vlastní topologii viz obr. 4.37, kterou následně spustíme v emulátoru Mininet a nastavíme síti zadané chování.



Obr. 4.37: Schéma topologie s vyznačenými požadavky

V této topologii bude stanice **h1**, která představuje hosta a stanice **h2**, na které bude spuštěn webový server. Dále se zde nachází přepínače **s1** a **s2**, které jsou vzájemně propojeny. Mezi jednotlivými uzly sítě jsou různé rychlosti linek. Mezi přepínači se jedná o přenosovou rychlost 10 Mbit/s, stanice jsou pak připojeny k přepínačům pomocí linky o kapacitě 1 Mbit/s.

1. Spustíme si náš virtuální Linuxový stroj s názvem `SDN_tutorial_VM_64bit`.
2. Po nastartování systému se v otevřeném terminálovém okně přesuneme do adresáře `mininet/custom`, kde si vypíšeme jeho obsah. Vidíme zde, že obsahuje soubor `topo-2sw-2host.py` viz Obr.4.38. Tento vzorový soubor v sobě uchovává informace o struktuře vlastní topologie pro simulátor Mininet.
3. Tento soubor si otevřeme a podíváme se jaké informace obsahuje pomocí příkazu `nano topo-2sw-2host.py`. Vidíme, že soubor obsahuje objekty jako jsou hosté, přepínače a linky mezi nimi. Můžeme zde vidět, že se jedná o podobnou topologii jako je v zadání našeho úkolu viz obr. 4.39.

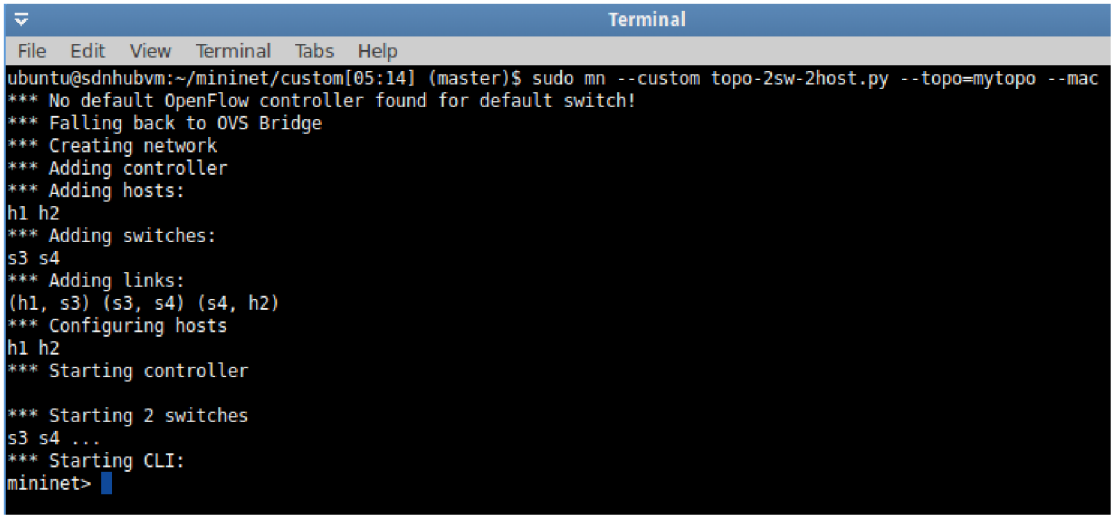


Obr. 4.38: Výpis adresáře custom

```
"""
   host --- switch --- switch --- host

"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Obr. 4.39: Struktura vzorové topologie

4. Tuto topologii si spustíme v emulátoru Mininet. Do terminálového okna zadáme příkaz `sudo mn --custom topo-2sw-2host.py --topo=mytopo --mac`. Topologie by se pak měla v Mininetu bez problému spustit viz obr. 4.40.
   - `sudo mn` - tento příkaz spouští Mininet
   - `--custom topo-2sw-2host.py --topo=mytopo` - spuštění vlastní topologie
5. Nyní překročíme k tvorbě topologie dle našeho zadání. Inspirovat se můžeme touto vzorovou topologií.
6. Zkopírujeme si proto obsah souboru `topo-2sw-2host.py` a vytvoříme naši vlastní topologii s názvem `lab3-ukol2.py`. Ke vytvoření kopie použijte li-



Obr. 4.40: Spusteni vzorove topologie v prostředí Mininet

nuxový příkaz `cp`. Podrobnosti o tom jak jej použít případně vyhledejte na internetu.

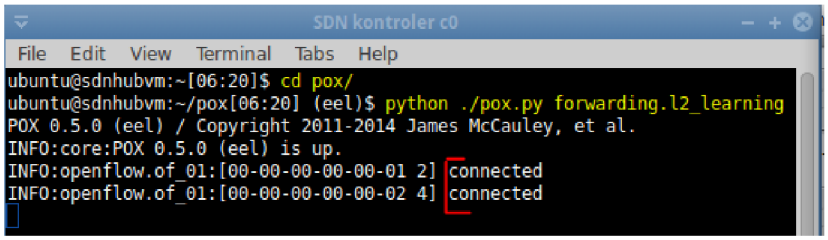
7. Otevřeme si nově vytvořený soubor `lab3-ukol2.py` a upravíme topologii tak, aby vyhovovala potřebám našeho zadání. Je nezbytné nastavit zadané IP adresy a přenosové rychlosti linek mezi uzly viz obr. 4.37.
8. Výsledný soubor měl vypadat obdobně viz obr. 4.41. Pokud ano, je naše topologie vytvořená. Soubor uložíme a zavřeme.

```
from mininet.topo import Topo
from mininet.link import TCLink                    a)

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Vytvoreni prepinacu a hostu
        print " *** VYTVARENI POCITACU A HOSTU *** "
        h1 = self.addHost( 'h1', ip="10.0.0.1" )
        h2 = self.addHost( 'h2', ip="20.0.0.2" )    b)
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )

        # Pridat linky mezi prepinaci
        print " *** VYTVARENI LINEK MEZI PREPINACI *** "
        self.addLink( s1 , s2, bw=10 )              c)

        # Pridat linky mezi hosty
        print " *** VYTVARENI LINEK MEZI HOSTY  ***"
        self.addLink( h1, s1, bw=1 )
        self.addLink( h2, s2, bw=1 )                c)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Obr. 4.41: Konfigurace zadané topologie

- a) importování knihovny `TCLink`, která nám dovoluje nastavovat parametry linek jako jsou propustnost, ztrátovost, zpoždění a podobně.
- b) při tvorbě hostů můžeme pomocí funkce `addHost` vložit jako argument i IP adresu ve tvatu `ip="xxx.xxx.xxx.xxx"`
- c) obdobně můžeme při vytváření linek mezi uzly do funkce `addLink` vložit jako argument rozličné QoS parametry linek. Mimo použitou šířku pásma, která je symbolizována parametrem `bw` v jednotkách Mbit/s můžeme přidat zpoždění v milisekundách `delay='5ms'`, velikost fronty v paketech `max_queue_size=1000` či ztrátovost v jednotkách procent `loss=10`.

9. Nyní je potřeba si otevřít dvě terminálová okna. V jednom okně si spustíme SDN kontrolér POX a v druhém okně si v prostředí Mininet spustíme námi vytvořenou topologii. Nejdříve je třeba spustit kontrolér. Pomocí příkazu `cd pox/` se dostaneme do adresáře s kontrolérem a ten následně spustíme příkazem `python ./pox.py forwarding.l2_learning`. Kontrolér se následně spustí, což je symbolizováno stavem `up` viz Obr. 4.42.



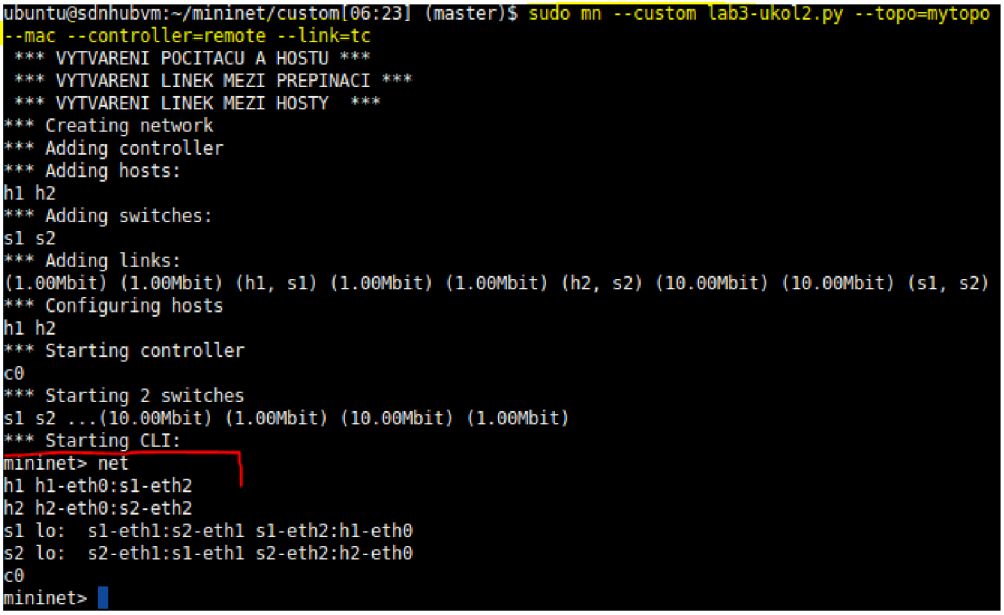
Obr. 4.42: terminálové okno kontroléru

Následně je potřeba spustit naši topologii v Mininetu viz obr. 4.43. Toto uděláme příkazem `sudo mn --custom lab3-ukol2.py --topo=mytopo --mac --controller=remote --link=tc`.

- `--controller=remote` - parametr, kterým se připojíme k spuštěnému kontroléru
- `--link=tc` - parametr, kterým povolíme použití knihovny, pomocí které jsme nastavovali parametry linky

Správnost námi vytvořené topologie zkontrolujeme příkazem `net`, který nám ukáže, jakým způsobem jsou jednotlivé uzly propojeny viz obr. 4.43 (označeno červenou barvou).

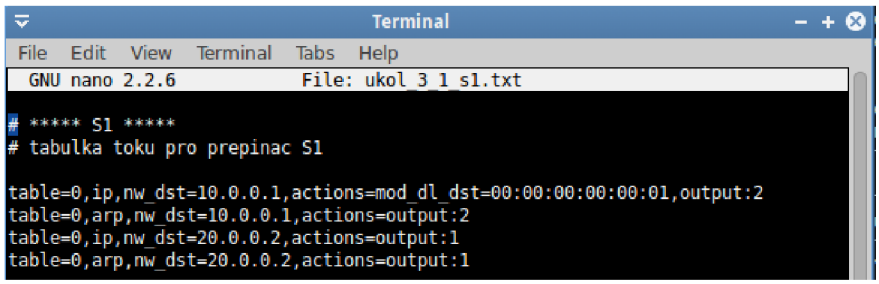
10. Nyní vyzkoušejme příkaz `pingall`. Co tento příkaz dělá a proč nefunguje?
11. Dalším nezbytným krokem je nastavení výchozích cest u jednotlivých stanic a nakonfigurování jejich `arp` záznamů. V prostředí Mininetu postupně zadáme následující příkazy.



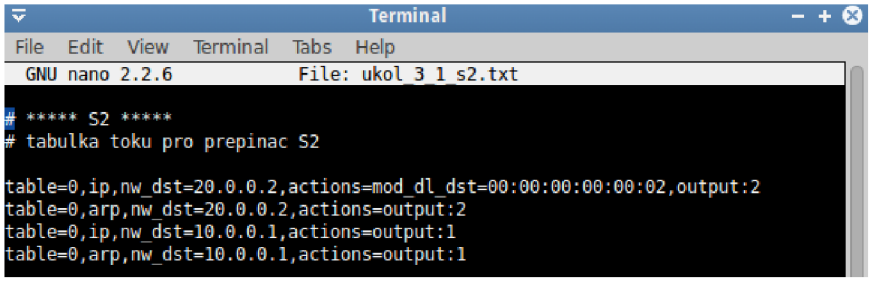
Obr. 4.43: Spuštění vlastní topologie v prostřední Mininet

```
h1 route ad default gw 10.0.0.254 h1-eth0
h1 arp -s 10.0.0.254 00:00:00:00:11:11

h2 route ad default gw 20.0.0.254 h2-eth0
h2 arp -s 20.0.0.254 00:00:00:00:22:22
h2 sudo python -m SimpleHTTPServer 80 &
```

12. Aby nám fungovala požadovaná konektivita v síti bude potřeba naplnit tabulky toků jednotlivým přepínačům `s1` a `s2`. V předešlé laboratorní úloze jsme se seznámili s tím, že místo postupného vkládání jednotlivých záznamů do tabulky můžeme tyto záznamy do přepínačů vložit najednou prostřednictvím textového souboru, ve kterém budou jednotlivá pravidla.
13. Pro přepínač `s1` si vytvoříme textový soubor `lab3-ukol2-s1.txt` obdobně pak pro přepínač `s2`. V těchto souborech budou jednotlivé záznamy toků, které nám zajistí požadovanou konektivitu mezi stanicemi.
14. Nyní budeme potřebovat naplnit naše textové soubory pro přepínače následujícími konfiguracemi viz obr. 4.44 a obr. 4.45. Tyto konfigurace pak v podobě textových souborů nahrajeme do jednotlivých zařízení. Zamyslete se nad tím co jednotlivé záznamy v tabulkách znamenají a k čemu slouží.



Obr. 4.44: Tabulka toků pro přepínač s1



Obr. 4.45: Tabulka toků pro přepínač s2

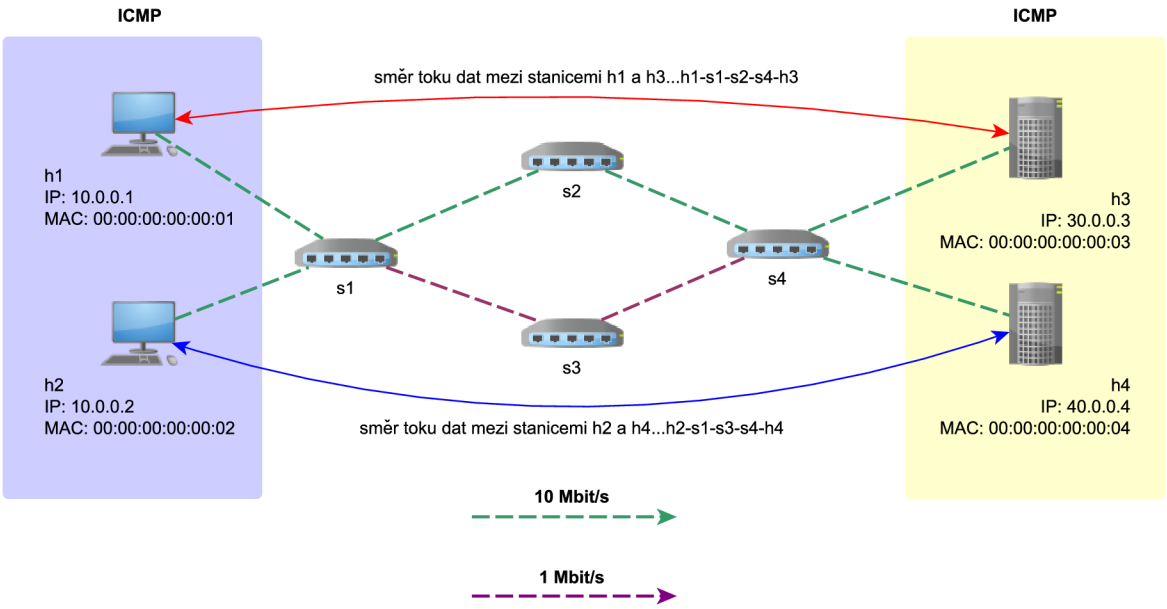
15. Následujícími příkazy můžeme spravovat tabulky toků na jednotlivých přepínačích.

```
s1 ovs-ofctl dump-flows "s1"
                #zobrazí tabulku toků v přepínači s1
s1 ovs-ofctl add-flows "s1" tabulka_toku.txt
                #přidá do tabulky obsah souboru *.txt
s1 ovs-ofctl del-flows "s1"
                #smaže tabulku toků přepínače s1
```

16. Nyní vyzkoušejte konektivitu v síti. Jaký je výsledek?
17. Vyzkoušejte dotaz na webový server běžícím na stanici `h2`. Použijte přitom příkaz `curl`. V případě nejasností na internetu vyhledejte, jakým způsobem příkaz pracuje.
18. V případě, že jste s úkolem hotovi, prezentujte vyučujícímu funkčnost úlohy a odpovězte na případné dotazy.

Úkol č. 3

V tomto úkolu si samostatně vytvoříte topologii viz obr. 4.46 a nastavíte požadované politiky v síti. Vycházejte z předešlého úkolu a využijte znalostí, které jste získali v předešlých laboratorních úlohách.



Obr. 4.46: Schéma topologie pro samostatný úkol

- stanice `h1` a `h2` mezi sebou pomocí `icmp` zpráv
- stanice `h3` a `h4` mezi sebou pomocí `icmp` zpráv
- stanice `h1` bude moci komunikovat pomocí `icmp` zpráv se stanicí `h3` a to skrze cestu přes přepínače `s1-s2-s4`
- stanice `h2` bude moci komunikovat pomocí `icmp` zpráv se stanicí `h4` a to skrze cestu přes přepínače `s1-s3-s4`
- stanice `h1` nebude moci komunikovat s `h4`
- stanice `h2` nebude moci komunikovat s `h3`
- linky mezi `s1, s2, s4` budou mít kapacitu 10 Mbit/s
- linky mezi `s1, s3, s4` budou mít kapacitu 1 Mbit/s
- stanice budou připojeny k přepínači linkou s kapacitou 10 Mbit/s

Malá nápověda k řešení úkolu:

- Nejdříve zprovozněte levou větev sítě. To znamená zprovozněte konektivitu mezi stanicemi `h1` a `h2`. Následně to samé udělejte s pravou větví sítě a nastavte konektivitu mezi `h3` a `h4`.

- Poté se pusťte do nastavování požadovaného toku dat mezi levou a pravou částí sítě.
- V topologii jsou 4 přepínače, proto budeme potřebovat 4 tabulky toků

4.6.6 Kontrolní otázky

1. K čemu slouží příkaz `net`?
2. Proč v úkolu č. 2, bod 10 nefunguje příkaz `pingall`?
3. Jakým příkazem dojde k zobrazení tabulky toků na přepínači `s2`?
4. V samostatném úkolu č. 3 vyzkoušejte příkaz `iperf` mezi stanicemi `h1-h3` a `h2-h4`. Jaké hodnoty jste takto získali?

4.6.7 Seznam zkratek

ARP Address Resoulution Protocol
IP Internet Protocol
QoS Quality of Service
SDN Software Defined Network

4.6.8 Literatura

[1] OpenNetworking.org, *Software defined networking: The New Norm for Networks.* [online]. [cit. 17. 10. 2015]. Dostupné z URL: <`https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`>.

[2] Mininet.org, *Mininet Walkthrough* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <`http://mininet.org/walkthrough/`>.

5 ZÁVĚR

V této diplomové práci byla zaměřena pozornost na novou architekturu softwarově definovaných sítí. V teoretické části byla podrobně tato architektura rozebrána včetně všech tří vrstev, mezi které se řadí vrstva infrastruktury, řídící a aplikační vrstva. Byla zde popsána a vysvětlena důležitost role SDN kontroléru, jakožto centrálního a nejdůležitějšího prvku celé sítě, který je zodpovědný za chování podřízených síťových zařízení, která jsou k němu připojena. Diskutována zde byla i nezastupitelná role OpenFlow protokolu, jakožto jednoho ze základních kamenů softwarově definovaných sítí. Tento protokol je jediným standardizovaným prvkem celé architektury a zajišťuje komunikaci mezi SDN kontrolérem a k němu připojeným hardwarovým zařízením.

Architektura softwarově definovaných sítí se zdá být důstojným nástupcem tradiční síťové architektury oproti níž přináší řadu výhod co se týče zjednodušení správy sítě, možností preciznějšího nastavení toků v síti či rychlejšího zavádění nových služeb a aplikací na základě klientských požadavků. Velcí hráči jako jsou Google, Facebook již tuto technologii používají ve svých datových centrech po celém světě.

Jako každá nová technologie, tak i softwarově definované sítě s sebou nesou i jisté nevýhody. Jednou z nich mohou být různé verze implementací kontrolérů a aplikací, které fungují nad nimi. Mnozí z předních výrobců, byť jsou v rámci jedné organizace Open Networking Foundation, totiž vyvíjí své verze proprietárních kontrolérů v návaznosti na své komerční portfolium služeb. Jednou z dalších nevýhod je pak laxnost firem a organizací měnit cokoliv na již zaběhnuté architektuře, která již jistým způsobem funguje a nemožnost docenit výhody, které tato nová architektura přináší. Na druhou stranu se každoročně zvětšuje množství zákazníků zajímající se o technologii SDN. Jedná se hlavně o datová centra, poskytovatele internetu a větší korporace.

V této diplomové práci byla rozebrána a popsána problematika tvorby aplikací pro softwarově definované sítě, konkrétně pak pro SDN kontrolér RYU. V další části byl pak kladen důraz na vytvoření série laboratorních úloh, které by vhodnou formou představily tuto architekturu softwarově definovaných sítí, ukázaly jejich výhody, nevýhody a možnosti jejich nasazení. V laboratorních úlohách se studenti seznámí s těmito sítěmi i po praktické stránce a vyzkouší si jejich konfiguraci v šesti úkolech, ve kterých si ověří nabyté teoretické znalosti a vyzkouší si jak se tyto sítě chovají ve virtualizované prostředí síťového emulátoru Mininet. Součástí těchto laboratorních úloh jsou pak vzorové laboratorní protokoly a dokumentace pro vyučujícího k jednotlivým úlohám.

LITERATURA

[1] BRAUN, Wolfgang a MENTH Daniel. *Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices* [online]. In: Future Internet Journal, Volume 6, 2014-05-12, s. 302-306 [cit. 20. 11. 2015]. Dostupné z URL: <http://www.mdpi.com/1999-5903/6/2/302/pdf>.

[2] EVANS, Steve. *The history of OpenFlow* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <http://www.computerweekly.com/feature/The-history-of-OpenFlow>.

[3] GUANG YAO, JUN BI a LUYI GUO. *On the cascading failures of multi-controllers in Software Defined Networks.* In: 2013 21st IEEE International Conference on Network Protocols (ICNP) [online]. IEEE, 2013, 2015-11-18, s. 1-2 [cit. 18. 11. 2015]. ISBN 978-1-4799-1270-4. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6733624>.

[4] HIROTSU, Darien. *REST APIs in SDN: An introduction for network engineers* [online]. 2015 [cit. 2016-03-17]. Dostupné z: <http://searchsdn.techtarget.com/tip/REST-APIs-in-SDN-An-introduction-for-network-engineers>.

[5] JIMENEZ, Yury, Cristina CERVELLO-PASTOR a Aurelio J. GARCIA. *On the controller placement for designing a distributed SDN control layer.* In: 2014 IFIP Networking Conference [online]. IEEE, 2014, 2015-11-14, s. 1-9 [cit. 2015-11-14].ISBN 978-3-901882-58-6. Dostupné z URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6857117>.

[6] KLEVIANSKY, Daniel *Separation of Control Plane and Data Plane in Performance Networks.* [online]. [cit. 17. 10. 2015]. Dostupné z URL: <http://danielkleviansky.com/separation-of-control-plane-and-data-plane-in-performance-networks/>.

[7] KRISHNAMURTHY, Anand, Shoban P. CHANDRABOSE a Aaron GEMBER-JACOBSON. Pratyaastha. In: *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN '14* [online]. New York, New York, USA: ACM Press, 2014, 2015-11-18, s. 133-138 [cit. 18. 11. 2015]. ISBN 9781450329897. Dostupné z URL: <http://dl.acm.org/citation.cfm?doid=2620728.2620748>.

[8] MeruNetworks.com, *An introduction to SDN.* [online]. [cit. 19. 10. 2015]. Dostupné z URL: <http://www.merunetworks.com/collateral/solution-briefs/an-introduction-to-sdn-sb.pdf>.

[9] NADEAU, Thomas D a Kenneth GRAY. *SDN: software defined networks.* 1st ed. Sebastopol, CA: O'Reilly Media, 2013, xxvii, 352 s.[cit. 17. 11. 2015]. ISBN 978-1-4493-4230-2.

[10] OpenNetworking.org, *OpenFlow Switch Specification v.1.5.0* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>.

[11] OpenNetworking.org, *Software defined networking: The New Norm for Networks.* [online]. [cit. 17. 10. 2015]. Dostupné z URL: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>.

[12] PlVision.eu, *Software-Defined Networking.* [online]. [cit. 19. 10. 2015]. Dostupné z URL: <http://plvision.eu/expertise/networking/software-defined-networking/>.

[13] SAKYA, Rumus. *Benefits of SDN – Solving Network Infrastructure's Root-Canal Problem* [online]. [cit. 17. 11. 2015]. Dostupné z URL: <https://edgewaternetworks.com/2014/07/benefits-sdn-solving-network-infrastructures-root-canal-problem/>.

[14] SALISBURY, Brent. *OpenFlow: Proactive vs Reactive Flows* [online]. [cit. 19. 11. 2015]. Dostupné z URL: <http://networkstatic.net/openflow-proactive-vs-reactive-flows/>.

[15] SHARMA, Nirmal. *Eight Big Benefits of Software-Defined Networking* [online]. [cit. 17. 11. 2015]. Dostupné z URL: <http://www.serverwatch.com/server-tutorials/eight-big-benefits-of-software-defined-networking.html>.

[16] STALLING, William. *Software-Defined Networks and OpenFlow* [online]. In: The Internet Protocol Journal, Volume 16, 2013-03, s. 2-14 [cit. 19. 11. 2015]. Dostupné z URL: <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/ipj_16-1.pdf>.

[17] RYU PROJECT TEAM. *RYU SDN Framework* [online]. [cit. 19. 3. 2016]. Dostupné z URL: <https://osrg.github.io/ryu-book/en/Ryubook.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACL Access Control List

API Application Programming Interface

BGP Border Gateway Protocol

ESP Encrypted Security Payload

ICMP Internet Control Message Protocol

IPv6 Internet Protocol version 6

ISO/OSI International Standards Organization / Open Systen Interconnection

JSON JavaScript Object Notation

MPLS Multiprotocol Label Switching

OF OpenFlow

OSPF Open Shortest Path Protocol

QoS Quality of Service

REST Representational State Transfer

RIP Routing Information Protocol

SDN Software Defined Network

TCP Transmission Control Protocol

TLS Transport Layer Security

URI Uniform Resource Identifier

SEZNAM PŘÍLOH

A VZOROVÝ PROTOKOL K ÚLOZE Č. 1 – ÚVOD DO SOFTWAROVĚ DEFINOVANÝCH SÍTÍ

A.1 Laboratorní úloha č. 1 – Úvod do softwarově definovaných sítí

Datum:	15. 4. 2016	**Jméno a Příjmení:**	Vojtěch Piska	**VUT ID:**	136039

A.2 Cíl

Cílem této laboratorní úlohy by mělo být nahlédnutí do problematiky softwarově definovaných sítí – SDN[1]. Seznámíme se s jejich architekturou, protokolem OpenFlow a také se síťovým emulátorem Mininet, ve kterém budou všechny naše simulace prováděny.

A.3 Úkoly

1. Seznamte se s pojmem softwarově definovaná síť.
2. Nastudujte roli kontroléru a protokolu OpenFlow.
3. Seznamte se síťovým emulátorem Mininet a prozkoumejte jeho možnosti.
4. Dle zadání vytvořte SDN topologii a analyzujte pomocí analyzátoru Wireshark OpenFlow zprávy.
5. Odpovězte na kontrolní otázky.

A.4 Kontrolní otázky

1. Popište jednotlivé vrstvy architektu SDN a stručně popište jejich funkci.
2. Jaký je rozdíl mezi zprávami `Packet-In` a `Packet-Out`.
3. Vysvětlete rozdíl mezi *proaktivním*, *reaktivním* a *hybridním* režimem kontroléru.
4. Co je tzv. *Northbound* a *Southbound*.

[1] SDN - Software Defined Networks

A.5 Odpovědi na kontrolní otázky

1. Jedná se o třívrstvý model, který se skládá z následujících částí:
   - **Vrstva infrastruktury:** Tato vrstva obsahuje tradiční hardwarové prvky jako jsou přepínače či směrovače. Tyto prvky musí podporovat protokol OpenFlow, kterým komunikují s kontrolérem. Komunikace mezi kontrolérem a podřízenými prvky bývá označováno jako takzvané „jižní rozhraní“ neboli „Southbound“.
   - **Řídící vrstva:** V této vrstvě se nachází kontrolér, který je centrálním prvkem sítě. Udržuje si přehled o stavu sítě, provádí abstrakci pro nad ním bežící aplikace, řídí činnost všech podřízených zařízení.
   - **Aplikační vrstva:** Obsahuje aplikace, které běží na kontroléru. Takovými aplikacemi může být například firewall, load-balancer, monitor stavu sítě a podobně. Aplikační vrstva díky kontroléru zprostředkovaný pohled na síť a může jednotlivé prvky skrze kontrolér programovat. Aplikační vrstva komunikuje s kontrolérem skrze takzvané „severní rozhraní“ neboli „Northbound“.
2. - `Packet-In` zpráva - Jedná se o zprávy, které jsou zasílány jednotlivými prvky kontroléru v případě, že se žádný záznam jejich tabulky toků neshoduje s příchozí datovou jednotkou. Zapouzdří tedy neznámou datovou jednotkou do této `Packet-In` zprávy a dotážou se kontroléru, jak s takovou datovou jednotkou naložit.
   - `Packet-Out` zpráva - Jedná se odpověď na zprávu `Packet-In`. V této zprávě kontrolér odesílá podřízeným prvkům informace jak naložit s danou datovou jednotkou, na kterou se ho v odpovídající zprávě `Packet-In` dotazovaly.
3. - **Reaktivní režim:** Přepínač v tomto režimu funguje tím způsobem, že OpenFlow tabulky jsou ze začátku prázdné a neobsahují žádné záznamy o tom, jak s daným rámcem zacházet. Při přijetí rámce pak přepínač vytvoří zprávu typu *Packet-In*, ve které se dotáže kontroléru jak s tímto rámcem zacházet. Ten mu požadované informace nejen zašle, ale zajistí naprogramování OpenFlow tabulky tak, že pro další příchozí rámec stejného typu již bude přepínač vědět jak s rámcem zacházet a nebude se již muset znovu dotazovat kontroléru. Tento režim dává kontroléru dobrý přehled o dění v síti. Mírnou nevýhodou může být vyšší zpoždění v případě rozsáhlejších sítí.
   - **Proaktivní režim:** Tento režim funguje tak, že kontrolér dopředu nastaví všechny záznamy v OpenFlow tabulce pro všechny možné situace na základě svých informací. OpenFlow tabulka tak při příchodu rámce

již ví jak s daným rámcem naložit.

- **Hybridní režim:** Tento režim umožňuje předchozí zmiňované techniky kombinovat a dokonce využít i využít lokální *řídíci rovinu* přepínače a to třeba v případě, že OpenFlow tabulka neobsahuje žádný záznam, který by odpovídal přijatému rámci. Přepínač se tak může rozhodnout, že jeho zpracování zajistí klasickým způsobem.[4]

4.
- **Southbound:** Jedná se o takzvané „jižní rozhraní“. Toto rozhraní je mezi infrastrukturní a řídící vrstvou SDN architektury. Jedná se o rozhraní, kterým spolu komunikují jednotlivé síťové prvky a kontrolér. Komunikuje se skrze OpenFlow protokol.
- **Northbound:** Toto rozhraní se také nazývá „severní“. Nachází se mezi řídící vrstvou a vrstvou aplikační. Komunikace těchto dvou vrstev probíhá skrze API (Application Programming Interface). Toto rozhraní není standardizované, a proto může být v kontrolérech různých výrobců implementováno odlišně.

A.6 Závěr

V laboratorní úloze jsme se seznámili s principy architektury softwarově definovaných sítí, jejichž základem je SND kontrolér jako řídící část sítě, který ovládá podřízená síťová zařízení skrze protokol OpenFlow. Dále jsme se seznámili se síťovým emulátorem Mininet, ve kterém jsme simulovali SDN topologii zadanou v úkolu číslo 4. V tomto úkolu jsme analyzovali jednotlivé typy OpenFlow zpráv pomocí protokolového analyzátoru Wireshark a ověřili jejich funkce. Následně jsme odpověděli na kontrolní otázky k této laboratorní úloze.

B DOKUMENTACE PRO VYUČUJÍCÍHO K ÚLOZE Č. 1 – ÚVOD DO SOFTWAROVĚ DEFINOVANÝCH SÍTÍ

B.1 Odpovědi na kontrolní otázky

1. Jedná se o třívrstvý model, který se skládá z následujících částí:
   - **Vrstva infrastruktury:** Tato vrstva obsahuje tradiční hardwarové prvky jako jsou přepínače či směrovače. Tyto prvky musí podporovat protokol OpenFlow, kterým komunikují s kontrolérem. Komunikace mezi kontrolérem a podřízenými prvky bývá označováno jako takzvané „jižní rozhraní“ neboli „Southbound“.
   - **Řídící vrstva:** V této vrstvě se nachází kontrolér, který je centrálním prvkem sítě. Udržuje si přehled o stavu sítě, provádí abstrakci pro nad ním bežící aplikace, řídí činnost všech podřízených zařízení.
   - **Aplikační vrstva:** Obsahuje aplikace, které běží na kontroléru. Takovými aplikacemi může být například firewall, load-balancer, monitor stavu sítě a podobně. Aplikační vrstva díky kontroléru zprostředkovaný pohled na síť a může jednotlivé prvky skrze kontrolér programovat. Aplikační vrstva komunikuje s kontrolérem skrze takzvané „severní rozhraní“ neboli „Northbound“.
2. - `Packet-In` zpráva - Jedná se o zprávy, které jsou zasílány jednotlivými prvky kontroléru v případě, že se žádný záznam jejich tabulky toků neshoduje s příchozí datovou jednotkou. Zapouzdří tedy neznámou datovou jednotkou do této `Packet-In` zprávy a dotážou se kontroléru, jak s takovou datovou jednotkou naložit.
   - `Packet-Out` zpráva - Jedná se odpověď na zprávu `Packet-In`. V této zprávě kontrolér odesílá podřízeným prvkům informace jak naložit s danou datovou jednotkou, na kterou se ho v odpovídající zprávě `Packet-In` dotazovaly.
3. - **Reaktivní režim:** Přepínač v tomto režimu funguje tím způsobem, že OpenFlow tabulky jsou ze začátku prázdné a neobsahují žádné záznamy o tom, jak s daným rámcem zacházet. Při přijetí rámce pak přepínač vytvoří zprávu typu *Packet-In*, ve které se dotáže kontroléru jak s tímto rámcem zacházet. Ten mu požadované informace nejen zašle, ale zajistí naprogramování OpenFlow tabulky tak, že pro další příchozí rámec stejného typu již bude přepínač vědět jak s rámcem zacházet a nebude se

již muset znovu dotazovat kontroléru. Tento režim dává kontroléru dobrý přehled o dění v síti. Mírnou nevýhodou může být vyšší zpoždění v případě rozsáhlejších sítí.

- **Proaktivní režim:** Tento režim funguje tak, že kontrolér dopředu nastaví všechny záznamy v OpenFlow tabulce pro všechny možné situace na základě svých informací. OpenFlow tabulka tak při příchodu rámce již ví jak s daným rámcem naložit.
- **Hybridní režim:** Tento režim umožňuje předchozí zmiňované techniky kombinovat a dokonce využít i využít lokální *řídící rovinu* přepínače a to třeba v případě, že OpenFlow tabulka neobsahuje žádný záznam, který by odpovídal přijatému rámci. Přepínač se tak může rozhodnout, že jeho zpracování zajistí klasickým způsobem.[4]

4.
- **Southbound:** Jedná se o takzvané „jižní rozhraní“. Toto rozhraní je mezi infrastrukturní a řídící vrstvou SDN architektury. Jedná se o rozhraní, kterým spolu komunikují jednotlivé síťové prvky a kontrolér. Komunikuje se skrze OpenFlow protokol.
- **Northbound:** Toto rozhraní se také nazývá „severní“. Nachází se mezi řídící vrstvou a vrstvou aplikační. Komunikace těchto dvou vrstev probíhá skrze API (Application Programming Interface). Toto rozhraní není standardizované, a proto může být v kontrolérech různých výrobců implementováno odlišně.

C VZOROVÝ PROTOKOL K ÚLOZE Č. 2 – VÍCENÁSOBNÉ TABULKY TOKŮ A ACCESS CONTROL MANAGEMENT V SÍTI

C.1 Laboratorní úloha č. 2 – Vícenásobné tabulky toků a Access Control Management v síti.

Datum:	15. 4. 2016	**Jméno a Příjmení:**	Vojtěch Piska	**VUT ID:**	136039

C.2 Cíl

Cílem této laboratorní úlohy by mělo být nahlédnutí do problematiky zpracování datových jednotek v OpenFlow přepínači v rámci více tabulek toků. Dále pak využití přepínače jako Access Control prvku, který stanovuje pravidla pro přístup k jednotlivým službám a uzlům v rámci sítě.

C.3 Úkoly

1. Seznamte se způsobem zpracování datových jednotek v OpenFlow přepínači.
2. Nastudujte možnosti klasifikace datových jednotek a možnosti jejich zpracování, které OpenFlow nabízí.
3. Vytvořte zadanou topologii sítě a ověřte její funkčnost.
4. Podle zadání vytvořte topologii sítě, nastavte Access Control pravidla a ověřte jejich funkčnost.
5. Odpovězte na kontrolní otázky.

C.4 Kontrolní otázky

1. Popište zpracování datové jednotky při příchodu do OpenFlow přepínače.
2. Jaký je rozdíl mezi zpracováním v rámci jedné tabulky toků a mezi více tabulkami.
3. Jaký smysl má nastavovat Access Control Management pravidla v síti.

C.5 Odpovědi na kontrolní otázky

1. Přepínač při příchodu kontroluje záhlaví datové jednotky a testuje ho na shodu s jednotlivými záznamy tabulky toků. Jakmile je shoda nalezena, datové jednotce je přiřazena odpovídající akce a podle ní proběhne zpracování.
2. Zpracování pomocí jedné tabulky bylo podporováno již v OpenFlow specifikaci 1.0. Příchozí jednotka je porovnávána vůči jednotlivým záznamům v tabulce toků. V případě, že je shoda nalezena je přiřazena odpovídající akce a datová jednotka podle ní zpracována. Když shoda není, je bud odeslána na kontrolér nebo zahozena. V případě více tabulek toků, které byly představeny ve specifikaci 1.1. Probíhá hledání shody v `tabulce 0` až `n`. Je zde specifikován nový typ akce `resubmit()`, v jejímž argumentu se nachází číselná hodnota tabulky, do které se má při nalezení shody přeskočit. Přeskakovat se přitom dá jen do tabulky vyšší. Velkou výhodou zpracování ve více tabulkách, je že v každé tabulce můžeme mít nastavena rozličná pravidla pro specifické služby či části sítě. V první tabulce třeba můžeme mít `ACL` pravidla, v další pravidla pro `NAT` zatímco v další mohou být informace pro přepínání a směrování.
3. Více méně v každé sítě mají uživatelé různé postavení a z toho vyplývající i různá práva a omezení pro přístup ke zdrojům v síti. Například uživatelé z vývojového oddělení nebudou mít přístup na servery, kde se nachází data o finančních informacích z finančního oddělení. Na druhou stranu budou mít přístup k serverům testovacího oddělení, kde sdílí vývojové a testovací oddělení potřebné soubory.

C.6 Závěr

V druhé laboratorní úloze jsme se dozvěděli podle jakých kritérií můžeme třídit jednotlivé datové jednotky a jak využít více tabulek toků v rámci jednoho přepínače k hierarchičtějšímu nastavení daných požadavků. V třetím úkolu jsme vytvořili zadanou topologii a nastavili požadovanou funkčnost. Ve čtvrtém úkolu jsme pak nastavili požadovanou topologii podle nových požadavků a funkčnost nechali zkontrolovat vyučujícím. Na závěr jsme pak odpověděli na kontrolní otázky, které se vázaly k této úloze.

D DOKUMENTACE PRO VYUČUJÍCÍHO K ÚLOZE Č. 2 – VÍCENÁSOBNÉ TABULKY TOKŮ A ACCESS CONTROL MANAGEMENT V SÍTI.

D.1 Odpovědi na kontrolní otázky

1. Přepínač při příchodu kontroluje záhlaví datové jednotky a testuje ho na shodu s jednotlivými záznamy tabulky toků. Jakmile je shoda nalezena, datové jednotce je přiřazena odpovídající akce a podle ní proběhne zpracování.
2. Zpracování pomocí jedné tabulky bylo podporováno již v OpenFlow specifikaci 1.0. Příchozí jednotka je porovnávána vůči jednotlivým záznamům v tabulce toků. V případě, že je shoda nalezena je přiřazena odpovídající akce a datová jednotka podle ní zpracována. Když shoda není, je bud odeslána na kontrolér nebo zahozena. V případě více tabulek toků, které byly představeny ve specifikaci 1.1. Probíhá hledání shody v `tabulce 0` až `n`. Je zde specifikován nový typ akce `resubmit()`, v jejímž argumentu se nachází číselná hodnota tabulky, do které se má při nalezení shody přeskočit. Přeskakovat se přitom dá jen do tabulky vyšší. Velkou výhodou zpracování ve více tabulkách, je že v každé tabulce můžeme mít nastavena rozličná pravidla pro specifické služby či části sítě. V první tabulce třeba můžeme mít `ACL` pravidla, v další pravidla pro `NAT` zatímco v další mohou být informace pro přepínání a směrování.
3. Více méně v každé sítě mají uživatelé různé postavení a z toho vyplývající i různá práva a omezení pro přístup ke zdrojům v síti. Například uživatelé z vývojového oddělení nebudou mít přístup na servery, kde se nachází data o finančních informacích z finančního oddělení. Na druhou stranu budou mít přístup k serverům testovacího oddělení, kde sdílí vývojové a testovací oddělení potřebné soubory.

D.2 Řešení samostatného úkolu č. 4

Vytvoření zadané topologie sítě

```
sudo mn --topo=single,5 --controller remote --link tc,bw=100 --mac
```

Konfigurace jednotlivých uzlů v síti

```
h1 route add default gw 10.0.0.254 h1-eth0
h1 arp -s 10.0.0.254 00:00:00:00:11:11
```

```
h2 route add default gw 10.0.0.254 h2-eth0
h2 arp -s 10.0.0.254 00:00:00:00:22:22

h3 route add default gw 10.0.0.254 h3-eth0
h3 arp -s 10.0.0.254 00:00:00:00:33:33

h4 ifconfig h4-eth0 40.0.0.4 netmask 255.255.255.0
h4 route add default gw 40.0.0.254 h4-eth0
h4 arp -s 40.0.0.254 00:00:00:00:44:44
h4 sudo python -m SimpleHTTPServer 80 &

h5 ifconfig h5-eth0 50.0.0.5 netmask 255.255.255.0
h5 route add default gw 50.0.0.254 h5-eth0
h5 arp -s 50.0.0.254 00:00:00:00:55:55
h5 sudo python -m SimpleHTTPServer 80 &
```

Příkazy pro práci s tabulkou toků

Zobrazení tabulek toků na přepínači

```
s1 ovs-ofctl dump-flows "s1"
```

Vložení záznamů do tabulky souborů z textového souboru

```
s1 ovs-ofctl add-flows "s1" jmeno_souboru.txt
```

Vymazání obsahu tabulky toků z přepínače

```
s1 ovs-ofctl del-flows "s1"
```

Obsah textového souboru s jednotlivými záznamy

```
#tabulka 0 - Access Control

#povoleni vzajemne komunikace v 10.0.0.0/24 siti
table=0,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=resubmit(,1)
table=0,arp,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=resubmit(,1)

#blokování komunikace na stanici h3
table=0,priority=50000, icmp,nw_src=10.0.0.1,nw_dst=10.0.0.3,actions=drop
table=0,priority=50000, icmp,nw_src=10.0.0.2,nw_dst=10.0.0.3,actions=drop
table=0,arp,nw_src=10.0.0.0/24,nw_dst=10.0.0.3,actions=drop

#h1 tcp80 na h5
```

```
table=0,tcp,nw_src=10.0.0.1,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)

#h2 tcp80 na h4
table=0,tcp,nw_src=10.0.0.2,nw_dst=40.0.0.4,tcp_dst=80,actions=resubmit(,1)

#h3 icmp,tcp80 na h4 a h5
table=0,tcp,nw_src=10.0.0.3,nw_dst=40.0.0.4,tcp_dst=80,actions=resubmit(,1)
table=0,icmp,nw_src=10.0.0.3,nw_dst=40.0.0.4, actions=resubmit(,1)
table=0,tcp,nw_src=10.0.0.3,nw_dst=50.0.0.5,tcp_dst=80,actions=resubmit(,1)
table=0,icmp,nw_src=10.0.0.3,nw_dst=50.0.0.5,actions=resubmit(,1)

table=0,icmp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=resubmit(,1)
table=0,tcp,nw_src=40.0.0.4,nw_dst=50.0.0.5,actions=resubmit(,1)

table=0,ip,nw_src=40.0.0.4,actions=resubmit(,1)
table=0,ip,nw_src=50.0.0.5,actions=resubmit(,1)
table=0,priority=0,actions=drop

# table 1 -- Přepínání a Směrování
table=1,ip,nw_dst=10.0.0.1,actions=mod_dl_dst=00:00:00:00:00:01,output:1
table=1,ip,nw_dst=10.0.0.2,actions=mod_dl_dst=00:00:00:00:00:02,output:2
table=1,ip,nw_dst=10.0.0.3,actions=mod_dl_dst=00:00:00:00:00:03,output:3
table=1,ip,nw_dst=40.0.0.4,actions=mod_dl_dst=00:00:00:00:00:04,output:4
table=1,ip,nw_dst=50.0.0.5,actions=mod_dl_dst=00:00:00:00:00:05,output:5

table=1,priority=0,arp,nw_dst=10.0.0.1,actions=output:1
table=1,priority=0,arp,nw_dst=10.0.0.2,actions=output:2
table=1,priority=0,arp,nw_dst=10.0.0.3,actions=output:3
table=1,priority=0,arp,nw_dst=40.0.0.4,actions=output:4
table=1,priority=0,arp,nw_dst=50.0.0.5,actions=output:5
```

E VZOROVÝ PROTOKOL K ÚLOZE Č. 3 – TVORBA VLASTNÍ TOPOLOGIE A POLICY-BASED ROUTING

E.1 Laboratorní úloha č. 3 – Tvorba vlastní topologie a policy-based routing

Datum:	15. 4. 2016	**Jméno a Příjmení:**	Vojtěch Piska	**VUT ID:**	136039

E.2 Cíl

V této laboratorní úloze se podíváme jak vytvářet vlastní topologie v prostředí Mininet, kde se okrajově seznámíme se skriptovacím jazykem Python. V další části si pak vytvoříme zadanou topologii a nastavíme politiky v síti dle zadaných požadavků. Nově se zde setkáme s pojmem Policy-based Routing, který je v SDN sítích hojně využíván. Použijeme přitom již nabyté znalosti z předchozích úloh.

E.3 Úkoly

1. Seznamte se s teoretickým úvodem k této úloze, kde se zaměřte na to jakým způsobem tvořit vlastní topologie a na informace týkající se Policy-based Routingu.
2. Vytvořte zadanou topologii sítě a nastavte její chování podle zadání.
3. Samostatně vytvořte zadanou topologii, nastavte její chování podle zadání. Následně pak demonstrujte její chování vyučujícímu.

E.4 Kontrolní otázky

1. K čemu slouží příkaz `net`?
2. Proč v úkolu č. 2, bod 10 nefunguje příkaz `pingall`?
3. Jakým příkazem dojde k zobrazení tabulky toků na přepínači `s2`?
4. V samostatném úkolu č. 3 vyzkoušejte příkaz `iperf` mezi stanicemi `h1-h3` a `h2-h4`. Jaké hodnoty jste takto získali?

E.5 Odpovědi na kontrolní otázky

1. Příkaz `net` v prostředí Mininetu zobrazí propojení jednotlivých uzlů včetně rozhraní jimiž jsou připojeny.
2. Zatím v tabulkách toků na přepínačích `s1` a `s2` nejsou žádné záznamy. Hosté jsou v jiném subnetu a proto mezi nimi nemůže fungovat konektivita.
3. Tabulku toků na přepínači `s2` zobrazíme zadáním příkazu:

   ```
   s2 ovs-ofctl dump-flows "s2"
   ```

4. Příkaz `iperf` mezí propustnost mezi dvěma uzly v síti. Propustnost by se v našem případě mezi stanicemi `h1-h3` měla blížit 10 Mbit/s, mezi stanicemi `h2-h4` by pak měla dosahovat hranice 1 Mbit/s.

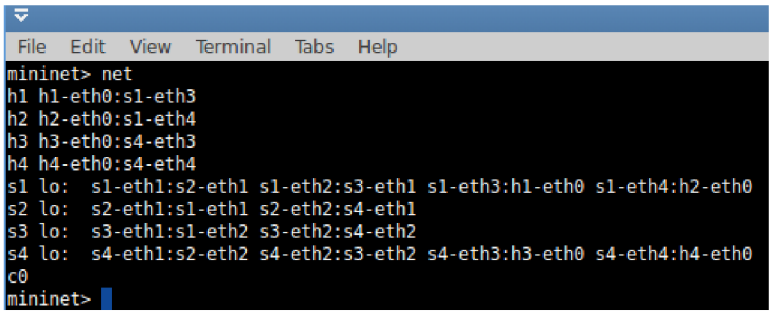
E.6 Závěr

V této laboratorní úloze jsme se dozvěděli, že si v Mininetu můžeme vytvářet i vlastní topologie a nespoléhat jen na topologie, které Mininet již obsahuje. Vyzkoušeli jsme si jak takovou topologii podle zadání vytvořit a spustit. K tvorbě topologie jsme používali jazyk Python, kde jsme se seznámili s jeho základy. V druhé části laboratorní úlohy jsme měli samostatně vytvořit topologii a nastavit síti dané vlastnosti. Prakticky jsme si zde vyzkoušeli Policy-based Routing, který jsme nastavili tak aby komunikace mezi stanicemi h1 a h3 probíhala přes rychlejší 10 Mbitovou linku mezi přepínači s1, s2 a s4, zatímco komunikace mezi stanicemi h2 a h4 probíhala přes pomalejší 1 Mbitovou linku skrze přepínače s1, s3 a s4. Toto jsme prakticky demonstrovali vyučujícímu a na závěr jsme vypracovali odpovědi na kontrolní otázky.

F DOKUMENTACE PRO VYUČUJÍCÍHO K ÚLOZE Č. 3 – TVORBA VLASTNÍ TOPOLOGIE A POLICY-BASED ROUTING

F.1 Odpovědi na kontrolní otázky

1. Příkaz `net` v prostředí Mininetu zobrazí propojení jednotlivých uzlů včetně rozhraní jimiž jsou připojeny



Obr. F.1: Příkaz net

2. Zatím v tabulkách toků na přepínačích `s1` a `s2` nejsou žádné záznamy. Hosté jsou v jiném subnetu a proto mezi nimi nemůže fungovat konektivita.
3. Tabulku toků na přepínači `s2` zobrazíme zadáním příkazu:

```
s2 ovs-ofctl dump-flows "s2"
```

4. Příkaz `iperf` mezí propustnost mezi dvěma uzly v síti. Propustnost by se v našem případě mezi stanicemi `h1-h3` měla blížit 10 Mbit/s, mezi stanicemi `h2-h4` by pak měla dosahovat hranice 1 Mbit/s.

F.2 Řešení samostatného úkolu č. 3

Vytvoření zadané topologie sítě

1. Nejdříve vytvoříme soubor `lab3-ukol3.py` pomocí příkazu `nano lab3-ukol3.py` v prostředí linuxové terminálu.
2. Následně do něj překopírujeme následující příkazy a soubor uložíme. Tímto máme topologii naši sítě vytvořenou.

```
from mininet.topo import Topo
```

```
from mininet.link import TCLink

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Vytvoreni prepinacu a hostu
        h1 = self.addHost( 'h1', ip="10.0.0.1" )
        h2 = self.addHost( 'h2', ip="10.0.0.2" )
        h3 = self.addHost( 'h3', ip="30.0.0.3" )
        h4 = self.addHost( 'h4', ip="40.0.0.4" )
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )

        # Pridat linky mezi prepinaci
        self.addLink( s1 , s2 , bw=10 )
        self.addLink( s2 , s4 , bw=10 )
        self.addLink( s1 , s3 , bw=1 )
        self.addLink( s3 , s4 , bw=1 )

      # Pridat linky mezi hosty
      self.addLink( h1, s1 , bw=10 )
      self.addLink( h2, s1 , bw=10 )
      self.addLink( h3, s4 , bw=10 )
      self.addLink( h4, s4 , bw=10 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

3. Nejdříve ze přejdeme do adresáře `pox` pomocí příkazu `cd pox/` a spustíme SDN kontrolér příkazem `python ./pox.py forwarding.l2_learning` stejně jako v předešlém úkolu.
4. Topologii následně můžeme spustit v emulátoru Mininet. To provede spuštěním

následujícího příkazu v okně linuxového terminálu.

```
#příkaz je na jednom řádku

sudo mn --custom lab3-ukol2.py --topo=mytopo --controller=remote
--link=tc --mac
```

5. Nyní nastavíme dodatečné informace stanicím v již spuštěném mininetu. Postupně zadáme následující příkazy:

```
h1 route add default gw 10.0.0.254 h1-eth0
h1 arp -s 10.0.0.254 00:00:00:00:11:11

h2 route add default gw 10.0.0.254 h2-eth0
h2 arp -s 10.0.0.254 00:00:00:00:22:22

h3 route add default gw 30.0.0.254 h3-eth0
h3 arp -s 10.0.0.254 00:00:00:00:33:33

h4 route add default gw 40.0.0.254 h4-eth0
h4 arp -s 40.0.0.254 00:00:00:00:44:44
```

6. Nyní je potřeba naplnit tabulky toků na jednotlivých přepínačích `s1, s2, s3` a `s4`. Postup popsaný níže je pro přepínač `s1`. U zbylých přepínačů je postup analogický.
   (a) Vytvoříme si nový textový soubor, který bude sloužit jako tabulka toků pro přepínač `s1`. Otevřeme si nové okno v linuxovém terminále a zadáme `nano lab3-ukol3-s1.txt`.
   (b) Do souboru vložíme následující obsah a soubor uložíme.

```
# ***** S1 *****
# tabulka toku pro prepinac S1

##### zprovozneni konektivity mezi hosty h1 a h2
table=0,ip,nw_dst=10.0.0.1,actions=mod_dl_dst=00:00:00
:00:00:01,output:3
table=0,ip,nw_dst=10.0.0.2,actions=mod_dl_dst=00:00:00
:00:00:02,output:4

table=0,arp,nw_dst=10.0.0.1,actions=output:3
table=0,arp,nw_dst=10.0.0.2,actions=output:4
```

```
#trasa h1---s1--s2--s4--h3
table=0,ip,nw_src=10.0.0.1,nw_dst=30.0.0.3,actions=output:1
table=0,arp,nw_src=10.0.0.1,nw_dst=30.0.0.3,actions=output:1

#trasa h2---s1---s3---s4---h4
table=0,ip,nw_src=10.0.0.2,nw_dst=40.0.0.4,actions=output:2
table=0,arp,nw_src=10.0.0.2,nw_dst=40.0.0.4,actions=output:2
```

(c) Nyní se opět vrátíme do terminálového okna a pomocí následujícího příkazu nahrajeme tabulku toků do přepínače `s1`.

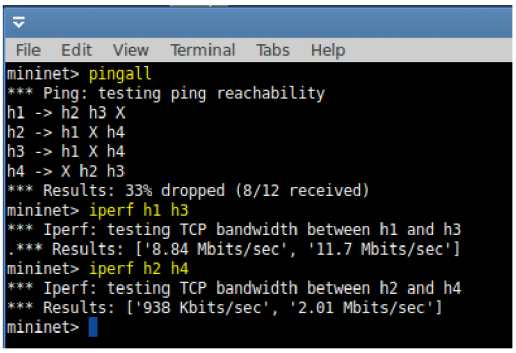
```
#vložení tabulky do přepínače
s1 ovs-ofctl add-flows "s1" lab3-ukol3-s1.txt

#zobrazení záznamů v tabulce toků přepínače
s1 ovs-ofctl dump-flows "s1"

#smazání tabulky toků v přepínači
s1 ovs-ofctl del-flows "s1"
```

7. Celý postup zopakujeme i pro zbylé přepínače `s2, s3, s4`. Níže jsou zobrazeny konfigurace pro jednotlivé přepínače. Uložíme je do analogicky pojmenovaných souborů `lab3-ukol3-s2.txt, lab3-ukol3-s3.txt, lab3-ukol3-s4.txt` a nahrajeme do zbylých přepínačů.

```
# ***** S2 *****
# tabulka toku pro prepinac S2

#trasa s2---s4
table=0,ip,nw_src=10.0.0.1,nw_dst=30.0.0.3,actions=output:2
table=0,arp,nw_src=10.0.0.1,nw_dst=30.0.0.3,actions=output:2

#trasa s4---s2
table=0,ip,nw_src=30.0.0.3,nw_dst=10.0.0.1,actions=output:1
table=0,arp,nw_src=30.0.0.3,nw_dst=10.0.0.1,actions=output:1


# ***** S3 *****
# tabulka toku pro prepinac S3

#trasa s3---s4
```

```
table=0,ip,nw_src=10.0.0.2,nw_dst=40.0.0.4,actions=output:2
table=0,arp,nw_src=10.0.0.2,nw_dst=40.0.0.4,actions=output:2

#trasa s4---s3
table=0,ip,nw_src=40.0.0.4,nw_dst=10.0.0.2,actions=output:1
table=0,arp,nw_src=40.0.0.4,nw_dst=10.0.0.2,actions=output:1


# ***** S4 *****
# tabulka toku pro prepinac S4

##### zprovozneni konektivity mezi hosty h3 a h4
table=0,ip,nw_dst=30.0.0.3,nw_src=40.0.0.4,actions=mod_dl_dst=00:00:00
:00:00:03,output:3
table=0,ip,nw_dst=40.0.0.4,nw_src=30.0.0.3,actions=mod_dl_dst=00:00:00
:00:00:04,output:4
table=0,arp,nw_dst=30.0.0.3,actions=output:3
table=0,arp,nw_dst=40.0.0.4,actions=output:4


#trasa h3--s4--s2--s1--h1
table=0,arp,nw_dst=10.0.0.1,actions=output:1
table=0,ip,nw_dst=10.0.0.1,actions=output:1
table=0,ip,nw_dst=30.0.0.3,nw_src=10.0.0.1,actions=mod_dl_dst=00:00:00
:00:00:03,output:3

#trasa h4--s4--s3--s1--h2
table=0,arp,nw_dst=10.0.0.2,actions=output:2
table=0,ip,nw_dst=10.0.0.2,actions=output:2
table=0,ip,nw_dst=40.0.0.4,nw_src=10.0.0.2,actions=mod_dl_dst=00:00:00
:00:00:04,output:4
```

8. Nyní by již v rámci sítě měla fungovat požadovaná konektivita, což si můžeme ověřit příkazem `pingall`. Příkazem `iperf` ještě zkontrolujeme propustnost tras.



Obr. F.2: Kontrola správnosti samostatného úkolu č. 3

G INSTALACE A KONFIGURACE POUŽITÉHO SOFTWAROVÉHO VYBAVENÍ

1. Z webové stránky vmware.com stáhněte virtualizační software *VMware Workstation Player*. Stažený program následně nainstalujte.
2. Ze stránek sdnhub.org stáhněte obraz linuxové distribuce s předinstalovanými nástroji pro vývoj a testování softwarově definovaných sítí *All-in-one SDN App Development Starter VM*. Je doporučeno použít 64-bitovou verzi.
3. Poklikáním otevřete stažený soubor *SDN tutorial VM 64bit.ova*. Zobrazí se nabídka na importování do programu *VMware Workstation Player*. Odsouhlaste objevující se dialogové okna a spusťte import, který bude trvat několik minut v závislosti na výkonu počítače.
4. Po dokončení importu se začne spouštět linuxová distribuce a přivítá nás úvodní obrazovka plochy.
5. Soubory k jednotlivým laboratorním úlohám najdete ve složce Laboratorní úlohy na přiloženém USB klíči.