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

SOFTWAROVĚ DEFINOVANÉ SÍTĚ

SOFTWARE DEFINED NETWORKS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Artem Denisov

AUTHOR

VEDOUCÍ PRÁCE — Ing. Adrián Tomašov

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Artem Denisov ***ID:*** 196472

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Softwarově definované sítě

POKYNY PRO VYPRACOVÁNÍ:

Práce se zaměřuje na moderní trendy v řízení síťové infrastruktury. Její hlavní část spočívá v popisu kontroléru softwarově definované sítě ONOS a využívaných technologií (OpenFlow, OpenVSwitch). V bakalářské práci student prostuduje využití technologie VXLAN v ONOS a vytvoří tři experimentální topologie, kde demonstruje využití technologií v reálném scénáři. Na základě simulovaných topologií vytvoří dvě laboratorní úlohy.

DOPORUČENÁ LITERATURA:

[1] BADOTRA, Sumit; PANDA, Surya Narayan. Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking. Cluster Computing, 2020, 23.2: 1281-1291.

[2] KAUR, Karamjeet; SINGH, Japinder; GHUMMAN, Navtej Singh. Mininet as software defined networking testing platform. In: International Conference on Communication, Computing & Systems (ICCCS). 2014. p. 139-42.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Adrián Tomašov

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Obsahem této práce je popis kontroléru ONOS pro softwarově definované sítě (SDN) a návrh laboratorních úloh pro demonstraci činností kontroléru. Součástí práce je také popis základních technologií, na kterých je založen provoz mechanismů sítí SDN a také využití technologie VxLAN v těchto sítích. Dále budou popsané principy testovány v podmínkách virtualizace síťové infrastruktury.

KLÍČOVÁ SLOVA

SDN, OpenFlow, Open vSwitch, ONOS, VxLAN, OpenDaylight, GÉANT, ICONA

ABSTRACT

The content of this work is a description of the ONOS controller for software defined networks (SDN) and the design of laboratory tasks to demonstrate the activities of the controller. Part of the work is also a description of the basic technologies on which the operation of SDN mechanisms and the use of VxLAN technology in these networks is based. Next, the described principles will be tested in terms of network infrastructure virtualization.

KEYWORDS

SDN, OpenFlow, Open vSwitch, ONOS, VxLAN, OpenDaylight, GÉANT, ICONA

DENISOV, Artem. *Softwarově definované sítě*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 61 s. Bakalářská práce. Vedoucí práce: Ing. Adrián Tomašov, Ing.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Artem Denisov
VUT ID autora:	196472
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Softwarově definované sítě

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné 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*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing.Adrianu Tomašovi a panu Ing.Tomáši Horváthovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci

Obsah

Seznam obrázků

Úvod

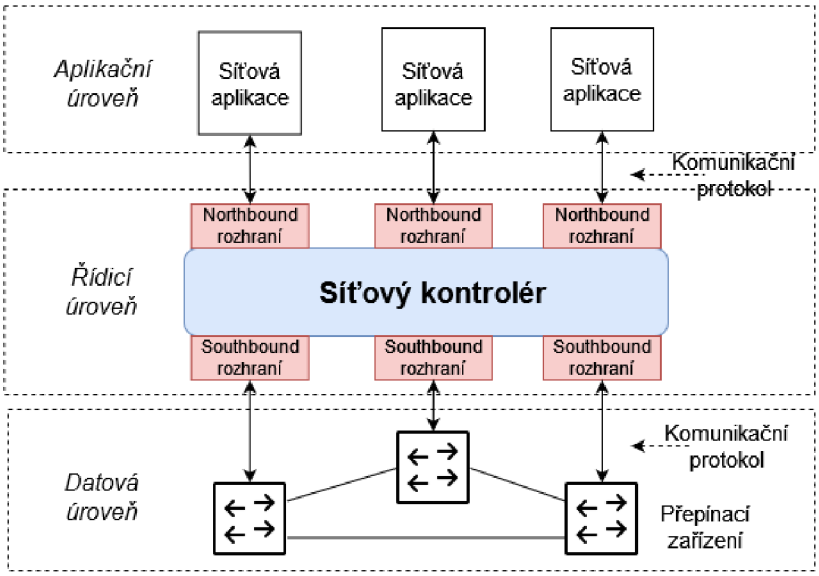
Tradiční počítačové a komunikační sítě jsou složité na nastavení a správu. Zahrnují mnoho typů zařízení, která fungují pomocí softwarů, mající uzavřené zdrojové kódy a vyžadující roky standardizace a testování. Správci systému musí konfigurovat jednotlivá síťová zařízení pomocí konfiguračních rozhraní, která se liší od dodavatele k dodavateli a často dokonce i mezi různými produkty od stejného dodavatele. Tento přístup zpomaluje inovace, zvyšuje složitost technické podpory a zvyšuje kapitálové náklady na provoz sítě.

Tato práce je věnována oblasti technologií síťové komunikace a jejím účelem je seznámit čtenáře s moderními metodami správy sítě založených na principech SDN. Největší pozornost bude věnována používání kontroléru ONOS v těchto sítích. Celá práce je rozdělena na teoretickou a praktickou část. Teoretická část obsahuje obecný úvod do sítí SDN a také informace o případech užití a principech fungování síťových protokolů a komponent, jako jsou OpenFlow a OpenvSwitch. Teoretická část také obsahuje podrobný popis kontroléru ONOS a také popis základních principů a příkladů použití VXLAN v moderních infrastrukturách. Praktická část obsahuje sádu laboratorní úkolů, během kterých čtenář v operačním systému Ubuntu vytvoří prostředí sestávající z virtuálních SDN sítí ovládaných kontrolérem ONOS. V rámci těchto úkolů bude čtenář přeposílat pakety mezi síťovými uzly a sledovat jejich pohyb pomocí analyzátoru sítě.

1 SDN sítě a jejich základní komponenty

Softwarově definovaná síť nebo Software defined network (SDN) je konceptem komunikační sítě která využívá softwarově založené kontroléry ke komunikaci se základními přepínacími zařízeními a přímému provozu v síti. V závislosti na použitých protokolech lze strukturu sítě rozdělit do několika funkčních úrovní, z nichž každá je schopna vykonávat určité funkce.

Roli směrovacích zařízení mohou plnit síťové přepínače založené na hardwaru nebo softwaru. Hardwarové přepínače poskytují lepší výkon sítě, zatímco softwarové přepínače poskytují lepší přizpůsobivost ke změnám v síti. Když paket dorazí do směrovacího zařízení, tak ono buď ví, co s paketem dělat, nebo jej předá kontroléru. Síťové aplikace, které pracují přímo s kontrolérem, rozhodují o tom, zda je třeba paket modifikovat, přesměrovat nebo zahodit. Přepínač si vybranou akci zapamatuje pomocí mezipaměti a příště o takovém paketu rozhodne sám. Tento postup se opakuje pro všechna směrovací zařízení sítě SDN, dokud paket neopustí řízenou síť.



Obr. 1.1: Zjednodušený pohled na architekturu SDN [1]

Interakce mezi kontrolérem SDN a zařízeními na datové úrovni probíhá prostřednictvím programovatelných „souththbound“ rozhraní Application programming interface (API), které používají speciální komunikační protokol. Kromě výše uvedených funkcí pro regulaci toku paketů jsou tyto protokoly schopny provádět funkce přenosu dat o stavu a statistikách sítě do síťového kontroléru.

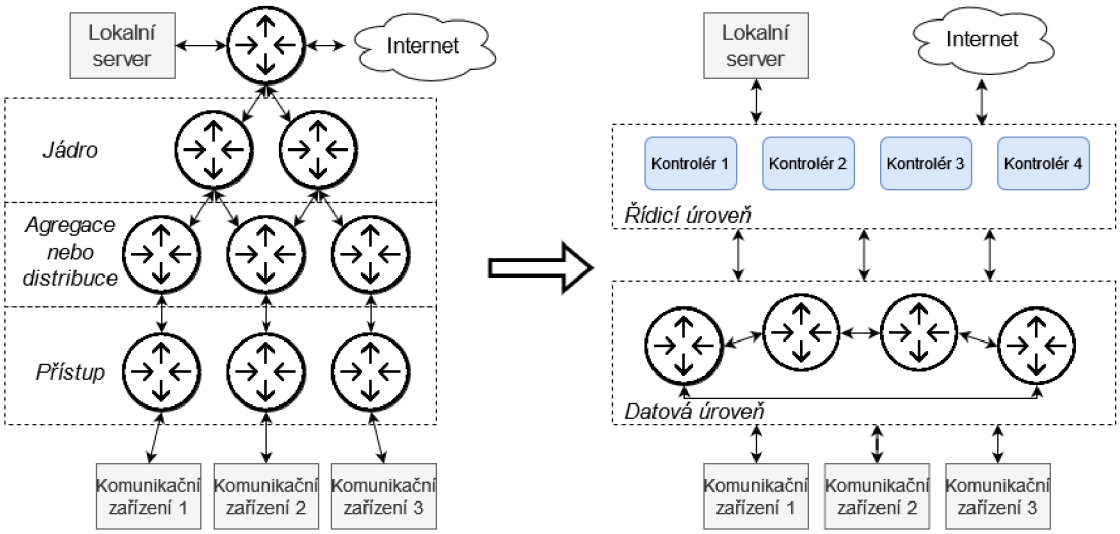
Flexibilní centralizovaná struktura sítí SDN umožňuje správcům pomocí síťových aplikací přizpůsobit síť všem nezbytným potřebám. V závislosti na pravidlech nastavených kontrolérem může jím ovládané zařízení fungovat jako přepínač, směrovač, firewall a mnoho dalších. Díky těmto vlastnostem jsou SDN sítě univerzálním řešením při implementaci správy takových síťových infrastruktur, jako jsou datová centra, kampusové a mobilní sítě.

1.1 Současný stav SDN sítí

Tradiční síťové technologie často nedrží krok s rostoucími na ně požadavky. Nárůst počtu různých zařízení a aplikací a také různé způsoby získávání přístupu k síti neustále mění konfiguraci vzdělávacích a firemních sítí bez ohledu na jejich původní topologii a velikost. Využití principů SDN v takových případech má za cíl zvýšit škálovatelnost sítě a snížit její zátěž.

Kampusové sítě

Příkladem infrastruktury, která čelí výše uvedeným výzvám, je tradiční kampusová síť. Kampusové sítě jsou navrženy tak, aby poskytovaly své služby širokému spektru osob s různou úrovní přístupu (pracovníci, zákazníci, návštěvníci, studenti atd...). Každý z uživatelů sítě potenciálně má celou sadu vlastních moderních zařízení která se mohou připojit k síti nebo využívá zařízení a služby poskytované organizací. Taková infrastruktura by měla být schopna podporovat mnoho mechanismů pro podporu a nasazení aplikací, vývojových a testovacích prostředí a měla by být také dobře adaptabilní na nové konfigurace.
Typické architektury kampusových sítí jsou striktně strukturovány do tří vrstev [2] – jádro, agregace nebo distribuce, přístup. Tradičně, organizace řeší problémy se škálovatelností a spolehlivostí pomocí přístupových bodů Wi-Fi a sítí Virtual Local Area Network (VLAN) pro izolaci přístupové vrstvy a virtuálního směrování pro izolaci provozu na síťové vrstvě. Logická síť je vytvořena přidružením fyzického portu přepínače nebo VLAN ke konkrétnímu a jedinému logickému síťovému ID s vlastní instancí směrovacího protokolu a směrovací tabulkou. Paket přesměrovaný do sítě bude přidružen k logické síti na základě portu, na který dorazil, nebo VLAN ID. Problém spočívá v tom, že tento port může patřit pouze do jedné logické sítě, a proto nemůže podporovat více toků které jsou zpracovávány v jiných sítích.



Obr. 1.2: Změna tradiční struktury kampusových sítí na softwarově definovanou

Použitím principů SDN můžeme oddělit řídicí vrstvu několika různými síťovými kontroléry, z nichž každý bude plnit specifickou roli. Například kontrolér č. 1 může být zodpovědný za provoz studentské sítě a kontrolér č. 2 za síťový provoz zaměstnanců instituce. Kontrolér může určit logickou síť pro každý tok paketů a poté tunelovat provoz na konec logické sítě. Síť se stává méně závislou na fyzických omezeních hardwarových síťových přepínačů a zjednodušuje definici logických sítí. Tím se eliminuje potřeba vytvářet instanci směrovacího protokolu v každém přepínači pro každou logickou síť.

Internet věcí

Ideálním prostředím pro nasazování softwarově definovaných sítí je technologie Internet of things (IoT), což je stabilním trendem v informačních technologiích právě díky vývoji a osvojování principů SDN. Technologie IoT by podle definice měla být snadno přizpůsobitelná změnám topologie sítě a požadavkům uživatelů v různých aspektech dynamicky se měnícího světa. Většina požadavků popsaných v předchozí podkapitole je aplikovatelná na IoT, ale v širším měřítku a s přidáním možnosti analyzovat získaná data.
Typickou IoT síť lze rozdělit do několika funkčních prvků, z nichž v každém přinese aplikace principů SDN určité výhody. Takže při interakci s aktivním vzdáleným přístupovým uzlem používaným pro bezdrátovou komunikaci na krátké vzdálenosti je SDN schopna poskytnout dynamickou alokace šířky pásma, diferenciace služeb a monitorování celého síťového provozu. Při komunikaci na velké vzdálenosti pomocí optických nebo mobilních sítí představují flexibilní kontroléry SDN způsob, jak zlepšit analýzu síťového toku při hodnoceni kvality služeb sítě Quality of service

(QoS).[3]
Jádro sítě IoT si lze představit jako bezdrátovou senzorovou síť, jejíž funkcionalitu lze rozšiřovat pomocí algoritmů SDN (například pomocí WISE [4] tabulky na senzorových uzlech). Všechny informace z koncových bodů komunikace jsou shromažďovány a přesměrovány do datového centra k analýze. Nasazení SDN kontroléru na takový objekt nám umožní mít možnost monitorovat každý výpočetní hardware nebo virtuální stroj, možnost rozšíření systému IoT a také globální pohled na veškerý provoz v síti.

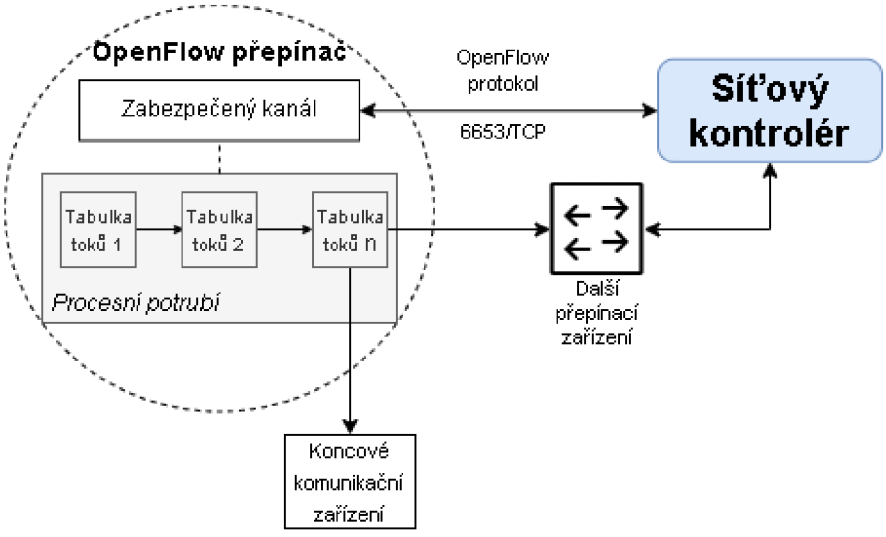
Je tedy snadné navrhnout teoretický model sítě SDN zobrazený na obrázku 1.1, který bude rozšířen o síťovou infrastrukturu s přístupovými body a senzorovými uzly [5]. Senzory budou zodpovědné za záznam dat z okolního prostředí za účelem jejich použití v různých aplikacích. Agenty, které jsou integrovány do senzorů, budou moci fungovat jako rozhraní pro komunikaci se zařízeními a kontroléry vyšší vrstvy prostřednictvím sítě přístupových bodů a sítě přepínacích a směrovacích zařízení na datové vrstvě. Řídicí úroveň, stejně jako model SDN na obrázku 1.2, bude obsahovat několik kontrolérů, které se vzájemně komunikují a provádějí určité funkce.

1.2 OpenFlow

OpenFlow je komunikační protokol, který umožňuje síťovým kontrolérům určovat cestu síťových paketů přes síť přepínačů.
První verze protokolu OpenFlow byla vydána Open Networking Foundation v únoru 2011 a je prvním a pravděpodobně nejznámějším „souththbound“ rozhraním, které je dodnes standardem pro implementaci sítí SDN. OpenFlow odděluje programovací vrstvu od zařízení pro přenos dat a implementuje je vzdáleně, pomocí softwarů a virtualizačních technologií. Díky oddělení řídicí a datové roviny (obrázek 1.1), logika sítě se zůstává na starosti síťového kontroléru a síťové přepínače se stávají jednoduchými zařízeními pro předávání paketů. Účelem vytvoření OpenFlow bylo vyvinout alternativu ke stávajícím protokolům, které byly příliš složité a obtížně použitelné pro efektivní vzdálenou správu.

Síťové přepínače pracující na protokolu OpenFlow se skládají ze tří částí: [6]

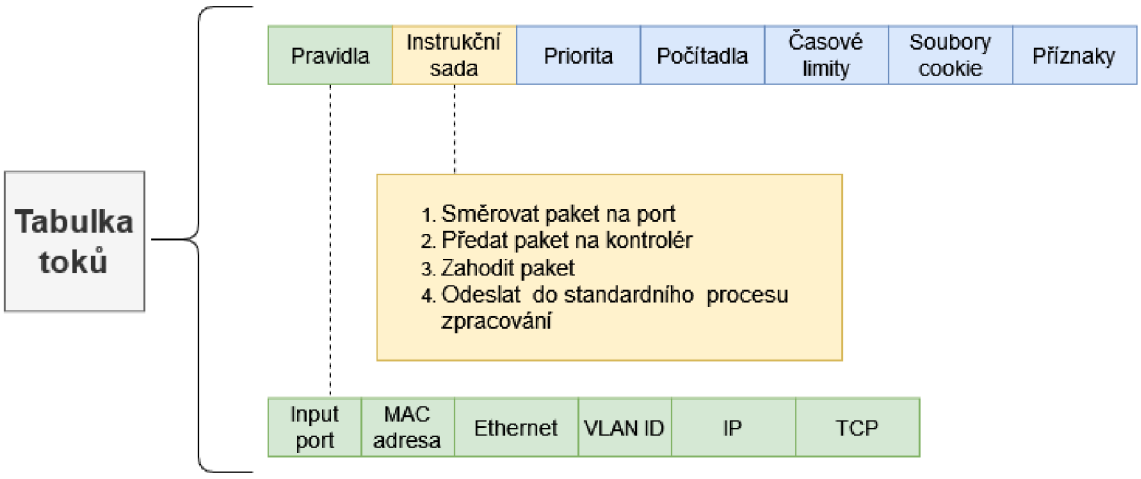
- *Tabulka toků (Flow Table)* – má akci spojenou s každým záznamem toku, která říká přepínači, jak zpracovat pakety
- *Zabezpečený kanál* – spojuje přepínač s vzdáleným ovládáním
- *OpenFlow protokol* – umožňuje odesílání příkazů a paketů mezi kontrolérem a přepínačem



Obr. 1.3: Příklad modelu přepínače založeného na OpenFlow [6]

Ve vztahu mezi OpenFlow kontrolérem a přepínačem role směrovací tabulky je vykonávána tabulkou toků. Hlavní rozdíl mezi nimi je v tom, že směrovací tabulka tradičního přepínače používá Internet protocol (IP) a Media Access Control (MAC) adresy k určení dalšího cíle paketu, zatímco správně nakonfigurovaná tabulka toků

k tomu může použít jakékoli informace obsažené v paketu. Typický přepínač OpenFlow může v závislosti na konfiguraci obsahovat několik [7] tabulek toků.
Při zpracování tabulkou toků je paket porovnán se záznamy v tabulce a v případě nalezení vhodného záznamu toku provede se příslušná instrukční sada. Tyto instrukce mohou explicitně směrovat paket do jiné tabulky toků anebo do jiného směrovacího zařízení. Pokud v tabulce nejsou žádné záznamy, odpovídající vlastnostem paketu, bude paket zpracován v souladu s pravidly stanovenými kontrolérem. Protokol OpenFlow je také schopen koexistovat s komerčními zařízeními [6]. Přepínač s podporou OpenFlow rozšiřuje možnosti tradičního přepínače přidáním vlastní sady pravidel pro zpracování paketů nad standardní pravidla deklarovaná v přepínači.



Obr. 1.4: Struktura tabulky toků v přepínačích s podporou OpenFlow

OpenFlow je navrstven na transportní protokol Transmission Control Protocol (TCP) a je implementován pomocí zpráv přenášených přes zabezpečený kanál. Je popsán specifickou strukturou, která začíná společnou osmibajtovou hlavičkou, skládající se z následujících částí [6][7]:

- *Verze protokolu (1B)* - určuje používanou verzi protokolu OpenFlow.
- *Typ zprávy (1B)* – hodnota, která kóduje jeden z 3 typů a 36 podtypů zpráv
  - *od-kontroléru-k-přepínači* – řízení nebo kontrola přepínače
  - *asynchronní* – síťový přepínač aktualizuje kontrolér o síťových událostech a změnách přepínače
  - *symetrické* – nastavení připojení, echo a zprávy specifické pro dodavatele
- *Délka zprávy (2B)* - označuje celkovou délku zprávy, tak aby k rozlišení jednoho rámce od druhého nebylo použito žádné další rámování
- *ID zprávy (4B)* - idetifikační číslo transakce spojené s tímto paketem (odpovědi používají stejné ID jako v požadavku)

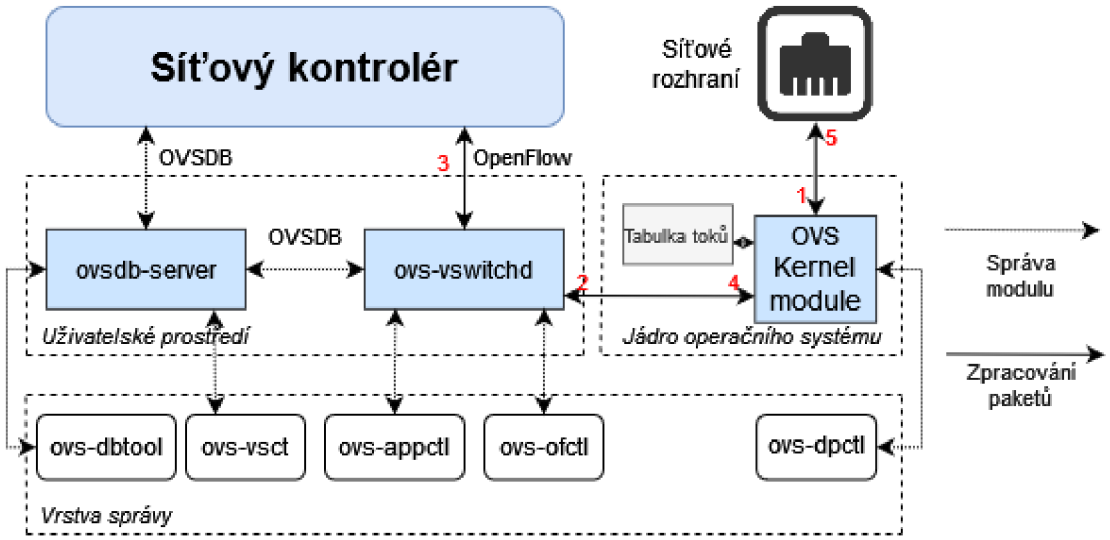
Komunikace mezi síťovými zařízeními a kontrolérem vždy začíná zprávou OFPT-HELLO následovanou OFPT-FEATURES-REQUEST pro přiřazení identifikačních vlastností přepínačům. Kontrolér periodicky posílá do zařízení zprávy typu *od-kontroléru-k-přepínači*. Při příjmu paketu, který neodpovídá záznamům v tabulce toků, odešle přepínač do kontroléru zprávu OFPT-PACKET-IN. Ve většině případů, kontrolér bude chtít přesměrovat paket do cílového směru a odpoví přepínači zprávou OFPT-PACKET-OUT. Kontrolér je také zodpovědný za úpravu tokové tabulky pomocí paketu typu OFPT-FLOW-MOD. Tento paket obsahuje všechny informace potřebné k vytvoření záznamu toku v tabulce.

Podobnosti a rozdíly mezi protokolem OpenFlow a dalšími protokoly pro správu sítě lze vidět na příkladu použití protokolu známého jako OpFlex [16]. OpFlex byl vyvinut společností Cisco a při řešení svých úkolů zaujímá opačný přístup než OpenFlow. Namísto umístění síťových zásad a správy sítě na samostatný softwarový kontrolér, OpFlex posílá zásady do síťových zařízení, která se mohou konfigurovat téměř automaticky na základě požadavků aplikace a překonfigurovat, když se požadavky změní. Zásady a pravidla síťového provozu jsou formulovány centrálně, ale rozhodnutí o tom, jak tyto zásady vynutit, dělají samotné síťové komponenty. Takové řešení problému však bude závislé na výrobci hardwaru a nebude respektovat princip oddělení řídicí a datové roviny, což by komplikovalo síťovou infrastrukturu.

1.3 Open vSwitch

Open vSwitch (OVS) je softwarová implementace virtuálního síťového přepínače s otevřeným zdrojovým kódem. Jako alternativa k tradičním hardwarovým přepínačům, je hlavním účelem Open vSwitch směrovat pakety mezi rozhraními a poskytovat přepínací prostředí pro virtuální stroje. Open vSwitch je jednou z nejpopulárnějších implementací Open vSwitch přepínače. Open vSwitch se skládá ze tří softwarových modulů [8], z nichž každý je schopen komunikovat se síťovým kontrolérem pomocí jednoho nebo více CLI příkazů:

- *ovs-vswitchd* – proces, který implementuje přepínač pro přepínání paketů na základě toků
  - *ovs-appctl* – konfiguruje a spravuje procesy na pozadí
  - *ovs-ofctl* – nástroj pro monitorování a správu Open vSwitch a jakéhokoli přepínače založeného na OpenFlow
- *ovsdb-server* – komponenta, která udržuje konfigurační databázi o síťových mostech a rozhraních
  - *ovsdb-tool* – spravuje soubory databáze Open vSwitch (OVSDB)
  - *ovs-vsctl* – dotazuje na změny v databázi a v případě potřeby je aplikuje
- *OVS Kernel module* – modul jádra operačního systému, který zajišťuje přepínání a tunelování paketů
  - *ovs-dpctl* – může vytvářet, upravovat a odstraňovat virtuální mosty Open vSwitch



Obr. 1.5: Interakce základních komponent Open vSwitch

Obrázek výše ukazuje příklad modelu Open vSwitch. Čísla popisují pořadí zpracování příchozího paketu, který neodpovídá položkám v tabulce toků modulu jádra

OS. Ze schématického hlediska lze všechny výše uvedené prvky rozdělit na uživatelské prostředí, prostředí jádra operačního systému a vrstvu správy.

Modul jádra implementuje několik datových cest [9], což je pouze soubor virtuálních portů. Každá datová cesta má přidruženou tabulku toků, kterou uživatelský prostor naplňuje toky, které mapují na sady akcí na základě hlaviček paketů a metadat. OVS modul v jádře operačního systému přijímá pakety jako první, z fyzického síťového rozhraní nebo virtuálního. Nyní buď ovs-vswitchd instruoval datovou cestu, jak zpracovávat pakety tohoto typu, nebo ne. V prvním případě se modul datové cesty jednoduše řídí pokyny zadanými ovs-vswitchd, které určují porty nebo tunely, na kterých se paket přenáší. Pokud se paket neshoduje s tokovou tabulkou, tak se zařadí do modulu ovs-vswitchd ke zpracování. Ovs-vswitchd přijímá tabulky toků z SDN kontroléru, porovnává všechny pakety přijaté z OVS modulu s těmito tabulkami, shromažďuje aplikované akce a nakonec uloží výsledek do mezipaměti v modulu jádra. To umožňuje OVS modulu, aby si nebyl vědom podrobností protokolu, což dále zjednodušuje celý systém. Z pohledu kontroléru, navštíví každý paket sérii z tokových tabulek a přepínač najde tok s nejvyšší prioritou, jehož podmínky paket splňuje. Poté dojde k akci odpovídající pravidlu toku. Ukládání do mezipaměti a rozdělení na uživatelské a jaderné komponenty jsou pro kontrolér neviditelnými implementačními detaily.

Algoritmus zpracování paketů Open vSwitch se mírně liší od standardního algoritmu Linux Bridge, který je také schopen předávat pakety mezi síťovými rozhraními. Linux Bridge přesměrovává všechny pakety sám a spoléhá na informace o IP a MAC adresách. Během zpracování paket nikdy neopustí modul jádra systému. Na rozdíl od Linux Bridge není Open vSwitch standardní funkcí operačních systémů založených na jádře Linux. Open vSwitch vyžaduje odlišnou instalaci na operační systém ale zároveň nabízí svým uživatelům širší škálu podporovaných protokolů a flexibilní systém pro monitorování a správu sítí v souladu se zásadami SDN.

1.4 Open Network Operating System (ONOS)

Kontrolér ONOS poskytuje řídicí rovinu pro softwarově definovanou síť, spravuje síťové komponenty, jako jsou přepínače a propojení, a spouští softwarové programy nebo moduly pro poskytování komunikačních služeb koncovým hostitelům a sousedním sítím. Cílem projektu je vytvoření SDN operačního systému pro poskytovatele komunikačních služeb, který je navržen pro škálovatelnost, vysoký výkon, dostupnost a odolnost proti chybám.

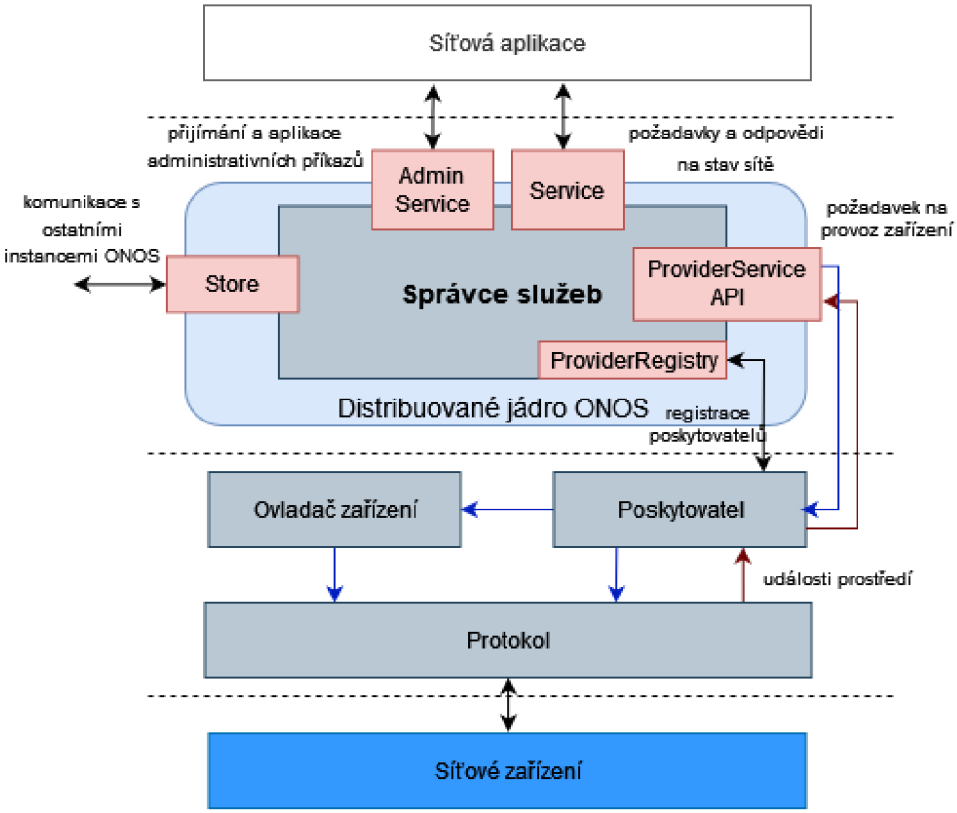
ONOS je od základu postaven jako distribuovaný operační systém. V důsledku toho může být ONOS nasazen jako kolekce serverů, které se vzájemně koordinují, aby poskytovaly možnosti, které jsou větší než součet jeho částí. Operátor sítě může přidávat servery postupně, bez přerušení, podle potřeby pro další kapacitu řídicí vrstvy. Instance ONOS spolupracují na vytvoření toho, co se zbytku sítě a aplikacím jeví jako jediná platforma. Aplikace a síťová zařízení nemusí vědět, zda pracují s jednou instancí nebo s více instancemi síťového kontroléru.

1.4.1 Architektura a subsystémy

Pro správu sítě, ONOS definuje služby – kolekci softwarových komponent, které jsou zodpovědné za konkrétní aspekt síťové infrastruktury. Vzhledem k modulární povaze tohoto kontroléru a přítomnosti různých možností rozšíření funkčnosti, dnes ONOS podporuje působivě dlouhý seznam subsystémů a služeb. Nicméně můžeme vyčlenit několik primárních služeb [10], které tvoří základ projektu, a popsat orientační systém interakce mezi jejich prvky.

- *Subsystém zařízení* – odpovědný za zjišťování a sledování zařízení, která tvoří síť, a za umožnění operátorům a aplikacím je ovládat.
- *Linkový subsystém* - spravuje propojení infrastruktury
- *Subsystem hostitelů* - spravuje hostitele koncových stanic a jejich umístění v síti
- *Subsystém topologie* - spravuje časově uspořádané snímky síťové topologie
- *PathService* - služba pro získání předem vypočítaných cest nebo pro vyžádání výpočtu cest pomocí aktuálního snímku topologie
- *Subsystém FlowRules* - zodpovědný za správu pravidel toků v systému a jejich instalaci do zařízení v síti
- *Subsystém paketů* - umožňuje aplikacím naslouchat datovým paketům přijatým ze síťových zařízení a vysílat datové pakety do sítě

Architektura ONOS je striktně rozdělena na protokolově agnostickou vrstvu jádra systému a vrstvu poskytovatelů, které jsou na používaném protokolu závisle. Jádro ONOS je zodpovědné za sledování informací o síťovém prostředí, jejich distribuci do aplikací a synchronizaci stavu mezi distribuovanými síťovými prvky. Prostřednictvím správce služeb, poskytujicích několik vícesměrových rozhraní, jádro spolupracuje se základní sítí za pomoci poskytovatelů. Hlavní funkcí poskytovatelů je provádět požadavky pocházející z jádra, zpracovávat a informovat jádro o událostech pocházejících z infrastrukturních zařízení. Při realizaci požadavek zasílaných jádrem systému a zaměřených na vrstvu přenosu dat, poskytovatel spolupracuje s modulem Protokol, který obsahuje implementaci různých komunikačních protokolů jako OpenFlow, NETCONF, SNMP, OVSDB, atd... . Správci služeb jsou také schopni komunikovat se službami ostatních uzlů v infrastruktuře pomocí modulu Store umístěného v jádru systému. Tímto způsobem je každý uzel v infrastruktuře informován o stavu dílčí části celé sítě.



Obr. 1.6: Architektura kontroléru ONOS a interakce modulů jeho služeb [11]

1.4.2 Některé alternativy

ONOS není jediným zástupcem SDN kontrolérů a má řadu alternativních projektů, mezi kterými vyčnívají OpenDayLight (ODL) a Ryu. Oba tyto projekty mají otevřené zdrojové kódy a nabízejí odlišný přístup ke správě SDN sítí.

ODL, stejně jako ONOS, má vestavěné mechanismy pro připojení kódových modulů a centralizovanou architekturu, která zjednodušuje podporu systému a poskytuje menší latenci mezi úzce souvisejícím jižním a severním API. Na rozdíl od ONOS a jeho komplexní řady subsystémů, ODL používá model-view-control framework a funguje na silné centrální úrovni abstrakce. Rozdíly mezi dvěma nejpoužívanějšími kontroléry lze pozorovat i v jejich výkonu. V experimentech popsaných ve výzkumné práci [12] se dokazuje, že ODL produkuje méně intenzivní tok paketů mezi kontrolérem a přepínačem, což umožňuje pohodlnější monitorování sítě. ONOS je zároveň stabilnější a rychleji reaguje na změny v topologie sítě.

Ryu je softwarově definovaný síťový framework založený na komponentách. Poskytuje softwarové komponenty s dobře definovaným API, které vývojářům usnadňují vytváření nových aplikací pro správu a řízení sítě. Základní konfigurace Ryu, na rozdíl od výše zmíněných kontrolérů, nepodporuje žádné nástroje pro monitorování sítě. Od svých uživatelů vyžaduje, aby si vytvářeli vlastní řešení úloh nastavených pro síť SDN, nebo používali produkty třetích stran. I když to vyžaduje odborné znalosti v oblasti softwarového vývoje, Ryu poskytuje úplnou flexibilitu a průhlednost řešení SDN sítí.

1.4.3 Aplikace v reálném světě

Díky své modularitě a širokému potenciálu pro rozšíření své funkčnosti je ONOS ideálním kandidátem pro nasazení softwarově definované rozlehlé sítě nebo Wide Area Network (WAN). Síť GÉANT může sloužit jako vzor pro studium této problematiky.

SDX body v GÉANT sítě

GÉANT [13] je panevropský síťový poskytovatel sjednocující přibližně 38 National research and education network (NREN) sítí pro 50 milionů uživatelů. V roce 2017 vývojový tým projektu představil dlouhodobý plán integrace principů a prvků SDN s řadou síťových služeb GÉANT. V době zveřejnění plánu byla většina služeb a připojení této sítě založena na tradiční architektuře IP/MPLS a běžela na složitém a drahém proprietárním zařízení, což bylo překážkou rozvoje projektu a komplikovalo poskytování a správu systému. Jedním z nejvýznamnějších milníků této iniciativy

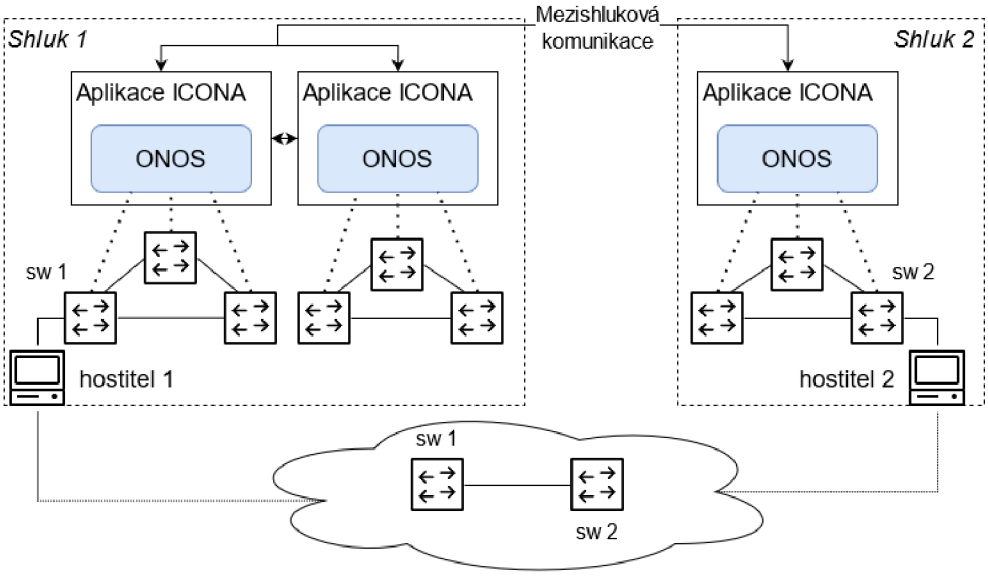
je vytvoření Software defined exchange points (SDX) – softwarově definovaných Internet exchange points (IXP). Tato technologie má za účel nahradit službu GEANT Open a skládá se ze dvou aplikací vyvinutých na bázi kontroléru ONOS – L3-SDX a L2-SDX.
L2-SDX byl realizován jako nová aplikace ONOS, která je schopna vytvářet vícedoménové ethernetové okruhy mezi fyzickými nebo virtuálními síťovými rozhraními poskytováním služeb spojové vrstvy. Poskytuje také nezbytné zajišťovací a monitorovací služby. Všechny požadavky na ovládání vytvořené uživateli jsou automaticky převedeny ONOS do toků na zařízeních SDN.
L3-SDX byl implementován nad existující ONOS aplikaci, nazvanou SDN-IP, která umožňuje síti SDN připojit se k externím sítím na internetu pomocí standardního protokolu Border Gateway Protocol (BGP). L3-SDX rozšiřuje výchozí funkcionalitu SDN-IP tím, že umožňuje propojení mezi více partnery patřícími do stejného autonomního systému prostřednictvím různých připojovacích bodů.

Při jednom z testů [13] realizovaných vývojovým týmem bylo použito 7 přepínačů OpenFlow jako zástupců sítě GÉANT na území, které sdružuje největší města severní, jižní a východní Evropy a také sever Velké Británie. Řídicí úroveň v této topologii byla tvořena třemi instancemi ONOS, nesoucími implementaci modulů L2-SDX a L3-SDX. Výsledky tohoto testu potvrdily očekávání vývojářů o výkonu, škálovatelnosti a potenciální schopnosti těchto síťových aplikací podporovat a zpracovávat současnou infrastrukturu GÉANT.

Inter Cluster ONOS Network Application

Jednou z definujících vlastností kontroléru ONOS je jeho distribuovaná architektura, která umožňuje jednotlivým kontrolérům vyměňovat si informace o zařízeních, která ovládají. V případě sítí WAN [15] však mohou různá vlastnictví obsahovat předpisy omezující znalosti kontrolérů o konfiguraci a topologii řídicích jednotek sousedních shluků. Centralizovaná řídicí rovina omezuje rychlost odezvy kvůli nevyhnutelným latencím způsobeným fyzickými vzdálenostmi. K překonání těchto překážek byl vytvořen nástroj ICONA.
Inter Cluster ONOS Network Application (ICONA) [14] je vyvinuta nad jádrem ONOS a navržena tak, aby rozšiřovala možnosti ONOS v síťových scénářích, které zahrnují přísné požadavky z hlediska odezvy řídicí roviny. Jako člen aplikační vrstvy ICONA komunikuje s kontrolérem pomocí „northbound“ rozhrání a je schopna interagovat s jak mezi prvky vlastního shluku, tak mezi instancemi aplikace spravujícími jiné shluky. Všechny hlavní funkce ICONA jsou implementovány pomocí tří interních softwarových modulů -– správce topologie, správce služeb a správce zálohování.

Správce služeb umožňuje síťovým operátorům interagovat s řídicí vrstvou sítě shluků, upravovat služby, posílat požadavky aplikaci ICONA pro poskytování záznamů síťových událostí. Správce topologie je zodpovědný za poskytování informací o zařízení datové roviny a jejich vzájemných vztahů. Tento modul používá upravenou verzi protokolu Link Layer Discovery Protocol (LLDP) k simulaci fyzického spojení mezi clustery pro výměnu informací o spravovaných koncových zařízení. Pro každé spojení mezi shluky je na základě různých měření vytvořena alternativní komunikační linka, kterou v případě výpadku komunikace mezi shluky použije správce zálohování k přesměrování síťového provozu. Při odesílání požadavků mezi zařízeními různých shluků, poskytuje ICONA každému jednotlivému účastníkovi dialogu zjednodušenou topologii sítě svého partnera v podobě jednoho přepínače.



Obr. 1.7: Základní model interakce mezi instancemi ICONA a jednotlivými hostiteli. [14]

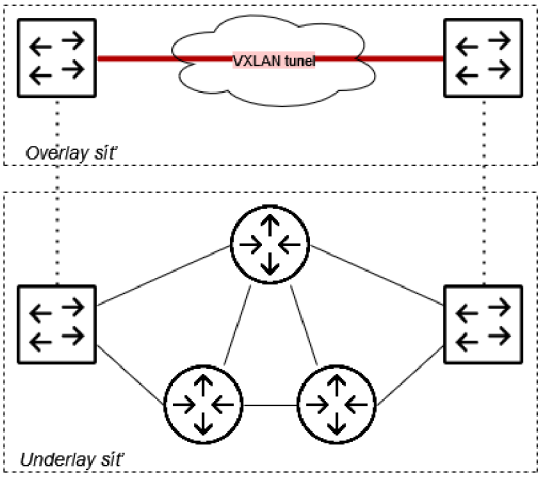
Experimenty provedené v článku [14] pomocí sítě GÉANT dokazují, že infrastruktura využívající kombinaci aplikací ONOS a ICONA je odolnější vůči neočekávaným výpadkům síťového připojení a poskytuje rychlejší zotavení ve srovnání se standardními mechanismy ONOS.

1.5 Virtual extensible Local Area network (VxLAN)

Hlavními problémy při implementaci síťových infrastruktur na datově vrstvě, je některé nedostatky Spanning Tree protokolu a nedostačující škálovatelnost ve VLAN sítích. Podle standartu IEEE 802.1Q, VLAN má délku identifikátoru 12 bitů, což znamená, že celkem můžeme vytvořit 4094 VLAN. Tak malý počet dostupných virtuálních sítí většinou neodpovídá požadavkům na poskytování služeb zákazníkům (např. pro 500 zákazníků bude pro každého z nich dostupných pouze 8 VLAN). Je také nutné zmínit skutečnost, že při implementaci fyzických infrastruktur, typický přepínač umístěný na nejvyšší úrovni racku si bude muset pamatovat MAC adresu každého virtuálního stroje běžícího na serveru, který je k přepínači připojen. Počet virtuálních strojů na server může dosáhnout několika stovek, což bude pro tento přepínač způsobit značnou zátěž ve srovnání s nevirtualizovaným prostředím. Protokol VxLAN, schopný tunelovat provoz Ethernet přes síť IP může sloužit řešením zmíněných problémů.

1.5.1 Overlay síť a zapouzdření paketů

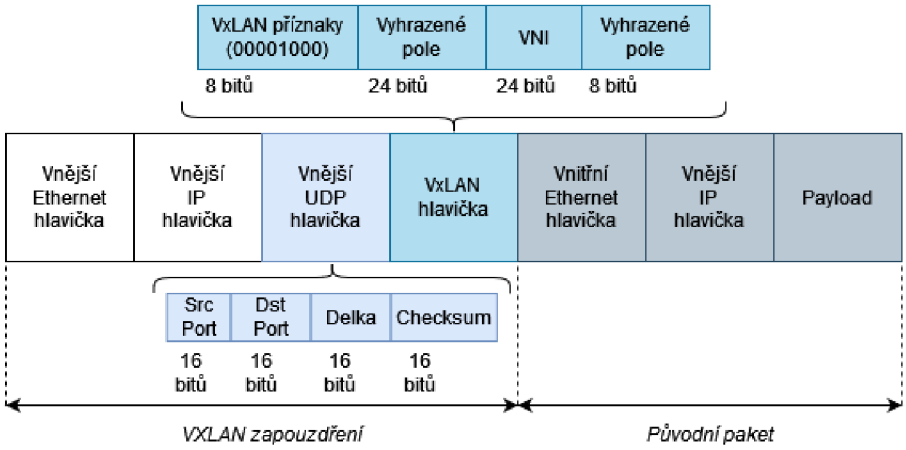
Hlavní myšlenkou VxLAN je implementovat sítě „navrch“ ostatních prostřednictvím tunelů přes zavedenou síťovou infrastrukturu. V tomto vztahu můžeme definovat překryvnou síť, která je reprezentována datovou vrstvou a podložní síť, která představuje vrstvu transportní.



Obr. 1.8: Zjednodušený model VxLAN sítě

Jedním ze způsobů nasazení překryvných architektur je zapouzdření paketů přidáním hlavičky UDP a hlavičky VxLAN před původním ethernetovým paketem. Hlavička VxLAN obsahuje 2 poli a několik vyhrazených bitů:

- *Příznaky (8b)* - jeden z bitů označuje validní stav VxLAN ID a ostatní jsou rezervované a nastavené na 0
- *VxLAN ID (24b)* - také nazývaný identifikátor virtuální sítě (VNI) a používá se k identifikaci konkrétní virtuální sítě na datové vrstvě



Obr. 1.9: Struktura rámce VxLAN

Důležitou součástí sítě VxLAN je VxLAN Tunnel Endpoint (VTEP). VTEP je koncovým bodem, který je zodpovědný za zapouzdření ethernetového paketu do VxLAN rámce, jeho předání do transportní sítě a také obrácení tohoto procesu pro příjem příchozího VxLAN rámce. Každý VTEP má dvě rozhraní: jedno je rozhraní přepínače na místním segmentu LAN pro podporu lokální komunikace s koncovým bodem a druhé je rozhraní k IP síti, která se používá pro přenos zapouzdřených rámců. Zařízení VTEP také zjišťuje vyhledávání dalších vzdálených koncových bodů a učí se vzdálené MAC prostřednictvím procesu analýzy dat, ve kterém VTEP posílá ARP rámce skupinám vícesměrového vysílání spojených s VxLAN.
Příkladem technologie, která využívá zapouzdření VxLAN, je Ethernet Virtual Private Network (EVPN). Zatímco tunely VxLAN se používají k roztažení 2. vrstvy mezi koncovými body sítě, rovina EVPN se používá k rozšíření překrytí mezi podnikovými lokalitami. EVPN je často popisován jako řídicí rovina pro VxLAN a zahrnuje několik funkcí které zajišťuji minimalizaci šíření ARP a snížení zahlcování unicastů.

1.5.2 Aplikace v SDN sítích: Migrace virtuálních strojů

Kromě zjevných výhod rozšíření počtu izolovaných systémů a snížení zátěže síťových zařízení, vlastnosti VxLAN v kombinaci s principy softwarově definovaných síti otevírají nové obzory pro správu síťových infrastruktur a také příležitosti ke zlepšení

jejich výkonu.

Jedná se o proces přemístění běžícího virtuálního stroje nebo aplikace mezi různými fyzickými stroji bez vypnutí klienta nebo aplikace. Tím se přenese operační paměť, uložená data a síťová připojení virtuálního stroje ze zdrojového fyzického nosiče do cílového. Pomocí technologie VxLAN je možné implementovat takovou operaci v rámci několika podsítích. Navzdory tomu, bez další podpory od třetí strany, během procesu migrace nebude virtuální stroj dostupný pro komunikaci s ostatními členy sítě. V případě použití VxLAN v softwarově definovaných sítích může přítomnost SDN kontroléru určitým způsobem přispět k procesu migrace [17]. Kontrolér bude sloužit jako prostředník mezi výpočetním centrem a zdrojovým virtuálním strojem, interagujícím prostřednictvím hypervizoru. Bude také zodpovědný za interakci s cílovým umístěním nového virtuálního stroje a vytvoření spojení mezi zdrojem a cílem migrace. Hlavní výhodou je skutečnost, že virtuální zařízení zůstává dostupné po celou dobu procesu migrace díky zrcadlovému obrazu stroje vytvořeným kontrolérem na cílovém médiu. Při pokusu o interakci s migrujícím virtuálním strojem budou všechny požadavky na připojení přesměrovány příslušným VTEP bodem do nového umístění virtuálního stroje.
Migrace virtuálních strojů není omezena na přesun mezi médii v rámci jednoho datového centra. V současné době existují technologie jako Mobile IP a L2VPN, které realizují pohyb virtuálních strojů mezi médii v centrech umístěných ve velké vzdálenosti od sebe. Avšak při použití většiny klasických migračních metod trpí WAN síť vážné potíže s udržováním konfigurace virtuálního zařízení po migraci. Dlouhou dobu konvergence sítě lze teoretické korigovat pomocí technologií SDN, nicméně použití jediného kontroléru, spravujícího celou WAN síť je zdaleka nejméně optimální řešení než použití více entit a rozložení zátěže řídicí roviny. Jistá studie [18] proto navrhuje řešení zmíněných problémů pomocí EVPN a VxLAN overlay technologií. EVPN poskytuje proces inzerování aktualizovaných síťových a strojových adres, který umožňuje geograficky distribuovaným datovým centrům, propojeným sítí VxLAN a migrujícím virtuálním strojím udržovat neustálou komunikaci. S EVPN, jakmile se přepínač lokálně naučí MAC adresu, okamžitě inzeruje tyto informace (IP/MAC Bindise Advertisement) všem svým MP-BGP kolegům se stejným VNI.

Migraci virtuálních zařízení lze také použít i v případech provádění procesu vyvažování zátěže v komunikačních SDN síti. Jedna se o techniku distribuci provozu po síti tak, aby bylo dosaženo požadovaného cíle, např. maximalizace propustnosti sítě nebo minimalizace latence. Load balancer jako modul pro síťový kontrolér může obsahovat několik algoritmů pro implementaci procesu vyvažování. Výpočetní centrum zase rozhodne, který z algoritmů nejlépe vyhovuje danému úkolu [19]. Podobná

architektura je popsaná v článku [20] a demonstruje vyvažování zátěže sítě založené na algoritmu Round Robin. Při dosažení určité zátěže na VTEP umístěné na úrovni přístupu k virtuálním zařízením, inteligentní centrum provede proaktivní migraci virtuálního stroje, aby přenesl zátěž na jiný VTEP.

2 Laboratorní úlohy

Tato kapitola je rozdělena na 2 části. Účelem první laboratorní úlohy je na příkladu zvážit princip fungování síťového kontroléru ONOS a také základní principy protokolu OpenFlow na jednoduché topologii virtuální sítě. Druhý laboratorní úkol bude věnován studiu principů fungování VxLAN na příkladu topologie sítě rozdělené do několika podsítí a manipulaci s tokovými tabulkami OVS přepínačů. Všechny níže uvedené virtuální stroje, síťová zařízení a další softwarové nástroje poběží na operačním systému Ubuntu 18.04.

2.1 Úloha č.1: Interakce ONOS se síťovými zařízeními

2.1.1 Příprava laboratorního prostředí

Mininet je simulátorem virtuální sítí a umožňuje nasadit sítě libovolného počtu přepínačů a hostitelů, které podporují standardní síťové nástroje dostupné na systémech Unix a Linux. Mininet také poskytuje rozšiřitelné Python API pro vytváření vlastních topologií sítě.

```
1  from mininet.topo import Topo
2  class triangleTopo( Topo ):
3      def __init__( self ):
4          # Initialize topology
5          Topo.__init__(self)
6
7          #info( '*** Adding hosts\n' )
8          h1 = self.addHost( 'h1' )
9          h2 = self.addHost( 'h2' )
10         h3 = self.addHost( 'h3' )
11
12         #info( '*** Adding switches\n' )
13         s1 = self.addSwitch('s1')
14         s2 = self.addSwitch('s2')
15         s3 = self.addSwitch('s3')
16
17         #info( '*** Creating links\n' )
18         self.addLink( s1, s2 )
19         self.addLink( s2, s3 )
20         self.addLink( s3, s1 )
21         self.addLink( h1, s1 )
22         self.addLink( h2, s2 )
23         self.addLink( h3, s3 )
24
25 topos = {'triangle': (lambda: triangleTopo())}
```

Výpis 2.1: Python skript pro vytváření trojúhelníkové topologie sítě v Mininet

Pro provedení instalace Mininet na Ubuntu, je potřeba zadat příkaz do terminalu:

```
$ sudo apt-get install mininet
```

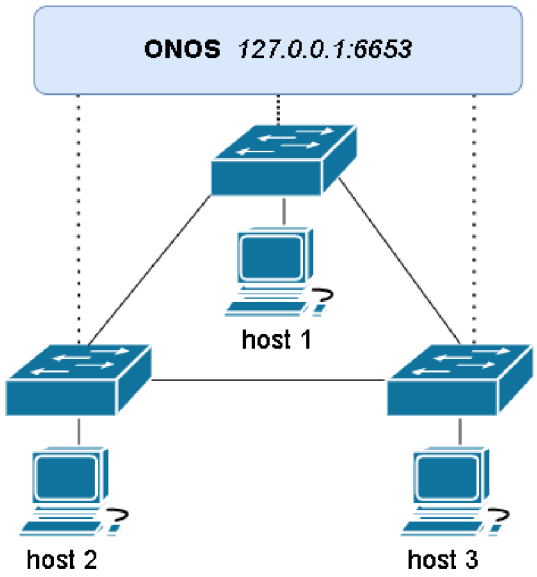
Jako síťový kontrolér se bude používat volně dostupná verze ONOS převzatá z hostingu GitHub (`https://github.com/opennetworkinglab/onos`). Verzi programu umístěnou ve vývojové větvi, označené jako „master“, je nutné zkopírovat pomocí nástroje *git* nebo stáhnout jako celý archiv, rozbalit a umístit do libovolného vhodného adresáře.

```
$ git clone https://gerrit.onosproject.org/onos
```

V kořenovém adresáře ONOS s pomocí Bazel, vícejazyčného nástroje pro kompilaci softwaru, spustitelný soubor lze vytvořit příkazem:

```
$ bazel build onos
```

Nakonec pro demonstraci pohybu paketů v síti je potřeba zapnout program Wireshark.



Obr. 2.1: Topologie laboratorní úlohy č.1

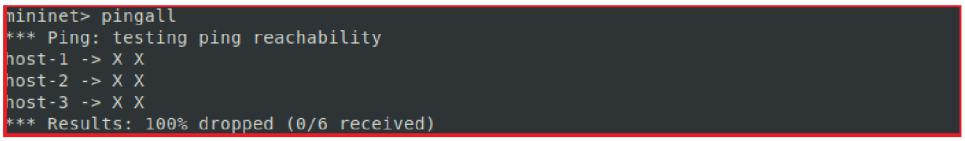
2.1.2 Vytvoření virtuální SDN sítě

Vytvoření virtuální SDN sítě v prostředí Mininet se provádí zadáním příkazu:

```
$ sudo mn --custom=triangleTopo.py --topo=triangle --mac \
--switch=ovs,protocols=OpenFlow13 --controller=remote
```

- *mn* – zkracený název programu Mininet
- *–custom* – umožňuje Mininetu číst informace z uživatelských skriptů
- *–topo* – zvolená topologie sítě
- *–mac* – zjednodušené MAC adresy
- *–switch* – zvolený přepínač
- *–protocols* – protokol používaný přepínačem pro komunikaci s kontrolérem
- *–controller* – zvolený kontrolér

Kvůli tomu, že ve výše uvedeném příkazu nespecifikován kontrolér a neuvádí se jeho vzdálená adresa, tak při pokusu odeslat jednoduchý požadavek Internet Control Message Protocol (ICMP) v příkazovém řádku Mininet oznámí se ztracené pakety.



Obr. 2.2: Komunikace virtuálních přepínačů v síti SDN bez připojeného kontroléru

Simulaci již možně zrušit příslušným příkazem

```
mininet> exit
```

Pro spuštění ONOS serveru na lokálním PC, z kořenového adresáře ONOS se zadává příkaz:

```
$ bazel run onos-local clean debug
```

ONOS poskytuje grafické uživatelské rozhraní, které obsahuje vizuální reprezentaci jedné instance nebo celého shluku kontrolérů. Toto rozhraní lze zobrazit ve webovém prohlížeči na adrese `http://127.0.0.1:8181/onos/ui`. Obsah této stránky je momentálně prázdný, protože nebyla přidána požadovaná síťová zařízení. Pro správu serveru, je nezbytné k němu připojit konzolu. V případě práci s lokálním serverem lze to provést pomocí spustitelného souboru v kořenovém adresáři ONOS:

```
$ ./onos/tools/test/bin/onos 127.0.0.1
```

Pro povolení předávání paketů OpenFlow na serveru ONOS, se do příkazového řádku konzoly zadávají následující příkazy:

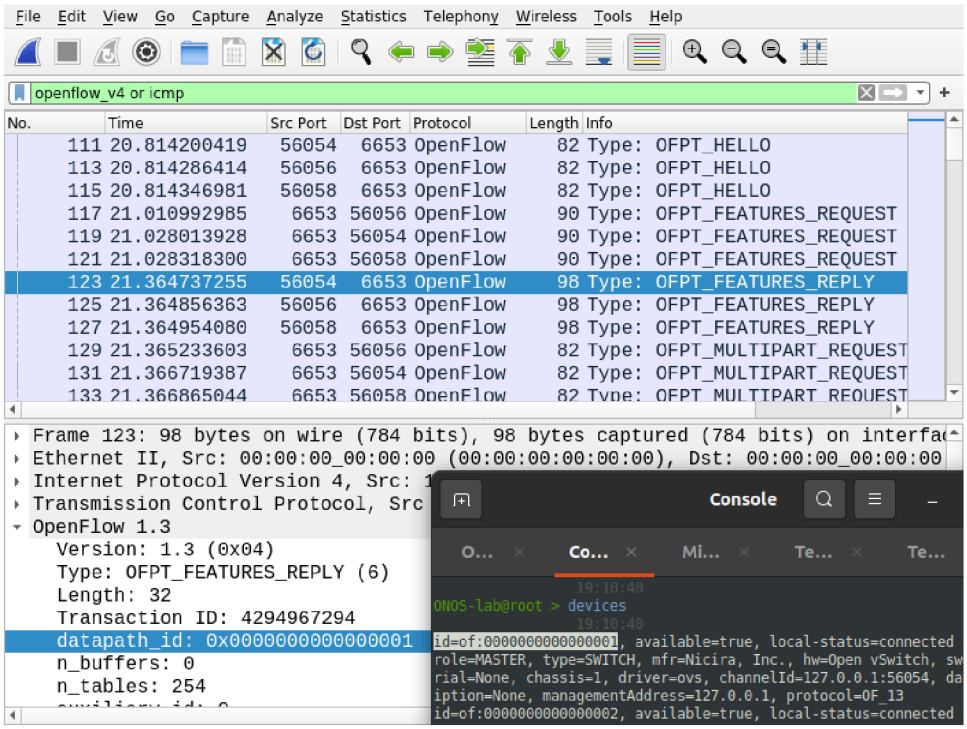
```
onos$ app activate org.onosproject.openflow
onos$ app activate org.onosproject.fwd
```

Nyní je potřeba zopakovat výše uvedený příkaz (2.1.2) pro vytvoření sítě SDN. Příkazem *pingall* je možně ověřit dostupnost jednotlivých hostitelů.

Obr. 2.3: Komunikace virtuálních přepínačů v síti SDN s připojeným kontrolérem

2.1.3 Demonstrace činnosti ONOS

Před provedením následujících kroků se doporučuje restartovat (příkaz v sekce 2.1.2) ONOS server. Jakmile bude ONOS připraven k provozu, bude v programu Wireshark povoleno sledování paketů na rozhraní označeném jako *loopback: lo*. Zároveň je také nutné zapnout zachycení paketů ve Wiresharku na datové úrovni (příkaz v sekce 2.1.2). Po připojení konzoly k serveru ONOS bude zadán příkaz *devices*. Tento příkaz umožní zvážit základní vlastnosti jednotlivých síťových zařízení.



Obr. 2.4: Data zachycená ve Wireshark při připojování přepínacích zařízení

Na obrázku výše je zvýrazněná zpráva odpovědí přepínače č.1 na požadavek zaslaný kontrolérem, aby zjistil identifikační vlastnosti přepínacího zařízení. Po přijetí tohoto paketu kontrolér zaznamená informace přijaté z přepínače a přiřadí odpovídající datové cestě unikátní identifikační číslo.

Síťové zařízení	Číslo portu	WAN IP	LAN IP	MAC adresa
Kontrolér	6653	127.0.0.1	-	-
Přepínač 1	56054	127.0.0.1	10.0.0.1/24	00:00:00:00:00:01
Přepínač 2	56056	127.0.0.1	10.0.0.2/24	00:00:00:00:00:02
Přepínač 3	56058	127.0.0.1	10.0.0.3/24	00:00:00:00:00:03

Tab. 2.1: Základní vlastnosti prvků vytvořené sítě z pohledu lokálního serveru

Do konzole ONOS bude zadán příkaz *flows*. Tento příkaz umožňuje pozorovat pravidla, která jsou obsažena v tabulkách toků síťových přepínačů. Každé ze síťových zařízení ONOS zpočátku naplní 4 pravidly, z nichž každé dává zařízení pokyn k přesměrování určitého typu paketů do kontroléru k dalšímu zpracování. V současném okamžiku mají zařízení pravidla pro následující typy paketů:

- *bddp* a *lldp* - jsou používány aplikací OpenFlow Link Discovery k vytváření a udržování informací o propojení pro datové cesty OpenFlow
- *arp* - slouží k určení MAC adresy
- *ipv4* - standardní internetový protokol

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	2	27,565	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	4	27,565	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	17,784	27,565	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	17,784	27,565	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

Obr. 2.5: Původní pravidla toků

Mezi klientskými hostiteli č.1 a č.2 budou v intervalu dvou sekund odeslány tři páry ICMP zpráv.

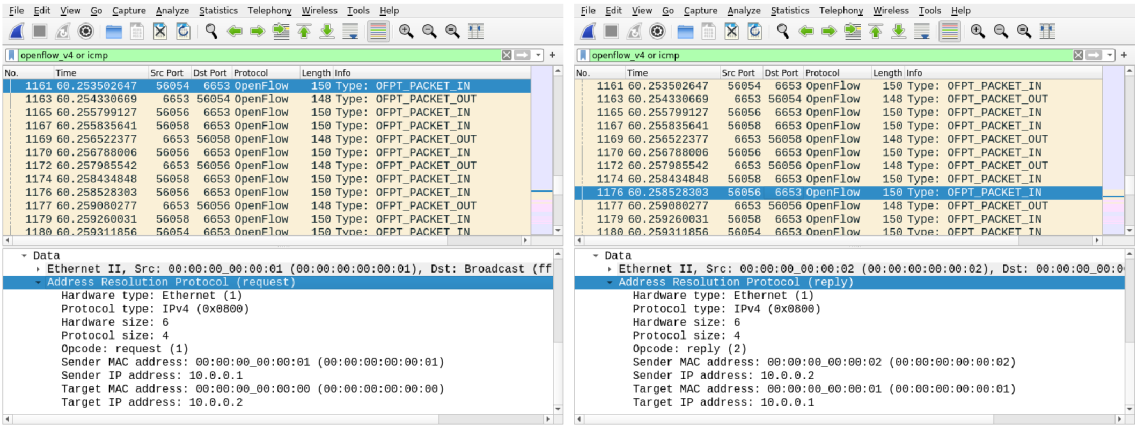
```
$ ping -i 2 -c 3 h1 h2
```

Po obdržení potvrzení, že všechny pakety dosáhly svých cílů, v konzoli ONOS budou zkontrolovány pravidla toku. Na příkladu přepínače č.1 lze vidět, že k výchozím pravidlům byla přidána pravidla pro příjem a odesílání paketů z přepínače č.2. Podobná situace je i na přepínači č.2. Tabulka toků přepínače č.3 zůstala nezměněna, protože se neúčastnila v transakci.

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	2	5	10	0	IN_PORT:3, ETH_DST:00:00:00:00:00:02, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:1], cleared:false	*fwd
Added	2	5	10	0	IN_PORT:1, ETH_DST:00:00:00:00:00:01, ETH_SRC:00:00:00:00:00:02	imm[OUTPUT:3], cleared:false	*fwd
Added	36	96,455	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	68	96,455	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	62,228	96,455	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	62,228	96,455	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

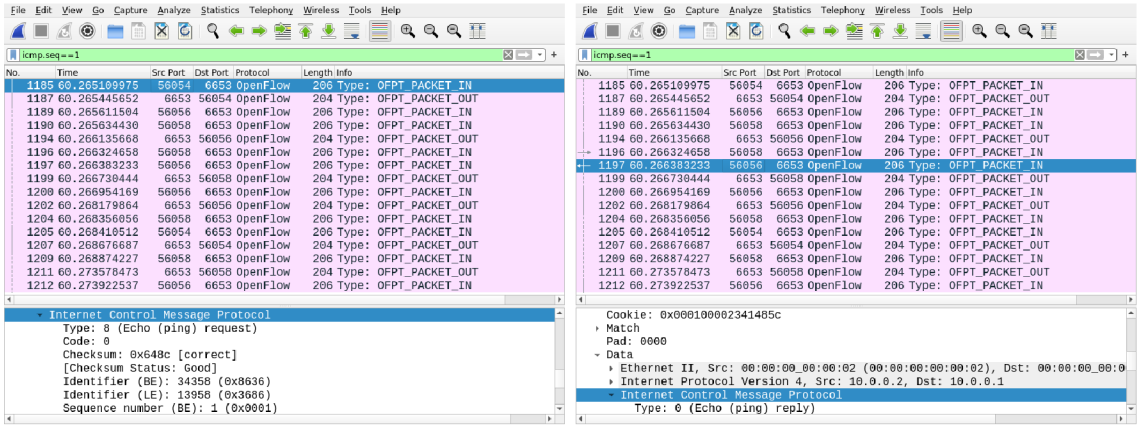
Obr. 2.6: Nová pravidla toků

Na začátku komunikace ve SDN sítích nemají jednotlivé hostitelé tušení o poloze svých sousedů, proto před odesláním paketu ICMP se mezi přepínači a kontrolérem vyměňují zprávy ARP, dokud se hostitelé nenajdou.



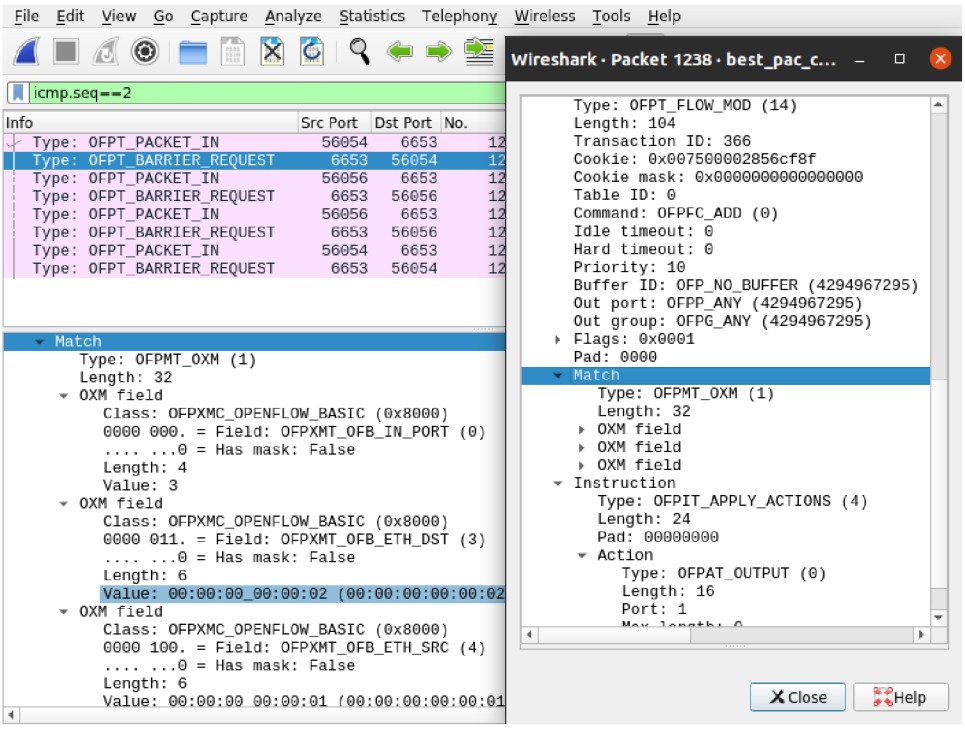
Obr. 2.7: Výměna ARP zpráv mezi hostitelem a kontrolérem

Při vstupu do sítě je první požadavek ICMP ve frontě předán kontroléru a teprve poté do cílového přepínače. Celá síť přitom aktivně reaguje na pohyb paketu a informuje o tom kontrolér. Síť reaguje podobným způsobem i na ICMP odpověď.



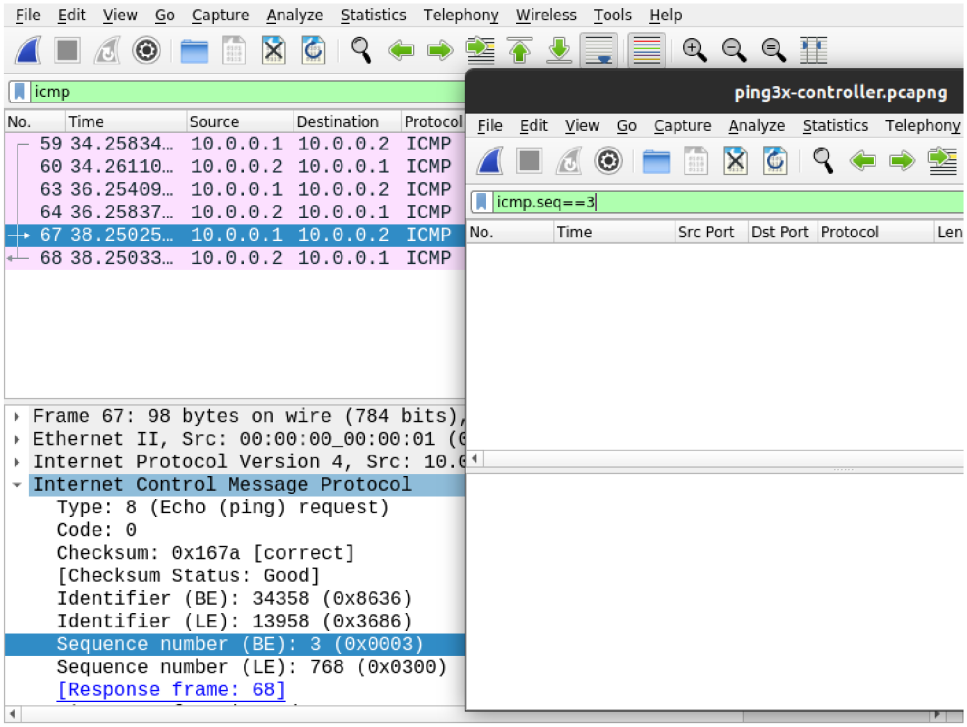
Obr. 2.8: První ICMP sekvence

Ve druhé ICMP sekvenci pakety jsou stále předávány kontroléru, ale tentokrát přepínače obdrží jako odpověď FLOW-MOD paket obsahující konfiguraci tabulky toku přepínačů.



Obr. 2.9: Druhá ICMP sekvence

Při pokusu o zobrazení třetí sekvence v prostředí kontroléru na adrese 127.0.0.1, nezobrazí se žádný provoz. Na obrázku níže lze vidět, že všechny tři páry zpráv žádost-odpověď úspěšně dosáhly svých cílů. To je způsobeno skutečností, že toky obsažené v tabulce jsou nyní naprogramovány do přepínače pomocí mezipaměti a mohou se pohybovat v datové vrstvě bez nutnosti kontaktovat síťový kontrolér.

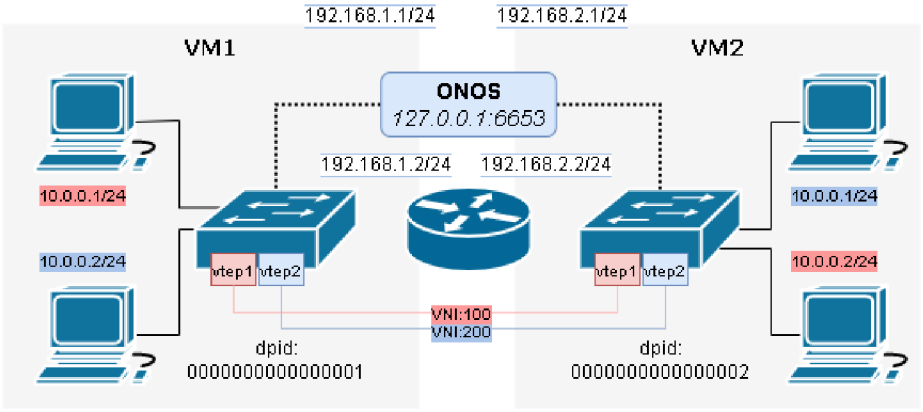


Obr. 2.10: Třetí ICMP sekvence

Je třeba mít na paměti, že obecně není zaručeno [7], že zprávy OpenFlow budou zpracovány v určitém pořadí, takže pokud zpráva OFPT-PACKET-OUT používající OFPP-TABLE závisí na toku, který byl nedávno odeslán do přepínače (se zprávou OFPT-FLOW-MOD), může být před zprávou OFPT-PACKET-OUT vyžadována zpráva OFPT-BARRIER-REQUEST, aby se zajistilo, že záznam toku byl potvrzen do tabulky toků před provedením OFPP-TABLE.

2.2 Úloha č.2: ONOS spravující VxLAN a editace tabulek toků

Pro tento laboratorní úkol budou použity tři virtuální stroje, které budou běžet na jednom hostiteli VirtualBox. Dva z těchto strojů budou obsahovat nakonfigurované prostředí Mininet se dvěma hostiteli, které budou připojeni k jedinému OVS přepínači. Třetí virtuální stroj nám poslouží jako router. Účelem tohoto úkolu bude



Obr. 2.11: První topologie laboratorní úlohy č.2

vytvoření dvou VxLAN spojení mezi hostiteli, nacházejících se v různých podsítích a neschopných mezi sebou komunikovat za normálních podmínkách. Tento úkol bude komplikovaný tím, že bude na datové úrovni nacházet hostitel, který má podobné adresy IP a MAC adresy jako hostitel cílový. V rámci tohoto úkolu bude zvážena manuální úprava tokových tabulek OVS přepínačů a správa přepínačů pomocí síťového kontroléru.

2.2.1 Vytvoření spojení mezi virtuálními stroji ve VirtualBox

Pro vytvořeni komunikační síti mezi jednotlivými virtuálními stroji v programu VirtualBox, jsou vyžadovány následující kroky:

1. Vybrat virtuální stroj
2. Na panelu nástrojů vybrat *Nastavení*, *Síť*
3. Vybrat „Povolit síťový adaptér“
4. V poli „Připojeno k“ vybrat „Interní síť“
5. Do pole „Jméno“ zadat libovolný vhodný název sítě (členové stejné interní sítě musí mít stejné názvy adaptérů)
6. V poli „Promiskuitní režim“ vybrat „povolit všem“

Podobná operace musí být provedena pro každý stroj zapojený do sítě. Po dokončení

výše uvedených kroků bude mít virtuální systém nové dostupné rozhraní, které lze staticky konfigurovat pomocí nástroje *ifconfig*:

```
$ sudo ifconfg JMENO_ROZHRANÍ IP/MASKA
$ sudo ifconfg JMENO_ROZHRANÍ up
```

Spojení mezi dvěma počítači lze zkontrolovat pomocí příkazu *ping*.

```
-> ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.821 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.458 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.448 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2018ms
rtt min/avg/max/mdev = 0.448/0.575/0.821/0.173 ms


ONOS-lab
-> ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.789 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.378 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=64 time=0.394 ms
^C
--- 192.168.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.378/0.520/0.789/0.190 ms
```

Obr. 2.12: Komunikace směrovače s jednotlivými virtuálními stroji

2.2.2 Ruční konfigurace jednotlivých přepínačů

Na virtuálních strojích obsahujících OVS přepínače je potřeba zkompilovat (sekce 2.1.1) a spustit lokální ONOS aplikaci (sekce 2.1.2). Zatím bude nasledovat vytvoření síťových zařízení a hostů. Pro správné připojení k ONOS musí každý z přepínačů obsahovat jedinečné identifikační číslo zařízení. Toto číslo lze zadat pomocí souboru obsahujícího konfiguraci vlastní topologie.

```
    #info( '*** Adding hosts\n' )
    h1 = self.addHost( 'h1', ip='10.0.0.1/24',
                       mac='00:00:00:00:00:01' )
    h2 = self.addHost( 'h2', ip='10.0.0.2/24',
                       mac='00:00:00:00:00:02' )

    #info( '*** Adding switches\n' )
    s1 = self.addSwitch( 's1', dpid="0000000000000001")
```

Výpis 2.2: Definice vlastností zařízení ve vlastní topologii

```
$ sudo mn --custom=topo1.py --topo=single2 --mac --switch=ovs,
protocols=OpenFlow13 --controler=remote
```

Ve výchozím nastavení ONOS přepíše všechna pravidla, která nebyla nainstalována síťovým kontrolérem. Pro demonstrační účely lze tuto funkci vypnout následujícím příkazem:

```
ONOS@root> cfg set\
org.onosproject.net.flow.impl.FlowRuleManager\
allowExtraneousRules true
```

Nyní by měli být vytvořené koncové body VxLAN. Na OVS přepínačích budou nakonfigurovaný virtuální porty, které budou ukazovat na vzdálenou IP adresu, patřící jinému virtuálnímu stroji. K dosažení tohoto cíle by měl být následující příkazy spuštěny na obou virtuálních počítačích v prostředí Mininet.

```
mininet> sh ovs-vsctl --protocols=OpenFlow13 add-port s1 vtep1
--set interface vtep1 type=vxlan option:remote_ip=192.168.2.1
option:key=100 ofport_request=10
```

```
mininet> sh ovs-vsctl --protocols=OpenFlow13 add-port s1 vtep2
--set interface vtep2 type=vxlan option:remote_ip=192.168.2.1
option:key=200 ofport_request=20
```

- *sh* – provést shell příkaz
- *ovs-vsctl* – nástroj pro dotazování a konfiguraci OVS přepínače
- *protocols* – protokol používaný přepínačem pro komunikaci s kontrolérem
- *add-port* – přidávaní nového síťového rozhraní přepínači
- *s1* – jmeno přepínače, na kterém bude port nasazen
- *vtep1* – jmeno portu
- *set interface* – nastavení konfiguraci síťového rozhraní/portu
- *type* – nastavení typu síťového rozhraní/portu
- *option:remote-ip* – nastavení vzdalené IP adresy
- *option:key* – definice identifikátoru virtuální sítě (VNI)
- *ofport-request* – definice čísla portu

Stejné příkazy by měli být spuštěny na protějším virtuálním stroji, ale s jinou vzdálenou IP adresou (*option:remote-ip=192.168.1.1*). Nyní musí být přepínače naplněny pravidly toku. Nástroje OVS umožňují přidávat k přepínačům jednotlivá pravidla nebo několik pravidel současně pomocí textových souborů.

```
table=0,in_port=1,dl_dst=00:00:00:00:00:02,actions=output:10,resubmit(,2)
table=0,in_port=1,arp,nw_dst=10.0.0.2,actions=output:10,resubmit(,2)
table=0,in_port=2,dl_dst=00:00:00:00:00:01,actions=output:20,resubmit(,2)
table=0,in_port=2,arp,nw_dst=10.0.0.1,actions=output:20,resubmit(,2)
table=0,actions=resubmit(,1)

table=1,dl_dst=00:00:00:00:00:01,actions=output:1
table=1,arp,nw_dst=10.0.0.1,actions=output:1
table=1,dl_dst=00:00:00:00:00:02,actions=output:2
table=1,arp,nw_dst=10.0.0.2,actions=output:2
table=1,actions=resubmit(,2)

table=2,priority=100,actions=drop
```

Výpis 2.3: Soubor obsahující pravidla toků pro OVS přepínač na VM1

Z výše uvedeného výpisu lze vidět, že tato sada pravidel obsahuje tři tokové tabulky, které postupně kontrolují každý příchozí paket. Tabulka číslo 0 je tedy zodpovědná za přepínání paketů z vnitřní sítě do příslušných VxLAN tunelů. Pokud příchozí paket nesplňuje podmínky popsané v prvních čtyřech řádcích souboru, je přesměrován do tabulky číslo 1, kde je považován za paket z vnější sítě. Po provedení příslušných operací bude paket přesměrován do tabulky číslo 2, kde bude zahozen.

```
table=0,in_port=1,dl_dst=00:00:00:00:00:02,actions=output:20,resubmit(,2)
table=0,in_port=1,arp,nw_dst=10.0.0.2,actions=output:20,resubmit(,2)
table=0,in_port=2,dl_dst=00:00:00:00:00:01,actions=output:10,resubmit(,2)
table=0,in_port=2,arp,nw_dst=10.0.0.1,actions=output:10,resubmit(,2)
table=0,actions=resubmit(,1)

table=1,dl_dst=00:00:00:00:00:01,actions=output:1
table=1,arp,nw_dst=10.0.0.1,actions=output:1
table=1,dl_dst=00:00:00:00:00:02,actions=output:2
table=1,arp,nw_dst=10.0.0.2,actions=output:2
table=1,actions=resubmit(,2)

table=2,priority=100,actions=drop
```

Výpis 2.4: Soubor obsahující pravidla toků pro OVS přepínač na VM2

K přidání těchto toků do konfigurace síťových zařízení se používá utilita *ovs-ofctl*.

```
mininet> sh ovs-ofctl --protocols=OpenFlow13 add-flows s1
flows.txt
```

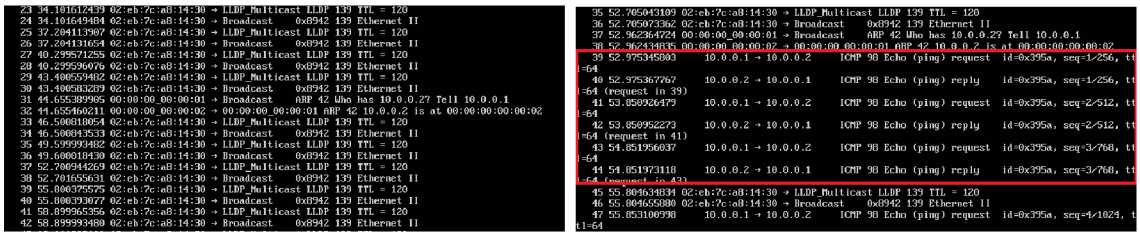
Kvůli skutečnosti že virtuální stroje, obsahující koncové body komunikaci nacházejí se v různých podsítích, je potřeba nastavit směrovací cesty a zároveň povolit směrování mezi síťovými rozhraní v operačním prostředí směrovače.

```
$ sudo ip route add 192.168.2.0/24 via 192.168.1.2
```

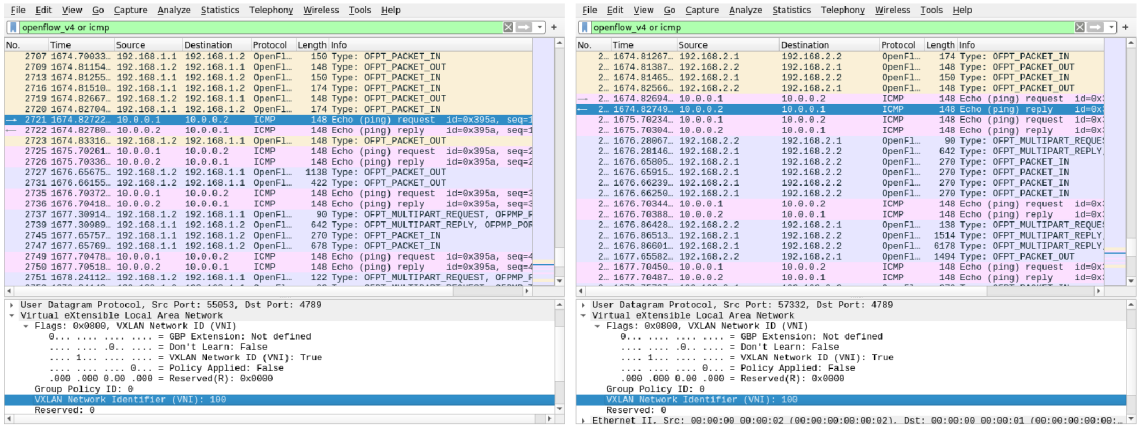
```
$ sudo ip route add 192.168.1.0/24 via 192.168.2.2
```

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

Ve vytvořeném virtuálním prostředí umístěném ve VM1 bude zadán příkaz *h1 ping 10.0.0.2*. Na obrázcích níže lze vidět pohyb paketů na interním a externím rozhraní virtuálních strojů a také si všimnout absenci provozu na rozhraní, které má podobnou IP adresu jako cílová adresa ve výše uvedeném příkazu.



Obr. 2.13: Provoz na 10.0.0.2/24 v síti 192.168.1.0/24 (vlevo) a 192.168.2.0/24 (vpravo)



Obr. 2.14: Provoz na 192.168.1.1/24 (vlevo) a 192.168.2.1/24 (vpravo)

Namísto hostitele umístěného v lokální podsíti bude paket přesměrován přes router do síťového prostředí VM2. Při pohybu ICMP paketů transportní vrstvou sítě bude k původním paketům přidána hlavička VxLAN — výsledek činnosti komunikačních koncových bodů v virtuální síti. Při překročení hranic VTEP směrem k cílovému hostiteli bude hlavička odstraněna a původní paket bude předložen příjemci bez dalších informací o připojení VxLAN.

2.2.3 Konfigurace přepínačů pomocí kontroléru

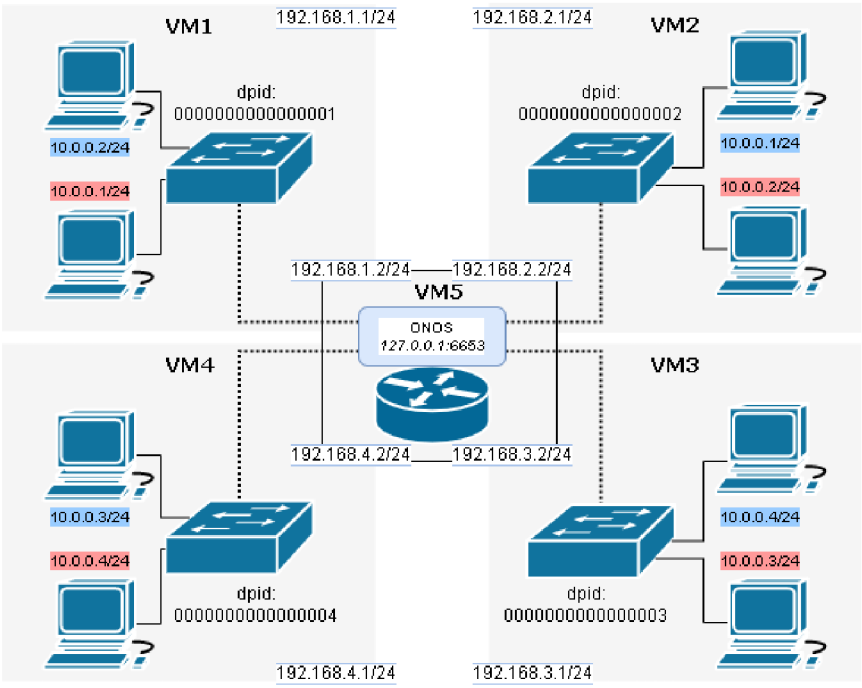
Navzdory tomu, že předchozí úkol byl úspěšně splněn, takový přístup porušuje princip softwarově definovaných sítí o oddělení úrovně řízení a úrovně přenosu dat v síťových infrastrukturách. Efektivnějším způsobem konfigurace přepínačů v sítích SDN je jejich vzdálena konfigurace prostřednictvím síťového kontroléru. Prostřednictvím REST API je ONOS schopen komunikovat s různými síťovými aplikacemi, včetně přijímání konfigurace pro zařízení, která má pod kontrolou. Tím pádem lze pravidla toků naprogramovat do přepínačů odesláním řídící instanci ONOS požadavku POST s přiloženým json souborem, obsahujícím popis těchto pravidel.

```
$ curl --user ONOS_USER:ONOS_PASWD -X POST --header\
'Content-Type: application/json' --header 'Accept:\
application/json' -d @PATH_TO_JSON\
http://ONOS_IP:8181/onos/v1/flows
```

Pro správné nahrávání pravidel toků na odpovídající zařízení musí být v json souboru definovány následující parametry:

1. *isPermanent* – trvání existence toku (pokud nastavena na hodnotu "false", je nutně definovat parametr "timeout")
2. *priority* – priorita toků
3. *deviceId* – unikátní číslo zařízení s předponou "of:“
4. *selector* – podmínky shodování toků (lze definovat uvnitř parametru "criteria"(ref.))
5. *treatment* – pokyny pro akce (mělo by být uvedeno v parametru "instructions"(ref.)

V tomto laboratorním úkolu bude použita předchozí topologie, která bude rozšířená o 2 další přepínače, obsahující hostitele 10.0.0.3/24 a 10.0.0.4/24. Ke každému z přepínačů bude připojeno (příkazy v sekci 2.2.2) 6 externích virtuálních VTEP portů, tak aby každý pár zařízení bude propojen pomocí dvou VxLAN tunelů (VNI=100 a VNI=200).



Obr. 2.15: Druhá topologie laboratorní úlohy č.2

Zdrojová IP adresa	VTEP port	VNI	Vzdálené rozhraní	Cílové zařízení (dPID)	Cílová IP adresa
10.0.0.1/24	10	100	192.168.2.1	of:0000000000000002	10.0.0.2/24
	20	100	192.168.3.1	of:0000000000000003	10.0.0.3/24
	30	100	192.168.4.1	of:0000000000000004	10.0.0.4/24
10.0.0.2/24	15	200	192.168.2.1	of:0000000000000002	10.0.0.1/24
	25	200	192.168.3.1	of:0000000000000003	10.0.0.4/24
	35	200	192.168.4.1	of:0000000000000004	10.0.0.3/24

Tab. 2.2: Konfigurace přepínače č.1 (of:0000000000000001)

Zdrojová IP adresa	VTEP port	VNI	Vzdálené rozhraní	Cílové zařízení (dPID)	Cílová IP adresa
10.0.0.2/24	10	100	192.168.1.1	of:0000000000000001	10.0.0.1/24
	20	100	192.168.4.1	of:0000000000000004	10.0.0.4/24
	30	100	192.168.3.1	of:0000000000000003	10.0.0.3/24
10.0.0.1/24	15	200	192.168.1.1	of:0000000000000001	10.0.0.2/24
	25	200	192.168.3.1	of:0000000000000004	10.0.0.3/24
	35	200	192.168.4.1	of:0000000000000003	10.0.0.4/24

Tab. 2.3: Konfigurace přepínače č.2 (of:0000000000000002)

Zdrojová IP adresa	VTEP port	VNI	Vzdálené rozhraní	Cílové zařízení (dPID)	Cílová IP adresa
10.0.0.3/24	10	100	192.168.4.1	of:0000000000000004	10.0.0.4/24
	20	100	192.168.1.1	of:0000000000000001	10.0.0.1/24
	30	100	192.168.2.1	of:0000000000000002	10.0.0.2/24
10.0.0.4/24	15	200	192.168.4.1	of:0000000000000004	10.0.0.3/24
	25	200	192.168.1.1	of:0000000000000001	10.0.0.2/24
	35	200	192.168.2.1	of:0000000000000002	10.0.0.1/24

Tab. 2.4: Konfigurace přepínače č.3 (of:0000000000000003)

Zdrojová IP adresa	VTEP port	VNI	Vzdálené rozhraní	Cílové zařízení (dPID)	Cílová IP adresa
10.0.0.4/24	10	100	192.168.3.1	of:0000000000000003	10.0.0.3/24
	20	100	192.168.2.1	of:0000000000000002	10.0.0.2/24
	30	100	192.168.1.1	of:0000000000000001	10.0.0.1/24
10.0.0.3/24	15	200	192.168.3.1	of:0000000000000003	10.0.0.4/24
	25	200	192.168.2.1	of:0000000000000002	10.0.0.1/24
	35	200	192.168.1.1	of:0000000000000001	10.0.0.2/24

Tab. 2.5: Konfigurace přepínače č.4 (of:0000000000000004)

Ke konfiguraci přepínačů v této topologii a vytvoření komunikačního mostu mezi konkrétními hostiteli byl použit vlastní shell skript nazvaný „push-flows“. Tento skript přijímá na vstup 3 argumenty — unikátní číslo zařízení OVS (lze zjistit v konzole ONOS pomocí příkazu *devices*), zdrojovou a cílovou IP adresu.
Při provádění příkazu *push-flows 0000000000000001 10.0.0.1 10.0.0.3* se tedy do přepínače č.1 zapíše trvalé pravidlo toku směřující pakety od 10.0.0.1 do 10.0.0.3. Opačné pravidlo pro vysílání odpovědí bude zaznamenáváno na přepínači č.3. Nově nainstalované pravidla lze zkontrolovat pomocí příkazu *flows* v konzole ONOS.

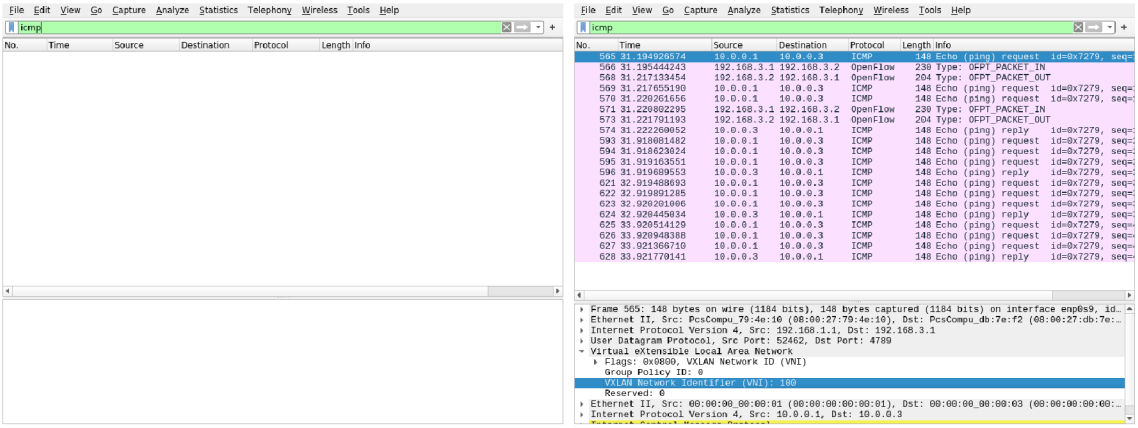
```
    id=b0000099608edf, state=ADDED, bytes=0, packets=0, duration=15, liveType=UNKNOWN, priority=10, tableId=0, a
ppId=org.onosproject.rest, selector=[IN_PORT:1, ETH_TYPE:ipv4, IPV4_DST:10.0.0.3/32], treatment=DefaultTrafficTr
eatment{immediate=[OUTPUT:20], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata
=null}
```

Obr. 2.16: Nově vytvořené pravidlo toku na OVS přepínači č.1

```
    id=b000007dc0cd9f, state=ADDED, bytes=0, packets=0, duration=16, liveType=UNKNOWN, priority=10, tableId=0, a
ppId=org.onosproject.rest, selector=[IN_PORT:1, ETH_TYPE:ipv4, IPV4_DST:10.0.0.1/32], treatment=DefaultTrafficTr
eatment{immediate=[OUTPUT:20], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata
=null}
```

Obr. 2.17: Nově vytvořené pravidlo toku na OVS přepínači č.3

Stejně jako v předchozí úloze, obsahují podsítě 192.168.3.0/24 a 192.168.4.0/24 klienty s identickými IP adresami. Ve virtuálním prostředí VM1 bude zadán příkaz *h1 ping 10.0.0.3*. Podle toků uložených v paměti přepínače budou pakety ICMP přesměrovány přes port VTEP ukazující na vzdálené síťové rozhraní s adresou 192.168.3.1/24. Hostitel s adresou 10.0.0.1/24 tedy bude komunikovat s hostitelem 10.0.0.3/24 přes VxLAN s identifikátorem 100.



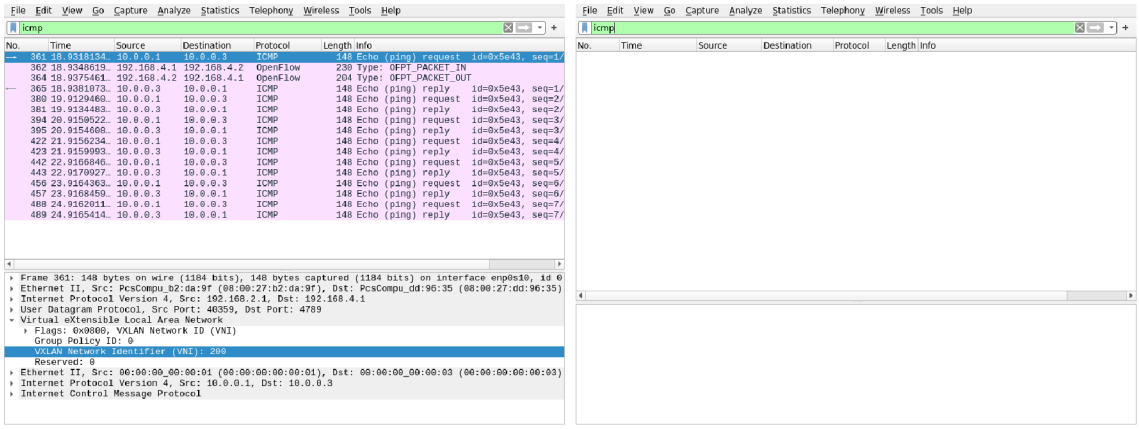
Obr. 2.18: Provoz na 192.168.4.1/24 (vlevo) a 192.168.3.1/24 (vpravo)

Pravidla uložená v paměti přepínače, stejně jako zapouzdření paketů VxLAN pomocí VTEP portů, umožní vyhnout se dodatečnému ARP vysílání a možnému přesměrování paketů do nesprávné podsítě, v případě existenci klientských hostitelů

s IP adresami shodnými s IP adresami hostitelů v sousedních podsítích. Podobný výsledek lze vidět také při nastavování komunikace hostitele s adresou 10.0.0.1/24, který se nachází na podsíti 192.168.2.0/24 s hostitelem 10.0.0.3/24 přes síť VxLAN s identifikátorem 200.

```
$ push-flows 0000000000000002 10.0.0.1 10.0.0.3
```

```
mininet> h1 ping 10.0.0.3
```



Obr. 2.19: Provoz na 192.168.4.1/24 (vlevo) a 192.168.3.1/24 (vpravo)

Použité programy a nástroje:

- Virtualizační nástroj – VirtualBox v6.1
- Operační systém – Ubuntu v18.06
- Síťový kontrolér – ONOS v2.4.0 (stabilní verze)
- Nástroj pro kompilaci – Bazel v6.0.0
- Nástroj pro virtualizaci – Mininet v2.3.0
- Virtuální síťový přepínač – Open vSwitch v2.16.90
- Analyzátor sítového provozu – Wireshark v3.6.0

Závěr

V tomto článku byla prozkoumaná technologie softwarově definovaných sítí, jejich definující vlastnosti a základní prvky, jako jsou OpenFlow, Open vSwitch a ONOS. Byla zvažována řada případů nasazení a integrace principů a prvků sítí SDN v různých oblastech aplikace síťových technologií. V praktické části práce byla pomocí programu Mininet vytvořena virtuální síť ve formě jednoduché trojúhelníkové topologie fungující na bázi softwarového kontroléru ONOS umístěného na lokálním serveru. V průběhu praktického zadání byly zvažovány aspekty interakce ONOS s jím řízenými zařízeními a také struktura předávaných zpráv. V druhé části praktického úkolu, na základě několika virtuálních topologiích obsahujících 2 a více podsítí, byla předvedena komunikace mezi vybranými klientskými hostiteli přes síť VxLAN a také ukázka schopnosti konfigurovat síťová zařízení přímo a vzdáleně pomocí SDN kontroléru. Výsledek tohoto experimentu ukázal, že členové sítě VxLAN jsou schopni mezi sebou bezproblémově komunikovat, aniž by měli informace o implementaci síťové vrstvy. Další zkoumání tématu konfigurace přepínačů pomocí kontroléru ONOS může zahrnovat vzdálené vytváření koncových bodů VxLAN.

Z dat získaných z teoretické a praktické části lze usoudit, že síťový kontrolér ONOS je v základní konfiguraci plně schopen samostatné spravovat malé lokální, rozšiřitelné lokální a globální sítě. Rozšíření funkčnosti a integrace nových aplikací do jádra kontroléru dělá z ONOS důstojného, dobře adaptabilního kandidáta pro řešení problémů úrovně řízení sítí SDN s dynamicky se měnící klientskou základnou a širokou škálou požadovaných funkcí.

Literatura

[1] KREUTZ, D.; RAMOS, F.M.V.; VERISSIMO, P.; ROTHENBERG, C.; AZODOlMOLKY, S.; UHLIG, S.; *Software-Defined Networking: A Comprehensive Survey*, IEEE Systems Journal, 2014, ISSN 0018-9219

[2] KATUKAM, S.; LECLERC, M.; COHN, M.; KANTAK, P.; NATHAN, S.; POODARI, K.; TSOU, T.; *SDN in the Campus Environment*, Open Networking Foundation, 2013 Dostupné z URL:`https://opennetworking.org/wp-content/uploads/2013/03/sb-enterprise-campus.pdf`

[3] BERA, S.; MISRA, S.; VASILAKOS, V. A.; *Software-Defined Networking for Internet of Things: A Survey*, IEEE Systems Journal, 2017 ISSN 2327-4662

[4] GALLUCIO, A.; MILARDO, S.; MORABITO, G.; PALLAZO, S.; *SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks*, IEEE Systems Journal, 2015, ISSN 0743-166X

[5] BERA, S.; MISRA, S.; ROY, S. K.; OBAIDAT, M. S.; *Soft-WSN: Software-Defined WSN Management System for IoT Applications*, IEEE Systems Journal, 2018, ISSN 1937-9234

[6] MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REFORD, J.; SHENKER, S.; TURNER, J.; *OpenFlow: Enabling Innovation in Campus Networks*, ACM SIGCOMM Computer Communication, Volume 38, Number 2

[7] *OpenFlow Switch Specification v1.3.0*, Open Networking Foundation, 2013 Dostupné z URL:`https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf`

[8] PETTIT, J.; LOPEZ, E.; *OpenStack: OVS Deep Dive*, VMware Inc. 2013 Dostupné z URL:`https://www.openvswitch.org/support/slides/OpenStack-131107.pdf`

[9] PFAFF, B.; PETTIT, J.; KOPONEN, T.; JACKSON, E.; ZHOU, A.; RAJAHALME, J.; GROSS, J.; WANG, A.; STRINGER, J.; SHELAR, P.; AMIDON, K.; CASADO, M.; *The Design and Implementation of Open vSwitch*, 12th USENIX Symposium on Networked Systems, USENIX Association

[10] *ONOS project: System Components*, Dostupné z URL:`https://wiki.onosproject.org/display/ONOS/System+Components`

[11] *ONOS project: Southbound*, Dostupné z URL:`https://wiki.onosproject.org/display/ONOS/Southbound`

[12] SECCI, s.; Diamanti, A.; Sanchez, JM; Bah, MT.; Vizzarreta, P.; Mas, C.; Machuca, Scott-Hayward, S.; Smith, D.; *Security and Performance Comparison of ONOS and ODL controllers*, Open Networking Foundation, 2019

[13] VENTRE, P.L.; SALSANO, S.; GEROLA, M.; SALVADORI, E.; USMAN, M.; BUSCAGLIONE, S.; *SDN-Based IP and Layer 2 Services with an Open Networking Operating System in the GÉANT Service Provider Network*, Institute of Electrical and Electronics Engineers, 2017 ISSN 0163-6804

[14] GEROLA, M.; SANTUARI, M.; SALVADORI, E.; SALSANO, S.; CAMPANELA, M.; VENTRE, P.L.; AL-SHABIBI, A.; SNOW, W.; *ICONA: Inter Cluster ONOS Network Application*, Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) ISBN 978-1-4799-7899-1

[15] GEROLA, M.; LUCREZIA, F.; SALVADORI, E.; SALSANO, S.; CAMPANELA, M.; SANTUARI, M.; *ICONA: A Peer-to-Peer Approach for Software Defined Wide Area Networks Using ONOS*, Fifth European Workshop on Software-Defined Networks (EWSDN), 2016 ISSN 2379-0369

[16] SMITH, M.; DVORKIN, M.; ADAMS, R.; LARIBI, Y.; PANDEY, V.; GARG, P.; WEIDENBACHER, N.; *OpFlex Control Protocol draft-smith-opflex-03*, Internet Engineering Task Force 2016

[17] ZHIFENG ZHAO; FENG HONG; RONGPENG LI; *SDN Based VxLAN Optimization in Cloud Computing Networks*, IEEE Access 2017

[18] NOGHANI, K.A; KASSLER, A.; GOPANNAN, P.S.; *EVPN/SDN Assisted Live VM Migration between Geo-Distributed Data Centers*, 2018 IEEE International Conference on Network Softwarization

[19] BENET, C.H; NOGHANI, K.A; KASSLER, A.; *Policy-based Routing and Load Balancing for EVPN-based Data Center Interconnections*, 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks

[20] SENTHIL GANESH N.; RANJANI S.; *Dynamic Load Balancing using Software Defined Networks*, International Journal of Computer Applications (0975 – 8887) International Conference on Current Trends in Advanced Computing (ICCTAC-2015)

Seznam symbolů a zkratek

API Application programming interface. 19, 30, 37, 49
ARP Address Resolution Protocol. 15, 34, 42, 52

BGP Border Gateway Protocol. 31

EVPN Ethernet Virtual Private Network. 34, 35

ICMP Internet Control Message Protocol. 39, 41–43, 49, 52
ICONA Inter Cluster ONOS Network Application. 31, 32
IoT Internet of things. 21, 22
IP Internet protocol. 23, 27, 30, 34, 35, 40, 45, 47, 48, 52, 53
IXP Internet exchange points. 31

LLDP Link Layer Discovery Protocol. 32

MAC Media Access Control. 23, 27, 33–35, 40, 45
MPLS Multiprotocol label switching. 30

NREN National research and education network. 30

ODL OpenDayLight. 30
ONOS Open Network Operating System. 28–32, 37–41, 46, 47, 49, 52, 55
OVS Open vSwitch. 26, 27, 37, 45–47, 52, 55

QoS Quality of service. 21, 22

SDN Software defined network. 19–23, 27, 28, 30, 31, 35, 38, 42, 49, 55
SDX Software defined exchange points. 31

TCP Transmission Control Protocol. 24

VLAN Virtual Local Area Network. 20, 33
VNI Virtual Network Identifier. 47, 50
VTEP VxLAN Tunnel Endpoint. 34–36, 49, 50, 52
VxLAN Virtual extensible Local Area network. 33–35, 37, 45, 47–50, 52, 53, 55

WAN Wide Area Network. 30, 35, 40

A Příloha

A.1 Obsah elektronické přílohy

- *images.zip* – archiv bude obsahovat všechny obrázky použité v této práci
- *push-flows.sh* – skript pro záznam pravidel toku
- *flows.zip* – obsahuje textové soubory s pravidly toku
- *ports.zip* – obsahuje soubory se sadou příkazů pro vytváření portů VTEP
- *topo.zip* – obsahuje skripty pro vytváření jednotlivých Mininet sítí