UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Využití Raspberry Pi jako OpenFlow switch

Bc. Lukáš Holoubek

Diplomová práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš Holoubek**
Osobní číslo:	**I15207**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Využití Raspberry Pi jako OpenFlow switch**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce bude popsat a realizovat možnosti implementace softwarových switchů podporující OpenFlow protokol do Raspberry Pi. Tato implementace umožní testování SDN funkcionalit za použití Raspberry Pi zařízení a libovolného SDN kontroléru. Autor popíše principy fungování různých softwarových switchů a v praktické části pak jejich nasazení do operačního systému Linux běžícího na Raspberry Pi. Funkčnost tohoto řešení bude otestována v topologii cloudového datového centra.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

***AZODOLMOLKY, Siamak. Software defined networking with OpenFlow: get hands-on with the platforms and development tools used to build OpenFlow network applications. Birmingham: Packt Publishing, c2013. ISBN 978-1-84969-872-6.**
***SHUKLA, Vishal. Introduction to software defined networking: OpenFlow & VxLAN. North Charleston: CreateSpace, c2013. ISBN 978-1-48267-813-0.**

Vedoucí diplomové práce: **Ing. Soňa Neradová, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2016**
Termín odevzdání diplomové práce: **17. května 2017**

Univerzita Pardubice -8- Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2016

Prohlášení autora

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 16. 5. 2017 Bc. Lukáš Holoubek

PODĚKOVÁNÍ

Chtěl bych zde poděkovat své vedoucí práce Ing. Soně Neradové, Ph.D. za ochotu při konzultacích v průběhu zpracování celé práce a také své rodině, která mi poskytla potřebné klidné zázemí.

ANOTACE

Práce se věnuje možnostem implementace softwarových přepínačů na počítačích Raspberry Pi. V teoretické části práce je popsána architektura softwarově definovaných sítí a protokol OpenFlow. Jsou zde také představeny softwarové přepínače podporující protokol OpenFlow na OS Linux. Praktická část se věnuje instalaci a konfiguraci zmíněných přepínačů a jejich případným nasazením do prostředí cloudového datového centra.

KLÍČOVÁ SLOVA

Softwarově definované sítě, OpenFlow, Raspberry Pi, Open vSwitch

TITLE

Utilization of Raspberry Pi as an OpenFlow Switch

ANNOTATION

The thesis deals with possible implementations of software switches on Raspberry Pi computers. Theoretical part includes description of software-defined networking and OpenFlow protocol. Set of available software switches on Linux operating systém is listed here as well. Practical part consists of instalation and cofiguration of available software switches and if possible their deployment into cloud datacenter.

KEYWORDS

Software-Defined Networking, OpenFlow, Raspberry Pi, Open vSwitch

OBSAH

SEZNAM ZKRATEK

ACL	Access Control List
API	Application Programming Interface
ARM	Advanced RISC Machine
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuit
BBC	British Broadcasting Corporation
BGP	Border Gateway Protocol
BPDU	Bridge Protocol Data Unit
CAM	Content Addressable Memory
CEF	Cisco Express Forwarding
CLI	Command Line Interface
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CPU	Central Processing Unit
CSI	Camera Serial Interface
DHCP	Dynamic Host Configuration Protocol
DKMS	Dynamic Kernel Module Support
DSI	Display Serial Interface
EU	Evropská unie
FIB	Forwarding Information Base
FIFO	First In, First Out
GB	Gigabajt
GPIO	General Purpose Input/Output

HDMI	High-Definition Multimedia Interface
HP	Hewlett Packard
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protokol
IoT	Internet of Things
IP	Internet Protocol
ISO/OSI	International Standards Organization / Open System Interconnection
IT	Informační technologie
IVS	Indigo Virtual Switch
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
L2	Layer 2
L3	Layer 3
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
LED	Light-Emitting Diode
LINC	Link Is Not Closed
LLDP	Link Layer Discovery protocol
LTS	Long Term Support
MAC	Media Access Control

MLS	Multilayer Switching
MPLS	Multiprotocol Label Switching
MySQL	My Structured Query Language
NAS	Network Attached Storage
NAT	Network Address Translation
NEC	Nippon Electric Company
NETCONF	Network Configuration Protocol
NOOBS	New Out of the Box Software
NTT	Nippon Telegraph and Telephone
OF-CONFIG	OpenFlow Management and Configuration Protocol
ONF	Open Networking Foundation
OS	Operační systém
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVSDB	Open vSwitch Database Management Protocol
OXM	OpenFlow Extensible Match
PBB	Provider Backbone Bridge
PEM	Privacy-enhanced Electronic Mail
PID	Process ID
POF	Protocol-Oblivious Forwarding
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational state transfer

RISC	Reduced Instruction Set Computing
RSTP	Rapid Spanning Tree Protocol
Sb.	Sbírka zákonů
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
STP	Spanning Tree Protocol
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
TLS	Transport Layer Security;
URL	Uniform Resource Locator
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
WAN	Wide Area Network

SEZNAM ILUSTRACÍ A TABULEK

ÚVOD

Diplomová práce se zabývá možnostmi implementace různých softwarových přepínačů pro operační systém Linux. Tyto přepínače podporují protokol OpenFlow a řadí se tedy do oblasti softwarově definovaných sítí. Jejich realizace je předvedena na počítačích Raspberry Pi a otestována v prostředí cloudového datového centra.

Teoretická část práce uvádí čtenáře do oblasti softwarově definovaných sítí, důvodů pro jejich vznik a popisuje princip této síťové architektury. V prostředí softwarově definovaných sítí je velmi rozšířen protokol OpenFlow, který slouží pro komunikaci síťových zařízení s centrálním řídicím prvkem – kontrolérem. Protokol OpenFlow nedefinuje pouze způsob komunikace síťových prvků (přepínačů) s kontrolérem, ale také klade určité požadavky na přepínače, které tento protokol implementují.

Následně je popsána činnost přepínačů na linkové vrstvě modelu ISO/OSI včetně činnosti hardwarových komponent tradičního a OpenFlow přepínače a porovnání hardwarového a softwarového přepínání paketů.

V teoretické části práce je také představen počítač Raspberry Pi, na kterém probíhá implementace a nasazení softwarových přepínačů a kontroléru v praktické části práce.

Praktická část následně obsahuje postup při nasazení jednotlivých přepínačů na vybrané linuxové distribuce dostupné na počítače Raspberry Pi. V průběhu instalace jsou řešeny případné problémy, se kterými by se uživatel mohl setkat. Je zde popsáno nasazení vybraného kontroléru na Raspberry Pi a potřebná konfigurace ke spojení s přepínači. Nasazení bylo otestováno v prostředí coudového datového centra. Topologie a potřebná konfigurace jsou zde zdokumentovány. Nakonec je popsána aplikace kontroléru řídící směrování provozu připojenými přepínači.

1 SOFTWAROVĚ DEFINOVANÉ SÍTĚ

1.1 Motivace k nové síťové architektuře

Rapidní nárůst počtu mobilních zařízení a jejich rozšíření napříč populací, virtualizace serverů a příchod cloudových služeb patří k trendům vedoucím k přezkoumání tradičních síťových architektur. Mnoho konvenčních sítí je hierarchických s vrstvami ethernetových prvků uspořádaných do stromové struktury. Tento design dává smysl, pokud je dominantní výpočetní model klient-server, ale potřebám dynamických výpočtů a úložišť dnešních datových center již nevyhovuje. Následující odstavce popíší některé trendy vytvářející potřebu nového síťového paradigmatu.

Oproti aplikacím využívajících model klient-server, kde se komunikace odehrává převážně mezi jedním serverem a jedním klientem, přistupují dnešní aplikace k různým databázím a serverům zároveň. To vytváří množství provozu spíše „horizontálního“ typu v hierarchické struktuře než klasického „vertikálního,“ kde se řada zařízení dotazuje jednoho nebo málo serverů. Zároveň také uživatelé mění vzory v komunikaci tím, že přistupují k firemním aplikacím a obsahu ze zařízení různých typů, odkudkoliv a v různých časech.

Podniky také začaly v poslední době se zájmem využívat jak privátní, tak veřejné cloudové služby, což vyústilo v nebývalý růst těchto služeb. Flexibilita dostupnosti aplikací, infrastruktury a dalších IT zdrojů začala být žádaná. Poskytování cloudových služeb tak vyžaduje pružné škálování výpočetního výkonu, úložiště i síťových zdrojů, ideálně centralizovaně spravované s běžně dostupnou sadou nástrojů. Mnoho manažerů podnikových datových center proto zvažuje použití cloudových služeb, což ústí v další provoz přes WAN (*Wide Area Network*). Plánované cloudové služby musí být navíc vytvořeny v prostředí zvýšené bezpečnosti a kompatibility s ohledem na reorganizace a slučování podniků.

S časem a dostupnými prostředky je různými podniky a organizacemi uchováváno v elektronické podobě stále větší množství informací převážně historického charakteru, tzv. *Big Data*. Zpracování takových dat často vyžaduje masivní paralelní zpracování na stovkách a tisících serverů, které vyžadují vzájemné spojení. Nárůst množství dat vytváří stálou poptávku po další přenosové kapacitě. (Open Networking Foundation, 2012)

1.2 Omezení současných síťových technologií

1.2.1 Vertikální integrace

Počítačové sítě mohou být rozděleny do tří rovin podle své funkce: datová (*data plane*), řídicí (*control plane*) a rovina managementu (*management plane*). Datová rovina odpovídá síťovým zařízením, které jsou zodpovědné za (efektivní) přeposílání dat. Řídicí rovina představuje protokoly používané k naplnění směrovacích tabulek prvků z datové roviny. Rovina managementu zahrnuje softwarové služby, jako SNMP, používané ke vzdálenému monitorování a konfiguraci řídicí funkcionality. Síťová politika je definována v rovině managementu, řídicí rovina tyto politiky uplatňuje s pomocí datové roviny, která tyto politiky vykonává směrováním provozu příslušným způsobem.

V rámci tradiční IP sítě je řídicí a datová rovina úzce spjata, vestavěna do jednoho zařízení. Celková struktura je pak vysoce decentralizovaná. To bylo považováno v počátcích Internetu za důležité, neboť to bylo považováno za nejlepší způsob dosažení odolnosti vůči výpadkům. Ve skutečnosti byl tento návrh docela efektivní ve smyslu výkonu sítě s rychlým nárůstem rychlosti linek a hustoty portů na zařízeních. Výsledkem je nicméně velmi složitá a relativně statická architektura. To je základním důvodem, proč je správa a řízení tradiční sítě tak komplexní a málo pružné.

Spojení řídicí a datové roviny do jednoho zařízení, tzv. vertikální integrace, má sice výše zmíněné výhody, ale snižuje flexibilitu a překáží inovaci a rozvoji síťové infrastruktury. Důsledkem takovýchto překážek je např. přechod z protokolu IPv4 na novější IPv6, který začal před více než deseti lety a stále není z velké části kompletní. (Kreutz a další, 2014)

1.2.2 Složitost

Počítačové sítě jsou typicky zkonstruované z velkého množství síťových zařízení, jako jsou přepínače, směrovače, firewally atd., s mnoha protokoly, které jsou do nich vestavěné (ať už se instalovaný software nazývá operační systém nebo firmware). Protokoly tedy tíhnou k tomu být definovány odděleně, přičemž řeší jeden konkrétní problém. To ústí v jedno z hlavních omezení dnešních sítí – složitost. Například při přesunu nebo přidání některého síťového zařízení v rámci podnikové sítě je nutné upravit konfiguraci vícera přepínačů, směrovačů, firewallů, aktualizovat ACL (*Access Control List*), VLAN, QoS (*Quality of Service*) a další protokoly pomocí nástrojů na úrovni zařízení.

Inženýři, kteří mají na starost tvorbu a správu takové sítě, jsou zodpovědní za konfiguraci politik, které by měly odpovídat mnoha různým událostem a scénářům napříč sítí. Tyto vysokoúrovňové politiky je pak nutné aplikovat na nízké úrovni pomocí konfiguračních příkazů (často specifických pro konkrétního výrobce). Po zprovoznění například nového virtuálního počítače může trvat IT oddělení hodiny a někdy i dny nakonfigurovat ACL, QoS, bezpečnostní a další politiky napříč celou sítí. Udržovatelnost politik se navíc ztěžuje tím, jak uživatelé přistupují k podnikové síti z mobilních zařízení mimo podnikovou síť. (Open Networking Foundation, 2012; Azodolmolky, 2013)

Chyby v konfiguraci jsou v dnešních sítích extrémně rozšířené. Z jednoho chybně nastaveného zařízení může vzniknout značně nežádoucí chování sítě včetně ztráty paketů, směrovacích smyček nebo tvorby nechtěných cest v síti. (Kreutz a další, 2014)

1.2.3 Statická povaha

Důsledkem složitosti sítí, a tedy i rizika chyb v konfiguracích a následném výpadku služeb při změnách, zůstávají dnešní sítě poměrně statické. Statická povaha tradičních sítí je v přímém kontrastu s dynamikou dnešního serverového prostředí. S příchodem virtualizace došlo k velkému nárůstu počtu hostitelů v síti a od základu změnilo pojetí jejich fyzického umístění. (Open Networking Foundation, 2012)

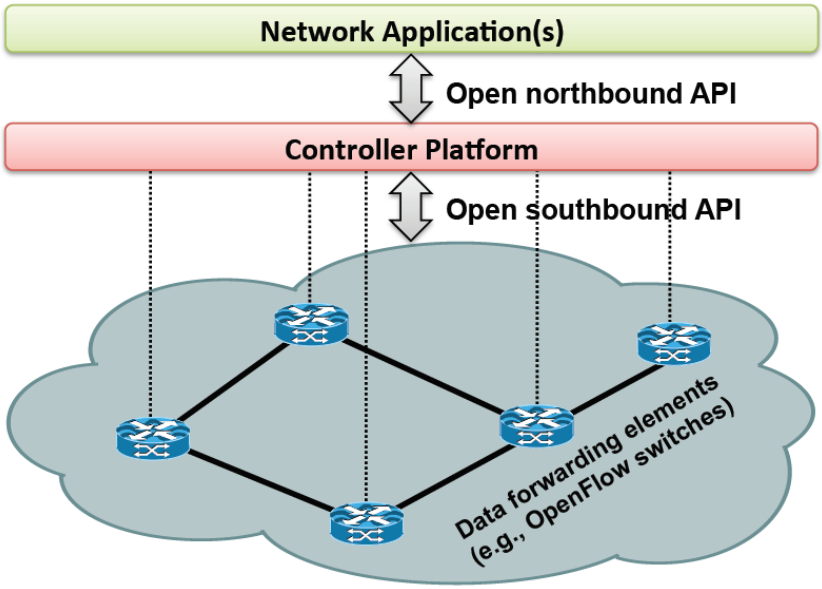
Podniky a poskytovatelé přenosových sítí se snaží rychle odpovídat na změny v požadavcích firem a uživatelů. Jelikož jsou zařízení zahrnující obě zmíněné roviny plně v režii výrobce síťových zařízení, schopnost podniků je v tomto ohledu závislá na produktových cyklech výrobců, které mohou trvat i roky. Nedostatek standardních a otevřených rozhraní také limituje schopnost přizpůsobit síť konkrétnímu prostředí.

Tento nesoulad mezi požadavky trhu a schopnostmi dosavadních sítí vedl kde zlomovému bodu, kdy byla průmyslem vyvinuta architektura softwarově definované sítě (*Software-Defined Networking* – SDN) a započal vývoj přidružených standardů. (Open Networking Foundation, 2012)

1.3 Architektura softwarově definovaných sítí

Softwarově definovaná síť je nově vznikající paradigma, které by mělo přinést změnu v omezení současné síťové infrastruktury. Toto paradigma ruší vertikální integraci na síťových zařízeních oddělením řídicí roviny od roviny datové, tj. samotných směrovačů a přepínačů, které přeposílají provoz. Oddělením řídicí roviny od datové se síťové přepínače stávají

jednoduchými zařízeními směrujícími provoz, zatímco řídicí logika je implementována do logicky centralizovaného kontroléru (*Controller*). To výrazně zjednodušuje aplikování politik, změnu konfigurace sítě a její rozvoj. Funkci kontroléru mohou rozšiřovat síťové aplikace (*Network Applications*). Znázornění takovéto architektury ukazuje Obrázek 1. (Kreutz a další, 2014)



Obrázek 1 – Architektura SDN

Zdroj: (Kreutz a další, 2014)

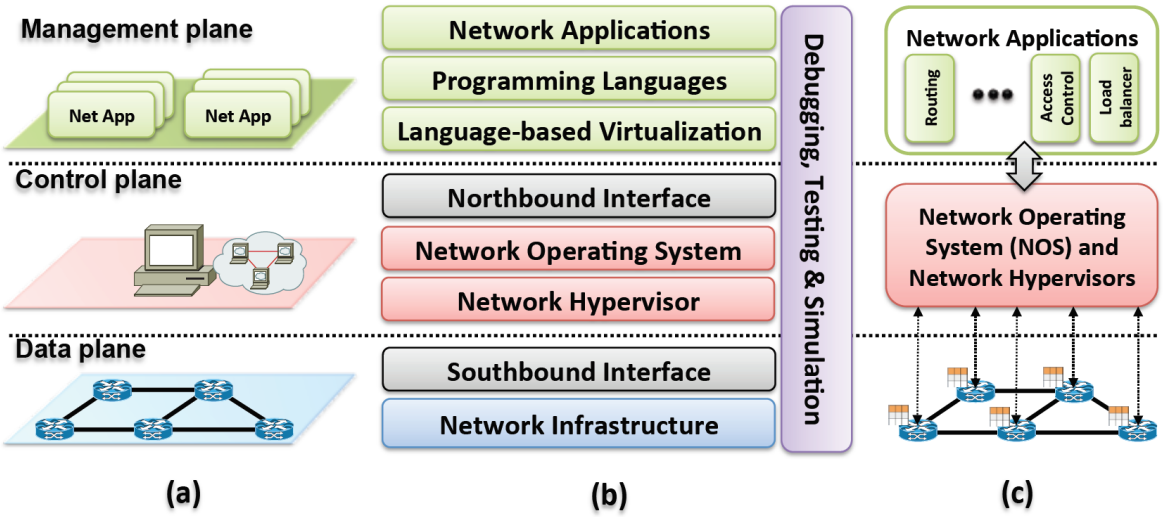
Oddělení řídicí roviny od datové je řešeno programovým rozhraním (API) mezi kontrolérem a přepínači, který pomocí tohoto rozhraní řídí stav datové roviny tvořené připojenými přepínači. Příkladem takového rozhraní je protokol OpenFlow. OpenFlow přepínač obsahuje jednu nebo více tabulek s pravidly pro nakládání s pakety. Každé pravidlo odpovídá určité podmnožině provozu a určuje akce, které budou s tímto provozem provedeny, ať už zahození, úprava nebo přeposlání. Na základě pravidel, která přepínač (*OpenFlow switch*) obdržel od kontroléru, se může tento chovat jako směrovač, přepínač, firewall nebo zastávat jinou funkci, např. *load balancer*.

Důležitý princip softwarově definovaných sítí je princip oddělení zodpovědnosti (*separation of concerns*), tj. oddělení definice politik, jejich implementace v přepínacím hardware a směrování provozu. Toto oddělení je klíčové pro dosažení žádané flexibility, neboť rozdělením problému na zpracovatelné části je snazší vytvořit a zakomponovat nové síťové abstrakce, což usnadňuje správu sítě a vytváří vhodné prostředí pro rozvoj a inovaci. (Kreutz a další, 2014)

1.4 Vrstvy architektury SDN

Architektura SDN může být vyobrazena jako kompozice různých vrstev, které jsou obsaženy v následujícím obrázku. Každá vrstva má svoji specifickou funkci. Zatímco některé vrstvy jsou v softwarově definované sítí přítomny vždy, jako např. *Southbound* a *Northbound Interface*, *Network Operating System* nebo *Network Applications*, jiné se mohou vyskytovat pouze v rámci určitého nasazení. (Kreutz a další, 2014)

V následujících odstavcích je stručně popsána funkce jednotlivých vrstev. Vrstvy jsou popisovány v pořadí zdola nahoru.



Obrázek 2 – Architektura SDN: (a) roviny, (b) vrstvy a (c) systémová perspektiva

Zdroj: (Kreutz a další, 2014)

Vrstva *Network Infrastructure* je v rámci architektury SDN podobná tradiční síti v tom, že sestává ze síťových zařízení (přepínačů, směrovačů). Hlavní rozdíl, jak již bylo zmíněno, však spočívá v tom, že zařízení zde neobsahují vestavěný řídicí software umožňující vykonávání autonomních rozhodnutí. (Kreutz a další, 2014)

Southbound API je programové rozhraní, které propojuje prvky z vrstvy *Network Infrastructure* s řídicími prvky vyšších vrstev. Jde tedy o kritický prvek v oddělení funkcionality datové a řídicí roviny. Ačkoliv je na této úrovni nejznámějším standardem OpenFlow, není to jediné dostupné *Southbound* rozhraní. Patří sem také protokol *OVSDB*, navržený pro pokročilou správu přepínače Open vSwitch, *Protocol-Oblivious Forwarding* (POF), který nabízí vyšší nezávislost od síťových protokolů, nebo *OpFlex* vytvořený firmou Cisco. (Kreutz a další, 2014; Song, 2013; Oswalt, 2014)

Hypervisor je software, který existuje mimo hostovaný operační systém a překládá jeho příkazy k hardware. Jeho hlavní úloha je umožnit víceru „strojů“ sdílet jednu hardwarovou platformu. V prostředí cloudu typu *Infrastructure-as-a-service* může mít uživatel vlastní virtuální zdroje od výpočetních po úložné. Výhodou virtuálních strojů je, že mohou být migrovány z jednoho fyzického serveru na druhý a mohou být vytvořeny/zrušeny na požádání. Přes tyto výhody zůstává síť většinou statická. Mimo poskytnutí kompletní virtualizace by tedy měla síť nabídnout také možnost volby libovolné topologie a adresování. Každý uživatel by měl mít možnost nakonfigurovat si své výpočetní uzly a síť současně. Migrace hostitele by pak měla automaticky spustit migraci odpovídajících portů virtuální sítě. (Kreutz a další, 2014; Garrison, 2011)

Network Operating System, zde zaměnitelný s kontrolérem, je kritický prvek v architektuře SDN, představující podporu pro řídicí logiku (aplikace) k tvorbě konfigurace sítě podle politik zadaných operátorem sítě. Podobně jako tradiční operační systém nabízí tato abstrakce základní služby a programová rozhraní vývojářům aplikací. Pro komunikaci kontrolérů navzájem jsou využívána tzv. východní a západní rozhraní, která jsou základním prvkem distribuovaných kontrolérů. (Kreutz a další, 2014)

Rozhraní *Northbound API* na rozdíl od *Southbound API* slouží pro komunikaci síťových aplikací s řídicí rovinou. *Northbound API* typicky vytváří abstrakci od nízkoúrovňových instrukcí používaných Southbound rozhraním. Řada kontrolérů má toto rozhraní ve formě RESTového API. (Kreutz a další, 2014; Azodolmolky, 2013)

Vrstvy *Language-Based Virualization* a *Programming Languages* spolu úzce souvisí a obě poskytují potřebnou abstrakci pro usnadnění konfigurace sítě. Virtualizační techniky umožňují mimo jiné různé pohledy na jednu fyzickou infrastrukturu. Například pohled ve formě jednoho „velkého přepínače“ by mohl představovat kombinaci několika směrovacích zařízení. Taková abstrakce by značně zjednodušila vývoj a nasazení komplexních síťových aplikací. Příkladem programovacího jazyka, který nabízí takovýto druh abstrakce, je *Pyretic*. Poskytuje programátorům možnost soustředit se na specifikaci síťové politiky na vysoké úrovni abstrakce ve formě funkcí aplikovaných na síť jako celek namísto instalace jednotlivých nízkoúrovňových pravidel na každý přepínač zvlášť. (Kreutz a další, 2014; Reich a další, 2013)

Síťové aplikace představují hlavní řídicí prvky sítě. Implementují řídicí logiku, která je překládána do příkazů pro datovou rovinu. Například směrovací aplikace musí definovat cestu, kterou budou pakety putovat z bodu A do bodu B. K dosažení tohoto cíle musí aplikace

instruovat kontrolér, aby instaloval odpovídající pravidla na zařízení po cestě. Současné síťové aplikace provádí jak tradiční funkce, jako směrování, vyvažování zátěže a uplatňování bezpečnostních politik, tak např. snižování spotřeby energie. (Kreutz a další, 2014)

2 OPENFLOW

Stávající sítě jsou do jisté míry izolované od služeb, které poskytují. Malé množství výrobců nabízí pro účely správy sítě svá proprietární řešení, ať už hardwarového nebo softwarového charakteru. Tato řešení jsou však uzavřená a nelze jejich chování programovat. Často se také prodávané řešení týká pouze části sítě, a společnost je pak nucena zakoupit více produktů od různých výrobců, zatímco stále postrádá centralizovaný nástroj pro celou síť. (Kreutz a další, 2014)

Výzkumníci na Stanfordově univerzitě a na univerzitě v Berkeley začali pracovat na konceptu softwarově definovaných sítí v roce 2002 a v roce 2008 začali s protokolem OpenFlow. Identifikací běžných funkcí ethernetových přepínačů autoři poskytli standardizovaný protokol pro řízení tabulky pravidel na přepínači pomocí software. Protokol OpenFlow umožňuje ovládat přepínač bez nutnosti otevření zdrojových kódů výrobcem. (Lara a další, 2014; Vaughan-Nichols, 2011)

Mezi hlavní možnosti poskytované protokolem OpenFlow patří automatizované a konzistentní nasazení poskytovaných služeb do sítě. Oddělením řídicí roviny do kontroléru poskytuje OpenFlow řízení sítě z centralizovaného pohledu. To umožňuje také relativně snadné zavedení konzistentních bezpečnostních politik napříč sítí. Konfigurace síťových zařízení může být do jisté míry automatizována a dynamicky reagovat na změny v síti. Nad to poskytuje protokol větší prostor pro experimenty nad sítěmi. (Hégr, 2016)

Z počátku byl protokol nasazen na půdě amerických vysokých škol. Také průmysl se začal zajímat o softwarově definované sítě a OpenFlow jako prostředek k zjednodušení správy sítě a snadnějšímu zavedení nové funkcionality. Již v roce 2010 začali výrobci, jako např. Juniper, vydávat své produkty z oblasti SDN a OpenFlow. Mezi výrobce produkující přepínače podporující OpenFlow dále patří Cisco, HP, NEC, Dell, Pronto a další. (Lara a další, 2014; Vaughan-Nichols, 2011)

V roce 2011 byla společnostmi Deutsche Telekom, Facebook, Google, Microsoft, Verizon a Yahoo založena organizace *Open Networking Foundation* (ONF), jejímž úkolem je propagace SDN a sítí založených na OpenFlow. Nyní má tato organizace již přes 140 členů. (Lara a další, 2014; Robuck, 2015)

2.1 OpenFlow – specifikace, protokol nebo architektura?

Na OpenFlow lze nahlížet jako na specifikaci v kontextu OpenFlow přepínače. Přepínač se stává OpenFlow přepínačem tehdy, když implementuje požadavky dané specifikací OpenFlow. Například dle specifikace OpenFlow je nutné, aby přepínač podporoval akci FLOOD na pakety odpovídající určitému pravidlu. Zda bude tato funkcionalita implementována či nikoliv je rozhodnutí výrobce, nicméně OpenFlow přepínač musí tuto funkcionalitu poskytovat.

Protokol OpenFlow definuje formát zpráv předávaných zabezpečeným kanálem mezi řídicí rovinou a OpenFlow přepínačem. Formát zprávy musí být známý oběma komunikujícím stranám. Ve skutečnosti je protokol OpenFlow součástí specifikace OpenFlow a vztahuje se jak na řídicí rovinu OpenFlow, tak na OpenFlow přepínač. (Lara a další, 2014)

Jelikož protokol OpenFlow slouží k oddělení řídicí roviny (ovládající software) od datové (hardware), není pokračovatelem stávajících protokolů sloužících k správě síťových zařízení, jako je např. SNMP. Ačkoliv i v oblasti managementu síťových prvků je k dispozici řada moderních protokolů a přístupů, např. NETCONF, OF-CONFIG, OVSDB nebo REST, mohou být tyto využívány spíše jako doplněk k OpenFlow než jeho náhrada. Protokol OpenFlow lze také nazvat jako *assembler* příkazů vyšší úrovně do nízkoúrovňových příkazů ovládajících příslušná zařízení. (Kubica, 2014)

Nakonec může být OpenFlow v kontextu celé sítě pokládán za architekturu. V síti OpenFlow jsou OpenFlow přepínače řízeny jedním nebo více OpenFlow kontroléry. O této síti je pak možné říci, že podporuje architekturu OpenFlow.

Je však dobré poznamenat, že za datovou rovinu přepínače je kompletně zodpovědný výrobce zařízení. Ačkoliv dva přepínače mohou vyhovovat specifikaci OpenFlow, nemusí implementovat všechny vlastnosti této specifikace. Je tedy možné, že aplikace založená na OpenFlow bude s jedním přepínačem fungovat, zatímco s jiným nikoliv. (Lara a další, 2014)

2.2 OpenFlow přepínač

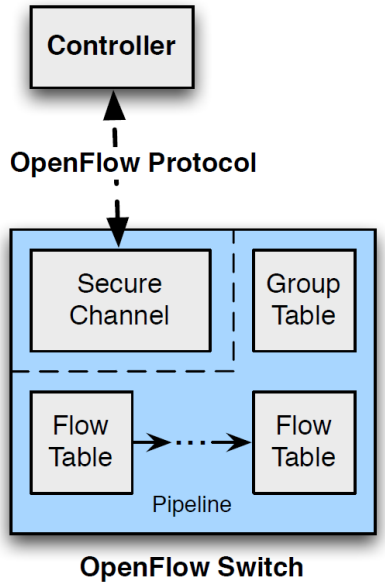
Přepínače v SDN jsou typicky jednoduchá zařízení, která pouze přeposílají provoz. Řídit je lze pomocí otevřeného rozhraní. Tímto způsobem jsou řídicí logika a příslušné algoritmy přesunuty na kontrolér.

OpenFlow přepínače se mohou vyskytovat ve dvou variantách:

- čisté (*OpenFlow-only*),
- hybridní (*OpenFlow-enabled*).

Čisté OpenFlow přepínače neobsahují žádné tradiční řídicí funkcionality a v rozhodování o směrování provozu se plně spoléhají na připojený kontrolér. Většina dostupných komerčních přepínačů je hybridní. Takové zařízení je schopno samostatné činnosti, stejně jako tradiční přepínač nebo směrovač. Hybridní přepínač si může sám rozhodnout, jestli bude provoz zpracovávat podle pravidel OpenFlow nebo jej zpracuje tradičním způsobem pomocí protokolů linkové a síťové vrstvy, ACL nebo QoS. Může tak například vyčlenit některé porty pro zpracování pomocí OpenFlow nebo zpracovat provoz, který protokol OpenFlow určil ke zpracování tradičním způsobem. (Azodolmolky, 2013; The Open Networking Foundation, 2012)

V této kapitole jsou popsány základní komponenty přepínače odpovídající OpenFlow verze 1.3, neboť vyšší verze nejsou mezi výrobci zatím tolik rozšířené (viz také kapitolu 4). Základní části přepínače jsou znázorněny na následujícím obrázku. (The Open Networking Foundation, 2015; Hendriks a další, 2016)



Obrázek 3 – Hlavní komponenty OpenFlow přepínače (v1.3.0)

Zdroj: (The Open Networking Foundation, 2012)

Přepínač si vyměňuje zprávy s kontrolérem po zabezpečeném kanálu. Pomocí protokolu OpenFlow může kontrolér přidat, upravit nebo smazat záznam tabulky pravidel, a to buď reaktivně, jako odpověď na příchozí pakety, nebo proaktivně.

OpenFlow přepínač sestává z jedné nebo více tabulek pravidel (*Flow Table*), tabulky skupin (*Group Table*) a kanálu pro komunikaci s kontrolérem. Tabulka skupin slouží k seskupení akcí, které je potřeba provést nad pakety odpovídajícím různým pravidlům. Tato tabulka může být také odkazována z tabulky pravidel. Skupiny umožňují komplexnější směrování, jako je *multipathing* (využívání více cest do stejného cíle) nebo agregace linek. Přepínač má schopnost omezovat provoz odpovídající určitým pravidlům (tokům) a implementovat další funkcionality QoS. K tomuto účelu využívá přepínač další tabulku, tzv. *Meter table*.

Tabulka 1 – Hlavní položky záznamu tabulky pravidel

Pole pro testování shody	Priorita	Čítače	Instrukce	Čas expirace	Cookie

Zdroj: (The Open Networking Foundation, 2012)

Každá tabulka pravidel na přepínači obsahuje množinu záznamů. Záznamy v této tabulce slouží k identifikaci provozu a provedení různých akcí s tímto provozem. Každý záznam sestává z následujících položek:

- pole pro testování shody s paketem (*match fields*),
- priorita,
- čítače (*counters*),
- sada instrukcí,
- čas do zneplatnění záznamu v tabulce,
- cookie.

Seznam polí pro testování příslušnosti paketu k určitému záznamu tabulky pravidel může obsahovat zdrojovou či cílovou IP adresu, protokol vyšší vrstvy a další pole. Priorita určuje pořadí, ve kterém jsou záznamy tabulky pravidel porovnávány s příchozím paketem. Čítače jsou vedeny pro všechny tabulky, porty, fronty a některé části tabulek. Čítače nejčastěji zaznamenávají počty paketů a bajtů, dobu existence záznamu/portu/fronty atd. Implementace většiny z nich je nepovinná.

Každý záznam tabulky obsahuje sadu instrukcí. Každý paket, který prochází skrze OpenFlow switch má navíc přiřazenou sadu akcí, které s ním mají být provedeny. Instrukce mohou s touto

sadou akcí manipulovat, např. přidávat a mazat akce, nebo okamžitě provést určitý seznam akcí nad paketem. Instrukce mohou také měnit metadata paketu, která slouží k předávání informací mezi tabulkami. K předání paketu k porovnání s jinou tabulkou pravidel slouží instrukce *Goto-Table*. Posledním druhem instrukce je přiřazení měřiče, díky kterému lze omezovat provoz. Každá sada instrukcí může obsahovat nejvýše jeden z každého typu instrukce. Pořadí jejich zpracování je také určitým způsobem omezeno.

Mezi povinně podporované akce, které se mají provést s paketem, patří odeslání paketu na port, zahození paketu a zpracování paketu na základě záznamu v tabulce skupin.

Čas do zneplatnění položky (*timeout*) určuje, po jaké době bude záznam v tabulce pravidel vymazán. Čas do zneplatnění může odpovídat době, po kterou neodpovídal záznamu žádný paket, nebo může být tento čas nastaven napevno. Účel položky cookie je popsán v podkapitole 2.3.5. (The Open Networking Foundation, 2012)

2.2.1 Porty přepínače

Prvním z typů portů OpenFlow přepínače jsou fyzické porty. Ty odpovídají hardwarovým rozhraním. V některých nasazeních mohou být rozhraní přepínače virtualizována. V takovém případě odpovídají fyzické porty této virtualizaci.

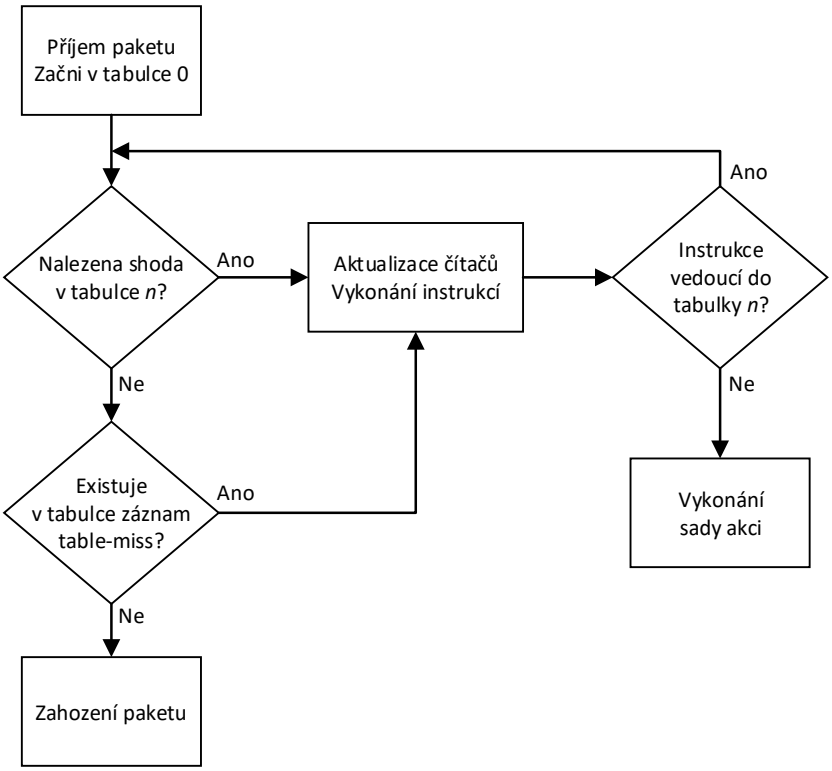
Logické porty nemusí vždy korespondovat s fyzickým rozhraním. Jde o vyšší úroveň abstrakce, která umožňuje přepínači použití metod jako je agregace linek, tunelování nebo rozhraní loopback.

Rezervované porty mohou specifikovat obecné směrovací akce, jako odeslání na kontrolér, rozeslání paketu na všechny porty (FLOOD) nebo zpracování paketu tradičním přepínáním (NORMAL). Poslední dva zmíněné porty FLOOD a NORMAL nejsou podporovány přepínači, který jsou OpenFlow-only. Pokud není k dispozici port FLOOD, lze využít k rozeslání paketu na všechny standardní porty mimo vstupní port paketu port ALL. K odeslání paketu na port, ze kterého byl přijat, lze využít rezervovaný port IN_PORT. Speciální hodnotou v některých příkazech OpenFlow, kdy není port specifikován, je port ANY.

2.2.2 Proces zpracování paketu

Proces porovnávání paketů začíná v první tabulce pravidel, přičemž může pokračovat do dalších. Vývojový diagram jednotlivých kroků tohoto procesu se nachází na následujícím obrázku. Paket je porovnáván se záznamy tabulky v pořadí podle jejich priority. Musí být tedy

vybráno odpovídající pravidlo s nejvyšší prioritou. Pokud je nalezen odpovídající záznam, jsou provedeny instrukce přiřazené tomuto záznamu. (The Open Networking Foundation, 2012)



Obrázek 4 – Vývojový diagram zpracování paketu OpenFlow přepínačem

Zdroj: zpracováno podle (The Open Networking Foundation, 2012)

Pokud není v tabulce pravidel nalezen žádný odpovídající záznam (*table-miss*), je paket zpracován podle tzv. table-miss záznamu. Jde o záznam, který odpovídá všem paketům a disponuje nejnižší (nulovou) prioritou, a který musí každá tabulka pravidel podporovat. Paket, kterému neodpovídal žádný záznam, pak může být například přeposlán na kontrolér, zahozen nebo může pokračovat do další tabulky pravidel. (The Open Networking Foundation, 2012)

2.3 Verze OpenFlow

2.3.1 Nulté verze

První verze protokolu OpenFlow s číslem 0.2.0 byla vydána v březnu roku 2008. V květnu téhož roku přišly verze 0.8.0 a 0.8.1, které přidávaly mj. prioritu k pravidlům (*flow priority*). V říjnu 2008 byla vydána verze 0.8.2, podporující zprávy *Echo Request* a *Echo Reply* pro ověření spojení mezi přepínačem a kontrolérem. Teprve verze 0.8.9 z prosince roku 2008 přidává podporu masek IP podsítí, rozšiřuje možnosti statistik a přináší podporu

protokolu 802.1d *Spanning Tree* (STP) a další aktualizace. (Lara a další, 2014; The Open Networking Foundation, 2015)

Verze 0.9 byla vydána v červenci roku 2009. Přepínač odpovídající této specifikaci může být nyní nakonfigurován se seznamem kontrolérů pro případ selhání aktuálně prvního. Protokol byl také rozšířen o podporu nouzové vyrovnávací paměti pravidel. Ty zůstávají neaktivní, dokud se nepřeruší spojení s kontrolérem. V případě výpadku spojení jsou všechny normální záznamy pravidel zneplatněny a nouzové záznamy jsou zkopírovány do normální tabulky pravidel. Při obnovení spojení s kontrolérem zůstávají všechny záznamy vyrovnávací paměti pravidel (*flow cache*) aktivní a kontrolér má možnost tuto paměť resetovat. Od této verze se také stává TCP port číslo 6633 doporučeným portem pro OpenFlow. Ačkoliv přináší verze 0.9 řadu dalších aktualizací, je již zastaralá a byla nahrazena verzí 1.0. (The Open Networking Foundation, 2015)

2.3.2 Verze 1.0

Nejrozšířenější verzí napříč přepínači je stále verze 1.0.0 vydaná na konci roku 2009. Přepínač implementující tuto specifikaci disponuje pouze jednou tabulkou pravidel a pouze dvanácti poli k porovnávání paketu se záznamem v tabulce pravidel, včetně vstupního portu, jak je obsahuje Tabulka 2. Nicméně již tato verze obsahovala zabezpečený kanál pro komunikaci OpenFlow přepínače s kontrolérem. Implementace tohoto kanálu se mohou lišit, nicméně předávané zprávy musí odpovídat protokolu OpenFlow. Podpora spojení přepínače s více kontroléry je v této verzi nedefinovaná. (Hendriks a další, 2016; The Open Networking Foundation, 2009)

Tabulka 2 – Pole pro testování shody v tabulce pravidel v OpenFlow (v1.0.0)

Ingress Port	Ether source	Ether dst	Ether type	VLAN id	VLAN priority	IP src	IP dst	IP proto	IP ToS bits	TCP/ UDP src port	TCP/ UDP dst port

Zdroj: (The Open Networking Foundation, 2009)

2.3.3 Verze 1.1

Ve verzi 1.1.0 z počátku roku 2011 specifikace OpenFlow již může přepínač obsahovat více než jednu tabulku pravidel a také tabulku skupin. Obrázek 3 tedy popisuje přepínač odpovídající již této verzi specifikace. Zpřístupnění vícera tabulek má nejméně dvě výhody. První výhoda spočívá v efektivnější spolupráci s hardware, který často interně využívá několik tabulek (*L2 table*, *L3 table* nebo pohledy do paměti TCAM – viz také kapitolu 3.4). Druhou výhodou je více tabulek pravidel, což pomáhá rozdělit zpracování paketů tak, aby odpovídalo různým hardwarovým zdrojům. Mnoho sítí také využívá zpracování paketů v několika sadách

(například ACL, QoS nebo směrování). Vložení všech pravidel do jedné tabulky by lehce vyústilo v obrovskou sadu pravidel. Tabulky jsou zřetězeny do tzv. *pipeline*. (Lara a další, 2014; The Open Networking Foundation, 2015)

Předchozí verze OpenFlow předpokládaly, že všechny porty OpenFlow přepínače jsou fyzické. Nově může přepínač pracovat také s virtuálními porty. Rozšířena byla také podpora protokolů VLAN a MPLS. Byla však odstraněna nouzová vyrovnávací paměť pravidel. Poslední zmíněnou aktualizací je nahrazení akcí v záznamech tabulky pravidel instrukcemi. Akce však z tabulky nezmizely, jen byly zahrnuty v instrukcích, které jsou komplexnější a umožňují např. modifikaci paketu nebo poslání paketu do jiné tabulky.

2.3.4 Verze 1.2

V prosinci stejného roku následovala verze OpenFlow 1.2, která opět přináší některé důležité změny. První je tzv. *OpenFlow Extensible Match* (OXM), výrazně vylepšený prostředek k porovnávání paketů, který nahrazuje předchozí statickou strukturu pevné délky. Přidána byla také základní podpora IPv6 včetně ICMPv6 type a ICMPv6 code. Mimo další změny byl uveden mechanismus změny role kontroléru (*equal*, *master* a *slave*). Ten je využíván při připojení přepínače k více kontrolérům pro případ selhání nebo pro vyvažování zátěže. Tento mechanismus je řízen výhradně kontroléry. Přepínač potřebuje znát pouze roli každého připojeného kontroléru.

2.3.5 Verze 1.3

Specifikace OpenFlow verze 1.3 byla vydána v dubnu 2012. Co se děje s paketem, který neodpovídá žádnému záznamu v tabulce pravidel, jak to bylo popsáno v předchozí podkapitole, je novinkou v této verzi. Dříve způsob zpracování tohoto paketu určovaly příznaky v konfiguraci tabulky s třemi možnými akcemi. Rozšířena byla podpora pro IPv6 extension headers. Množství provozu pro jednotlivé toky dat může být nyní řízeno pomocí tzv. *per-flow meters*, které mohou být připojeny k záznamům tabulky pravidel.

V předchozích verzích specifikace bylo spojení přepínače s kontrolérem tvořeno jediným TCP spojením. OpenFlow verze 1.3 umožňuje přepínači vytvoření pomocných spojení k hlavnímu spojení na kontrolér a využití dostupného paralelismu. Pomocná spojení jsou užitečná především pro zprávy typu *Packet-in* a *Packet-out*. Zlepšena byla také podpora prostředí, kde je přepínač připojen k více kontrolérům, přidáním možnosti filtrace asynchronních zpráv od připojených přepínačů.

Do zprávy typu *Packet-in* posílané na kontrolér byla také přidána položka cookie. Ta obsahuje hodnotu záznamu tabulky pravidel, který odeslal paket na kontrolér. Tato hodnota posléze ušetří kontroléru porovnávání paketu vůči celé tabulce pravidel. Mezi další změny patří také přidání políčka trvání (*duration*) ke statistikám nebo podpora *Provider Backbone Bridge* (PBB). V následujících letech byly vydávány ještě další revize této verze od 1.3.1 po 1.3.5 z roku 2015.

2.3.6 Verze 1.4

Verze 1.4 specifikace OpenFlow umožňuje kontroléru v síti, kde vícero kontrolérů může spravovat jeden přepínač, sledovat změny na přepínači učiněné jiným kontrolérem v tabulce pravidel pomocí mechanizmu flow monitoring. Verze 1.2 umožnila kontroléru v takovémto prostředí nastavit si svoji roli vůči jiným kontrolérům. Pokud si nějaký kontrolér například změnil roli ze *slave* na *master*, předchozí *master* získává roli *slave*. Ten však o tom dříve nebyl vůbec informován. To se nyní mění.

Většina tabulek pravidel má omezenou kapacitu. Při zaplnění tabulky nemohou být žádné nové záznamy přidány a na kontrolér je odeslána chyba. Kontrolér v předchozích verzích potřeboval čas k vyřešení tohoto problému, což mohlo vést i k přerušení dostupnosti služby. Verze 1.4 přidává mechanismus umožňující přepínači mazat pravidla s nižší prioritou pro uvolnění místa pro nová pravidla. Kontrolér je o této události informován.

Mezi nové funkce patří také tzv. *bundles*, které umožňují seskupit několik zpráv OpenFlow a vykonat je jako jednu operaci. Mnoho přepínačů provádí vícenásobné hledání nad stejnými daty. Například učení se fyzických adres a přepínání na linkové vrstvě využívá stejnou množinu MAC adres. Funkce synchronizovaných tabulek umožňuje reprezentovat tato data jako dvě tabulky, jejichž obsah je synchronizován, což přináší možnost paralelního přístupu k těmto datům.

Organizace IANA přidělila organizaci *Open Networking Foundation* TCP port 6653 pro použití protokolem OpenFlow, což také ovlivnilo verzi 1.3.3. Všechna použití předchozích čísel 6633 a 976 by tedy měla být přerušena. OpenFlow přepínače a kontroléry musí od této verze používat ve výchozím nastavení port 6653.

Nová verze kromě výše zmíněných nových funkcionalit přenáší také nové chybové kódy, notifikace o změnách v tabulce skupin a v tabulce pro omezení provozu (*Meters Table*),

podporu optických portů a další změny. Revize 1.4.1 z roku 2015 přináší další drobné změny a vyjasnění.

2.3.7 Verze 1.5

Poslední hlavní verze OpenFlow z prosince 2014 nese číslo 1.5. Nejvýznačnější novinkou v této verzi jsou pravděpodobně výstupní tabulky (*Egress Tables*). Doposud bylo zpracování paketu vykonáváno v kontextu vstupního portu. Když je paket odeslán na výstupní port, tehdy začíná být porovnáván se záznamy ve výstupní tabulce, kde je možné sledovat také přiřazený výstupní port. Její chování je z větší části podobné vstupním tabulkám.

Scheduled bundles jsou seskupené zprávy OpenFlow představené v předchozí verzi s tím rozdílem, že jejich vykonávání může být naplánováno na určitý čas. Dále byly rozšířeny statistiky, podobně jako struktura pro porovnávání paketů ve verzi 1.2. Mezi ostatními novinkami lze také zmínit podporu paketů jiného typu než ethernetový rámec, např. PPP, nebo zjištění kontrolérem stavu připojení přepínačů k ostatním kontrolérům. Ve stejný den jako poslední revize dvou předchozích verzí 1.3.5 a 1.4.1 byla vydána i revize této verze 1.5.1 obsahující několik menších úprav. (The Open Networking Foundation, 2015)

2.4 Zprávy OpenFlow

2.4.1 Druhy zpráv

Komunikace mezi přepínačem a kontrolérem se děje s pomocí protokolu OpenFlow, který definuje množinu zpráv, které mohou být mezi těmito prvky vyměňovány. Existují tři druhy zpráv:

A. *controller-to-switch*,
B. *asynchronous*,
C. *symmetric*.

Zprávy typu *controller-to-switch* jsou iniciovány kontrolérem a slouží buď ke správě, nebo ke zjištění stavu přepínače. Naproti tomu asynchronní zprávy jsou iniciovány přepínačem a slouží k notifikaci kontroléru o událostech na síti nebo změnách stavu přepínače. Symetrické zprávy mohou být poslány oběma stranami bez předchozího vyžádání.

A. První skupinou zpráv jsou zprávy posílané kontrolérem přepínači. Při ustanovení spojení jsou běžně používány zprávy *Features*, kterými se kontrolér dotazuje přepínače na jím

podporované funkce. Konfigurační parametry může kontrolér zjišťovat nebo nastavovat pomocí zpráv *Configuration*.

Primární účel zpráv typu *Modify-State* je přidávat, měnit a mazat záznamy v tabulce pravidel nebo skupin a měnit vlastnosti portů, jinými slovy měnit stav přepínače. Stav přepínače, což může také představovat aktuální konfiguraci, statistiky nebo schopnosti, může být zjišťován zprávami *Read-State*.

Packet-out jsou zprávy používané kontrolérem k odeslání paketu z určitého portu přepínače a k přeposílání paketů přijatých ve zprávě *Packet-in*. Zprávy *Packet-out* musí obsahovat buď celý paket, nebo ID bufferu identifikující paket na přepínači. Zpráva také musí obsahovat seznam akcí k provedení s paketem.

Pokud potřebuje kontrolér získat upozornění o dokončených operacích, může využít zprávu typu *Barrier*. Zpráva nemá žádné tělo a před vrácením odpovědi kontroléru musí přepínač dokončit zpracování všech dříve přijatých zpráv.

V prostředí, kde je přepínač připojen k více kontrolérům, jsou užitečné zprávy typu *Role-Request*, které nastavují nebo zjišťují roli kontroléru vůči ostatním kontrolérům. V prostředí více kontrolérů je také užitečná zpráva *Asynchronous-Configuration*, kterou používá kontrolér k filtrování zpráv, které chce od přepínače na svém kanálu přijímat, nebo k dotazování se a tento filtr.

B. Mezi asynchronní zprávy, tj. zprávy odesílané přepínačem na kontrolér, patří především zprávy *Packet-in*. Událost *Packet-in* nastává vždy, když je paket odeslán na rezervovaný port CONTROLLER, a to buď záznamem v tabulce pravidel, nebo tzv. table-miss záznamem. Pakety pro tyto události mohou být ukládány do vyrovnávací paměti. Na kontrolér jsou poté odesílány pouze části těchto paketů. Jejich zpracování pak obvykle probíhá podle zprávy *Packet-out*.

Zpráva typu *Flow-Removed* slouží, jak již název napovídá, k informování kontroléru o smazání záznamu z tabulky pravidel. To se týká pouze záznamů s nastaveným příslušným příznakem. Odeslání zprávy může být výsledkem expirace záznamu nebo požadavku na smazání od kontroléru. Při změně stavu nebo konfigurace portu se očekává odeslání zprávy *Port-status* kontroléru. Přepínač je samozřejmě schopen upozornit kontrolér také na chyby pomocí zpráv typu *Error*.

C. Jednou ze symetrických zpráv je již zmíněná zpráva *Hello*, která se využívá pro navázání spojení přepínače s kontrolérem. Zprávy *Echo Request* a *Echo Reply* jsou využívány především pro ověřování funkčnosti spojení mezi kontrolérem a přepínačem. Mohou však také pomoci zjišťovat odezvu nebo šířku pásma. Posledním typem zprávy je *Experimenter*, který nabízí prostor pro přidání nových funkcionalit v rámci zpráv OpenFlow. (Azodolmolky, 2013; The Open Networking Foundation, 2012)

2.4.2 Navázání spojení

Jak ukazuje Obrázek 3, přepínač s kontrolérem komunikují zabezpečeným kanálem. Jediná podmínka pro síť přenášející komunikaci mezi OpenFlow přepínačem a kontrolérem je podpora protokolu TCP/IP. Kanál je obvykle šifrován pomocí protokolu *Transport Layer Security* (TLS). Kontrolér a přepínač se navzájem autentizují výměnou certifikátů podepsaných klíčem pro danou síť. (Azodolmolky, 2013; The Open Networking Foundation, 2012)

Alternativně lze pro komunikaci přepínače s kontrolérem využívat prosté TCP. V takovém případě se však doporučuje zajištění alternativních bezpečnostních opatření zabraňujících odposlechu, podvržení kontroléru či jiným útokům.

Navázání spojení vždy začíná přepínač, a to obvykle při svém zapnutí. Jelikož je v této kapitole uvažována verze OpenFlow 1.3 z důvodu zmíněném v kapitole o OpenFlow přepínači, platí jako výchozí port kontroléru TCP port 6633. Když je spojení navazováno, obě strany si musí poslat zprávu *Hello*, která obsahuje nejvyšší podporovanou verzi OpenFlow na odesílajícím zařízení. Po přijetí zprávy je dohodnuto používání nižší verze z přijaté zprávy a verze podporované příjemcem. Pokud příjemce podporuje vyjednanou verzi, spojení je ustanoveno. V opačném případě je odeslána příslušná chybová zpráva.

Při výpadku spojení způsobeném například vypršením echo request timeout nebo TLS session timeout přechází přepínač v závislosti na své implementaci a konfiguraci okamžitě do stavu *fail secure mode* nebo *fail standalone mode*. V režimu *fail secure mode* je jedinou změnou v chování přepínače zahození všech paketů směřujících na kontrolér. Záznamy v tabulce pravidel expirují dle svých intervalů. Přepínač v režimu *fail standalone mode* zpracovává všechny pakety pomocí rezervovaného portu NORMAL a chová se tedy jako tradiční přepínač nebo směrovač. Tato funkcionalita je, jak již bylo zmíněno, obvykle dostupná pouze na hybridních přepínačích. V jednom z těchto režimů se přepínač také nachází okamžitě po svém zapnutí před navázáním spojení s kontrolérem.

Provoz mezi kontrolérem a přepínačem nepodléhá zpracování v *OpenFlow pipeline*, tj. porovnávání s tabulkami pravidel. (The Open Networking Foundation, 2012)

2.4.3 Zpracování zpráv

Protokol OpenFlow poskytuje spolehlivé doručení zpráv a jejich zpracování, ale neposkytuje automatická potvrzení ani nezajišťuje dané pořadí zpracování zpráv. Spolehlivé doručení však není zaručeno u pomocných spojení na kontrolér, která používají nespolehlivý způsob přenosu. Přepínače musí plně zpracovat každou zprávu přijatou od kontroléru a případně odeslat odpověď. Pokud přepínač nemůže zprávu od kontroléru plně zpracovat, musí odpovědět chybovou zprávou. Kontroléry mohou ignorovat jakékoliv zprávy, které dostávají od přepínačů. Měly by však odpovídat na zprávy typu *Echo*, aby připojený přepínač neukončil s kontrolérem spojení.

Pořadí zpracování zpráv může být zajištěno bariérovými zprávami. Bez těchto zpráv může přepínač v některých případech pro maximalizaci výkonu změnit pořadí vykonávání zpráv. Například záznamy tabulky pravidel mohou být do tabulky uloženy v jiném pořadí, než v jakém byly přepínačem přijaty. Kontroléry by tedy neměly spoléhat na nějaké specifické pořadí zpracování. Změny v pořadí zpracování však nesmí překročit hranice dané bariérovými zprávami, jak již bylo zmíněno výše. (The Open Networking Foundation, 2012)

3 ČINNOST PŘEPÍNAČE

3.1 Činnost přepínače na linkové vrstvě

Přepínač na linkové vrstvě ISO/OSI modelu, tzv. *L2 switch* (někdy nazývaný také *bridge*, neboť se shodují v základní činnosti), je primárně zodpovědný za přeposílání rámců na lokální síti (LAN). Přepínání rámců funguje v základu tím způsobem, že přepínač přijme na vstupní port rámec a podle cílové adresy rámce určí výstupní port. Touto cestou je zajištěno, že rámce putují pouze skrze určené porty a zbytečně nezahlcují provozem ostatní porty. Při přepínání na 2. vrstvě jsou využívány tyto tři základní operace:

1. *Address learning* – přepínač si automaticky ukládá do tabulky MAC adres zdrojové MAC adresy příchozích rámců k příslušným portům,
2. *L2 forwarding* – rámce jsou odeslány podle tabulky MAC adres na příslušné porty (pokud přepínač nemá k cílové adrese přiřazený port, je rámec odeslán na všechny ostatní porty, tzv. *flooding*),
3. *Address ageing* – záznamy v tabule MAC adres jsou po určité době jejich nepoužití vymazány – typicky pro případ odpojení/přesunu připojeného uzlu. (Bavota, 2017)

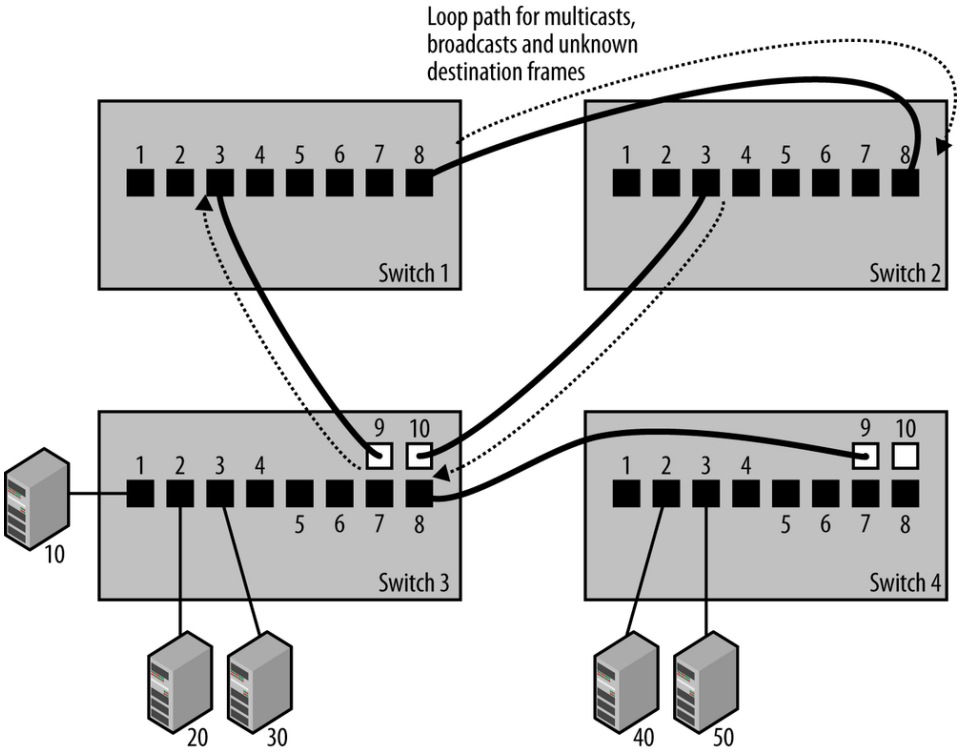
Ethernetový přepínač je navržený tak, aby pracoval pro zařízení na síti neviditelně. Tento princip je také nazýván *transparent bridging*. Protože přepínač rámce, které jím procházejí, nijak nemodifikuje, zařízení v síti nemusí o jeho přítomnosti vůbec vědět. (Zimmerman a další, 2013)

3.2 Problémy smyček v síti

Návrh a činnost ethernetu vyžaduje existenci pouze jedné cesty mezi dvěma uzly sítě. Jak se síť rozrůstá, zvyšuje se i složitost sítě, a tedy i pravděpodobnější výskyt smyčky v síti. Smyčky mohou vzniknout nejméně ze tří různých důvodů:

- omylem návrháře sítě,
- redundancí, tj. nadbytečností prvků pro zvýšení spolehlivosti v případě výpadku,
- vyvažováním zátěže mezi prvky zvýšením počtu jejich spojení.

Pokud přepínače vytvoří logickou smyčku, mohou mezi nimi začít rámce kolovat do nekonečna, tzv. *broadcast storm*, což způsobí neúnosné zatížení sítě, případně další problémy. (Zimmerman a další, 2013; Bouška, 2005)



Obrázek 5 – Logická smyčka mezi přepínači

Zdroj: (Zimmerman a další, 2013)

V zapojení, které obsahuje Obrázek 5, je viditelný výskyt takové smyčky v síti. Pro snažší pochopení je zde uveden příklad výskytu broadcastové bouře. Libovolné zařízení v síti odešle broadcastový rámec. Switch 3 jej přepošle na všechny porty kromě vstupního, tedy i na Switch 1 a Switch 2. Switche si rámce pošlou navzájem a posléze zpět přepínači č. 3. Ten broadcastové rámce znovu rozešle a takto rámce dále kolují v síti, neboť rámce nedisponují žádnou položkou omezující počet skoků mezi uzly sítě. Pokud by mezi přepínači existovaly redundantní spoje, broadcastové rámce by navíc přibývaly, dokud by se síť nezahltila.

Podobný problém se může vyskytnout i při odeslání unicastového rámce. Například zařízení č. 10 chce poslat rámec zařízení č. 30. Jelikož Switch 3 od zařízení č. 30 zatím žádný rámec nedostal, nemá k tomuto portu přiřazenou jeho MAC adresu. Rámec adresovaný zařízení č. 30 je tedy přeposlán na všechny ostatní porty. Tyto přepínače cílovou adresu též neznají, takže rámec posílají dále, až se rámec z obou přepínačů vrátí na Switch 3. Jelikož se na Switch 3 vrátí rámce se zdrojovou MAC adresou zařízení č. 10, jsou další rámce přeposílány na port, na který

naposledy přišel rámec s touto zdrojovou adresou, a přiřazená adresa již nemusí neodpovídat skutečnosti.

Tento jev se nazývá nestabilita tabulek MAC adres. Díky smyčkám se na přepínač vrací rámce, které byly původně odeslány z jiného portu. V takové situaci změní přepínač záznam v tabulce MAC adres a zdrojovou MAC adresu přiřadí novému (nesprávnému) portu. Přepínač pak předpokládá, že stanice je připojená k jinému portu, a rámce doručuje na špatné místo. Tento port se navíc díky příchozím rámcům neustále mění. (Bouška, 2005)

Řešení tohoto problému přináší standard IEEE 802.1d, který popisuje činnost protokolu *Spanning Tree*. Každý přepínač vyhovující tomuto standardu musí tento protokol implementovat. (Zimmerman a další, 2013)

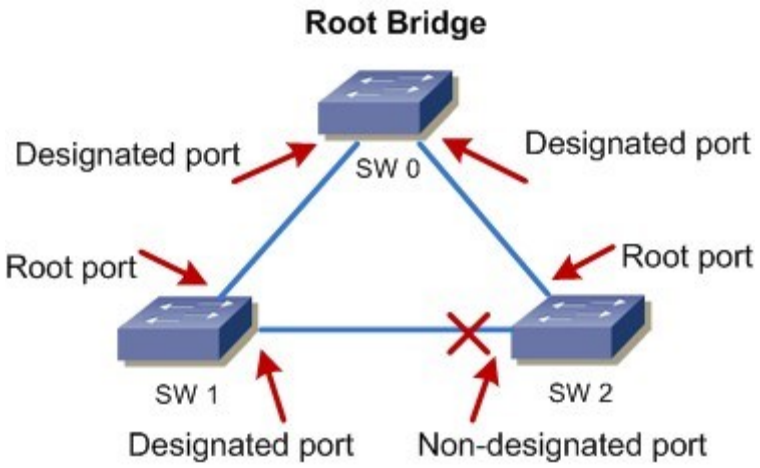
3.3 Spanning Tree Protocol

Počítačovou síť lze vyjádřit formou grafu, kde vrcholy jsou síťová zařízení (typicky přepínače) a hrany jsou jejich spojení. *Spanning Tree Protocol* vytváří logickou kostru sítě – stromovou strukturu propojující všechny přepínače bez výskytu smyček. Přepínače se v rámci protokolu domlouvají pomocí zpráv *Bridge Protocol Data Unit* (BPDU). Podle informací v nich obsažených se volí tzv. *Root Bridge*, jenž je kořenem stromu – přepínač s nejnižším *Bridge ID*. Obvykle jde o centrální prvek. Z tohoto důvodu by měl být zvolený přepínač zároveň nejvýkonnější.

Za účelem dosažení stromové struktury sítě jsou jednotlivé porty přepínačů nastaveny na jeden ze tří typů:

- *Root port* – port disponující nejkratší cestou k *Root Bridge*,
- *Designated port* – port připojený k jinému přepínači než *Root Bridge*,
- *Non-designated port* – blokovaný port. Jeho aktivace by způsobila smyčku v síti. Může fungovat jako záložní port v případě selhání aktivního portu. (Zimmerman a další, 2013; Bouška, 2005)

Umístění portů ve vzorové topologii obsahuje Obrázek 6.



Obrázek 6 – Typy portů protokolu STP

Zdroj: (Bouška, 2005)

Protokol STP může být také využit k rozkládání zátěže napříč VLAN, a to buď podle priority portu, nebo ceny cesty. Nad standard IEEE 802.1d vznikla ještě další vylepšení tohoto protokolu umožňující běh protokolu STP na více VLAN nebo rychlejší konvergenci, ta jsou však již nad rámec této práce. (Bouška, 2005)

3.4 Hardwarové přepínání

3.4.1 Části tradičního přepínače

Průmyslový termín hardwarové přepínání (*hardware-switching*) označuje zpracovávání paketů jakékoliv vrstvy ISO/OSI modelu od druhé po sedmou pomocí specializovaných hardwarových komponent, tzv. *Application-Specific Intergrated Circuits* (ASIC). Tyto komponenty – obvody obvykle dosahují propustnosti na úrovni kabelové rychlosti bez degradace výkonu včetně využití pokročilých funkcí, jako je QoS, zpracování ACL nebo *IP rewriting*.

Jiný termín, který kdysi označoval hardwarové přepínání například v souvislosti s přepínačem Catalyst 5500, je *Multilayer Switching* (MLS). V této souvislosti může být však tento termín zavádějící. Dnes totiž označuje schopnost směrování paketů na síťové a přepínání rámců na linkové vrstvě zároveň v rychlosti všech připojených linek, tzv. *line-rate*. Při tom dokáže využívat pokročilé funkce jako je NAT, QoS, ACL atd. pomocí obvodů ASIC.

Komponenty ASIC bývají omezeny velikostí paměti, což například ovlivňuje počet záznamů ACL, které může přepínač udržovat. Velikost paměti ASIC obecně souvisí s cenou a způsobem použití přepínače. (Froom a další, 2015)

Porty přepínače na 2. vrstvě modelu ISO/OSI též využívají fronty příchozích a odchozích rámců. Z těchto front přepínač odebírá přijaté rámce k analýze a přeposlání, resp. do nich vkládá pro jejich odeslání. Port přepínače může využívat i více vstupních nebo výstupních front, což umožňuje prioritizaci provozu.

Na přepínačích Cisco je tabulka MAC adres uchovávána v paměti *Content Addressable Memory* (CAM). Paměť CAM se od mnohem rozšířenější paměti RAM liší především v přístupu k záznamům. Zatímco paměť RAM je dotazována na určitou fyzickou adresu v paměti, načež vrátí data na daném místě. Paměť CAM funguje na přesně opačném principu. Namísto toho jsou data v paměti CAM hledána podle obsahu. Poté je vrácen všechen odpovídající a přiřazený obsah. V případě přepínačů je vyhledáván podle MAC adresy příslušný odchozí port – přiřazený obsah.

Záznamy v paměti CAM, které nebyly více než 300 sekund použity, jsou po této výchozí době smazány s výjimkou statických MAC adres. Příslušný časovač je resetován s každým přijetím rámce s odpovídající MAC adresou a portem. Pokud přepínač přijme rámec se stejnou zdrojovou MAC adresou z jiného portu, je původní záznam okamžitě smazán, což je žádané chování. Adresy MAC jsou brány za unikátní, proto by MAC adresa neměla být přiřazena více než jednomu portu. Záznam v paměti obsahuje kromě MAC adresy a portu také číslo VLAN pro port a časové razítko indikující zastarání.

Pro zvýšení výkonu na síťové vrstvě lze využít speciální paměť, tzv. *Ternary Content Addressable Memory* (TCAM). Paměť TCAM poskytuje vysokorychlostní přístup pro dvě další funkce:

- filtrování provozu s použitím ACL,
- prioritizaci provozu pomocí QoS.

Některé přepínače 3. vrstvy ukládají směrovací tabulku také v paměti TCAM. Většina L3 přepínačů podporuje pro oddělenou správu ACL pro příchozí a odchozí provoz a QoS vícero pamětí TCAM. Každý záznam paměti TCAM sestává ze tří částí:

- hodnoty (*values*) – adresy a/nebo porty,
- masky (*masks*) – udává, jaká část adresy se musí shodovat,
- výsledek (*result*) – udává akci, která je pro odpovídající záznamy vykonána (*permit*/*deny* pro ACL, jiné hodnoty pro QoS).

Přepínače na 3. vrstvě využívají navíc pro směrovací rozhodnutí *Forwarding Information Base* (FIB). Jde o reorganizovanou směrovací tabulku, kde jsou pro zrychlení hledání nejspecifičtější cesty umístěny na vrcholu. Veškeré změny ve směrovací tabulce jsou do této tabulky ihned promítány. (Balchunas, 2014b; Balchunas, 2014a)

Tabulku FIB pak používá komponenta *Layer-3 Forwarding Engine*, která podle této tabulky hardwarově přepíná provoz, což ústí v menší odezvu než v případě komponenty *Layer-3 Engine*, která provádí směrování tradičním způsobem. Pokud nemůže být paket směrován pomocí *Forwarding Engine*, je směrován pomocí *Layer-3 Engine*.

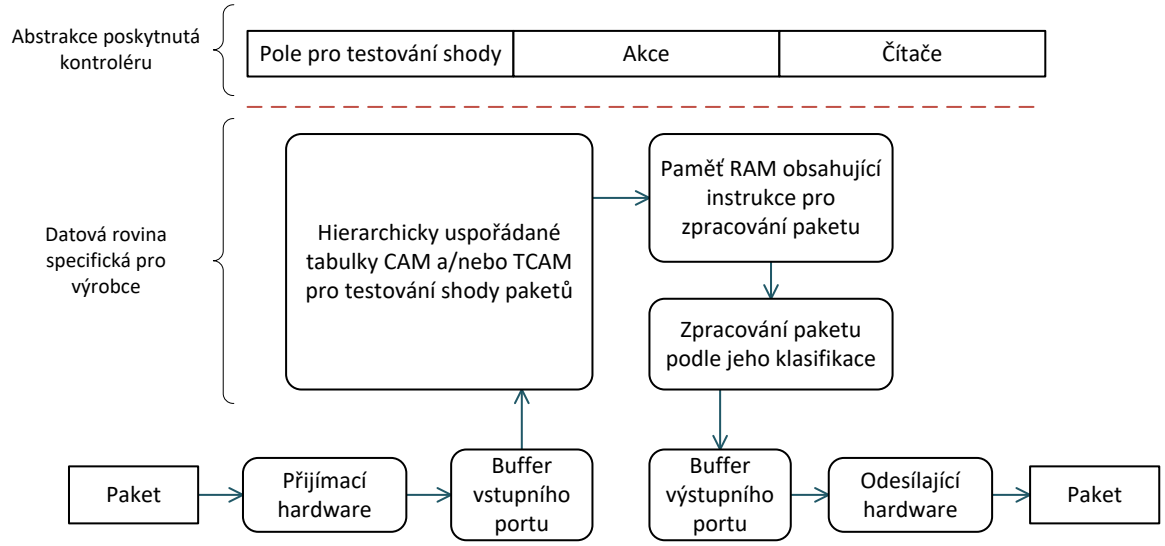
Záznamy tabulky FIB obsahují *next-hop*[1] 3. vrstvy pro každou cílovou síť. Implementace vícevrstvého přepínání od firmy Cisco – *Cisco Express Forwarding* (CEF), která obsahuje zmíněné dvě komponenty, dále vytváří tabulku MAC adres pro každý *next-hop* v tabulce FIB, tzv. *Adjacency Table*, čímž je odstraněno zpoždění spočívající v ARP dotazech při směrování provozu. Pokud není žádaná MAC adresa nalezena, je paket předán komponentě *Layer-3 Engine*, která získá požadovanou adresu s pomocí ARP dotazu. (Balchunas, 2014a)

3.4.2 Části OpenFlow přepínače

Přepínač vyhovující specifikaci OpenFlow musí být schopen směrovat pakety způsobem daným záznamy v tabulce pravidel. Obrázek 7 ukazuje, jakým způsobem síťové zařízení směruje přijaté pakety a některé interní komponenty, které k tomuto účelu využívá.

[1] Next-hop označuje sousední, přímo připojený síťový prvek.

Ke zrychlení přepínání mohou být využity výše zmíněné hardwarové tabulky CAM nebo TCAM. Instrukce pro zpracování paketu jsou pak uloženy v paměti RAM. (Lara a další, 2014)

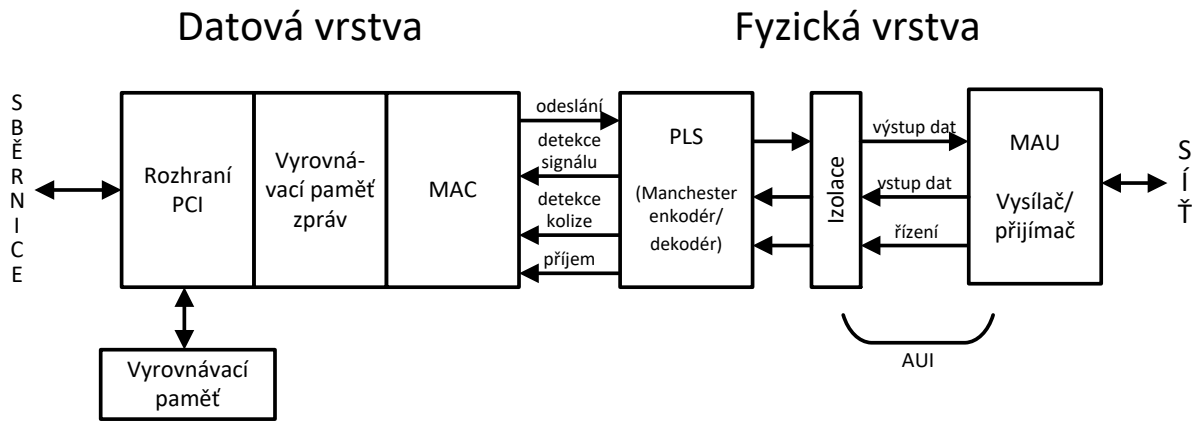


Obrázek 7 – Prvky OpenFlow přepínače

Zdroj: zpracováno podle (Lara a další, 2014)

3.5 Hardwarová část softwarového přepínání

Pro přepínání rámců lze na síti využít softwarové přepínání. To umožní vyhnout se přílišné závislosti na výrobci hardware. Softwarové přepínání již nevyužívá pro svoji práci specializované obvody, nýbrž využívá CPU. Pro připojení počítače do sítě je stále nutné určité vybavení, typicky realizované síťovou kartou. Ačkoliv implementace jednotlivých síťových karet jsou různé, s různými vylepšeními, jejich základní architektura je stále stejná. Obrázek 8 znázorňuje blokové schéma ethernetové síťové karty.



Obrázek 8 – Blokové schéma ethernetové síťové karty

Zdroj: zpracováno podle (Sterling, 2002)

Vrstva datová je zodpovědná za sestavení zprávy do takové podoby, jaká bude aplikována na přenosové médium. Když je zpráva přijímána, datová vrstva přijme paket z fyzické vrstvy a určí, zda je zpráva opravdu určena pro toto zařízení. Dále zkontroluje bitovou integritu, rozbalí data do sekvence bajtů, vloží data do vyrovnávací paměti a předá procesoru hostitelského zařízení. Podobně jsou i výstupní data nejprve ukládána do vyrovnávací paměti typu FIFO.

Fyzická vrstva pak převede zprávu přijatou od datové vrstvy na elektrické signály a odešle na přenosové médium. (Sterling, 2002)

3.6 Porovnání hardwarového a softwarového přepínání

Přepínání a směrování síťového provozu pomocí hardwarové metody je značně rychlejší než tradiční softwarové s použitím CPU. Mnoho obvodů ASIC, obzvláště ty pro 3. vrstvu ISO/OSI modelu, zvyšuje výkon použitím pamětí TCAM společně s algoritmy pro porovnávání paketů, zatímco CPU v roli univerzální výpočetní jednotky přidává výkon především vyšším počtem operací za sekundu. Obvody ASIC se v přepínané architektuře oproti CPU lépe škálují. Navíc mohou být integrovány k jednotlivým portům a přepínat pakety distribuovaným způsobem. (Froom a další, 2015)

Softwarové přepínání však přináší mnohem větší flexibilitu a větší nezávislost na hardware. Například pro přidání nové funkce již není nutné kupovat celé nové zařízení nebo jeho část, ale pouze aktualizovat programové vybavení.

4 SOFTWAROVÉ PŘEPÍNAČE

Tato kapitola obsahuje přehled dostupných softwarových přepínačů s otevřeným zdrojovým kódem pro operační systém Linux. Všechny zmíněné přepínače podporují protokol OpenFlow v různých verzích. Na konci kapitoly je výčet přepínačů shrnut v přehledné tabulce.

4.1 Open vSwitch

Prostředí virtualizovaných serverů používá virtuální přepínač (*vswitch*) pro směrování provozu jak mezi jednotlivými virtuálními stroji na jednom fyzickém hostiteli, tak mezi hostiteli po fyzické síti. Open vSwitch (OVS) je open-source softwarový přepínač navržený jako virtuální přepínač právě pro taková prostředí. Je otevřený programovým rozšířením a ovládání pomocí protokolu OpenFlow a protokolu pro správu – OVSDB (*Open vSwitch Database*). (Pfaff a další, 2013)

Na rozdíl od tradičních síťových zařízení, ať softwarových nebo hardwarových, které dosahují vysokého výkonu díky své specializaci, je Open vSwitch navržen především pro flexibilitu a univerzálnost použití. (Pfaff a další, 2015)

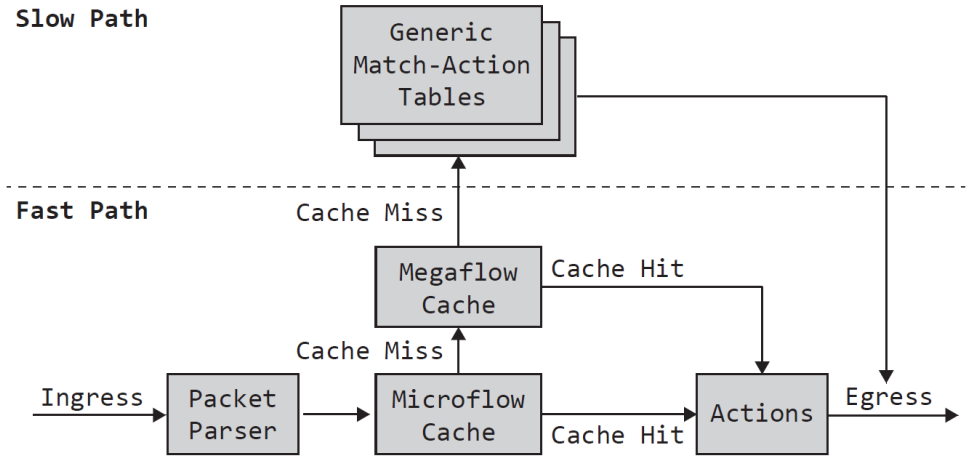
Open vSwitch pracuje na více vrstvách ISO/OSI modelu, jde tedy o tzv. *multilayer switch*. Navíc je navržen tak, aby podporoval distribuovanost na více fyzických serverech, podobně jako virtuální přepínač vNetwork Distributed Switch od firmy VMware nebo Nexus 1000V od firmy Cisco. Open vSwitch tedy může pracovat jako softwarový přepínač běžící uvnitř hypervisora[2] i jako software ovládající specializovaný hardware. (Azodolmolky, 2013)

Převážná část zdrojového kódu je napsána v platformně nezávislém jazyce C. Od verze Linuxu 3.3 je Open vSwitch součástí kernelu. Balíčky spouštěné v uživatelském režimu jsou dostupné pro nejznámější linuxové distribuce – *Ubuntu*, *Debian* a *Fedoru*. Podporovány jsou také unixové operační systémy FreeBSD a NetBSD. Je také integrován do řady systémů pro správu virtuálních strojů, jako například OpenStack, openQRM, OpenNebula a oVirt. Distribuce probíhá pod licencí Apache 2.0.[3] (Pfaff a další, 2015; Azodolmolky, 2013)

[2] Hypervizor je také někdy nazýván Virtual Machine Manager (VMM). (Garrison, 2011)
[3] Repozitáře se zdrojovými kódy přepínače Open vSwitch jsou dostupné na webu https://github.com/openvswitch. Přepínač lze také stáhnout na oficiálních stránkách http://openvswitch.org.

Přepínač Open vSwitch se skládá ze dvou důležitých částí:

- *Slow Path*,
- *Fast Path*.



Obrázek 9 – Části Open vSwitch – Slow Path a Fast Path

Zdroj: (Shahbaz a další, 2016)

Část *Slow Path* je program v uživatelském režimu, kde se nachází většina inteligence OVS. Část *Fast Path* funguje jako vrstva pro vyrovnávací paměť. Každý paket přicházející na Open vSwitch prochází nejprve vrstvou vyrovnávací paměti. Pokud zde není žádný odpovídající záznam (*cache miss*), je paket přesunut do části *Slow Path* pro získání instrukcí k dalšímu zpracování paketu, jak to znázorňuje Obrázek 9. (Shahbaz a další, 2016)

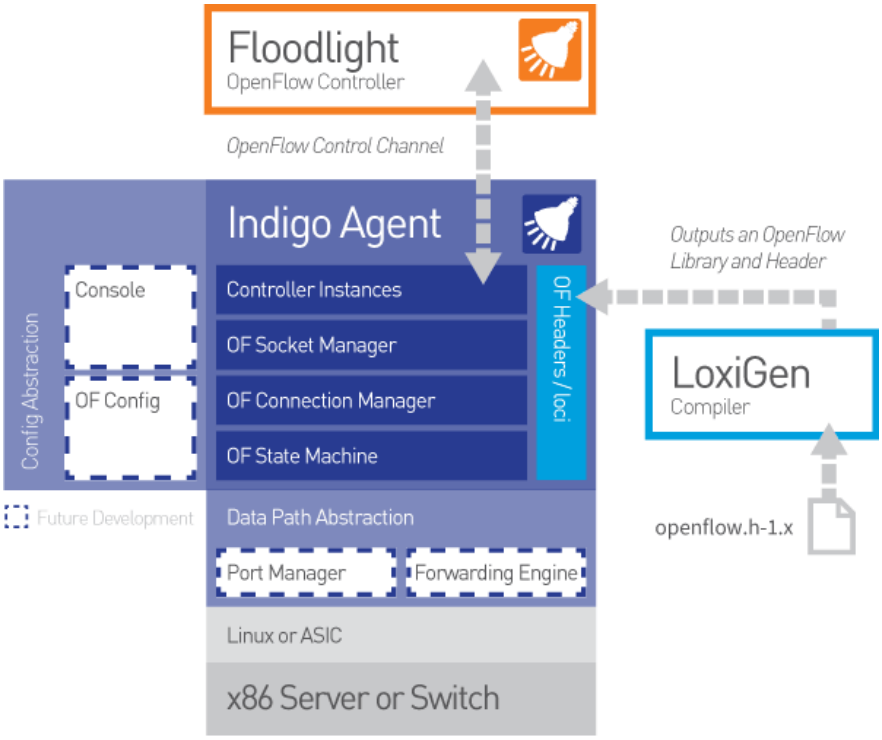
4.2 Indigo Virtual Switch

Framework *Indigo* byl vyvinut v rámci projektu *Floodlight*. Jde o implementaci OpenFlow schopnou využít přednosti hardwarových přepínačů využitím specializovaných obvodů ASIC, která se stala základem pro platformu Switch Light společnosti Big Switch Networks. *Indigo* sestává ze dvou hlavních komponent:

- *Indigo Agent* – představuje ústřední knihovny a obsahuje abstrakční vrstvu nad hardware a abstrakční vrstvu konfigurace pro podporu OpenFlow v tzv. hybridním režimu,
- *LoxiGen* – kompilátor knihoven v různých programovacích jazycích pro práci s protokolem OpenFlow.

LoxiGen v současnosti podporuje pouze jazyk C. Varianty pro jazyky Java a Python jsou stále ve vývoji. K sestavení kompletního OpenFlow přepínače je nutné komponentu *Indigo Agent* doplnit moduly *Port Manager* a *Forwarding*, které jsou specifické pro platformu. Strukturu frameworku včetně napojení na kontrolér Floodlight, hardware přepínače a obsažených knihoven obsahuje Obrázek 10. (Azodolmolky, 2013; Project Floodlight, 2017)

Komponenty *Indigo Agent* a *LoxiGen* jsou licencovány pod licencí Eclipse Public License verze 1 (EPLv1). Pro komerční využití pro výrobce, kteří ve svých produktech využívají Indigo, ale nezveřejňují zdrojový kód pod stejnou licencí, nabízí společnost Big Switch Networks komerční licenci. Obě komponenty jsou k dispozici na webu GitHub.[4] (Project Floodlight, 2017)



Obrázek 10 – Architektura frameworku Indigo

Zdroj: (Project Floodlight, 2017)

Indigo Virtual Switch (IVS) je open-source virtuální přepínač pro Linux, postavený na frameworku Indigo a kompatibilní s hypervisorem KVM. Pro přepínání paketů je využíván

[4] Zdrojový kód komponenty Indigo Agent je společně s dokumentací dostupný na: https://github.com/floodlight/indigo. Stejně tak komponenta LogiGen je dostupná zde: https://github.com/floodlight/loxigen.

modul jádra Linuxu, již zmiňovaný Open vSwitch. Softwarový přepínač IVS využívá pro práci s OpenFlow kompilátor *LoxiGen* (konkrétně variantu pro jazyk C – *Loci*).

Projekt IVS je stejně jako framework Indigo distribuovaný pod licencí Eclipse Public License verze 1, dostupný na webu GitHub[5] a spravovaný komunitou vývojářů ze společnosti Big Switch Networks. Ta projekt využívá pod komerčním názvem Switch Light for Linux. (Project Floodlight, 2017)

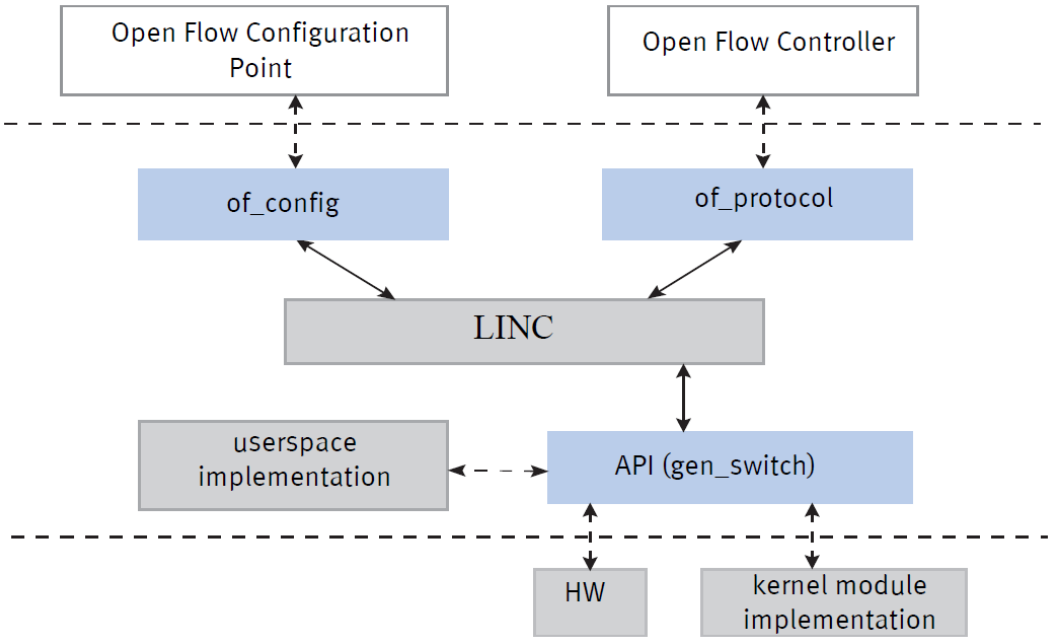
4.3 LINC

Softwarový open-source přepínač LINC (*Link Is Not Closed*) je napsán v programovacím jazyce Erlang. Je tedy spouštěn v běhovém prostředí Erlang, jako tzv. *Erlang node*, v uživatelském režimu Linuxu i na dalších operačních systémech. Takový přístup sice není nejefektivnější, ale přidává na flexibilitě a umožňuje rychlý vývoj a testování nových funkcí OpenFlow. Prostředí Erlang také poskytuje velmi dobré možnosti manipulace s bity, což je vhodné právě pro práci s protokoly a nízkoúrovňovou komunikaci.

Projekt je vedený komunitou FlowForwarding.Org, která jej distribuuje pod licencí Apache 2.0. Přepínač LINC byl vyvinut jako referenční implementace s plnou podporou funkcí specifikace

[5] Projekt je dostupný na adrese https://github.com/floodlight/ivs.

OpenFlow verze 1.2 a 1.3.x a protokolu OF-CONFIG 1.1.1. Architektura přepínače je znázorněna na následujícím obrázku. (Azodolmolky, 2013; FlowForwarding.Org, 2014)



Obrázek 11 – Architektura přepínače LINC

Zdroj: (FlowForwarding.Org, 2014)

Protokol OpenFlow je implementován v knihovně *of_protocol*. Zde jsou definovány příslušné struktury, datové typy, kódování a validace zpráv OpenFlow. Hlavní funkcionalita přepínače se nachází v knihovně LINC, která vykonává příkazy přijaté z části *of_config* a stará se o běh logických přepínačů. Knihovna LINC také komunikuje s back-end komponentami ve formě API (*gen_switch*). Tyto komponenty pak mají na starost např. porty nebo tabulky OpenFlow. (FlowForwarding.Org, 2014)

Zdrojové kódy přepínače jsou k dispozici na serveru GitHub[6]. K dispozici je také nová, rychlejší verze LINCX, která běží v rámci separátní domény hypervisoru Xen (LING). (Azodolmolky, 2013)

4.4 Lagopus

Japonskou společností NTT (*Nippon Telegraph and Telephone*) byl v červenci 2014 vydán open-source softwarový přepínač Lagopus. Lagopus není zaměřen pouze na datová centra, nýbrž také na oblast WAN spravovanou telekomunikačními společnostmi, kde je vyžadována

[6] Zdrojové kódy přepínače lze stáhnout zde: https://github.com/FlowForwarding/LINC-Switch.

vysoká propustnost. Pro tyto účely podporuje Lagopus přenosové rychlosti 10 Gbit/s a přes milion záznamů v tabulce pravidel. S vývojem přenosových technologií a cen je projekt cílen na ještě vyšší rychlosti. Pro zrychlení zpracovávání paketů je využívána sada knihoven a ovladačů DPDK (*Data Plane Development Kit*).

Z funkčního pohledu podporuje Lagopus specifikaci OpenFlow verze 1.3.4 včetně funkcí týkajících se sítí WAN, mezi které patří protokol MPLS, standard PBB a řízení šířky pásma. Podpora protokolů pro správu softwarového přepínače OF-CONFIG a OVSDB je stále ve vývoji. Přístup k přepínači je stále možný přes konzoli (CLI) a protokol SNMP.

Přepínač Lagopus je distribuován pod licencí Apache 2.0.[7] (Sakaida a další, 2016)

4.5 OF13SoftSwitch

OF13SoftSwitch je softwarový přepínač běžící v uživatelském režimu. Jde o projekt založený na implementaci *Ericsson TrafficLab 1.1 SoftSwitch*. Projekt je podporovaný brazilským *Ericsson Innovation Center* a udržovaný společností CPqD ve spolupráci s *Ericsson Research*.

Jelikož přepínač vychází z referenční implementace *Stanford 1.0.0 reference switch*, je přepínač kvůli některým společným částem kódu distribuován pod licencí BSD. Přepínač poskytuje kompatibilitu s protokolem OpenFlow verze 1.3 a obsahuje následující části:

- *datapath* – implementace přepínače,
- *ofprotocol* – zajišťuje bezpečné připojení ke kontroléru,
- *oflib* – knihovny pro práci s protokolem OpenFlow,
- *dpctl* – nástroj pro konfiguraci OpenFlow přepínače.[8] (Azodolmolky, 2013)

4.6 XORPlus

Další z dostupných softwarových přepínačů je XORPlus, který cílí především na hardwarové přepínače se specializovanými obvody ASIC. Podporuje nejznámější protokoly 2. a 3. vrstvy ISO/OSI modelu, z nichž lze zmínit STP, LACP, VLAN (802.1q), LLDP nebo ACL, resp. OSPF, BGP, IGMP nebo IPv6, které patří mezi protokoly síťové vrstvy. Protokol OpenFlow je

[7] Přepínač lze stáhnout na adrese https://github.com/lagopus/lagopus.
[8] Zdrojové kódy společně s pokyny pro instalaci jsou k dispozici webu https://github.com/CPqD/ofsoftswitch13.

podporován skrze Open vSwitch verze 1.1, který podporuje pouze OpenFlow ve verzi 1.0.[9] (Azodolmolky, 2013)

Poslední verze projektu XORPlus pochází z roku 2009. Společnost Pica8, od které tento projekt pochází, nyní vyvíjí vlastní komerční software *PicOS* založený na Linuxu. (Pica8, Inc., 2014)

4.7 Shrnutí softwarových přepínačů

Následující tabulka poskytuje stručné shrnutí zmíněných softwarových přepínačů. Kromě jednoznačných vlastností, jako jsou podporované verze protokolu OpenFlow nebo podporovaná platforma, zde byla zahrnuta i informace o vhodnosti (připravenosti) softwarového přepínače do produkčního prostředí, jak to uvádí oficiální stránky jednotlivých projektů.

Tabulka 3 – Shrnutí popsaných softwarových přepínačů

Název	***Vhodný do produkčního prostředí***	***SW platforma***	***Programovací jazyk***	***Podporované verze OpenFlow***
Open vSwitch	Ano	Linux, FreeBSD, NetBSD a Windows	C	1.1–1.5
Indigo Virtual Switch	Ano	Linux	C	1.0–1.5.1
LINC	Ne[10]	Erlang	Erlang	1.2 a 1.3
Lagopus	Ne	Linux, FreeBSD a NetBSD	C	1.3.4
OF13SoftSwitch	Ne	Linux	C	1.3
XORPlus	Ne[11]	Linux	C++	1.0

Zdroj: zpracováno dle (Azodolmolky, 2013; Project Floodlight, 2017; Nippon Telegraph and Telephone Corporation, 2016; Sakaida a další, 2016)

[9] Zdrojové kódy XORPlus jsou dostupné na webu https://sourceforge.net/projects/xorplus.
[10] Z důvodu neefektivity využitím běhového prostředí Erlang je vhodný především pro testování.
[11] Poslední verze pochází z roku 2009. Novější systém je distribuován komerčně pod názvem picOS. (Pica8, Inc., 2014)

5 RASPBERRY PI

5.1 Vznik počítače Raspberry Pi

Práci na miniaturním a levném počítači začal Eben Upton se svými kolegy přibližně v roce 2006 v době své působnosti na univerzitě v Cambridge. Původně měl počítač Raspberry Pi sloužit středoškolákům, kteří by ho dostali zdarma na dni otevřených dveří na univerzitě v Cambridge s tím, že by po několika měsících byli na přijímacím pohovoru dotázáni, co s tímto počítačem udělali. Do vzdělávacího programu by pak byli pozváni ti, kteří s počítačem provedli něco zajímavého. Projekt Raspberry Pi nebyl tedy původně příliš ambiciózní a měl sloužit spíše lokální potřebě. Po několika schůzkách a diskuzích o stavu informatického vzdělávání založil Eben Upton se svými kolegy nadaci Raspberry Pi Foundation.

Již od počátku vývoje bylo jasné, že používání počítače Raspberry Pi by mělo děti lákat, i když se doposud příliš nezajímaly o programování. Také by měl nový počítač, podobně jako počítač *ZX Spectrum* z 80. let, být schopen se připojit k televizoru a ušetřit tak uživatelům výdaj za monitor. Proto bylo Raspberry Pi vybaveno mimo port HDMI také kompozitním výstupem, kterým je schopné se připojit ke starším analogovým televizím. Jako paměťové zařízení byly z počátku zvoleny relativně levné a dostupné karty SD místo karet microSD, které jsou kvůli své drobné velikosti poměrně náchylné ke ztrátě. Napájení počítače je řešeno kabelem micro USB, který slouží také jako standardní kabel pro nabíjení mobilních telefonů v rámci EU.

Po asi pěti letech byl vyvinut poměrně atraktivní prototyp základní desky. Ačkoliv nevyšla spolupráce s oddělením výzkumu a vývoje společnosti BBC, byla domluvena spolupráce s technickým publicistou Rory Cellan-Jonesem, který pomohl s propagací počítače Raspberry Pi umístěním videa na svůj blog, které také nemalým způsobem přispělo k rozšíření povědomí o plánovaném zařízení.

Jelikož neměl malý prototyp dost místa na všechny potřebné porty a také byla slíbena cena 25 dolarů (Raspberry Pi měl totiž stát přibližně jako učebnice), byl stráven další rok s návrhem základní desky. Zároveň měl být malý počítač co nejvíce užitečný i bez rozličných periferií.

Koncem roku 2011, pár měsíců před plánovaným vydáním, začalo být zřejmé, že poptávka bude mnohem vyšší, než bylo předpokládáno. Nadace měla prostředky k produkci cca deseti tisíc počítačů během jednoho měsíce. Díky mimořádnému úspěchu při budování online

komunity však bylo v seznamu 100 000 zájemců o objednání nového malého počítače v den vydání. Pro vyřízení takového počtu zájemců byly proto kontaktovány firmy element14 a RS Components, které se staraly o výrobu a distribuci počítačů Raspberry Pi.

Jak vlastně vznikl název Raspberry Pi? Části názvu pochází od různých členů dozorčí rady nadace Raspbery Pi Foundation. První slovo z názvu – Raspberry – znamená v překladu malina a bylo vybráno proto, že značky pojmenované po ovoci mají v IT firmách již určitou tradici. Nejznámější z nich je Apple se svým znakem nakousnutého jablka. Druhá část – Pi – je pak zkomolenina názvu programovacího jazyka Python. Původně se totiž tvůrci domnívali, že půjde o jediný jazyk dostupný na mnohem slabší platformě, než jaká byla nakonec do Raspberry Pi zařazena. (Upton a další, 2013)

5.2 Přehled modelů

Od vydání první verze počítače Raspberry Pi přišla na trh celá řada modelů a revizí tohoto malého počítače. První verze Raspberry Pi přišla ve dvou modelech:

- Model A (vydaný později s redukovanou hardwarovou výbavou),
- Model B (představený jako první, s plnou hardwarovou výbavou).

Hlavní rozdíly mezi těmito modely spočívají ve velikosti instalované paměti RAM, která u modelu A činí 256 MB, zatímco model B (revize 2.0) disponuje dvojnásobnou pamětí. Tato paměť je sdílena s grafickým procesorem, což je také třeba brát v úvahu. Model A též postrádá ethernetový port. Připojení k síti lze realizovat pomocí adaptéru do USB. Model A však disponuje pouze jedním portem USB, proto bude pravděpodobně nutné použití rozbočovače. Spotřeba méně vybaveného modelu je však méně než poloviční. (Gay, 2014)

Tabulka 4 – Srovnání modelů Raspberry Pi

Model	**B**	**A+**	**B+**	**Zero**	**2 B**	**3 B**
Cena	$25	$20	$25	Od $5	$35	$35
Datum vydání	29. 2. 2012 (1. revize)	10. 11. 2014	14. 7. 2014	29. 11. 2015	2. 2. 2015	29. 2. 2016
Jádro procesoru	ARM1176J ZF-S	ARM1176J ZF-S	ARM1176J ZF-S	ARM1176J ZF-S	ARM Cortex-A7	ARM Cortex-A53
Frekvence CPU	700 MHz	700 MHz	700 MHz	700 MHz	900 MHz	1,2 GHz
Počet jader CPU	1	1	1	1	4	4
Paměť RAM	512 MB (1. revize 256 MB)	256 MB	512 MB	512 MB	1 GB	1 GB
Počet portů USB	2	1	4	1 (micro)	4	4

Model	B	A+	B+	Zero	2 B	3 B
Úložiště	SD	Micro-SD	Micro-SD	Micro-SD	Micro-SD	Micro-SD
Ethernetový port	Ano	Ne	Ano	Ne	Ano	Ano
Hlavní charakteristika	Původní verze	Cena	Ethernetový port	Cena, velikost	Výkon, Ethernet	Výkon, Wi-fi, Bluetooth

Zdroj: zpracováno dle (Molloy, 2016; Monk, 2016; FatDog.NL, 2016)

Tabulka 4 poskytuje přehled parametrů různých modelů počítače Raspberry Pi. Model A byl nahrazen modelem A+, proto zde nebyla tato starší verze zahrnuta. (Raspberry Pi Foundation, 2017)

Všechny modely disponují pro účely zobrazování portem HDMI a kompozitním výstupem v různých variantách. Model Zero se v tomto liší použitím menší verze portu HDMI – mini-HDMI. Zvukový výstup je realizován digitálně skrze HDMI nebo analogově 3,5mm Jackem, kterým nedisponuje model Zero. Pro účely připojení modulu fotoaparátu nebo displeje obsahuje základní deska počítačů Raspberry Pi porty CSI, respektive DSI. (Molloy, 2016)

Jedna z věcí, která odlišuje Raspberry Pi od klasických stolních počítačů a notebooků, je možnost přímé komunikace s elektronikou pomocí vývodů (pinů) GPIO (*General Purpose Input/Output*), které lze programovat pomocí softwaru. Výstupy slouží ke komunikaci s hardware na nízké úrovni přijímáním nebo odesíláním elektrických signálů z těchto vývodů. Lze tedy s nimi např. rozsvítit diody LED, ovládat elektrický motor apod. Původní model Raspberry Pi B disponuje celkem 26 piny GPIO, zatímco ostatní modely mají těchto pinů 40. Piny jsou připojeny bez ochrany přímo na čip, proto je nutné je chránit od příliš vysokého napětí a statické elektřiny. Programovat aplikace pro rozhraní GPIO lze v jazycích C, Python, Java a dalších. (Molloy, 2016; Horáček, 2012)

5.3 Dostupné linuxové distribuce

Ačkoliv existuje mnoho různých distribucí Linuxu pro vestavěné systémy, všechny používají linuxové jádro. Čím se distribuce liší, jsou různé nástroje a konfigurace, které poskytují uživateli další funkce nebo usnadňují použití. Mezi hlavní linuxové distribuce používané na počítači Raspberry Pi patří *Raspbian*, *Ubuntu*, *OpenELEC* a *Arch Linux*.

Raspbian je verze distribuce Debian, vydaná speciálně pro Raspberry Pi. Debian je distribuce Linuxu, která klade důraz na vývoj s otevřeným zdrojovým kódem. Na vývoji Debianu se nepodílí žádné komerční organizace. Distribuce *Raspbian* rozšiřuje Debian o nástroje

specifické pro Raspberry Pi a softwarové balíky, jako je Java, Mathematica nebo výukový programovací jazyk Scratch. V současnosti je k dispozici verze *Raspbian Jessie* založená na distribuci Debian Jessie (verze 8).

Distribuce *Ubuntu* je úzce spjatá se Debianem, neboť je na něm založená. *Ubuntu* je jedna z nejoblíbenějších linuxových distribucí převážně proto, že cílí na přístupnost Linuxu novým uživatelům. Snadno se instaluje a existují také distribuce pro Raspberry Pi. Hlavní výhodou *Ubuntu* je kvalita uživatelského prostředí, může se tedy jednat o dobrou volbu pro zařízení s všestranným použitím.

Pro účely domácího multimediálního centra, např. s použitím software *Kodi*, je zaměřena distribuce *OpenELEC*. Pro zvýšení výkonu a spolehlivosti používají distribuce *OpenELEC* typicky souborový systém pouze pro čtení. Takovéto optimalizace však mohou ztěžovat vývoj v rámci této distribuce.

Arch Linux je distribuce cílená na jednoduchost a nenáročnost systému. Uživatelům poskytuje plnou kontrolu nad konfigurací systému, a míří tedy spíše ke zkušenějším uživatelům. V porovnání s jinými distribucemi je však méně vhodná pro nové uživatele na platformě Raspberry Pi. (Molloy, 2016)

Mezi populární distribuce lze také zařadit distribuci *Pidora*, která vychází z projektu Fedora společnosti Red Hat, nebo *RaspBMC*, specializovanou na multimédia podobně jako distribuce OpenELEC. (Upton a další, 2013)

Nadace Raspberry Pi Foundation vyvinula také instalátor Linuxu pro nové uživatele nazvaný NOOBS (*New Out of the Box Software*), který obsahuje distribuci *Raspbian* a poskytuje jednoduchost stažení a instalace jako u jiných linuxových distribucí. Mnoho sestav s Raspberry Pi obsahuje kartu SD s nahraným instalátorem NOOBS. (Molloy, 2016; Monk, 2016)

Mimo linuxové distribuce existují další operační systémy pro platformu Raspberry Pi, například *Windows 10 IoT* nebo *RISC OS*, poprvé vydaný v roce 1987 společností Acorn, která vyvinula procesor ARM. (Molloy, 2016; RISC OS Open Limited, 2011)

5.4 Možná využití

Jak již může vyplývat z předchozího textu této kapitoly, malý počítač Raspberry Pi lze využít pro mnoho různých účelů, od přímého ovládání jiné elektroniky přes součást

Internetu věcí (IoT) nebo prodejního automatu po domácí multimediální centrum nebo webový server. Raspberry Pi je díky svému procesoru, který byl navrhnut pro multimediální použití a disponuje hardwarovou implementací kodeku H.264, schopné přehrávat video o vysokém rozlišení, což je obzvláště užitečné v rámci zmíněného multimediálního centra. Podporou 3D grafiky má Raspberry Pi potenciál stát se herní platformou.

Hlavní výhodou Raspberry Pi a dalších vestavěných zařízení využívajících operační systém Linux oproti tradičnějším zařízením, jako je např. Arduino, je právě široká škála aplikací, které jsou na OS Linux dostupné. Například v případě výstavby domácího systému poskytujícího automatizační nebo senzorickou funkci lze nainstalovat webový server a použít libovolný serverový programovací jazyk a tak zpřístupnit data na Internetu nebo místní síti. (Molloy, 2016)

Komunita okolo projektu Raspberry Pi sama přichází s nepřeberným množstvím nápadů s použitím Raspberry Pi, například projekt australské školy na sledování meteorů nebo robot ovládaný zařízením snímajícím mozkové vlny a další. Na propojených počítačích Raspberry Pi lze provozovat *cluster*, vytvořit superpočítač nebo si vyzkoušet poskytování cloudových služeb. Všechny tyto možnosti použití mohou sloužit k výukovým účelům, pro které bylo Raspberry Pi určeno. (Upton a další, 2013; Vaughan-Nichols, 2013; Čevela, 2015)

Různé modely počítače Raspberry Pi se mohou lišit také oblastmi svého využití. Například pro účely všeobecného použití se nejlépe hodí poslední verze 3. Se svým čtyřjádrovým procesorem a gigabajtem paměti poskytuje nejlepší výkon ze všech modelů. Toho je využito také v praktické části této práce. Aplikace, které zprostředkovávají přístup jiné elektroniky do počítačové sítě, mohou být realizovány verzemi 3, 2 a B+, které disponují ethernetovým portem. Ušetřit několik dolarů nebo centimetrů lze v některých projektech nevyužívajících pro daný účel komponenty z vyšších modelů zahrnutím variant A+ nebo Zero. (Molloy, 2016)

6 INSTALACE SOFTWAROVÝCH PŘEPÍNAČŮ

6.1 Open vSwitch

Pro implementaci OpenFlow přepínače na počítači Raspberry Pi byl vybrán Open vSwitch. V této kapitole je představen postup instalace na dvou linuxových distribucích, které disponují podporou přímo pro Raspberry Pi. Jde o oficiální distribuci nadace Raspberry Pi Foundation – *Raspbian* a distribuci *Ubuntu* s grafickým prostředím *MATE*.

Nejnovější verze distribuce *Raspbian* je dostupná přímo z oficiálních stránek[12] Raspberry Pi a pochází z března 2017. Jde o distribuci založenou na Debian Jessie s jádrem Linuxu verze 4.4. Druhou linuxovou distribucí, která je dostupná pro počítač Raspberry Pi, je *Ubuntu MATE*. Variantu pro počítače Raspberry Pi lze stáhnout z oficiálních stránek[13] distribuce ve verzi 16.04.2 LTS. Společnou vlastností těchto systémů je mj. jejich velikost přesahující 4 GB, proto je nutné počítač Raspberry Pi vybavit kartou SD o velikosti minimálně 8 GB.

Instalace přepínače je zde popsána pro distribuci *Raspbian*. Jelikož jsou oba systémy založeny na distribuci Debian, instalace na nich probíhá do jisté míry podobně. Na konci kapitoly jsou pak popsány rozdíly v postupu instalace na prostředí Ubuntu MATE.

Oba systémy jsou na webu dostupné jako obraz disku (soubor img), který se nahraje na kartu SD pomocí speciálního software. Nejde tedy o pouhé zkopírování souboru na kartu, ale o nahrazení celého souborového systému. Pro tyto účely byl využit nástroj *Win32DiskImager*[14]. Jde o jednoduchý program pro Windows, který umožňuje zápis obrazů na médium (*Write*) nebo jeho získávání (*Read*). Systém je potom schopen z takto připravené karty provést boot operačního systému.

6.1.1 Příprava systému

Po připojení systému k Internetu, případně dalším nastavení, bude provedena aktualizace seznamu dostupných balíčků, jejich verzí a závislostí.

```
sudo apt-get update
```

[12] Distribuce Raspbian je k dispozici ke stažení na oficiálních stránkách https://www.raspberrypi.org/downloads/raspbian/.
[13] Distribuci lze stáhnout na adrese https://ubuntu-mate.org/download/.
[14] Program lze stáhnout na adrese https://sourceforge.net/projects/win32diskimager/.

V rámci aktualizace systému bude proveden také upgrade kernelu, nicméně pro instalaci stabilní verze Open vSwitch 2.5.2 bude nutné provést downgrade jádra na verzi nižší než 4.3. Pro zabránění zbytečné aktualizaci kernelu před jeho downgrade je potřeba označit balíčky, které se starají o jeho aktualizaci, aby se neaktualizovaly.

```
sudo apt-mark hold raspberrypi-kernel raspberrypi-bootloader
```

Následuje aktualizace systému. Zatímco příkaz `apt-get upgrade` provede pouze aktualizaci nainstalovaných balíčků na nejnovější verzi a za žádných okolností nenainstaluje balíčky nové či stávající nesmaže, příkaz `apt-get dist-upgrade` se dokáže vypořádat také se změnami závislostí nových verzí balíčků a v případě potřeby je doinstalovat. Tato operace může trvat několik desítek minut.

```
sudo apt-get dist-upgrade
```

V případě instalace Open vSwitch verze 2.5.2 s dlouhodobou podporou je nutné provést downgrade kernelu na podporovanou verzi. Pro instalaci specifické verze kernelu bude použit nástroj pro aktualizaci firmware `rpi-update`[15]. Ten je již součástí distribuce Raspbian.

Spuštění příkazu `rpi-update` bez parametrů aktualizuje firmware na nejnovější verzi. Pro instalaci specifické verze kernelu je nutno předat nástroji jako parametr hash příslušného commitu z repozitáře *rpi-firmware*[16]. Tento repozitář obsahuje některé soubory zrcadlené z oficiálního repozitáře. Pro podporu zmíněné verze Open vSwitch byl vyhledán commit obsahující nejvyšší verzi kernelu nižší než 4.3. Tomu odpovídá commit s názvem „kernel: bump to 4.1.21.“ V detailech commitu lze nalézt jeho hash, který je následně předán příkazu.

```
sudo rpi-update dea25fa62132365c11087e51e416df656db28bf3
```

Pro provedení všech změn v systému bude vyžadován restart.

Po nainstalování správné verze kernelu budou potřeba pro sestavení modulu kernelu hlavičkové soubory jádra, pro který bude modul kompilován. Jelikož hlavičkové soubory dostupné z oficiálních balíčků mohou být zastaralé nebo nemusí přesně odpovídat verzi jádra, která je

[15] Zdrojový kód tohoto nástroje společně s pokyny k jeho použití lze nalézt na webu https://github.com/Hexxeh/rpi-update.
[16] Repozitář se nachází na adrese: https://github.com/Hexxeh/rpi-firmware.

aktuálně nainstalována, bude ke stažení těchto souborů použit nástroj `rpi-source`. Zdrojové kódy lze nalézt opět na webu GitHub[17]. Instalace bude provedena příkazem:

```
sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source -O /usr/bin/rpi-source && sudo chmod +x /usr/bin/rpi-source && /usr/bin/rpi-source -q --tag-update
```

Předchozí příkaz lze rozdělit do tří částí. První část stáhne soubor se zdrojovým kódem do složky `/usr/bin` do souboru `rpi-source`. Tomuto souboru jsou poté přidána práva na spuštění, čímž se z něho stane spustitelný soubor. Nakonec program označí pro mechanizmus aktualizací současnou verzi programu jako aktuální. Přepínač `-q` potlačí výstup o této operaci.

Spuštění programu bez parametrů provede stažení zdrojových kódů jádra operačního systému. Program není nutné spouštět pod právy uživatele `root`, neboť sám interně používá příkaz `sudo`.

```
rpi-source
```

Pokud by program vypsal chybu verze `gcc`, lze postupovat podle kapitoly 6.1.11. Program stáhne zdrojové kódy právě nainstalované verze kernelu Linuxu do domovského adresáře uživatele do složky s názvem `linux-<hash-commitu-z-oficiálního-repozitáře>`[18] a následně vytvoří na tuto složku odkazy `build` a `source` do složky `/lib/modules/<verze-kernelu>`.

6.1.2 Stažení zdrojových kódů Open vSwitch

Pro stažení nejnovější verze přepínače Open vSwitch lze využít repozitář na serveru GitHub.[19] Číslo aktuální verze je 2.7.90. K jejímu stažení lze využít příkaz `git clone`. Tímto bude vytvořena v aktuálním adresáři složka `ovs` se zdrojovými kódy přepínače. Zdrojové kódy lze také z repozitáře stáhnout ve formátu zip.

```
git clone https://github.com/openvswitch/ovs.git
```

Open vSwitch lze také stáhnout z oficiálních stránek přepínače ve formátu archivu tar.[20] Nejnovější vydání z aktuální řady je ve verzi 2.7.0. K dispozici je také nejnovější verze s dlouhodobou podporou (LTS) 2.5.2.

[17] Zdrojový kód nástroje rpi-source s dalšími informacemi lze nalézt na adrese: https://github.com/notro/rpi-source.
[18] Repozitář se sestaveními Linuxu pro Raspberry Pi lze nalézt na webu GitHub zde: https://github.com/raspberrypi/linux.
[19] Repozitář se nachází na adrese https://github.com/openvswitch/ovs.
[20] Formát je též nazýván tarball.

Jediná verze, která se ukázala jako funkční na obou zmiňovaných distribucích, byla verze dlouhodobou podporou 2.5.2. Po instalaci novějších verzí modulů kernelu a programů pro část user-space systém po spuštění zamrzal. Částečným řešením je spouštění přepínače ručně po spuštění, avšak také ukončení systému se spuštěnými procesy Open vSwitch neprobíhalo úspěšně. V této práci je tedy dále pracováno s verzí 2.5.2. Postup instalace vyšších verzí je však prakticky totožný.

Stažení stabilní verze z oficiálních stránek a její rozbalení do aktuálního adresáře lze provést následující sérií příkazů.

```
wget http://openvswitch.org/releases/openvswitch-2.5.2.tar.gz
tar -xf openvswitch-2.5.2.tar.gz
```

Instalace přepínače bude probíhat ze staženého adresáře zdrojovými kódy.

```
cd openvswitch-2.5.2/
```

6.1.3 Instalace potřebných závislostí

Zjištění chybějících balíčků, které jsou potřebné pro sestavení přepínače, provede následující příkaz.

```
dpkg-checkbuilddeps
```

Ten na základě informací obsažených v souboru `debian/control` a aktuálně nainstalovaných balíčků zobrazí případné chybějící závislosti, přičemž skončí s nenulovým kódem. Pokud jsou všechny potřebné balíčky nainstalované, příkaz nic nevypíše. Výpis tohoto příkazu může vypadat následovně:

```
dpkg-checkbuilddeps: Unmet build dependencies: graphviz autoconf (>= 2.64)
automake (>= 1.10) | automake1.10 debhelper (>= 8) dh-autoreconf libssl-dev
libtool python-all (>= 2.7) python-qt4 python-twisted-conch python-
zopeinterface
```

Na základě této informace lze spustit instalaci chybějících balíčků.

```
sudo apt-get install graphviz autoconf automake debhelper dh-autoreconf
libssl-dev libtool python-all python-qt4 python-twisted-conch python-
zopeinterface
```

Volitelným závislým balíčkem je `libcap-ng`. Ten umožňuje démonům OVS (tj. procesům běžícím na pozadí), běh pod uživatelem, který není `root` a má tak omezená oprávnění. Pokud je balíček `libcap-ng` nainstalován, je Open vSwitch automaticky sestaven s podporou pro něj.

V opačném případě dojde při konfiguraci sestavení k výpisu varování. Balíček je v repozitáři dostupný pod názvem `libcap-ng-dev`.

```
sudo apt-get install libcap-ng-dev
```

6.1.4 Instalace pomocí nástroje make

Jednou z možností instalace Open vSwitch ze zdrojových kódů je pomocí nástroje `make`. Pokud byly zdrojové kódy staženy z repozitáře, bude nutné nejdříve spustit soubor `boot.sh`. Ten spouští příkaz `autoreconf` a sestavuje konfigurační skript. Pokud byly zdrojové kódy rozbaleny ze staženého archivu tar, není nutné tento krok absolvovat.

```
./boot.sh
```

Konfigurační skript může být spuštěn s různými parametry i bez parametrů. V parametrech lze například zvolit kompilátor, příp. mu předávat parametry nebo povolit dynamicky linkované knihovny. V případě, že bude sestavován také modul kernelu, což je v této práci prováděno, bude konfigurační skript spuštěn s přepínačem `--with-linux`, který přijímá cestu k adresáři se zdrojovými kódy jádra, které byly staženy pomocí nástroje `rpi-source`. V rámci konfigurace je mimo jiné také kontrolována verze kernelu.

```
./configure --with-linux=/lib/modules/$(uname -r)/build
```

Část příkazu `$(uname -r)` představuje subshell. Jde o vnořený příkaz, který do nadřazeného příkazu vloží svůj výstup. Zde vložený příkaz obsahuje `uname -r`, který vrací aktuální verzi kernelu.

Konfigurační skript vytvořil soubor `Makefile`, který má v sobě uchované potřebné instrukce pro kompilaci programu. Build přepínače lze spustit prostým příkazem `make` ve stejném adresáři jako onen soubor.

```
make
```

Proces kompilace a sestavování programu trval na Raspberry Pi 3 modelu B cca 15 minut. Po dokončení konfigurace a sestavení lze provést testy. Open vSwitch obsahuje sadu testů, které lze spustit příkazem `make check`. Bez dalších parametrů budou testy prováděny sériově, jeden po druhém. Zrychlení provádění využitím paralelismu na vícejádrových procesorech lze zapnout přidáním parametru `TESTSUITEFLAGS=-j#`, kde `#` představuje počet vláken spuštěných pro testy, maximálně 8.

```
make check TESTSUITEFLAGS=-j8
```

Využitím osmi vláken bylo zpracování testů zrychleno přibližně trojnásobně. I přes běh na více vláknech trvá provedení všech cca 2300 testů déle než samotné sestavení programu. U nejnovějších verzí je možné, že některé testy budou neúspěšné. Po sestavení Open vSwitch verze 2.7.0 na aktuální distribuci Raspbian vyšlo ze všech testů pět neúspěšných. Verze s dlouhodobou podporou disponovala úspěšným provedením všech testů. Podrobné informace o neúspěšných testech se nachází v adresáři `tests`.

Instalace spustitelných souborů a manuálových stránek do běžícího systému bude provedena příkazem:

```
sudo make install
```

Jelikož byl proveden také build modulů kernelu, jejich instalace bude vykonána následujícím způsobem.

```
sudo make modules_install
```

Jelikož jádro Linuxu již obsahuje modul přepínače Open vSwitch, je nutné vytvořit konfigurační soubor v adresáři `/etc/depmod.d`, který preferuje nově nainstalované moduly kernelu před stávajícími. Pokud adresář neexistuje, je potřeba jej vytvořit.

```
sudo mkdir -p /etc/depmod.d
```

Protože bude následující skript přistupovat ke složce a souboru, ke kterým je potřeba oprávnění `root`, bude provádění následujících příkazů probíhat pod uživatelem `root`.

```
sudo su
```

Nyní bude zapsána potřebná konfigurace do souboru `openvswitch.conf` v adresáři `/etc/depmod.d`. K tomu bude využit cyklus, který projde všechny moduly kernelu z adresáře `datapath/linux` a zapíše jejich názvy do konfiguračního souboru.

```
config_file="/etc/depmod.d/openvswitch.conf"
for module in datapath/linux/*.ko; do
      modname="$(basename ${module})"
      echo "override ${modname%.ko} * extra" >> "$config_file"
      echo "override ${modname%.ko} * weak-updates" >> "$config_file"
      done
depmod -a
```

Z režimu uživatele `root` lze nyní odejít.

```
exit
```

Nakonec je možno modul kernelu načíst do paměti příkazem `modprobe`.

```
sudo modprobe openvswitch
```

6.1.5 Inicializace přepínače

Před spuštěním procesu `ovs-vswitchd` je nutné vytvořit databázi, kterou bude používat databázový server Open vSwitch (`ovsdb-server`). K tomuto účelu bude využit nástroj Open vSwitch database management utility (příkaz `ovsdb-tool`). Příkaz `create` vytvoří novou databázi na základě souboru, který obsahuje OVSDB schema ve formátu JSON. Zde jde o soubor `vswitch.ovsschema` ve složce `vswitchd`. Prvním parametrem příkazu je cesta k souboru nové databáze. Je dobré připomenout, že příkazy jsou stále prováděny z adresáře se zdrojovými kódy Open vSwitch.

```
sudo ovsdb-tool create /usr/local/etc/openvswitch/conf.db \
    vswitchd/vswitch.ovsschema
```

Následujícím krokem je konfigurace databázového serveru. V příkazu jsou specifikovány dva přepínače `--remote`, které definují způsoby připojení k databázi. První z nich způsobí naslouchání na unixovém soketu ze souboru. Druhý přepínač `--remote` načítá způsoby připojení z databáze, konkrétně ze všech řádků určitého sloupce. Argumenty jsou mu předány ve formátu `databáze,tabulka,sloupec`.

Následují tři parametry týkající se zabezpečení připojení. Přepínač `--private-key` specifikuje soubor PEM, který obsahuje privátní klíč používaný pro odchozí spojení databázového serveru. Specifikovaný privátní klíč je ověřen certifikátem, který se nachází v souboru specifikovaném v přepínači `--certificate`. Certifikát musí být podepsán certifikační autoritou, kterou používá také druhá strana připojení. Příchozí spojení jsou verifikována certifikátem ze souboru PEM, který je zadán přepínačem `--bootstrap-ca-cert`. Všechny tyto parametry jsou načítány z databáze.

Poslední tři přepínače se týkají démona serveru a určují vytvoření souboru PID, který obsahuje PID běžícího procesu, spuštění serveru jako proces na pozadí a zapnutí ukládání záznamů o činnosti progamu do souboru.

```
sudo ovsdb-server \
    --remote=punix:/usr/local/var/run/openvswitch/db.sock \
    --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
    --private-key=db:Open_vSwitch,SSL,private_key \
    --certificate=db:Open_vSwitch,SSL,certificate \
    --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
    --pidfile --detach --log-file
```

Před prvním spuštěním databáze, vytvořené nástrojem `ovsdb-tool`, je nutné ji nejprve inicializovat.

```
sudo ovs-vsctl --no-wait init
```

Nakonec bude spuštěn proces Open vSwitch. Parametry mají ekvivalentní význam k přepínačům specifikovaným v konfiguraci databázového serveru.

```
sudo ovs-vswitchd --pidfile --detach --log-file
```

6.1.6 Spuštění Open vSwitch po startu systému

Ačkoliv je Open vSwitch nyní spuštěný, po restartu systému bude nutné provést spuštění znovu. Aby nebylo nutné tyto příkazy pokaždé zadávat ručně, lze vytvořit spustitelný soubor, který potřebné příkazy vykoná při spuštění systému. Pokud by nebylo žádané spouštět Open vSwitch při každém startu sytému, stačí soubor spustit ručně. Soubor je zde pojmenován `ovs-init` a je do něj uložen následující obsah.

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          openvswitch
# Required-Start:    $network $named $remote_fs $syslog
# Required-Stop:     $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start Open vSwitch processes
# Description:       Load openvswitch module to kernel, start ovsdb-server
and ovs-vswitchd
### END INIT INFO

modprobe openvswitch
ovsdb-server \
    --remote=punix:/usr/local/var/run/openvswitch/db.sock \
    --remote=db:Open_vSwitch,Open_vSwitch,manager_options \
    --private-key=db:Open_vSwitch,SSL,private_key \
    --certificate=db:Open_vSwitch,SSL,certificate \
    --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
    --pidfile --detach --log-file
ovs-vswitchd --pidfile --detach --log-file
```

První řádek souboru určuje, který program bude interpretovat zde zadaný skript. Následující úsek ohraničený řádky `### BEGIN INIT INFO` a `### END INIT INFO` je informace LSB (*Linux Standard Base*). Nachází se zde například informace o závislostech, úrovních běhu, při kterých bude skript spouštěn/zastavován, či popis služby. Za popisnou částí následují některé ze zmíněných příkazů, které zajistí spuštění procesů Open vSwitch.

Aby bylo možno soubor spouštět, je nutné mu přidělit práva na spouštění (*execute*).

```
chmod +x ovs-init
```

Soubor je pak potřeba zkopírovat mezi ostatní spouštěcí skripty do složky `/etc/init.d`.

```
sudo cp ovs-init /etc/init.d
```

Podle toho, ve kterých úrovních běhu má služba nastartovat, je nutné vytvořit symbolické odkazy na spouštěcí skript v adresářích `/etc/rc#.d`, kde `#` odpovídá úrovni běhu. Místo ruční tvorby symbolických odkazů lze využít nástroj `update-rc.d`, který odkazy vytvoří sám a vhodně je přejmenuje.

```
sudo update-rc.d ovs-init defaults
```

Tímto je skript připravený ke spuštění po startu systému. Pokud by bylo potřeba spouštění přepínače po startu systému zrušit, stačí vyměnit příkaz `defaults` za `remove`, který odstraní příslušné odkazy na zmíněný skript.

6.1.7 Instalace pomocí balíčků deb

Podstatně jednodušší možnost instalace přepínače Open vSwitch poskytuje sestavení a následná instalace balíčků pro distribuci Debian. Pro sestavení balíčků je nutné mít nainstalované balíčky `build-essential` a `fakeroot`. Oba byly v distribuci Raspbian již přítomny.

Nejrychlejší metodou vytvoření balíčků pro Debian je spuštění kompilace následujícím příkazem.

```
DEB_BUILD_OPTIONS='parallel=8 nocheck' fakeroot debian/rules binary
```

Do proměnné `DEB_BUILD_OPTIONS` byla vložena informace o zpracování programu na osmi vláknech a slovem `nocheck` je zabráněno vykonávání testů po skončení kompilace. Je dobré zde upozornit, že při neúspěšném skončení některých testů nedojde k vytvoření balíčků.

Příkazy pro instalaci balíčků budou prováděny z nadřazeného adresáře složky se zdrojovými kódy přepínače, kam byly balíčky zkompilovány.

```
cd ..
```

Pro instalaci modulu kernelu bude využit nástroj *Dynamic Kernel Module Support* (DKMS), který umožňuje instalaci nových modulů do jádra systému a také jejich rekompilaci v případě změny verze kernelu. Program je potřeba do systému doinstalovat.

```
sudo apt-get install dkms
```

Nyní již lze přikročit k instalaci balíčku obsahujícího modul kernelu.

```
sudo dpkg -i openvswitch-datapath-dkms_2.5.2-1_all.deb
```

Komponenty úrovně user-space jsou obsaženy v balíčcích `openvswitch-common` a `openvswitch-switch`. Druhý balíček je závislý na prvním, proto bude nejprve nainstalován ten první.

```
sudo dpkg -i openvswitch-common_2.5.2-1_armhf.deb
```

Před instalací druhé komponenty je nutné doinstalovat balíček pro generování 128 bitových unikátních identifikátorů `uuid-runtime`.

```
sudo apt-get install uuid-runtime
```

Instalace poslední komponenty nakonfiguruje načtení modulu kernelu po startu systému a provede počáteční konfiguraci přepínače.

```
sudo dpkg -i openvswitch-switch_2.5.2-1_armhf.deb
```

Nyní je přepínač funkční a je také spouštěn při načítání systému.

6.1.8 Instalace z repozitáře Raspbian

Nejjednodušší způsob instalace přepínače Open vSwitch poskytuje bezesporu oficiální repozitář. Nenabízí však možnost instalovat nejnovější verze ze zdrojového kódu. Součástí oficiálního repozitáře rovněž není balíček pro instalaci modulu kernelu. Aktuálně je v repozitáři distribuce Raspbian dostupná verze přepínače 2.3.0. Tu lze nainstalovat jednoduchým příkazem.

```
sudo apt-get install openvswitch-switch
```

Tímto způsobem jsou nainstalovány obě komponenty `openvswitch-common` i `openvswitch-switch` a provedeno potřebné počáteční nastavení.

6.1.9 Validace instalace

Validaci, zda se modul kernelu nahrál správně, lze vykonat příkazem `modinfo`.

```
modinfo openvswitch
```

Výstup příkazu pro verzi modulu, která je již součástí jádra Linuxu, vypadá následovně:

```
filename:       /lib/modules/4.1.21-
v7+/kernel/net/openvswitch/openvswitch.ko
license:        GPL
```

```
description:    Open vSwitch switching datapath
srcversion:     6C6C3247944378D892D13F4
depends:
intree:         Y
vermagic:       4.1.21-v7+ SMP mod_unload modversions ARMv7
```

Instalovaná verze s dlouhodobou podporou již vrací jiný výstup:

```
filename:       /lib/modules/4.1.21-v7+/extra/openvswitch.ko
version:        2.5.2
license:        GPL
description:    Open vSwitch switching datapath
srcversion:     4EDCF09217F397B8EA747DF
depends:        nf_conntrack,ipv6,nf_defrag_ipv6,gre,nf_defrag_ipv4
vermagic:       4.1.21-v7+ SMP mod_unload modversions ARMv7
parm:           udp_port:Destination UDP port (ushort)
```

Načtení modulu kernelu do operační paměti lze ověřit nalezením modulu `openvswitch` ve výpise aktuálně načtených modulů jádra.

```
lsmod
```

Úspěšnost instalace části pro úroveň user-space a spuštění potřebných procesů lze ověřit například vytvořením jednoduchého mostu (*bridge*), který představuje virtuální přepínač, nástrojem pro dotazování a konfiguraci procesu Open vSwitch `ovs-vsctl`. Virtuální přepínač je v příkladu pojmenován `bridge0`.

```
sudo ovs-vsctl add-br bridge0
```

Po vytvoření mostu je možno vypsat přehled obsahu databáze příkazem:

```
sudo ovs-vsctl show
```

Příkaz by měl vypsat informaci o existujícím bridge.

```
4c86058c-e1b4-431e-9b60-250fc63093b1
      Bridge "bridge0"
            Port "bridge0"
                  Interface "bridge0"
                        type: internal
```

6.1.10 Odlišnosti v instalaci na distribuci Ubuntu MATE

Instalace přepínače na této distribuci se liší především v balíčcích, které tato nemá od začátku nainstalované. Pro instalaci bude například nutné doinstalovat balíčky `git` nebo `fakeroot`. Také se zde nenachází balíčky `raspberrypi-kernel` a `raspberrypi-bootloader`. Naopak pro instalaci balíčku `openvswitch-switch` zde nebylo nutné doinstalovat balíček `uuid-runtime`.

Dalším drobným rozdílem je dostupnost vyšší verze přepínače Open vSwitch z repozitáře distribuce, a sice 2.5.0. Největší překážkou, která se na distribuci Raspbian nevyskytla, zde byl nesoulad verzí kompilátoru `gcc` s verzí, kterou byl sestaven kernel systému. Řešení tohoto problému je představeno v následující podkapitole.

6.1.11 Nesoulad verzí gcc

Po aktualizaci systému a stažení programu `rpi-source` se může stát, že po spuštění programu bude uživatel informován o chybě formou výpisu níže.

```
ERROR:
gcc version check failed: could not extract version numbers
Skip this check with --skip-gcc

Help: https://github.com/notro/rpi-source/wiki
```

Ačkoliv program vypsal, že nemůže zkontrolovat čísla verze kompilátoru, kterým byl sestaven přítomný kernel, a verzi kompilátoru `gcc` aktuálně nainstalovaného, problém je v nesouladu těchto verzí, konkrétně prvních dvou čísel verze. Chybová hláška může také obsahovat text: „gcc version check: mismatch between gcc …," který už je více informativní. Problém lze tedy vyřešit buď změnou verze kernelu nebo instalací patřičné verze `gcc`.

Aktuální verzi kernelu společně s verzí kompilátoru, kterou byl kernel sestaven, lze zjistit z výpisu z virtuálního souborového systému `/proc`.

```
cat /proc/version
```

Verze kernelu, která byla snížena kvůli podpoře Open vSwitch 2.5.2, podává následující výpis.

```
Linux version 4.1.21-v7+ (dc4@dc4-XPS13-9333) (gcc version 4.9.3
(crosstool-NG crosstool-ng-1.22.0-88-g8460611) ) #872 SMP Wed Apr 6
17:34:14 BST 2016
```

Verzi nainstalovaného kompilátoru lze zjistit z něho samého přepínačem `--version`.

```
gcc --version
```

Z výpisu je vidět, že se verze opravdu neshodují.

```
gcc (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Bude tedy provedena instalace verze `gcc`, která byla použita k sestavení kernelu. Konkrétní verzi kompilátoru poskytne nástroj `apt-get`. Zjištění dostupných verzí lze provést zadáním

do příkazu název balíčku s pomlčkou a stisknutím tabulátoru. V tomto případě bude stažena verze 4.9.3, která se nachází pod názvem `gcc-4.9`.

```
sudo apt-get install gcc-4.9
```

Protože je v systému stále aktivní novější verze, příkaz `gcc` pracuje s ní. Je tedy nutné přepnout výchozí příkaz na starší verzi. Pro jednoduché přepnutí lze použít nástroj `update-alternatives`, který jako argumenty přepínače `--install` přijímá cestu k symbolickému odkazu a název výchozího příkazu, dále cestu k odkazu na alternativu, která nahradí stávající příkaz, a prioritu, která upřednostní alternativy s vyšším číslem.

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 40
```

Tímto byl problém neshody verzí kompilátorů vyřešen. Ověření lze provést opětovným výpisem verze `gcc` příkazem výše. Při sestavování modulů kernelu je potřeba stejná verze kompilátoru, jaká bylo použita k sestavení kernelu, pro který je modul kompilován. Po provedení potřebných operací lze příkazu `gcc` vrátit původní verzi obdobným způsobem, pouze s vyšší prioritou.

6.2 Indigo Virtual Switch

Další možnou implementací softwarového přepínače je Indigo Virtual Switch. Přepínač pro přepínání paketů využívá modul kernelu Open vSwitch. Je proto nutné mít tento modul nainstalovaný. Instalaci modulu lze provést jedním ze způsobů popsaných v předchozí kapitole.

Instalace byla, stejně jako instalace Open vSwitch, provedena na distribucích Raspbian a Ubuntu MATE. Protože jsou obě distribuce založeny na distribuci Debian, instalace na nich probíhá stejně. Distribuce Ubuntu MATE pouze postrádala program `git`, který lze jednoduše doinstalovat pomocí příkazu `apt-get`.

Instalace přepínače sestává ze stažení zdrojových kódů z webu GitHub a následné kompilace. Stažení zdrojových kódů včetně submodulů bude provedeno v rámci klonování repozitáře.

```
git clone --recurse-submodules https://github.com/floodlight/ivs.git
```

Instalace bude dále prováděna z adresáře se staženým repozitářem.

```
cd ivs
```

Během klonování repozitáře se kvůli zabezpečení a přístupovým právům pravděpodobně nepodaří stáhnout moduly `bigcode` a `indigo`. Pro jejich dodatečné stažení je k dispozici příkaz

`submodule sync`, který synchronizuje URL adresy v konfiguračním souboru `.gitmodules`, a příkaz `submodule update`, který provede stažení chybějících modulů z repozitáře včetně potřebné inicializace konfigurace.

```
git submodule sync
git submodule update --init
```

Po stažení všech zdrojových kódů bude provedeno zjištění chybějících potřebných závislostí podobným způsobem jako před instalací přepínače Open vSwitch.

```
dpkg-checkbuilddeps
```

Na základě výstupu byl sestaven příkaz pro instalaci balíčků. Ze seznamu byl vyřazen balíček `libnl3-dev`, který není v repozitáři distribuce k dispozici.

```
sudo apt-get install debhelper libnl-3-dev libnl-genl-3-dev libnl-route-3-
dev python-tz libcap-dev
```

Dále bude pro úspěšnou kompilaci potřeba nainstalovat balíček `libssl-dev`.

```
sudo apt-get install libssl-dev
```

Dalším krokem je kompilace programu na základě informací v souborsu `Makefile`. Ve výchozím nastavení kompilace jsou varování kompilátoru považována za chyby. V takovém případě by nebyla kompilace úspěšná. Proto je třeba pro změnu tohoto nastavení přiřadit proměnné `GCC_NO_WERROR` jakoukoliv hodnotu. Tím je docíleno nevložení přepínače `-Werror` ke kompilaci modulu `infra`.

```
make GCC_NO_WERROR=1
```

Kompilace souborů v jazyce C trvá přibližně 15 minut. Nakonec lze provést instalaci, čímž se zpřístupní příkazy `ivs` a `ivs-ctl`.

```
sudo make install
```

6.3 Přepínač LINC

Přepínač LINC je spouštěn v běhovém prostředí Erlang. Běhové prostředí lze nainstalovat z balíčku, který je dostupný v oficiálním repozitáři, příkazem:

```
sudo apt-get install erlang
```

Zdrojové kódy běhového prostředí je možné také získat z repozitáře GitHub,[21] avšak aktuálně dostupná verze (R20) není přepínačem LINC podporována. Kompilace přepínače poté neproběhne úspěšně ani po úpravě kontroly verze Erlang v konfiguračních souborech pro kompilaci. Verze R17, dostupná v repozitáři distribuce Raspbian, je podporována bez problémů.

Zdrojové kódy přepínače jsou k dispozici, podobně jako ostatní přepínače zmíněné v této práci, z repozitáře GitHub.

```
git clone https://github.com/FlowForwarding/LINC-Switch.git
```

Instalace bude probíhat z vytvořeného adresáře `LINC-Switch`.

```
cd LINC-Switch/
```

Přepínač vyžaduje instalaci nástroje `bridge-utils` pro tvorbu zařízení typu bridge, balíček `libpcap` pro odchytávání paketů a balíček `uml-utilities`, poskytující virtualizační vrstvu kernelu na úrovni user-space.

```
sudo apt-get install bridge-utils libpcap0.8 libpcap-dev uml-utilities
```

Před samotnou kompilací programu je nutné vyřešit některé problémy. Prvním z nich je nepřítomnost konfiguračního souboru `sys.config` v adresáři `rel/files/`. Konfigurační soubor obsahuje např. parametry pro připojení přepínače ke kontroléru nebo porty, které bude přepínač používat. Tento soubor bude upravován později. Ve stejném adresáři se nachází vzorový konfigurační soubor `sys.config.orig`. Pro samotnou instalaci lze použít tento a vytvořit z něj soubor potřebný pro kompilaci.

```
cp rel/files/sys.config.orig rel/files/sys.config
```

Konfiguraci je také možné vygenerovat pomocí skriptu `config_gen`. Zvolené parametry odpovídají topologii z kapitoly 8. Příkaz vytvoří logický přepínač s číslem 0 a přiřadí mu specifikované porty a parametry připojení ke kontroléru. Konfigurace bude vygenerována do souboru specifikovaného posledním přepínačem příkazu.

```
./scripts/config_gen -s 0 eth1 eth2 eth3 -c tcp:192.168.0.100:6633 -o
rel/files/sys.config
```

[21] Zdrojové kódy běhového prostředí Erlang se nachází na adrese https://github.com/erlang/otp.

Druhým problémem je existence neplatných odkazů v adresáři s manuálovými stránkami prostředí Erlang. Chyba vyvolaná při kompilaci vypadá takto:

```
ERROR: Unable to generate spec: read file info
/usr/lib/erlang/man/man3/cerfl.3.gz failed
ERROR: Unexpected error: rebar_abort
ERROR: generate failed while processing /home/pi/LINC-Switch/rel:
rebar_abort
Makefile:4: recipe for target 'rel' failed
make: *** [rel] Error 1
```

V chybovém hlášení je sice vypsán jeden soubor, ale ve stejném adresáři jsou takové soubory čtyři. Všechny tyto odkazy je potřeba odstranit, aby kompilace proběhla úspěšně.

```
cd /usr/lib/erlang/man/man3/
sudo rm cerfl.3.gz cerfcl.3.gz cerff.3.gz cerfcf.3.gz
```

Při samotné kompilaci dojde také ke stažení dalších komponent přepínače ze serveru GitHub.

```
make
```

Soubor `Makefile` neobsahuje instrukce pro instalaci přepínače do systémových složek. Přepínač bude tedy spouštěn z adresáře se zdrojovými kódy.

6.4 Ostatní přepínače

6.4.1 Lagopus

Přepínač Lagopus se nepodařilo na počítači Raspberry Pi nainstalovat, neboť tento přepínač nepodporuje architekturu ARM. Z tohoto důvodů vrací kompilace programu následující chybu.

```
/tmp/cccxvESm.s: Assembler messages:
/tmp/cccxvESm.s:857: Error: bad instruction `bsr [fp,#-8],r3'
```

6.4.2 OF13SoftSwitch

Posledním z testovaných softwarových přepínačů byl OF13SoftSwitch, jehož zdrojové kódy lze také nalézt na webových stránkách GitHub. Ten pro parsování paketů využívá knihovnu *NetBee*. Ta je oficiálně dostupná pouze pro platformu x86. Pro kompilaci knihovny na Raspberry Pi byl využit projekt,[22] který je odvozený z oficiálních zdrojových kódů a umožňuje kompilaci knihovny na architektuře ARM. Aby byla kompilace knihovny na systému Raspbian a Ubuntu MATE úspěšná, bylo nutné snížit verzi balíčku `bison`

[22] Repozitář projektu je dostupný na: https://github.com/shawnsschen/netbee-lite.

na verzi 2.5.[23] Dále bylo potřeba upravit soubory `src/CMakeLists.txt` a `src/nbnetvm/CMakeLists.txt` na řádcích 4 a 5, resp. 881 a 882, a změnit zde kompilátory jazyka C a C++ za kompilátory přítomné v systému. Zmíněné dvojice řádků jsou v obou souborech stejné a vypadají následovně:

```
SET(CMAKE_C_COMPILER "arm-linux-gnueabihf-gcc")
SET(CMAKE_CXX_COMPILER "arm-linux-gnueabihf-g++")
```

I přes úspěšnou kompilaci knihovny NetBee a odstranění přepínače `-Werror`, který změní všechna varování kompilátoru na chyby, ze souboru `Makefile.in` v adresáři `ofsoftswitch13`, nebyla kompilace přepínače OF13SoftSwitch úspěšná a skončila s následujícím výpisem.

```
nbee_link/libnbee_link.a(nbee_link.o): In function `sigint_handler':
/home/pi/ofsoftswitch13/nbee_link/nbee_link.cpp:42: undefined reference to
`nbDeallocatePacketDecoder(nbPacketDecoder*)'
/home/pi/ofsoftswitch13/nbee_link/nbee_link.cpp:43: undefined reference to
`nbCleanup()'
nbee_link/libnbee_link.a(nbee_link.o): In function `nblink_initialize':
/home/pi/ofsoftswitch13/nbee_link/nbee_link.cpp:72: undefined reference to
`nbIsInitialized()'
/home/pi/ofsoftswitch13/nbee_link/nbee_link.cpp:86: undefined reference to
`nbAllocatePacketDecoder(int, char*, int)'
/home/pi/ofsoftswitch13/nbee_link/nbee_link.cpp:79: undefined reference to
`nbInitialize(char const*, int, char*, int)'
collect2: error: ld returned 1 exit status
Makefile:1440: recipe for target 'udatapath/ofdatapath' failed
make[2]: *** [udatapath/ofdatapath] Error 1
make[2]: Leaving directory '/home/pi/ofsoftswitch13'
Makefile:2369: recipe for target 'all-recursive' failed
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory '/home/pi/ofsoftswitch13'
Makefile:1077: recipe for target 'all' failed
make: *** [all] Error 2
```

6.4.3 XORPlus

Přepínač XORPlus nebyl jako jediný testován, neboť jeho poslední verze pochází z roku 2009 a není dále rozvíjena. Podporuje navíc protokol OpenFlow pouze verze 1.0.

[23] Balíček a příslušnou knihovnu lze stáhnout ve formátu deb na adrese https://launchpad.net/ubuntu/+source/bison/1:2.5.dfsg-2.1/+build/3270700 a poté nainstalovat oba soubory nástrojem `dpkg`.

7 KONTROLÉR RYU

Pro separaci řídicí roviny od datové v rámci topologie cloudového datového centra byl vybrán kontrolér Ryu. Projekt je podporován japonskou společností NTT a jeho zdrojové kódy lze stáhnout na webu GitHub[24] pod licencí Apache License 2.0. Kontrolér je napsaný v jazyce Python a v tomto jazyce pro něj budou také psány aplikace. Dle Khondoke a dalších (2014) byl kontrolér Ryu podle různých kritérií, jako je modularita, podporovaná platforma, přítomnost REST API nebo úroveň dokumentace, vyhodnocen jako nejlepší. Postrádá však některé bezpečnostní funkce, jako je izolace řídicích procesů nebo podpora více instancí kontroléru/aplikací. Druhé nejvyšší skóre získal kontrolér Floodlight, který je vyvíjen společně s přepínačem Indigo Virtual Switch. (Scott-Hayward, 2015)

Ryu se skládá z předdefinovaných komponent. Tyto komponenty mohou být upravovány, rozšířeny nebo použity do nové aplikace. Hlavní část zdrojového kódu kontroléru se nachází v adresáři `ryu`. Adresář je členěn následující způsobem.

- Adresář `app` – obsahuje zdrojové kódy aplikací běžících nad kontrolérem.
- Adresář `base` – v souboru `app_manager.py` se nachází základní třída RyuApp, ze které jsou odvozovány nové aplikace pro Ryu. Také se zde nachází třída `AppManager` spravující běh aplikací.
- Adresář `controller` – zde se nachází třídy a funkce pracující s protokolem OpenFlow. Tato část se stará např. o vyřizování událostí, generování pravidel nebo zpracování paketů z/pro přepínač (třída `Datapath`).
- Adresář `lib` – poskytuje řadu knihoven k parsování hlaviček různých protokolů. Také se zde nachází knihovny pro další protokoly pro správu přepínače, jako jsou OF-CONFIG, NetFlow nebo sFlow (podadresář `xflow`).
- Adresář `ofproto` – zde se nachází zdrojové kódy pro podporu různých verzí protokolu OpenFlow (1.0 až 1.5).
- Adresář `topology` – obsahuje kód, který se stará o zjišťování topologie v okolí kontroléru pomocí protokolu LLDP.
- Další zdrojové kódy v tomto adresáři pracují např. s protokoly BGP nebo OVSDB nebo obsahují část pracující s projektem OpenStack Neutron.

[24] Zdrojové kódy kontroléru jsou k dispozici zde: https://github.com/osrg/ryu.

7.1 Instalace kontroléru

Pro instalaci kontroléru je využit správce balíčků napsaných v jazyce Python – `pip`. Ten by již měl být součástí distribuce Raspbian. Pokud by tomu tak nebylo, lze nástroj stáhnout a nainstalovat balíčkovacím nástrojem `apt`.

```
sudo apt-get install python-pip
```

Před samotnou instalací kontroléru je nutné doinstalovat balíček s hlavičkovými soubory a vývojářskými nástroji pro jazyk Python `python-dev`. Bez těchto souborů by nebyla instalace kontroléru úspěšná.

```
sudo apt-get install python-dev
```

Následně jsou k dispozici dvě varianty instalace kontroléru Ryu. Kontrolér lze nainstalovat přímo z repozitáře balíčkovacího nástroje `pip` příkazem:

```
sudo pip install ryu
```

Tímto způsobem bude kontrolér nainstalován jako běžný program do systémových adresářů a zpřístupněn příkaz `ryu` ze standardní složky programů unixových systémů.

Druhou možností je instalace ze zdrojových kódů kontroléru, čehož je využito i v této práci. Zdrojové kódy budou staženy naklonováním repozitáře git.

```
git clone git://github.com/osrg/ryu.git
```

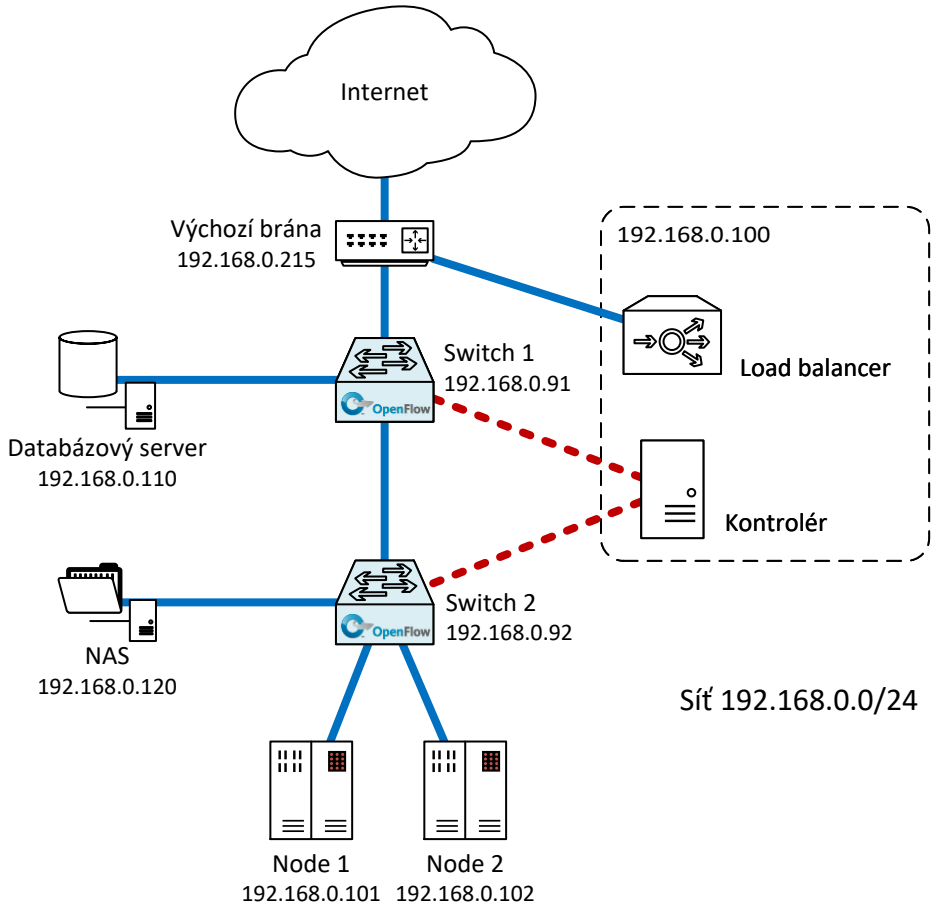
Naklonováním repozitáře vznikla v aktuálním adresáři složka `ryu` se zdrojovými kódy kontroléru, odkud bude spuštěna instalace. Zde bude využit nástroj `pip`. Parametr `.` příkazu `install` představuje aktuální složku.

```
cd ryu
sudo pip install .
```

V rámci instalace budou staženy potřebné závislosti. Ani po předchozí instalaci hlavičkových souborů kernelu, případně balíčků, jako `cython` a `cffi`, nebyla během instalace kontroléru úspěšná kompilace rozšíření v jazyce C, které slouží pro zrychlení některých operací kontroléru. Na potřebnou funkcionalitu to však vliv nemá.

8 NASAZENÍ V PROSTŘEDÍ CLOUDOVÉHO DATOVÉHO CENTRA

Funkce kontroléru a připojených přepínačů byla otestována v prostředí cloudového datového centra. Datové centrum obsahuje databázový server MySQL, úložiště NAS, výpočetní uzly pojmenované Node 1 a Node 2. Pro vyvažování zátěže na oba uzly slouží TCP/HTTP load balancer HAProxy. Load balancer a kontrolér jsou z důvodu snížení počtu zařízení oba umístěny na jednom počítači Raspberry Pi. Topologie je přehledně zobrazena na následujícím obrázku.



Obrázek 12 – Topologie cloudového datového centra

Zdroj: vlastní

Praktická část této práce se zabývá konfigurací a instalací OpenFlow přepínačů Switch 1 a Switch 2 a připojeného kontroléru. Ostatní prvky sítě jsou popsány v diplomové práci Bc. Čeňka Pozdníka „*Využití Raspberry Pi pro cloudové datové centrum*.“ Pro provoz

na lokální síti byla stanovena určitá bezpečnostní pravidla umožňující přístup na databázový server a server NAS pouze z hostitelů Node 1 a Node 2. Implementace těchto pravidel je popsána v podkapitole 8.4.

8.1 Navázání spojení Open vSwitch s kontrolérem

Spojení OpenFlow přepínače s kontrolérem může probíhat buď nezabezpečeným kanálem pouze s pomocí protokolu TCP, nebo může být komunikace šifrována. Pro ověření správnosti základní konfigurace je vhodné nejprve vyzkoušet nezabezpečené spojení. Následně je v této kapitole popsána konfigurace pro šifrování komunikace mezi přepínačem a kontrolérem. Většina konfigurace spojení probíhá na straně přepínače.

8.1.1 Nezabezpečené spojení

Na straně přepínače je potřeba v rámci Open vSwitch vytvořit nový bridge, který představuje virtuální přepínač např. s názvem `switch1`.

```
sudo ovs-vsctl add-br switch1
```

Pro tento přepínač bude nastavováno připojení ke kontroléru. Kontrolér disponuje IP adresou 192.168.0.100 a naslouchá na portu 6633, což je výchozí port Ryu. Komunikace s ním zde bude probíhat prostým TCP.

```
sudo ovs-vsctl set-controller switch1 tcp:192.168.0.100:6633
```

Dále je nutné mezi přepínačem a kontrolérem zajistit IP konektivitu. Na straně přepínače lze nastavit IP adresu buď na rozhraní mimo bridge nebo na rozhraní představující virtuální přepínač.

V této práci je využito připojení přes rozhraní, které je také použito pro směrování provozu, a tedy přiřazené k bridge. V takovém případě se IP adresa nenastavuje na síťové rozhraní, ale na bridge, který se také nachází ve výpise síťových rozhraní např. z příkazu `ifconfig`. Virtuálnímu přepínači je tak možné přiřadit IP adresu jako jakémukoliv jinému rozhraní.

Aby byla statická adresa přiřazena rozhraní i po startu systému, je nutné upravit konfigurační soubor klienta DHCP. K úpravě konfigurace lze využít například program `nano`, případně jakýkoliv jiný textový editor.

```
sudo nano /etc/dhcpcd.conf
```

Na konec souboru je potřeba přidat řádky pro konfiguraci příslušných rozhraní. Pro spojení s kontrolérem není nutné na rozhraní nastavovat adresu výchozí brány nebo serveru DNS.

```
interface switch1
static ip_address=192.168.0.91/24
```

Přiřazení IP adresy na rozhraní lze provést restartem systému nebo novým načtením konfiguračních souborů služby klienta DHCP.

```
sudo /etc/init.d/dhcpcd reload
```

Pro funkci softwarového přepínače je potřeba k přepínači přiřadit fyzická rozhraní. Jelikož Raspberry Pi disponuje pouze jedním ethernetovým portem, byl počítač vybaven adaptéry pro rozhraní USB 2.0, konkrétně modely PremiumCord KUETHERNET2. Ty se ihned po připojení stávají dalšími síťovými rozhraními připravenými k použití. Jejich názvy jsou vytvářeny podle pořadí připojení k počítači `eth1`, `eth2` a tak dále. Přiřazení rozhraní k přepínači provádí následující příkazy.

```
sudo ovs-vsctl add-port switch1 eth0
sudo ovs-vsctl add-port switch1 eth1
# a tak dále
```

Pro připojení ke kontroléru je nutné, aby rozhraní připojená k přepínači neměla nastavenou IP adresu. Vymazání IP adresy na rozhraní `eth0` lze provést takto:

```
sudo ifconfig eth0 0
```

Odebrání IP adres ostatním rozhraním je analogické k předchozímu. Systém automaticky přiřazuje novým aktivním rozhraním bez IP adresy tzv. *link-local* adresu z rozsahu 169.254.0.0/16. Tomu lze zabránit vložením další konfigurace zabraňující automatickému nastavení IP adresy do výše zmíněného konfiguračního souboru.

```
interface eth0
static ip_address=

interface eth1
static ip_address=

# a tak dále
```

Ačkoliv se může tato konfigurace jevit jako chybná, zpracování případných dalších příkazů to nebrání.

Na straně kontroléru je rozhraní připojenému k přepínači přiřazena adresa, pro kterou byl nastaven přepínač, v tomto případě 192.168.0.100. Konfiguraci statické IP adresy lze provést obdobným způsobem, jaký byl zmíněn výše.

Kontrolér bude spouštěn z adresáře s jeho zdrojovými kódy. Pokud by byl pro spojení s přepínačem použit jiný než výchozí port, např. současný oficiální port protokolu OpenFlow 6653, je potřeba port kontroléru specifikovat přepínačem `--ofp-tcp--listen-port 6653` v příkazu pro spuštění kontroléru.

```
./bin/ryu-manager --verbose
```

Výstup s úspěšným připojením dvou přepínačů může vypadat následovně:

```
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK ofp_event
  CONSUMES EventOFPHello
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPEchoReply
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7567e330>
address:('192.168.0.91', 50249)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x75688150>
move onto config mode
switch features ev
version=0x4,msg_type=0x6,msg_len=0x20,xid=0xf4f6d575L,OFPSwitchFeatures(aux
iliary_id=0,capabilities=79,datapath_id=1,n_buffers=256,n_tables=254)
move onto main mode
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7567e250>
address:('192.168.0.92', 54522)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x756884b0>
move onto config mode
switch features ev
version=0x4,msg_type=0x6,msg_len=0x20,xid=0xa77cb12cL,OFPSwitchFeatures(aux
iliary_id=0,capabilities=79,datapath_id=202481586937805L,n_buffers=256,n_ta
bles=254)
move onto main mode
```

Pomocí aplikace `gui_topology`, která se také nachází v adresáři `app`, lze zobrazit topologii připojených přepínačů. Příkaz je pak proveden s přepínačem `--observe-links` a cestou k souboru se zdrojovým kódem aplikace.

```
./bin/ryu-manager --observe-links ryu/app/gui_topology/gui_topology.py
```

Po spuštění aplikace je k dispozici webové rozhraní dostupné na IP adrese kontroléru a portu 8080. Po zadání URL http://192.168.0.100:8080 do prohlížeče je zobrazeno grafické znázornění připojených přepínačů.

8.1.2 Zabezpečené spojení

K zabezpečení komunikačního kanálu mezi přepínačem a kontrolérem jsou využívány privátní klíče, certifikáty a další soubory, které poskytuje přepínač Open vSwitch. Pro získání těchto souborů slouží nástroj pro správu a distribuci veřejných klíčů *OpenFlow public key infrastructure management utility* – `ovs-pki`. Příkazem `init` vytvoří nástroj do adresáře `/var/lib/openvswitch/pki` dvojici souborů s certifikačními autoritami pro přepínače a kontrolér. Jde o soubory `controllerca/cacert.pem` a `switchca/cacert.pem`. Obsah těchto souborů může být bezpečně zveřejněn.

```
sudo ovs-pki init
```

Dalším krokem je vytvoření privátních klíčů a příslušných certifikátů. K tomu slouží dvojice příkazů `req` a `sign`. První z nich vytvoří privátní klíč a žádost o certifikát s názvem, který je příkazu specifikován, a příponou `-privkey.pem`, resp. `-req.pem`. Následně je příkazem `sign` podepsána žádost o certifikát a vytvořen soubor s certifikátem s příponou `-cert.pem`. Oba příkazy lze zkombinovat do jednoho kroku. Soubory budou vytvořeny do aktuálního adresáře, ze kterého je příkaz prováděn. Privátní klíče a certifikáty pro kontrolér a pro přepínač zde budou vytvořeny s názvy `ctl`, resp. `sc`.

```
sudo ovs-pki req+sign ctl controller
sudo ovs-pki req+sign sc switch
```

Příkazy společně vytvořily soubory `ctl-cert.pem`, `ctl-privkey.pem`, `ctl-req.pem`, `sc-cert.pem`, `sc-privkey.pem` a `sc-req.pem`. Soubory určené pro přepínač budou využity pro nastavení TLS spojení nástrojem `ovs-vsctl`. Příkaz `set-ssl` přijímá jako parametry privátní klíč, certifikát a soubor s certifikační autoritou pro kontrolér. Soubory by měly být předány formou absolutní cesty, protože aktuální složka démona `ovs-vswitchd` a aktuální složka k vykonání příkazu se mohou lišit. Následující příkaz pracuje s privátním klíčem a certifikátem uloženým v domovském adresáři uživatele `pi`.

```
sudo ovs-vsctl set-ssl /home/pi/sc-privkey.pem /home/pi/sc-cert.pem
/var/lib/openvswitch/pki/controllerca/cacert.pem
```

Nakonec je přepínač nastaven pro použití protokolu TLS při komunikaci s kontrolérem.

```
sudo ovs-vsctl set-controller switch1 ssl:192.168.0.100:6633
```

Stejné soubory lze přesunout např. pomocí flash disku na druhý přepínač a provést stejné nastavení pomocí `set-ssl` a `set-controller`. Vygenerované privátní klíče nemají nastavená

oprávnění na čtení, proto bude pro jejich kopírování potřeba zvýšených oprávnění uživatele `root`.

Na kontrolér je potřeba přenést soubory `ctl-cert.pem`, `ctl-privkey.pem` a soubor `cacert.pem`, který se nachází v adresáři `/var/lib/openvswitch/pki/switchca/`. Kontrolér poté stačí spustit s parametry specifikujícími uvedené soubory.

```
ryu/bin/ryu-manager --ctl-privkey ctl-privkey.pem --ctl-cert ctl-cert.pem --ca-certs cacert.pem --verbose
```

Pokud po spuštění kontroléru dojde k výpisu následující chyby, jde pravděpodobně o rozdílný čas na zařízeních. Je dobré jej tedy zkontrolovat, případně aktualizovat z Internetu.

```
SSLError: [SSL: SSLV3_ALERT_BAD_CERTIFICATE] sslv3 alert bad certificate (_ssl.c:581)
```

Výstup úspěšně připojeného kontroléru vypadá téměř stejně jako při nezabezpečeném připojení. Pouze třída `eventlet.greenio.base.GreenSocket` je ve výpisu nahrazena třídou `eventlet.green.ssl.GreenSSLSocket`.

8.1.3 Čísla portů a Datapath ID

Porty jsou v rámci protokolu OpenFlow reprezentovány celými čísly, a toto očíslování je také využíváno v aplikaci na kontroléru. Číslo, které odpovídá určitému rozhraní na přepínači, lze zjistit příkazem `ovs-ofctl dump-ports` s názvem bridge a rozhraní, například:

```
sudo ovs-ofctl dump-ports switch1 eth0
```

Kontrolér rozlišuje jednotlivé přepínače podle hodnoty zvané Datapath ID. Jde o celé číslo představující MAC adresu rozhraní, které připojuje virtuální přepínač ke kontroléru. (Zde bylo rozhraní vytvořeno s novým bridge příkazem `add-br`.) Identifikační číslo přepínače `switch1` bylo v rámci Open vSwitch nastaveno takto:

```
sudo ovs-vsctl set bridge switch1 other-config:hwaddr=00:00:00:00:00:01
```

8.2 Navázání spojení Indigo Virtual Switch s kontrolérem

Pro připojení ke kontroléru využívá Indigo Virtual Switch na rozdíl od přepínače Open vSwitch, pouze rozhraní, které není přiřazeno k virtuálnímu přepínači. Na tomto rozhraní je nastavena IP adresa pro připojení ke kontroléru. Ostatní rozhraní, včetně rozhraní `ivs` reprezentujícího virtuální přepínač, IP adresu nemají. IP adresu a port kontroléru, ID přepínače, případně porty,

které bude přepínač využívat, lze specifikovat v rámci příkazu `ivs` při spuštění virtuálního přepínače.

```
sudo ivs -c 192.168.0.100:6633 --dpid=1 -i eth1 -i eth2
```

Výstup programu při úspěšném připojení ke kontroléru na základě předchozího příkazu vypadá následovně:

```
04-17 11:12:17.439816 [ivs] Starting ivs 0.5 (devel local) pid 1864
04-17 11:12:17.475066 [ovsdriver] using 4 upcall threads
04-17 11:12:17.507210 [ovsdriver] Reusing kernel datapath ivs
04-17 11:12:17.519695 [ovsdriver] Added port local
04-17 11:12:17.521630 [ovsdriver] Added port eth1
04-17 11:12:17.522458 [ofstatemanager] Setting switch DPID to
0000b827eba4a784
04-17 11:12:17.522755 [ofstatemanager] Setting switch DPID to
0000000000000001
04-17 11:12:17.523392 [ofconnectionmanager] Controller add:
tcp:192.168.0.100:6633 (remote, v5)
04-17 11:12:17.523943 [ofconnectionmanager] Controller add: /var/run/ivs-
openflow.ivs.sock (listen, v5)
04-17 11:12:17.524641 [ivs] Datapath description: raspberrypi.(none) pid
1864
04-17 11:12:17.560482 [ovsdriver] Added port eth2
04-17 11:12:17.617889 [ofconnectionmanager] Connected to controller
tcp:192.168.0.100:6633:0
04-17 11:12:17.618359 [ofconnectionmanager] Connected controllers: 1:
tcp:192.168.0.100:6633
```

Další příkazy na spuštěný přepínač, jako je výpis portů, pravidel pro směrování provozu nebo stavu, jsou realizovány programem `ivs-ctl` z jiné instance terminálu. Očíslování portů lze zjistit z výpisu příkazu `ivs-ctl show`.

```
ivs:
  kernel lookups: hit=2311 missed=371 lost=1629
  kernel flows=0
  ports:
    0 ivs (internal)
      rx: packets=0 bytes=0 errors=0 dropped=0
      tx: packets=33 bytes=4854 errors=0 dropped=0
    1 eth1
      rx: packets=2625 bytes=158805 errors=0 dropped=0
      tx: packets=175 bytes=16911 errors=0 dropped=0
    2 eth2
      rx: packets=24 bytes=3878 errors=0 dropped=0
      tx: packets=25 bytes=4098 errors=0 dropped=0
```

Odtud je možné také přidávat nebo odebírat porty přepínači. Oficiální dokumentace[25] je na popis programů `ivs` a `ivs-ctl` poměrně skoupá. Více informací o dostupných možnostech programů poskytuje příkaz `ivs -h`, resp. `ivs-ctl help`.

Z důvodu zmíněné nutnosti připojení kontroléru na rozhraní, které není součástí virtuálního přepínače, bylo nasazení v rámci cloudového datového centra otestováno pouze s přepínačem Open vSwitch.

8.3 Navázání spojení přepínače LINC s kontrolérem

Pro spojení s kontrolérem je využíváno fyzické rozhraní, které může i nemusí být zahrnuto do virtuálního přepínače. Pokud však bude využíváno také jako port přepínače, bude docházet k častým událostem Packet-in pro provoz mezi přepínačem a kontrolérem, pokud pro tento provoz nebudou existovat pravidla.

Na rozhraní připojeném ke kontroléru je nastavována příslušná IP adresa. Ostatní rozhraní IP adresu nemají. Nastavení `datapath_id` v konfiguračním souboru nemělo efekt pro komunikaci s kontrolérem. Alternativní možnost nastavení identifikace přepínače pro kontrolér je změna MAC adresy rozhraní, které připojuje přepínač ke kontroléru. Při změně MAC adresy musí být vypnuta síťová služba Linuxu a také rozhraní musí být ve stavu `down`. Po změně MAC adresy bude vrácen stav rozhraní zpět a služba znovu spuštěna.

```
sudo /etc/init.d/networking stop
sudo ifconfig eth0 down
sudo ifconfig eth0 hw ether 00:00:00:00:00:01
sudo ifconfig eth0 up
sudo /etc/init.d/networking start
```

Na rozdíl od výše zmíněných přepínačů, kde je konfigurace prováděna formou příkazů v CLI, je spojení přepínače LINC s kontrolérem definováno v konfiguračním souboru `sys.config`. Konfigurace je napsána v jazyce Erlang a její syntaxe připomíná formát JSON. Soubor, který zde bude upravován, se nachází v adresáři `rel/linc/releases/1.0/`. Základní konfigurace je představena níže.

```
[
 {linc,
  [
   {capable_switch_ports,
    [
      {port, 1, [{interface, "eth1"}]},
```

[25] Dokumentace se nachází na adrese https://floodlight.atlassian.net/wiki/spaces/indigodocs/pages/2719759/User+Documentation.

```
      {port, 2, [{interface, "eth2"}]},
      {port, 3, [{interface, "eth3"}]}
    ]},
   {capable_switch_queues,
    [
    ]},
   {logical_switches,
    [
     {switch, 0,
      [
       {controllers,
        [
          {"switch1-controller", "192.168.0.100", 6633, tcp}
        ]},
       {datapath_id, "00:00:00:00:00:01"},
       {ports, [
                {port, 1, [{queues, []}]},
                {port, 2, [{queues, []}]},
                {port, 3, [{queues, []}]}
               ]}
      ]}
    ]}
  ]}
  %% Konfigurace závislostí vynechána.
].
```

První část `capable_switch_ports` udává, která síťová rozhraní mohou být použita pro směrování. Porty přepínače byly ponechány bez front. Následuje definice logických přepínačů v bloku `logical_switches`. V testované topologii je na obou zařízeních právě jeden virtuální přepínač. Tak je tomu i v této konfiguraci. Pro logický přepínač je specifikováno připojení ke kontroléru a přiřazené porty.

Spuštění přepínače provádí následující příkaz.

```
sudo rel/linc/bin/linc console
```

Při úspěšném připojení ke kontroléru po spuštění přepínače podává přepínač následující výstup.

```
Exec: /home/pi/LINC-Switch/rel/linc/erts-6.2/bin/erlexec -boot
/home/pi/LINC-Switch/rel/linc/releases/1.0/linc -mode embedded -config
/home/pi/LINC-Switch/rel/linc/releases/1.0/sys.config -args_file
/home/pi/LINC-Switch/rel/linc/releases/1.0/vm.args -- console
Root: /home/pi/LINC-Switch/rel/linc
Erlang/OTP 17 [erts-6.2] [source] [smp:4:4] [async-threads:10] [kernel-
poll:false]

load_driver 'netlink_drv' from: '/home/pi/LINC-Switch/rel/linc/lib/netlink-
1/priv'
19:43:09.559 [info] Application lager started on node linc@raspberrypi
19:43:09.561 [info] Application ssh started on node linc@raspberrypi
19:43:09.564 [info] Application enetconf started on node linc@raspberrypi
19:43:09.818 [info] Application linc started on node linc@raspberrypi
Eshell V6.2  (abort with ^G)
(linc@raspberrypi)1> 19:43:10.735 [info] Created port:
{port,1,[{queues_status,disabled},{queues,[]},{config,{port_configuration,u
```

```
ndefined,up,false,false,false}},{features,{features,undefined,'100Mb-
FD',true,copper,unsupported}},{interface,"eth1"}]}
19:43:10.886 [info] Created port:
{port,2,[{queues_status,disabled},{queues,[]},{config,{port_configuration,u
ndefined,up,false,false,false}},{features,{features,undefined,'100Mb-
FD',true,copper,unsupported}},{interface,"eth2"}]}
19:43:11.035 [info] Created port:
{port,3,[{queues status,disabled},{queues,[]},{config,{port configuration,u
ndefined,up,false,false,false}},{features,{features,undefined,'100Mb-
FD',true,copper,unsupported}},{interface,"eth3"}]}
19:43:11.048 [info] Connected to controller 192.168.0.100:6633/0 using OFP
v4
```

Nyní je k dispozici příkazový řádek přepínače. Ačkoli je aktuální výzva příkazového řádku zaplněna informacemi o připojených portech přepínače, nebrání to v zadávání nových příkazů. Pro nové zobrazení výzvy stačí stisknout klávesu Enter.

Pravidla pro směrování provozu, která přepínač přijal od kontroléru, lze zobrazit následujícím příkazem. Jde o dotaz na přepínač číslo 0 a tabulku pravidel s ID 0.

```
ets:tab2list(linc:lookup(0, flow_table_0)).
```

8.4 Tvorba aplikace

Jak již bylo zmíněno výše, aplikace, které běží v rámci kontroléru, se nachází v adresáři `app`. Aplikace, která byla použita v rámci cloudového datového centra, vychází z již v kontroléru přítomné aplikace `SimpleSwitch13`. Číslo `13` v názvu značí OpenFlow verze 1.3, pro který je aplikace určena. Tato třída již obsahuje metody pro obsluhu událostí Packet-in, které vznikají při doručení paketu, pro který přepínač nemá odpovídající pravidlo, a zpráv Features reply, které jsou posílány při ustanovování spojení mezi přepínačem a kontrolérem. Aplikace `SimpleSwitch13` disponuje funkcionalitou učení MAC adres, jakou má tradiční přepínač. Ta byla pro účely datového centra odstraněna a nahrazena vkládáním vlastních pravidel.

Aplikace pro cloudové datové centrum se nazývá `SimpleCloudSwitch`. Hlavička třídy s konstruktorem aplikace pro Ryu vypadá následovně:

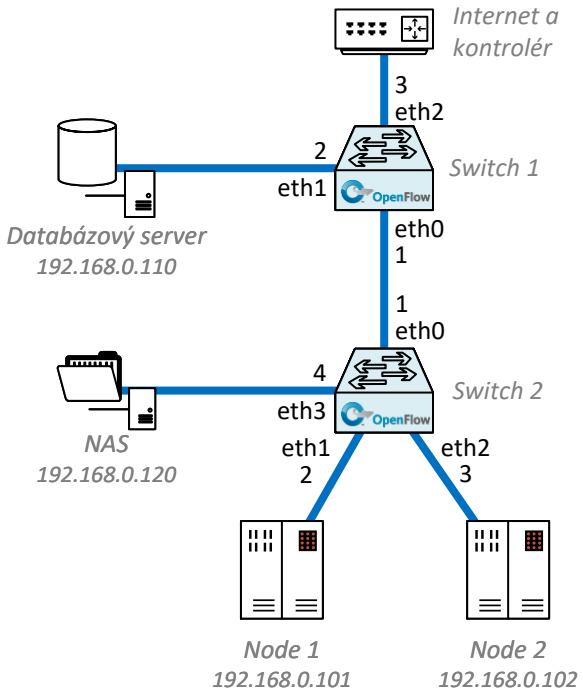
```
from ryu.base import app_manager

class SimpleCloudSwitch(app_manager.RyuApp):

      def __init__(self, *args, **kwargs):
            super(SimpleCloudSwitch, self).__init__(*args, **kwargs)
```

V aplikaci jsou využity další importy např. pro parsovaní hlaviček paketů nebo naslouchání událostí.

Hlavička funkce vyřizující událost při připojení přepínače ke kontroléru je dekorována třídou `EventOFPSwitchFeatures`. Tím je funkce přiřazena k události a je při jejím výskytu s příslušnými parametry spouštěna. Popis funkcí pro ošetření dalších událostí OpenFlow verze 1.3 včetně příslušných datových struktur se nachází v oficiální dokumentaci.[26]

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
```

Ve výše zmíněné funkci jsou do přepínače uložena pravidla směrování provozu na úrovní protokolu IP. Přepínač je zde rozlišen podle hodnoty `ev.msg.datapath.id`. Pravidla v této aplikaci vždy pracují se vstupním portem paketu a nastavují výstupní port. Porty jsou v aplikaci identifikovány celým číslem. Pro přehlednost je užitečné jejich zakreslení do topologie, viz Obrázek 13.



Obrázek 13 – Očíslování portů přepínačů v rámci topologie

Zdroj: vlastní

Pravidla pro směrování provozu jsou do přepínačů vkládána pomocí metody `add_flow`, která přijímá jako parametr objekt `datapath` představující přepínač, prioritu, objekt `match` s údaji pro testování shody s paketem, `idle` a `hard_timeout`, který určuje dobu, po kterou je pravidlo platné, a parametr, který umožňuje pracovat s vyrovnávací pamětí na přepínači. Hodnoty `idle`

[26] Dokumentace se nachází na adrese http://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html.

a `hard_timeout` jsou zde nastavovány na nulovou hodnotu. Pravidla tedy nikdy neztratí platnost.

Následující volání povoluje provoz na hostitele Node 1 z Internetu na přepínači Switch 1. Volání funkce tedy předchází podmínka `datapath.id == 1`. Pokud je kontrolována IP adresa paketu, je nutné specifikovat protokol vyšší vrstvy v hlavičce ethernetového rámce, tzv. *EtherType*. Podobně je tomu u dalších protokolů 3. a vyšší vrstvy.

```
self.add_flow(datapath, 100, parser.OFPMatch(in_port=2, eth_type=0x800,
ipv4_dst='192.168.0.101'), parser.OFPActionOutput(4,
ofproto.OFPCML_NO_BUFFER)])
```

Aby mohly uzly sítě kontaktovat jiné zařízení pomocí IP adresy, musí nejdříve zjistit MAC adresu následujícího zařízení. K tomu slouží protokol ARP. Případné ARP dotazy jsou zachyceny v metodě obsluhující události Packet-in a povoleny na určitých portech a pro povolené IP adresy. Objekt `OFPMatch` zde obsahuje hodnotu EtherType pro protokol ARP `0x806` a zdrojovou a/nebo cílovou IP adresu (pole `arp_spa`, resp. `arp_tpa`). Pro sestavení zprávy typu Modify-State, která slouží k ukládání pravidel na přepínač, je využíván objekt `OFPFlowMod`. Instrukce pro zpracování aktuálního paketu, přijatého ze zprávy Packet-in, dostane přepínač formou zprávy typu Packet-out, sestavenou pomocí třídy `OFPPacketOut`.

Spuštění aplikace pak bylo provedeno z adresáře `ryu` následujícím příkazem:

```
./bin/ryu-manager --ctl-privkey /home/pi/certificates/ctl-
privkey.pem --ctl-cert /home/pi/certificates/ctl-cert.pem --ca-certs
/home/pi/certificates/switchca/cacert.pem --verbose
ryu/app/simple_cloud_switch.py
```

Tím, že přepínač dostane instrukce pro zpracování paketu až při jeho prvním přijetí, dochází u prvního požadavku k mírnému zpoždění. Zpoždění prvního IP paketu je také způsobeno samotným ARP požadavkem, který je potřebné provést pouze poprvé.

```
Pinging 192.168.0.102 from 192.168.0.140 with 32 bytes of data:
Reply from 192.168.0.102: bytes=32 time=36ms TTL=128
Reply from 192.168.0.102: bytes=32 time=1ms TTL=128
Reply from 192.168.0.102: bytes=32 time=1ms TTL=128
Reply from 192.168.0.102: bytes=32 time=1ms TTL=128
```

Do výpisu kontroléru bylo také přidáno varování při pokusu o přístup k serveru NAS z jiné IP adresy, než je na uzlech Node 1 a Node 2. Varování se vyskytne jak pro protokol IP, tak pro protokol ARP.

```
EVENT ofp_event->SimpleCloudSwitch EventOFPPacketIn
paket in datapath: 1, port: 3
```

```
Attempt to reach NAS server from 192.168.0.47
EVENT ofp_event->SimpleCloudSwitch EventOFPPacketIn
paket in datapath: 1, port: 3
Attempt to reach NAS server from 192.168.0.215
```

8.5 Další možnosti konfigurace přepínače Open vSwitch

V rámci zabezpečení sítě je možné nastavit přepínači Open vSwitch režim chování při ztrátě spojení s kontrolérem. Tyto režimy jsou k dispozici dva a byly stručně popsány v kapitole 2.4.2. Není-li žádný režim nastaven nebo je nastaven režim `standalone`, dojde po ztrátě spojení s kontrolérem ke smazání všech pravidel pro směrování provozu přijatých od kontroléru. V této chvíli se Open vSwitch začne chovat jako tradiční přepínač, který přeposílá provoz podle MAC adres přijatých na rozhraních.

Naproti tomu režim `secure` způsobí zachování pravidel i při ztrátě spojení s kontrolérem. Provoz je tedy směrován stále stejným způsobem, pouze pakety směrované na kontrolér jsou zahozeny. Pokud se v tomto režimu nenachází na přepínači žádná pravidla, je veškerý příchozí provoz blokován.

Pro testovací účely se více hodí první režim, kdy při restartu spojení s kontrolérem dojde k vymazání pravidel z přepínače a nahrání nových, a nezůstávají zde chybná pravidla z přechozích pokusů. Bezpečný režim lze nastavit na přepínači Open vSwitch takto:

```
sudo ovs-vsctl set-fail-mode switch1 secure
```

Opětovné nastavení režimu `standalone` lze provést buď smazáním režimu příkazem `del-fail-mode` nebo nahrazením parametru `secure` za `standalone`. Zjištění aktuálního režimu lze dosáhnout příkazem `show` nebo `get-fail-mode`.

Pokud by se v topologii vyskytly logické smyčky, typicky z důvodu redundance, poskytuje Open vSwitch implementaci protokolů STP (IEEE 802.1d) a RSTP (IEEE 802.1d-2004). Zapojení přepínače do činnosti protokolu STP lze zapnout následujícím příkazem.

```
sudo ovs-vsctl set bridge switch1 stp_enable=true
```

Protokol RSTP je konfigurován obdobně, přičemž poskytuje výrazně více možností konfigurace. Další dostupné příkazy ovlivňující práci protokolů STP a RSTP na přepínači Open vSwitch lze nalézt v dokumentaci nástroje `ovs-vsctl`.[27] Funkcionalitu protokolu STP lze také

[27] Dokumentace příkazu je dostupná ve formátu HTML na stránce http://openvswitch.org/support/dist-docs-2.5/ovs-vsctl.8.html. Dokumentace ostatních příkazů je dostupná v nadřazeném adresáři dist-docs-2.5.

zajistit aplikací na kontroléru. Například kontrolér Ryu disponuje vzorovými aplikacemi `simple_switch_stp.py` a `simple_switch_stp_13.py`.

ZÁVĚR

Softwarově definované sítě je nové paradigma vznikající na základě omezení tradičních síťových technologií. Toto nové paradigma umožňuje vyčlenit řídicí část ze síťových zařízení do centralizovaného kontroléru. Síťová zařízení se tak ze samostatných jednotek stávají jednoduchými prvky čekajícími na instrukce z kontroléru. Použitím standardizovaného protokolu OpenFlow jsou síťové přepínače snadno nahraditelné a odpadá tak závislost sítě na výrobci.

V teoretické části práce bylo představeno šest dostupných přepínačů pro operační systém Linux. Všechny zde zmíněné přepínače disponují volně přístupným zdrojovým kódem, převážně na serveru GitHub. Podpora protokolu OpenFlow byla podmínkou pro zařazení do seznamu dostupných přepínačů.

Praktická část se převážně zabývá instalací přepínačů na systémy Raspbian a Ubuntu MATE, běžících na počítačích Raspberry Pi. Instalace přepínačů většinou neprobíhala podle dostupných pokynů k instalaci, proto zde bylo popsáno také řešení případných problémů, na kterých by se nezkušený uživatel mohl delší dobu zdržet.

Nejprve je popsána instalace přepínače Open vSwitch. Tento přepínač disponuje nejkomplexnější instalací s několika možnými postupy. Pro zrychlení zpracování paketů je zde využíván modul jádra operačního systému, jehož kompilace a zavedení do systému byly také v rámci instalace popsány.

Druhým softwarovým přepínačem, jehož instalace byla zde popsána, je Indigo Virtual Switch, který poskytuje podstatně menší možnosti konfigurace než přepínač Open vSwitch. Také dokumentace k tomuto virtuálnímu přepínači je poměrně málo obsáhlá. Více informací poskytne nápověda obsažená v samotných příkazech. Protože přepínač neumožňuje spojení s kontrolérem přes port, který je zároveň používán virtuálním přepínačem, nebyl tento přepínač z důvodu nedostatku síťových portů nasazen do prostředí cloudového datového centra.

Posledním instalovaným a konfigurovaným přepínačem je LINC. Ten je na rozdíl od ostatních realizovaných přepínačů spouštěn v běhovém prostředí Erlang a jeho konfigurace je specifikována v konfiguračním souboru. Při komunikaci s kontrolérem přes port, který je zároveň součástí virtuálního přepínače, docházelo ke zpracování tohoto provozu protokolem OpenFlow, což nebylo v rámci nasazení žádoucí.

Kompilace přepínače Lagopus nebyla úspěšná z důvodu nekompatibility programu s architekturou procesoru. Přepínač OF13SoftSwitch se nepodařilo nainstalovat kvůli nefunkčnímu propojení s knihovnou NetBee a přepínač XORPlus byl vyhodnocen jako zastaralý a jeho nasazení na počítač Raspberry Pi proto nebylo realizováno.

V prostředí cloudového datového centra byl otestován přepínač Open vSwitch. Ten úspěšně směroval a filtroval provoz podle pravidel přijatých od kontroléru. Úspěšná byla také komunikace Open vSwitch s kontrolérem zabezpečeným kanálem pomocí vygenerovaných certifikátů. V rámci nasazení byla popsána konfigurace spojení s kontrolérem pro tři instalované přepínače a důležité aspekty aplikace běžící v rámci kontroléru.

Práce je vhodná jako základ pro budování složitější softwarově definované sítě složené ze zařízení Raspberry Pi. Složitější možnosti nasazení pak spočívají převážně v logice aplikací definovaných na kontroléru.

POUŽITÁ LITERATURA

AZODOLMOLKY, Siamak. 2013. *Software Defined Networking with OpenFlow*. Birmingham: Packt Publishing. ISBN 978-1-84969-872-6.

BALCHUNAS, Aaron. 2014a. Multilayer Switching. In: *Router Alley* [online]. [cit. 18. 1. 2017]. Dostupné z: http://www.routeralley.com/guides/multilayer_switching.pdf

BALCHUNAS, Aaron. 2014b. Switching Tables. In: *Router Alley* [online]. [cit. 18. 1. 2017]. Dostupné z: http://www.routeralley.com/guides/switching_tables.pdf

BAVOTA, C. 2017. *Basic Theory of Operation of a L2 Switch or Bridge* [online]. [cit. 17. 01. 2017]. Dostupné z: http://computernetworkingsimplified.com/data-link-layer/basic-theory-operation-l2-switch/

BOUŠKA, Petr. 2005. *Cisco IOS 9 - Spanning Tree Protocol* [online]. [cit. 18. 1. 2017]. Dostupné z: http://www.samuraj-cz.com/clanek/cisco-ios-9-spanning-tree-protocol/

ČEVELA, Lubomír. 2015. OwnCloud na Raspberry Pi 2. In: *Linux EXPRES* [online]. [cit. 18. 2. 2017]. Dostupné z: https://www.linuxexpres.cz/hardware/owncloud-na-raspberry-pi-2

FATDOG.NL. 2016. Raspberry Pi Review. In: *Slackware ARM on a Raspberry Pi* [online]. [cit. 18. 2. 2017]. Dostupné z: http://rpi.fatdog.eu/index.php?p=rpiviews

FLOWFORWARDING.ORG. 2014. Quick Start Guide. In: *LINC OpenFlow Switch* [online]. [cit. 20. 4. 2017]. Dostupné z: https://github.com/FlowForwarding/LINC-Switch/blob/master/docs/LINC_Switch_Quick_Start_Guide.pdf

FROOM, Richard, Balaji SIVASUBRAMANIAN a Erum FRAHIM. 2015. *Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: Foundation learning for SWITCH 642-813*. 1st. Cisco Press. ISBN 978-1-58705-884-4.

GARRISON, Justin. 2011. *What Is a Virtual Machine Hypervisor?* [online]. [cit. 2. 2. 2017]. Dostupné z: http://www.howtogeek.com/66734/htg-explains-what-is-a-hypervisor/

GAY, Warren. 2014. *Mastering the Raspberry Pi*. 1. New York: Apress. ISBN 978-1-4842-0181-7.

HÉGR, Tomáš. 2016. SDN technologie v souvislostech. [cit. 10. 3. 2017]. Dostupné z: http://www.cybersecurity.cz/data/CVUT.pdf

HENDRIKS, Luuk, Ricardo SCHMIDT, Ramin SADRE a další. 2016. Assessing the Quality of Flow Measurements from OpenFlow Devices. In: *Traffic Measurement and Analisys workshop, Belgium* [online]. [cit. 8. 2. 2017]. Dostupné z: http://tma.ifip.org/2016/papers/tma2016-final34.pdf

HORÁČEK, Petr. 2012. Raspberry π VIII. - Úvod do GPIO. In: *Linux Software* [online]. [cit. 18. 2. 2017]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1953

KHONDOKE, Rahamatullah, Adel ZAALOUK, Ronald MARX a další. 2014. Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers. In: *Semantic Scholar* [online]. [cit. 14. 4. 2017]. Dostupné z: https://pdfs.semanticscholar.org/1fae/98c9d3bed22a539155d5e93a8d420bd22837.pdf

KREUTZ, Diego, Fernando RAMOS, Paulo VERISSIMO a další. 2014. Software-Defined Networking: A Comprehensive Survey. In: *arXiv:1406.0440v3* [online]. [cit. 26. 1. 2017]. Dostupné z: https://arxiv.org/pdf/1406.0440.pdf

KUBICA, Tomáš. 2014. Část první – REST rozhraní. In: *Vývoj aplikací pro HP VAN SDN kontroler* [online]. [cit. 10. 3. 2017]. Dostupné z: https://www.netsvet.cz/wp-content/uploads/2015/10/hp-sdn-python-lab-1.02.pdf

LARA, Adrian, Anisha KOLASANI a Byrav RAMAMURTHY. 2014. Network Innovation using OpenFlow: A Survey. In: [online]. s. 493 – 512 [cit. 8. 2. 2017]. ISSN 1553-877x. Dostupné z: http://ieeexplore.ieee.org/document/6587999/

MOLLOY, Derek. 2016. *Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux*. Indianapolis, Indiana: John Wiley & Sons, Inc. ISBN 978-1-119-18868-1.

MONK, Simon. 2016. *Raspberry Pi Cookbook*. 2. Sebastopol, CA: O'Reilly Media. ISBN 978-1-4919-3904-8.

NIPPON TELEGRAPH AND TELEPHONE CORPORATION. 2016. *Introduction to Lagopus software switch* [online]. [cit. 29. 1. 2016]. Dostupné z: http://www.lagopus.org/lagopus-book/en/html/introduction.html#software-requirement

OPEN NETWORKING FOUNDATION. 2012. Software-Defined Networking: The New Norm for Networks. In: *Open Networking Foundation White Paper* [online]. [cit. 26. 1. 2017]. Dostupné z: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

OSWALT, Matt. 2014. *[SDN Protocols] Part 4 - OpFlex and Declarative Networking* [online]. [cit. 2. 2. 2017]. Dostupné z: https://keepingitclassless.net/2014/09/sdn-protocols-4-opflex-declarative-networking/

PFAFF, Ben a Bruce DAVIE. 2013. *RFC 7047 - The Open vSwitch Database Management Protocol* [online]. [cit. 19. 1. 2017]. Dostupné z: https://tools.ietf.org/html/rfc7047

PFAFF, Ben, Justin PETTIT, Teemu KOPONEN a další. 2015. The Design and Implementation of Open vSwitch. *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15).* Oakland, CA: USENIX Association, s. 117 – 130. ISBN 978-1-931971-218. Dostupné z: https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf

PICA8, INC.. 2014. PicOS Overview. In: *Pica8* [online]. [cit. 20. 1. 2017]. Dostupné z: http://www.pica8.com/wp-content/uploads/2015/08/pica8-whitepaper-picos-overview.pdf

PROJECT FLOODLIGHT. 2017. Indigo Virtual Switch. In: *Project Floodlight* [online]. [cit. 19. 1. 2017]. Dostupné z: http://www.projectfloodlight.org/indigo-virtual-switch/

PROJECT FLOODLIGHT. 2017. Indigo. In: *Project Floodlight* [online]. [cit. 19. 1. 2017]. Dostupné z: http://www.projectfloodlight.org/indigo/

RASPBERRY PI FOUNDATION. 2017. *Raspberry Pi 1 Model A+* [online]. [cit. 18. 2. 2017]. Dostupné z: https://www.raspberrypi.org/products/model-a-plus/

REICH, Joshua, Christopher MONSANTO, Nate FOSTER a další. 2013. Modular SDN Programming with Pyretic. In: *Princeton CS* [online]. [cit. 11. 2. 2017]. Dostupné z: https://www.cs.princeton.edu/~jrex/papers/pyretic13.pdf

RISC OS OPEN LIMITED. 2011. Welcome. In: *RISC OS Open* [online]. [cit. 16. 2. 2017]. Dostupné z: https://www.riscosopen.org/content/

ROBUCK, Mike. 2015. *What's Next Open Networking Foundation Evolution?* [online]. [cit. 8. 2. 2017]. Dostupné z: https://www.sdxcentral.com/articles/news/whats-next-for-the-onf/2015/10/

SAKAIDA, Norio, Hirokazu TAKAHASHI, Masahiro YOSHIDA a další. 2016. SDN Software Switch Lagopus and NFV Service Orchestrator vConductor for Developing SDN/NFV. In: *NTT Technical Review, Vol. 14 No. 3 Mar. 2016* [online]. [cit. 21. 1. 2017]. Dostupné z: https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201603fa2.pdf

SCOTT-HAYWARD, Sandra. 2015. Design and deployment of secure, robust, and resilient SDN Controllers. In: *2015 1st IEEE Conference on Network Softwarization (NetSoft)* [online]. [cit. 24. 4. 2017]. Dostupné z: http://pure.qub.ac.uk/portal/files/17774519/Design_and_Deployment.pdf

SHAHBAZ, Muhammad, Sean CHOI, Ben PFAFF a další. 2016. PISCES: A Programmable, Protocol-Independent Software Switch. *ACM SIGCOMM.* Florianópolis, Brazil. Dostupné z: http://pisces.cs.princeton.edu/papers/sigcomm16-pisces.pdf

SONG, Haoyu. 2013. Protocol-Oblivious Forwarding: Unleash the Power of SDN through a Future-Proof Forwarding Plane. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* [online]. [cit. 2. 2. 2017]. Dostupné z: https://pdfs.semanticscholar.org/0e44/a8e99cc9c5d999297b9f0edd0be58f16e75d.pdf

STERLING, Thomas. 2002. *Beowulf Cluster Computing with Linux.* Cambridge, Massachusetts, London: The MIT Press, s. 102 – 104. ISBN 0-262-69274-0.

THE OPEN NETWORKING FOUNDATION. 2009. OpenFlow Switch Specification Version 1.0.0 (Wire Protocol 0x01). In: *ONF Technical Library* [online]. [cit. 9. 2. 2017]. Dostupné z: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

THE OPEN NETWORKING FOUNDATION. 2012. OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04). In: *ONF Technical Library* [online]. [cit. 8. 2. 2017].

Dostupné z: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

THE OPEN NETWORKING FOUNDATION. 2015. OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06). In: *ONF Technical Library* [online]. [cit. 8. 2. 2017]. Dostupné z: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

UPTON, Eben a Gareth HALFACREE. 2013. *Raspberry Pi Uživatelská příručka*. 1. Brno: Computer Press. ISBN 978-80-251-4116-8.

VAUGHAN-NICHOLS, Steven. 2013. Build your own supercomputer out of Raspberry Pi boards. In: *ZDNet* [online]. [cit. 18. 2. 2017]. Dostupné z: http://www.zdnet.com/article/build-your-own-supercomputer-out-of-raspberry-pi-boards/

VAUGHAN-NICHOLS, Steven. 2011. *OpenFlow: The Next Generation of the Network?* [online]. Long Beach, CA: IEEE Computer Society [cit. 8. 2. 2017]. ISSN 0018-9162. Dostupné z: http://www.ic.unicamp.br/~bit/ensino/mo809_1s13/papers/SDN/auxiliar/OpenFlow-%20The%20Next%20Generation%20of%20the%20Network%3F%20.pdf

ZIMMERMAN, Joann a Charles SPURGEON. 2013. *Ethernet Switches*. Sebastopol, CA: O'Reilly Media, Inc. ISBN 978-1-449-36730-5.

SEZNAM PŘÍLOH

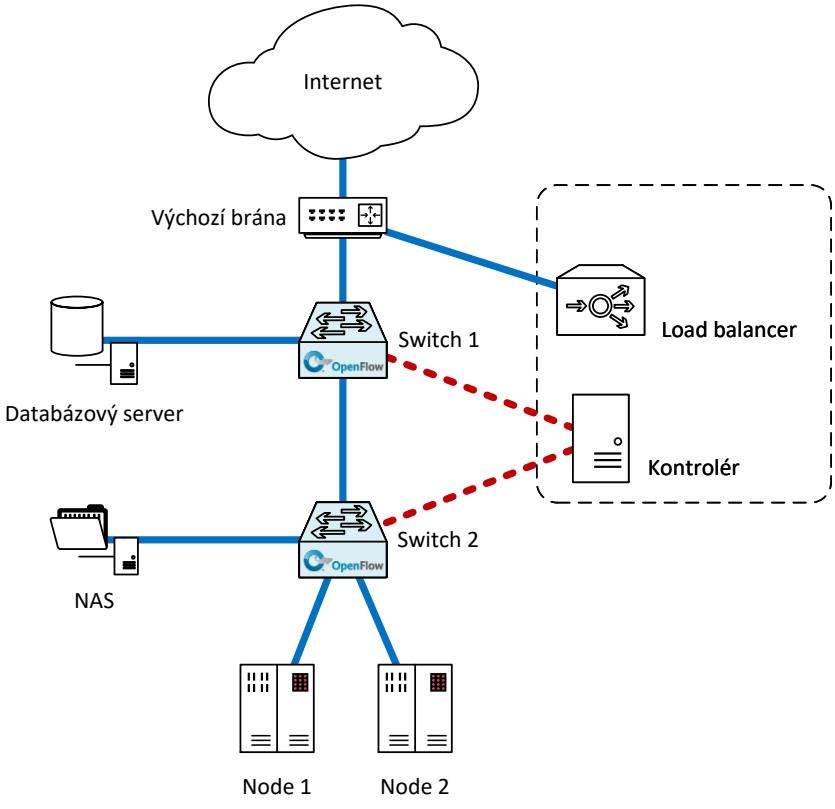
Příloha A – *Požadavky na realizaci topologie v rámci cloudového datového centra*

Pro realizaci síťové topologie představené v této práci jsou potřeba tři zařízení Raspberry Pi vybavené následujícími komponentami:

- nejméně dva USB porty na jednom zařízení a tři na druhém, v případě absence ethernetového portu o jeden USB port na každém zařízení navíc,
- karta microSD o velikosti minimálně 8 GB,
- nejméně pět adaptérů z USB na RJ-45, v případě absence ethernetového portu na zařízení o jeden navíc za každý chybějící port,
- pokud nedostačuje počet portů USB na zařízení, je možné použít USB hub.

Mezi základní potřebné softwarové vybavení patří operační systém, např. Raspbian nebo Ubuntu MATE, přepínač Open vSwitch a kontrolér Ryu:

- Raspbian, ke stažení zde: https://www.raspberrypi.org/downloads/raspbian/,
- Ubuntu MATE na adrese https://ubuntu-mate.org/download/,
- Open vSwitch na serveru GitHub: https://github.com/openvswitch/ovs nebo na oficiálních stránkách přepínače: http://openvswitch.org,
- kontrolér Ryu také na serveru GitHub: https://github.com/osrg/ryu.



Příloha B – *Obsah přiloženého CD*

Na přiloženém CD se nachází:

- aplikace pro kontrolér v souboru `simple_cloud_switch.py`,
- text práce ve formátu pdf.