SOFTWAROVĚ DEFINOVANÉ SÍTĚ

Filip Holík

Soňa Neradová

Recenzenti:
doc. Ing. Jitka Komárková, Ph.D.
doc. Ing. Ladislav Soběslav, Ph.D.

ISBN 978-80-7560-235-0 (pdf)

Tato e-kniha neprošla jazykovou korekturou.

Tato e-kniha s názvem Softwarově definované sítě byla vytvořena s finanční podporou TA ČR.

Předmluva

O publikaci

Cílem této publikace je srozumitelným způsobem představit revoluční technologii v počítačových a datových sítích – softwarově definované sítě (SDN). Publikace je zaměřena na čtenáře, kteří o této technologii nemají hlubší znalosti, ale rádi by získali povědomí alespoň o základních principech. A kdo ví, třeba si i možná prakticky vyzkoušeli nějaké jednodušší příklady. Text proto nejde zbytečně do hloubky a místo toho shrnuje to nejdůležitější ze všech podstatných oblastí SDN.

Hlavní motivací pro vznik této publikace byla absence podobných materiálu v českém jazyce. Publikace tedy slouží hlavně pro seznámení se s technologií SDN a pro podrobnější proniknutí "pod pokličku"je nutné použít zahraniční literaturu – zejména oficiální standardy a dokumentace. Pro tyto případy publikace obsahuje příslušné odkazy a tipy na to, kde najít více informací (což v této oblasti bohužel nebývá úplně snadné).

Na druhou stranu, publikace obsahuje přehledové tabulky a seznamy těch nejdůležitějších a nejpoužívanějších parametrů, funkcí a vlastností spojených s SDN. Pro základní práci by tak nemělo být potřeba vyhledávat v externích dokumentacích.

Struktura publikace

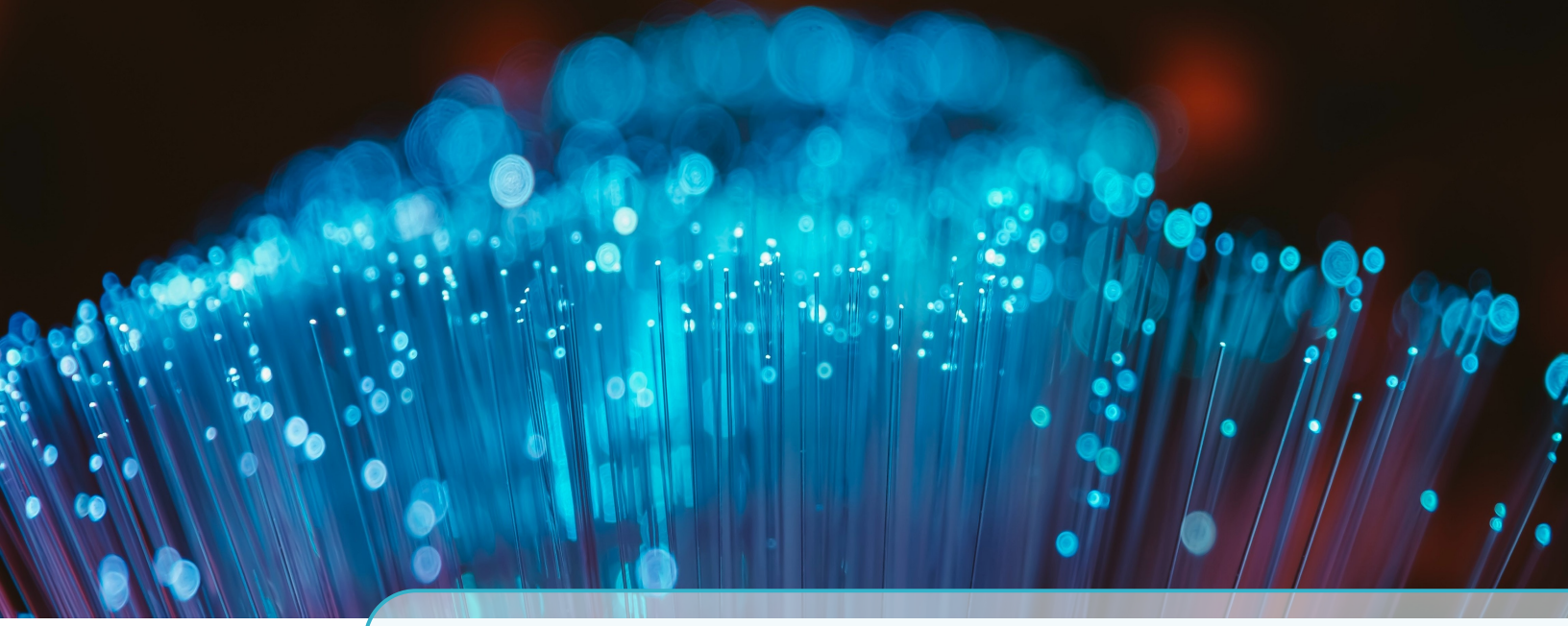
Publikace začíná srovnáním tradičních sítí a SDN, historií jejich vzniku a popisem obecné architektury SDN. Následně je popsán nejpoužívanější protokol pro implementaci technologie SDN – protokol OpenFlow. Ten je jedním z nejflexibilnějších přístupů implementace otevřených SDN. Tato část opět popisuje pouze ty nejdůležitější principy protokolu včetně jeho používaných verzí a dostupných vlastností. Následně jsou shrnuty open source SDN kontroléry, které je aktuálně možné rovnou využít pro vytvoření softwarově definované sítě.

Pátá kapitola pak představuje síťový emulátor Mininet. Díky tomuto nenáročnému nástroji je možné vyzkoušet si SDN i na průměrných počítačích a noteboocích s libovolným operačním systémem. Případní zájemci o technologii SDN se tak s pomocí uvedeného návodu mohou okamžitě pustit do tvorby inovativních síťových funkcionalit.

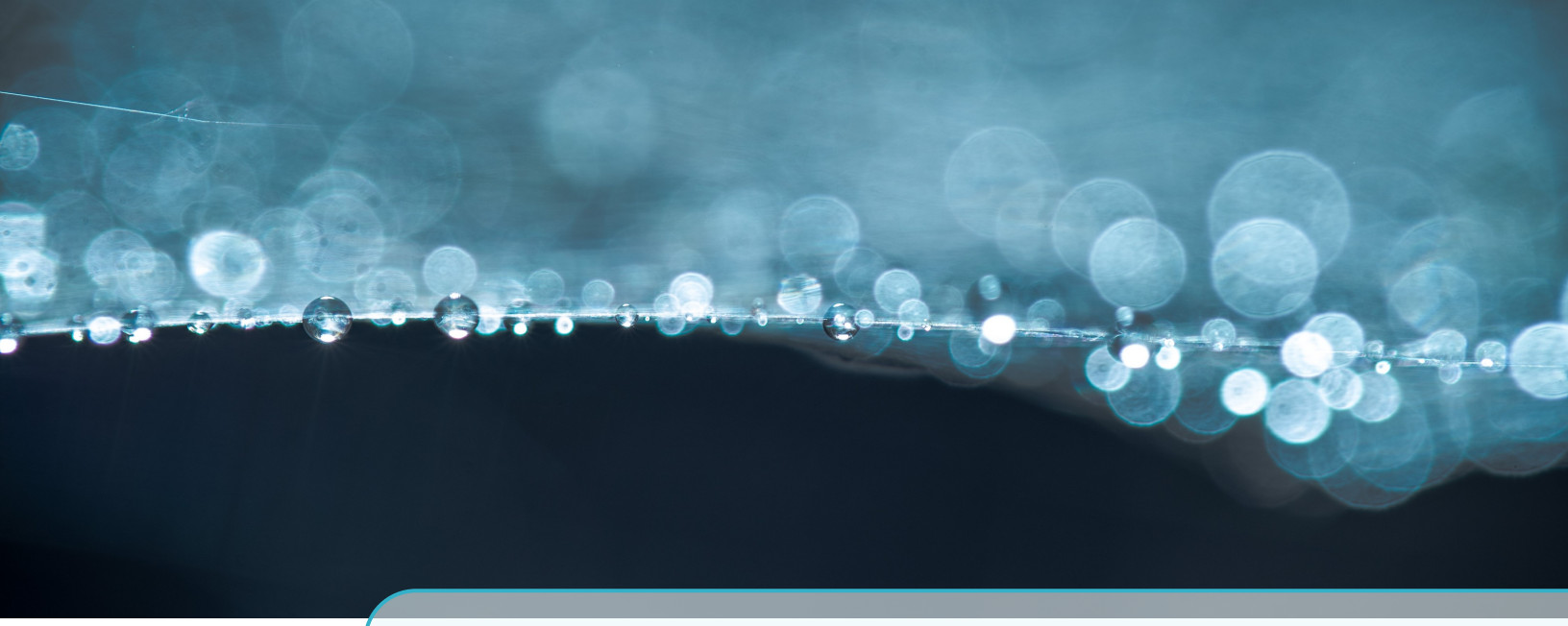
Poděkování

Tímto bychom rádi poděkovali Ing. Janu Mokráčkovi za přínos, který do této publikace vnesl v páté kapitole zaměřené na nasazení a správu SDN kontrolérů.

Dále bychom rádi poděkovali všem lidem, kteří se podíleli na vypracování této publikace. Ať už se jednalo o korektury, metodické připomínky, nebo odborné posudky, bez této pomoci by publikace nemohla vzniknout.



Obsah

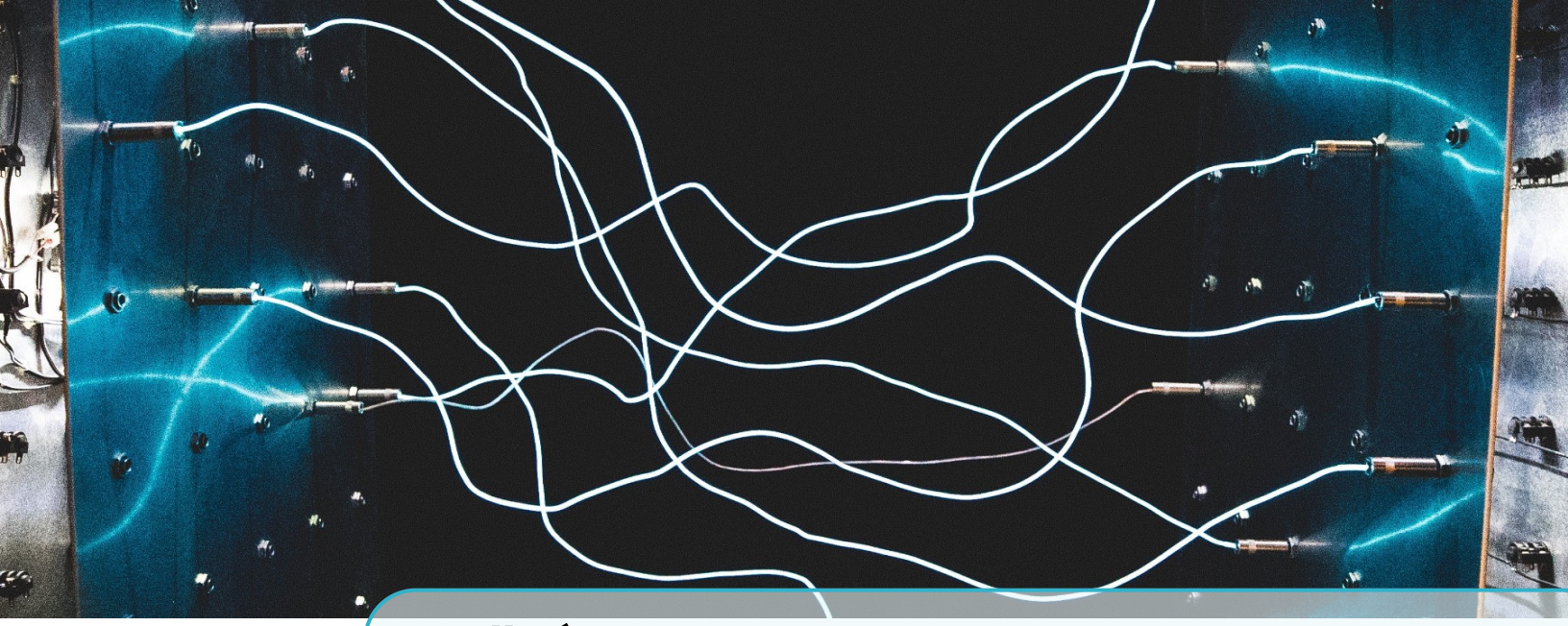


Seznam zkratek

Zkratka	Pojem	Vysvětlení
AAA	Authentication, Authorization, Accounting	Protokol zajišťující autentizaci, autorizaci a účtování
ACL	Access Control List	Mechanismus řízení přístupu pomocí pravidel
ALTO	Application Layer Traffic Optimization	Protokol poskytující aplikacím síťové informace
ARP	Address Resolution Protocol	Mechanismus získání MAC adresy na základě IP adresy
ARPANET	Advanced Research Projects Agency Network	První počítačová síť z roku 1969
AS	Autonomous System	Počítačová síť pod jednotnou správou
ASIC	Application Specific Integrated Circuit	Specificky zaměřený integrovaný obvod
BoS	Bottom of Stack	Označení posledního záhlaví protokolu MPLS
BPDU	Bridge Protocol Data Unit	Informační rámce protokolu STP
BGP	Border Gateway Protocol	Směrovací L3 protokol pro použití mezi AS
CAPWAP	Control And Provisioning of Wireless Access Points	Protokol pro centralizovanou správu bezdrátové sítě
CLI	Command Line Interface	Uživatelské rozhraní ve formě příkazového řádku
CRC	Cyclic Redundancy Check	Kontrolní kód pro detekci chyb
DARPA	The Defense Advanced Research Projects Agency	Agentura pro výzkum a vývoj technologií vhodných zejména pro vojenské použití
DHCP	Dynamic Host Configuration Protocol	Protokol pro automatickou konfiguraci připojených zařízení
DLUX	OpenDaylight User Interface	Webové grafické rozhraní kontroléru OpenDaylight
DNS	Domain Name System	Systém převádějící doménové jména na IP adresy
DSCP	Differentiated Services Code Point	6bitová součást pole pro QoS funkcionalitu
DTLS	Datagram Transport Layer Security	Bezpečnostní protokol založený na TLS
ECN	Explicit Congestion Notification	Datové pole informující o zahlcení sítové trasy

Zkratka	Pojem	Vysvětlení
ForCES	Forwarding and Control Element Separation	Protokol pro komunikaci mezi řídící a datovou vrstvou
GBP	Group Based Policy	Framework využívající systém záměrů
GDPR	General Data Protection Regulation	Evropský zákon o ochraně osobních údajů
GUI	Graphical User Interface	Grafické uživatelské prostředí
IANA	Internet Assigned Numbers Authority	Nezisková organizace spravující různé síťové zdroje
ICMP	Internet Control Message Protocol	Protokol poskytující podpůrné síťové funkce
ID	Identification Number	Unikátní číslo určené pro identifikaci entit
IDE	Integrated Development Environment	Softwarové prostředí umožňující vývoj aplikací
IED	Intelligent Electronic Device	Senzorové a kontrolní prvky v elektrických sítích
IoT	Internet of Things	Internet věcí
IoTDM	Internet of Things Data Management	Abstrakční vrstva pro zprostředkování komunikace typu M2M
IP	Internet Protocol	L3 protokol určený pro směrování komunikace
ISIS	Intermediate System to Intermediate System	Směrovací protokol pro použití v rámci AS
ISO	International Organization for Standardization	Mezinárodní organizace pro síťové normy
JAR	Java ARchive	Formát balíčků souborů určený pro jejich distribuci
JDK	Java Development Kit	Kompletní prostředí pro vývoj Java aplikací
JSON	JavaScript Object Notation	Flexibilní formát pro výměnu dat
JVM	Java Virtual Machine	Virtuální stroj umožňující spuštění Java aplikací
LACP	Link Aggregation Control Protocol	Protokol pro logické sdružení fyzických spojů
LAN	Local Area Network	Lokální síť
LISP	Locator/ID Separation Protocol	Směrovací protokol využívající jedno datové pole pro dva účely – směrování a identifikaci
LLDP	Link Layer Discovery Protocol	Protokol pro zjišťování síťových zařízení
LTS	Long Term Support	Vydání s delší dobou podpory (typicky 3 – 5 let)
M2M	Machine-to-Machine	Komunikace pouze mezi autonomními prvky
MAC	Media Access Control	Podvrstva L2 vrstvy zajišťující přístup k fyzickému médiu
MitM	Man-in-the-Middle	Typ útoku využívající zachycení a modifikaci komunikace při přenosu dat
MPLS	Multiprotocol Label Switching	Směrovací mechanismus založený na *návěštích* místo síťových adres
NDP	Neighbor Discovery Protocol	Protokol pro zjišťování síťových prvků IPv6 sítí
NeMo	Network Modeling	Northbound jazyk pro použití logiky záměrů
NETCONF	Network Configuration Protocol	Protokol pro správu síťových zařízení
NFV	Network Functions Virtualization	Koncept virtualizace síťových služeb
OFCTL	–	Nástroj pro manuální správu OpenFlow zařízení
ONAP	Open Networking Automation Platform	Platforma pro kompletní správu a automatizaci síťové infrastruktury
ONF	Open Networking Foundation	Nezisková organizace pro softwarové inovace v PS
ONOS	Open Network Operating System	Pokročilý open source SDN kontrolér
OPNFV	Open Platform for NFV	Skupina nástrojů pro rozvoj sítí poskytovatelů služeb
OS	Operating System	Operační systém

Zkratka	Pojem	Vysvětlení
OSI	Open Systems Interconnection	Referenční sedmivrstvý model funkcionality PS
OSGi	Open Services Gateway initiative	Modulární systém pro dynamické načítání svázaných modulů v jazyce Java
OSPF	Open Shortest Path First	Populární směrovací protokol v rámci AS
OVSDB	Open vSwitch Database	Protokol pro konfiguraci síťových prvků
OXM	OpenFlow Extensible Match	Formát pro flexibilní definici struktur protokolu
OXS	OpenFlow Extensible Stats	Formát pro flexibilní definici struktur statistik
PCEP	Path Computation Element Communication Protocol	Mechanismus zajišťující výpočet směrovacích cest
PDU	Protocol Data Unit	Zpráva na libovolné vrstvě OSI modelu
PIM	Protocol Independent Multicast	Skupina vícecestných směrovacích protokolů
PPP	Point-to-Point Protocol	L2 protokol pro přímou komunikaci dvou zařízení
PS	Počítačová Síť	Skupina zařízení zprostředkující komunikaci
QoS	Quality of Service	Mechanismus prioritizace síťových služeb
RAM	Random Access Memory	Rychlá paměť pro používaná data a spuštěný kód
REST	Representational State Transfer	Architektura pro tvorbu webových služeb
RSTP	Rapid Spanning Tree Protocol	Pokročilejší varianta STP
SCTP	Stream Control Transmission Protocol	L4 protokol se zvýšenou odolností a spolehlivostí
SDN	Software-Defined Networking	Softwarově definovaná síť (sítě)
SFC	Service Function Chaining	Vlastnost umožňující zřetězení několika služeb
SNBI	Secure Network Bootstrapping Interface	Nástroj pro celkovou správu síťové infrastruktury
SNMP	Simple Network Management Protocol	Protokol pro sběr informací a modifikaci nastavení síťových zařízení
STP	Spanning Tree Protocol	L2 protokol zabraňující vzniku síťových smyček
TCAM	Ternary Content Addressable Memory	Specializovaný typ paměti pro rychlé prohledávání
TCP	Transmission Control Protocol	L4 transportní protokol
TL1	Transaction Language 1	Protokol pro správu hlavně optických sítí
TLS	Transport Layer Security	Šifrovací protokol pro zabezpečenou komunikaci
TLV	Type, Length, Value	Formát umožňující dynamickou definici struktur použitých v protokolu OpenFlow
TRILL	TRansparent Interconnection of Lots of Links	Protokol kombinující techniky L2 a L3 pro zabránění vzniku smyček
TS	Tradiční Sítě	Počítačové sítě nepodporující SDN
UDP	User Datagram Protocol	Jednoduchý L4 transportní protokol
UPS	Uninterruptible Power Supply	Elektrické zařízení poskytující souvislou dodávku elektrické energie
USC	Unified Secure Channel	Funkcionalita pro vytvoření zabezpečeného spojení
VTEP	Virtual Tunnel End Point	Entita poskytující (de)enkapsulaci v sítích VXLAN
VLAN	Virtual LAN	L2 technologie umožňující logickou segmentaci broadcast domény
VPN	Virtual Private Network	Rozšíření privátní sítě přes veřejnou síť
VTN	Virtual Tenant Network	Technologie pro virtuální segmentaci sítí
VXLAN	Virtual Extensible LAN	Technologie síťové virtualizace
WAN	Wide Area Network	Geograficky rozlehlá síť



Použité pojmy a symboly

Použité pojmy

Tato publikace používá obecně známé pojmy z klasických počítačových sítí. Jedná se hlavně o:

- **Datagram** – zpráva využívající TCP na transportní (L4) vrstvě ISO/OSI modelu.
- **Enkapsulace** – postupné předávání dat nižším vrstvám v ISO/OSI modelu. Každá vrstva přidává k originální zprávě svá data v podobě záhlaví (a případě zápatí). Opakem je pak deenkapsulace.
- **Hop** – jedná se o jedno ze síťových zařízení na cestě datového toku. Vzdálenost cílového zařízení se může udávat pomocí tzv. hopů, tedy přeskoků určujících, přes kolik síťových prvků je zprávu nutné přenést.
- **ISO/OSI model** – referenční sedmivrstvý model komunikace v počítačových sítích.
- **Mininet** - síťový emulátor umožňující spuštění kompletní síťové topologie včetně SDN.
- **OpenFlow** – protokol zajišťující komunikaci mezi SDN kontrolérem a síťovými prvky, které tento protokol podporují.
- **Paket** – zpráva využívající síťovou (L3) vrstvu ISO/OSI modelu.
- **Rámec** – zpráva využívající linkovou (L2) vrstvu ISO/OSI modelu.
- **Segment** – zpráva využívající UDP na transportní (L4) vrstvě ISO/OSI modelu.
- **Síťový prvek** – označení pro síťová zařízení podporující technologii SDN využívající protokol OpenFlow.
- **Typy vysílání** – vícecestné (*multicast*, zpráva je odeslána všem prvkům, které o ní mají zájem), všesměrové (*broadcast*, zpráva je doručena všem zařízením v daném síťovém segmentu).
- **L1 – L7** – označení pro vrstvy ISO/OSI modelu. Tyto vrstvy jsou postupně: fyzická (1), linková, síťová, transportní, relační, prezentační a aplikační (7).
- **Zpráva** – označení pro obecnou zprávu poslanou v počítačových sítích. Ta může využívat libovolnou vrstvu ISO/OSI modelu. V angličtině se pro tento typ používá pojem PDU (Protocol Data Unit).

Použité symboly

Publikace používá zvýrazněné sekce označené symbolem *i*. Ty poskytují poznámky, důležité informace a odkazy na rozšiřující literaturu.

Příklad zvýrazněné sekce, poskytující poznámky, důležité informace a odkazy na rozšiřující literaturu.

1. Úvod

Internet se stal jednou z nejdůležitějších součástí moderního života. Celosvětově ho v roce 2017 používala více než polovina domácností a 71% mladé populace v rozmezí 15 – 24 let[1]. Internet zastupuje nedílné role ať už v oblasti práce, zábavy, vzdělávání, informovanosti nebo elektronické platby.

Všechny tyto technologie však posouvají nároky kladené na počítačové sítě. Služby přenosu videa ve vysoké kvalitě a připojení bilionů chytrých zařízení sebou nese enormní nároky na přenosovou kapacitu a rychlost. Současně bývají kladeny další požadavky jako je minimální latence, bez výpadková dostupnost a co nejvyšší bezpečnost. Jak se s těmito rostoucími požadavky tradiční počítačové sítě vyrovnávají?

Tradiční počítačové sítě

Současné počítačové sítě používající IP jsou založené na projektu ARPANET (Advanced Research Projects Agency Network) z roku 1969. Tento projekt poprvé představil technologii *přepínání paketů* místo do té doby používaného *vytváření okruhů* známého především z telefonních sítí. Původně spojoval pouze čtyři zařízení, a i když se postupně podstatně rozrostl, stále zůstal pouze jednou sítí. První významná revoluce v počítačových sítí nastala v roce 1973 s příchodem technologie Ethernet, která později vyústila ve vznik internetu.

Internet začal propojovat jednotlivé oddělené sítě, čímž vznikl termín *internetting*[2]. Internet pak přinesl čtyři základní principy: autonomnost, decentralizovanost, doručování bez garance (*best effort*) a bezstavové spojení. V následujících dekádách pak přišlo několik evolučních technologií, ale základní koncept počítačových sítí založených na protokolech TCP/IP zůstal nezměněn.

Počítačové sítě za dobu své existence zažívají konzistentní růst ať už co se týče jejich velikosti, tak požadavků na výkon. Zejména v posledních letech však tyto nároky začaly růst progresivněji než v minulosti. To způsobil příchod několika moderních technologií, z nichž nejvýznamnější vliv má **video** ve vysokém rozlišení (4K, fullHD). Navíc zájem o konzumaci videa se stále zvyšuje. Podle

[1] `https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf`
[2] V. G. Cerf a R. E. Icahn, *A protocol for packet network intercommunication*, 2005

Cisco[3] tvoří video více než polovinu objemu přenesených dat už od roku 2016. V roce 2020 by se však mělo jednat až o 82% celkového objemu přenesených dat[4].

Další technologií s vysokými požadavky na síťové parametry je **Internet věcí** (IoT – Internet of Things). Ten propojuje enormní množství autonomních senzorů a prvků a vytváří tak komunikaci typu M2M (machine-to-machine). I když jednotlivé prvky negenerují velký provoz, díky jejich počtu se celkově jedná o podstatnou část komunikace, která už v roce 2016 předčila provoz generovaný lidmi[5]. V následujících letech je pak očekávaný další růst. Provoz těchto zařízení navíc klade na síť další nároky – zejména na spolehlivost a minimální latenci.

Ostatní typy provozu jako je sdílení dat, zálohování, stahování souborů, web, email nebo online hraní pak přispívají k celkovému růstu spíše pozvolna.

Nevýhody tradičních sítí

Koncept tradičních počítačových sítí je téměř 50 let starý, což sebou nese určitá omezení. Mezi ty největší patří:

- **Decentralizované rozhodování** – každý síťový prvek autonomně rozhoduje o dílčích úkolech jako je přeposílání rámců, směrování paketů nebo filtrování provozu. V tradičních sítích neexistuje centrální prvek spravující všechna zařízení, a proto bývá složité určit aktuální chování sítě. Zejména pokud jsou použity dynamické směrovací protokoly, měnící použitou síťovou topologii.
- **Lokální konfigurace** – základní konfigurace síťových prvků vyžaduje lokální přístup. Tento způsob konfigurace může být dále vyžadován při konfigurační chybě nebo jiným problémům. To sebou nese riziko delšího intervalu provádění případných změn a potenciálně vysoké náklady v případě geograficky rozlehlých sítích.
- **Náchylnost k chybám** – vzhledem k náročnosti konfigurace rozsáhlejších počítačových sítí dochází často ke konfiguračním chybám. Ty mohou způsobit neočekávané chování sítě, od snížení výkonu až po celkovou nedostupnost sítě. Některé typy chyb mohou také zůstat delší dobu neodhaleny a způsobit bezpečnostní problémy.
- **Náročná konfigurace** – problematika správné konfigurace tradičních počítačových sítí je extrémně rozsáhlá a vyžaduje časově náročné vzdělání.
- **Nedeterministická doba oprav** – proces hledání a oprava konfiguračních chyb – tzv. *trouble-shooting*, je v tradičních sítích časově nepředvídatelný. Určité typy chyb navíc mohou vyžadovat lokální konfiguraci (bez možnosti vzdáleného přístupu), což sebou přináší zvýšené náklady a prodlužuje dobu nutnou k uvedení sítě do požadovaného stavu. Tyto vlastnosti jsou zvláště problematické hlavně v produkčních sítích, které často vyžadují minimální (či žádné) odstávky – tzv. *downtime*.
- **Omezená škálovatelnost** – jakékoliv změny v síti vyžadují úpravu statické konfigurace na všech prvcích, kterých se změna týká. To sebou také přináší riziko dočasných výpadků části infrastruktury při přechodu na novou konfiguraci. Tato nedostupnost vzniká nekonzistentním nastavením mezi částmi sítě před provedením změn a částí, kde již proběhla implementace nové konfigurace.
- **Problematické aktualizace síťových prvků** – jakékoliv softwarové aktualizace síťových

[3] `http://techblog.comsoc.org/2017/06/10/cisco-increased-use-of-web-video-to-be-82-of-all-internet-traffic-by-2021/`

[4] `https://contently.com/future-media-marketing-video/`

[5] `https://hostadvice.com/blog/internet-usage-facts-figures/`

prvků jsou plně v režii výrobce těchto prvků. Obvykle tak nebývá zaručena dostupnost nových funkcionalit pro starší prvky, pro které například uplynula doba podpory, případně nesplňují (výrobcem určené) požadavky.

- **Statická / ruční konfigurace** – tradiční sítě neumožňují automaticky měnit konfiguraci v závislosti na dynamických událostech.
- **Závislost na jednom výrobci** – tradiční síťové prvky různých výrobců obvykle obsahují proprietární software, který je typicky nekompatibilní s řešeními ostatních výrobců. Tradiční počítačová síť tak bývá nejčastěji složena kompletně z prvků od jednoho výrobce.

Tyto a další méně závažná omezení komplikují správu tradičních sítí, snižují administrátorský komfort a tím pádem vedou ke zvýšení ceny výsledného řešení. I když většina zmíněných omezení nemusela být zásadních v počítačových sítích minulého století, při rostoucích požadavcích moderních sítí je situace opačná. Jedním z možných řešení jsou softwarově definované sítě, kterým se věnuje následující kapitola.

2. Softwarově definované sítě

2.1 Koncept SDN

Minulá kapitola poukázala na nedostatky tradičních počítačových sítí. Existuje tedy nějaký lepší způsob jak tyto sítě spravovat? V závislosti na cílovém použití, mohou být jedním z řešení softwarově definované sítě. Ty využívají současný trend programovatelnosti. Místo uzavřených síťových prvků, které lze *pouze* konfigurovat pomocí pevně daných příkazů a parametrů, umožňují SDN programovatelnost generických síťových zařízení. V SDN už tak nezáleží na tom, zda daný síťový prvek je přepínač, směrovač nebo firewall. v SDN se nachází generické boxy, které provádí funkcionalitu v závislosti na programu, který je řídí.

SDN tedy fyzicky oddělují datovou a kontrolní vrstvu. Síťové prvky již neprovádí pevně danou funkcionalitu, ale místo toho se chovají podle instrukcí, které jim předává externí entita – tzv. SDN kontrolér. Ten reprezentuje kontrolní vrstvu a je umístěn na centralizovaném zařízení, typicky reprezentovaným klasickým serverem.

2.1.1 Definice SDN

Tradiční definice softwarově definovaných sítí je založena na dvou základních premisách:

1. **Oddělení datové a kontrolní vrstvy** – místo integrace těchto dvou vrstev v každém zařízení, jako je tomu u tradičních sítí, dochází v SDN k jejich separaci. Na síťových zařízeních zůstává pouze datová vrstva, která nedisponuje žádným inteligentním rozhodováním. To je přesunuto do jednoho centrálního prvku – tzv. kontroléru.
2. **Logická centralizace síťového řízení** – o řízení celé sítě rozhoduje právě SDN kontrolér. To je aplikace typicky spuštěná na prakticky libovolném serveru. Kontrolér komunikuje s *hloupými* síťovými zařízeními a předává jim instrukce, podle kterých daná zařízení provádí požadovanou síťovou funkcionalitu.

Tyto dvě vlastnosti jsou jedinými a postačujícími podmínkami pro definování SDN. V různé literatuře a dalších zdrojích lze nalézt případy, kdy za SDN je označena jiná technologie, která těmto podmínkám nevyhovuje. Takové řešení by pak nemělo být pokládáno za *čisté SDN*.

Pojem *čisté SDN* je pak také alternativně spojován s použitím protokolu OpenFlow, který zajišťuje komunikaci mezi datovou a kontrolní vrstvou.

2.1.2 Vlastnosti SDN

Koncept SDN, v podobě oddělení datové a kontrolní vrstvy a centralizovaného řízení, pak eliminuje většinu problémů tradičních počítačových sítí a přináší následující vlastnosti (ty však také mohou mít určité – zde zmíněné – nevýhody):

- **Centralizované řízení** – celá síť je řízená z jednoho centrální prvku (SDN kontroléru). To podstatně zjednodušuje a zefektivňuje správu sítě. Na druhou stranu kontrolér představuje tzv. *single point of failure* – klíčový bod, jehož funkčnost je kritická. Jakékoliv jeho selhání tak může způsobit nefunkčnost celé sítě.
- **Generické síťové prvky** – SDN dokáže efektivně využít síťové prvky obsahující procesory běžné z osobních počítačů a serverů, které mohou být řízeny otevřeným operačním systémem (postaveným nad systémem UNIX). To poskytuje flexibilitu a možné snížení výsledné ceny. V současnosti je však cena těchto prvků paradoxně často vyšší, než cena podobně vybavených zařízení obsahujících specializované obvody a uzavřené operační systémy (a většinou bohužel nepodporující SDN).
- **Hardwarová neutralita** – síť může být založena na otevřených standardech a tedy nezávislá na konkrétním výrobci hardwarových prvků a použitém SDN kontroléru.
- **Programovatelnost** – SDN kontrolér umožňuje implementaci prakticky libovolné funkcionality pouze modifikací softwaru. Aplikace mohou být napsané přímo v kontroléru nebo s ním mohou komunikovat přes jedno z mnoha rozhraní. Konfigurační změny sítě pak mohou být prováděny dynamicky, včetně reakcí na různé události. Klíčovou podmínkou je ale kvalita vytvořeného kódu. Špatně napsaná aplikace může způsobit nestabilitu SDN kontroléru a ohrozit tak funkčnost sítě nebo umožnit případný útok.
- **Rozšiřitelnost** – nová funkcionalita může být do sítě jednoduše přidána pouhou modifikací softwarové aplikace. Změny se pak téměř zároveň aplikují na všech síťových prvcích.
- **Škálovatelnost** – průběžný růst sítě nevyžaduje rozsáhlé konfigurační změny. Nastavení nových hardwarových prvků může proběhnout automaticky. Zatímco rozšíření sítě o nové prvky je tak maximálně zjednodušeno, SDN kontrolér se může stát limitujícím bodem (úzké hrdlo, tzv. *bottleneck*). V takovém případě je jeho výkon pro potřeby rozrůstající se sítě dále nedostačující. Takovou situaci lze řešit použitím výkonějšího serveru pro umístění kontroléru, případně použitím architektury složené z více kontrolérů.
- **Virtualizace** – síťové prvky mohou mít podobu softwarových nástrojů umístěných na virtualizovaných zařízeních. Jejich organizace a umístění je pak z pohledu řízení flexibilní. Příkladem takového řešení může být použití softwarového přepínače *Open vSwitch*[1].

2.1.3 Nevýhody SDN

Uvedené vlastnosti dokazují, že SDN mohou vyřešit většinu omezení tradičních počítačových sítí. Na druhou stranu mají své specifické nevýhody, které je potřeba vzít v potaz. Kromě výše uvedených se pak jedná hlavně o:

- **Bezpečnost** – jako každá nová technologie, ani SDN zatím stále nejsou dostatečně prověřeny

[1] `https://www.openvswitch.org/`

z pohledu bezpečnosti. Softwarová povaha kontroléru spoléhá na jeho řádné zabezpečení (fyzický přístup, operační systém, aktualizované programy). Mezi další zranitelné komponenty patří komunikační protokoly používané v SDN – hlavně mezi síťovými prvky a kontrolérem. Tato komunikace může být šifrována, ale oproti tradičním sítím se stále jedná o jednu z případných možností útoku.

- **Neaktuální dokumentace** – většina open source kontrolérů prochází dynamickým vývojem, se kterým oficiálně vydávaná dokumentace většinou neudržuje krok. V lepším případě je dokumentace dostupná pouze pro starší verze kontrolérů, v horším případě dokumentace téměř chybí.
- **Omezená podpora** – většina technologií použitých v SDN je open source. To sice snižuje náklady na pořízení, na druhou stranu to sebou ale nese riziko chybějící profesionální podpory. Pokud je taková služba vyžadována, je možné zvolit některé z komerčních řešení.
- **Výkon softwarových komponent** – SDN jsou založeny na softwarovém přístupu, který je ze svého principu vždy pomalejší, než hardwarové zpracování známé z tradičních sítí. V závislosti na použitém scénáři se pak tato vlastnost může, ale nemusí projevit. Na druhou stranu, celkový výkon sítě může být v SDN díky efektivnějšímu řízení vyšší.

2.2 Historie SDN

Většina principů, které SDN využívají, není úplně nových. V historii proběhlo několik pokusů o vytvoření podobně programovatelných sítí. První z nich se objevily už téměř před 30 lety.

2.2.1 Cesta k SDN

Nejvýznamnější technologie vedoucí ke vzniku SDN jsou aktivní sítě (*active networking*) a oddělení datové a řídící vrstvy[2].

Aktivní sítě

Koncept aktivních sítí vnikl počátkem 90 let. Jeho hlavním cílem bylo umožnit programovatelnost tradičních počítačových sítí. Motivace byla podobná jako u současných SDN: rychlejší nasazení nových služeb, přidaná hodnota sítě, dynamické reakce na specifické požadavky sítě a prostor pro provádění experimentů. Aktivní sítě se těchto cílů snažily dosáhnout pomocí dvou modelů s odlišnou funkcionalitou. V obou případech bylo umožněno vložení vlastního programového kódu, který umožňoval provádění dodatečné funkcionality:

- **Kapsulový model** – v tomto případě byl programovatelný kód vložen do obsahu jednotlivých zpráv. Koncept byl určen primárně pro koncové uživatele, kteří by si mohli vyvíjet svoje vlastní aplikace.
- **Programovatelný model** – programovatelný kód byl umístěn na síťových prvcích. Tento model byl zaměřen spíše na síťové administrátory, kteří měli oprávnění manipulovat se síťovými prvky.

Aktivní sítě nabízely na svou dobu revoluční vlastnosti, ale trpěly podstatnými, hlavně bezpečnostními, problémy. Zejména kapsulový model byl velice náchylný na útoky typu MitM (Man-in-the-Middle), kde útočník mohl změnit obsah kódu. Technologie také podstatně snižovala výkon sítě a nakonec se ukázalo, že ani zájem uživatelů není nijak velký. Z toho důvodu chyběly opravdu

[2] N. Feamster, J. Rexford, a E. Zegura, *The road to SDN: an intellectual history of programmable networks*, 2014

nezbytné aplikace a koncept tak skončil jako kolekce testovacích projektů financovaných organizací DARPA.

Oddělení datové a řídící vrstvy

Na začátku nového století se začaly objevovat první snahy o oddělení datové a řídící vrstvy. Hlavní motivací byl rapidní nárůst datového provozu, který potřeboval být efektivně spravován. Koncept byl tentokrát zaměřen hlavně na poskytovatele internetu a síťové administrátory. Přes veškeré snahy však skončil jako experimentální technologie v několika průmyslových sítích, kde umožňoval dynamické směrování v závislosti na vytížení sítě, snížení doby výpadků a zvýšení bezpečnosti.

Tento koncept se sice v tehdejší podobě neujal, ale přinesl dvě hlavní inovace, které později přispěly ke vzniku SDN:

- **Otevřené rozhraní mezi datovou a řídící vrstvou** – komunikace mezi těmito vrstvami byla možná použitím nového protokolu ForCES nebo stávajících protokolů jako je BGP.
- **Logicky centralizované řízení** – to je v současných SDN reprezentováno právě v podobě kontroléru.

2.2.2 Vznik SDN

Protokol OpenFlow

Samotný vznik technologie SDN je spojen se vznikem protokolu OpenFlow. Ten byl z počátku synonymem pro SDN. Protokol OpenFlow vznikl v akademickém prostředí univerzit Stanford a California, Bekeley. Tento výzkumný tým pokračoval v práci, kterou v roce 2006 započal PhD student Martin Casado (z univerzity Stanford). Ten vytvořil projekt *Ethane*, jehož cílem bylo zvýšit bezpečnost počítačových sítí použitím centralizovaného prvku, který zajišťoval centralizovanou správu sítě pomocí pravidel aplikovaných na datové toky. Tým práci podstatně rozšířil a v roce 2009 pak zveřejnil jako první verzi protokolu OpenFlow. Ta definovala mechanismus porovnávání shody datových polí (tzv. *match*) na základě údajů obsažených v hlavičkách jednotlivých síťových protokolů.

Protokol OpenFlow se stal okamžitě úspěšným hlavně díky tomu, že se mu podařilo zkombinovat dvě zásadní vlastnosti: programovatelnost a jednoduchou implementaci protokolu. Ta nevyžadovala specializovaný hardware, ale mohla být jednoduše přidána do typických síťových prvků softwarovou aktualizací.

Protokol OpenFlow je pak od roku 2011 pod správou neziskové organizace ONF[3], jejímž cílem je podpora rozvoje SDN pomocí otevřených technologií. Organizace byla založena v roce 2011 za podpory mnoha společností včetně Google, Facebook, Yahoo, Microsoft, Verizon a dalších.

Současné SDN

S rostoucí oblíbeností technologie SDN a zájmem průmyslu, došlo k oddělení pojmů SDN a OpenFlow. SDN už nebyly založeny pouze na protokolu OpenFlow, ale stala se z nich komplexní technologie využívající velké množství odlišných komunikačních protokolů, určených pro různé možnosti použití. Postupně začaly vznikat dedikované aplikace pro řízení SDN – kontroléry. OpenFlow pak zůstal *jen* jeden z možných protokolů pro implementaci SDN – zajišťující komunikaci mezi síťovými prvky a kontroléry.

[3] `https://www.opennetworking.org/`

2.2.3 Nejznámější SDN sítě

SDN se z akademického prostředí a zájmu experimentátorů začaly nejprve používat hlavně v datových centrech a privátních sítích. Dnes se ale hlavně kvůli výhodám v podobě nižší ceny a lepšího využití infrastruktury těší zájmu především velkých poskytovatelů internetu[4]. V blízké budoucnosti je pak očekáváno jejich uplatnění zejména v sítích chytrých měst (*Smart Cities*) a mobilních sítí nové generace – 5G[5].

Velké společnosti jako je Google, Amazon a Microsoft už ale SDN úspěšně používají spousty let.

Azure – Microsoft

Cloudová služba Azure využívá technologii SDN, založenou na vlastním protokolu. Ten však používá podobné principy jako standardní protokol OpenFlow – zejména *flow* tabulky obsahující síťová pravidla. Řešení využívá hierarchickou strukturu kontrolérů. Ty jsou rozděleny do tzv. *clusterů*, a dosahují tak vysoké dostupnosti. Hlavními důvody nasazení SDN v prostředí Azure je automatizace a optimalizace latence a výkonu sítě. Řešení využívají nejoblíbenější Microsoft služby včetně Office365, Skype, OneDrive, Xbox a vyhledávače Bing[6].

B4 WAN – Google

Google používá SDN na své páteřní WAN síti od roku 2011 a jedná se o jednu z největších sítí tohoto typu. Síť propojuje 12 datových center rozmístěných po celém světě a slouží pro zrcadlení uživatelských dat, vyhledávacích metadat a interní aplikace (uživatelé Google služeb jsou připojeni k oddělené síti).

Hlavními výhodami privátní sítě B4 jsou:

- **Centralizovaná správa** – umožňuje mnohem efektivnější a jednodušší práci.
- **Maximalizace využití přenosové kapacity** – technologie díky své dynamičnosti umožňuje bezpečné využití spojů až na téměř 100%. Oproti klasickým sítím, kde jsou spoje typicky projektovány na 30 – 40% vytížení, tak Google dokázal téměř 3x snížit náklady na konektivitu. Spolehlivost tohoto využití je pak zajištěna vícecestným směrováním a faktem, že používané aplikace jsou navrženy tak, aby odolaly dočasným výpadkům nebo snížení přenosové rychlosti.
- **Separace hardware a software** – podstatně zjednodušuje plánované odstávky a aktualizace síťové infrastruktury.

Publikace *B4: Experience with a Globally-Deployed Software Defined WAN*[7] poskytuje užitečné informace o fungování sítě, a jedná se tak prakticky o jediný zdroj ukazující funkcionalitu rozsáhlé SDN v praxi. Vysoká použitelnost SDN je v článku demonstrována srovnáním spolehlivosti s tradičními sítěmi – v pozorovaném období došlo jen k jednomu výpadku sítě, který byl plně srovnatelný s tradičními sítěmi.

[4] `https://www.sdxcentral.com/articles/contributed/how-stay-online-storm-open-source-sdn-solutions-telcos-isps/2017/04/`

[5] R. Guerzoni, R. Trivisonno, a D. Soldani, *SDN-based architecture and procedures for 5G networks*, 2014

[6] `https://searchnetworking.techtarget.com/news/2240215890/Microsofts-Windows-Azure-network-is-a-massive-virtual-SDN`

[7] S. Jain et al., *B4: Experience with a Globally-Deployed Software Defined WAN*, 2013

Bristol is Open

Použitelnost SDN v segmentu chytrých měst je aktuálně testována v projektu *Bristol is Open*[8] nasazeném ve stejnojmenném britském městě. Topologie používá optickou páteřní síť, na kterou jsou přes bezdrátové *mesh* sítě připojeny senzory využívající hlavně lampy pouličního osvětlení. Ty sbírají různá data a zároveň poskytují bezdrátovou konektivitu uživatelům. Samotná SDN je pak založena na vlastním kontroléru *Smart City OS – NetOS* a otevřeném protokolu OpenFlow verze 1.3. Síť umožňuje současný běh různých aplikací, které jsou virtuálně odděleny pro dosažení vysoké bezpečnosti a minimální latence.

EC2 – Amazon

SDN platformy EC2 (Elastic Compute Cloud 2), která je součástí služeb *Amazon Web Services*, je použita pro automatizaci, dynamické vyvažování zátěže, síťovou izolaci a poskytování bezpečnostních zón. Automatizace umožňuje provádět dynamické, ale bezpečné změny v síťové topologii, jako je přidávání a odebírání virtualizovaných serverů apod. Síťová izolace je pak důležitá pro propojení sítí zákazníků s překrývajícími se adresními rozsahy.

2.3 Použití SDN ve specifických případech

SDN jsou technologií s širokými možnostmi uplatnění. Vlastnosti a výhody poskytované SDN, mohou být užitečné prakticky ve všech možných typech počítačových a datových sítí. Následující sekce popisují oblasti různých typů moderních síťových topologií včetně jejich běžných omezení a přínosů nasazení SDN.

2.3.1 Datová centra a cloudová úložiště

Datové centrum je specifické prostředí, ve kterém se nachází velké množství počítačových systémů (serverů). Cílem datových center je poskytovat výpočetní výkon, datové úložiště a s tím související datovou konektivitu. Cloudové úložiště je pak specifický případ datového centra – poskytuje podobné služby, ale typicky přes internet.

Tradiční síťové topologie datových center jsou postaveny na třívrstvém hierarchickém modelu L2 sítí. Tato architektura má však určitá omezení, která začínají být více patrná s rostoucí komplexností datového centra. Mezi největší omezení architektury patří:

- **Velké množství všesměrového provozu** – klasická L2 topologie obsahuje pouze jednu velkou *broadcast* doménu. Veškerý provoz určený pro všechna zařízení je směrován na velké množství zařízení, což negativně ovlivňuje výkon sítě i koncových prvků a zároveň snižuje bezpečnost. Takový provoz je generován například protokoly ARP a DHCP.
- **Velké množství MAC adres** – tradiční síťové přepínače mají omezené kapacity tabulek pro ukládání MAC adres. Ve velkých datových centrech může dojít k jejich zaplnění, pokud je aktivní velké množství koncových zařízení zároveň. To má za následek neschopnost přepínače efektivně směrovat datový provoz, který je místo toho posílán na všechny porty (tzv. *flood*). To opět snižuje výkon sítě a bezpečnost.
- **Omezený maximální počet VLAN** – klasická síťová virtualizace v podobě technologie VLAN umožňuje logické rozdělení na pouze 4096 sítí. To může být v typickém datovém centru omezující faktor, zejména pokud je tato technologie použita pro oddělení jednotlivých zákazníků datového centra, kterých bývá velké množství.

[8] `https://www.bristolisopen.com/wp-content/uploads/2015/11/BIO-Application-SDN-and-NFV.pdf`

- **Vznik smyček v redundantních topologiích** – z důvodu vysoké dostupnosti používají datová centra redundantní síťovou topologii. V případě využití L2 topologie je ale síť spravována L2 protokoly, které obvykle neumožňují efektivní využití paralelních spojů. Mezi tyto L2 protokoly patří STP (Spanning Tree Protocol) a jeho varianty. Alternativní spoje jsou těmito protokoly pouze blokovány, dokud nenastane situace, že jediný aktivní port selže. Toto plýtvání zdroji výrazně prodražuje cenu celé sítě.

Implementace SDN do datových center dokáže odstranit zmíněná omezení a zároveň může oproti tradiční architektuře dosáhnout vyšší efektivity a rychlejšího zotavení se při selhání[9]. Stejně tak velikost MAC tabulek nehraje v SDN roli – místo těchto tabulek se používají *flow* tabulky, které mohou být v případě potřeby implementovány v software a mít tudíž prakticky neomezenou kapacitu. V extrémních případech lze pak využít agregovaná pravidla, sdružující několik datových toků do jednoho pravidla. Místo VLAN lze pak použít vlastní technologii, jako například řešení popsané v článku *Design of SDN-enabled cloud data center*[10], které podporuje až 64 000 kompletně oddělených virtuálních sítí. SDN pak také umožňují vyvažování zátěže mezi redundantními spoji.

2.3.2 Chytrá města

Chytré město lze definovat jako město, jehož infrastruktura dokáže efektivně zpracovávat propojené informace za účelem zlepšení všech aspektů fungování města[11]. Podrobnější definice pak říká, že chytré město integruje soukromé a veřejné zařízení (osobní, rezidenční, komerční, obecní), vybavení, budovy a systémy. Díky této integraci pak občanům poskytuje služby pro zlepšení bezpečnosti, efektivity a pohodlí[12]. K dosažení tohoto cíle jsou použity technologie internetu, IoT a webové služby.

Síťové topologie v chytrých městech nejsou v současnosti pevně definované, a proto se řešení jednotlivých měst liší. Mezi společné problematické oblasti ale patří:

- **Propojení velkého množství senzorů** – ty mohou používat různorodé komunikační protokoly, což vyžaduje specifické řešení pro každý typ komunikace.
- **Bezpečnost** – sítě chytrých měst mohou spadat pod *kritickou infrastrukturu*[13]. V takovém případě je pak bezpodmínečně vyžadováno jejich bezpečné fungování.
- **Důvěrnost dat** – sítě chytrých měst často přenáší citlivá data uživatelů, jako je jejich pozice nebo záběry z bezpečnostních kamer. Různé zákony (například GDPR[14]) mohou vyžadovat specifické zpracování a ukládání dat. Příkladem je takové nakládání s daty, aby byla zajištěna nedohledatelnost konkrétních osob.
- **Kvalita služeb** – velké množství přenášených různorodých dat vyžaduje inteligentní řízení provozu. Zatímco některé typy komunikace musí být doručeny prioritně, jiné nemusí vyžadovat žádné speciální směrování.
- **Spolehlivost** – síť musí být schopna fungovat dlouhodobě bez jakýkoliv výpadků.
- **Škálovatelnost** – budoucí rozvoj technologie chytrých měst není zatím jasně daný, ale lze očekávat její výrazný růst včetně potřeby implementace nových funkcí a protokolů.

[9] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, a C. Wietfeld, *Software-defined networking for Smart Grid communications: Applications, challenges and advantages*, 2014

[10] R. Hwang, H. Tseng, a Y. Tang, *Design of SDN-enabled cloud data center*, 2015

[11] I. Celino a S. Kotoulas, *Smart Cities*, 2013

[12] A. Gharaibeh et al., *Smart Cities: A survey on data management, security and enabling technologies*, 2017

[13] `https://www.zakonyprolidi.cz/cs/2010-432`

[14] `https://eugdpr.org/`

Zmíněné problematické oblasti lze řešit pomocí tradičních sítí, ale nasazení SDN poskytuje výhodu v podobě ucelené platformy, která může vyřešit veškeré problémy jednotně.

2.3.3 Smart grid sítě

Smart grid sítě jsou konceptem integrujícím tradiční elektrické sítě s klasickými datovými sítěmi poskytujícími podpůrné informace. Smart grid sítě lze definovat jako elektrický systém používající obousměrnou zabezpečenou komunikaci a výpočetní inteligenci v rámci celé elektrické infrastruktury (generace, distribuce, spotřeba), za účelem dosažení čistého, bezpečného, spolehlivého, odolného, efektivního a udržitelného systému[15].

Většina smart grid sítí využívá technologie MPLS a OSPF, které mohou být snadno nahrazeny pomocí SDN, což dokazuje článek *Software-defined networking for smart grid communications: Applications, challenges and advantages*[16]. Použití SDN v této oblasti dokáže snížit cenu výsledného řešení, zvýšit výkon a ani doba nasazení nemusí být nijak dlouhá. Experiment popsaný v článku *Using GENI for experimental evaluation of software defined networking in smart grids*[17] prezentoval nasazení takové sítě v horizontu pouhých dvou týdnů.

SDN také může efektivně vyřešit problém vícesměrového vysílání – *multicast*, které se ve smart grid sítích často používá hlavně mezi různými senzory, jako jsou IED (Intelligent Electronic Device). Různá řešení, jako například práce *Recovery from link failures in a smart grid communication network using OpenFlow*[18] dokázala, že SDN mohou ve vícesměrovém vysílání rychle detekovat případný výpadek a v minimálním čase přepnout přenos na záložní trasu, vytvořenou pomocí předpočítaného *multicast* stromu a zároveň nabídnout efektivnější doručování dat.

2.3.4 Internet věcí

Internet věcí (IoT) je koncept sjednocení reálného a digitálního světa propojením velkého množství fyzických prvků pomocí internetu. Tyto prvky pak sbírají různá data o fyzickém světě a mohou být vzdáleně ovládány. Kolektivní propojení prvků umožňuje inteligentní chování celé sítě. Internet věcí je nicméně značně obecný pojem, na který existuje mnoho (často protichůdných) definicí. Ty nejpodstatnější jsou[19]: *IoT je svět, ve kterém fyzické objekty, lidé, ale také virtuální data a fyzické prostředí společně interagují* a *IoT umožňuje libovolným lidem a prvkům komunikovat kdykoliv a kdekoliv, ideálně přes jakoukoliv službu a síť*.

IoT sítě musí zajišťovat propojení enormního množství zařízení, které mají náročné požadavky a používají často rozdílné komunikační protokoly. Tyto sítě jsou proto z pohledu jejich správy velice komplexní, čehož může využít právě centralizovaná technologie jako je SDN. Z nalezených problematických vlastností IoT sítí[19] mohou SDN zlepšit následující funkčnost:

- **Bezpečnost a soukromí** – SDN v této oblasti přináší velkou výhodu softwarových aktualizací, které mohou zabránit nově objeveným hrozbám a útokům.
- **Dostupnost** – distribuované architektura SDN kontrolérů má přehled nad celou sítí a může dynamicky směrovat provoz v závislosti na různých podmínkách. To umožňuje dosáhnout

[15] B. Ge a W.-T. Zhu, *Preserving User Privacy in the Smart Grid by Hiding Appliance Load Characteristics*, 2013

[16] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, a C. Wietfeld, *Software-defined networking for smart grid communications: Applications, challenges and advantages*, 2014

[17] A. Sydney, D. S. Ochs, C. Scoglio, D. Gruenbacher, a R. Miller, *Using GENI for experimental evaluation of software defined networking in smart grids*, 2014

[18] D. Gyllstrom, N. Braga, a J. Kurose, *Recovery from link failures in a smart grid communication network using OpenFlow*, 2014

[19] O. Vermesan et al., *Vision and challenges for realising the internet of things*, 2010

vysoké dostupnosti celé sítě.

- **Dynamická změna funkcionalit** – SDN umožňují rychle a efektivně kompletně změnit funkcionalitu jednotlivých síťových prvků. To podstatně usnadňuje jakékoliv budoucí modifikace sítě.
- **Efektivita** – SDN umožňují implementovat prakticky jakoukoliv síťovou funkcionalitu včetně dynamického směrování na základě využití sítě. V případě nízkého vytížení je tak například možné část infrastruktury dočasně vypnout za účelem snížení spotřeby elektrické energie.
- **Interoperabilita** – technologie SDN je založena na otevřených standardech a umožňuje proto kombinovat síťové prvky různých výrobců. SDN může také zajišťovat abstrakci od odlišných komunikačních protokolů použitých senzorovými prvky internetu věcí.
- **Mobilita** – některé senzory a prvky v IoT sítích mohou dynamicky měnit svou polohu. To od sítě vyžaduje bezvýpadkové přepínání mezi různými přístupovými body a v případě potřeby dočasné ukládání (*cache*) dat. Tyto mechanismy mohou být efektivně implementovány pomocí centralizovaného kontroléru SDN.
- **Spolehlivost** – je definována jako dostupnost v delším časovém období. SDN může opět za určitých podmínek dosáhnout vyšší spolehlivosti než tradiční sítě.
- **Správa** – SDN sítě s centralizovaným pohledem na celou síťovou topologii umožňují flexibilnější a efektivnější správu celé sítě. Ta může být dále usnadněna hromadnou implementací různých síťových politik.
- **Škálovatelnost** – s očekávaným rozvojem IoT sítí bude zřejmě nutné implementovat nové síťové funkcionality. Programovatelnost SDN umožňuje přidávání nových funkcí bez nutnosti vypnutí částí sítě, což u tradičních sítí nebývá možné.
- **Výkon** – SDN dokáží dosáhnout stejného výkonu, jako tradiční počítačové sítě a v určitých případech je i překonat (například v případě nižší latence první zprávy při použití proaktivního vkládání *flow* pravidel). Ve většině případů je ale potřeba zvážit softwarovou povahu kontroléru, který se může stát prvkem omezující celkový výkon.

2.3.5 Další možnosti použití SDN

Největší výhodou SDN je fakt, že díky své programovatelnosti je jejich záběr použití prakticky neomezený. Téměř jakákoliv budoucí síť tak pravděpodobně bude moci být řízená SDN. Pokud by ale přesto nastala situace, kdy SDN již žádány nebudou, obecně není velkým problémem vypnout SDN logiku a přejít zpět na klasické řízení tradičními počítačovými sítěmi.

2.4 Architektura SDN

Základní architektura SDN se skládá ze tří vrstev, které jsou mezi sebou propojeny dvěma rozhraními. Tato architektura je znázorněna na obrázku 2.1. Jednotlivé vrstvy jsou pak zodpovědné především za:

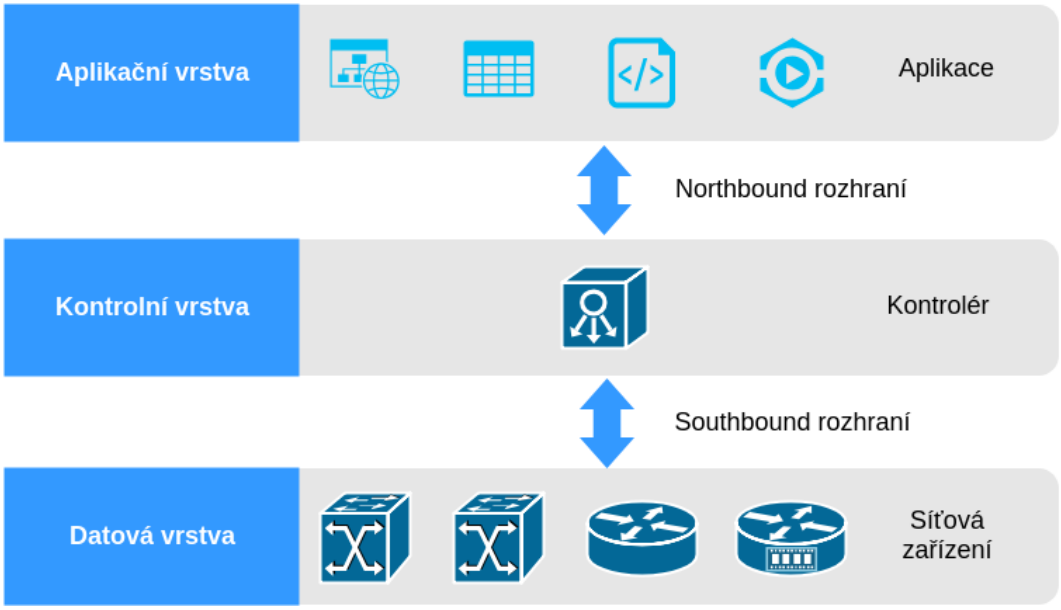
1. **Datová vrstva** – obsahuje síťové prvky podporující SDN. Typicky se jedná o hardwarové přepínače (switch) a směrovače (router), nicméně jejich funkcionalita může být implementována formou softwarové aplikace na virtualizovaném či generickém zařízení.
2. **Kontrolní vrstva** – představuje kontrolér SDN, který řídí zařízení nacházející se na datové vrstvě, poskytuje základní síťové funkcionality a umožňuje integrovat dodatečné aplikace.
3. **Aplikační vrstva** – poskytuje prostor pro programování vlastních aplikací. Ty komunikují

s SDN kontrolérem, který jejich požadavky překládá a předává síťovým prvkům na datové vrstvě.

Zmíněná rozhraní pak zajišťují komunikaci mezi uvedenými vrstvami. Konkrétně se jedná o:

1. **Southbound (jižní) rozhraní** – je stěžejním rozhraním SDN, které zprostředkovává komunikaci mezi síťovými prvky a SDN kontrolérem. Nejčastěji je představováno protokolem OpenFlow (ten býval často chybně zaměňován právě s pojmem SDN).
2. **Northbound (severní) rozhraní** – toto rozhraní je využito pouze v případě, že řešení SDN využívá dodatečnou aplikaci komunikující s kontrolérem. Nejčastěji bývá realizováno pomocí REST, RESTful nebo RESTCONF technologií.

V některé literatuře se pak uvádí další dvě dodatečná rozhraní – *westbound* (západní) a *eastbound* (východní). *Westbound* rozhraní obvykle zodpovídá za komunikaci mezi kontroléry v případě použití distribuované architektury. *Eastbound* rozhraní se pak používá pro komunikaci mezi kontrolérem a tradičními síťovými zařízeními.



Obrázek 2.1: Architektura SDN

2.4.1 Datová vrstva

Datová vrstva obsahuje síťová zařízení podporující SDN. Tato zařízení se mohou označovat jako SDN zařízení, SDN prvky a další (v angličtině často *SDN-enabled*, *forwarding* nebo *data plane devices*).

> Síťová zařízení podporující SDN budou v této publikaci označovány obecně jako **síťové prvky** (fyzicky se může jednat o síťové přepínače, směrovače nebo generické boxy).

Tato podpora je realizována přes libovolný protokol na *southbound* rozhraní. Nejčastěji se na tomto rozhraní používá OpenFlow protokol, který od síťových zařízení vyžaduje podporu určitých vlastností. V závislosti na verzi OpenFlow se jedná zejména o implementaci alespoň jedné datové

struktury – tzv. *flow* tabulky (*flow table*). Síťové zařízení pak dále musí být schopno navázat spojení s SDN kontrolérem. To probíhá přes TCP (nezašifrované), či TLS (zašifrované) protokoly.

Flow tabulka

Flow tabulka je datová struktura nacházející se na síťovém prvku podporujícího SDN. Slouží pro uchování flow pravidel. Ty obsahují pokyny od SDN kontroléru pro provádění požadované funkcionality. Flow tabulka může být realizována softwarově, ale z důvodu vyššího výkonu často využívá specializované hardwarové tabulky tradičních síťových zařízení – typicky TCAM (Ternary Content-Addressable Memory).

Každé flow pravidlo nacházející se v tabulce obsahuje:

- **Pole shod** (*match fields*) – určují, která datová pole v hlavičce zprávy musí odpovídat, aby byla vybrána pro zpracování pravidlem. Protokol OpenFlow v současných verzích podporuje poměrně velké množství různých datových polí, ale pouze některé z nich jsou standardem vyžadované – implementace zbývajících je pouze volitelná. Různí výrobci se pak mohou rozhodnout pro jejich implementaci, či nikoliv. Obecně je podpora volitelných polí mnohem lepší v softwarových flow tabulkách. Realizace procesu porovnávání v hardwarových tabulkách je totiž mnohem náročnější a ne všechny kombinace různých datových polí mohou být efektivně porovnávány současně.
- **Instrukce** (*instructions*) – definují, co se má provést se zprávou, která odpovídá určitému pravidlu. Mezi typické instrukce patří: přeposlat na specifický port (včetně logického portu odkazujícího na SDN kontrolér), zablokovat nebo předat k dalšímu zpracování (další flow tabulka, speciální tabulka, nebo zpracování vrstvou tradičních počítačových sítí).
- **Čítače** (*counters*) – uchovávají různé statistické informace o počtu odpovídajících zpráv, množství přeneseného provozu, doby platnosti pravidla, atd.
- **Priorita** (priority) – každé pravidlo má definovanou prioritu v rozmezí 0 (nejnižší) – 65535 (nejvyšší) [4]. Nejnižší priorita (0) se typicky používá pro generické pravidlo směřující všechny zprávy na SDN kontrolér (tzv. *defaultní* pravidlo nebo také *table-miss*).
- **Časovače** (*timeouts*) – definují dobu platnosti konkrétního pravidla. Jsou definovány dva druhy: první může vypršet bez ohledu na to, zda je, či není používané (*hard-timeout*) a druhé pak pouze v případě, že není používané (*idle-timeout*).
- **Cookies** – je hodnota používaná pouze SDN kontrolérem pro efektivnější rozpoznání konkrétního pravidla. Při zpracování přijatých zpráv na síťovém zařízení se nepoužívá.
- **Přepínače** (*flags*) – ovlivňují podrobné vlastnosti jednotlivých pravidel (například upozornění kontroléru při odstranění pravidla).

Flow tabulka jako každá datová struktura má konečnou velikost. Mnohem více omezující bývá velikost flow tabulek implementovaných v hardwaru. Ty využívají existující specializované tabulky, jejichž velikost je také omezená. Počet flow pravidel, které lze do těchto struktur uložit závisí na jejich komplexnosti a cílové implementaci. Obvykle se však pohybuje mezi 8 000 pravidly pro klasická zařízení, až po 1 000 000 pravidel pro nejvybavenější zařízení určených do datových center.

Síťová zařízení také často kombinují implementace flow tabulek v hardware a software. Například přepínače firmy HP řady 3800 obsahují pouze jednu hardwarovou flow tabulku (ID 100) a čtyři softwarové (ID 200, 201, 202 a 203) [1, str. 47]. Touto kombinací je možné spojit rychlost zpracování (hardwarové tabulky) s vyšší kapacitou a flexibilitou (softwarové tabulky).

Proces zpracování zprávy

Při přijetí nové zprávy na síťovém zařízení je flow tabulka prohledána, což vyústí v jeden ze dvou možných scénářů:

1. **Je nalezena shoda** – zpráva bude zpracována podle odpovídajícího pravidla. To je vybrané na základě nejlepší shody pravidel a případně nejvyšší priority.
2. **Není nalezena shoda** – jedná se o tzv. *table-miss*. V tomto případě je zpráva zahozena a dále se nezpracovává. Z toho důvodu se do flow tabulky typicky vkládá *defaultní* pravidlo s nejnižší prioritou (0), které definuje přeposlání všech zpráv na SDN kontrolér. Přeposlány budou zprávy, které nebyly zachyceny více specifickým pravidlem. Kontrolér pak rozhodne o zpracování takové zprávy a případně může do síťových zařízení vložit nová flow pravidla, která zachytí budoucí zprávy stejného datového toku.

2.4.2 Kontrolní vrstva

Kontrolní vrstva je reprezentována jedním, nebo více SDN kontroléry. Obecně je použit jeden centralizovaný kontrolér, ale z důvodu redundance se často využívá více kontrolérů. Tyto dva přístupy tvoří následující architektury:

- **Centralizovaná** – nejčastěji použitý typ architektury se skládá z jednoho SDN kontroléru. Výhodou je jednoduché nasazení a podpora všech SDN kontrolérů. Architektura však nenabízí zvýšenou odolnost proti výpadkům sítě. Na druhou stranu, poskytovaný výkon kontroléru typicky nepředstavuje problém, protože moderní SDN kontroléry jsou napsány jako vysoce paralelní systémy. Několik samostatných aplikačních vláken je tak schopno simultánně pokrýt aplikační požadavky.
- **Distribuovaná** – v síťové topologii jsou použity dva, nebo více SDN kontrolérů. Hlavní motivací této architektury je zachování plné funkcionality i v případě výpadku jednoho z kontrolérů. Architektura odstraňuje zjevnou potenciální nevýhodu SDN v podobě jednoho centralizovaného bodu, který reprezentuje tzv. *single point of failure* (tedy při jeho selhání dojde k selhání celého systému). Distribuovaná architektura je podporována pouze některými SDN kontroléry (z nejpoužívanějších open source kontrolérů jsou to Open Daylight a ONOS) a vyžaduje také podporu vybraného *southbound* protokolu (OpenFlow protokol ji podporuje od verze 1.2).
  Nevýhodou distribuované architektury je její komplexnost. Její nasazení vyžaduje vyřešení problémů, jako je konzistence nastavení a obrazu sítě prezentovaných v jednotlivých kontrolérech. Ta vyžaduje bezchybné zvládnutí simultánní změny konfigurace. Dalším problémem je rychlá detekce selhání kontroléru a potenciální přepnutí řízení na jeden ze záložních kontrolérů. To bývá problematické zejména v případě, kdy kontrolér zůstane dostupný, ale neprovádí vyžadovanou funkcionalitu (chybná konfigurace, zamrznutí, chyba v aktualizaci). Právě pro tento typ komunikace mezi kontroléry je pak použito *westbound* rozhraní.

2.4.3 Aplikační vrstva

Aplikační vrstva poskytuje prostor pro programování libovolných aplikací využívajících SDN. Jediným funkčním požadavkem je podpora *northbound* rozhraní, která bývá nejčastěji realizovaná pomocí REST protokolu. Výhodou takto naprogramovaných aplikací je jejich nezávislost na architektuře SDN kontroléru: Aplikace mohou proto být naprogramovány v jiném programovacím jazyce než používá samotný kontrolér. V případě, že kontrolér disponuje vyžadovaným rozhraním,

není nutná žádná jeho modifikace. Aplikace pak může teoreticky být přenositelná mezi různými SDN kontroléry. Pokud však rozhraní kontroléru požadovanou funkcionalitou nedisponuje, je nutné provést jeho úpravu modifikací na kontrolní vrstvě.

Alternativou k aplikacím napsaným v aplikační vrstvě jsou aplikace psané přímo v rámci SDN kontroléru. Ty tím pádem fungují na kontrolní vrstvě a pro komunikaci s kontrolérem využívají jeho interní rozhraní – musí být tedy napsané ve stejném programovacím jazyce jako kontrolér samotný. Tyto aplikace se častěji nazývají *moduly* a jejich hlavní výhodou je vyšší rychlost.

2.5 Koncepty spojené s SDN

SDN je v současnosti velice populární téma a spousta konceptů a technologií je s SDN spojováno, nebo přímo označováno, byť se podle definice o SDN nejedná.

> SDN mají jasně danou definici, která byla popsána v kapitole 2.1: oddělují datovou a kontrolní vrstvu a poskytují logickou centralizaci síťového řízení.

Definici SDN však odpovídá více konceptů. Příkladem může být řešení od firmy Cisco – SD-WAN, které umožňuje *pouhé* definování konfigurace zařízení v grafickém prostředí a následné nahrání konfigurace do síťových prvků. Toto řešení sice naplňuje definici SDN, ale neposkytuje veškeré vlastnosti uvedené v kapitole 2.1.2.

> V této publikaci bude za SDN považována pouze technologie odpovídající definici v kapitole 2.1, která je navíc založena na protokolu OpenFlow.

Mezi další koncepty spojené s SDN pak patří hybridní SDN, síťová virtualizace funkcí (NFV) a plně programovatelná datová vrstva. Tyto technologie budou pro úplnost stručně popsány v následujících podkapitolách.

2.5.1 Hybridní SDN

Za hybridní SDN mohou být považovány následující dva typy přístupů:

- Využití OpenFlow funkce *NORMAL*.
- Kombinace tradičních sítí a SDN.

OpenFlow funkce NORMAL

OpenFlow funkce NORMAL je součástí protokolu od první verze standardu – 1.0. Síťový prvek funguje v režimu SDN – tedy pro směrování je využita OpenFlow tabulka s tím, že jsou použita OpenFlow pravidla s instrukcí typu NORMAL. Ta říká, že daný datový tok bude směrován podle mechanismu tradičního síťování. Pravidla s instrukcí NORMAL bývají nejčastěji vložena právě ve flow tabulce (může jich být jedno, nebo více), ale samotná instrukce může být provedena i reaktivně na SDN kontroléru pomocí zprávy typu *packet-out*.

Jedná se o volitelnou funkci, která je většinou podporována pouze u určitých hardwarových prvků (například HP 3800, Aruba 3810M a další). Aby mohla být funkce implementována, zařízení musí být schopné autonomního provozu bez SDN. Konkrétní konfigurace takového zařízení je pak klíčová pro správné fungování sítě.

Kombinací SDN a tradičního síťování v tomto případě lze například využít pokročilé protokoly zabraňující vzniku smyček (RSTP, TRILL), které jinak nebývají v SDN kontrolérech implementované.

Kombinace tradičních sítí a SDN

V tomto případě jsou v jedné administrativní doméně použity jak tradiční počítačové sítě, tak sítě SDN. V publikaci *Opportunities and research challenges of hybrid software defined networks*[20] byl tento koncept rozdělen do čtyř odlišných modelů, znázorněných na obrázku 2.2:

1. **Model založený na topologii** – síť je rozdělena do oblastí, které jsou spravovány buď tradičními sítěmi (na obrázku reprezentované zkratkou TS), nebo pomocí SDN. Největším úskalím tohoto modelu je zajištění komunikace mezi jednotlivými oblastmi.
2. **Model založený na službách** – jednotlivá síťová zařízení mohou implementovat funkcionality tradičních sítí i SDN. Každý ze způsobů je zodpovědný za určité služby. Typicky základní přepínání a směrování je ponecháno tradičním sítím, zatímco pokročilejší služby (které nemusí být vyžadovány na všech zařízeních) obsluhuje SDN.
3. **Model založený na třídách** – každé ze síťových zařízení implementuje logiku tradičních sítí i SDN pro obsluhu odlišných typů datového provozu.
4. **Integrovaný model** – SDN je použito jako hlavní technologie pro obsluhu celé topologie, ale ke své funkci využívá protokoly tradičních počítačových sítí. Ty jsou využity jako *southbound* rozhraní a díky tomu je možné SDN nasadit i na sítích nepodporujících protokol OpenFlow. Typické protokoly pro toto použití jsou BGP a MPLS.

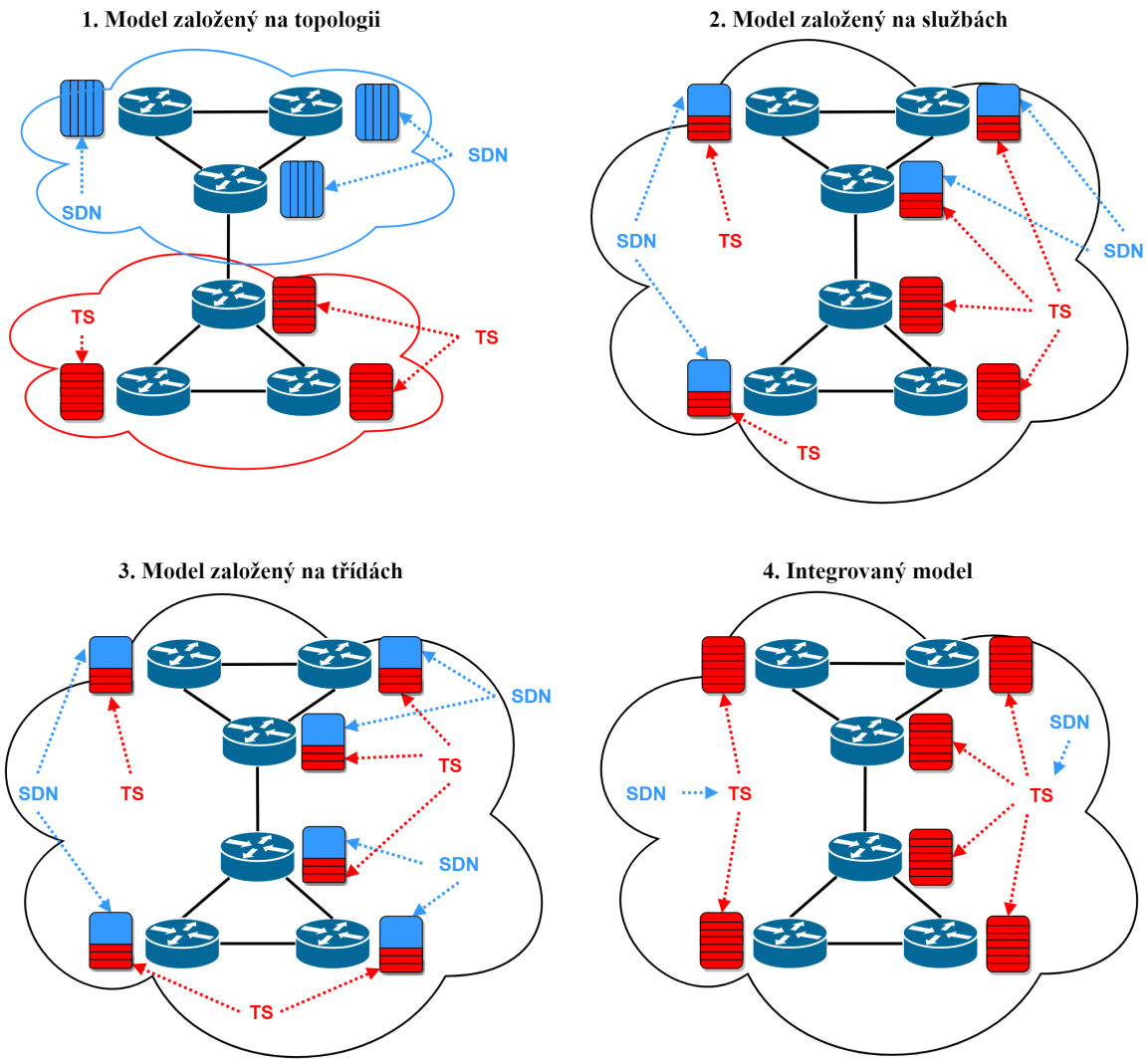
2.5.2 NFV – Network Functions Virtualization

NFV je technologie umožňující virtualizaci síťových služeb (protokolů, funkcí) a síťových zařízení (přepínačů, směrovačů, datových úložišť). Ke své funkcionalitě technologie používá známé koncepty z cloudových služeb: hardwarovou virtualizaci (hypervisor), síťovou virtualizaci (softwarové přepínače) a pokročilé funkce správy zařízení (automatické zálohování, tvorbu snapshotů, škálovatelnost). Vše je založené na otevřených protokolech umožňující integraci funkcí.

Hlavním cílem NFV je přesun síťových funkcí ze specializovaných (tradičních) zařízení na generické prvky. Ty pak mohou být snadno virtualizované a poskytují tak další výhody známé z datových center. Jednou z hlavních výhod je snížení celkové ceny řešení. To je promítnuto nižší cenou za infrastrukturu (generické prvky jsou při stejném výkonu levnější než specializovaná zařízení) a nižší spotřebou elektrické energie (virtualizace umožňuje dynamické škálování prostředků v závislosti na aktuálně používaném výkonu). Mezi další výhody pak patří dynamická škálovatelnost a snazší úpravy infrastruktury. Stejně jako v případě SDN, NFV umožňuje pokročilé funkce pro správu sítě – automatickou instalaci, škálovatelnost, znovupoužitelnost (předpřipravené virtuální obrazy zařízení) a pokročilou analýzu datového provozu. To vše pak přináší vyšší spolehlivost a výkon sítě.

NFV má oproti SDN potenciálně mnohem širší záběr uplatnitelnosti (SDN pouze odděluje datovou a kontrolní vrstvu a poskytuje logickou centralizaci). NFV často používá SDN jako jednu z implementačních technologií. Dalším rozdílem mezi technologiemi je jejich cílová síťová vrstva – zatímco SDN nejčastěji využívá vrstvy 2–4, NFV typicky pracuje na 7. vrstvě (ale může být nasazeno i na nižších vrstvách).

[20] S. Vissicchio, L. Vanbever, a O. Bonaventure, *Opportunities and research challenges of hybrid software defined networks*, 2014



Obrázek 2.2: Modely kombinací tradičních sítí a SDN

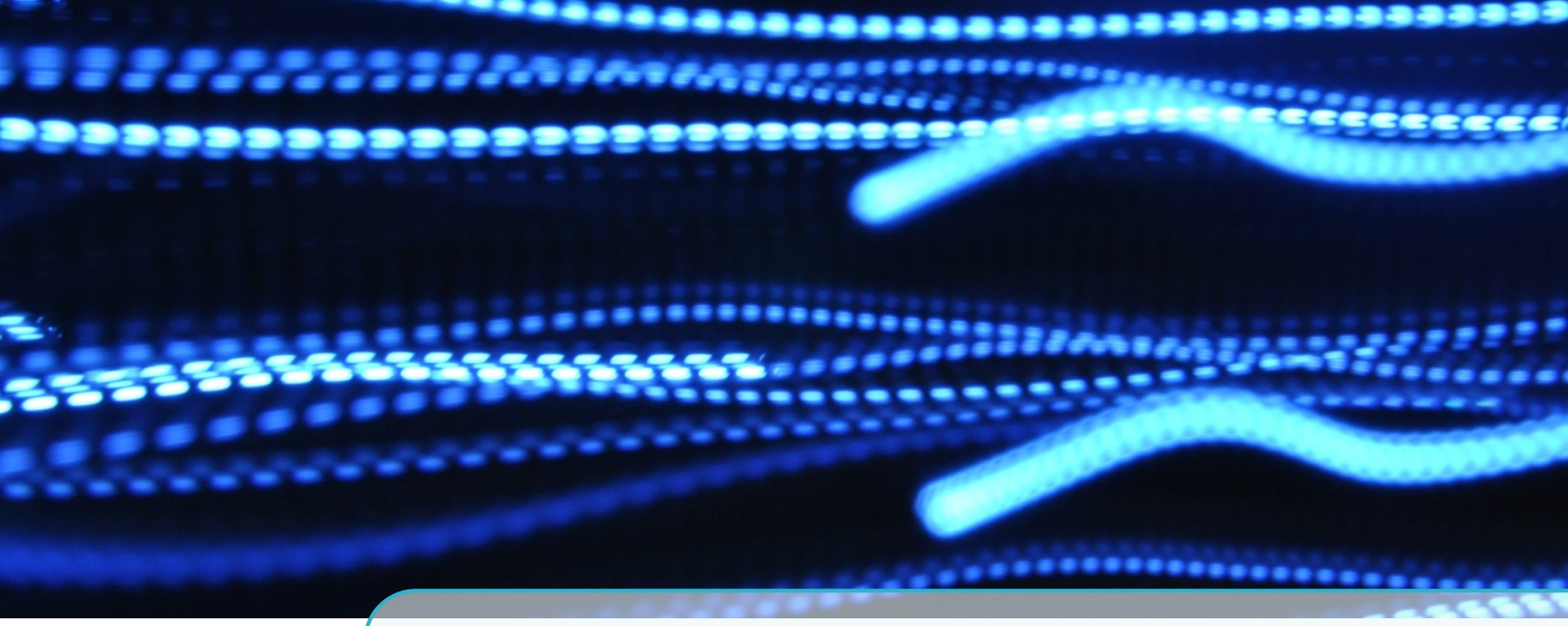
2.5.3 Plně programovatelná datová vrstva

Jedním z nejnovějších konceptů je plně programovatelná datová vrstva. Na rozdíl od protokolu OpenFlow, který pevně definuje položky, které mohou být porovnávány a upravovány, tento koncept umožňuje naprostou flexibilitu v prováděných akcích. Pro porovnávání zpráv tak mohou být použita jakákoliv data a stejně flexibilní jsou i prováděné akce a sběr statistik. Hlavně z tohoto důvodu bývá tento koncept (nesprávně) označován jako OpenFlow 2.0. Hlavními výhodami jsou tedy: flexibilita, nezávislost na použitém hardware a výhody vyšších programovacích jazyků. Ty poskytují pokročilé programovací algoritmy, automatickou kontrolu kódu, abstrakci, softwarové knihovny, debuggovací nástroje a další možnosti.

Koncept je založen na zvýšení výkonu dnešních generických čipů, které se vyrovnají specializovaným čipům tradičních síťových prvků nebo je předčí.

Příkladem implementace konceptu je jazyk *P4*[21]. Ten je podporován například v kontroléru ONOS a objevují se první snahy o jeho implementaci v softwarovém přepínači Open vSwitch.

[21] `https://p4.org/`



3. OpenFlow protokol

3.1 Představení

OpenFlow protokol je nejdůležitějším protokolem pro komunikaci mezi SDN kontrolérem a síťovými prvky. V architektuře SDN se tedy nachází na *southbound* rozhraní. V minulosti byl často zaměňován právě s pojmem SDN – byť se ve skutečnosti jedná pouze o jeden z možných protokolů pro implementaci SDN. Dnes bývá také někdy označován jako jediný protokol umožňující *čisté* SDN. Toto označení vychází z faktu, že OpenFlow protokol stál za vznikem SDN a jeho vlastnosti umožňují implementaci nejflexibilnější funkcionality.

3.2 Architektura protokolu

Architektura OpenFlow se skládá ze tří částí zobrazených na obrázku 3.1:

- Implementace na síťovém prvku.
- Implementace na SDN kontroléru.
- Komunikačního protokolu.

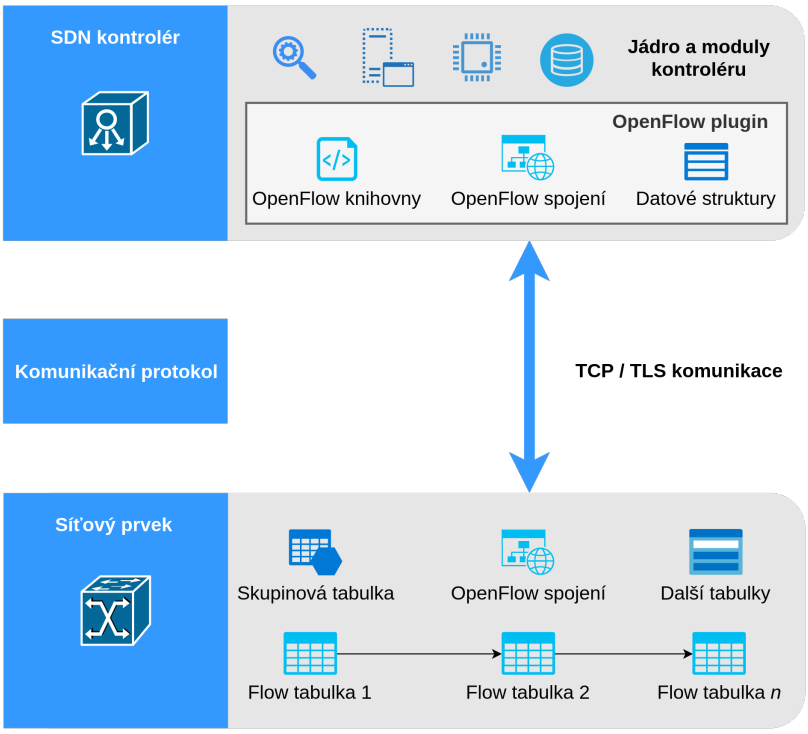
3.2.1 Implementace na síťovém prvku

Síťový prvek podporující OpenFlow protokol musí implementovat minimálně jednu jeho verzi se všemi vyžadovanými parametry. Volitelné parametry této verze pak mohou, ale nemusí být implementovány.

Mezi nejdůležitější datové struktury a funkcionality implementované na síťovém prvku patří: flow tabulka, skupinová tabulka a mechanismus pro navázání a udržování OpenFlow spojení.

Flow tabulka

Hlavní datovou strukturou, kterou OpenFlow používá pro svou funkcionalitu je tzv. flow tabulka. První verze protokolu měla pouze jednu flow tabulku, ale další verze přidaly podporu více flow tabulek. V typických hardwarových prvcích pak tyto tabulky využívají specializované hardwarové čipy (ASIC, TCAM), které dosahují vysokého výkonu při zpracovávání zpráv. Při směrování tak lze



Obrázek 3.1: Architektura protokolu OpenFlow

dosáhnout výkonu a latence na úrovni odpovídající tradičním počítačovým sítím. Nevýhodou těchto struktur je jejich omezená kapacita. Z tohoto důvodu bývá počet tabulek využívajících tyto čipy omezen a další flow tabulky jsou implementovány pomocí software a umístěny v RAM. Provádění jejich operací je pak značně pomalejší. V takovém případě je vhodné zvážit, do které z tabulek danou funkcionalitu umístit.

Obsah flow tabulek ve formě tzv. flow pravidel je typicky spravován SDN kontrolérem (může být ovládán i manuálně, případně pomocí vlastního programu). Flow pravidla definují primárně pole pro rozpoznání odpovídajících zpráv a akce, které budou s těmito zprávami provedeny. V tabulce by typicky mělo být vloženo alespoň jedno tzv. defaultní (výchozí) flow pravidlo. To musí mít nejnižší prioritu, ale být co nejvíce obecné – aby zachytilo všechny zprávy, pro které nebylo nalezeno žádné jiné odpovídající pravidlo. K tomuto účelu se používá prázdné pole pro rozpoznávání zpráv – to automaticky znamená shodu s libovolnými daty.

Defaultní pravidlo pak obsahuje akci *přeposlání na kontrolér*. Kontrolér na základě přijaté zprávy může rozhodnout o vložení nového flow pravidla, které dokáže zachytit a zpracovat zprávy ze stejného datového toku. V takovém případě jsou pak všechny následující odpovídající zprávy zpracovávány přímo síťovým prvkem bez nutnosti konzultovat kontrolér (tento mechanismus využívá reaktivní logiku).

Vytvořením flow pravidla s akcí přesunutí zprávy do jiné flow tabulky je pak možné zřetězit jednotlivé tabulky do tzv. *pipeline*.

Skupinová tabulka

Skupinová tabulka (*group table*) obsahuje místo klasických OpenFlow pravidel tzv. skupinové záznamy (*group entries*). Každý takový záznam pak může obsahovat listy akcí (*action buckets*),

které umožňují provést několik funkcí najednou. Tyto tabulky se většinou využívají pro specifické přeposílání zpráv (všesměrové, vícesměrové, vyvažování zátěže, záloha). Tyto typy tabulek jsou podporovány od verze protokolu 1.1.

OpenFlow spojení

Síťový prvek musí také implementovat logiku umožňující navázání a udržování OpenFlow spojení s kontrolérem. K tomuto účelu slouží modul *OpenFlow channel*. Ten může být jeden, ale od verze protokolu 1.3 existuje možnost využití více modulů. Tím je možné dosáhnout vyššího výkonu při odesílání a příjmu OpenFlow zpráv. Mechanismus totiž využívá paralelní architektury vnitřních komponent síťových prvků.

3.2.2 Implementace na SDN kontroléru

Stejně jako síťový prvek, i SDN kontrolér podporující OpenFlow, musí implementovat veškeré povinné vlastnosti alespoň jedné verze protokolu. Na rozdíl od síťových prvků je však implementace na kontroléru kompletně v softwaru a tato část tedy nevyužívá žádné specializované datové struktury využívající specifické hardwarové řešení. Tím pádem nebývá implementace funkcí protokolu tak náročná a obecně se dá říci, že SDN kontroléry podporují většinu volitelných funkcí protokolu. Rozhodně větší množství než bývá implementováno na hardwarových síťových prvcích.

> Oficiální specifikace OpenFlow protokolu [4, str. 200] obsahuje kompletní definici hlavičkové struktury s veškerými strukturami protokolu, které by kontrolér měl implementovat.

Kontrolér musí zajistit navázání a udržování spojení, zpracování OpenFlow zpráv a poskytovat rozhraní pro obsluhu. Nejkomplikovanější částí je zajištění zpracování OpenFlow zpráv. To musí být dostatečně robustní a efektivní, protože na něm z velké části závisí výkon celého kontroléru a potenciálně i celé sítě. Moderní SDN kontroléry proto využívají paralelní vlákna, frontové zpracování událostí a efektivní datové struktury a algoritmy.

Pro zvýšení spolehlivosti celé sítě lze pak využít distribuovanou architekturu SDN kontrolérů, která umožňuje zvýšení výkonu, distribuci zátěže a převzetí funkcionality v případě selhání. Toto řešení však vyžaduje podporu mezi použitými kontroléry.

3.2.3 Komunikační protokol

Poslední částí architektury OpenFlow protokolu je komunikační protokol mezi SDN kontrolérem a síťovými prvky. Ten je založen na TCP v případě nezašifrovaného spojení, nebo TLS v případě šifrovaného spojení. To bylo definováno už v první verzi protokolu a mělo by být použito v každé produkční (ne-experimentální) síti i tehdy, pokud samotné spojení mezi SDN kontrolérem a síťovými prvky využívá dedikované privátní spoje (*out-of-band*).

Komunikační protokol definuje tři typy OpenFlow zpráv o pevně dané struktuře – tyto zprávy jsou detailně popsány v kapitole 3.4. Protokol také umožňuje přenos enkapsulovaných zpráv. Ty jsou, nezávisle na použité vrstvě ISO/OSI modelu, kompletně zabaleny do užitečných dat TCP, nebo TLS datagramu a poslány mezi síťovým prvkem a kontrolérem (a naopak).

3.3 Verze protokolu

Specifikace protokolu OpenFlow definovala postupně několik verzí. Každá novější specifikace uvádí veškeré vlastnosti předchozích verzí a pro přehled všech funkcionalit je proto nejlepší použít tu

nejnovější veřejně dostupnou – aktuálně je to verze 1.5.1[1].

Veškeré verze specifikace jsou dostupné na stránkách organizace *Open Networking Foundation*:
`https://www.opennetworking.org/software-defined-standards/specifications/`

Specifikace definuje dva typy funkcí – povinné (*required*) a volitelné (*optional*). Pouze povinné funkce musí být implementovány pro dodržení souladu s danou verzí specifikace. Implementace volitelných funkcí je pak plně v režii daného výrobce prvku či komunity vyvíjející programový kód.

3.3.1 OpenFlow 1.0

První verze protokolu je označována jako *0x01* a pochází z roku 2009. Poskytuje pouze základní funkcionalitu, ale i přes to se stále jedná o používanou verzi, zejména z důvodu její nejširší podpory mezi různými zařízeními a kontroléry.

Verze poprvé definovala tři typy zpráv (podrobně popsaných v kapitole 3.4) a pojem flow tabulka (*flow table*) včetně její struktury skládající se z jednotlivých flow pravidel (*flow rules*). Ty určují funkcionalitu síťového prvku. Tyto pravidla v klasických SDN spravuje kontrolér.

Každé flow pravidlo se skládá ze struktury obsahující: pole shod, akce, čítače. První verze protokolu pak definuje následující vlastnosti těchto atributů:

- **Pole shod** – určují, která pole hlaviček zpráv lze použít pro porovnávání shody. Pokud nejsou uvedeny, jako shoda bude vyhodnocena libovolná hodnota. Některá pole podporují formát masky. Povinná pole první verze OpenFlow jsou: příchozí port, zdrojová a cílová MAC adresa, typ rámce, VLAN ID, VLAN priorita, IP zdrojová a cílová adresa, IP protokol, IP ToS pole, TCP nebo UDP zdrojové a cílové porty.
- **Akce** – definují, jak bude zpracována zpráva, která odpovídá definovaným pravidlům. Každé pravidlo může obsahovat *0–n* akcí. V případě, že akce není definována, je odpovídající zpráva vždy zahozena. Pokud je definováno více akcí, jsou vždy provedeny v uvedeném pořadí (pokud jsou poslány na stejný port, zpracování na tomto portu už závisí na konkrétní implementaci). Akce se dělí na povinné a volitelné. **Povinné** akce jsou:
  - ✓ Přeposlání (*Forward*) – na fyzické porty a následující definované logické porty:
    - *All* – přepošle na všechny porty, kromě příchozího.
    - *Controller* – přepošle na připojený SDN kontrolér.
    - *Local* – přepošle na tradiční vrstvu přepínače (ignoruje SDN).
    - *Table* – provede akci ve flow tabulce (dostupné pouze pro zprávy typu *packet-out*).
    - *In_port* – přepošle zpět na příchozí port.
  - ✓ Zahození (*Drop*) – stejně jako v případě prázdné akce dojde k zahození zprávy.

  Volitelné akce pak jsou:
  - ✓ Přeposlání (*Forward*) – definuje další dva typy přeposlání: *normal* (pro přeposlání se využije tradiční vrstva počítačových sítí – L2/L3) a *flood* (přepošle na všechny aktivní porty v rámci *spanning-tree* topologie, kromě příchozího portu).
  - ✓ Zařazení do fronty (*Enqueue*) – zpráva je předána frontě asociované s konkrétním portem. Umožňuje aplikaci mechanismu QoS (Quality of Service).

[1] Specifikace protokolu OpenFlow verze 1.5.1 [4] je dostupná na adrese:
`https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf`

  - ✓ Modifikace hlavičky zprávy (*Modify-Field*) – umožňuje úpravu podporovaných polí hlaviček zprávy. Mezi podporovaná pole patří: VLAN ID, VLAN priorita, odstranění VLAN informace, zdrojové a cílové Ethernet / IP adresy, ToS, zdrojové a cílové TCP / UDP porty.

- **Čítače** – definují několik typů statistik (pro tabulku, pro port a pro frontu). Implementace konkrétního mechanismu sběru statistik závisí na výrobci zařízení a může být hardwarová, softwarová nebo jejich kombinace. Podporované statistiky jsou:
  - ✓ Pro tabulku – počet aktivních záznamů, počet vyhledání a počet shod odpovídajících zpráv.
  - ✓ Pro datový tok – počet přijatých paketů, množství přijatých bytů a doba trvání pravidla (v sekundách a nanosekundách).
  - ✓ Pro port – počet přijatých / odeslaných paketů a bytů, počet přijatých a odeslaných paketů určených k zahození, počet chybně přijatých a odeslaných paketů (včetně statistik pro přijaté chyby typu *frame alignment*, *overrun* a *CRC*) a počet kolizí.
  - ✓ Pro frontu – počet odeslaných paketů, bytů a počet chyb typu *overrun*.

3.3.2 OpenFlow 1.1

Druhá verze používá označení *0x02* a byla vydána počátkem roku 2011. Přinesla hlavně podporu více flow tabulek, skupinových portů, mechanismy při ztrátě konektivity s SDN kontrolérem a přidala několik nových polí pro porovnávání shod. Tyto změny sebou však přinesly podstatné zvýšení velikosti porovnávané struktury z 40 na 88 bytů.

- **Více flow tabulek** – umožňují zřetězení zpracování příchozích zpráv, lepší využití stávajících prostředků síťového prvku (využití specializovaných hardwarových tabulek) a logické oddělení prováděných síťových funkcí (ACL, QoS, směrování). Každá příchozí zpráva je nejprve zpracována podle pravidel v první flow tabulce. Ta může obsahovat pravidla typu *goto*, předávající odpovídající zprávu ke zpracování na další definovanou tabulku.
- **Skupiny (groups)** – umožňují logické sloučení několika fyzických portů do jedné skupiny. Tato vlastnost se hodí zejména pro specifické způsoby přeposílání zpráv – vícesměrového a různé typy všesměrového vysílání. K logicky uskupeným portům jsou pak přiřazeny skupiny akcí – tzv. *buckets*. Podporovány jsou následující typy skupin:
  - ✓ Všechny (*All*) – povinná vlastnost provede všechny akce v dané logické skupině. Typické použití je pro vícesměrové a všesměrové vysílání.
  - ✓ Nepřímé (*Indirect*) – povinná vlastnost provede pouze první skupinu akcí. V případě, že je definována právě jedna skupina, funkcionalita je stejná jako u typu *všechny*.
  - ✓ Vybrané (*Select*) – volitelná vlastnost provede pouze jednu skupinu akcí vybranou na základě implementovaného algoritmu výběru. Ten ve specifikaci OpenFlow není definován, ale obvykle se používá jednoduchý *round robin*. Tento typ se používá hlavně pro vyvažování zátěže (*load-balancing*) a vícecestné (*multipath*) směrování.
  - ✓ Rychlé přepnutí (*Fast failover*) – volitelná vlastnost provede pouze první skupinu, která má všechny přiřazené porty aktivní (ve stavu *up*). Typ zajišťuje zejména vysokou dostupnost. V případě selhání portu může být okamžitě k dispozici záložní trasa, bez nutnosti komunikace s kontrolérem.

- **Mechanismy při ztrátě konektivity s kontrolérem** – v případě výpadku konektivity jsou definovány dva módy chování síťového prvku:
  - ✓ Zabezpečený (*Fail-secure*) – z flow tabulky zařízení je odstraněno pouze defaultní pravidlo (určující přeposlání zprávy na kontrolér). Ostatní pravidla jsou nadále používána, nicméně v závislosti na nastavení jejich časovačů mohou expirovat.
  - ✓ Samostatný (*Fail-standalone*) – z flow tabulky zařízení jsou odstraněna veškerá pravidla a zařízení je přepnuto do módu tradičních počítačových sítí (ignoruje SDN). Zařízení se bude chovat podle tradiční konfigurace. V případě využití tohoto módu je proto nutné provést standardní konfiguraci síťového prvku, která bude použita právě v případě výpadku kontroléru.
- **Další změny** – rozšíření podporovaných polí pro porovnávání shody o SCTP zdrojový a cílový port, MPLS pole (label a traffic control), volitelná 64-bitů dlouhá metadata a rozšíření velikosti portů, které nyní mohou mít velikost až 32 bitů (oproti dřívějším 16).

3.3.3 OpenFlow 1.2

Třetí verze používá označení *0x03* a byla vydána koncem roku 2011. Přidala hlavně podporu nových polí pro porovnávání shod včetně IPv6, podporu distribuované architektury SDN kontrolérů a změnu rozpoznávacího mechanismu pravidel na dynamickou strukturu – OXM (OpenFlow Extensible Match). Ta umožňuje flexibilní přidávání nových polí shod bez neustálého zvyšování velikosti porovnávané struktury. To je důležité hlavně u nově podporovaných polí, které mají větší velikost – hlavně IPv6 adresy. Díky tomuto mechanismu mohlo být také přidáno pole pro testovací využití – *experimenter field*. Současně s touto změnou došlo k definici tzv. *prerekvizit*, zajišťujících konzistenci porovnávaných pravidel. Například porovnávání cílové IP adresy tak vyžaduje současné porovnání typu L3 struktury (*eth_type*), která musí odpovídat hodnotě *0x0800*.

- **Nová pole shod** – protokol nyní umožňuje rozpoznávání zpráv i na základě ARP adresy, IPv4 ECN pole a následujících IPv6 polí: zdrojové a cílové adresy, *flow label*, DSCP, ECN, čísla protokolu, ICMPv6 typu a kódu a ICMPv6 Neighbor Discovery zdrojové a cílové MAC adresy.
- **Dynamická struktura pravidel** – místo statické struktury definované v předchozích verzích protokolu je nyní použita OXM struktura o proměnné velikosti 5–529 bytů. Konkrétní velikost rozpoznávaného pravidla je definována ve formátu TLV (*type-length-value*).
- **Distribuovaná architektura SDN kontrolérů** – místo jednoho centralizovaného SDN kontroléru je nyní možné využít distribuovanou architekturu složenou z více kontrolérů. Jednotlivé síťové prvky se připojují ke všem kontrolérům, které mohou být nastaveny do jedné z následujících rolí:
  - ✓ Ekvivalentní (*equal*) – výchozí role, ve které má kontrolér plná přístupová práva na síťové prvky. Pokud je tento režim použit, všechny kontroléry v architektuře musí být nastaveny právě do tohoto režimu a všechny prvky jsou pak připojeny ke všem kontrolérům.
  - ✓ Podřízený (*slave*) – kontrolér má pouze práva pro čtení z připojených prvků, od kterých automaticky nedostává žádné zprávy kromě zpráv typu *port-status*. Kontrolér se pomocí dotazovacích zpráv může informovat na stav zařízení.
  - ✓ Hlavní (*master*) – v tomto režimu má kontrolér plná práva jako v případě ekvivalentního

režimu, ale v rámci jedné architektury může být v tomto režimu nastaven pouze jediný kontrolér (ostatní musí být v podřízeném režimu).

3.3.4 OpenFlow 1.3

Čtvrtá verze používá označení *0x04* a byla vydána v roce 2012. V současnosti se stále jedná o nejrozšířenější verzi podporovanou většinou hardwarových prvků. Tato verze přinesla velkou spoustu, zejména drobnějších úprav. Mezi ty patří:

- **Rozšíření podpory IPv6** – od této verze je možné rozpoznávat shodu na základě rozšířené hlavičky (*IPv6 extension header*). Ta je umístěna mezi klasickou hlavičku IPv6 a data z vyšších vrstev protokolů. Podporovaná pole rozšířené hlavičky pro porovnávání jsou: *hop-by-hop, router, fragmentation, destination option, authentication, encapsulating security payload, no next header, out of preferred order* a *unexpected*.
- **Měření a kontrola datových toků** – implementace tzv. *meter* tabulek umožňuje měření množství zpracovávaných zpráv a následnou implementaci QoS. Tato funkcionalita je dostupná pro jednotlivé datové toky (určené flow pravidly).
- **Paralelní spoje** – umožňují síťovému prvku navázat několik paralelních TCP, UDP nebo DTLS spojení na stejný SDN kontrolér z důvodu vyššího výkonu. Toho je dosaženo využitím paralelní architektury těchto zařízení. Paralelní spojení jsou použita pro zprávy typu *packet-in* a *packet-out*.
- **Rozšíření statistik** – mezi měřenými statistikami je nyní čítač informující o době trvání jednotlivého údaje (například aktivní doba flow pravidla / aktivního portu, atd.).
- **Výchozí flow pravidlo** – v případě, že přijatá zpráva neodpovídá žádnému pravidlu ve flow tabulce, dojde k použití nově definovaného výchozího pravidla (*table-miss*). To může být nastaveno na zahození zprávy (výchozí hodnota), přeposlání na kontrolér, nebo předání další flow tabulce.
- **Další změny** – nová pole pro porovnávání zpráv (MPLS BoS, tunnel-ID) a *cookies* v OpenFlow zprávách pro jejich rychlejší identifikaci.

3.3.5 OpenFlow 1.4

Pátá verze používá označení *0x05* a byla vydána v roce 2013. Tato verze přinesla zajímavé pokročilé funkce, ale nejdůležitější změnou je použití nového TCP portu pro komunikaci mezi SDN kontrolérem a síťovými prvky. Tento port, rezervovaný organizací IANA[2], je 6653 a specifikace doporučuje, aby starší porty (6633 a 976) nebyly dále používány. Tato změna je podstatná i v případě použití emulátoru Mininet, který ve verzi 2.2.1 ve výchozím stavu používá stále TCP port 6633[3].

Mezi další podstatné změny patří:

- **Lepší rozšiřitelnost** – flexibilní dynamická struktura OXM, poprvé použita v protokolu verze 1.2 pro porovnávání zpráv, je v této verzi použita i pro další struktury protokolu, jako jsou chybové zprávy, porty, tabulky, fronty, instrukce, akce a další části.

[2]IANA (Internet Assigned Numbers Authority) je organizace spravující hlavní DNS záznamy a přidělující IP adresy, TCP a UDP porty a zajišťující další internetové funkce. Více informací lze nalézt na oficiálních stránkách: `https://www.iana.org/`

[3]To mimo jiné znamená, že v případě použití dané verze emulátoru s výchozími hodnotami a kontrolérem Floodlight (který používá nové číslo portu 6653) nedojde k navázání spojení. V tomto případě je nutné upravit nastavení emulátoru, případně Floodlight kontroléru v souboru *floodlightdefault.properties* (položka *openFlowPort*.)

- **Pokročilý real-time monitoring** – umožňuje kontroléru monitorování změn ve flow tabulkách, skupinách a frontách zařízení v reálném čase. Toho je využito zejména v distribuované architektuře, kde ostatní kontroléry mohou provádět změny v nastavení jednotlivých prvků. Ty jsou pak okamžitě zaznamenány jiným kontrolérem.
- **Automatické mazání flow pravidel** – každé flow pravidlo nyní obsahuje atribut důležitosti (*importance*). Pravidla s nižší důležitostí pak mohou být automaticky odstraněna v případě zaplnění flow tabulky. Výhodou tohoto řešení je, že je provedeno autonomně síťovým prvkem bez nutnosti kontaktování kontroléru.
- **Transakční zpracování OpenFlow zpráv** – jednotlivé zprávy mohou být sdruženy do logických svazků, které se pak provedou jako jedna transakce. V případě selhání provedení některé ze zpráv pak nedojde k provedení transakce.
- **Paralelní flow tabulky** – síťový prvek nyní může obsahovat paralelní flow tabulky, které mají stejný obsah, který je průběžně automaticky synchronizován. Tento mechanismus umožňuje využít paralelní architekturu moderních síťových zařízení pro dosažení lepšího výkonu ve vyhledávacích operacích.
- **Další změny** – dále byl upraven popis zdůvodňující typ *packet-in* zprávy, byly přidány nové chybové kódy a přidána podpora optických portů.

3.3.6 OpenFlow 1.5

Oficiální nejnovější verze používá označení *0x06* a byla vydána v roce 2014. Její aktuální revize má pak označení 1.5.1 a pochází z roku 2015. Tato verze přinesla velké mnnožství drobnějších změn, mezi které patří:

- **Odchozí flow tabulky (egress)** – umožňují dodatečné zpracování zpráv při jejich umístění na odchozí port. S jedním portem může být svázáno více odchozích tabulek.
- **Vylepšení statistik** – veškeré datové struktury používané statistikami byly převedeny do flexibilního formátu OXS (OpenFlow eXtensible Statistics), který je kompatibilní s formátem OXM a používá stejnou TLV strukturu. Kromě již podporovaných funkcí (doba trvání flow pravidla, počet flow pravidel, počet paketů a bytů) byl také přidán čítač *flow idle time* (určující poslední použití pravidla). Dalším vylepšením je volitelná možnost zaslání statistik pouze při dosažení definovaného limitu. Tím je možné snížit režii jinak nutně způsobenou periodickým dotazováním a sběrem statistik.
- **Pokročilá podpora TCP** – flow pravidlo nyní může zachytit i vnořené tagy používané TCP – SYN, ACK a FIN. Touto vlastností je tak možné implementovat funkcionalitu stavového firewallu, udržujícího informaci o stavu spojení.
- **Plánované provedení OpenFlow zpráv** – transakční zpracování OpenFlow zpráv z verze 1.4 bylo rozšířeno o jejich možné provedení v pevně definovaný čas. Součástí tohoto vylepšení je také možnost zjistit podporované vlastnosti zařízení – zda podporuje transakční zpracování, plánované zpracování a zpracování v pevně určeném pořadí.
- **Podpora ne-Ethernetových rámců** – díky implementaci formátu OXM je nyní možné provádět porovnávání rámců typu PPP (Point-to-Point Protocol).
- **Další změny** – dále byla přidána podpora monitorování stavu kontrolérů, nová akce *copy-field* (umožňující zkopírování jednoho pole hlavičky zprávy do jiné zprávy), nový typ skupinových portů (*selective*), přidání metadat a masky do akce *set-field*, zvětšení velikosti experimentálního OXM na 64-bitů a sjednocení formátu dotazovacích zpráv na skupiny, porty a fronty.

3.3.7 OpenFlow 1.6

Tato verze je v současnosti dostupná pouze pro členy organizace *Open Networking Foundation*[4] a zatím nebyla vydána pro veřejnost.

3.4 Zprávy protokolu

OpenFlow protokol od první verze definuje tři druhy zpráv:

- **Jednosměrné** – zprávy jsou posílány kontrolérem na síťový prvek.
- **Asynchronní** – zprávy jsou posílány síťovým prvkem na kontrolér.
- **Symetrické** – zprávy mohou být posílány v obou směrech (inicializované kontrolérem nebo síťovým prvkem).

Jednotlivé typy OpenFlow zpráv jsou popsány v oficiální specifikaci, případně přehledně na stránkách: `http://flowgrammable.org/sdn/openflow/message-layer/`

Následující výpis obsahuje typy zpráv podle seznamu uvedeného ve specifikaci OpenFlow 1.5.1.[4, str. 38]. Tento seznam uvádí obecné názvy typů zpráv (ne názvy konkrétních zpráv implementovaných protokolem).

3.4.1 Jednosměrné

Tyto zprávy vytváří kontrolér a od síťového prvku může být (ale nemusí) vyžadována odpověď. Mezi tyto zprávy patří:

- **Features** – jedná se o dotaz na podporované funkce prvku. Ten musí odpovědět zprávou *features reply*, ve které jsou uvedené podporované funkce zařízení. Jak již bylo uvedeno, zařízení nemusí podporovat veškeré volitelné funkce OpenFlow protokolu, a také může mít nestandardní implementaci některých volitelných vlastností (včetně například čísel flow tabulek). Tato zpráva je posílána při prvotním navázání spojení mezi prvky (ihned po úspěšném navázání TCP / TLS spojení).
  Potvrzující zpráva *features reply* obsahuje identifikaci síťového prvku (*datapath-id*), maximální počet zpráv síťové komunikace, které mohou být zařazeny do fronty pro odeslání na kontrolér (zprávy typu *packet-in*), počet flow tabulek, podporované vlastnosti (statistiky, STP, fronty, ARP), podporované akce a výpis fyzických portů.
- **Configuration** – zpráva umožňuje dotázání se na nastavení konkrétního prvku a případné nastavení vlastností spojených s fragmentací přeposílaných zpráv síťové komunikace. Zařízení pak musí zaslat odpověď pouze v případě dotazovací zprávy.
- **Modify-state** – jsou hlavní typy zpráv umožňující přidání, smazání nebo modifikaci flow pravidla, skupinového pravidla, skupinové akce nebo nastavení vlastností portu. Zpráva obsahuje: definici polí pro rozpoznání zprávy síťové komunikace (*match*), *cookie*, příkaz (přidání, modifikace, smazání), časovače (*idle_timeout*, *hard_timeout*), prioritu, *buffer_id*, odchozí port, přepínače a akce. Zpráva není nijak potvrzována.
- **Read-state** – zpráva umožňuje získat z prvku různé informace o nastavení, podporovaných funkcích a statistikách.
- **Packet-out** – jeden z nejdůležitějších typů umožňuje zaslat zprávu síťové komunikace na zařízení, které ji potom zpracuje podle definované akce. Tímto typem zprávy se nejčastěji

[4] `https://www.opennetworking.org/membership-info/`

vrací zpráva síťové komunikace, pro kterou nebylo na zařízení nalezeno flow pravidlo (a bylo tak použito pravidlo přeposlání na kontrolér pomocí zprávy typu *packet-in*). *Packet-out* zpráva obsahuje: *buffer_id*, příchozí port, definované akce a samotná data původní zprávy síťové komunikace. Zpráva není nijak potvrzována.

- **Barrier** – tyto zprávy typu žádost / odpověď se používají pro potvrzování provedení požadovaných funkcí a dokončených operací. Potvrzující odpověď určuje synchronizační body, které definují provedení všech předchozích zpráv.
- **Role-request** – používá se při distribuované architektuře s více kontroléry, kdy kontrolér může nastavit nebo zjistit svou roli (ekvivalentní, podřízený, hlavní). Zpráva vyžaduje odpověď.
- **Asynchronous-configuration** – používá se opět v distribuované architektuře, typicky při navazování OpenFlow spojení. Pomocí těchto zpráv kontrolér určí, jaké typy asynchronních zpráv vyžaduje od zařízení dostávat, případně se dotáže na aktuální nastavení. Nastavení probíhá na základě stavových kódů událostí: *packet-in*, *port-status* a *flow-removed*. Dotazovací zpráva vyžaduje odpověď, nastavovací zpráva odpověď nevyžaduje.

3.4.2 Asynchronní

Asynchronní zprávy jsou zasílány síťovým prvkem na kontrolér. Typicky se jedná o upozornění na vzniklé události různých typů – chybové zprávy, přijetí zprávy síťové komunikace z nového datového toku, atd. Asynchronní zprávy nevyžadují žádné potvrzení. Nejedná se tedy o zprávy typu odpověď nebo potvrzení na *jednosměrné* zprávy od kontroléru. Mezi tyto zprávy patří:

- **Packet-in** – nejdůležitější typ asynchronní zprávy, který přeposílá přijatou zprávu síťové komunikace na kontrolér. V závislosti na nastavení flow tabulky se typicky jedná o zprávu, která neodpovídá žádnému pravidlu (*table-miss*), odpovídá defaultnímu pravidlu (s akcí odeslání na kontrolér), případně vlastnímu pravidlu (se stejnou akcí). Původní zpráva síťové komunikace je kompletně zabalena do nového datagramu OpenFlow protokolu, který je odeslán na kontrolér.
  Přeposílání kompletních zpráv síťové komunikace pro zpracování na kontrolér typicky představuje největší zátěž a může negativně ovlivnit výkon datové sítě. Pro urychlení zpracování je možné využít *bufferování* zpráv síťové komunikace na síťovém prvku. V tom případě je na kontrolér odeslána pouze část zprávy síťové komunikace (ve výchozím nastavení prvních 128 bytů). Zpráva síťové komunikace uložená na síťovém prvku je pak typicky zpracována na základě vrácené OpenFlow zprávy typu *packet-out*, případně *flow-mod*.
- **Flow-removed** – zpráva informuje kontrolér o odstranění flow pravidla. To může nastat z důvodu vypršení časovačů (*idle, hard*), na základě žádosti od kontroléru, nebo z jiných důvodů (odstranění asociované skupiny, nebo automatické odstranění funkcí *eviction*). Aby tato zpráva byla poslána, dané flow pravidlo musí mít nastaven příznak *OFPFF_SEND_FLOW_REM*.

  > Ve výchozím stavu není příznak *OFPFF_SEND_FLOW_ REM* pro potvrzení odstranění pravidla nastaven a zařízení v případě odstranění takového pravidla nijak neinformuje!

  Zpráva obsahuje pole shod *match*, *cookie*, prioritu, důvod a statistiky spojené s daným pravidlem (doba trvání, časovače a počet paketů a bytů).
- **Port-status** – zpráva informuje kontrolér o změně stavu daného portu. To typicky nastává v případě (de)aktivování portu, případně selhání spoje.
- **Role-status** – informuje kontrolér v případě změny jeho role. To nastává nejčastěji v případě zvolení nového hlavního (*master*) kontroléru, kdy proběhne informování původního hlavního

kontroléru.

- **Controller-status** – informuje kontrolér v případě změny OpenFlow spojení. Tento typ zprávy se používá nejčastěji při detekci ztráty konektivity mezi kontroléry v distribuované architektuře.
- **Flow-monitor** – informuje kontrolér o změně ve flow tabulce. O jaký typ změny se jedná závisí na konfiguraci kontrolérem.

3.4.3 Symetrické

Tyto typy zpráv mohou být vytvořeny jak SDN kontrolérem, tak síťovým prvkem a slouží hlavně pro zasílání periodických a chybových zpráv. Mezi podporované typy zpráv patří:

- **Hello** – tento typ zprávy je použit při navazování spojení mezi kontrolérem a síťovými prvky a slouží hlavně pro rozpoznání a dohodnutí o podporované verzi protokolu. Zpráva proto obsahuje pole s nejvyšší podporovanou verzí protokolu daného zařízení. V případě selhání tohoto mechanismu je vygenerována chybová zpráva typu *OFPET_HELLO_FAILED* s kódem *OFPHFC_INCOMPATIBLE*.
- **Echo** – obsahuje dva typy zpráv – dotaz (*request*) a odpověď (*reply*). Dotazovací zprávy vždy vyžadují odpověď. Zprávy slouží hlavně pro udržování aktivního spojení a případnou detekci výpadku. Mohou být také použity pro měření latence a přenosové rychlosti. V tom případě jsou v těle samotné zprávy (pole *data*) umístěna dodatečná data (typicky *časové razítko*). Specifikace nijak neurčuje strukturu, ani velikost těchto dat. Zpráva typu odpověď však musí tyto data poslat zpět nezměněné.
- **Error** – tyto zprávy informují druhou stranu o vzniklém problému. Nejčastěji se jedná o zprávy odeslané síťovým prvkem, informující kontrolér o nepodporované funkci, kterou se kontrolér pokoušel použít. Chybové zprávy mají své číselné kódy, podle kterých je lze rychle dohledat. Jedná se o obecný typ (*OFP_ERROR_TYPE*) a kód konkrétní chyby, jehož název se liší podle typu zprávy.

  Seznam všech chybových typů a kódů začíná v oficiální dokumentaci OpenFlow 1.5.1 [4] na straně 184.

- **Experimenter** – tento typ zpráv je rezervován pro budoucí použití a testování nových funkcionalit.

3.5 Mechanismus porovnávání shod datových polí

Mechanismus porovnávání shod datových polí určuje, jakým způsobem mají být hodnoty v hlavičkách zprávy porovnávány s hodnotami zadanými v polích shod (*match fields*). Protokol OpenFlow podporuje vybraná pole z hlaviček zpráv napříč ISO/OSI modelem. Samotné rozpoznávání shody pak probíhá dvěma základními způsoby:

- **Přesná shoda** – zadává se pouze porovnávaná hodnota datového pole. Tento způsob je podporován všemi poli.
- **Bitová shoda** – kromě samotné hodnoty datového pole definuje také masku (nejedná se o síťovou masku!). Ta má stejnou velikost jako porovnávané datové pole. Hodnoty 1 v masce určují, které bity v datovém poli budou porovnávány a hodnoty 0 určují bity, které budou ignorovány. Pokud maska není definována, jsou porovnávány všechny bity (ekvivalent k masce složené ze samých jedniček). Některé typy datových polí mají speciální formát masky. To je i případ IP adres, u kterých má maska formát síťové masky (například *255.255.255.0* u IPv4 adresy).

U některých datových polí mohou být podporovány následující rozšiřující způsoby porovnávání:

- **Definovaná shoda** – je určena skupinou několika hodnot. Příkladem může být několik TCP portů.
- **Shoda rozsahu** – je definována dvěma hodnotami ohraničujícími interval odpovídajících hodnot. Příklad u protokolu TCP by tak mohl být rozsah portů *100–200* definovaný: $100 \leq tcp\_dst \leq 200$.
- **Shoda nerovnosti** – určuje hodnotu, která se nesmí rovnat pro vyhodnocení zachycení zprávy. Všechny ostatní zprávy, jejichž hodnota specifikovaného pole bude odlišná, budou zachyceny.
- **Konjunktivní shoda** – speciální typ shody, která umožňuje kombinovat několik předchozích typů pomocí operátoru *nebo* (*OR*). Jedná se o rozšíření standardu OpenFlow a tento způsob tedy nemusí být podporován na všech zařízeních.

3.6 Funkcionalita OpenFlow protokolu

Tato podkapitola shrnuje nejdůležitější vlastnosti protokolu OpenFlow používané v SDN. Je určena zejména pro rychlý přehled podporovaných funkcí při vývoji vlastních SDN aplikací a vychází z normy OpenFlow protokolu verze 1.5.1 [4].

3.6.1 Správa flow pravidel

Obsah flow tabulek síťových prvků – jednotlivá flow pravidla – lze spravovat několika způsoby. Nejčastějším řešením je aplikací implementovanou v SDN kontroléru. Flow pravidla lze také spravovat ručně pomocí různých nástrojů emulujících zasílání OpenFlow zpráv. Síťové prvky obvykle také nabízí možnost vypsání obsahu flow tabulky přímo z jejich konfiguračního prostředí (CLI).

Zprávy modifikující flow pravidla

Typ zprávy umožňující modifikaci flow pravidel se označuje jako *OFP_FLOW_MOD* a spadá pod zprávy typu *modify-state*. Zpráva definuje pět druhů možných akcí (*command*):

- **Add** – umožňuje přidat nové flow pravidlo. Pokud již flow tabulka obsahuje stejné pravidlo se stejnou prioritou, dojde k jeho přepsání. Statistiky asociované s původním pravidlem pak budou zkopírovány do nového pravidla (pokud nové pravidlo nemá aktivní přepínač *OFPFF_RESET_COUNTS*). U nového pravidla lze volitelně nastavit přepínač, který nejprve ověří, zda se ve flow tabulce nenachází pravidlo, jehož porovnávané hodnoty by se překrývaly s novým pravidlem (*OFPFF_CHECK_OVERLAP*). V případě, že ano, dojde k vygenerování chybové zprávy (*OFPFMFC_OVERLAP*) a odmítnutí nového pravidla.
- **Modify** – umožňuje modifikovat všechna odpovídající pravidla pouze na základě shody *match*. Tímto způsobem je možné modifikovat akce definované v pravidlu. Ostatní vlastnosti (*cookie*, časovače, hodnota *importance*, přepínače a statistiky) zůstávají stejné. Statistiky lze volitelně vynulovat nastavením přepínače *OFPFF_RESET_COUNTS*. V případě, že se v tabulce nenachází žádné odpovídající pravidlo, nedojde k žádné akci ani k vygenerování chybové zprávy!
- **Modify_strict** – umožňuje modifikovat pouze pravidla přesně odpovídající definici shody včetně masky a priority.
- **Delete** – umožňuje smazání všech odpovídajících pravidel na základě shody. Pokud má dané pravidlo nastavený příznak *OFPFF_SEND_FLOW_REM*, zařízení vygeneruje zprávu typu

flow-removed. V opačném případě dojde pouze ke smazání pravidla. Pokud se v tabulce nenachází žádné odpovídající pravidlo, stejně jako v případě modifikace nedojde k žádné akci ani k žádnému upozornění. Všechna pravidla z flow tabulky lze odstranit speciálním příznakem *OFPTT_ALL*.

- **Delete_strict** – umožňuje smazání pouze pravidel přesně odpovídajících definici shody včetně masky a priority.

Pro snadnější definici porovnávaných pravidel lze kromě polí shod a priorit použít i hodnotu *cookie* (nesmí být 0) včetně asociované masky.

Automatické mazání flow pravidel

Každá flow tabulka má limitovanou velikost a z tohoto důvodu je vhodné zajistit mechanismus odstraňování nepotřebných OpenFlow pravidel. Existují tři způsoby odstraňování flow pravidel, z nichž jeden je manuální (pomocí *flow-modification* zpráv s akcí *delete*) a dva automatické:

- **Časová expirace** – Každé flow pravidlo obsahuje dva typy čítačů (*timeout*): *idle* a *hard*. Ty jsou ve výchozím stavu nastaveny na 0, která reprezentuje nekonečno. Taková pravidla pak nejsou automaticky odstraněná nikdy. Tento přístup se hodí hlavně pro výchozí (defaultní) pravidlo směrující neznámé zprávy na kontrolér. Nenulová hodnota těchto čítačů pak definuje počet sekund, po kterých dojde k odstranění pravidla. S tím, že *idle* čítač je obnoven vždy v případě, kdy je pravidlo použito. Pokud je tedy pravidlo používáno, nedojde k jeho odstranění. *Hard* čítač naopak nebere ohled na používání pravidla a po jeho vypršení dojde k vymazání pravidla nezávisle na tom, zda bylo, nebo nebylo používané. Pokud jsou definovány oba druhy čítačů, ke smazání pravidla dojde při expiraci toho prvního.
- **Mechanismus eviction** (definovaný v OpenFlow 1.4) – v případě, že je tento volitelný mechanismus zapnutý (ve výchozím stavu je vypnutý), jsou automaticky mazána flow pravidla s nižší důležitostí. Mechanismus je aktivován při dosažení určité hranice zaplnění kapacity flow tabulky. Konkrétní algoritmus výběru pravidel je závislý na implementaci na konkrétním prvku a ve specifikaci protokolu není uveden.

3.6.2 Způsoby vkládání flow pravidel

SDN kontrolér může použít dva odlišné způsoby vkládání flow pravidel: reaktivní a proaktivní.

Reaktivní vkládání flow pravidel

Reaktivní vkládání flow pravidel (*reactive*) je výchozím a standardním způsobem fungování SDN. V tomto módu je při navázání spojení mezi kontrolérem a síťovým prvkem vloženo defaultní (výchozí) flow pravidlo. To má nejnižší prioritu (0), shodu polí nastavenou na libovolnou (prázdný parametr *match*) a akci přeposlání na kontrolér. V případě přijetí zprávy z nového datového toku je tak nalezena shoda pouze s tímto pravidlem a síťový prvek vygeneruje OpenFlow zprávu typu *packet-in*. Ta vytvoří nový TCP segment, jehož data obsahují zabalenou původní zprávu (případně pouze její část, pokud je použita funkce *bufferování*). *Packet-in* zpráva je pak doručena na kontrolér, který je zodpovědný za její zpracování v závislosti na jeho konfiguraci a použité aplikaci.

Kontrolér typicky rozhodne o dalším směrování zprávy – v hlavičce původní *packet-in* zprávy upraví akci (kam má zařízení zprávu poslat) a vytvoří z ní novou OpenFlow zprávu typu *packet-out*. Ta je pak zaslána zpět na síťový prvek, který zprávu rozbalí a původní zprávu síťové komunikace nacházející se v datech OpenFlow zprávy, zpracuje podle definované akce.

Tímto způsobem by bylo možné obsluhovat veškerou síťovou funkcionalitu, nicméně takovéto zpracovaní všech zpráv kontrolérem by bylo značně pomalé. Veškeré zpracování by totiž probíhalo

v softwarové vrstvě a navíc by vyžadovalo minimálně dva datové přenosy mezi každým síťovým prvkem a kontrolérem. Z toho důvodu kontrolér typicky ještě před vytvořením *packet-out* zprávy vytvoří nové flow pravidlo. To instruuje síťový prvek o tom, jak má zpracovávat zprávy ze stejného datového toku (například na základě stejných MAC / IP adres). Toto pravidlo pak zašle zařízení v OpenFlow zprávě typu *flow-modification* (spadající pod zprávy *modify-state*). Veškeré další zprávy z daného toku pak budou zachyceny tímto pravidlem a budou zpracovány přímo síťovým prvkem bez nutnosti opětovného dotazování kontroléru.

Tento způsob správy flow pravidel sebou přináší vyšší latenci v případě přijetí nového datového toku. První zprávy tohoto toku budou doručeny o něco později z důvodu nutného provedení popsaných činností (vytvoření *packet-in* zprávy, doručení na kontrolér, zpracování kontrolérem, vytvoření *packet-out* zprávy, vytvoření flow pravidla). Největší časové zdržení typicky představuje SDN kontrolér, jehož vytížení roste s velikostí sítě a množstvím datového provozu.

Naopak výhodou tohoto způsobu je efektivita využití flow tabulek. Do nich se vkládají pouze pravidla, která jsou v síti skutečně používaná. Pravidla, která po nějakou dobu používaná nejsou, mohou být automaticky odstraněna definováním čítače *idle_timeout*. Pokud bude v budoucnosti přijata zpráva z datového toku, pro který byla pravidla již odstraněna, zpráva bude opět zachycena defaultním pravidlem a zpracována jako nový datový tok.

Proaktivní vkládání flow pravidel

Proaktivní vkládání flow pravidel (*proactive*) se snaží odstranit zvýšenou latenci prvních zpráv v novém datovém toku při použití reaktivního vkládání. Za tímto účelem je použito prediktivní vkládání zatím nepoužitých flow pravidel. SDN kontrolér (respektive jeho modul nebo aplikace) musí implementovat logiku, která dokáže určit, jaká pravidla budou používána. Vzhledem k tomu, že kapacity flow tabulek síťových prvků mají omezenou velikost, je přesnost této predikce klíčová.

Tento způsob správy flow tabulek lze uplatnit pouze v sítích, u kterých je možné s dostatečnou přesností odhadnout podobu jednotlivých datových toků. Mezi takové sítě patří například sítě s jedinou možnou cestou ven – sítě typu *stub*.

Pro lepší efektivitu funkcionality lze využít pokročilé funkce OpenFlow protokolu, jako je automatické odstraňování flow pravidel pomocí mechanismu *eviction*.

Poznámka k uvedené latenci

Praktické měření hodnoty latence bylo provedeno v práci *Using SDN to Enhance IoT Security* [2, str. 128]. Naměřená hodnota je silně ovlivněna velikostí sítě, použitými zařízeními a zvoleným SDN kontrolérem, ale řádově se jedná o jednotky až desítky ms (v práci uvedeno 17 ms). S tím, že následující zprávy mají tuto latenci typicky 10x–100x nižší (v práci uvedeno 0,5 ms).

3.6.3 Virtuální porty

OpenFlow specifikace definuje několik typů speciálních virtuálních portů. Ty mohou být použity pro specifické směrování zpráv například na kontrolér, příchozí port nebo na více portů. Jejich kompletní výpis včetně označení a čísel je uveden v tabulce 3.1.

3.6.4 Podporovaná pole shod

Seznam veškerých podporovaných polí pro porovnávání zpráv (*match fields*) pro verzi protokolu 1.5.1 je uveden v tabulce 3.2. Tabulka uvádí oficiální název pole, zda je možné pro porovnání použít masku (M), zda je pole povinné (P), jeho prerekvizity a popis.

Tabulka 3.1: Seznam virtuálních portů

Název	Číslo	Účel
OFPP_MAX	0xffffff00	Maximální rozsah portů
OFPP_UNSET	0xfffffff7	Nespecifikovaná hodnota v rámci pole *action-set*
OFPP_IN_PORT	0xfffffff8	Port, na který byla zpráva přijata
OFPP_TABLE	0xfffffff9	Označení první flow tabulky pro zpracování zprávy typu *packet-out*
OFPP_NORMAL	0xfffffffa	Zpracování tradiční síťovou vrstvou
OFPP_FLOOD	0xfffffffb	*Flood* pomocí tradiční síťové vrstvy
OFPP_ALL	0xfffffffc	Odeslání na všechny porty kromě příchozího
OFPP_CONTROLLER	0xfffffffd	Přeposlání na připojený SDN kontrolér
OFPP_LOCAL	0xfffffffe	Lokální OpenFlow port
OFPP_ANY	0xffffffff	Nespecifikovaná hodnota používaná pro speciální případy (maska)

V případě, že dané pole podporuje masku, ta může být uvedena, ale nemusí. V tom případě pak musí hodnota přesně odpovídat.

Sloupec *P* pak určuje, zda je implementace pole podle specifikace povinná. Pole s hodnotou A (ano) by tedy měly být implementovány na všech zařízeních a ve všech aplikacích podporujících specifikaci OpenFlow verze 1.5.1. Pole s hodnotou N (ne) jsou pak pouze volitelná a nemusí být implementována.

Prerekvizity určují vyžadovaná dodatečná pole, která musí být současně nastavená při porovnávání zpráv. Jedná se o pole z hlaviček protokolů nižších vrstev ISO/OSI modelu, která určují přítomnost porovnávaného pole v dané zprávě.

3.6.5 Podporované instrukce

Specifikace OpenFlow umožňuje v rámci flow pravidla provádění následujících instrukcí:

- **Goto_table** – specifikuje označení následující tabulky, do které bude zpráva přesunuta pro další zpracování.
- **Write_metadata** – nastavení pole s volitelnými údaji použitelnými při zpracování v dalších tabulkách (data se neposílají mimo zařízení). Tato akce obsahuje také volitelné pole *metadata_mask*, které lze využít jako ukazatel na vybraná pole hlavičky.
- **Write_actions** – zapíše akce do setu konkrétního zařízení. Pokud již daný set obsahuje nějaké akce, nové akce budou přidány. V případě, že dvě akce jsou stejné, zařízení může změny odmítnout a odpovědět chybovou zprávou, nebo může přepsat starší akci.
- **Apply_actions** – okamžitě aplikuje list specifikovaných akcí, které jsou v rámci zprávy prováděny sekvenčně (*in-order).*
- **Clear_actions** – z daného setu odstraní veškeré definované akce.
- **Stat_trigger** – umožňuje poslání statistik na kontrolér. Tato akce vyžaduje nastavení podmínek spouštění (list obsahující jednu až více podmínek – *thresholds*). Tyto podmínky pak mohou nastávat periodicky, nebo pouze jednorázově – nastavení se provádí příznakem *flags*.
- **Experimenter** – pole pro testování ve formátu typické experimentální struktury.

Tabulka 3.2: Seznam podporovaných polí shody (match fields)

Název	M.	P.	Prerekvizity	Popis
Rámec				
IN_PORT	N	A	–	Příchozí port (fyzický, nebo logický)
IN_PHY_PORT	N	N	–	Fyzický příchozí port
METADATA	A	A	–	Pomocná data pro využití při zpracování mezi flow tabulkami
ETH_DST	A	A	–	Ethernetová cílová adresa
ETH_SRC	A	A	–	Ethernetová zdrojová adresa
ETH_TYPE	N	A	–	Typ rámce
VLAN_VID	A	N	–	VLAN ID protokolu 802.1Q
VLAN_PCP	N	N	VLAN_VID != 0	VLAN priorita protokolu 802.1Q
ARP_OP	N	N	ETH_TYPE=0x0806	ARP opcode
ARP_SPA	A	N	ETH_TYPE=0x0806	ARP zdrojová IPv4 adresa (v datech)
ARP_TPA	A	N	ETH_TYPE=0x0806	ARP cílová IPv4 adresa (v datech)
ARP_SHA	A	N	ETH_TYPE=0x0806	ARP zdrojová MAC adresa (v datech)
ARP_THA	A	N	ETH_TYPE=0x0806	ARP cílová MAC adresa (v datech)
PBB_ISID	A	N	ETH_TYPE=0x88E7	I-SID v první PBB tagu
PBB_UCA	N	N	ETH_TYPE=0x88E7	UCA pole v prvním servisním tagu architektury Provider Backbone Bridges
IPv4 paket				
IP_DSCP	N	N	ETH_TYPE=0x0800 / 0x86dd	DSCP pole součástí ToS pro mechanismus QoS
IP_ECN	N	N	ETH_TYPE=0x0800 / 0x86dd	ECN část pole ToS pro mechanismus QoS
IP_PROTO	N	A	ETH_TYPE=0x0800 / 0x86dd	IP protokol
IPV4_SRC	A	A	ETH_TYPE=0x0800	IPv4 zdrojová adresa
IPV4_DST	A	A	ETH_TYPE=0x0800	IPv4 cílová adresa
ICMPV4_TYPE	N	N	IP_PROTO = 1	ICMP typ
ICMPV4_CODE	N	N	IP_PROTO = 1	ICMP kód
MPLS_LABEL	N	N	ETH_TYPE=0x8847 / 0x8848	MPLS label v první SHIM hlavičce
MPLS_TC	N	N	ETH_TYPE=0x8847 / 0x8848	TC v první SHIM hlavičce
MPLS_BOS	N	N	ETH_TYPE=0x8847 / 0x8848	Bottom of Stack bit v první SHIM hlavičce
IPv6 paket				
IPV6_SRC	A	A	ETH_TYPE=0x86dd	Zdrojová IPv6 adresa
IPV6_DST	A	A	ETH_TYPE=0x86dd	Cílová IPv6 adresa
IPV6_FLABEL	A	N	ETH_TYPE=0x86dd	IPv6 flow label
ICMPV6_TYPE	N	N	IP_PROTO = 58	ICMPv6 typ
ICMPV6_CODE	N	N	IP_PROTO = 58	ICMPv6 kód
IPV6_ND_TARGET	N	N	ICMPV6_TYPE=135 / 136	Cílová adresa IPv6 Neighbor Discovery
IPV6_ND_SLL	N	N	ICMPV6_TYPE=135	Zdrojová link-local adresa v IPv6 Neighbor Discovery
IPV6_ND_TLL	N	N	ICMPV6_TYPE=136	Cílová link-layer adresa v IPv6 Neighbor Discovery
IPV6_EXTHDR	A	N	ETH_TYPE=0x86dd	IPv6 extension header

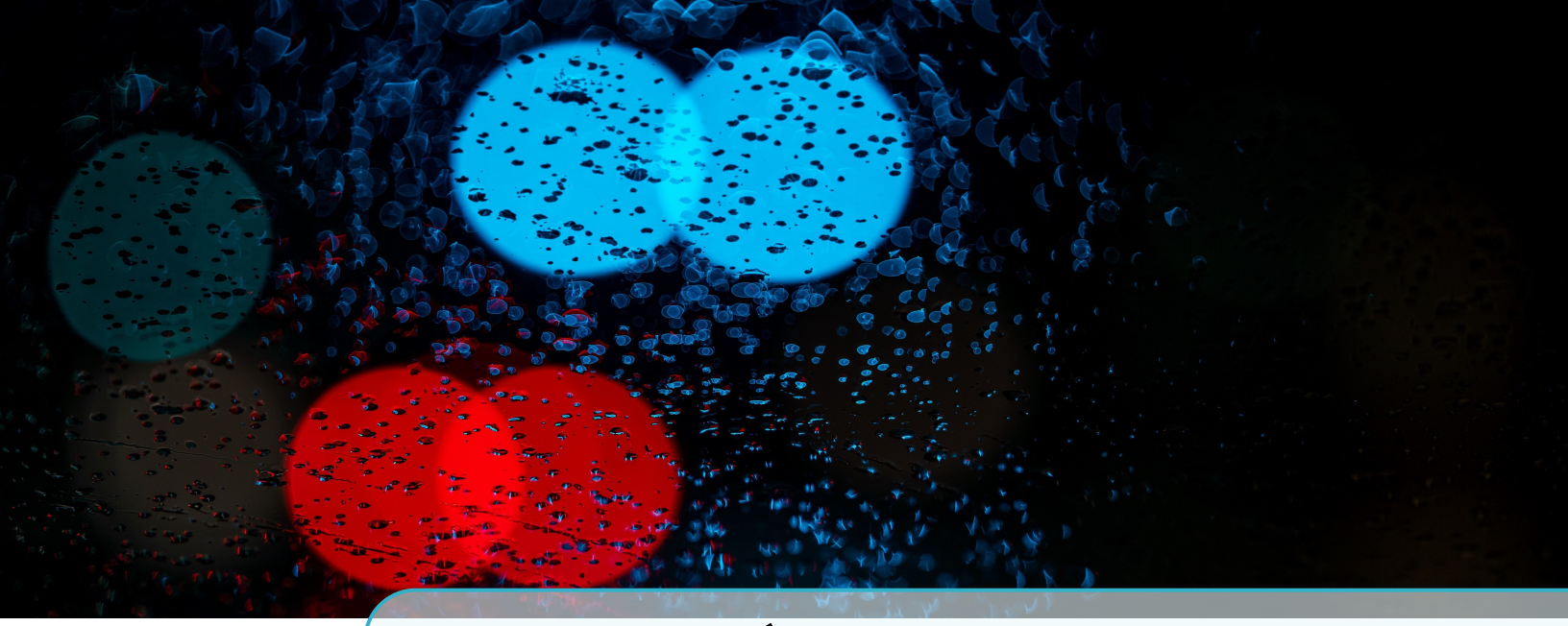
Název	M.	P.	Prerekvizity	Popis
		Segment		
TCP_SRC	N	A	IP_PROTO = 6	TCP zdrojový port
TCP_DST	N	A	IP_PROTO = 6	TCP cílový port
UDP_SRC	N	A	IP_PROTO = 17	UDP zdrojový port
UDP_DST	N	A	IP_PROTO = 17	UDP cílový port
SCTP_SRC	N	N	IP_PROTO = 132	SCTP zdrojový port
SCTP_DST	N	N	IP_PROTO = 132	SCTP cílový port
TCP_FLAGS	A	N	IP_PROTO = 6	TCP přepínače

* M = maska, P = povinný parametr specifikace OpenFlow

3.6.6 Podporované akce

Každé flow pravidlo může mít definován libovolný počet akcí. V případě, že akce není definována, je odpovídající zpráva zahozena. V opačném případě je provedena odpovídající akce. Pokud je akcí více, jsou provedeny sekvenčně. Akce lze definovat instrukcí *APPLY_ACTIONS* a mezi podporované patří:

- **Output** (povinná) – odešle zprávu na definovaný port. Ten může být fyzický, logický nebo speciální viz tabulka 3.1.
- **Group** (povinná) – předá zprávu pro zpracování konkrétní skupinou.
- **Drop** (povinná) – zahodí zprávu. Tato akce však nemá žádnou slovní hodnotu a je definována pouze nastavením prázdné akce *output*, nebo *group*, případně použitím instrukce *clear_actions*.
- **Set-Queue** (volitelná) – předá zprávu na specifikovanou frontu asociovanou s konkrétním fyzickým portem.
- **Meter** (volitelná) – předá zprávu na definovaný meter, který je zodpovědný za další zpracování.
- **Push-Tag / Pop-Tag** (volitelná) – umožňuje přidání nebo odebrání následujících tagů (musí se ve zprávě nacházet): VLAN, MPLS, PBB. Specifikace doporučuje implementaci alespoň pro tag VLAN.
- **Set-Field** (volitelná) – umožňuje modifikaci polí definovaných v hlavičkách zprávy. Specifikace doporučuje implementaci alespoň polí spojených s funkcionalitou VLAN.
- **Copy-Field** (volitelná) – v závislosti na typu zprávy akce umožňuje zkopírování dat mezi hlavičkou, hlavičkou a registrem, nebo hlavičkami odlišných zpráv. Akce má dva parametry – zdrojové a cílové pole.



4. SDN kontrolér

SDN kontrolér je software, který spravuje veškeré síťové prvky podporující SDN v dané administrativní doméně. Kontrolér na sebe přejímá veškerou logiku síťových funkcí z připojených SDN prvků. Ty sami o sobě neprovádí žádné složité funkce a místo toho od kontroléru očekávají *pokyny*. Hlavní funkcionality kontroléru jsou napsány v jednotném programovacím jazyce. Rozšiřující funkcionality pak mohou být napsány v odlišném jazyce za předpokladu, že kontrolér poskytuje aplikační rozhraní pro výměnu informací.

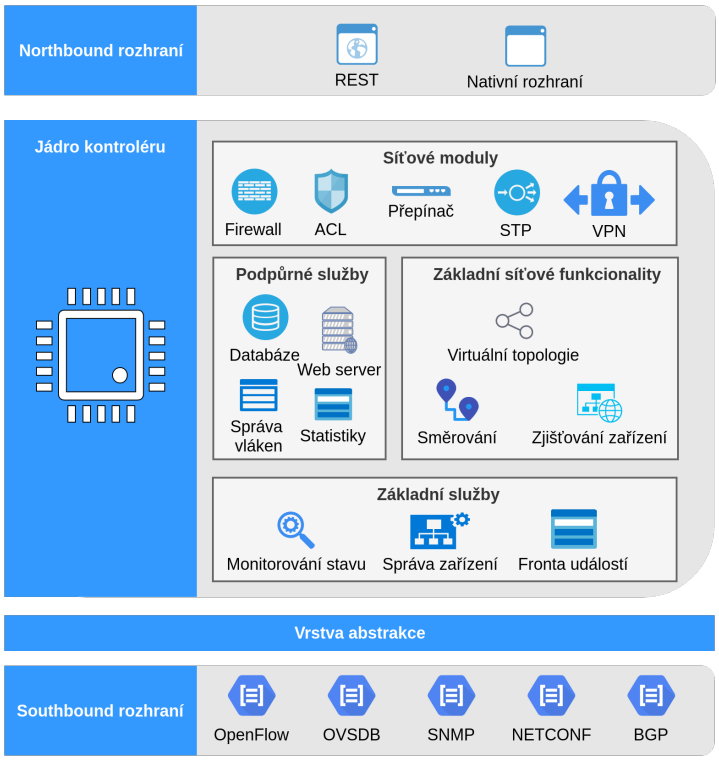
Druhy kontrolérů

SDN kontroléry lze rozdělit na dva základní druhy:

- **Open source** – jedná se o volně dostupné kontroléry, které lze stáhnout a používat zdarma. Těchto kontrolérů existuje velké množství, od prvotních experimentálních až po pokročilé nástroje vhodné pro produkční sítě. Tyto kontroléry bývají často vyvíjeny komunitou a jejich kompletní zdrojový kód je volně přístupný. Takové kontroléry využívají otevřená rozhraní a lze je tak využít pro správu většiny síťových prvků.
- **Komerční** – tyto kontroléry jsou vyvíjeny korporacemi a prodávány za jednorázovou platbu, nebo periodický licenční poplatek. Často bývají založeny na open source kontrolérech a pouze rozšiřují či upravují jejich funkcionalitu. Součástí poplatku za zakoupení kontroléru pak bývá i profesionální podpora. Nevýhodou těchto kontrolérů je, že často využívají proprietární protokoly, které mohou být omezeny pouze na síťové prvky daného výrobce.

4.1 Architektura SDN kontroléru

Obecná architektura SDN kontroléru obsahuje vnitřní moduly, které poskytují základní síťovou funkcionalitu, rozhraní pro komunikaci se síťovými prvky a volitelně také rozhraní pro komunikaci s externími aplikacemi, které mohou dále rozšiřovat funkcionalitu řídící vrstvy. Tato architektura je znázorněna na obrázku 4.1.



Obrázek 4.1: Architektura SDN kontroléru

4.1.1 Fyzická podoba

Kontrolér bývá nejčastěji umístěn na dedikovaném serveru, který ideálně disponuje dostatečným výkonem podpořeným moderními technologiemi jako je virtualizace a vysoká dostupnost. Virtualizace umožňuje dynamické škálování výkonu a tím pádem minimalizuje riziko výpadků spojené s neočekávaným nárůstem vytížení sítě. Technologie vysoké dostupnosti (redundantní, za běhu vyměnitelné komponenty, zdvojené elektrické zdroje a systém UPS) snižují riziko kompletního výpadku serveru. Současné použití těchto technologií eliminuje nevýhodu v podobě jednoho centralizovaného prvku, při jehož selhání by mohlo dojít k nedostupnosti celé sítě.

4.1.2 Rozhraní kontroléru

Jižní (southbound) rozhraní

Každý SDN kontrolér musí podporovat alespoň jedno *southbound* rozhraní (typicky OpenFlow). Pokročilejší kontroléry pak umožňují využít více typů různých *southbound* protokolů, které mohou být využity najednou, nebo jednotlivě. Mezi nejpoužívanější *southbound* protokoly patří:

- **CAPWAP (Control and Provisioning of Wireless Access Points)** – protokol zejména pro řízení přístupových bodů.
- **BGP (Border Gateway Protocol)** – směrovací protokol známý z tradičních počítačových sítí umožňuje implementaci logiky SDN.
- **NETCONF (Network Configuration Protocol)** – protokol umožňuje konfiguraci síťových prvků.
- **OF-Config** – nástroj pro správu prvků podporujících OpenFlow protokol.

- **OpenFlow** – nejpoužívanější protokol pro implementaci SDN logiky. Bývá také označován jako jediný protokol umožňující *čisté SDN*.
- **OVSDB (Open vSwitch Database)** – původně vyvinutý protokol pouze pro správu softwarových přepínačů Open vSwitch je nyní podporován větším množstvím zařízení.
- **SNMP (Simple Network Management Protocol)** – velice rozšířený protokol umožňující správu většiny síťových prvků.

Severní (northbound) rozhraní

Většina kontrolérů pak dále poskytuje *northbound* rozhraní. To slouží pro komunikaci mezi kontrolérem a externími aplikacemi, které mohou významným způsobem rozšiřovat funkcionalitu kontroléru. Tyto aplikace mohou být napsány v libovolném programovacím jazyce nezávisle na jazyku, který používá kontrolér. Jediným předpokladem je podpora daného rozhraní. Nejčastěji bývá použito rozhraní REST, ke kterému lze snadno přistupovat webovým prohlížečem zadáním URL adresy (alternativou je využití nástroje jako je například *curl*[1]).

4.1.3 Funkcionality kontroléru

Kontrolér zajišťuje tradiční síťové funkce a v závislosti na jeho zaměření pak může poskytovat i pokročilé aplikace. Tyto funkce mohou být naprogramovány jako tzv. moduly nebo aplikace. Terminologie jednotlivých kontrolérů se liší, ale obecně je platné následující rozdělení:

- **Moduly** – jsou součástí jádra kontroléru a využívají stejný programovací jazyk, jako samotný kontrolér. Moduly spolu komunikují přímo přes vnitřní instrukce a dosahují tak nejvyšší možné rychlosti zpracování a poskytují nejširší spektrum dostupných funkcí.
- **Aplikace** – aplikace jsou napsány jako samostatný kód nacházející se mimo vlastní kontrolér. Pro komunikaci s kontrolérem využívají *northbound* rozhraní.

Jednotlivé kontroléry tyto pojmy často zaměňují, nebo důsledně nedodržují. Je proto vhodné si ověřit použitou architekturu konkrétní funkce.

Základní moduly

Každý modul by měl poskytovat pouze jeden typ síťové funkcionality – například směrování. Kontrolér umožňuje aktivování a deaktivování jednotlivých modulů v závislosti na požadované funkcionalitě. Pouze některé kontroléry však umožňují dynamické aktivování nebo deaktivování modulu za běhu kontroléru.

Mezi nejčastěji podporované moduly patří:

- **ACL (Access Control List)** – jednoduchý mechanismus pro filtrování provozu.
- **Firewall** – pokročilejší mechanismus pro stavové i bezestavové filtrování provozu.
- **L2 přepínač** – zajišťuje přeposílání rámců pomocí MAC adres.
- **L3 směrovač** – zajišťuje přeposílání paketů pomocí IP adres.
- **Protokoly L2 pro zabránění vzniku smyček** – protokoly typu Spanning Tree (STP) umožňují použití redundantních topologií i na sítích neumožňujících směrování (tzv. *flat* sítě).

[1] `https://curl.haxx.se/`

Pokročilé moduly

Mezi pokročilejší funkce poskytované moduly pak patří zejména:

- **AAA funkcionalita** – pokročilejší kontroléry využívají bezpečnostní principy autentizace, autorizace a účtování, použité hlavně při komunikaci s uživatelem.
- **GUI** – téměř veškeré kontroléry umožňují alespoň základní správu přes grafické rozhraní. To bývá nejčastěji implementované ve webovém formátu a tedy dostupné přes jakékoliv koncové zařízení.
- **Přehled aktivních koncových prvků** – kontrolér si pomocí *packet-in* zpráv může udržovat seznam koncových prvků, které se pokoušely o navázání komunikace.
- **Sběr statistik** – kontrolér typicky poskytuje přehled o stavu síťové topologie včetně množství provozu.
- **Správa připojených síťových prvků** – kontrolér obsahuje databázi s připojenými síťovými prvky, na které si udržuje reference a zajišťuje periodickou komunikaci.
- **Udržování směrovacích informací** – kontrolér musí vytvářet pravidla pro přeposílání a provádění dalších akcí s různými datovými toky. K tomu potřebuje znát kompletní síťovou topologii, pro kterou si udržuje směrovací informace.
- **Zjišťování topologie** – kontrolér si dokáže vytvořit obraz kompletní topologie pomocí zpráv přijatých od připojených síťových prvků. Ty typicky obsahují *hello* zprávy, zprávy s informacemi o aktuálních datových tocích a také zprávy protokolů jako je LLDP (Link Layer Discovery Protokol).
- **VPN** – kontroléry se zaměřením na bezpečnost mohou poskytovat funkcionalitu pro navázání šifrovaných virtuálních spojení pomocí technologie VPN (Virtual Private Network).

Aplikace

Externí aplikace se využívají zejména pro pokročilejší funkce, u kterých nevadí nižší rychlost zpracování. Mezi další výhody aplikací oproti modulům pak patří:

- **Přenositelnost** – aplikace mohou být přenositelné mezi různými kontroléry, za předpokladu, že tyto kontroléry poskytují stejné rozhraní.
- **Nezávislost na jádře kontroléru** – aktualizace kontroléru by neměla způsobit nefunkčnost aplikace, což u modulu nemusí platit.
- **Podpora libovolného programovacího jazyka** – aplikace může být napsána v libovolném programovacím jazyce.
- **Flexibilita umístění** – samotná aplikace může být spuštěna na libovolném zařízení, které má konektivitu na kontrolér. Tím je možné logicky i fyzicky oddělit jednotlivé síťové funkce.

4.2 Historie SDN kontrolérů

4.2.1 První a experimentální kontroléry

Za první kontrolér SDN je považován **NOX**, který vznikl současně s první verzí protokolu OpenFlow. Tento kontrolér byl původně vyvíjen firmou Nicira Networks, ale v roce 2008 byl uvolněn jako open source. Kontrolér byl dále vyvíjen komunitou a jeho zdrojové kódy jsou dostupné na platformě Github. NOX je napsaný v jazyce C++ a podporuje pouze OpenFlow verze 1.0. Umožňuje pouze základní síťovou funkcionalitu jako je L2 přepínač a zjišťování topologie. Tento kontrolér sloužil hlavně jako první experiment a vzhledem k jeho neaktuálnosti už není používán.

Zdrojové kódy kontroléru NOX jsou dostupné na platformě GitHub:
`https://github.com/noxrepo/nox`

Na základě kontroléru NOX vznikla v roce 2010 verze napsaná v jazyce Python – **POX**. Na rozdíl od původního kontroléru je multiplatformní (jedinou podmínkou je podpora jazyka Python verze 2.6 / 2.7) a výkonově je na tom velice podobně jako kontrolér NOX (i přes použití jazyka Python). Stejně jako kontrolér NOX podporuje pouze první verzi protokolu OpenFlow.

V současnosti POX není pouze SDN kontrolérem, ale zastává také funkci softwarového přepínače podporujícího OpenFlow protokol.

Zdrojové kódy kontroléru POX jsou dostupné na platformě GitHub:
`https://github.com/noxrepo/pox`

V roce 2010 byl také vyvinut první SDN kontrolér využívající programovací jazyk Java – **Beacon**. Ten je také multiplatformní a využívá vícevláknovou architekturu, která umožňuje dosažení vysokého výkonu. To do jisté míry eliminuje použití obecně méně efektivního jazyka Java[2]. Oproti starším kontrolérům pak poskytuje i webové rozhraní a umožňuje dynamické spouštění či zastavování služeb bez nutnosti vypnutí kontroléru. Kontrolér podporuje L2 přeposílání i L3 směrování a opět pouze první verzi protokolu OpenFlow.

4.2.2 Aktuální kontroléry

Na základě prvotních experimentů vzniklo několik kontrolérů, které jsou stále udržované a pravidelně aktualizované. Hodí se zejména pro experimentování s technologií SDN a nasazení v menších či testovacích sítích. Hlavní výhodou těchto kontrolérů je jejich relativní jednoduchost a hardwarová nenáročnost.

Pravděpodobně nejúspěšnějším kontrolérem využívajícím jazyk Python je kontrolér **RYU**. Ten je i v současnosti stále aktuálně udržován a podporuje i verzi protokolu OpenFlow 1.5. Kontrolér nabízí několik síťových modulů včetně L2 přepínače (dostupný pro verze OpenFlow 1.0, 1.1, 1.2, 1.3, 1.4 a 1.5), L3 směrovače, firewallu, STP, mechanismu QoS, monitorování topologie a dalších.

Zdrojové kódy kontroléru jsou podrobně vysvětleny v publikaci *RYU SDN Framework*, kterou lze volně stáhnout ze stránek kontroléru[3].

Oficiální webové stránky kontroléru RYU, obsahující dokumentaci a veškeré zdrojové kódy:
`https://osrg.github.io/ryu/`

Na základě kontroléru Beacon byl vytvořen kontrolér **Floodlight**. Ten také používá programovací jazyk Java a je vyvíjen komunitou. Floodlight podporuje několik *southbound* rozhraní (ne jenom OpenFlow protokol) a umožňuje kombinovat prvky podporující OpenFlow protokol s klasickými zařízeními. Kromě modulů umožňujících různé síťové funkce pak obsahuje i pokročilejší aplikace využívající *northbound* rozhraní.

Oficiální webové stránky kontroléru Floodlight, obsahující dokumentaci a veškeré zdrojové kódy: `http://www.projectfloodlight.org/floodlight/`

Po úspěchu kontrolérů Beacon a Floodlight začaly vznikat první komerční kontroléry od firem jako je Cisco, HP, IBM, VMWare, Juniper a další.

[2] Podrobnosti o výkonu kontroléru Beacon lze nalézt v článku: D. Erickson, *The Beacon OpenFlow Controller*, 2013
[3] `https://osrg.github.io/ryu/resources.html`

4.2.3 Moderní komplexní platformy

S rostoucí popularitou SDN vyvstala potřeba spolehlivých kontrolérů, které by se hodily pro nasazení do produkčních sítí, vyžadujících vysokou spolehlivost a dostupnost. Tyto kontroléry jsou pravidelně aktualizovány.

Hlavním zástupcem této kategorie a v současnosti nejrozšířenější SDN kontrolér je **OpenDaylight**. Ten je založen na jazyce Java a částečně vychází z kontroléru Beacon. Kontrolér se soustředí na vysokou dostupnost a podporuje velké množství *southbound* protokolů. Poslední verze kontroléru pochází z března 2019 (*Neon*). Kontrolér je vyvíjen velkým množstvím různých společností a bývá také často používaným základem pro komerční kontroléry.

Oficiální webové stránky kontroléru OpenDaylight, obsahující dokumentaci a veškeré zdrojové kódy: `https://www.opendaylight.org/`

Hlavním konkurentem kontroléru OpenDaylight je pak kontrolér **ONOS** (Open Networking Operating System), který byl firmou On.Lab uvolněn jako open source. Kontrolér je zaměřen převážně na sítě vyžadující škálovatelnost, výkon a vysokou dostupnost. Hodí se tak zejména pro sítě poskytovatelů konektivity.

Podobně jako OpenDaylight, podporuje ONOS velké množství *southbound* protokolů včetně BGP, CLI, NETCONF, OpenFlow, P4, RESTCONF, SNMP, TL1 a dalších.

Oficiální webové stránky kontroléru ONOS, obsahující dokumentaci a veškeré zdrojové kódy: `https://onosproject.org/`

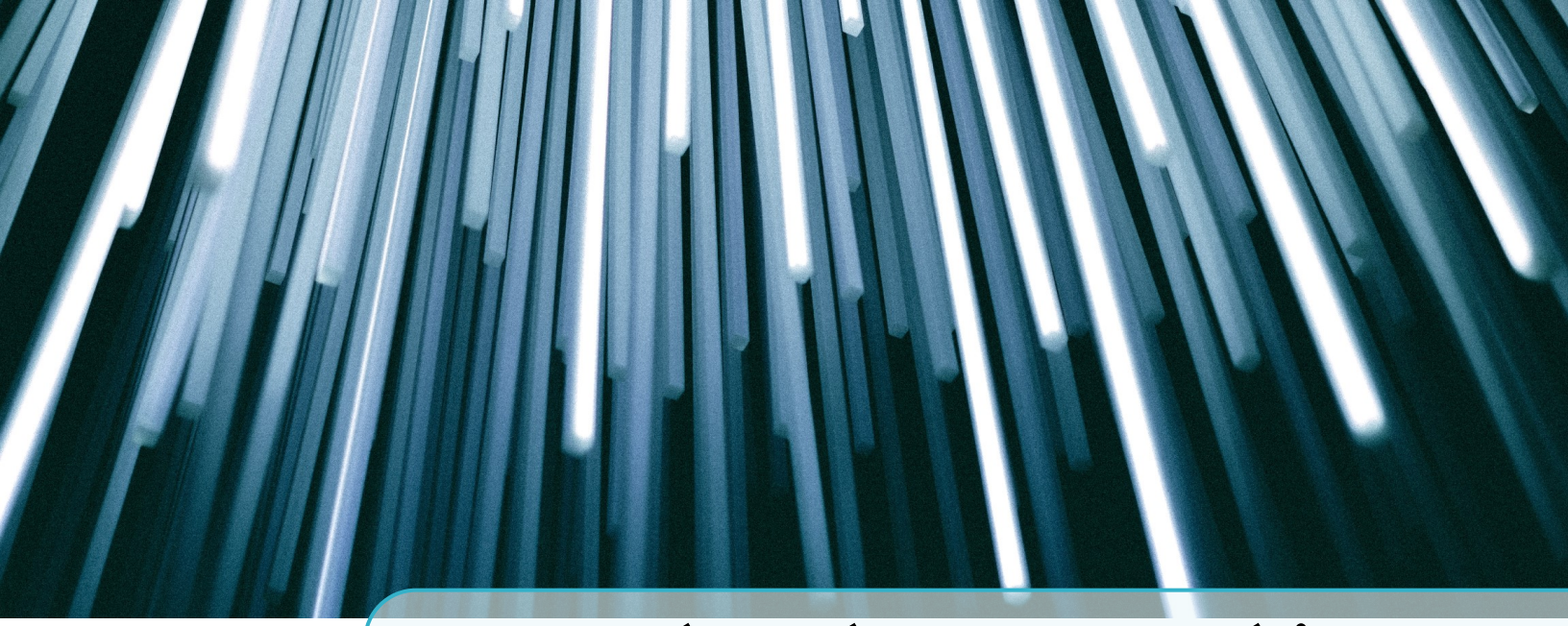
4.2.4 OpenFlow kontrolér

Jak již bylo zmíněno v úvodu této kapitoly, SDN kontrolér může, ale nemusí podporovat OpenFlow protokol. Pokud je protokol OpenFlow podporován, lze o kontroléru hovořit jako o OpenFlow kontroléru. Většina moderních kontrolérů však podporuje více typů *southbound* rozhraní (viz obrázek 4.1, a proto se označují obecně jako SDN kontroléry.

Z pohledu protokolu OpenFlow musí kontrolér podporující tento protokol zajišťovat následující funkcionality:

- Navázání a udržování komunikace se síťovými zařízeními.
- Správa flow tabulek připojených prvků.
- Zpracování přijatých *packet-in* zpráv a následující reakce (*packet-out*).
- Zpracování chybových zpráv a jejich případný výpis.

Veškeré uvedené funkce musí být zajištěny dodržením přesných specifikací dané verze OpenFlow protokolu včetně implementace všech povinných parametrů. Volitelné parametry implementovány být nemusí.



5. Nasazení a správa SDN kontrolérů

SDN kontroléry se liší v použitém programovacím jazyce, počtu modulů, v poskytovaných funkcionalitách, podporovaných rozhraní a v systému instalace. V této kapitole jsou představeny postupy pro instalaci a základní správu jednotlivých kontrolérů. Ta nejčastěji probíhá přes webové rozhraní, kde je možné nastavit požadované síťové funkcionality (například firewall, DHCP, atd.).

5.1 RYU

RYU jako jeden ze snadno pochopitelných aktuálních kontrolérů bývá součástí různých virtuálních obrazů pro vývoj SDN aplikací. RYU je postaven na jednoduché programovací architektuře, kterou lze snadno modifikovat. Díky tomu je tak kontrolér vhodný pro vývojáře začínající s technologií SDN. Další výhodou jsou nízké hardwarové nároky.

Programovací architektura RYU kontroléru obsahuje:

- **Frontu událostí** – je použita pro ukládání přijatých OpenFlow zpráv.
- **Vlákno obsluhy** – každý modul a služba kontroléru běží ve vlastním vláknu, které z fronty událostí postupně odebírá zprávy a volá rutiny deklarované pro jejich obsluhu.
- **Obslužné rutiny** – funkce definované pomocí pevně daných *handlerů* jsou automaticky volány při nastání specifické události.

5.1.1 Moduly kontroléru

RYU kontrolér implementuje veškerou jeho poskytovanou funkcionalitu formou modulů. Ty jsou umístěny v adresáři */ryu/app/*.

Veškeré uvedené moduly využívají rozhraní OpenFlow a RYU kontrolér je tedy možné označit jako OpenFlow kontrolér.

Označení *app* (*applications*) dokazuje nekonzistentní použití pojmů modul a aplikace, jak bylo zmíněno v kapitole 4.1.3. Ve skutečnosti se opravdu jedná o moduly.

Moduly mají formu Python skriptů (*.py*) a následující klíčová slova uvedená v jejich názvu značí podporovanou funkcionalitu:

- **rest** – modul podporuje REST rozhraní. K tomu je možné přistupovat pomocí jakéhokoliv prohlížeče, případně nástrojem *curl*.
- **12, 13, 14** – značí verzi OpenFlow protokolu, pro kterou je modul určen (1.2, 1.3, 1.4). Modul bez tohoto označení – například *simple_switch.py* je určen pro verzi 1.0.

Seznam modulů

Mezi nejpoužívanější moduly dodávané s kontrolérem patří (v závorkách jsou uvedeny názvy souborů jednotlivých modulů):

- **Firewall** (*rest_firewall.py*) – umožňuje využití filtrovacího mechanismu ACL na základě IPv4 adres a VLAN. Samotné ovládání je možné provádět pomocí REST rozhraní.
- **LACP** (*simple_switch_lacp.py*, *simple_switch_lacp_13.py*) – Link Aggregation Control Protocol umožňuje vytvoření logického spoje z několika fyzických spojení za účelem dosažení vyšší rychlosti přenosu dat a vyvážení zatížení. Modul zároveň zajišťuje samotné přeposílání paketů pomocí funkcionality jednoduchého L2 přepínače (*simple_switch*).
- **QoS** (*rest_qos.py*) – Quality of Service je funkcionalita kvality služeb, která definuje priority různým typům datového provozu. Umožňuje rezervování garantované přenosové rychlosti a poskytování služeb s danými parametry. Implementace využívá vlastnosti OpenFlow protokolu verze 1.3 (*meter* tabulky). Samotné ovládání je opět možné provádět přes poskytované REST rozhraní.
- **L2 přepínač** (*simple_switch.py*, *simple_switch_XY.py*, *simple_switch_rest_13.py*) – implementace L2 přepínače dostupná pro různé verze OpenFlow protokolu. Číslo uvedené v názvu modulu (pozice *XY*) značí verzi OpenFlow protokolu (například modul *simple_switch_13.py* je určen pro verzi 1.3). Verze s klíčovým slovem *rest* pak poskytuje REST rozhraní. To umožňuje výpis obsahu MAC tabulky z vybraného zařízení (ve formě JSON souboru) a vložení nového flow pravidla na základě MAC adresy (při současném vložení této adresy do MAC tabulky).
- **Simple Monitor** (*simple_monitor_13.py*) – poskytuje statistiky o stavu síťové topologie. Ty jsou periodicky sbírané z různých datových struktur připojených zařízení. Modul obsahuje funkcionalitu L2 přepínače (ten by proto neměl být spouštěn zároveň). Modul je dostupný pouze pro OpenFlow verze 1.3.
- **OFCTL** (*ofctl_rest.py*) – nástroj pro monitorování a správu OpenFlow zařízení pomocí manuálního zasílání OpenFlow zpráv. Ty lze vytvářet přes REST rozhraní, které umožňuje výpis stavu a statistik zařízení, vkládání nových flow pravidel a další funkce.
- **Směrovač** (*rest_router.py*) – poskytuje funkcionalitu L3 směrovače. REST rozhraní pak umožňuje správu statického směrování, nastavení výchozí cesty, IP adres, VLAN a dalších funkcí.
- **STP** (*simple_switch_stp.py*, *simple_switch_stp_13.py*) – zabraňuje vzniku smyček v L2 topologiích. Redundantní spoje jsou blokovány na základě informací z přijatých BPDU (Bridge Protocol Data Unit). Ty protokolu umožňují vytvoření logické stromové topologie. Modul zároveň implementuje funkcionalitu L2 přepínače.
- **Topologie** (*rest_topology.py*) – poskytuje REST rozhraní pro zjišťování stavu přepínačů a jednotlivých spojů.
- **VTEP** (*rest_vtep.py*) – Virtual Tunnel End Point poskytuje enkapsulaci a deenkapsulaci zpráv

pro pakety typu VXLAN. Modul poskytuje REST rozhraní a využívá komunikaci pomocí protokolu OVSDB.

Aktuální verze RYU kontroléru obsahuje kromě modulů poskytujících základní síťovou funkcionalitu také moduly pro specifické případy, jako je integrace s nástrojem OpenStack. Jejich kompletní seznam, popis a zdrojové kódy lze nalézt na oficiálních GitHub stránkách kontroléru: `https://github.com/osrg/ryu/tree/master/ryu/app`

5.1.2 Instalace kontroléru

Pro nasazení RYU kontroléru je doporučen operační systém Ubuntu verze 16.04 nebo novější (preferované jsou LTS verze). Vlastní instalace kontroléru je jednoduchá a lze ji provést jedním příkazem pomocí nástroje *pip*[1]. Alternativou je instalace přímo ze zdrojových kódů dostupných na platformě GitHub.

```
# Instalace pomocí nástroje pip
pip install ryu

# Instalace ze zdrojových kódů
# 1) Stažení zdrojových kódů
git clone git://github.com/osrg/ryu.git

# 2) Instalace (v adresáři ryu)
pip install .
```

5.1.3 Spuštění kontroléru

Při samotném spuštění kontroléru nedojde automaticky k načtení žádného z uvedených modulů. Při spuštění je proto nutné definovat, který modul má být spuštěn (může jich být definováno více). RYU kontrolér se spouští příkazem *ryu-manager*.

```
ryu-manager [přepínače] [modul 1] [modul 2] [...]

# Příklad spuštění kontroléru se základním modulem simple switch a podrobným
# výpisem
ryu-manager ryu/app/simple_switch_13.py --verbose

# Přepínač pro definici vlastního OpenFlow portu
--ofp-tcp-listen-port

# Příklad spuštění s definicí vlastního portu
ryu-manager ryu/app/simple_switch_13.py --ofp-tcp-listen-port 6653

# Přepínač pro zobrazení všech podporovaných příkazů
--help
```

Deaktivace nebo aktivace vybraného modulu za běhu kontroléru není možná a jediným způsobem je zastavení celého kontroléru a jeho opětovné spuštění s nově vybranými moduly.

Více informací ohledně praktického používání kontroléru RYU ve spojení s emulátorem Mininet je popsáno v kapitole 6.5.4.

[1] Nástroj slouží pro správu a instalaci Python balíčků. Více informací: `https://pypi.org/project/pip/`

Více informací o architektuře a používání kontroléru RYU je možné najít v práci *Analýza kontrolérů softwarově definovaných sítí* [3, str. 22].

5.2 Floodlight

Floodlight kontrolér je oproti kontroléru RYU mnohem komplexnější, ale přesto má podobně nízké hardwarové požadavky. Kontrolér je vytvořen v programovacím jazyce Java a pro spuštění vyžaduje nainstalovaný JDK (Java Development Kit) obsahující JVM (Java Virtual Machine). Pro vývoj modulů a případné úpravy kódu kontroléru je doporučeno používat vývojové prostředí Eclipse[2], do kterého je možné kontrolér pomocí předpřipravených skriptů snadno integrovat. Toto prostředí bývá často dodávané v různých předpřipravených virtuálních obrazech počítačů pro práci s SDN.

Příkladem typického předpřipraveného obrazu je například *All-in-one SDN App Development Starter VM* dostupný na stránkách: `http://sdnhub.org/tutorials/sdn-tutorial-vm/`

Floodlight kontrolér je v současné době stále používán zejména pro menší a experimentální sítě.

Aktuální verze kontroléru je 1.2 z roku 2016. Její zdrojové kódy jsou dostupné na platformě GitHub: `https://github.com/floodlight/floodlight`

5.2.1 Moduly kontroléru

Floodlight kontrolér je dodáván s velkým množstvím modulů, které jsou typicky spuštěny zároveň v závislosti na nastavené konfiguraci. Spouštěné moduly a aplikace lze nastavit v konfiguračním souboru: *src/main/resources/floodlightdefault.properties*

Aktivní moduly

Floodlight kontrolér spouští ve výchozím nastavení následující moduly (v závorkách jsou uvedeny celé názvy modulů pro jejich správu):

- **ACL** (*net.floodlightcontroller.accesscontrollist.ACL*) – umožňuje jednoduché filtrovací operace.
- **Debug counter** (*net.floodlightcontroller.debugcounter.DebugCounterServiceImpl*) – poskytuje statistiky z připojených zařízení a umožňuje jejich vynulování.
- **DHCP server** (*net.floodlightcontroller.dhcpserver.DHCPServer*, *net.floodlightcontroller.dhcpserver.DHCPSwitchFlowSetter*) – přiděluje IP adresy, výchozí bránu a nastavený DNS (Domain Name System) server koncovým zařízením.
- **Firewall** (*net.floodlightcontroller.firewall.Firewall*) – jedná se o pokročilejší variantu modulu ACL. Umožňuje detailnější nastavení filtrovacích pravidel včetně priorit. Ve výchozím stavu je veškerá komunikace povolena. Při přidání alespoň jednoho pravidla (REST nebo grafickým rozhraním) dojde k defaultnímu blokování veškeré ostatní komunikace.
- **Forwarding** (*net.floodlightcontroller.forwarding.Forwarding*) – zajišťuje konektivitu a zabraňuje vzniku smyček použitím Dijsktrova algoritmu (nevyužívá protokol STP).
- **High availability** (*org.sdnplatform.sync*) – umožňuje využití distribuované architektury kontrolérů pro vysokou dostupnost.
- **Link discovery** (*net.floodlightcontroller.linkdiscovery*) – zjišťuje a udržuje stav jednotlivých spojů v topologii.

[2] `https://www.eclipse.org/`

- **Load balancer** (*net.floodlightcontroller.loadbalancer.LoadBalancer*) – v případě vícecestné topologie zajišťuje pro protokoly ICMP, TCP a UDP jednoduché vyvažování zátěže na základě algoritmu *round-robin*.
- **Packet-in processing time** (*net.floodlightcontroller.perfmon*) – umožňuje měřit výkon jednotlivých modulů.
- **Routing manager** (*net.floodlightcontroller.routing*) – zjišťuje dostupné cesty mezi jednotlivými koncovými body a poskytuje REST rozhraní. To umožňuje prohlížení informací o jednotlivých cestách a také nastavení metrik pro výpočet cen daných cest.
- **Simple fault tolerance** (*net.floodlightcontroller.simpleft.FT*) – umožňuje nastavení záložní vrstvy kontroléru.
- **Statistics collector** (*net.floodlightcontroller.statistics.StatisticsCollector*) – sbírá statistiky z jednotlivých síťových prvků a spojů v určených intervalech. Statistiky je možné využít ve vlastních aplikacích přes REST rozhraní.
- **Static flow entry pusher** (*net.floodlightcontroller.staticentry.StaticEntryPusher*) – umožňuje statické vkládání flow paketů přes REST rozhraní. Tímto nástrojem lze implementovat logiku proaktivního vkládání flow pravidel přes vlastní aplikaci.
- **Topology manager** (*net.floodlightcontroller.topology*) – zjišťuje a spravuje stav aktuální síťové topologie.

Další moduly

Mezi další moduly dodávané s kontrolérem, které se ale ve výchozím stavu nespouštějí automaticky, patří:

- **Device manager** (*net.floodlightcontroller.devicemanager*) – umožňuje provádění interakcí se zařízeními v síti.
- **Learning switch** (*net.floodlightcontroller.learningswitch*) – modul poskytuje základní funkcionalitu L2 přepínače.
- **Network virtualization** (*net.floodlightcontroller.virtualnetwork*) – umožňuje logickou separaci zařízení na základě jejich MAC adres.

5.2.2 Instalace kontroléru

Instalace kontroléru vyžaduje nástroj *Ant*[3], nebo *Maven*[4] pro sestavení (*build*) aplikace. Dále jsou vyžadovány balíčky *Python development*[5] a JDK. Samotná instalace pak probíhá přes nástroj *Git*[6].

```
# Instalace vyžadovaných balíčků Ant, Python development a Git
sudo apt-get install ant python-dev git

# Stažení zdrojových kódů
git clone git://github.com/floodlight/floodlight.git

# Inicializace submodelu (v adresáři floodlight)
git submodule init
```

[3] https://ant.apache.org/
[4] https://maven.apache.org/
[5] https://pypi.org/project/python-dev-tools/
[6] https://git-scm.com/

```
# Aktualizace submodelu (v adresáři floodlight)
git submodule update

# Rebuild kontroléru (v adresáři floodlight)
ant
```

Instalace dále vyžaduje vytvoření adresáře */var/lib/floodlight* s právy pro čtení, zápis a spouštění souborů.

5.2.3 Spuštění kontroléru

Před samotným spuštěním kontroléru je nutné provést jeho konfiguraci editací souboru: *src/main/resources/floodlightdefault.properties*

V souboru lze nastavit použité porty (pro připojení síťových prvků a pro přístup k webovému rozhraní), různé časové limity, spouštěné moduly a další parametry. Následující kód uvádí nejvíce používané parametry (jejich defaultní hodnoty):

```
# Seznam spouštěných modulů při startu kontroléru
floodlight.modules=\
net.floodlightcontroller.jython.JythonDebugInterface,\
... # Další moduly

# Nastavení modulu "Forwarding" - použité pole shod
net.floodlightcontroller.forwarding.Forwarding.match=in-port,
 vlan, mac, ip, transport, flag

# Nastavení modulu "Forwarding" - výchozí časovač flow pravidel (5 sekund)
net.floodlightcontroller.forwarding.Forwarding.idle-timeout=5

# Nastavení čísla portu protokolu OpenFlow (připojení síťových prvků)
net.floodlightcontroller.core.internal.FloodlightProvider.openFlowPort=6653
net.floodlightcontroller.core.internal.OFSwitchManager.openFlowPort=6653

# Možnost aktivace šifrovaného OpenFlow spojení
net.floodlightcontroller.core.internal.OFSwitchManager.useSsl=NO

# Podporované verze OpenFlow protokolu
net.floodlightcontroller.core.internal.OFSwitchManager.
 supportedOpenFlowVersions=1.0, 1.1, 1.2, 1.3, 1.4, 1.5

# Nastavení webového rozhraní - podpora protokolů HTTP / HTTPS a jejich
# použité porty
net.floodlightcontroller.restserver.RestApiServer.useHttps=NO
net.floodlightcontroller.restserver.RestApiServer.useHttp=YES
net.floodlightcontroller.restserver.RestApiServer.httpsPort=8081
net.floodlightcontroller.restserver.RestApiServer.httpPort=8080

# Nastavení webového rozhraní - klientská autentizace
net.floodlightcontroller.restserver.RestApiServer.
 httpsNeedClientAuthentication=NO
```

Spuštění kontroléru je možné provést přes CLI, případně přímo z vývojového prostředí (*Eclipse*),

pokud je v něm kontrolér integrován. V takovém případě probíhá spuštění kontroléru kliknutím na volbu *Run/Run*, případně přímo na *zelenou šipku*.

Návod na integraci kontroléru do prostředí Eclipse: `https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide`

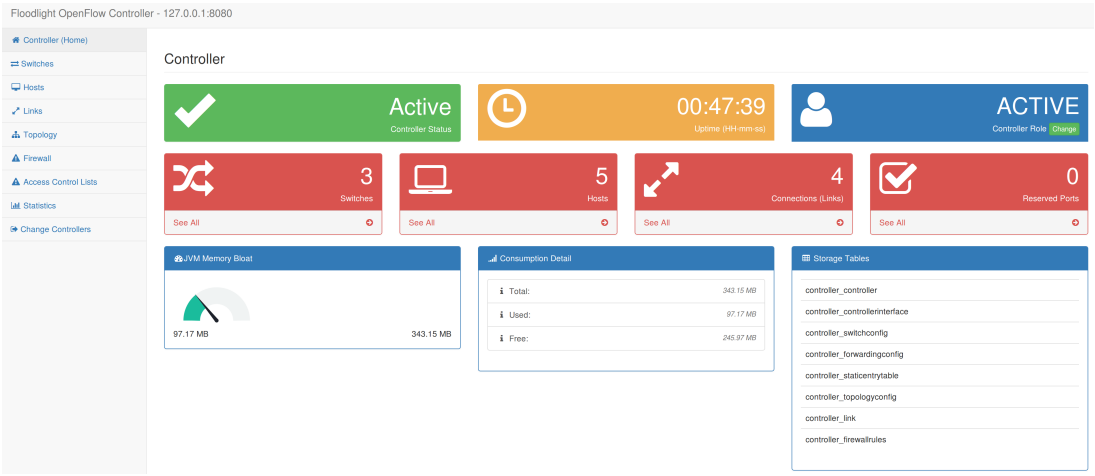
```
# Spuštění kontroléru Floodlight v prostředí terminálu
java -jar target/floodlight.jar

# Spuštění kontroléru Floodlight s vlastním konfiguračním souborem (SOUBOR)
java -jar target/floodlight.jar -cf SOUBOR
```

5.2.4 Grafické rozhraní

Kontrolér poskytuje grafické rozhraní, které se automaticky spouští při startu kontroléru. Rozhraní je dostupné ve formě webové stránky, která je ve výchozím nastavení dostupná na adrese: `http://<IP_ADRESA>:8080/ui/pages/index.html`

V případě, že je na stránku přistupováno ze stejného zařízení, na kterém je kontrolér spuštěn, jako IP adresu lze použít adresu *localhost* – 127.0.0.1, případně 0.0.0.0

Prostředí poskytuje informace o stavu topologie a umožňuje nastavení modulů firewallu, ACL a statistik.



Obrázek 5.1: Grafické rozhraní kontroléru Floodlight

Více informací o architektuře a používání kontroléru Floodlight je možné najít v práci *Analýza kontrolérů softwarově definovaných sítí* [3, str. 44].

5.2.5 Logika vytváření flow pravidel

Sekvenční

Podstatným rozdílem mezi kontroléry RYU a Floodlight je způsob vkládání nových flow pravidel. RYU využívá sekvenční způsob – jakmile je přijata zpráva *packet-in*, kontrolér vloží (pokud je to možné) nové flow pravidlo na síťový prvek, ze kterého byla zpráva přijatá. Původní zpráva je potom

poslána zpět na síťový prvek (*packet-out*), který ji přepošle na další *hop*. Na něm se situace bude opakovat – kontrolér opět vloží nové flow pravidlo pouze na tento síťový prvek.

Tento přístup je snadný na implementaci, ale s každým síťovým prvkem v cestě datového toku se zvyšuje latence (kvůli nutné komunikaci s kontrolérem a jeho softwarovému zpracování).

Simultánní

Kontrolér Floodlight používá odlišný způsob, ve kterém jsou veškerá flow pravidla vložena současně do všech prvků na cestě datového toku. Při přijetí první *packet-in* zprávy kontrolér ověří, zda zná cestu do cílového zařízení. Kontrolér si na základě spuštěných modulů udržuje informace o síťové topologii a připojených prvcích i koncových zařízení, a v ideálním stavu tak tuto cestu zná. V tom případě vytvoří flow pravidla pro všechny prvky na trase a pravidla do nich vloží. Původní zpráva je pak opět formou *packet-out* zprávy vrácena na výchozí síťový prvek.

Tento způsob je komplikovanější z pohledu implementace, ale poskytuje téměř konzistentní hodnotu latence prvním zprávám nového datového toku. Tato hodnota pak téměř není závislá na velikosti sítě a délce trasy.

Větší síťové topologie mají určitou režii (vytvoření více flow pravidel a četnější komunikace se síťovými prvky). Z pohledu celkové latence prvního datového toku je však zanedbatelná.

5.3 OpenDaylight

OpenDaylight je nejpoužívanějším SDN kontrolérem a často slouží jako základ pro komerční produkty[7]. Je používán hlavně v produkčních sítích, kde podle oficiálních informací[8] obsluhuje až bilion uživatelů. Kontrolér je spravován komunitou tvořenou velkým množstvím celosvětově známých společností.

OpenDaylight bývá označován jako modulární platforma, protože ze všech SDN kontrolérů podporuje největší množství *southbound* a *northbound* protokolů a je dodáván se spoustou modulů a aplikací.

Rozdíl mezi modulem a aplikací není v dokumentaci kontroléru rozlišován a oba typy jsou souhrnně označovány jako *feature*. V této publikaci pro ně bude použit pojem **funkcionality**.

OpenDaylight poskytuje robustní řešení, které je možné si flexibilně upravit podle požadavků konkrétního nasazení. To je možné díky jeho modulární architektuře. Kontrolér je tak možné propojit například se službami jako ONAP, OpenStack a OPNFV. Kontrolér je pak důsledně testován v oblasti *S3P*: bezpečnosti (*security*), škálovatelnosti (*scalability*), stabilitě (*stability*) a výkonu (*performance*), přičemž hlavní důraz je kladen právě na bezpečnost. Kontrolér v této oblasti poskytuje framework AAA a plně využívá výhod otevřeného zdrojového kódu. Samozřejmostí pak je podpora distribuované architektury pro nasazení na více zařízeních.

Kontrolér má za sebou více než šestiletou historii a nové verze jsou pravidelně vydávány dvakrát ročně. Pro každou verzi pak vychází drobnější aktualizace opravující případné problémy a vylepšující funkčnost kontroléru.

Aktuální verze kontroléru nazvaná *Neon* pochází z března 2019 a je dostupná ke stažení: `https://docs.opendaylight.org/en/latest/downloads.html`

[7] `https://www.opendaylight.org/what-we-do/odl-platform-overview`

[8] `https://www.opendaylight.org/announcement/2019/03/26/opendaylight-most-pervasive-open-source-sdn-controller-celebrates-sixth-anniversary-with-neon-release`

Nevýhodou kontroléru je pak právě jeho komplexnost. Kontrolér není vhodný pro úvodní seznámení se s technologií SDN nebo pro nasazení v malých experimentálních sítích na zařízení s omezeným hardwarovým výkonem.

5.3.1 Podporované funkcionality

Kontrolér je dodáván s následujícími funkcionalitami (jejich oficiální názvy, podle kterých je lze aktivovat, jsou uvedeny v závorkách):

- **ALTO** (*odl-alto-release*) – Application-Layer Traffic Optimization je protokol poskytující aplikační vrstvě pohledy na síť a dostupné služby.
- **AAA** (*odl-aaa-shiro*) – framework pro autentizaci, autorizaci a účtování. V případě instalace protokolu *RESTCONF* je framework nainstalován automaticky.
- **Centinel** (*odl-centinel-all*) – framework umožňující ostatním funkcionalitám přijímat události z více různých datových proudů a provádět hromadné akce.
- **Controller shield** (*odl-usecplugin*) – umožňuje vytvořit databázi bezpečnostních informací, která může být využita pro konfiguraci aplikace firewall nebo externími aplikacemi využívajícími *northbound* rozhraní.
- **Device Identification and Driver Management** (*odl-didm-all*) – poskytuje abstrakci od specifických zařízení.
- **GBP** (*odl-groupbasedpolicy-ofoverlay*, *odl-groupbasedpolicy-ui*) – Group Based Policy umožňuje inovativní způsob konfigurace nastavení sítě pomocí záměrů definovaných přes CLI nebo grafické rozhraní.
- **L2 switch** (*odl-l2switch-switch-ui*) – poskytuje základní funkcionalitu přepínače včetně implementace protokolu STP.
- **Messaging for transport** (*odl-mdsal-all*, *odl-messaging4transport-api*, *odl-messaging4transport*) – vrstva umožňující komunikaci softwarovým komponentám vyvinutých na různých platformách.
- **NetIDE** (*odl-netide-rest*) – komplexní prostředí pro konfiguraci sítě, vývoj aplikací a integraci aplikací z jiných SDN kontrolérů (RYU, Floodlight, Pyretic).
- **Network Intent Composition** (*odl-nic-core-mdsal*, *odl-nic-console*, *odl-nic-listeners*) – umožňuje spravovat síť pomocí záměrů (viz GBP).
- **NeMo** (*odl-nemo-openflow-renderer*, *odl-nemo-engine-ui*) – Network Modeling je jazyk umožňující popis síťových požadavků deklarativním způsobem (obdoba záměrů). Aplikace mohou žádat o síťové zdroje, služby a operace.
- **Neutron and OVSDB** (*odl-ovsdb-openstack*) – modul slouží pro zjednodušení integrace s frameworkem OpenStack Neutron.
- **SFC** (*odl-sfc-core*, *odl-sfc-ui*, *odl-sfc-netconf*, *odl-sfc-test-consumer*, *odl-sfc-ovs*, *odl-sfcofl2*) – Service Function Chaining umožňuje vytvoření posloupnosti akcí pro konkrétní zprávu.
- **Time Series Data Repository** (*odl-tsdr-hsqldb-all*) – poskytuje dvě databáze (NoSQL, MySQL) pro sběr statistik ze zařízení a služby pro automatický sběr těchto statistik (včetně protokolu OpenFlow).
- **Topology processing framework** (*odl-topoprocessing-framework*) – poskytuje náhled na síťovou topologii.
- **User network interface manager** (*odl-unimgr*) – umožňuje konfiguraci síťových zařízení

na hranici mezi zákazníkem a poskytovatelem připojení.
- **VPN** (*odl-vpnservice-core*) – Virtual Private Network umožňuje vytvořit L3 VPN založenou na BGP-MP.
- **VTN** (*odl-vtn-manager-rest*) – Virtual Tenant Network umožňuje vytvoření logické sítě nezávisle na fyzické topologii.
- **YANG PUBSUB** (*odl-yangpush-rest*) – nástroj umožňuje nastavení sběru statistik a údajů z různých zdrojů.

5.3.2 Podporované protokoly

OpenDaylight dále podporuje velké množství *southbound* protokolů (jejich oficiální názvy, podle kterých je lze aktivovat, jsou opět uvedeny v závorkách):

- **CAPWAP** (*odl-capwap-ac-rest*) – Control And Provisioning of Wireless Access Points je protokol umožňující správu přístupových bodů. Protokol poskytuje REST rozhraní, které mohou využívat externí aplikace.
- **BGP** (*odl-bgpcep-bgp*) – Border Gateway Protocol slouží pro směrování mezi autonomními systémy, ale lze ho využít i pro SDN funkcionalitu. Poskytuje konfiguraci přes REST rozhraní.
- **DLUX** (*odl-dlux-all*) – OpenDaylight User Experience je grafické prostředí kontroléru pro správu SDN.
- **IoTDM** (*odl-iotdm-onem2m*) – Internet of Things Data Management je middleware zprostředkující komunikaci mezi různými druhy IoT zařízení. Obsahuje několik dílčích protokolů: CoAP (Constrained Application Protocol), MQTT (MQ Telemetry Transport) a HTTP (Hypertext Transfer Protocol).
- **LACP** (*odl-lacp-ui*) – Link Aggregation Control Protocol zajišťuje automatické zjišťování sítě pomocí síťových prvků předávajících LACP zprávy. Protokol také umožňuje logickou agregaci fyzických spojů podle specifikace 802.3ad.
- **NETCONF** (*odl-netconf-mdsal*, *odl-netconf-connector*, *odl-netconf-topology*, *odl-restconf*) – Network Configuration Protocol umožňuje správu síťových zařízení pomocí příjmu a odesílání konfiguračních zpráv. Samotná komunikace používá formát XML (eXtensible Markup Language).
- **OF-CONFIG** (*odl-of-config-all*) – umožňuje spravovat zařízení podporující OpenFlow protokol. Je také součástí NETCONF protokolu.
- **OVSDB** (*odl-ovsdb-ui*) – Open vSwitch Database je protokol pro správu SDN prvků a typicky se používá jako doplněk protokolu OpenFlow. OVSDB je v takovém případě použit pro základní konfiguraci prvků (směrování, zabezpečení, atd.) a OpenFlow pak zajišťuje SDN funkcionalitu.
- **SNBI** (*odl-snbi-all*) – Secure Network Bootstrapping Interface spravuje zabezpečené spojení mezi zařízeními a poskytuje kontroléru data pro vytvoření obrazu fyzické síťové topologie.
- **SNMP** (*odl-snmp-plugin*) – Simple Network Management Protocol umožňuje kompletní konfiguraci zařízení a bývá podporován prakticky všemi síťovými prvky. Často se proto používá na starších síťových prvcích nepodporujících protokol OpenFlow.
- **Source Group Tag eXchange Protocol** (*odl-sxp-api*, *odl-sxp-core*, *odl-sxp-controller*) – je protokol zajišťující výměnu informací vazeb mezi IP adresami a SGT (Source Group Tag). Označení SGT umožňuje rozdělení koncových bodů do logických skupin, pro která mohou

být jednoduše aplikována jednotná síťová pravidla.

- **LISP** (*odl-lispflowmapping-msmr*) – The Locator/ID Separation Protocol je od roku 2013 volně dostupný protokol (předtím proprietarní protokol firmy Cisco), který logicky rozděluje IP adresu na dvě části: lokační a identifikační.
- **USC** (*odl-usc-channel-ui*) – Unified Secure Channel umožňuje vytvoření zabezpečeného spojení mezi kontrolérem a síťovými prvky pomocí protokolu TLS, případně DTLS.

5.3.3 Instalace kontroléru

Kontrolér je napsán jako software pro JVM a může být spuštěn na libovolném operačním systému podporujícím jazyk Java. Kontrolér využívá nástroj *Maven* pro sestavení aplikace a *OSGi framework*[9] pro dynamické načítání JAR souborů.

Prvním krokem je instalace JDK obsahující JVM. Samotná instalace probíhá rozbalením stažených souborů. Volitelně je doporučeno nastavení proměnné *JAVA_HOME*.

```
# Instalace JDK
sudo apt-get install openjdk-8-jdk

# Rozbalení staženého kontroléru (verze Neon)
tar zxvf opendaylight-0.10.1.tar.gz

# Nastavení proměnné JAVA_HOME (volitelný krok)
# 1) Zjištění cesty k nainstalovanému JDK
sudo update-alternatives --config java

# 2) Editace konfiguračního souboru
sudo nano /etc/environment

# 3) Přídání následujícího řádku do souboru "environment"
JAVA_HOME=‘‘/usr/lib/jvm/java-8-openjdk-amd64‘‘

# 4) Aplikace nastavení proměnné
source /etc/environment
```

5.3.4 Spuštění kontroléru

Jednotlivé funkcionality a podporované protokoly ve výchozím stavu nejsou nainstalovány. Po instalaci kontroléru je proto nutné pomocí dodávaného nástroje *Karaf*[10] funkce doinstalovat. Nástroj zajišťuje konzistentní instalaci všech provázaných balíčků. Instalace se provádí následujícími příkazy (nástroj se nachází ve složce s distribucí kontroléru):

```
# Spuštění aplikačního kontejneru Karaf:
.bin/karaf

# Výpis všech dostupných funkcionalit:
feature:list
```

[9] https://www.osgi.org/developer/architecture/

[10] https://wiki.opendaylight.org/view/Controller_Core_Functionality_Tutorials:Tutorials:Starting_A_Project:ch0

```
# Instalace funkcionalit:
feature:install [funkcionalita1] [funkcionalita2] ... [funkcionalita n]

# Ověření nainstalovaných funkcionalit:
feature:list -i

# Vymazání veškerých doinstalovaných funkcionalit:
bin/start clean
```

Nevýhodou tohoto přístupu je fakt, že nástroj nepodporuje odinstalaci jednotlivých funkcionalit a jedinou možností je odinstalování všech funkcionalit najednou. Tím dojde k navrácení kontroléru do výchozího stavu. Měly by být instalovány pouze skutečně vyžadované funkcionality – každá běžící funkcionalita zvyšuje hardwarové nároky kontroléru. Při instalaci funkcionality není nutné kontrolér restartovat a instalace tedy může být provedena za běhu kontroléru.

5.3.5 Grafické rozhraní

Grafické rozhraní kontroléru je zajištěno modulem DLUX (OpenDaylight User Experience). Ten je nutné doinstalovat. Základní verze (instalace příkazem *odl-dlux-core*) obsahuje pouze přehled síťové topologie. Další aplikace (v prostředí implementované v odlišných záložkách) se instalují jednotlivě, případně všechny najednou příkazem *odl-dlux-all*. Každá doinstalovaná funkcionalita pak obvykle v grafickém rozhraní vytvoří novou záložku se svým jménem, ve které je pak možné funkcionalitu spravovat. Aktuálně dostupné příkazy pro instalaci separátních vlastností modulu DLUX jsou:

- **odl-dlux-core** – základní stránka obsahující pouze náhled na síťovou topologii.
- **odl-dlux-all** – nainstaluje všechny dostupné aplikace funkcionality DLUX.
- **odl-dluxapss-nodes** – zobrazuje síťové statistiky a informace o portech připojených síťových prvků.
- **odl-dluxapps-topology** – zobrazuje grafickou reprezentaci síťové topologie, která obsahuje síťové prvky i koncová zařízení.
- **odl-dluxapps-yangui, odl-dluxapps-yangvisualizer, odl-dluxapps-yangman** – nainstaluje nástroj YANG, který umožňuje správu datového skladu MD-SAL.

Samotné rozhraní je ve výchozím stavu dostupné na adrese:
`http://<IP_ADRESA>:8181/index.html#/login`

5.3.6 Přístupové údaje

Přístup ke kontroléru vyžaduje zadání uživatelského jména a hesla. Ta jsou ve výchozím stavu nastavena na *admin*. Změnu přihlašovacích údajů za běhu kontroléru je možné provést dvěma způsoby:

1. Pomocí příkazové řádky nástroje Karaf.
2. Přes REST rozhraní.

Změna údajů pomocí **příkazové řádky KARAF** je jednoduchý a rychlý způsob. Jediným požadavkem je nutná instalace funkcionality AAA CLI. Kompletní postup je následující:

```
# 1) Instalace funkcionality AAA CLI
feature:install odl-aaa-cli
```

```
# 2) Změna přihlašovacích údajů pro uživatele "admin"
aaa:change-user-pwd -user admin

# Odpověď kontroléru:
Enter current password:
Enter new password:
admins password has been changed
```

K REST rozhraní je možné přistoupit nástrojem *curl* generujícím HTTP dotaz. Následující příklad ilustruje změnu hesla uživatele *admin*.

```
# 1) Získání údajů všech uživatelů
# (admin:admin určuje aktuálního uživatele - výchozí hodnoty)
curl -u admin:admin http://localhost:8181/auth/v1/users

# Seznam uživatelů obsahuje následující parametry:
# name, description, enabled, email, password

# 2) Vytvoření JSON souboru s novými údaji uživatele
cat user.json
{
 "name":"admin",
 "description":"admin account",
 "enabled":"true",
 "email":"",
 "password":"newadminpassword"
}

# 3) Poslání vytvořeného JSON souboru na kontrolér
curl -u admin:admin -X PUT -H "Content-Type: application/json"
 --data-binary @./user.json http://localhost:8181/auth/v1/users/1
```

Postupy včetně návodu pro změnu údajů při vypnutém kontroléru je možné najít v dokumentaci: `https://wiki.opendaylight.org/view/AAA:Changing_Account_Passwords`

Více informací o architektuře a používání kontroléru OpenDaylight je možné najít v práci *Analýza kontrolérů softwarově definovaných sítí* [3, str. 30].

5.4 ONOS

ONOS (Open Network Operating System) je další pokročilý kontrolér vhodný pro produkční nasazení. Je zaměřen zejména na sítě operátorů a poskytovatelů připojení. ONOS bývá také označován jako síťový operační systém nebo platforma SDN kontroléru. ONOS poskytuje hlavně následující funkce:

- **Vysoká dostupnost a odolnost** – minimalizace výpadků je zajištěna distribuovanou architekturou a nasazením na několika separátních zařízení ve formě tzv. *clusterů*.
- **Škálovatelný výkon** – kontrolér umožňuje dynamicky přidávat nové instance, které jsou zodpovědné za správu určité části sítě. V jakémkoliv režimu nasazení je pak zaručena maximální odezva na síťové události 50 ms.
- **Modulární software** – kontrolér je dodáván se 135 různými rozšířeními, které umožňují vysokou přizpůsobitelnost kontroléru.

- **Velké množství podporovaných protokolů** – to zahrnuje *northbound* i *southbound* protokoly.
- **Podpora nástroje YANG** – ten umožňuje abstraktní správu sítě přes definované modely vytvořené nástrojem YANG, případně s použitím formátů JSON a XML.

První verze kontroléru byla vydána v roce 2014 a nové verze kontroléru jsou od té doby vydávány v pravidelných tříměsíčních intervalech. Pro poslední dvě vydané verze pak vychází pravidelné aktualizace a bezpečnostní záplaty (kritické záplaty bývají dostupné i pro starší verze).

Aktuální verze kontroléru *2.1.0 Raven* pochází z dubna 2019 a je dostupná ke stažení: `https://wiki.onosproject.org/display/ONOS/Downloads`

Oblibu nasazení v kritických sítích vyžadujících vysokou spolehlivost si kontrolér získal především dodávanou platformou CORD (Central Office Re-architected as a Datacenter). Ta umožňuje transformaci klasických centrál na agilní datová centra pomocí několika moderních technologií: SDN, NFV a cloudu.

Druhou oblíbenou vlastností kontroléru je podpora širokého množství různých typů sítí (včetně optických) a jejich jednotné správy.

Stejně jako u kontroléru OpenDaylight, ani ONOS nerozlišuje mezi termíny aplikace a modul. Z toho důvodu jsou tak tyto vlastnosti opět nazývány obecně **funkcionality**.

5.4.1 Podporované funkcionality

Kontrolér ONOS je dodáván s velkým množstvím funkcionalit, ale pouze některé jsou ve výchozím stavu nainstalované. Konkrétní funkcionality závisí na verzi kontroléru, ale mezi ty nejdůležitější patří:

- **Access Control List** (*org.onosproject.acl*) – umožňuje základní filtrování provozu.
- **Castor** (*org.onosproject.castor*) – umožňuje nastavit tzv. *peering* mezi autonomními systémy. Jedná se tak o alternativu využívání připojení poskytovatele.
- **Control Plane Manager** (*org.onosproject.cpman*) – poskytuje informace o stavu kontroléru jako je vytížení jednotlivých serverů, statistiky o provozu a další.
- **DHCP Relay Agent** (*org.onosproject.dhcprelay*) – umožňuje přeposílání DHCP dotazů na DHCP server nacházející se v jiné síti.
- **DHCP Server** (*org.onosproject.dhcp*) – umožňuje automatické přidělování IP adres, adres DNS serverů a výchozí brány.
- **Event History** (*org.onosproject.events*) – umožňuje zobrazit historii událostí kontroléru.
- **Fault Management** (*org.onosproject.faultmanagement*) – slouží pro správu chybových hlášení přijatých od jednotlivých síťových prvků.
- **Ganglia Report and Query** (*org.onosproject.gangliametrics*) – zajišťuje přeposlání veškerých událostí zpracovaných kontrolérem na server Ganglia.
- **Host Mobility** (*org.onosproject.mobility*) – automaticky maže veškeré flow záznamy síťového zařízení, které bylo odstraněno či přesunuto.
- **InfluxDB Report and Query** (*org.onosproject.influxdbmetrics*) – zajišťuje přeposlání veškerých událostí zpracovaných kontrolérem na server InfluxDB.
- **Link Discovery Provider** (*org.onosproject.linkdiscovery*) – zajišťuje sledování stavu topologie.

- **Mastership Load Balancer** (*org.onosproject.mlb*) – zajišťuje vyvážení připojených síťových zařízení na příslušné kontroléry typu *master*.
- **Multicast Forwarding** (*org.onosproject.mfwd*) – umožňuje reaktivní směrování vícecestné komunikace typu *multicast*.
- **Network Config Host Provider** (*org.onosproject.netcfghostprovider*) – umožňuje povolení a zakázání komunikace jednotlivých koncových zařízení a získání jejich informací.
- **Network Config Link Provider** (*org.onosproject.netcfglinksprovider*) – umožňuje povolit a zakázat jednotlivé síťové spoje.
- **Packet/Optical** (*org.onosproject.optical-model*, *org.onosproject.newoptical*) – umožňuje sjednotit správu optické a klasické síťové topologie.
- **Path Visualization** (*org.onosproject.pathpainter*) – v grafickém zobrazení topologie barevně rozlišuje různé typy spojů.
- **Protocol Independent Multicast Emulation** (*org.onosproject.pim*) – umožňuje efektivní směrování komunikace typu *multicast* pomocí protokolů PIM.
- **Proxy ARP/NDP** (*org.onosproject.proxyarp*) – zajišťuje funkcionalitu ARP a NDP komunikace mezi zařízeními z různých lokálních sítí (adresa je nahrazena adresou síťového prvku).
- **Reactive Forwarding** (*org.onosproject.fwd*) – zajišťuje klasické směrování komunikace pomocí logiky L2 přepínače.
- **SDN-IP** (*org.onosproject.sdnip*) – umožňuje propojení SDN sítě a sítě používající BGP.
- **SDN-IP Reactive Routing** (*org.onosproject.reactive-routing*) – zajišťuje IPv4 a IPv6 směrování uvnitř SDN sítě. Pro komunikaci vně sítě slouží funkcionalita *SDN-IP*.
- **Scalable Gateway** (*org.onosproject.scalablegateway*) – umožňuje správu více výchozích bran pro vyvažování zátěže a vysokou dostupnost.
- **Segment Routing** (*org.onosproject.segmentrouting*) – nový typ směrování založený na sekvenčním zpracování instrukcí vkládaných do síťových prvků. Funkcionalitu lze kombinovat s MPLS.
- **VLAN L2 Broadcast Network** (*org.onosproject.vpls*) – umožňuje vytvoření (potenciálně překrývajících se) L2 *broadcast* domén, ve kterých mohou koncové zařízení společně komunikovat.
- **Virtual Broadband Gateway** (*org.onosproject.virtualbng*) – privátním zařízením umožňuje internetovou konektivitu pomocí mapování veřejných adres na privátní.
- **Virtual Router** (*org.onosproject.vrouter*) – zajišťuje funkcionalitu L3 směrovače.
- **YANG Management System** (*org.onosproject.yms*) – umožňuje správu funkcionalit pomocí nástroje YANG.

Mezi další funkcionality patří: Cluster IP alias (*org.onosproject.cip*), Driver Support Matrix (*org.onosproject.drivermatrix*), Flow Space Analysis (*org.onosproject.flowanalyzer*), Flow Throughput Demo (*org.onosproject.demo*), Graphite Report and Query (*org.onosproject.graphitemetrics*), Null Southbound Provider (*org.onosproject.null*), OpenStack (*org.onosproject.openstackinterface*, *org.onosproject.metrics*, *org.onosproject.openstacknetworking*, *org.onosproject.openstacknode*, *org.onosproject.openstackrouting*, *org.onosproject.openstackswitching*), Service function chaining (*org.onosproject.vtn*) a XOS Client (*org.onosproject.xosclient*).

Dále jsou dodávané různé funkcionality pro provádění testů.

5.4.2 Podporované protokoly

Kontrolér ONOS podporuje následující *southbound* protokoly (některé byly popsány v předchozích kapitolách). V případě potřeby je opět nutné je doinstalovat a aktivovat.

- **BGP** (*org.onosproject.bgp*, *org.onosproject.bgprouter*) – Border Gateway Protocol pro směrování mezi autonomními systémy lze využít pro nastavení specifického směrování.
- **ISIS** (*org.onosproject.isis*) – Intermediate System to Intermediate System je směrovací protokol pro použití v rámci autonomního systému. Kontrolér pomocí jeho informací může zjišťovat stav topologie.
- **NETCONF** (*org.onosproject.netconf*) – Network Configuration Protocol slouží zejména pro základní nastavení síťových prvků.
- **OVSDB** (*org.onosproject.ovsdb-base*, *org.onosproject.ovsdb*, *org.onosproject.ovsdbhostprovider*) – Open vSwitch Database slouží pro nastavení síťových prvků, nejčastěji v kombinaci s použitím protokolu OpenFlow.
- **PCEP** (*org.onosproject.pcep-api*, *org.onosproject.pcep*) – Path Computation Element Protocol poskytuje klíčovou funkcionalitu pro protokoly MPLS a GMPLS[11], kterým dodává data pro výpočet jednotlivých směrovacích cest.
- **SNMP** (*org.onosproject.snmp*) – Simple Network Management Protocol umožňuje monitorování a správu prakticky všech síťových prvků.
- **LISP** (*org.onosproject.lisp*) – The Locator/ID Separation Protocol je směrovací protokol využívající specifické směrování pomocí rozdělení IP adresy na dvě části.

5.4.3 Instalace kontroléru

Kontrolér má podobnou architekturu jako kontrolér OpenDaylight. Je také napsán v programovacím jazyce Java a pro načítání aplikačních balíčků využívá nástroj Karaf. Instalace samotných modulů a jejich dynamické spouštění má pak na starosti modulární systém OSGi. Instalace kontroléru vyžaduje JVM a dále je doporučeno nastavení proměnné prostředí *JAVA_HOME*. Tyto dva kroky jsou stejné jako u kontroléru OpenDaylight a postup jejich instalace a nastavení byl popsán v kapitole 5.3.3.

Samotná instalace kontroléru na operačním systému Linux[12] probíhá následovně:

```
# 1) Vytvoření standardního uživatele (bez práv root) - vyžadováno jméno "sdn"
sudo adduser sdn --system -group

# 2) Vytvoření adresáře v kořenovém umístění (/opt)
sudo mkdir opt

# 3) Rozbalení staženého balíčku ONOS (dané verze)
sudo tar xzf onos-VERZE.tar.gz
```

5.4.4 Spuštění kontroléru

Spuštění kontroléru probíhá příkazem (spuštěným v adresáři kontroléru):

```
sudo bin/onos-service start
```

[11] *Generalized MPLS* je rozšířená verze protokolu MPLS, která přidává hlavně podporu více typů přepínání.
[12] Linux je doporučený operační systém pro instalaci, ale podporovány jsou i jiné operační systémy.

Podle oficiální dokumentace nevyžaduje spuštění kontroléru práva uživatele root, nicméně bez nich nemusí dojít k navázání spojení se síťovými prvky.

Jednotlivé funkcionality mohou nabývat tří stavů: nainstalovány, nainstalovány a aktivovány, nebo kompletně odstraněny. Nainstalované aplikace, které nejsou aktivovány, jsou pouze umístěné v adresáři kontroléru, ale nejsou spouštěny při jeho startu.

Ve výchozím stavu kontrolér nabízí několik nainstalovaných funkcionalit, ale žádná (kromě základních ovladačů) není aktivována. Správa jednotlivých funkcionalit je možná přes CLI kontroléru, případně přes webové rozhraní a záložku *Applications*. V té je možné nahrát soubor s funkcionalitou (formát souboru *.oar*) a tu pak aktivovat. Podporováno je i odstranění funkcionality. Veškeré tyto úkony je možné provádět bez nutnosti restartování kontroléru.

CLI umožňuje stejnou funkcionalitu následujícími příkazy:

```
# Aktivace funkcionality
app activate org.onosproject.FUNKCIONALITA

# Deaktivace funkcionality
app deactivate org.onosproject.FUNKCIONALITA

# Odstranění funkcionality
app uninstall org.onosproject.FUNKCIONALITA

# Výpis nainstalovaných funkcionalit (aktivované jsou označeny hvězdičkou)
apps -s

# Výpis pouze aktivovaných funkcionalit
apps -s -a
```

5.4.5 Grafické rozhraní

Grafické rozhraní kontroléru je ve výchozím stavu aktivované a dostupné na adrese:
`http://<IP_ADRESA>:8181/onos/ui/index.html`

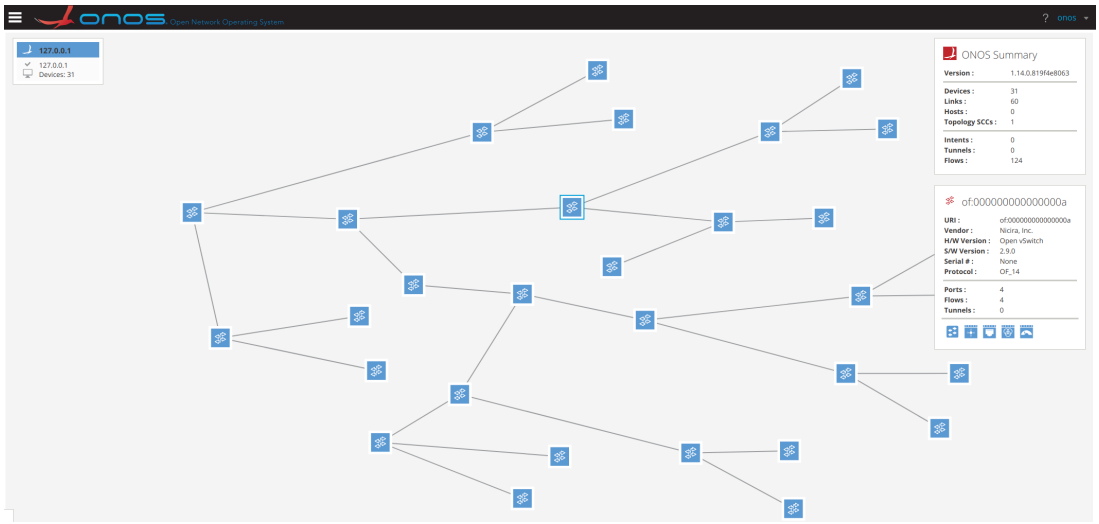
Grafické rozhraní poskytuje oproti předchozím kontrolérům určité unikátní vlastnosti. Například vizualizace topologie umožňuje mimo jiné zobrazení fyzického rozložení prvků v podobě geografické mapy. Dále je umožněno ruční přemístění těchto prvků, sledování aktuálního objemu dat na daných spojích a v případě aktivování funkcionality *Path Visualisation* i barevné odlišení těchto spojů.

5.4.6 Přístupové údaje

Přístup do prostředí kontroléru vyžaduje zadání přihlašovacích údajů. Ve výchozím stavu je jméno uživatele nastaveno na *onos* a heslo na *rocks*. Změna uživatelských údajů je možná úpravou konfiguračního souboru: */opt/onos/apache-karaf-3.0.5/etc/users.properties*

Soubor obsahuje uživatelská jména a hesla, která jsou uložena v čitelné podobě. Z tohoto důvodu je nutné zajistit odpovídající zabezpečení souboru. Aby se změny v souboru projevily, je bohužel nutné kontrolér restartovat – na rozdíl od kontroléru OpenDaylight.

Více informací o architektuře a používání kontroléru ONOS je možné najít v práci *Analýza kontrolérů softwarově definovaných sítí* [3, str. 53].



Obrázek 5.2: Grafické rozhraní kontroléru ONOS

6. Mininet

Tato kapitola přechází z teoretického popisu konceptů SDN do praktické oblasti. Informace zde uvedené by mělo být možné použít i bez znalosti principů popsaných v předchozích kapitolách. Některé informace se proto mohou opakovat. Hlavním cílem kapitoly je jednoduchým způsobem představit možnost vytvořit si vlastní emulovanou síťovou topologii, na které je možné vykoušet si SDN včetně programování vlastních aplikací.

6.1 Představení

Mininet je síťový emulátor, umožňující emulaci kompletní počítačové sítě i na zařízeních s poměrně nízkým výpočetním výkonem. I na většině průměrných přenosných počítačích lze v rámci emulátoru používat komplexní a rozsáhlé síťové topologie obsahující desítky až stovky prvků. Mininet podporuje vytvoření topologie obsahující softwarové přepínače, SDN kontrolér a virtualizovaná koncová zařízení. Tyto prvky sdílí jádro operačního systému a emulace je proto velice nenáročná na hardwarové zdroje. Omezením je pouze fakt, že vyžaduje operační systém Linux. Mininet však může být snadno nasazen ve virtualizovaném počítači a tím pádem ho lze používat na libovolném operačním systému.

Mininet v aktuální verzi 2.3 je možné volně stáhnout z platformy GitHub:
`https://github.com/mininet/mininet`

Výkon emulátoru

Tradiční pojetí emulátorů předpokládá konzistentní výkon nezávislý na platformě a hardware, na kterém je experiment spuštěn. V případě emulátoru Mininet to však neplatí. Z důvodu sdílení jádra operačního systému mezi hostitelským počítačem a virtualizovanými koncovými prvky je celkový výkon silně závislý na použitém hardware a dále pak na velikosti emulované topologie. S rostoucím počtem emulovaných prvků dochází ke snížení měřeného výkonu (propustnost, odezva). Více informací o této problematice lze nalézt v práci *Using SDN to Enhance IoT Security* [2, str. 160].

6.1.1 Integrované přepínače

Mininet podporuje emulaci několika různých OpenFlow softwarových přepínačů. Ty mají různé funkce a podporují odlišné verze OpenFlow protokolu. V závislosti na zvoleném přepínači pak lze využít různou SDN funkcionalitu. V současné verzi nástroje Mininet jsou podporované následující softwarové přepínače (v závorkách je uveden jejich název použitý v emulátoru):

- **OpenFlow reference switch (user)** – základní softwarový přepínač implementovaný v uživatelském režimu. Jedná se o výchozí přepínač, který je použit, pokud není parametrem definovaný jiný.
- **Open vSwitch (ovsk)** – celosvětově nejpoužívanější softwarový přepínač, který je implementován v režimu jádra OS (kernel). Z toho důvodu poskytuje vyšší výkon než přepínače implementované v uživatelském režimu. Je také součástí základní instalace Mininet.
- **Basic OpenFlow Software Switch (user)** – také označovaný jako BOFUSS nebo původně *ofsoftswitch13*, případně *of13softswitch*. Jedná se o přepínač pracující v uživatelském režimu. Ve výchozím stavu emulátoru není nainstalován. Lze ho však doinstalovat příkazem:

  ```
  install.sh -3f
  ```

  V tomto případě dojde k nahrazení *OpenFlow reference* přepínače právě tímto přepínačem. Parametr *user* tedy spustí přepínač *Basic OpenFlow Software Switch*[1].
- **Indigo Virtual Switch (IVSSwitch)** – softwarový přepínač vyvinutý v rámci projektu *Floodlight*. Se stejnojmenným SDN kontrolérem sdílí knihovní funkce (*LoxiGen*) a jeho hlavní funkcionalita implementovaná v modulu kernel, je převzatá z přepínače Open vSwitch. Přepínač není ve výchozím stavu nainstalován a lze ho doinstalovat příkazem:

  ```
  install.sh -i
  ```

6.1.2 Integrovaný SDN kontrolér

Základní instalace emulátoru Mininet obsahuje *OpenFlow reference controller*, který umožňuje pouze základní síťovou konektivitu. Většina předpřipravených obrazů virtuálních počítačů pro práci s SDN (popsány v kapitole 6.2.3) proto obsahuje pokročilejší SDN kontroléry, mezi kterými je možné si vybrat ten nejvhodnější. Typicky se jedná o NOX, POX, RYU a Floodlight.

Pokročilejší SDN kontroléry pak nejsou automaticky spravovány emulátorem Mininet, ale jsou spouštěny mimo vlastní emulovanou síť. Emulátor však poskytuje volbu *remote*, která umožňuje připojení k těmto kontrolérům. Kontrolér tak může být spuštěn na stejném zařízení, nebo i na jiném zařízení v síti (za předpokladu funkční konektivity).

6.2 Instalace a aktualizace emulátoru

6.2.1 Instalace

Instalaci emulátoru v prostředí OS Linux lze provést dvěma způsoby – ze zdrojových kódů, nebo v případě použití operačního systému Ubuntu přímo z oficiálních balíčků repozitáře.

[1] `https://github.com/mininet/mininet/wiki/FAQ`

Instalace ze zdrojových kódů

Instalace emulátoru podporuje následující možnosti:

- **-a** – provede instalaci emulátoru se všemi dodávanými funkcemi včetně kontroléru POX, podpory nástroje Wireshark[2] a softwarového přepínače Open vSwitch. Instalace bude provedena do domovského adresáře uživatele.
- **-nfv** – provede pouze instalaci emulátoru a dvou softwarových přepínačů: *OpenFlow reference switch* a *Open vSwitch*. Instalace bude provedena do domovského adresáře uživatele.
- **-s ADRESAR** – tato možnost umožňuje specifikaci vlastního adresáře pro instalaci. Lze ji kombinovat s předchozími parametry.

Instalaci nástroje Mininet ze zdrojových kódů stažených z platformy GitHub lze provést následujícími příkazy:

```
# Stažení zdrojových kódů:
git clone git://github.com/mininet/mininet

# Spuštění instalace:
mininet/util/install.sh [volby]

# Zobrazení dostupných možností a popis jednotlivých voleb:
install.sh -h

# Otestování úspěšné instalace:
sudo mn --test pingall
```

Instalace z balíčků repozitáře Ubuntu

Operační systém Ubuntu alternativně nabízí instalaci nástroje Mininet přímo z jejich oficiálního repozitáře. Tato volba však nemusí nutně nainstalovat nejnovější verzi nástroje, ale instalace je oproti první možnosti jednodušší. Spustit ji lze pouhým příkazem:

```
# Spuštění instalace z balíčku repozitáře:
sudo apt-get install mininet
```

6.2.2 Aktualizace

Aktualizace nástroje Mininet se liší podle typu instalace. Pro instalaci ze zdrojových kódů probíhá následujícím způsobem:

```
# Přechod do adresáře s nainstalovaným emulátorem
cd mininet

# Stažení aktuálních zdrojových kódů
git fetch

# Výběr hlavní verze emulátoru
# (případně je možné vybrat konkrétní verzi - například 2.2.1)
git checkout master
```

[2] https://www.wireshark.org/

```
# Stažení aktualizovaných souborů
git pull

# Start nové instalace
sudo make install
```

V případě instalace z repozitáře Ubuntu probíhá aktualizace nástroje následovně:

```
# Odstranění předchozí verze nástroje Mininet včetně softwarového přepínače
# Open vSwitch
sudo rm -rf /usr/local/bin/mn /usr/local/bin/mnexec \
    /usr/local/lib/python*/*/*mininet* \
    /usr/local/bin/ovs-* /usr/local/sbin/ovs-*

# Nová instalace z balíčku repozitáře:
sudo apt-get install mininet
```

6.2.3 Předinstalované virtuální počítače

Alternativa k výše uvedeným možnostem instalace je využití předpřipraveného obrazu virtuálního počítače, ve kterém je již emulátor Mininet nainstalován. Součástí takového obrazu pak typicky bývají další užitečné nástroje pro vývoj SDN aplikací. Mezi takové patří Wireshark, vývojová prostředí, pokročilé SDN kontroléry a testovací nástroje. Výhodou je nezávislost na použitém OS a okamžitá funkčnost bez nutnosti instalace a konfigurace. Nevýhodou pak bývá zejména přítomnost starších verzí přiložených nástrojů – u těch je často nutné provést aktualizaci.

Mininet VM

Oficiální obraz virtuálního počítače nástroje Mininet je dostupný ve formátu *.ovf* pro platformy VirtualBox[3] a Hyper-V[4]. Obraz obsahuje operační systém Ubuntu 14.04 LTS a Mininet verze 2.2.2 a je dostupný v 32bitové i 64bitové verzi.

Obraz virtuálního počítače *Mininet VM* je volně dostupný ke stažení na webu:
`https://github.com/mininet/mininet/wiki/Mininet-VM-Images`

All-in-one SDN App Development Starter VM

Alternativní obraz virtuálního počítače *All-in-one SDN App Development Starter VM* je založený na stejné platformě (Ubuntu 14.04 LTS) a je dodáván ve formátu *.ova*. Ten je možné importovat v nástrojích VirtualBox a VMware Player[5]. Obraz mimo jiné obsahuje pokročilé SDN kontroléry OpenDaylight a ONOS a vývojové prostředí *Eclipse Luna*, ve kterém je integrován kontrolér Floodlight. Obraz je dostupný v 32bitové i 64bitové verzi.

Obraz virtuálního počítače *All-in-one SDN App Development Starter VM* lze stáhnout na webu:
`http://sdnhub.org/tutorials/sdn-tutorial-vm/`

[3] `https://www.virtualbox.org/`
[4] `https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview`
[5] `https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html`

6.3 Spuštění

6.3.1 Základní spuštění

Nainstalovaný emulátor Mininet lze spustit z terminálového prostředí příkazem:

```
sudo mn
```

Příkaz vyžaduje *root* oprávnění (*sudo*). Příkaz *mn* bez dodatečných parametrů zajistí spuštění pouze základní topologie – *minimal* (popsané v kapitole 6.3.2). Podrobnější spuštění lze ovlivnit dodatečnými parametry popsanými níže. Seznam dostupných parametrů lze zobrazit parametrem *-h*:

```
sudo mn -h
```

6.3.2 Parametry spouštění

Předpřipravené síťové topologie

Mininet pro snadné a rychlé použití obsahuje několik předpřipravených topologií. Topologii lze vybrat přepínačem *--topo* s parametrem názvem vybrané topologie. Ta může dále obsahovat dodatečné parametry.

Předpřipravené topologie jsou následující (*n* značí konfigurovatelný parametr ve formátu celého čísla):

- **Minimal** – skládá se z jednoho přepínače a dvou přímo připojených hostů.
- **Single** – obsahuje jeden přepínač a *n* k němu připojených hostů.
- **Linear** – obsahuje *n* sériově propojených přepínačů. Ke každému přepínači je navíc připojen jeden host.
- **Tree** – stromová topologie, kde *n* značí počet *pater stromu* (hloubku). Každý přepínač počínaje *kořenem* stromu obsahuje tři podřazené přepínače (pokud je nižší patro dostupné v rámci definované hloubky). Ke každému přepínači v nejnižším definovaném patře jsou pak připojeni tři hosté.

Příklad spuštění lineární topologie o velikosti 3 (obsahující 3 přepínače a 3 hosty):

```
sudo mn --topo linear,3
```

Spuštění vlastní síťové topologie

Mininet kromě předpřipravených topologií podporuje možnost vytvořit si topologii vlastní. To je možné několika způsoby popsanými v kapitole 6.4. Spuštění vlastní topologie z připraveného skriptu probíhá příkazem:

```
# VLASTNI_TOPO.py uvádí název skriptu s vlastní topologií a příklad cesty
# NAZEV_TOPO je alias odkazující na třídu s topologií v souboru VLASTNI_TOPO.py
sudo mn --custom ~/mininet/custom/VLASTNI_TOPO.py --topo NAZEV_TOPO
```

Připojení vlastního SDN kontroléru

Ve výchozím nastavení nástroje Mininet je spuštěna topologie s integrovaným SDN kontrolérem (*OpenFlow reference controller*). Proto i obyčejné spuštění příkazem *sudo mn* zajišťuje základní konektivitu mezi zařízeními. Tento kontrolér je však možné nahradit kontrolérem vlastním. Ten může běžet na lokálním počítači (případně ve virtuálním stroji), nebo kdekoliv v počítačové síti za předpokladu, že bude dostupný. Spuštění nástroje Mininet v režimu naslouchání pro připojení vlastního kontroléru probíhá příkazem:

```
sudo mn --controller=remote,ip=[IP_ADRESA],port=[VLASTNI_PORT]
```

Pro adresu lokálního počítače je možné použít *127.0.0.1*, nebo *0.0.0.0*. Defaultní nastavení portu v emulátoru Mininet (verze 2.2.1) je 6633. Doporučené číslo portu od verze OpenFlow 1.4 je však 6653.

Spuštění s jinými typy SDN přepínačů

Použitý typ přepínačů emulovaných v síťové topologii lze změnit parametrem *--switch*:

```
sudo mn --switch [TYP_PREPINACE]
```

Mezi podporované hlavní typy přepínačů patří *user* a *ovsk*. Další přepínače a jejich detailní popis jsou uvedeny v kapitole 6.1.1.

Další parametry

Mezi další užitečné parametry patří:

- **Jednoduchý formát MAC adres** – MAC adresy hostů nemusí být generovány náhodně, ale mohou mít pevně přiřazenou hodnotu podle ID zařízení. V tomto případě bude tedy MAC adresa koncového prvku číslo 1 rovna hodnotě *00:00:00:00:00:01*. Tento způsob přiřazování je vhodný hlavně pro ladění a snadnou identifikaci koncových prvků. Aktivace probíhá parametrem:

  ```
  --mac
  ```

  Parametr lze přidat do spouštěcího příkazu na libovolnou pozici.
- **Vyčištění emulátoru po nekorektním ukončení** – při nekorektním ukončení emulátoru, či jeho pádu, se může stát, že část jeho vlastností zůstane ve spuštěném stavu. To se často projeví chybou při pokusu o opětovné spuštění (typicky hlášení o již používaném portu). V tomto případě je nutné použít příkaz pro kompletní ukončení všech běžících operací příkazem:

  ```
  sudo mn -c
  ```

6.3.3 Prostředí Mininet

Úspěšné spuštění emulátoru je v prostředí terminálu indikováno znakovým řetězcem *mininet*>. V tomto prostředí lze používat pouze příkazy podporované nástrojem Mininet. Mezi nejdůležitější příkazy emulátoru patří:

```
mininet> help # Výpis všech dostupných příkazů
mininet> nodes # Výpis spuštěných zařízení v topologii, příklad:
# c0 h1 h2 s1 s2

mininet> net  # Výpis spuštěných zařízení včetně jejich portů a síťových
# propojení, příklad: h1 h1-eth0:s3-eth1

mininet> dump # Výpis topologie včetně typů zařízení, jejich IP adres a process
# ID, příklad:
# <Host h1: h1-eth0:10.0.0.1 pid=6243>
# <OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=6286>
# <Controller c0: 127.0.0.1:6633 pid=6236>

mininet> h1 ping h2 # Provede klasický ping mezi definovanými prvky
```

```
mininet> pingall # Skript postupně provede ping mezi každým koncovým prvkem
# v topologii, nedostupná spojení jsou označena písmenem X
mininet> exit # Ukončí emulátor
```

6.4 Tvorba vlastních topologií

Emulátor podporuje možnost vytváření vlastních topologií. Ty mají podobu skriptů ve formátu **.py* – jsou tedy napsané v jazyce Python. Celkem existují tři způsoby vytváření vlastních topologií: jednoduchý skript, spustitelný skript a spustitelný skript vytvořený grafickým nástrojem *MiniEdit*.

6.4.1 Jednoduchý skript vlastní topologie

Nejjednodušší způsob vytváření vlastních topologií je pomocí skriptu, který je vložen jako parametr příkazu *mn*. Tento způsob je vhodný pro základní topologie, které nevyžadují komplexnější nastavení jednotlivých zařízení nebo síťových spojů.

Mininet obsahuje ukázkovou topologii, která se nachází v souboru: *mininet/custom/topo-2sw-2host.py*. Následující kód pochází z tohoto souboru, ale pro názornost je zde doplněn o komentáře vysvětlující jeho funkcionalitu:

```
"""Příklad vlastní topologie
Topologie obsahuje dva přímo propojené přepínače. Ke každému přepínači je
připojen jeden host. Topologie odpovídá předpřipravené lineární topologii
s parametrem 2. Schéma topologie je následující:
   host --- přepínač --- přepínač --- host
"""

# Import knihovny pro vytváření topologií
from mininet.topo import Topo

# Vytvoření třídy obsahující kompletní topologii
class MyTopo( Topo ):

    def __init__( self ):
        # Konstruktor obsahující kompletní nastavení topologie

        # Inicializace topologie
        Topo.__init__( self )

        # Přidání koncových zařízení
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )

        # Přidání přepínačů
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Přidání síťových spojů mezi všemi zařízeními
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )
```

```
# Přidání páru klíč/hodnota do slovníku 'topos'
# Umožňuje spuštění topologie příkazem '--topo=mytopo' z prostředí terminálu
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Spuštění vlastní topologie uvedené v předchozím skriptu je možné následujícím příkazem:

```
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo
```

6.4.2 Spustitelný skript vlastní topologie

V případě složitějších topologií je vhodnějším způsobem vytvoření samostatného spustitelného skriptu. Ten obsahuje kompletní topologii včetně nastavení veškerých parametrů při spuštění (vlastní kontrolér, typy přepínačů, atd.). Skript má opět podobu **.py* souboru, ale spouští se přímo jako kterýkoliv jiný Python soubor. Příkaz *mn* se tedy v tomto případě nepoužívá.

Následující kód zobrazuje ukázkovou topologii:

```
#!/usr/bin/python

# Import knihoven (ne všechny jsou v tomto příkladu využity)
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from subprocess import call

# Knihovna pro nastavování parametrů spojů
from mininet.link import TCLink

# Definice vlastní třídy pro vytvoření sítě
def myNetwork():

    # Základní nastavení emulátoru
    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    # Nastavení SDN kontroléru
    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=RemoteController,
                         ip='127.0.0.1',
                         protocol='tcp',
                         port=6653)

    # Přidání a nastavení přepínačů (typ Open vSwitch)
    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
```

```
    # Přidání a nastavení koncových zařízení
    info( '*** Add hosts\n')
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)

    # Přidání síťových spojů mezi všemi zařízeními
    info( '*** Add links\n')
    net.addLink(h1, s1)
    net.addLink(h2, s1)
    s1s2 = {'bw':100} # Volitelné nastavení rychlosti spoje (100Mbps)
    net.addLink(s1, s2, cls=TCLink , **s1s2) # Spoj s upravenými parametry

    # Spuštění emulátoru
    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s1').start([])
    net.get('s2').start([])

    info( '*** Post configure switches and hosts\n')

    # Volitelná konfigurace koncových prvků příkazem cmdPrint
    # (ukázka nastavení IPv6 adres)
    h1.cmdPrint('ifconfig h1-eth0 inet6 add fc00::1/64')
    h2.cmdPrint('ifconfig h2-eth0 inet6 add fc00::2/64')

    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Příkaz *cmdPrint* umožňuje aplikovat libovolný terminálový příkaz na konkrétní koncové zařízení. Příkaz bude proveden při spuštění emulované topologie. Nejčastější použití je pro podrobnou konfiguraci síťových rozhraní, ale lze ho použít pro libovolnou funkcionalitu.

6.4.3 Nástroj MiniEdit

MiniEdit je grafický nástroj pro vytváření vlastních topologií v interaktivním režimu. Za vznikem nástroje MiniEdit stojí hlavně Bob Lantz a Gregory Gee. V současnosti je zdrojový kód dostupný na platformě GitHub[6] a MiniEdit je součástí emulátoru Mininet. Kompletní MiniEdit je obsažen v souboru: *mininet/examples/miniedit.py*

Nástroj umožňuje vytvoření vlastní síťové topologie v grafickém prostředí a její následný export ve formátu spustitelného skriptu (stejný formát jako při vytváření vlastní topologie popsané v kapitole

[6] `https://github.com/mininet/mininet/blob/master/examples/miniedit.py`

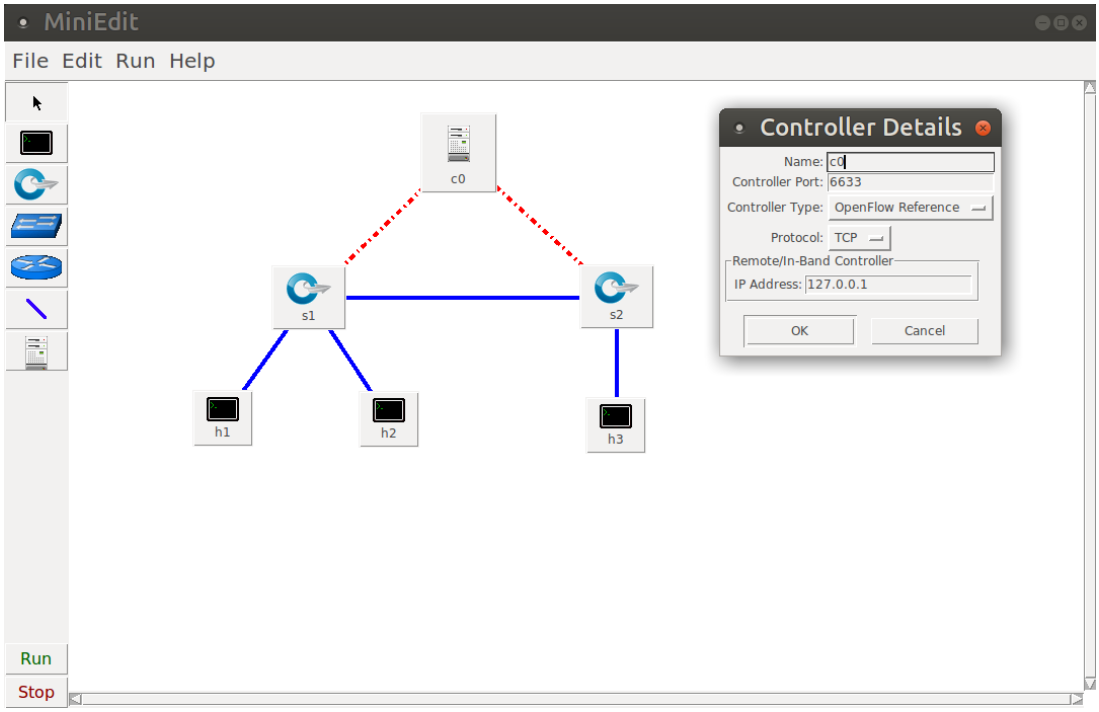
6.4.2). Nástroj volitelně umožňuje přímé spuštění vytvořené topologie. V takovém případě dojde k automatickému spuštění emulátoru Mininet.

Spuštění nástroje MiniEdit

Nástroj MiniEdit lze spustit pouhým provedením skriptu *miniedity.py*. Jediným rozdílem je, zda je skript spuštěn s klasickým oprávněním (neumožňuje přímé spuštění emulátoru Mininet), nebo s oprávněním *root* (umožňuje přímé spuštění emulátoru Mininet z prostředí nástroje MiniEdit).

```
# Základní spuštění pouze pro vytváření a exportování topologií
./mininet/examples/miniedit.py

# Spuštění umožňující přímý start emulátoru Mininet (oprávnění root)
sudo ./mininet/examples/miniedit.py
```

Prostředí nástroje MiniEdit



Obrázek 6.1: Prostředí nástroje MiniEdit

Nástroj poskytuje velice jednoduché rozhraní s hlavním ovládacím panelem umístěném v levém okraji okna. Ovládací panel obsahuje nástroj výběru a síťové prvky, které lze přidat do topologie. Základní okno programu je znázorněno na obrázku 6.1. Jednotlivé volby jsou následující (postupně seshora):

- **Nástroj výběru (select)** – volba umožňuje přesouvat jednotlivé prvky v topologii (držením levého tlačítka myši) a případně je mazat (kliknutím na prvek a následným stisknutím klávesy *DEL*).
- **Koncový prvek (host)** – vloží koncový prvek v podobě virtuálního počítače s operačním systémem Linux.

- **SDN přepínač (switch)** – vloží síťový prvek podporující protokol OpenFlow a tedy umožňující navázání spojení s SDN kontrolérem.
- **Tradiční přepínač (legacy switch)** – vloží prvek simulující klasický přepínač, který poskytuje pouze základní přepínací funkce.
- **Tradiční směrovač (legacy router)** – vloží prvek simulující klasický směrovač, který poskytuje pouze základní směrovací funkce.
- **Síťový spoj (net link)** – vytvoří síťový spoj mezi vybranými prvky v topologii. Vytvoření spojení probíhá dlouhým držením levého tlačítka myši za současného přesunu kurzoru mezi vybranými prvky. Při vytváření spojení mezi SDN kontrolérem a síťovým prvkem dojde k automatickému označení tohoto spoje červenou barvou a přerušovanou linkou.
- **SDN kontrolér (controller)** – vloží SDN kontrolér. Konkrétní typ lze vybrat po jeho přidání pomocí pravého tlačítky myši a volby *Properties*. Ta zobrazí dialogové okno *Controller Details* s možnostmi nastavení. Externí kontrolér může být nastaven volbou *Remote Controller*.
- **Ovládací prvky emulátoru Mininet (run / stop)** – v levém spodním rohu okna programu se nachází dvě tlačítka pro spuštění a zastavení emulátoru Mininet. Ke spuštění emulátoru musí být nástroj MiniEdit spuštěn s oprávněním *root*, jinak dojde k pádu programu!

Export topologie z nástroje MiniEdit

Spustitelný skript kompletní topologie lze vygenerovat kliknutím na volbu: *File/Export Level 2 Script*. Tím dojde k vygenerování souboru formátu *.py*. Před samotným procesem exportu je však vhodné aktuální projekt nejprve uložit pomocí volby: *File/Save*. Projekt samotný má formát souboru *.mn* a jedině načtením tohoto souboru, lze při pádu či ukončení programu, pokračovat v rozpracované topologii. Vyexportovaný skript topologie nelze použít pro import!

Topologii lze volitelně přímo z programu spustit (pokud je nástroj spuštěn s právy *root*) volbou *Run/Run* – případně tlačítkem *Run* v levém dolním rohu. Před touto akcí je opět vhodné projekt nejprve uložit.

Spuštění exportovaného skriptu

Exportovaný skript s topologií včetně kompletní konfigurace lze spustit stejně jako v případě spustitelného skriptu vlastní topologie popsaném v kapitole 6.4.2. Skript zajistí spuštění emulátoru Mininet, provede veškerou konfiguraci (včetně kontroléru, použitých přepínačů, nastavení IP adres, nastavení parametrů spojů, atd.). V tomto případě se proto opět nepoužívá příkaz *mn*.

Spuštění skriptu vyžaduje spustitelné oprávnění. To lze nastavit v závislosti na použitém OS přímo v grafickém rozhraní (obvykle volbou *Properties / Permissions / Alow this file to run as a program*), případně v terminálu příkazem *chmod*.

```
# Nastavení všech oprávnění všem uživatelům
sudo chmod 777 soubor.py

# Spuštění vyexportovaného skriptu z nástroje MiniEdit
sudo ./soubor.py
```

Podrobný návod pro nástroj MiniEdit je dostupný na webu: `http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/`

6.5 Ovládání topologie

Tato kapitola popisuje běžné úkony pro správu emulované síťové topologie v nástroji Mininet. Většina uvedených nástrojů a postupů je však platná i pro správu reálné SDN topologie.

6.5.1 Správa koncových prvků

Koncové prvky jsou v nástroji Mininet emulovány metodou sdílející jádro operačního systému. Z toho důvodu obsahuje každý emulovaný prvek stejné programy jako hostitelský operační systém. Systém souborů je také sdílený, což usnadňuje výměnu souborů a instalaci nových programů.

Jednotlivé koncové prvky jsou v terminologii emulátoru označeny jako *hosts* (hosté).

Terminál koncových prvků

Terminál jednotlivých prvků lze z prostředí emulátoru otevřít příkazem *xterm*.

```
# Spuštění terminálu vybraného hosta (příklad označení: h1)
mininet> xterm OZNACENI_HOSTA
```

Příkaz otevře nové okno s terminálem daného hosta. V tomto prostředí lze pak provádět veškeré unixové příkazy, včetně instalace dodatečného software, zachytávání paketů apod. Nelze však využívat aliasy a názvy jednotlivých zařízení používané v rámci příkazů emulátoru (*mininet*>). Nelze proto například spustit nástroj *ping* s parametrem *h1*, ale je nutné IP adresu specifikovat standardně (například *ping 10.0.0.1*).

Ve výchozím stavu je velikost terminálového okna a nastaveného fontu velice malá. Zvětšit je lze klávesovou zkratkou *shift* +.

Skripty emulátoru pro správu koncových prvků

Emulátor Mininet obsahuje několik integrovaných skriptů pro jednodušší práci s koncovými prvky. Ty lze spouštět přímo z příkazové řádky emulátoru. Výhodou těchto skriptů je mimo jiné fakt, že jako parametr pracují s aliasy jednotlivých zařízení (například *h1*). Není proto potřeba zadávat IP adresy, což usnadňuje orientaci v topologii. Mezi podporované skripty patří:

```
# Ping - otestuje konektivitu mezi definovanými zařízeními (h1 a h2)
mininet> h1 ping h2

# Iperf - otestuje propustnost mezi definovanými zařízeními (h1 a h2)
mininet> iperf h1 h2

# Příklad výsledku testování nástrojem Iperf
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['2.78 Gbits/sec', '2.78 Gbits/sec']

# Konfigurace síťových rozhraní nástrojem ifconfig
mininet> h1 ifconfig -h # Vypíše dostupné příkazy nástroje ifconfig
mininet> h1 ifconfig h1-eth0 up / down # Povolí / zakáže vybrané rozhraní
mininet> h1 ifconfig h1-eth0 IP/PREFIX # Nastaví definovanou adresu a prefix
```

6.5.2 Správa OpenFlow zařízení

Síťové prvky podporující protokol OpenFlow lze spravovat i manuálně bez SDN kontroléru. Takové použití je vhodné zejména pro edukační účely, při odstraňování chyb a při prosté kontrole stavu síťových prvků.

Nástroje pro manuální správu protokolem OpenFlow

Mininet obsahuje dva nástroje pro manuální správu OpenFlow prvků – *dpctl* a *ofctl*. Ty umožňují vytvoření OpenFlow zpráv přímo z prostředí terminálu. Mezi podporované funkce pak patří přidání, modifikace či odstranění flow pravidel, zobrazení flow tabulky, vypsání stavu zařízení a další.

Syntaxe uvedených nástrojů využívá stejné příkazy, jen s odlišným názvem nástroje. Nástroj *dpctl* tak lze používat prostým příkazem *dpctl*, zatímco nástroj *ofctl* vyžaduje příkaz *sh ovs-ofctl*. Hlavní rozdíl mezi nástroji je způsob specifikace konkrétního zařízení. Nástroj *dpctl* vyžaduje kromě názvu zařízení i specifikaci IP adresy a portu, což komplikuje jeho použití. Z toho důvodu je vhodnější použití tohoto nástroje pro dotazy na všechny prvky – v tom případě tyto údaje není potřeba vyplňovat. Příklad použití tohoto nástroje:

```
# Výpis flow tabulky zařízení s1, kde typicky IP = 127.0.01 a PORT = 6634
mininet> s1 dpctl dump-flows tcp:IP:PORT

# Výpis flow tabulek všech zařízení
mininet> dpctl dump-flows
```

Naopak nástroj *ofctl* vyžaduje pouze specifikaci aliasu zařízení. Není tedy nutné definovat port ani IP adresu a použití pro konkrétní zařízení je podstatně jednodušší. Stejný příkaz pak vypadá následovně:

```
# Výpis flow tabulky vybraného zařízení
mininet> sh ovs-ofctl dump-flows ZARIZENI

# Ukázkový výpis flow pravidla z flow tabulky zařízení s1
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=37.721s, table=0, n_packets=1, n_bytes=98,
 idle_timeout=60, idle_age=37, priority=65535,icmp,in_port=2,vlan_tci=0x0000,
 dl_src=92:74:5b:c4:be:da,dl_dst=1a:49:15:01:5e:18,nw_src=10.0.0.6,
 nw_dst=10.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:1
```

Mezi nejdůležitější příkazy obou nástrojů patří:

```
# Výpis všech dostupných příkazů obou nástrojů:
mininet> dpctl -h
mininet> sh ovs-ofctl -h

# Výpis stavu zařízení (ID, DPID, OpenFlow vlastnosti, síťová rozhraní)
mininet> dpctl show # Pro všechna zařízení
mininet> sh ovs-ofctl show ZARIZENI # Konkrétní zařízení - například s1

# Výpis flow tabulek
mininet> dpctl dump-flows # Pro všechna zařízení
mininet> sh ovs-ofctl dump-flows ZARIZENI # Konkrétní zařízení - například s1
```

```
# Výpis statistik portů zařízení (odeslané / přijaté pakety, data, zahozené
# zprávy a chyby)
mininet> dpctl dump-ports # Pro všechna zařízení
mininet> sh ovs-ofctl dump-ports ZARIZENI # Konkrétní zařízení - například s1
```

Tvorba OpenFlow pravidel

Mezi nejdůležitější OpenFlow funkcionalitu patří možnost vytvářet flow pravidla. Pro tyto účely je vhodnější nástroj *ofctl* z důvodu jeho zaměření na konkrétní zařízení. Příkaz pro přidání flow pravidla je následující:

```
# Zařízení například s1
# MATCH = pole shod
# AKCE = definované akce (například odchozí port)
# Parametry označující číslo flow tabulky a prioritu jsou volitelné
mininet> sh ovs-ofctl add-flow ZARIZENI MATCH,action=AKCE,
 [table=ID],[priority=PRIORITA]

# Příklad vložení flow pravidla na s1
# Pole shod = příchozí port číslo 1
# Akce = odeslání na port číslo 2
mininet> sh ovs-ofctl add-flow s1 in_port=1,action=output:2

# Příklad vložení pravidla zachytávajícího IPv6 pakety typu "NA"
mininet> sh ovs-ofctl add-flow s2 priority=40000,ipv6,nw_proto=58,
 icmp_type=136,action=output:3
```

Syntaxe příkazu neumožňuje vložení mezer mezi jednotlivé pole shod a akce. V případě vložení mezery dojde k chybové hlášce *command takes at most 2 arguments*.

Mazání OpenFlow pravidel

Uvedené nástroje podporují modifikaci i mazání vytvořených flow pravidel. Vhodnější nástroj je opět *ofctl*, z důvodu zaměření na konkrétní zařízení. Mazání pravidel probíhá příkazem *del-flows*, kde jako parametr je nutné specifikovat odpovídající pole shod. Volitelně lze použít parametr *–strict* pro přesnější porovnávání včetně masky (funkcionalita je vysvětlena v kapitole 3.6.1).

```
# Příklad vymazání předchozího pravidla
# (příkaz nepodporuje definici priority ani akcí)
mininet> sh ovs-ofctl del-flows s2 ipv6,nw_proto=58,icmp_type=136
```

Prerekvizity polí shod

Porovnávání některých polí hlaviček zprávy vyžaduje definování závislých polí – tzv. prerekvizity. Ty zaručují, že porovnávané pole se ve zprávě opravdu nachází. Příkladem je IPv4 adresa, která se ve zprávě může nacházet pouze pokud nižší protokol (Ethernet) definuje typ zprávy jako IPv4 paket – pomocí nastavení Ethernet typu na hodnotu *0x0800*.

Následující porovnávané pole vyžadují nastavení uvedených prerekvizit:

- **VLAN VID** – vlan_tci=0x1000.
- **ARP** – eth_type=0x0806 (pro ARP), nebo eth_type=0x8035 (pro RARP).
- **IPv4** – eth_type=0x0800.

- **IPv6** – eth_type=0x86dd.
- **MPLS** – eth_type=0x8847, nebo eth_type=0x8848.
- **TCP** – jako IPv4/IPv6 + ip_proto=6.
- **UDP** – jako IPv4/IPv6 + ip_proto=17.
- **SCTP** – jako IPv4/IPv6 + ip_proto=132.
- **ICMPv4** – jako IPv4 + ip_proto=1.
- **ICMPv6** – jako IPv6 + ip_proto=58.
- **ND, Neighbor Solicitation** – jako ICMPv6 a icmp_type=135 a icmp_code=0.
- **ND, Neighbor Advertisement** – jako ICMPv6 a icmp_type=136 a icmp_code=0.

Příklad vložení nového flow pravidla pro zachycení ICMP zprávy typu 0 (*echo reply*):

```
mininet> sh ovs-ofctl add-flow s1 eth_type=0x0800,ip_proto=1,icmp_type=0,action=
```

Porovnávání typu ICMP zprávy vyžaduje definici prerekvizit typu Ethernetu (*0x0800*) a IP protokolu (*1*). Uvedený příkaz definuje prázdnou akci, která odpovídá logice zahození zpráv zachycených pravidlem.

Aliasy polí shod

Uvedené nástroje pak podporují aliasy na různé typy shod. Místo definování čísla protokolu (například *0x0800*) je tak možné použít zkratku (například *ip*). Jedná se o:

- **ip** – eth_type=0x0800.
- **ipv6** – eth_type=0x86dd.
- **icmp** – eth_type=0x0800,ip_proto=1.
- **icmp6** – eth_type=0x86dd,ip_proto=58.
- **tcp** – eth_type=0x0800,ip_proto=6.
- **tcp6** – eth_type=0x86dd,ip_proto=6.
- **udp** – eth_type=0x0800,ip_proto=17.
- **udp6** – eth_type=0x86dd,ip_proto=17.
- **sctp** – eth_type=0x0800,ip_proto=132.
- **sctp6** – eth_type=0x86dd,ip_proto=132.
- **arp** – eth_type=0x0806.
- **rarp** – eth_type=0x8035.
- **mpls** – eth_type=0x8847.
- **mplsm** – eth_type=0x8848.

Způsob zápisu pomocí aliasů podstatně zjednodušuje proces definování prerekvizit, a je proto vhodným způsobem zápisu. V závislosti na použitém softwarovém přepínači může při klasickém zápisu dojít ke konverzi na zkrácený způsob a naopak. Ten se pak může projevit odlišným zápisem ve flow tabulce. Aby tyto aliasy fungovaly, musí být cílovým síťovým prvkem podporovány!

Uvedený seznam je platný pro softwarový přepínač Open vSwitch verze 2.8 a novější: `http://www.openvswitch.org/support/dist-docs/ovs-fields.7.pdf`

Akce

Akce určují, co se má provést se zprávou, jejíž pole odpovídají definovanému pravidlu (*match*). Pokud není definována žádná akce, dojde k zahození zprávy. Pokud není uvedeno jinak, v rámci

jednoho flow pravidla může být definováno více akcí. Mezi nejdůležitější podporované akce patří:

- **Output** – odešle zprávu na definovaný port.
- **Normal** – odešle zprávu ke zpracování tradiční síťovou vrstvou. Jedná se o koncept hybridního SDN, který nemusí být síťovým prvkem podporován.
- **All** – odešle zprávu na všechny porty, kromě portu, na kterém byla přijata.
- **Flood** – odešle zprávu na všechny porty, kromě příchozího portu a všech portů, které mají zakázán *flooding* (porty zablokované STP).
- **Controller** – odešle zprávu na SDN kontrolér jako *packet-in* událost.
- **Local** – odešle zprávu na port, který má stejný název jako název zařízení.
- **In_port** – odešle zprávu zpět na port, na kterém byla přijata.
- **Enqueue** – zařadí zprávu do fronty specifikovaného portu (tato funkcionalita nemusí být síťovým prvkem podporována).
- **Drop** – zahodí zprávu. V tomto případě nelze definovat žádnou dodatečnou akci.
- **Goto_table** – odešle zprávu k dalšímu zpracování definovanou tabulkou.
- **Akce typu "mod"(mod_ ...)** – umožňují definovat modifikaci polí hlaviček zprávy. Mezi podporovaná pole patří: VLAN (ID a priorita), zdrojová a cílová adresa (MAC a IP), zdrojový a cílový port (TCP), hodnota ToS (Type Of Service) a další.
- **Akce typu "set"(set_ ...)** – umožňují definování nových polí hlavičky zprávy. Mezi podporované patří: tunnel, queue a MPLS (label, traffic-class, TTL).
- **Další akce** obsahují například: strip_ ... (odstranění hodnoty, pokud je definována), push (push_vlan, push_mpls), resubmit, dec_ttl (snížení TTL o hodnotu jedna), note, meter_id, exit (ukončí zpracování) a další.

Podporované akce softwarového přepínače Open vSwitch jsou uvedeny ve specifikaci: `http://www.openvswitch.org/support/dist-docs/ovs-ofctl.8.txt`

6.5.3 Monitorování provozu

Pro efektivní správu síťové topologie je někdy vhodné mít možnost kompletního monitorování provozu. Jako v klasické síťové topologii, i v emulované síti lze nástrojem Mininet zachytávat veškerý datový provoz. Jako nejvhodnější nástroje pro zachytávání a následnou analýzu slouží *tcpdump* a Wireshark.

tcpdump

Tcpdump[7] je nástroj využívající CLI a umožňující online monitoring provozu, ale i ukládání zachycených dat ve formátu *.pcap*. V předpřipravených obrazech virtuálních počítačů pro SDN už tcpdump bývá nainstalován, případně ho lze snadno doinstalovat z oficiálních repozitářů následujícími příkazy:

```
# Instalace v Linuxových distribucích založených na Debian
apt-get install tcpdump

# Instalace v Linuxových distribucích založených na Red Hat
yum install tcpdump
```

[7] `https://www.tcpdump.org/`

Mezi nejpoužívanější příkazy nástroje patří:

```
# Vypsání dostupných rozhraní na zařízení
tcpdump -D

# Spuštění zachytávání provozu na rozhraní h1-eth0
tcpdump -i h1-eth0

# Zachycení pouze prvních 10 paketů
tcpdump -c 10 -i h1-eth0

# Vypsání obsahu provozu v ASCII formátu
tcpdump -A -i h1-eth0

# Vypsání obsahu provozu v HEX / ASCII formátu
tcpdump -XX -i h1-eth0

# Zachycení a uložení provozu do souboru provoz.pcap
tcpdump -w provoz.pcap -i h1-eth0

# Zobrazení zachyceného provozu ze souboru provoz.pcap
tcpdump -r provoz.pcap

# Parametry pro filtrování zachycení provozu:
tcpdump -i h1-eth0 tcp # TCP provoz
tcpdump -i h1-eth0 port 21 # Pouze specifikovaný port 21
tcpdump -i h1-eth0 src 10.0.0.1 # Pouze zdrojová IP adresa 10.0.0.1
tcpdump -i h1-eth0 dst 10.0.0.2 # Pouze cílová IP adresa 10.0.0.2
```

Tcpdump lze v emulátoru Mininet spustit přímo na koncových zařízeních příkazem *tcpdump*.

Kompletní nápovědu k nástroji tcpdump lze najít na webu:
`http://www.tcpdump.org/manpages/tcpdump.1.html`

Wireshark

Wireshark je pokročilý nástroj pro grafickou analýzu síťového provozu. Nástroj je multiplatformní, podporuje online i offline monitoring, velké množství protokolů včetně OpenFlow, a umožňuje export zachycených dat v různých formátech včetně *.pcap(ng)*, *.cap* a *.snoop*. Tyto soubory včetně mnoha dalších mohou být zpětně analyzovány.

Wireshark je open source program, a je proto zdarma dostupný ke stažení na webu:
`https://www.wireshark.org/`

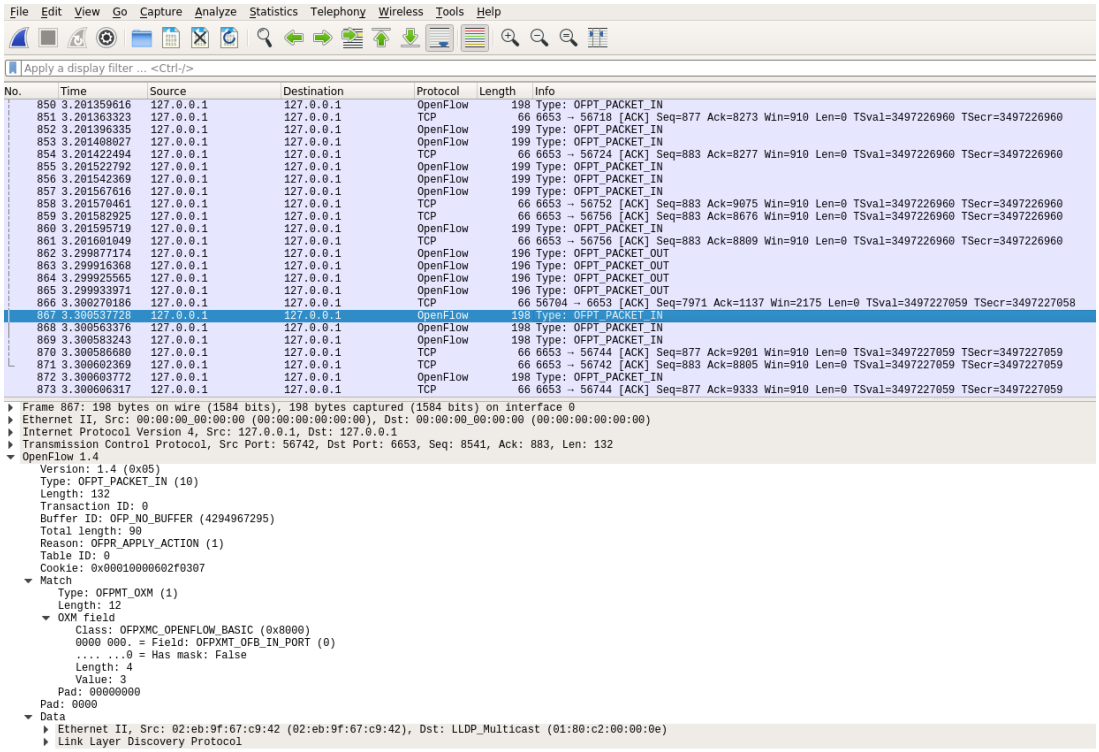
Ovládání programu odpovídá běžným konvencím. Spuštění zachytávání probíhá z menu: *Capture / Options* (výběr rozhraní pro zachytávání provozu) / *Start*. Zachytávaný provoz se poté zobrazuje v reálném čase ve hlavním okně programu. Zachytávání lze ukončit kliknutím na červené tlačítko *Stop*.

Wireshark umožňuje podrobné filtrování zachyceného provozu. K tomuto účelu je integrován nástroj *Filter Expression* pro vytváření komplexních pravidel a filtrů. Stejná pravidla lze vytvořit i přímo ručně přes pole *Filter string*. Příklad jednoduchého pravidla pro výpis pouze provozu se specifikovanou zdrojovou / cílovou IP adresou může vypadat například takto:

```
ip.addr==10.0.0.1
```

Nástroj Wireshark je v předpřipravených obrazech virtuálních počítačů pro SDN opět běžně nainstalován. Spustit ho lze přímo z grafického prostředí, či z terminálového prostředí příkazem *sudo wireshark*. Pro zachycení veškerého provozu emulované sítě Mininet je pak nutné zvolit rozhraní *any*. Pro výpis pouze z vybraného rozhraní konkrétního přepínače lze toto rozhraní přímo vybrat (například *s1-eth1*). Rozhraní koncových prvků nejsou z příkazové řádky emulátoru dostupné a v takovém případě je nutné vybrat rozhraní na přepínači, které vede k tomuto prvku. Alternativou je spustit nástroj Wireshark přímo na vybraném koncovém prvku (příkazem *wireshark* spuštěným z terminálu prvku). Takový postup je následující:

```
# Otevření terminálu hosta 1
mininet> xterm h1

# Spuštění nástroje Wireshark (v terminálu hosta 1)
wireshark
```



Obrázek 6.2: Prostředí programu Wireshark

6.5.4 RYU kontrolér: praktické použití

RYU kontrolér představuje ideální kompromis mezi podporovanými funkcemi a relativní jednoduchostí použití. Na rozdíl od integrovaných kontrolérů musí být spuštěn jako vzdálený kontrolér parametrem *–controller=remote* zadaným při spouštění emulátoru Mininet. Tento parametr instruuje Mininet k naslouchání pro připojení externího kontroléru.

```
# Parametr může obsahovat přepínače ip=[IP_ADRESA],port=[PORT]
sudo mn --controller=remote
```

Samotné spuštění kontroléru RYU probíhá příkazem *ryu-manager*. Tento příkaz je nutné zadat v novém terminálovém okně / záložce, případně na jiném zařízení, než na kterém je spuštěn emulátor Mininet.

```
# Spuštění nainstalovaného kontroléru z defaultního umístění
ryu-manager

# Spuštění kontroléru z adresáře "ryu" (bez nutnosti instalace kontroléru)
./ryu/bin/ryu-manager

# Výpis nápovědy kontroléru
ryu-manager -h

# Spuštění kontroléru s informačním výpisem (debug)
ryu-manager -verbose
```

RYU kontrolér je v současné verzi nastavený na naslouchání spojení na TCP (SSL) portu 6653. Z důvodu kompatibility je však jako záložní port definován TCP (SSL) port 6633. Z toho důvodu by mělo dojít k navázání spojení i s defaultně nastaveným emulátorem Mininet.

Defaultní nastavení portů pro navázání komunikace mezi RYU kontrolérem a síťovými prvky je možné změnit v souboru: `ryu/ryu/ofproto/ofproto_common.py`

Samotný příkaz *ryu-controller* spustí kontrolér ve výchozím stavu, tedy bez jakýkoliv modulů či aplikací. V tomto stavu kontrolér neprovádí žádnou síťovou funkcionalitu. Konektivita v dané síti tím pádem nebude fungovat. Funkcionality jako je L2 přeposílání, L3 směrování a další, jsou umístěny ve specifických modulech, které je potřeba spustit. Alternativně lze spustit vlastní modul s vyžadovanou funkcionalitou.

```
# Spuštění kontroléru s definovaným modulem
ryu-manager JMENO_MODULU

# Příklad spuštění typického modulu implementujícího logiku L2 přepínače
ryu-manager simple_switch_13.py
```

Postup instalace, spuštění a popis jednotlivých dodávaných modulů kontroléru je popsán v kapitole 5.1.

Oficiální dokumentace RYU kontroléru je dostupná na webu:
`https://osrg.github.io/ryu/resources.html#documentation`

Na oficiálním webu kontroléru lze také stáhnout elektronickou publikaci *EBOOK: RYU SDN FRAMEWORK*[8], ve které je popsaná instalace kontroléru a jednotlivé moduly a způsoby jejich použití.

Samotná online dokumentace pak poskytuje skvělý zdroj informací o veškerých funkcích OpenFlow protokolu a knihovnách kontroléru.

API OpenFlow protokolu v kontroléru RYU je detailně popsané na webu:
`https://ryu.readthedocs.io/en/latest/ofproto_ref.html`

[8] `https://osrg.github.io/ryu/resources.html`

7. Závěr

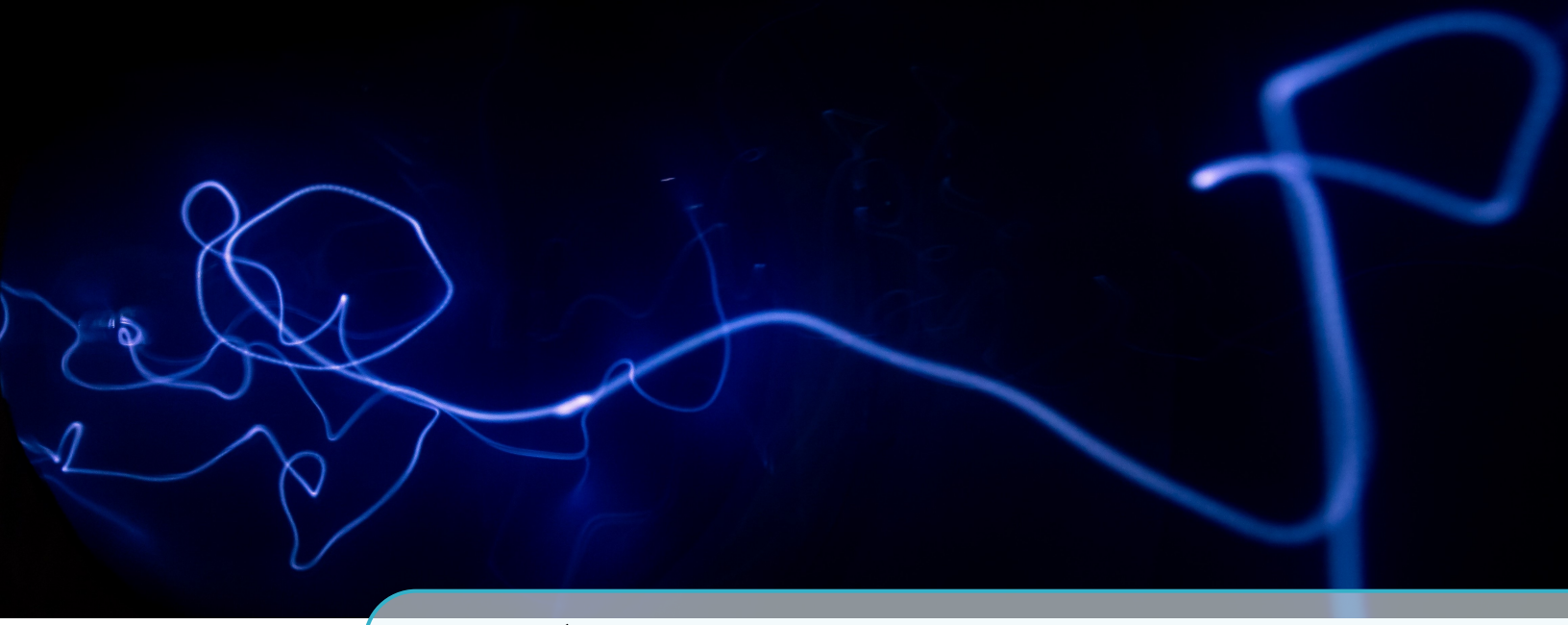
SDN mají za sebou dynamický vývoj. Od pouhého akademického a experimentálního testování se postupně rozšířily do datových center a začínají pronikat i do rozsáhlých sítí poskytovatelů služeb. SDN se staly obecně známým pojmem, rozšířeným podobně jako třeba cloud nebo IoT. To ale sebou přináší i jistá negativa. Pod pojem SDN je v současnosti snaha zařadit prakticky cokoliv, co se týká automatizace síťových procesů. Je proto důležité, zejména pokud se pohybujeme v oblasti počítačových sítí, znát koncept SDN. Tato publikace si kladla za cíl tento koncept objasnit a zpřístupnit čtenářům i praktickou realizací pomocí emulátoru.

Princip tohoto konceptu lze uplatnit v mnoha různorodých oblastech:

- Studentům počítačových sítí dovoluje modelovat činnosti síťových prvků, u kterých ještě nedávno bylo možné dosáhnout určitých síťových vlastností pouze použitím omezené sady příkazů. Studenti si díky SDN mohou vyzkoušet sami naprogramovat funkcionalitu přepínačů, směrovačů nebo vyvinout svůj vlastní protokol.
- Vědcům poskytuje platformu pro téměř neomezený výzkum. Umožňuje otestování a nasazení experimentálních funkcionalit, protokolů a síťových algoritmů.
- Průmyslu nabízí inovativní způsob správy sítě, který dokáže snížit celkové náklady, zvýšit efektivitu sítě a v určitých případech přinést vyšší výkon nebo lepší zabezpečení.

Aktuální trend vývoje SDN je v integraci několika technologií (NFV, OpenStack, SDN) pro dosažení kompletní správy infrastruktury a služeb. Tento princip reaguje na chování uživatelů a na obrovský nárůst přenášených dat.

Nicméně bez porozumění mechanizmu síťových protokolů a jejich vzájemné provázanosti v takové míře, v jaké se s nimi čtenáři seznámili při studiu tradičních sítí, nelze úspěšně vytvářet moderní programovatelné sítě. Doufáme, že tato publikace zprostředkuje snadnější vhled do oblasti SDN a zodpoví otázky na směr vývoje počítačových sítí.



Použitá literatura

[1] Hewlett-Packard. *HP Switch Software OpenFlow v1.3 Administrator Guide K/KA/KB/WB 15.18*. 5. vydání. [online], [cit. 2019-05-30]. Dub. 2016. URL: https://support.hpe.com/hpsc/doc/public/display?docId=c04777809.

[2] Filip Holík. „Using SDN to Enhance IoT Security". Disertační práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky, Pardubice, 2018 [online], [cit. 2019-05-30]. URL: https://theses.cz/id/p5l213/.

[3] Jan Mokráček. „Analýza kontrolérů softwarově definovaných sítí". Bakalářská práce. Univerzita Pardubice, Fakulta elektrotechniky a informatiky, Pardubice, 2017 [online], [cit. 2019-05-30]. URL: https://dk.upce.cz/handle/10195/68099.

[4] *OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06)*. [online], [cit. 2019-05-30]. ONF. Břez. 2015. URL: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

Název	Softwarově definované sítě
Autoři	Ing. Filip Holík, Ph.D.
	Ing. Soňa Neradová, Ph.D.
Vydavatel	Univerzita Pardubice
Odpovědný redaktor	Lenka Tobišková
Zveřejnění	červenec 2019
Stran	95
Vydání	první
Náklad	elektronická verze

ISBN 978-80-7560-235-0 (pdf)