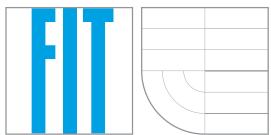
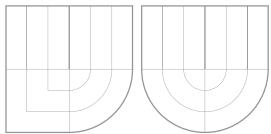




VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DEMONSTRACE MOŽNOSTÍ TECHNOLOGIE CISCO ONEPK

CISCO ONEPK TECHNOLOGY DEMONSTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RICHARD CHOMJAK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ONDŘEJ RYŠAVÝ, Ph.D.

BRNO 2014

Abstrakt

Cílem této bakalářské práce je demonstrace možností kvality služeb (QoS) v technologii Cisco onePK. Vytvořený prototyp (Gosia) umožňuje na základě jednoduchého konfiguračního jazyka dynamicky směrovat předem definovaná data. Jednotlivé změny chování směrování jsou odvozovány ze stavu sítové infrastruktury.

Abstract

This bachelors thesis aims to demonstrate the uses of Quality of Services (QoS) within Cisco onePK technology. The created prototype (Gosia) enables the dynamic direction of previously defined data on the basis of a simple configuration language. Individual changes in the direction of the data flow are derived from the state of the network infrastructure.

Klíčová slova

Cisco onePK, Cisco ONE, OpenFlow, SDN, QoS, RSVP, Gosia

Keywords

Cisco onePK, Cisco ONE, OpenFlow, SDN, QoS, RSVP, Gosia

Citace

Richard Chomjak: Demonstrační možnosti technologie Cisco OnePK, bakalářská práce, Brno, FIT VUT v Brně, 2014

Demonstrace možností technologie Cisco OnePK

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ondřeja Ryšavého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Richard Chomjak
8. května 2014

Poděkování

Primárne ďakujem Ing. Ondřejovi Ryšavému, Ph.D. za rady a nápady pri tvorbe práce. Za cenné informácie ohľadom technológie onePK patrí moja vrelá vďaka B.S. Josephovi Clarkovi.

© Richard Chomjak, 2014.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	3
2	Softvérom definované siete, SDN	4
2.1	Architektúra sieťových zariadení	5
2.2	Princípy SDN	5
2.2.1	Elementy v SDN	7
2.2.2	Programové rozhrania SDN	8
2.3	OpenFlow	10
2.3.1	Sieťový protokol OpenFlow	10
2.4	onePK	15
2.4.1	Architektúra onePK	16
2.4.2	Princíp použitia	19
2.5	Zhodnotenie	20
3	Kvalita služieb, QoS	21
3.1	Integrované služby, InterServ	21
3.1.1	RSVP	23
3.2	Diferencované služby, DiffServ	24
3.3	Performance Routing, PfR	25
3.4	Zaistenie kvality služieb na zariadeniach	25
3.4.1	Rozlíšenie a označenie dát	26
3.4.2	Správa dátového toku	26
3.4.3	Správa zahltenia	27
3.4.4	Zabránenie zahltenia	29
3.4.5	Signalizácia	29
3.4.6	Správa efektívneho využitia linky	29
3.5	Zhodnotenie	30
4	Gosia	31
4.1	Požiadavky na prototyp	31
4.2	Architektúra prototypu	32
4.3	Návrh	33
4.3.1	Aplikácia	33
4.3.2	Konfiguračný jazyk	35
4.4	Implementácia	37

5	Experimenty	39
5.1	Porovnanie EIGRP, PBR a Gosia	40
5.1.1	Výkonné porovnanie EIGRP s PBR	41
5.1.2	Gosia poll interval #1	42
5.1.3	Gosia poll interval #2	43
5.2	Emulácia reálnej komunikácie	44
5.2.1	Gosia #1	45
5.2.2	Gosia #2	47
5.2.3	Gosia #3	49
5.2.4	Gosia #4	51
5.3	Zhodnotenie	54
6	Záver	55
A	Konfigurácie	58
A.1	Konfigurácie zariadenia	58
B	Príklad použitia	59
B.1	Out of Band topológia	59
B.2	In band topológia	60
B.3	Príklad konfiguračného jazyka	61
C	Obsah CD	65
D	Zoznam skratiek	66

Kapitola 1

Úvod

Súčasná moderná spoločnosť je čoraz viac závislá na komunikačných technológiách, čo sa možrejme zvyšuje aj požiadavky na počítačové sieťové infraštruktúry. Podľa prognóz [8], zvýšenie dátovej prevádzky na sieti Internet sa očakáva v období medzi rokmi 2012 až 2017, a to približne 2,75krát.

Zvyšovanie dátovej prevádzky je spôsobené novodobými trendmi ako: nárast používateľov siete Internet, prenos veľkého množstva multimediálnych dát a pod. Ďalší nárast sieťovej prevádzky v infraštruktúrach je spôsobený zámerným preťažovaním, ktoré sa rozmoholo vo forme DoS (Denial of Service) útokov a jeho modifikáciách.

Nie len novodobé, ale aj staršie trendy sa stali príčinou vzniku nových pohľadov na návrh a využívanie počítačovej sieťovej infraštruktúry. Siete novej generácie NGN (Next Generation Networks) sa snažia zameráť na efektívne využívanie sieťových zariadení, kapacít sieťových spojení, automatizáciou sieťových zariadení, inteligentným smerovaním dát, virtualizáciou sieťových infraštruktúr atď.

Podmnožinou siete novej generácie sú takzvané softvérom definované siete SDN (Software Defined Networking). Softvérom definované siete umožňujú väčšiu kontrolu nad sieťovými infraštruktúrami a dovoľujú vytvárať zložitejšie sieťové infraštruktúry.

Cieľom tejto bakalárskej práce je využitie kvality služieb QoS (Quality of Services) so zameraním na rezerváciu zdrojov aplikácií v sieťových zariadeniach, v SDN sietiach z pohľadu systému Cisco ONE (Open Network Environment) za použitia Cisco onePK (One Platform Kit) SDK (Software Development Kit).

Práca je štruktúrovaná nasledovne. Kapitola 2 približuje koncepty softvérom definovaných sieti. Tieto koncepty sú doplnené o základné poznatky SDN architektúr OpenFlow a onePK. V kapitole 3 sú zhrnuté základné informácie o kvalite služieb, rozdiely medzi Diferencovanými službami a Integrovanými službami a nasledovne sú prezentované informácie o prostriedkoch k dosiahnutiu chovania kvality služieb. Nasledujúca kapitola 4 čitateľa oboznámi o návrhu a implementácii prototypu Gosia. Prototyp Gosia kombinuje poznatky z kvality služieb a umožňuje prioritizovať dátový tok na základe predom definovaného konfiguračného súboru. Kapitola 5 je venovaná experimentom na prototype Gosia. Prvá časť experimentov je venovaná meraniu stratovosti dát, druhá časť sa zaoberá meraniu na emulovanej reálnej dátovej prevádzke.

Kapitola 2

Softvérom definované siete, SDN

Uplynulé roky sú charakteristické tým, že návrh počítačových infraštruktúr rozdeľoval sieťové prostriedky, výpočtové prostriedky a dátové úložiska, ako fyzicky, tak aj logicky. V priebehu technologického vývoja v informačných technológiách, predošlé pohľady a charakteristiky počítačových infraštruktúr prestávajú plniť potrebné kritériá daného pokroku [11].

K najväčším technologickým posunom, ktoré bezprostredne ovplyvňujú dnešné informačné odvetvie môžeme zaradiť virtualizáciu. Jednou z vlastností tejto technológie je možnosť migrovania virtuálnych obrazov systémov do akejkoľvek fyzickej lokality. Táto technológia stala pri zrode ďalším miľníkom v informačných technológiách, ako napríklad efektívnejšie využívanie a zdieľanie počítačových zdrojov (elastic computing) a pod.

Vyššie spomenuté technológie umožňujú prevádzkovateľom zdieľať svoje výpočtové kapacity používateľom. Táto a iné vlastnosti vytvárajú niekoľko problémov z pohľadu počítačovej sieťovej infraštruktúry. Medzi jeden z problémov môžeme zaradiť separovanie jednotlivých používateľov pri zdieľaní výpočtových a sieťových prostriedkov.

Ďalším z nedostatkov predchádzajúcich pohľadov na sieťovú počítačovú infraštruktúru je proces inovácie v sieťovom odvetvi informatiky. Výrobcovia sieťových zariadení často nedokážu rýchlo reagovať na technologické požiadavky svojich zákazníkov. Zákazníci zvyčajne nemajú veľa možnosti pridania, respektíve modifikovania nových technológií do sieťových zariadení. Aj keď existujú možnosti manipulácie so sieťovými zariadeniami, ktoré umožňujú technologický vývoj, zväčša ho neumožňujú v potrebnom rozsahu [10, 11].

Technologický vývoj a následne jeho požiadavky sú hnacou silou, ktorá núti prehodnocovať a vyvíjať pohľady architektúr a návrhov počítačových systémov.

Softvérom definované siete definujú iné pohľady na počítačové siete, ktoré umožňujú väčšiu kontrolu nad sieťovými zariadeniami, vytvárať zložitejšie architektúry a vyvíjať nové riešenia¹. V týchto konceptoch sa načrtávajú prístupy k sieťovým zariadeniam typicky ako k elementárnym časťam systému. Hlavnou vlastnosťou týchto konceptov je separácia dátovej časti (data plane) sieťového zariadenia s riadiacou časťou (control plane) [11].

V nasledujúcich častiach kapitoly sú priblížené základné princípy SDN 2.2, koncepty OpenFlow 2.3 a onePK 2.4.

¹Bruce Davie, Network Virtualization: Delivering on the Promises of SDN <http://www.opennetsummit.org/archives-april2013.html>

2.1 Architektúra sietových zariadení

Pre hlbšie pochopenie princípov softvérom definovaných sieti je nutné oboznámiť čitateľa s riadiacou časťou sietového zariadenia (control plane) a dátovou časťou zariadenia (data plane).

Riadiaca časť (control plane)

Riadiaca časť zariadenia obsahuje funkcie siete, ktoré riadia správanie siete (napr. sietové cesty, chovanie preposielania dát)².

V smerovačoch riadiaca časť (control plane) je komponent, ktorý slúži k manipulácií smerovacej tabuľky RIB (Routing Information Base) a na základe informácií v smerovacej tabuľke sú modifikované informácie vo FIB (Forwarding Information Base). Zmeny v smerovacej tabuľke sú zvyčajne inicializované smerovacími protokolmi, napríklad RIP (Routing Information Protocol), OSPF (Open Shortest Path First), BGP (Border Gateway Protocol) [7].

Dátová časť (data plane)

Dátová časť zariadenia (data plane) je komponent, ktorý modifikuje, kontroluje vstupné a výstupné dátá. Niektoré úkony, ktoré vykonáva dátová časť zariadenia:

- Modifikácia TTL (Time To Live)/Hop Count.
- Kontrolný súčet dát (checksum).
- Hĺbková analýza paketov DPI (Deep Packet Inspection).
- Kvalita služieb QoS (Quality of Services).

2.2 Princípy SDN

Väčšina terajších zariadení neseparuje riadiacu a dátovú časť. Spravovať zariadenie je možné doceliť viacerými spôsobmi. K vlastnostiam softvérom definovaných sieti sa najviac približuje správa protokolmi SNMP (Simple Network Management Protocol)³ a NETCONF (Network Configuration Protocol)⁴. Tieto protokoly sa používajú na hromadnú správu, konfiguráciu a monitorovanie. Z pohľadu SDN hlavnými nedostatkami týchto protokolov sú priamy prístup k určitým časťiam zariadenia, modifikácie zariadení v reálnom čase.

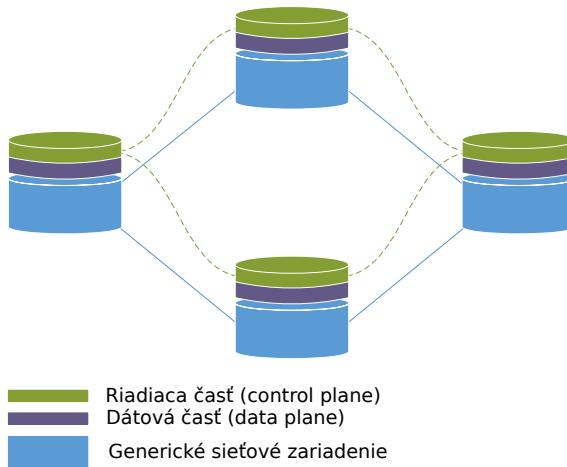
V tradičných architektúrách 2.1 je riadiaca a dátová časť zariadenia vyvýjaná ako celok. Tento návrh je často využívaný iba na jednu platformu zariadení.

²Software Defined Networking, Module 2.1: Control/Data Separation

<https://class.coursera.org/sdn-001/lecture/25>

³SNMPv3 RFC 3411 - 3418

⁴NETCONF RFC 6241 <http://tools.ietf.org/html/rfc6241>



Obr. 2.1: Tradičná riadiaca (control plane) architektúra.

Základným princípom softvérom definovných sietí je oddelenie riadiacej časti zariadenia od dátovej časti zariadenia. Táto myšlienka nepatrí medzi najnovšie, možno ju nájsť v modulárnych sietových zariadeniach a telekomunikáciach⁵. Na rozdiel od modulárnych zariadení v softvérom definovaných sieťach je riadiaca časť oddelená fyzicky na väčšiu vzdialenosť ako na pár centimetrov [11].

Hlavné dôvody separácie riadiacej a dátovej časti sú nasledujúce [11]⁶:

- Softvér riadenia sa vyvíja nezávisle od hardvéru.
- Vyššia abstrakcia riadiacého softvéru.
- Jednoduchšie ladenie riadiaceho programu.
- Programovateľnosť zariadení.
- Škálovateľnosť zariadení.

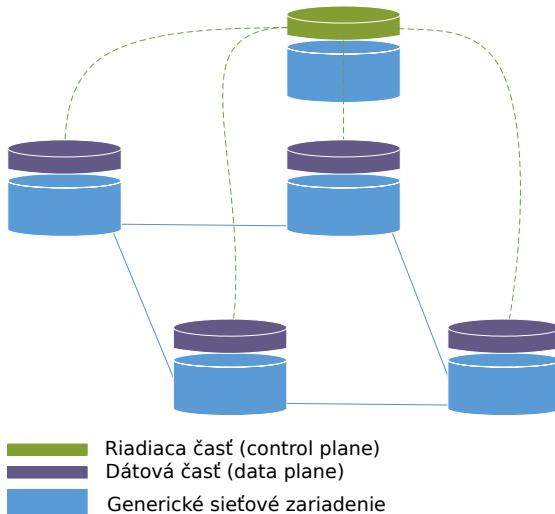
Najznámejšie architektúry v softvérom definovaných sieťach sú centralizované a distribuované. V závislosti od topológie siete je nutné vytvárať nezávislú paralelnú ovládaciu siet (out of band). Táto sieť slúži k dosiahnutiu vyšej dostupnosti hlavnej (inband) siete pri možnom výpadku komunikácie sietového zariadenia s kontrolérom.

Na obrázku 2.2 je zobrazená centralizovaná architektúra. V tejto architektúre sa pre riadenie zariadení využíva centrálny kontrolér. Tento kontrolér so všetkými zariadeniami navázuje a riadi komunikáciu. Keďže tento kontrolér riadi všetky zariadenia z pohľadu zaťaženia siete je najzraniteľnejšou časťou (single point of failure). Z geografického hľadiska siete, má architektúra s centrálnym kontrolérom tendenciu vyšzej odozvy.

⁵SS7 (Signalling System No. 7) <http://www.corp.att.com/cpetesting/ss7.html>

⁶ Software Defined Networking, Module 2.1: Control/Data Separation

<https://class.coursera.org/sdn-001/lecture/25>



Obr. 2.2: Centralizovaná riadiaca architektúra.

Princíp distribuovanej a čiastočne distribuovanej architektúry [14, 11] je zobrazený na obrázku 2.3. V tejto architektúre sa využívajú najmenej dva riadiace prvky, ktoré spolu komunikujú a predávajú si čiastočné informácie. Každý jeden kontrolér nemusí komunikovať s každým sieťovým zariadením, ale k riadeniu mu stačia čiastočné informácie o sieťových zariadeniach. Výhodou tejto architektúry je vysoká dostupnosť zariadení a možnosť lepšieho geografického návrhu siete.

Základné typy distribuovaných a čiastočne distribuovaných architektúr sú plošné (flat) a hierarchické (hierarchical). V hierarchických typoch si riadiace prvky vymieňajú informácie na základe ich hierarchickej úrovne. Hierarchicky nižšie riadiace prvky reagujú častejšie na lokálne udalosti, keďže vyššie riadiace prvky reagujú na globálne udalosti. Hierarchické nižšie riadiace prvky sú typicky súčasťou sieťového zariadenia.

Plošná architektúra využíva separované riadiace prvky, ktoré riadia určitú geografickú oblasť. Riadiace zariadenia v plošnej architektúre si typicky vymieňajú topologické informácie siete.

2.2.1 Elementy v SDN

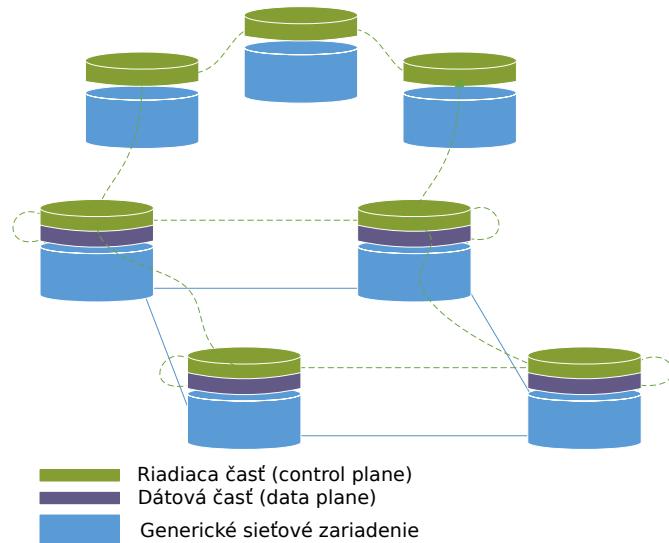
Základnými prvkami softvérom definovaných sieťach sú kontrolér (controller), sieťové zariadenie a komunikačný kanál (obr. 2.4).

Sieťové zariadenie

Sieťové zariadenie je komponent, ktorý komunikuje s kontrolérom, prijíma jeho príkazy, manipuluje s dátami atď. V závislosti od typu architektúry SDN, zariadenia umožňujú aplikovať určité operácie nad dátami.

Kontrolér (controller)

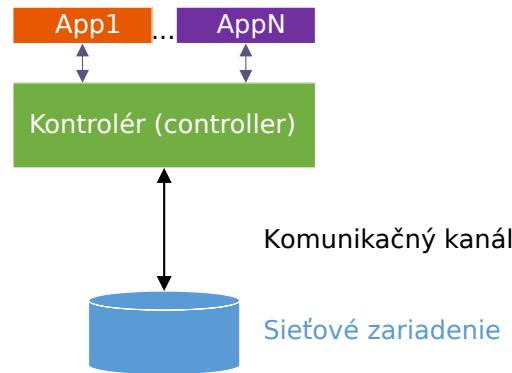
Kontrolér je medzník medzi sieťovými zariadeniami a aplikáciami. Kontrolér posiela riadiace informácie sieťovým zariadeniam, reaguje na udalosti, autentifikuje zariadenia, komunikuje s inými kontrolérmi a pod.



Obr. 2.3: Distribuovaná riadiaca architektúra.

Komunikačný kanál

Komunikačný kanál je komponent, ktorým komunikujú medzi sebou kontrolér a sietové zariadenie. Cez komunikačný kanál typicky prechádzajú riadiace a autentifikačné dátá.



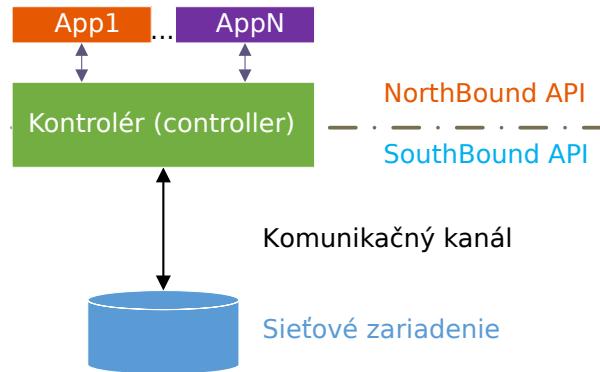
Obr. 2.4: Zobrazenie základných prvkov v SDN

2.2.2 Programové rozhrania SDN

V softvérom definovaných sieťach je riadenie zariadení rozdelené na komunikáciu, aplikácia – kontrolér a kontrolér – sietové zariadenie (obr. 2.5). Programové rozhranie medzi aplikáciou a kontrolérom sa nazýva NorthBound API a programové rozhranie medzi kontrolérom a sietovým zariadením sa nazýva SouthBound API.

SouthBound API

SouthBound API je nízko úrovňové programové rozhranie, ktoré typicky obsahuje iba nutnú funkciu na programovanie sietových zariadení.



Obr. 2.5: Zobrazenie programových rozhraní na elementoch SDN.

Typickým SouthBound API je OpenFlow (sekcia 2.3). Limitujúce vlastnosti nízko úrovňového programovacieho rozhrania v SDN sú nasledujúce⁷:

Race conditions

Správne poradie využívania operácií SouthBound API.

Nízka abstrakcia

Nižšia abstrakcia SouthBound API, sa ľahšie využíva na programovanie zložitejších funkcionálf siete.

Ľažšie vykonávanie rozličných úloh

Využitie SouthBound API pri rozličných požiadavkach na sieťovú funkcionalitu je zložité. Ako príklad na funkcionalitu siete je súčasné využívanie smerovania, prístupových práv a kvality služieb.

NorthBound API

Dôvodom na vznik NorthBound API bola reakcia na vyššie spomenuté problémy so SouthBound API. NorthBound API je programové rozhranie, ktoré komunikuje s kontrolérom a kontrolér využíva ako prostredník na programovanie sieťových zariadení. Niektoré z vlastností NorthBound API:

Nezávislosť na SouthBound API

NorthBound API umožňuje využívať heterogénne zariadenia, ktoré využívajú rozličné SouthBound API.

Rýchla zmena riadenia

Umožňuje využívať vysokoúrovňové programovacie jazyky k zmene chovania sieťových zariadení.

⁷Software Defined Networking, Module 6.1: Motivation for “Northbound APIs” and SDN Programming Languages <https://class.coursera.org/sdn-001/lecture/69>

Príklady možnosti využitia NorthBound API:

- Výpočet ciest
- Smerovanie
- Bezpečnosť

Príklady použitia NorthBound API v aplikáciách:

- Využívanie siete ako jedno virtuálne zariadenie.
- Bezpečnostné aplikácie.
- Aplikácie si rezervujú sieťové zdroje (šírka pásma, priorita . . .).

2.3 OpenFlow

Pojem OpenFlow zastrešuje súhrn protokolov, programových rozhraní a je to jeden z pohľadov na softvérom definované siete [11]. Protokoly v OpenFlow sa delia na sieťový protokol (OpenFlow) a konfiguračný protokol (OpenFlow-Config).

Sieťový protokol

Táto časť OpenFlow protokolov sa využíva na rýchlu zmenu stavu sieťových zariadení. Úlohami tohto protokolu je komunikácia s kontrolérom, reagovanie na jeho príkazy, modifikácia dátovej časti zariadenia a pod. Súčasná verzia sieťového protokolu OpenFlow je 1.4.x⁸ (9. mája 2014). Z programovej perspektívy sieťový protokol patrí pod SouthBound API.

Konfiguračný protokol

OpenFlow-Config je protokol využívaný na konfiguráciu sieťových zariadení. Tento protokol nemení dátovú časť (data plane) sieťového zariadenia, ale mení súčasti zariadenia. Príkladom použitia protokolu je zmena stavu sieťových rozhraní t.j. zapnutie respektívne vypnutie rozhrania alebo nastavenie IP adresy rozhraniu [13, 11].

2.3.1 Sieťový protokol OpenFlow

Na obrázku 2.6 sú zobrazené hlavné časti sieťového protokolu OpenFlow, OpenFlow switch (OpenFlow prepínač), controller (kontrolér) a secure channel (zabezpečený kanál).

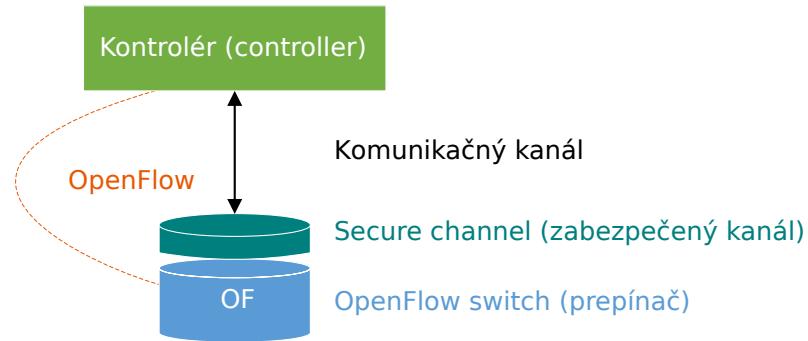
OpenFlow prepínač

Tradičné sieťové zariadenia sa pri výbere rozhrania, na ktoré budú aplikované určité operácie, rozhodujú podľa určitých informácií.

Na demonštráciu funkčnosti tradičných sieťových zariadení je použitý L2⁹ (Layer 2) prepínač, ktorý preposiela rámce na základe informácií v CAM (Content Addressable Memory) tabuľke – MAC (Media Access Control) adresy. Tieto adresy sú asociované s rozhraním

⁸OpenFlow Switch Specification 1.4.0 (Oct. 15, 2013) <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>

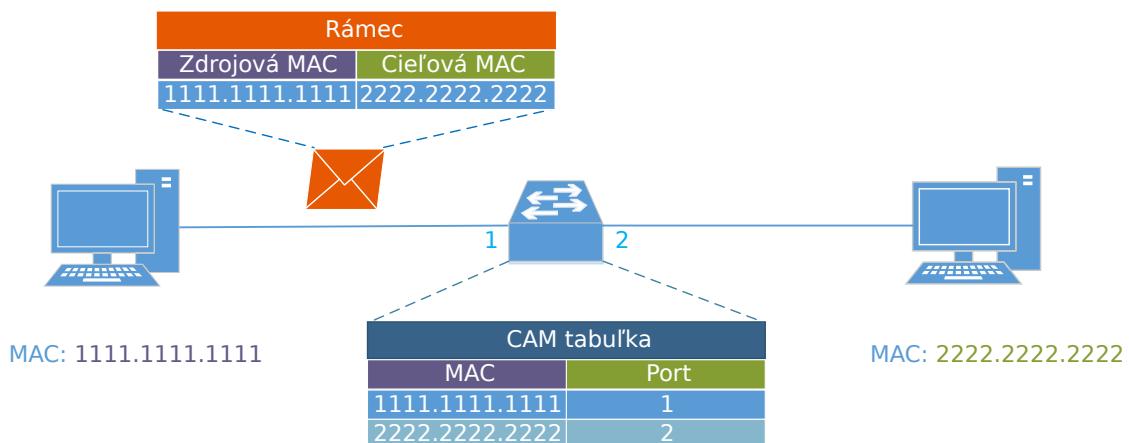
⁹Vrstva na ktorej pracuje zariadenie z pohľadu ISO/OSI.



Obr. 2.6: Časti sietového protokolu OpenFlow.

a cieľovou adresou v rámci. Jednotlivé položky CAM tabuľky sú vytvárané na základe informácií, ktoré prepínač spracuje [9].

Na obrázku 2.7 je načrtnutá funkčnosť L2 prepínača. Prepínač porovná cieľovú adresu (2222.2222.2222) prichádzajúceho rámca s položkami v CAM tabuľke. Po nájdení tejto adresy v CAM tabuľke je vykonaná operácia nad rámcom. V tomto prípade je to prenosanie prichádzajúceho rámca na port s číslom 2.

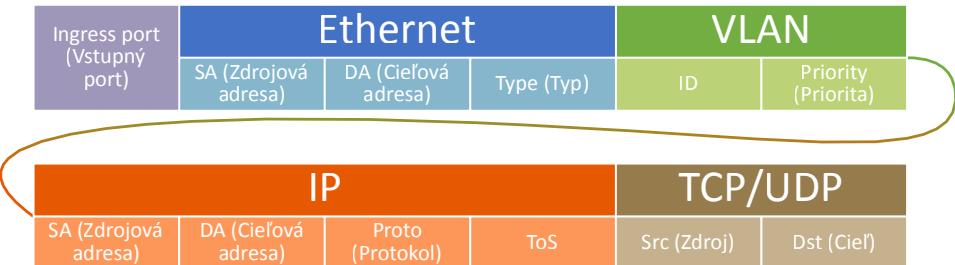


Obr. 2.7: Zobrazenie funkčností L2 prepínača.

OpenFlow prepínač rozhoduje na základe informácií, ktoré obsahuje flow table (tabuľka tokov). Táto tabuľka uchováva informácie flow (tok). Po porovnaní jednotlivých tokov v tabuľke tokov sa vykonajú operácie.

Flow

Flow (tok) sa uchováva vo flow table (tabuľke tokov) ako n-tica, ktorá obsahuje určité polia. Príklad je uvedený na obrázku 2.8. Na tomto obrázku sú zobrazené polia, ktoré sa porovnávajú voči prijatým informáciám [1].



Obr. 2.8: Polia porovnávania toku sieťového protokolu OpenFlow verzie 1.0.

Flow table

Flow table (tabuľka tokov) je základná štruktúra v OpenFlow prepínači. Na obrázku 2.9 sú zobrazené prvky, tabuľky tokov pre jeden tok, podľa štandardu sieťového protokolu OpenFlow 1.0 [1, 11]. Základné časti sú headers fields (pole hlavičiek), actions (akcie) a counters (počítadlá).

Header fields (pole hlavičiek)

Táto časť tabuľky tokov je využívaná na porovnávanie existujúcich záznamov, tokov s príchodzími dátami. Princíp tejto časti je podobný ako cieľová MAC adresa v L2 prepínačoch.

Na niektoré polia je možné aplikovať wildcard characters (zástupné znaky) typu any (každý), alebo ich maskovať.

Actions (akcie)

Actions (akcie) sú využívané na vykonanie operácie nad tokom, ktorý vyhovuje poľom hlavičiek. Z implementačného hľadiska v sietovom protokole OpenFlow verzie 1.0 sú povinné akcie forward (poslanie) a drop (zahodenie).

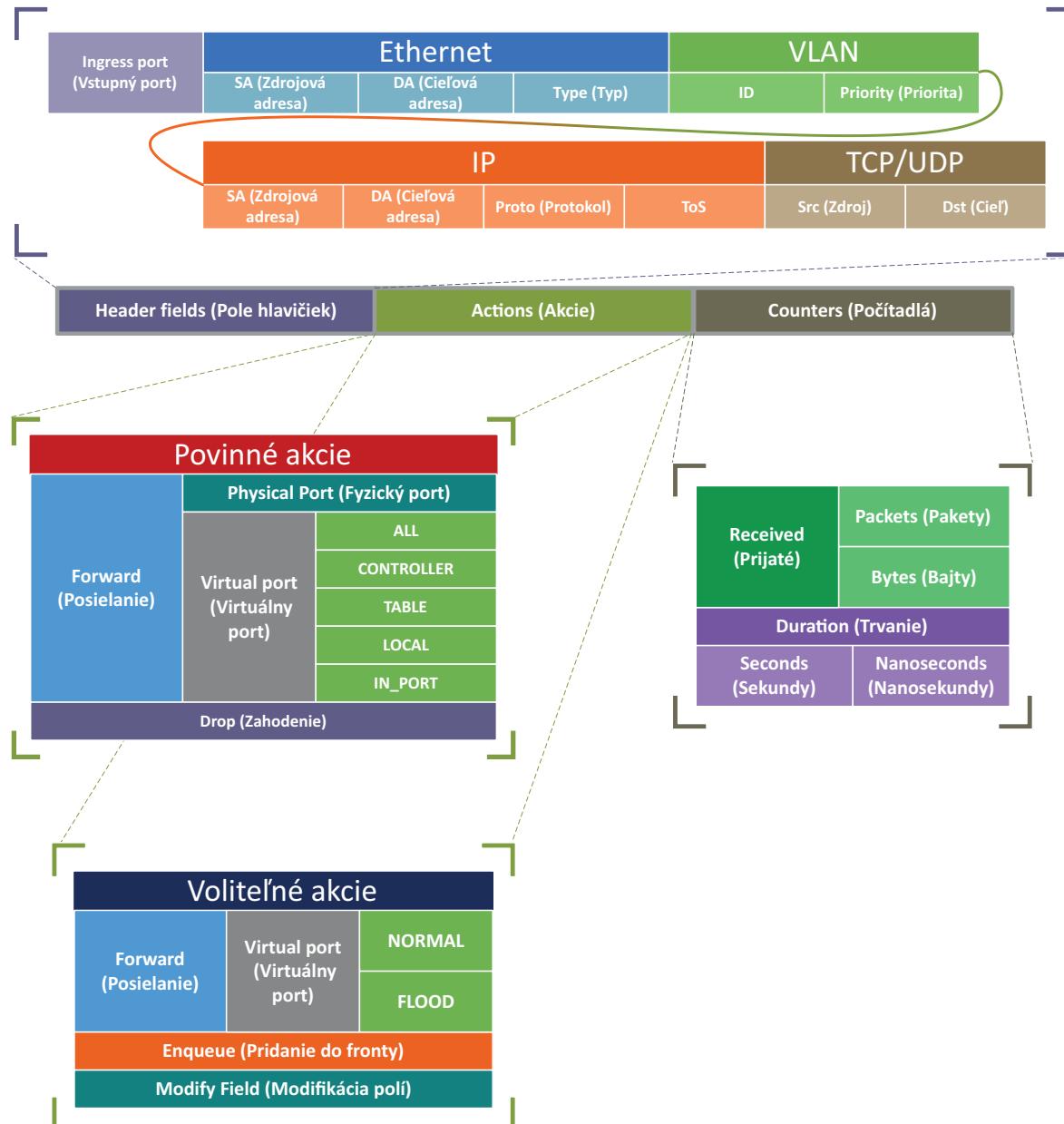
Pri vykonávaní akcie forward (poslanie) je nutné špecifikovať fyzicky alebo virtuálny port. Virtuálne porty sú nasledujúce ALL, CONTROLLER, LOCAL, TABLE, IN_PORT. Ďalšie virtuálne porty sú NORMAL a FLOOD. Tieto virtuálne porty sú voliteľné.

Voliteľné akcie podľa štandardu 1.0 sú enqueue (zaradenie do fronty) a modify field (modifikácia polí).

Counters (počítadlá) ¹⁰

V tejto časti tabuľky tokov sú uchovávané štatistiky o toku.

¹⁰Obrázok 2.9 v sekcií Counters (Počítadlá) nezahrnuje všetky časti počítadiel stanovené štandardom OpenFlow 1.0.



Obr. 2.9: Štruktúra toku podľa sieťového protokolu OpenFlow 1.0.

Secure channel (zabezpečený kanál)

Z pohľadu funkcionality zabezpečený kanál spĺňa vlastnosti, ktoré sú uvedené ako komunikačný kanál v sekcií 2.2.1.

Controller (kontrolér)

OpenFlow kontrolér spĺňa vlastnosti, ktoré sú uvedené v sekcií 2.2.1.

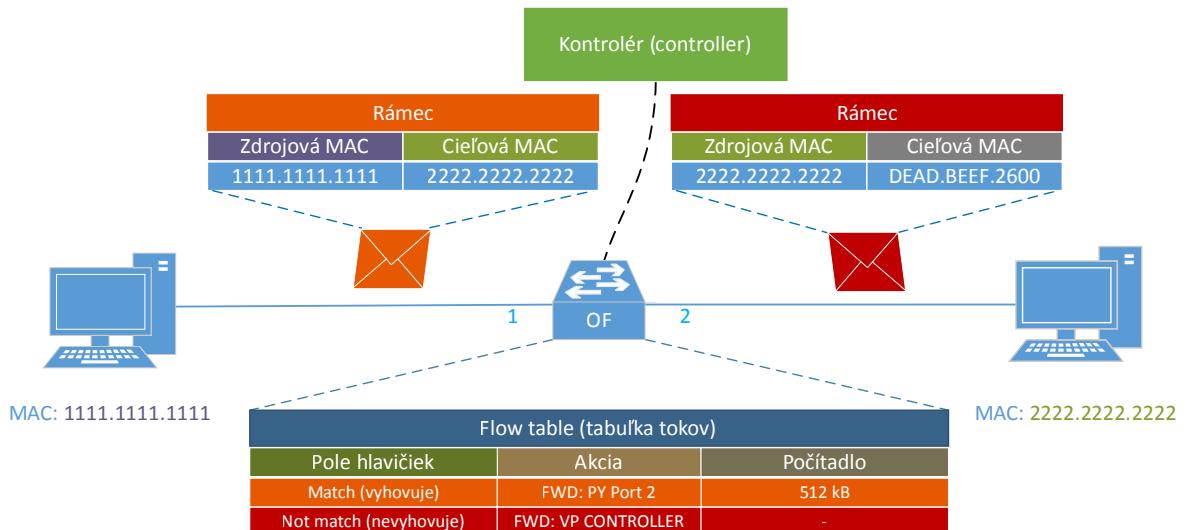
Proces riadenia

V OpenFlow prepínači proces modifikácie tabuľky tokov je odlišný ako v tradičných sieťových prvkoch¹¹.

Ak má prepínač spracovať rámec pre ktorý už je vytvorený záznam v tabuľke tokov, tak rámec bude spracovaný podľa akcie, ktorá zodpovedá danému toku. Ak má prepínač spracovať rámec, ktorý sa nenachádza v tabuľke tokov, tak prepínač rámec odošle kontroléru. Kontrolér spracuje rámec a prepínaču odošle informácie, ktoré budú zapísané do tabuľky tokov.

Na obrázku 2.10 je zobrazené chovanie OpenFlow prepínača a kontroléra. Hlavičky príchadzieho rámca (zdrojová MAC: 1111.1111.1111, cieľová MAC: 2222.2222.2222) sú porovávané s údajmi v tabuľke tokov. V tomto prípade je nájdený tok, ktorý vyhovuje toku v tabuľke tokov. Na základe akcie, ktorá je asociovaná s vyhovujúcim tokom, prepínač vykoná akciu, rámec bude odoslaný na port s číslom 2.

V druhom prípade, ak prepínač má spracovať rámec (zdrojová MAC: 2222.2222.2222, cieľová MAC: DEAD.BEAF.2600), ktorému hlavičky nevyhovujú záznamom v tabuľke tokov, prepínač následne odošle tento rámec do kontroléra. Kontrolér po spracovaní rámca odošle informácie OpenFlow prepínaču o spracovaní daného toku.

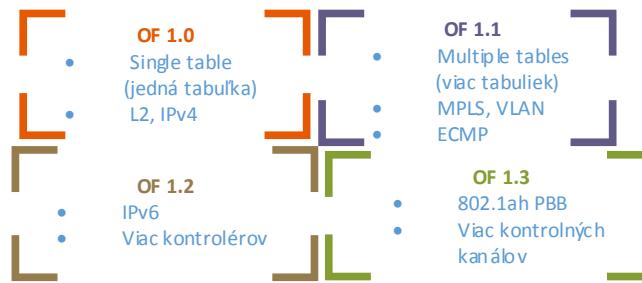


Obr. 2.10: Zobrazenie funkčnosti OpenFlow prepínača.

Porovnanie verzií sietového protokolu OpenFlow

Sieťový protokol OpenFlow od roku 2008 prešiel mnohými úpravami. Aktuálna verzia, OpenFlow 1.4.X (9. mája 2014), obsahuje podporu IPv4, IPv6, MPLS, VLAN, ECMP, 802.1ah, podporu viacerých kontrolérov atď [5] (obr. 2.11).

¹¹Software Defined Networking, Module 4.1: The Control Plane <https://class.coursera.org/sdn-001/lecture/57>

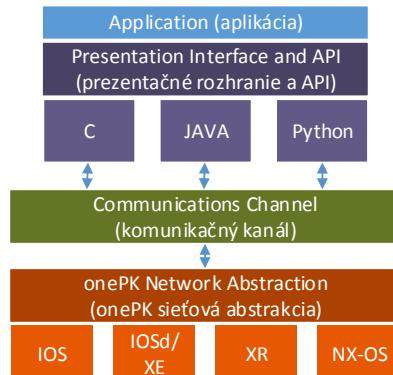


Obr. 2.11: História niektorých zmien v sieťovom protokole OpenFlow.

2.4 onePK

Cisco onePK (one Platform Kit) je nástroj, ktorý umožňuje programovo pristupovať, rozširovať a modifikovať funkcia funkcia sieťových zariadení Cisco Systems. [3] V prípade Cisco onePK neimplementuje podobné hardvérové súčasti ako je tabuľka tokov v OpenFlow, ale zmeny funkcia funkcia sieťových zariadení sú reprezentované v podobe aplikácií. Tieto aplikácie volajú procedúry, na ktoré zariadenie reaguje.

Z programového hľadiska SDN patrí onePK k SouthBound API. Nástroj onePK obsahuje tri elementy (obr. 2.12) [3]:



Obr. 2.12: Zjednodušená architektúra Cisco onePK.

Prezentačná vrstva (presentation layer)

Prezentačná vrstva onePK obsahuje súbor API knižníc, ktoré programátorovi umožňujú využívať funkcia funkcia sieťových zariadení v aplikáciách.

onePK API infraštruktúry (onePK API infrastructure)

Tento element poskytuje vnútornú komunikáciu so sieťovými zariadeniami. Cisco onePK sa snaží abstrahovať funkcia funkcia nad rozličnými softvérovými platformami.

Komunikačný kanál (communication channel)

Komunikačný kanál sa využíva na komunikáciu medzi aplikáciou a sieťovým zariadením, ako je to uvedené v sekcií [2.2.1](#).

Umiestnenie aplikácie

Vlastnosťou onePK je umožnenie behu aplikácie v rozličných umiestneniach [4, 2, 3].

Process hosting (procesové hostovanie)

V tomto prípade je aplikácia spúštaná v sieťovom zariadení. Na izoláciu medzi aplikáciou a operačným systémom sieťového zariadenia sa využívajú linuxové kontajnery¹² (containers). Komunikácia medzi operačným systémom zariadenia a aplikáciou využíva medziprocesovú komunikáciu IPC (Inter-Process Communication).

Blade hosting

K umiestneniu aplikácie sú využívané rozširujúce zásuvné moduly sieťových zariadení.

End-Point hosting (koncový bod)

Aplikácie je možno spúštať na externom zariadení.

2.4.1 Architektúra onePK

Na obrázku [2.13](#)¹³ sú zobrazené detailné časti architektúry Cisco onePK [4]. Hlavnými prvkami tejto architektúry sú service sets (servisné sady), application (aplikácia) a agent (agent).

Service sets (servisné sady)

Nástroj onePK je rozdelený na súbor vlastností, ktoré umožňujú programátorovi vytvárať aplikácie. Tieto vlastnosti sú rozdelené do takzvaných servisných sad.

Element

Táto servisná sada je jedna zo základných sád, ktorá umožňuje vytvárať určité základné štruktúry. Tieto štruktúry sa používajú v ďalších servisných sadách.

Discovery

Discovery servisná sada je využívaná na vytváranie topologických informácií. Vo verzii onePK 1.1.0 je využívaný protokol CDP.

Utility

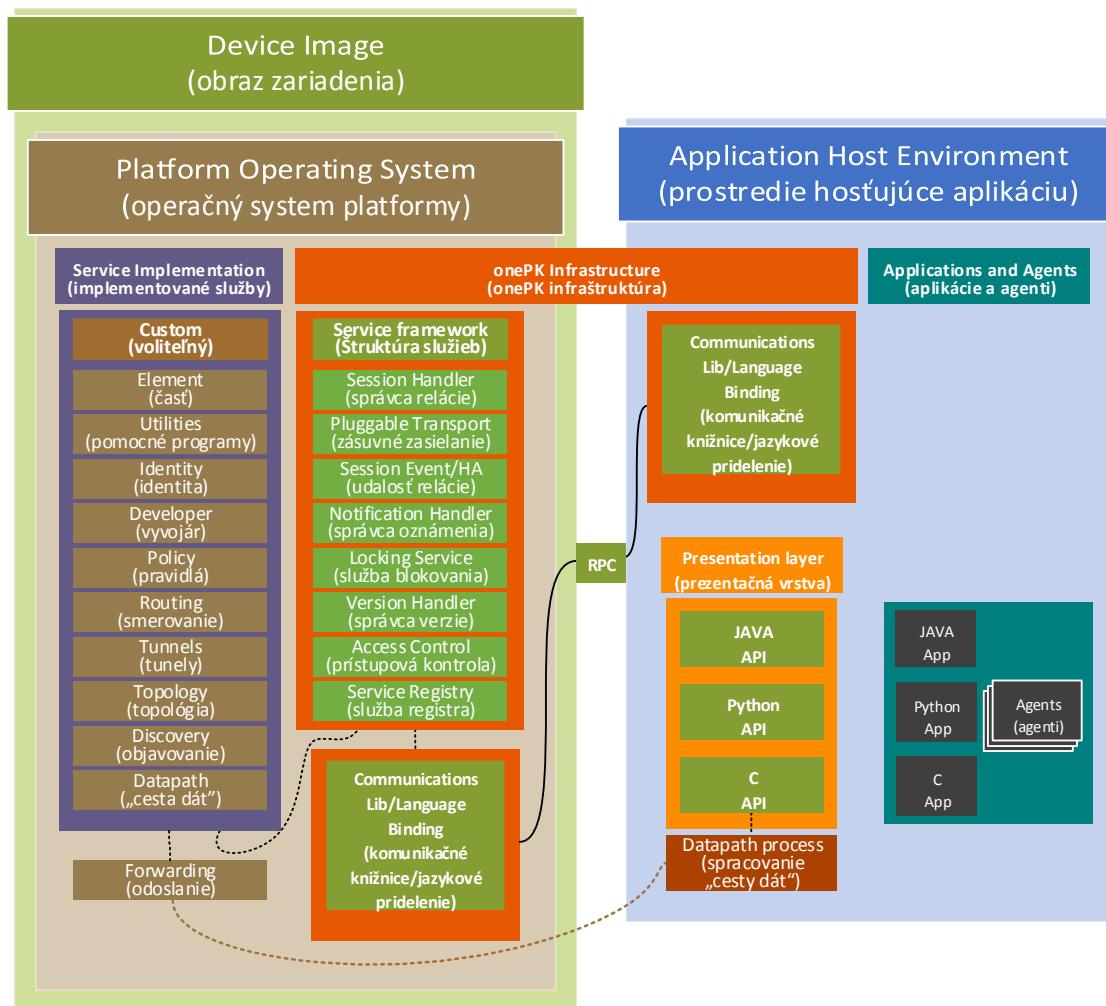
Utility servisná sada slúži na prácu s udalosťami Syslog a na využívanie autentizačného, autorizačného a účtovacieho, AAA (Authentication, Authorization and Accounting), rozhrania. Ďalšou súčasťou tejto servisnej sady je VTY. VTY servisná sada umožňuje manipuláciu so sieťovým zariadením podobne ako je to v prípade využitia služieb SSH alebo TELNET. Táto servisná sada modifikuje takzvaný running config.

Developer

Táto servisná sada umožňuje získať informácie o behu a stave aplikácie.

¹²Kontajnery separujú viaceré inštancie systémov na jednom kontrolnom hostiteľskom systéme.

¹³K prezentácnej vrstve a k vrstve aplikácií a agentov sú pridané časti týkajúce sa Python API. Python API bola pridaná až vo verzii 1.1.0.



Obr. 2.13: Architektúra Cisco onePK verzie 1.0.0.

DataPath

Servisná sada DataPath umožňuje kopírovať, modifikovať, presmerovať údaje. Túto servisnú sadu je možné využiť iba za použitia C API.

Policy

Táto servisná sada ma za úlohu filtrovanie dát so špecifickými atribútmi. Príkladom sú prístupové listy, ACL (Access Control List).

Routing

Routing servisná sada umožňuje pristupovať k sieťovým cestám zariadenia a pridať vlastné sieťové cesty.

Agent (agent)

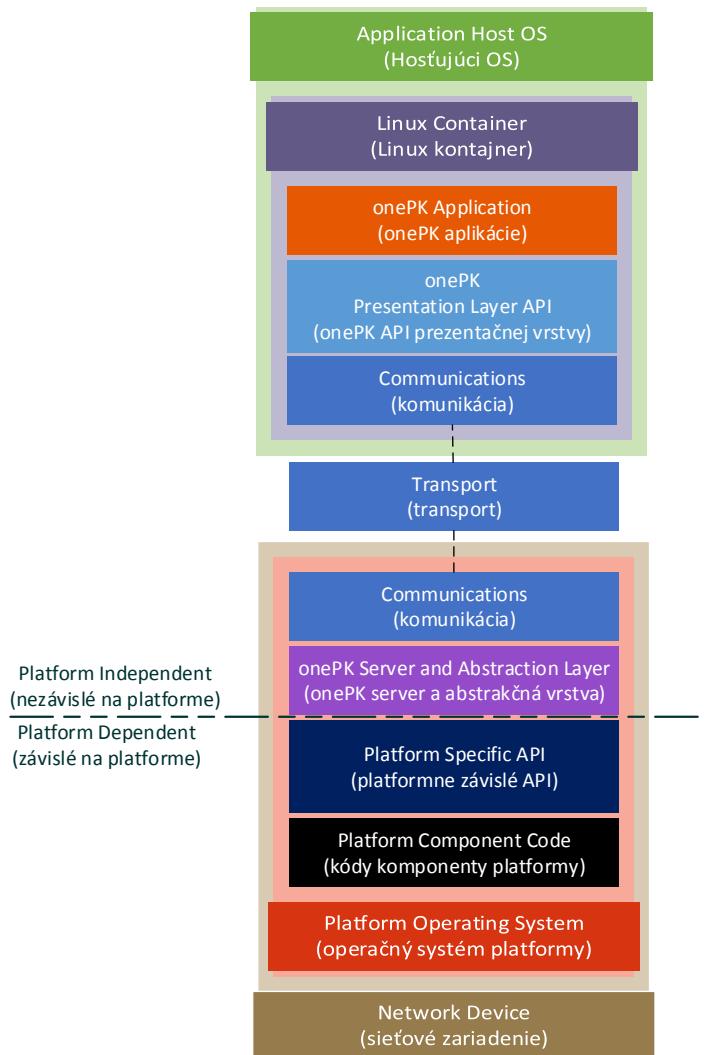
Agent v onePK je špeciálna aplikácia, ktorá komunikuje s operačným systémom zariadenia. Takáto aplikácia umožňuje implementovať ďalšie protokoly a rozhrania do zariadenia.

Príklady agentov, ktorí implementujú rozhrania a protokoly tretích strán sú OpenFlow, I2RS a Puppet. Pre priblíženie Agent využíva prvky onePK k implementovaniu vlastnej funkcionality.

Application (aplikácie)

Diagram na obrázku 2.14 ilustruje hostovanie aplikácie. Aplikácia je umiestnená v linuxovom kontajneri, kde využíva k svojej činnosti API prezentačnej vrstvy Cisco onePK. OnePK prezentačná vrstva komunikuje so zariadením cez transportnú vrstvu, ktorá u onePK je reprezentovaná ako vzdialené volanie procedúry, RPC (Remote Procedure Call). Na strane zariadenia sa nachádzajú časti systému, ktoré reagujú na príchodzie požiadavky. Niektoré časti systému sú závislé na platforme.

K vývoju aplikácie onePK prezentačná vrstva umožňuje využívať programovacie jazyky C, JAVA. Vo verzii 1.1.0 bola pridaná verejná podpora programovacieho jazyka Python.

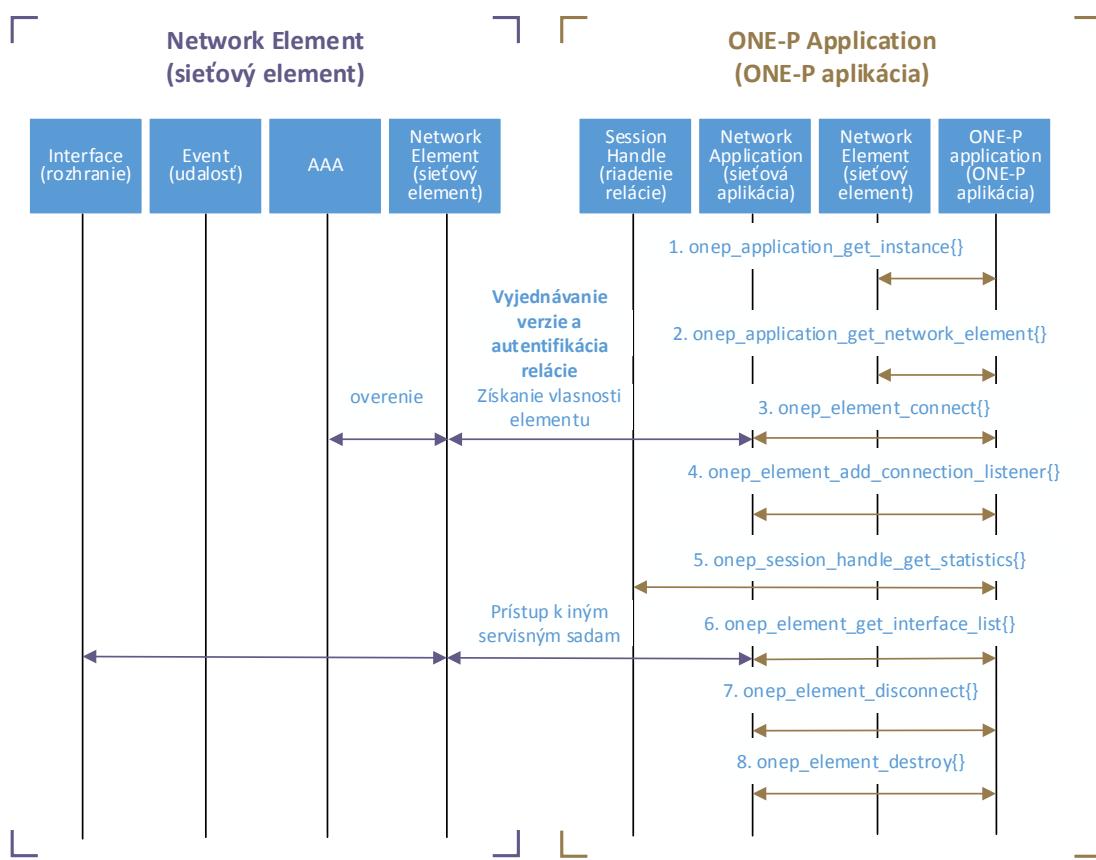


Obr. 2.14: Aplikačná časť architektúry onePK.

2.4.2 Princíp použitia

Na komunikáciu, ako bolo vyšie uvedené, onePK využíva RPC. Zariadenie aplikácií po úspešnom AAA procese umožňuje využívať svoje prostriedky.

Príklad použitia je zobrazený na obrázku 2.15. [4] V kroku 1 je vytvorená inštancia aplikácie. V kroku 2 sú inicializované štruktúry, ktoré v kroku 3 slúžia na pripojenie ku zariadeniu. Na strane zariadenia sa overujú autentifikačné informácie. V kroku 4 je zavolaná procedúra, ktorá má za úlohu reagovať na udalosti s pripojením na sieťový element. 5. krok slúži na získanie štatistik ohľadom relácie. V 6. kroku je zavolaná procedúra, ktorá vráti zoznam rozhraní na zariadení. 7. a 8. krok slúži na odpojenie sieťového elementu z aplikácie a uvoľnenie štruktúr, ktoré boli inicializované v kroku 2.



Obr. 2.15: Diagram komunikácie onePK.

Pri vytváraní aplikácií v technológií onePK je možné pri určitých elementoch nastavovať životnosť. Táto životnosť nastavuje, ktoré prostriedky budú uvoľnené po odpojení aplikácie. Príkladom na túto životnosť sú prístupové zoznamy. Ak sa nastaví životnosť prístupového zoznamu na trvalú, tak po odpojení aplikácie zo zariadenia, prístupové zoznamy nebudú uvoľnené zo zariadenia.

2.5 Zhodnotenie

V predchádzajúcich sekciách boli spomenuté základné princípy softvérom definovaných sietí. Tieto nové princípy prinášajú nové pohľady na dátové počítačové siete. Ako každá vec, tak aj tieto pohľady prinášajú s množstvom výhod aj množstvo nevýhod. Medzi niektoré výhody týchto pohľadov patrí jednoduchšia manipulácia so sieťovými zariadeniami a implementovanie vlastných funkcionálít do zariadení. K dosiahnutiu týchto výhod je v niektorých prípadoch nutné zabezpečiť podporné (out of band) siete a vysporiadať sa s problémami synchronizovanosti kontrolérov.

Z pohľadu OpenFlow, kde OpenFlow prepínač obsahuje flow table (tabuľku tokov) na základe, ktorej sa rozhodujú aké operácie budú vykonané nad dátami. Tieto operácie sú ovplyvňované kontrolérom.

Pre oboznámenie s technológiou OpenFlow je odporúčané využívať Mininet¹⁴, ktorý obsahuje Open Vswitch¹⁵. Ďalšou nutnou súčasťou pre oboznámenie sú kontroléry, v súčasnej dobe je ich veľké množstvo. Medzi najznámejšie kontroléry patria NOX¹⁶, POX¹⁷, Trema¹⁸, Ryu¹⁹. V poslednej dobe sa do popredia dostáva kontrolér OpenDayLight²⁰, ktorý je pod záštitou Linux Foundation.

Na druhú stranu Cisco onePK umožňuje pridávať funkcionality na existujúce sieťové zariadenia v podobe aplikácií. K týmto funkcionalitym je možné pridať podporu ďalších sieťových štandardov ako je OpenFlow.

Cisco poskytuje svoj vlastný kontrolér XNC (eXtensible Network Controller)²¹, ktorý je postavený na OpenDayLight.

Softvérom definované siete ovplyvnili mnoho častí sieťového odvetvia. Na základe týchto pohľadov boli modifikované alebo vytvorené ďalšie pohľady. Jeden z ďalších pohľadov je SDX (Software Defined Internet Exchange), ktorý využíva SDN myšlienky v peeringových centrách²².

¹⁴Mininet je nástroj, ktorý virtualizuje sieťové prvky. <http://mininet.org/>

¹⁵Open Vswitch je virtuálny prepínač, ktorý umožňuje pracovať so OpenFlow. <http://openvswitch.org/>

¹⁶NOX <http://www.noxrepo.org/nox/about-nox/>

¹⁷POX <http://www.noxrepo.org/pox/about-pox/>

¹⁸Trema <http://trema.github.io/trema/>

¹⁹Ryu <http://osrg.github.io/ryu/>

²⁰OpenDayLight <http://www.opendaylight.org/>

²¹XNC <http://www.cisco.com/go/xnc>

²²SDX (Software Defined Internet Exchange) <http://www.ietf.org/proceedings/86/slides/slides-86-sdnrg-6.pdf>

Kapitola 3

Kvalita služieb, QoS

Ako je spomenuté v úvode (kapitola 1), na dátové počítačové siete a sieť Internet sa kladú čoraz väčšie požiadavky. Tieto požiadavky sú spojené ako s prenosovou kapacitou infraštruktúr, tak ako aj s kvalitatívou úrovňou infraštruktúr.

Z pohľadu užívateľa dátových sieti je nutné udržiavať infraštruktúru dátových sieti, aby kvalitatívna úroveň aplikácií, vyžívajúcich dátové siete, dosahovala akceptovateľnej úrovne. Túto úroveň sietových infraštruktúr sa snažia docieliť rôzne architektúry kvality služieb, QoS (Quality of Services).

Architektúry pristupujú k problematike kvality služieb odlišne. Integrované služby pristupujú ku kvalite služieb z globálneho hľadiska, kde aplikácia si rezervuje určité prostriedky zariadení a následne sa vytvára „rezervačná“ sieťová cesta k požadovanému cieľu.

V prípade diferencovaných služieb sú dátá priamo označené a na základe označenia zariadenie vykonáva nezávisle operácie nad dátami. Príkladom použitia diferencovaných služieb je využívanie multimediálnych a textových dát. Tieto typy dát sa označia rozličnými značkami a každé sieťové zariadenie, ktoré príde do styku s dátami umožňuje, v závislosti od pridelenej značky, vykonať odlišnú operáciu nad dátami.

Novšie prístupy kvality služieb sa iba nezameriavajú na určitú časť, ale kombinujú tieto princípy a pridávajú nové vlastnosti.

Nasledujúce časti kapitoly sa venujú ďalším a vyššie spomínaným prístupom kvality služieb.

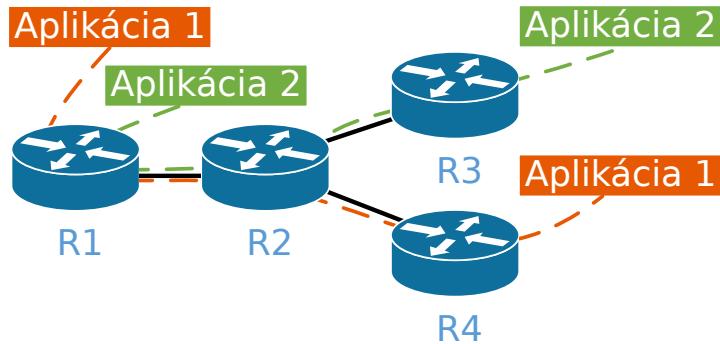
3.1 Integrované služby, InterServ

Ako bolo v úvode kapitoly spomenuté Integrované služby rezervujú zdroje sieťových zariadení. Obrázok 3.1 ilustruje funkcionality integrovaných služieb. Aplikácie, ktoré komunikujú so smerovačom (router) majú určité požiadavky na zariadenie. Na základe cieľovej adresy komunikácie sa tieto požiadavky pomocou signalizačného protokolu distribuujú ďalším zariadeniam. Táto distribúcia využíva existujúce smerovacie údaje zariadenia. Po vytvorení cesty a prijatími požiadavku zariadením sa začne samotná komunikácia aplikácií.

V tomto prípade¹ aplikácia 1, ktorá komunikuje so smerovačom R1, chce komunikovať s aplikáciou 1, na strane smerovača R4. Zariadenie na strane R1, ktoré obsluhuje aplikáciu 1 odošle signalizačným protokolom správu pre vytvorenie cesty. Táto správa je distribuovaná až ku zariadeniu aplikácie 1 smerovača R4. Po úspešnom vytvorení cesty až k aplikácii 1

¹Správanie procesu Integrovaných služieb je inšpirované protokolom RSVP.

smerovača R4, je späť poslaná správa po predom vytvorenej ceste o požiadavkách aplikácie. Po tomto procese začne samostatná komunikácia aplikácie 1 medzi jednotlivými prvkami.



Obr. 3.1: Princíp integrovanej služby.

Integrované služby implementujú na svoj chod signalizačné protokoly, admission control (kontrola prístupu), classifier (klasifikátor) a packet scheduler (plánovač dát) [15, 16].

Aby bolo možné obsluhovať požiadavky aplikácií na sieťové zariadenie je nutné zaviesť služby, ktoré definujú správanie zariadení pri alokácii požiadaviek a obsluhe dát.

Služby integrovaných služieb sú rozdelené do dvoch skupín Guaranteed Service (garantovaná služba) a Controlled Load Service (kontrolovaná služba zaťaženia).

Garantovaná služba je navrhnutá pre aplikácie, ktoré vyžadujú striktnú šírku pásma a oneskorenia. Príkladom pre tento typ sú aplikácie reálneho času napr. sledovanie živého prenosu [15, 16].

Kontrolovaná služba zaťaženia sa na problematiku rezervácie zdrojov pozera inak. Dôvod na iný pohľad je ten, že garantovaná služba nezohľadňuje možnosti nárazovej (burst) zmeny dát a v niektorých prípadoch je náročne určiť presné požiadavky aplikácie. [15, 16] Kontrolovaná služba namiesto explicitného rezervovania požiadaviek emuluje určité zaťaženie sieťovej infraštruktúry. Tento typ služby sa využíva pre adaptívne aplikácie, ktoré vyžadujú menej striktné požiadavky [15, 16].

Signalizačné protokoly

Signalizačné protokoly v Integrovaných službách sú určené na komunikáciu medzi jednotlivými sieťovými zariadeniami. Najznámejší signalizačný protokol v integrovaných službách je RSVP (Resource ReSerVation Protocol). V sekcií 3.1.1 sú predstavené niektoré prvky protokolu RSVP.

Kontrola prístupu (admission control)

V integrovaných službách plní úlohu monitorovania alokovaných zdrojov a pridelovania ďalších.

Klasifikátor (classifier)

Klasifikátor sa využíva ako prostriedok na klasifikovanie dát. Ďalšou úlohou klasifikátora v integrovaných službách je uloženie klasifikovaných dát do určitej fronty [16].

Plánovač dát (packet scheduler)

Plánovač dát je časť Integrovaných služieb, ktorá je zodpovedná za dodržiavanie požiadaviek aplikácií. Jedna z jeho úloh je prioritizovanie určitých dát pred inými [15, 16].

V dôsledku nutnej implementácií vyššie spomenutých časti integrovaných služieb, a s tým spojené veľké zaťaženie sieťových zariadení, s malou škálovateľnosťou, sa táto architektúra kvality služieb neujala [16].

3.1.1 RSVP

RSVP (Resource ReserVation Protocol) je signalizačný protokol, ktorý je využívaný v architektúre Integrovaných služieb. Dôvodom na jeho použitie v Integrovaných službách je fakt, že v tejto architektúre je nutné rezervovať zdroje zariadení pred vysielaním samotných dát.

RSVP rezervuje dáta simplexne, to znamená, že pri využívaní obojsmernej komunikácie je nutné vytvárať obojstranné rezervácie.

V protokole sú využívané existujúce smerovacie údaje zariadení. Rezervácie protokolom je nutné obnovovať, no po uplynutí nejakého časového úseku rezervácia bude zrušená, a zdroje využívané v tejto rezervácii budú navrátené zariadeniu k opäťovnému použitiu.

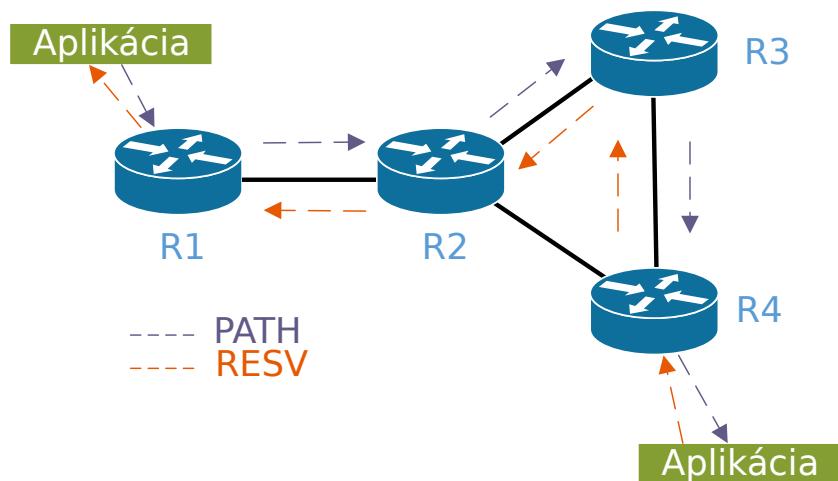
Základnými prvkami v tomto signalizačnom protokole sú správy PATH a RESV. Správy sú posielané postupne po uzloch (hop by hop).

Správa PATH je odosielaná z prvku unicastom alebo multicastom k príjmateľovi dát. V správe PATH sú uložené informácie o predchádzajúcom uzle PHOP (Previous Hop), informácie o identifikácii dátového toku a parametre pre Token Bucket (TSpec). Po úspešnom prijatí PATH správy príjmateľovi, príjmateľ začne odosielať RESV správu. Správa RESV využíva PHOP zo správy PATH k použitiu vyhradenej cesty.

RESV správa obsahuje objekty FlowSpec a FilterSpec. FlowSpec zapúzdruje informácie potrebné pre plánovač dát (packet scheduling). Súčasťou tohto objektu môžu byť informácie ohľadom rezervačných QoS požiadaviek (Rspec) a údajov, ktoré definujú tok dát. Tieto dodatočné informácie závisia od použitých služieb². [15]

Objekt FilterSpec definuje množinu dát, na ktoré sú aplikované údaje z objektu FlowSpec.

Na obrázku 3.2 sa nachádza zjednodušená demonštrácia procesu protokolu RSVP.



Obr. 3.2: Princíp protokolu RSVP.

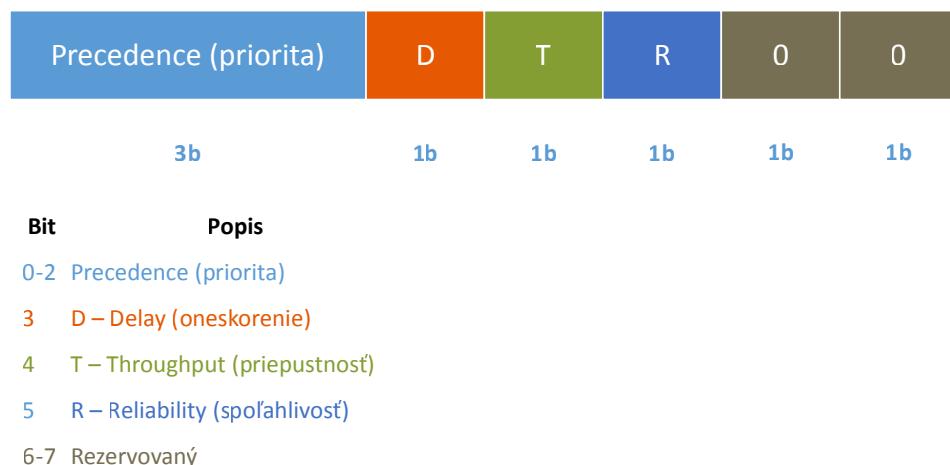
²Garantovaná služba alebo Kontrolovaná služba zaťaženia.

3.2 Diferencované služby, DiffServ

Ako reakcia na nedostatky Integrovaných služieb a nutnosť udržiavať kvalitatívnu úroveň aplikácií vznikla odlišná architektúra kvality služieb.

Táto architektúra sa od architektúry Integrovaných služieb odlišuje tým, že na základe označenia IP paketov umožňuje každému zariadeniu ich nezávislú manipuláciu. Táto manipulácia sa nazýva PHB (Per Hop Behaviour). Ďalším rozdielom je to, že Diferencované služby k svojej činnosti nevyužívajú signalizačný protokol ako v prípade integrovaných služieb.

Na označenie IPv4 paketov sa v hlavičke samotného paketu využívalo pole ToS (Type of Service). Hlavička ToS (obr. 3.3) obsahuje 8 bitov. Prvé (z ľava) 3 bytov sú priorita (precedence), nasledujúce 3 byty typ služby (Type of Service) a posledná časť hlavičky je nevyužívaná. Časť priorita bola využívaná na definovanie priority dát. V druhej trojbitovej časti, typ služby sa definuje správanie nad dátami ako je oneskorenie, prieplustnosť a spoľahlivosť. Neskôr hlavička ToS bola pozmenená na hlavičku DS (Differentiated Service). [15, 16, 12]



Obr. 3.3: Pole hlavičky ToS.

DS hlavička (obr. 3.4) obsahuje 8 bitov, kde prvé (z ľava) 6 bytov sú označené ako DSCP (Differentiated Service Code Point). Ďalšie 2 byty sú označené ako ECN (Explicit Congestion Notification). Časť DSCP slúži na identifikáciu paketov pre zariadenia, ECN časť je využívaná k informovaniu transportných protokolov o zahltení sieťovej infraštruktúry. [15, 16, 17]



Obr. 3.4: Pole hlavičky DS.

K dosiahnutiu určitého správania PHB v architektúre diferencovaných služieb existuje viacero PHB skupín. Tieto PHB skupiny doporučujú určité chovanie sieťových zariadení. K najpoužívanejším sú radené Assured Forwarding (AF), Expedited Forwarding (EF).

Expedited forwarding je využívané pre siete, v ktorých je nutné dodržiavať nízku stravovosť a nízke oneskorenie dát. Dáta, ktoré využívajú EF PHB by mali byť označené v poli

DSCP hodnotou $(101110)_2$. V zariadeniach pre splnenie požiadaviek EF sa typicky využívajú prioritné fronty. Príkladom protokolov, ktoré využívajú EF PHB sú smerovacie protokoly a protokoly reálneho času. [15]

Assured Forwarding (AF) PHB je reprezentované ako AF xy . Kde x je trieda prevádzky a y priorita zahodenia. Definované sú štyri triedy prevádzky, pričom každá obsahuje tri priority zahodenia. Triedy prevádzky majú priority, kde AF4 y má vyššiu prioritu ako AF1 y . Čo sa týka priority zahodenia, tak AF41 ma menšiu prioritu zahodenia ako AF43. Typicky AF je implementovaný ako WFQ (Weighted Fair Queue) 3.4.3. [15]

Pre spätnú kompatibilitu medzi hlavičkou DS a precedence častou v hlavičke ToS bola vytvorená CS (Class Selector) PHB ³.

3.3 Performance Routing, Pfr

Performance routing od firmy Cisco Systems je technológia, ktorá umožňuje inteligentné dynamické smerovanie dát v IP sieťach na základe latencie, stratovosti dát, zaťaženia, ceny a pod. K svojej činnosti využíva IP SLA⁴ alebo NetFlow⁵.

Táto technológia základnými princípmi pripomína SDN princípy, ktoré sú uvedené v sekcií 2.2. Hlavné časti tejto technológie sú kontrolér (master controller) a okrajový smerovač (border router).

Úlohou kontroléra je modifikovanie smerovania okrajových smerovačov. Samotná modifikácia smerovania je vykonávaná pomocou PBR alebo vnútornou modifikáciou údajov smerovacích protokolov. Pfr umožňuje nastaviť kontrolér v učiacom alebo explicitnom móde.

Okrajový smerovač zbiera informácie z Netflow alebo z IP SLA sondy a reaguje na príkazy kontroléra. [6]

3.4 Zaistenie kvality služieb na zariadeniach

Na zaistenie implementácie kvality služieb, ktorá bola popísaná vyššie v tejto kapitole, je nutné využívať určité hardvérové a softvérove prvky, ktoré umožňujú dosiahnuť takúto funkcionality.

Medzi základné oblasti týchto prvkov patria:

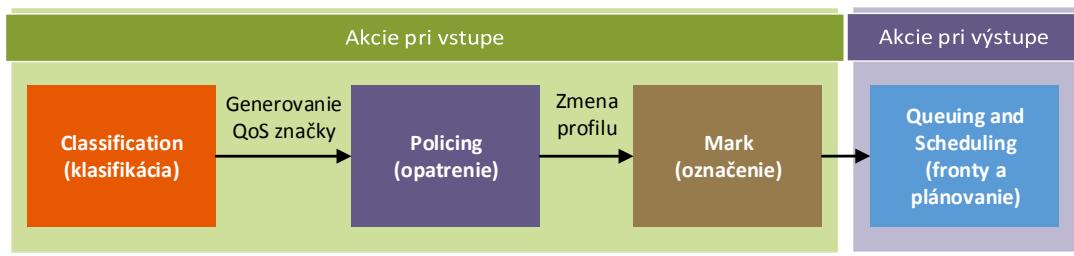
- Classification and Marking (rozlíšenie a označenie dát)
- Policing and Shaping (správa dátového toku, obmedzovanie a vyhradzovanie)
- Queuing (správa zahltenia)
- Congestion Avoidance (zabránenie zahltenia)
- Signaling (signalizácia)
- Link Efficiency Management (správa efektívneho využitia linky)

Na obrázku 3.5 sú zobrazené akcie pri vykonávaní jednoduchého procesu kvality služieb.

³Rezervované kódy DSCP pre spätnú kompatibilitu sú [xxx000].

⁴IP SLA umožňuje zisťovať kvalitatívnu úroveň sieťovej infraštruktúry. <http://cisco.com/go/ipsla>

⁵NetFlow je nástroj, ktorý umožňuje vytvárať štatistiky ohľadom dát. <http://cisco.com/go/netflow>



Obr. 3.5: Schéma jednoduchého QoS procesu.

3.4.1 Rozlíšenie a označenie dát

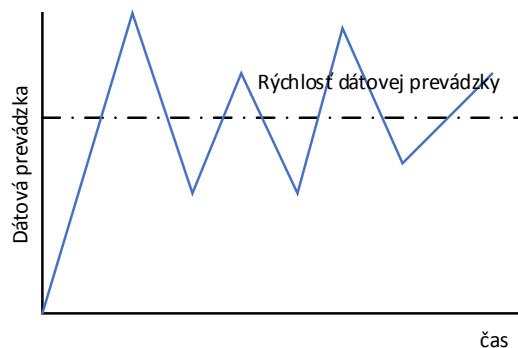
V architektúre integrovaných služieb (sekcia 3.1) sú na identifikáciu dát využívané: typ protokolu, zdrojové a cieľové informácie IP adres a portov. Tieto informácie sú pomocou signalizačného protokolu distribuované k cieľovým zariadeniam. [15]

V prípade diferencovaných služieb (sekcia 3.2) k rozlíšeniu dát sú využívané polia ToS respektíve DS. Na identifikáciu dát v tomto prípade je možné využívať niektoré informácie L1 až L7 vrstvy ISO/OSI⁶. Proces identifikácie dát a následne označenie v diferencovaných službách sa typicky vykonáva na okrajových sieťových zariadeniach v doméne. [15]

3.4.2 Správa dátového toku

Pre spravovanie dátového toku existujú dva princípy obmedzovanie a vyhradzovanie. Rozdiel medzi obmedzovaním (policing) a vyhradzovaním (shaping) je ten, že pri nedodržaní stanovených podmienok, budú dátá rozdielne spracované. V prípade obmedzovania, dátá budú zahodené a pri vyhradzovaní budú dátá ukladané do fronty.⁷

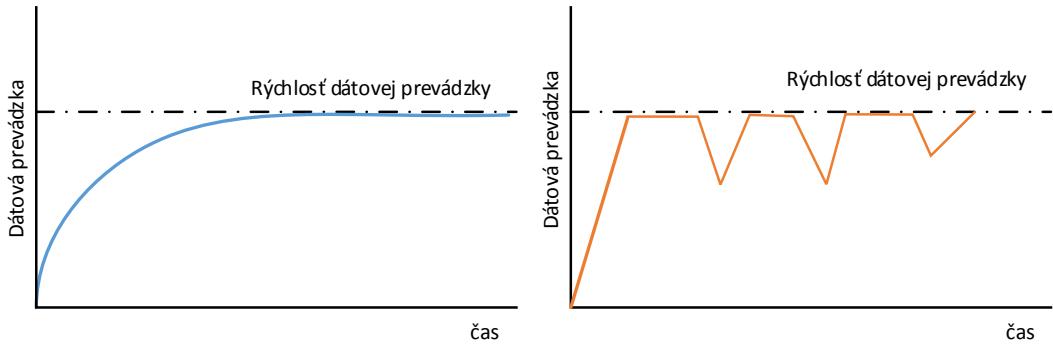
Závislosť 3.6 zobrazuje neupravenú rýchlosť dátovej prevádzky. Na obrázku 3.7 je zobrazená upravená závislosť metódou vyhradzovaním (shaping) a obmedzovaním (policing).



Obr. 3.6: Neupravená rýchlosť premávky.

⁶Ako referenčný model k TCP/IP.

⁷Petr Lapukhov, Shaping vs. Policing At A Glance <http://blog.ine.com/2008/07/03/at-a-glance-the-difference-between-shaping-and-policing/>



Obr. 3.7: Upravená rýchlosť premávky vyhľadzovaním a obmedzovaním.

Pre hlbšie pochopenie správy dátového toku je nutné sa venovať algoritmom, ktoré sa využívajú pri tejto činnosti.

Token Bucket algoritmus

Na obrázku 3.8 je zobrazená jednoduchá schéma popisujúca algoritmus Token Bucket. Do nádoby veľkosti Committed Burst Size (CBS), ktorá slúži ako protiváha prichádzajúcim dátam, sú vkladané žetóny. Ak veľkosť (B) prichádzajúcich dát je väčšia ako je momentálny počet žetónov v nádobe (T_c), tak sa vykoná akcia. V tomto prípade sú typické dva prípady, zahodenie dát alebo priradenie dát do fronty. V prípade, keď nádoba obsahuje dostatočný počet žetónov, tak je vykonaná iná akcia a sú odoberané žetóny z nádoby. Pridávanie žetónov do nádoby je vykonávané ideálnym stavom 1/CIR⁸ (Committed Information Rate). Ak je nádoba so žetónmi plná, tak sa ďalšie žetóny do nádoby nepridávajú. [2]

Dual Token Bucket algoritmus

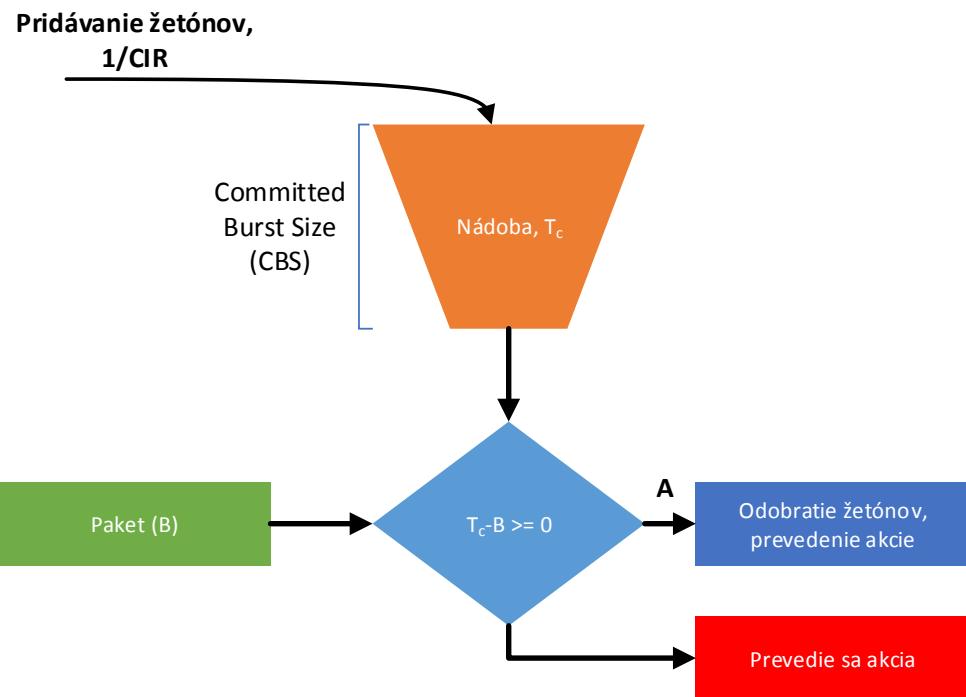
V prípade Dual Token Bucket (obr. 3.9) algoritmu existuje ďalšia nádoba o veľkosti Excess Burst Size (EBS). Rozdiel algoritmu ako v predchodom prípade je ten, že ak dáta sú väčšie ako aktuálny počet žetónov v nádobe o veľkosti CBS, tak dáta budú ešte porovnávané voči aktuálnej hodnote počtu žetónov (T_e) v nádobe o veľkosti EBS. Ďalšou zmenou voči predchádzajúcemu prípadu je plnenie žetónov do nádoby o veľkosti EBS, ktoré je vykonávané pretečenými žetónmi z nádoby o veľkosti CBS. [2] Rozšírením vyššie spomínaných algoritmov je možné vytvárať ďalšie nástroje na obmedzovanie a vyhľadzovanie dátovej prevádzky ako napríklad: Two-Rate Three-Color policers⁹.

3.4.3 Správa zahľtenia

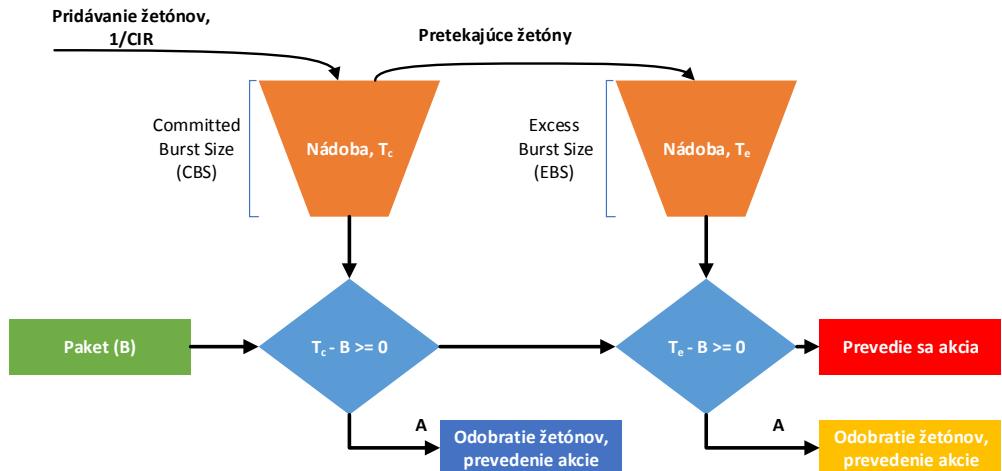
V prípade správy zahľtenia existuje množstvo mechanizmov na obsluhu. Táto sekcia je venovaná základným informáciám o obsluhe dát vo frontách. [2]

⁸Committed Information Rate, je parameter, ktorý udáva najnižšiu možnú garantovanú šírku pásma využitia sieťového spojenia.

⁹Petr Lapukhov, Understanding Single-Rate and Dual-Rate Traffic Policing <http://blog.ine.com/2011/05/22/understanding-single-rate-and-dual-rate-traffic-policing/>



Obr. 3.8: Schéma algoritmu Token Bucket.



Obr. 3.9: Schéma algoritmu Dual Token Bucket.

Férová fronta

Férová fronta (Fair Queueing) sa snaží deliť výstupnú šírku pásma medzi viaceré zdroje dát. Zdroje dát vytvárajú samostatné fronty, ktoré typicky pomocou algoritmu Round-Robin sú vyberané k možnosti prístupu spracovania dát.

Vážená férová fronta

Vážená férová fronta (Weighted Fair Queuing) rozširuje možnosti férovej fronty. V prípade váženej férovej fronty sú jednotlivým frontám pridelené váhy. Tieto váhy slúžia na pridelenie priorít jednotlivým frontám. Tieto priority zaručujú odbavenie viacerých dát v prípade vyšej priority fronty ako nižšej priority fronty.

Triedne založená fronta

Triedne založená fronta (Class-Based Weighted Fair Queuing) rozširuje váženú férovú frontu o možnosť definovania explicitnej dátovej triedy. Táto zmena umožňuje nastavenie parametrov triedy, ako je šírka pásma.

3.4.4 Zabránenie zahltenia

Pri zahltení front sietových zariadení je nutné, aby dané fronty nepríjmali ďalšie príchodzie dát. K tomu sa využívajú napríklad algoritmy RED (Random Early Detection) a WRED (Weighted Random Early Detection). Dôvodom na vznik takýchto algoritmov je ten, že jednoduché zamedzenie prístupu vstupných dát má za následok nežiaducé správanie¹⁰ niektorých dát.

RED

RED algoritmus využíva hodnotu priemernej dĺžky fronty zariadenia k zisteniu pravdepodobnosti zahodenia alebo zaradenia dát do fronty.

Ak je fronta skoro prázdna (nedosiahla úroveň začatia zahadzovania), tak všetky dáta budú prijaté. Ak fronta dosiahla určitú dĺžku, tak zväčšuje sa pravdepodobnosť zahadzovania. Pri určitej hodnote dĺžky fronty (fronta je skoro plná), tak pravdepodobnosť zahodenia dát nadobúda hodnotu 1.

WRED

WRED algoritmus rozširuje vyššie spomínaný algoritmus o to, že umožňuje na základe hodnôt poli v IP hlavičke meniť parametre pravdepodobnosti zahodenia takto označených dát.¹¹ [2]

3.4.5 Signalizácia

Signalizácia je využívaná tradične v Integrovaných službách pomocou protokolu RSVP. Táto časť je rozoberaná v sekcií 3.1.1.

3.4.6 Správa efektívneho využitia linky

Tieto mechanizmy sú typicky aplikované na relatívne pomalé linky. Medzi tieto mechanizmy patrí fragmentácia dát a kompresia niektorých hlavičiek dát.

Fragmentácia slúži na fragmentovanie dát z toho dôvodu, že dáta s veľkým MTU zvyšujú oneskorenie iných dát.

¹⁰Napríklad: TCP global synchronization.

¹¹Cisco Systems, Congestion Avoidance Overview http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcflem.html

Pre príklad slúži súbežne používanie interaktívnych protokolov s protokolmi slúžiacimi na prenos dát. Ak protokol na prenášanie dát používa dátu s veľkým MTU, tak zvyšuje oneskorenie interaktívnym protokolom. Tomuto javu sa snaží predísť fragmentácia, kde dátu s veľkým MTU sú fragmentované.

Kompresia hlavičiek sa snaží zmenšiť objem dát, ktoré sú prenášané zariadením.¹²

3.5 Zhodnotenie

Kvalita služby spadá pod veľkú oblasť v sieťových technológiách. V predchádzajúcich kapitolách boli spomenuté základné informácie ohľadom určitých architektúr kvality služieb a ich implementácií.

Integrované služby k svojmu účelu využívajú signalizačný protokol pomocou ktorého sú rezervované zdroje pre aplikácie. Tieto rezervácie obmedzujú škálovateľnosť využitia prostriedkov sieťových zariadení. Samotná zložitosť integrovaných služieb sa podpísala pod absenciu použitia v terajších dátových sietach.

Naproti tomu Diferencované služby využívajú informácie v IP hlavičkách a na základe ich hodnôt reagujú.

PfR kombinuje modifikáciu smerovacích údajov s PBR, pričom využíva informácie v IP hlavičkách podobne ako Diferencované služby.

¹²Cisco Systems, Link Efficiency Mechanisms Overview http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcflem.html

Kapitola 4

Gosia

Prototyp Gosia sa snaží uplatniť niektoré z vlastností vyššie spomenutých architektúr kvality služieb. Na základe predom definovaného správania zariadení, implementuje tieto poznatky z architektúr kvality služieb.

Z oblasti Integrovaných služieb prototyp využíva možnosť rezervácie zdrojov sieťových zariadení. Tieto rezervácie nie sú typické ako je to u Integrovaných služieb alebo v Diferencovaných službách.

Rezervácie využívajú informácie o stave prostriedkov zariadení a patrične na ne reagujú. Príkladom pre stav prostriedkov zariadení je zisťovanie rýchlosťi pretekajúcich dát rozhraním zariadenia.

Hlavnou vlastnosťou takéhoto riešenia je nevyhradenie striktných rezervácií zdrojov sieťových zariadení dátovým tokom. To má za následok zdieľanie zdrojov sieťového zariadenia medzi viaceré dátové toky a väčšiu škálovateľnosť.

Ďalšou z vlastností, ktorá bola využitá v prototype je dynamická modifikácia smerovacích údajov.

4.1 Požiadavky na prototyp

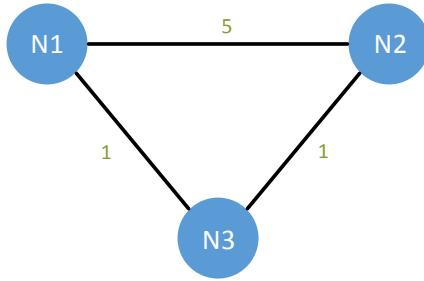
Ako bolo v predchádzajúcich kapitolách načrtnuté, základnou požiadavkou kvality služieb v dátových sietiach je kvalitatívne zabezpečenie úrovne chodu užívateľských dát. Keďže existuje mnoho druhov aplikácií, tým pádom existujú rôzne požiadavky na chod aplikačných dát, to umožňuje vytvárať určité prioritizovanie dát.

Medzi základné požiadavky prototypu Gosia patrí špecifické chovanie zariadení na základe hodnôt využitia zdrojov zariadení. Toto chovanie je definované ako dynamické smerovanie identifikovaných dát, ktoré je špecifikované v jednoduchom konfiguračnom jazyku.

Hlavným dôvodom pre vytvorenie takéhoto dynamického smerovania je to, že tradičné smerovacie protokoly neumožňujú prioritné špecifické smerovanie na základe identifikácie dát. Aj keď existujú možnosti ako docieliť tieto požiadavky, tie sú zvyčajne obtiažne dosiahnuteľné.

Príkladom pre tradičné smerovacie protokoly je vyhľadávanie najkratšej cesty. Na obrázku 4.1 je ilustrované chovanie tradičných smerovacích protokolov na cyklickom grafe. Jednotlivé hrany sú ohodnotené zelenou farbou.

Ak množina dát prichádza z uzla N1 do množiny uzla N2, tak smerovací protokol odošle dátu, hranou ohodnotenou 1, do uzla N3. Ďalej množinu dát z uzla N3 smeruje, hranou ohodnotenou 1, do uzla N2.



Obr. 4.1: Cyklický graf.

Takéto chovanie v tradičných smerovacích algoritmoch nie je stále vhodné. Dôvodov na zmenu chovania je viacero. Jednou z hlavných zmien v smerovacích protokoloch sú ekonomicke dôvody, jednostranné zaťažovanie infraštruktúr, a tým pádom nevyužitie potenciálu ostatných častí infraštruktúr.

Aby bolo možné využívať viaceré cesty v infraštruktúre je nutné túto množinu dát bližšie špecifikovať. Dôvodom pre tento krok je, že tradičné smerovacie protokoly pri smerovaní využívajú typicky iba cieľovú IP adresu. Smerovač na základe IP adresy, ktorá je porovávaná voči smerovacej tabuľke (RIB), zariadenia určí, na ktoré rozhranie dátu budú smerované. TCP/IP aplikácie využívajú sokety (sockets), ktoré sú identifikované ako päťica: {zdrojová IP adresa, cieľová IP adresa, zdrojový port, cieľový port, protokol}. Z tejto päťice je vidieť, že identifikácia iba na základe cieľovej IP adresy je nedostatočná.

Prototyp umožňuje pri predom špecifikovaných dátach využívať viaceré cesty. Ak existuje množina dát, ktoré spĺňajú kritériá pre definované smerovanie, tak jednotlivé zariadenia umožnia týmto dátam takto využívať cesty. Pre ilustráciu princípu opäť poslúži obrázok 4.1. Ak do uzla N1 prichádzajú dátu A a B, a tieto dátu spĺňajú potrebné kritériá pre definované smerovanie, tak jednotlivým dátam je umožnené využívať viaceré cesty. To znamená, že dátu A z uzla N1 do uzla N2 využívajú cestu cez uzol N3 a dátu B z uzla N1 do uzla N2 využijú hranu s ohodnotením 5. Ďalšou súčasťou je prioritizovanie určitého typu dát a na základe priority pri zahľtení jednotlivých zariadení na ceste, t.j. umožniť prioritizovanému dátovému toku si zachovať kvalitatívne podmienky.

Ako bolo vyššie spomenuté prototyp Gosia využíva jednoduchý konfiguračný jazyk. V tomto jazyku je možné definovať smerovanie identifikovaných dát a aplikovať rezervácie na rozhrania zariadení.

4.2 Architektúra prototypu

Základnými prvkami prototypu sú konfiguračný jazyk, riadiaca časť a hardvérové zariadenia. Obrázok 4.2 zobrazuje zjednodušenú architektúru prototypu. Prototyp využíva technológiu Cisco onePK, ktorá je súčasťou prostredia Cisco ONE.

Konfiguračný jazyk

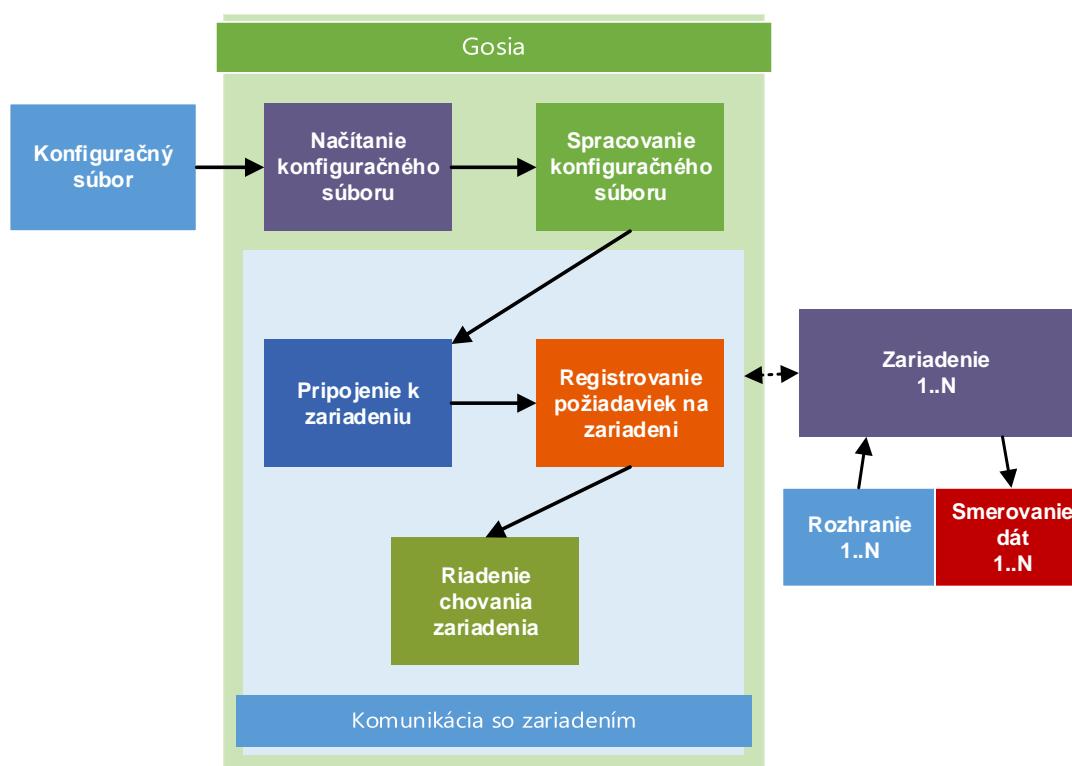
Konfiguračný jazyk popisuje nutné informácie ako sú: pripojenie aplikácií k zariadeniu, identifikácia dát, prioritizovanie identifikovaných dát. Ďalšou súčasťou jazyka je definovanie cest identifikovaných dát a rezervácia prostriedkov zariadenia.

Riadiaca časť

Riadiaca časť slúži na vyhodnocovanie informácií zo zariadenia a jeho manipuláciu.

Hardvérové zariadenia

Hardvérová časť obsahuje smerovače firmy Cisco Systems s podporou onePK verzie 1.1.0.



Obr. 4.2: Architektúra prototypu Gosia.

4.3 Návrh

Táto sekcia sa zaobrá návrhom riešenia prototypu Gosia. Sekcia je rozdelená na aplikačnú časť a popisom konfiguračného jazyka. Aplikačná časť rozoberá popis základnej časti aplikácie – smerovanie. Táto časť obsahuje úsek, kde je uvedený principiálny návrh smerovania. Konfiguračná časť obsahuje popis a ukážku konfiguračného jazyka pre manipuláciu chovania zariadení.

4.3.1 Aplikácia

Aplikačná časť k pripojeniu a manipulácií so zariadeniami využíva technológiu onePK od firmy Cisco Systems. V tejto sekcií sú priblížené základné vlastnosti aplikačnej časti.

Identifikácia dát

Táto časť slúži na identifikovanie dát. K identifikácii sú využívané ACL operačného systému Cisco IOS. Údaje, ktoré je možné vyžívať k identifikácii dát: zdrojová ip adresa, dĺžka prefixu zdrojovej sieťovej masky, cieľová ip adresa, dĺžka prefixu cieľovej sieťovej masky, hodnota DSCP poľa.

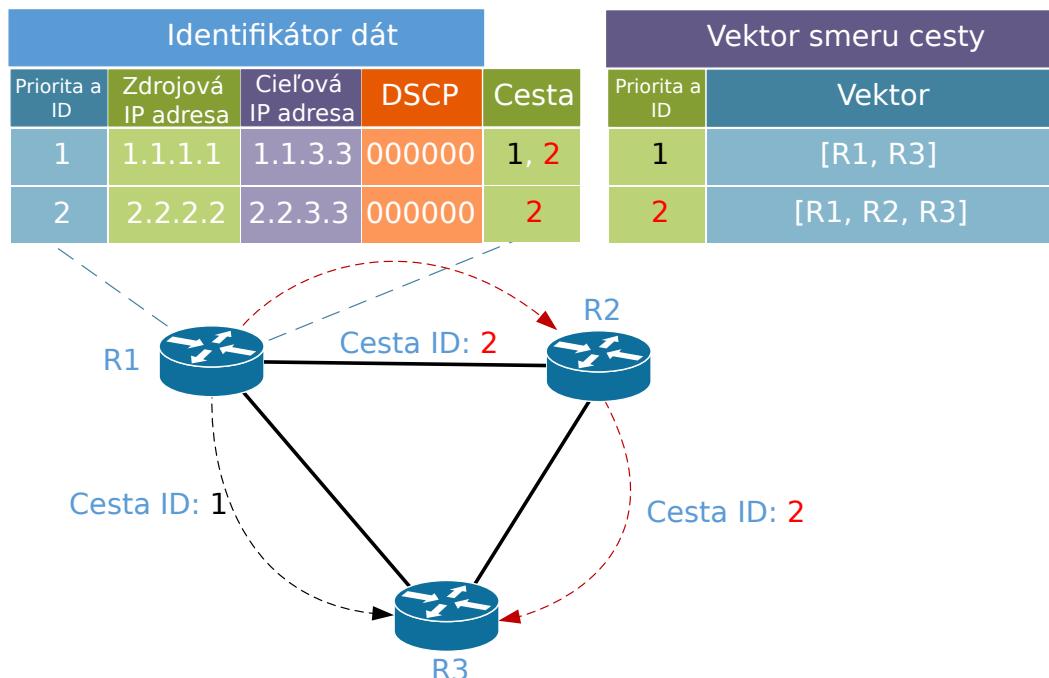
Registrovanie udalostí

Prototyp využíva registrovanie asynchronných udalostí, aby smerovacia časť na základe generovanej udalosti zariadenia mohla reagovať na vyskytnutú udalosť. Udalosti, ktoré je možné registrovať na jednotlivé zariadenia sú: maximálna rýchlosť pretekajúcich vstupných (RX) dát, minimálna rýchlosť pretekajúcich vstupných (RX) dát, maximálna rýchlosť pretekajúcich výstupných (TX) dát, minimálna rýchlosť pretekajúcich výstupných (TX) dát.

Vyššie spomenuté udalosti sú registrované na rozhraniach zariadenia. Tieto registrácie sú zadávané v jednotkách bit. Ďalšia udalosť, ktorá môže byť registrovaná na rozhraniach zariadenia je spoľahlivosť (reliability).

Smerovanie dát

Smerovanie dát využíva technológiu PBR, ktorá umožňuje zohľadniť smerovanie aj na základe iných atribútov ako je cieľová IP adresa. Princíp smerovania je zobrazený na obrázku 4.3.



Obr. 4.3: Princíp smerovania.

Dáta so zdrojovou IP adresou 1.1.1.1, cieľovou IP adresou 1.1.3.3 a hodnotou DSCP 000000 sú identifikované ako 1. Ďalšie dáta v tabuľke Identifikátor dát sú identifikované ako 2.

vané hodnotou 2. Dáta identifikované ako 1 umožňujú využívať cesty 1 a 2 a dáta identifikované ako 2 iba cestu 2.

Tieto cesty sú zobrazené v tabuľke Vektor smeru cesty. Jednotlivé dátá majú zdroj zo smerovača R1 a cieľ do smerovača R3.

Pri inicializácii ciest sú jednotlivým smerovačom inštalované cesty na základe tabuľky Vektor smeru cesty a vytvára sa globálna smerovacia tabuľka. Na základe druhu udalosti zo zariadenia prototyp reaguje. Ak rozhrania na ceste 1 vygenerujú udalosť prekročenia rýchlosťi, tak smerovanie sa snaží pre cestu s najmenšou prioritou nájsť inú voľnú cestu. V tomto prípade dátá s identifikátorom 1 sú jedine definované na tejto ceste. Keďže tieto dátá majú definovanú aj inú cestu, tak smerovacia časť zmení smerovacie údaje na smerovačoch, a pridá pre dátá s identifikátorom 1 záznam o smerovaní, a modifikuje globálnu smerovaciu tabuľku pre cestu 2. V prípade opäťovného vygenerovania udalosti prekročenia rýchlosťi, ale pre cestu 2, tak smerovanie vymaže cestu 2 pre dátá s identifikátorom 2 zo zariadení, a vyznačí cestu v globálnej smerovacej tabuľke. Ak rýchlosť na rozhraní padne na úroveň, minimálna rýchlosť pre cestu z rozhraní zariadení, tak smerovanie vyhľadáva v smerovacej tabuľke dátá, ktoré sa nevyužívajú pre opäťovné začlenenie do smerovania.

Prototyp neumožňuje dátam využívať súbežne viaceré cesty. To je spôsobené tým, že technológie, ktoré boli využívané neumožňovali využitie takýchto prostriedkov.

4.3.2 Konfiguračný jazyk

Konfiguračný jazyk využíva serializačný formát JSON. Tento konfiguračný jazyk je rozdelený do viacerých častí. V nasledujúcich častiach sú zobrazené príklady konfiguračného jazyka v serializačnom formáte JSON. Keďže JSON neumožňuje komentovanie kódu, tak príklady využívajú pomocný znak #.

Autentifikačná časť

Autentifikačná časť slúži k defininovaniu nutných informácií k pripojeniu zariadenia. V nasledujúcej časti je zobrazený príklad autentifikačnej časti konfiguračného jazyka.

```
"devices": {
    "router": {
        "1": { #interny identifikator zariadenia
            "ipv4": "1.2.3.4", #IPv4 adresa zariadenia
            "username": "admin", #uzivatelske meno
            "password": "nbusr123", #uzivatelske heslo
            "transport_type": "TLS", #Typ pripojenia
            "root_cert": "/home/cisco/ca.pem"
            #cesta k pem certifikatu pre TLS
        }
    }
}
```

Topologická časť

Topologická časť slúži k defininovaniu topológie sietových zariadení. Príklad demonštruje topologickú časť konfiguračného jazyka.

```
"1": { #zdrojovy identifikator
        #definovany z autentifikacnej casti
    "connected_to": {
        "2": { #cielovy identifikator
                #definovany z autentifikacnej casti
            "via": {
                "interface": ["GigabitEthernet0/1"]
                #rozhrania zariadenia
                #zdrojoveho idetifikatora
            }
        }
    }
}
```

Identifikačná časť

Táto časť umožňuje identifikovať dátá na základe ktorých prototyp dáva pokyny zariadeniam o smerovaní dát. Príklad znázorňuje nasledujúci ACE Cisco IOS ACL:

```
1 permit tcp host 5.6.7.8 host 9.10.11.12 dscp default
```

```
"access_list": {
    "1": { #interny identifikator ACL a priorita
        "1": { #onePK ACE identifikator
            "access": "permit",
            "src": "5.6.7.8",
            "src_prefix": "32",
            "dst": "9.10.11.12",
            "dst_prefix": "32",
            "proto": "tcp",
            "dscp": "000000"
            #DSCP hodnota v 10 zaklade
        }
        "router": ["1"]
        #zariadenia, kde ACL bude aplikovane
    }
}
```

Smerovacia časť

Smerovacia časť umožňuje definovať prioritné cesty pre definované ACL. V príklade sú definované dve cesty 1 a 2, kde nižšia hodnota určuje vyššiu prioritu cesty.

```
"gosia_routing": {
    "1": {#interna identifikacia ACL
        "1": {#identifikacia cesty a priorita
            {
                "via": ["1", "3"]
```

```

        #definicia smerovania
        #cez definovane zariadenia
        #zo autentifikacnej
        #sekcie
    },
"2": {
    {
        "via": ["1", "2", "3"]
    }
}

```

Rezervačná časť

V rezervačnej časti sa definujú hodnoty na rozhraniach zariadení. Ak rozhrania zariadení túto hodnotu dosiahnu, tak aplikáčná časť reaguje na túto udalosť.

```

"gosia_interface": {
    "router": {
        "1": {#identifikacia zariadenia
              #zo autentifikacnej casti
              "GigabitEthernet0/1": {
                  #nazov rozhrania
                  "tx_speed_per_s_max": "50000000",
                  #definovana rychlosť v bitoch za sekundu
                  "rx_speed_per_s_max": "50000000",
                  "tx_speed_per_s_min": "30000000",
                  "rx_speed_per_s_min": "30000000",
                  "reliability": "50"
                  #spolahlivosť rozhrania
              }
        }
    }
}

```

4.4 Implementácia

Prototyp je implementovaný v programovacom jazyku Python verzie 2.7.x. Na komunikáciu medzi zariadeniami a kontrolérom je využívané Cisco onePK python API verzie 1.1.0, Controlled Availability. Dokumentácie použitej verzii onePK SDK často nereflektovali implementovanú skutočnosť na strane zariadení. Táto absencia dokumentácie sa podpísala pod neefektívny proces implementácie prototypu. Riadiaca časť aplikácie sa nachádza na externom kontroléri. V nasledujúcich častiach sú rozobraté nutné časti implementácie smerovania.

Prístupové zoznamy

K využitiu identifikácie dát v prototype Gosia sú využívané prístupové zoznamy, keďže API onePK 1.1.0 neumožňuje definovať vlastné meno prístupového zoznamu, tým je

nutné mapovať interné meno prístupového zoznamu s užívateľským menom prístupového zoznamu.

Smerovanie

Smerovanie dát využíva PBR v Cisco terminológií route-map. Použitá verzia onePK neumožňuje využívať PBR, tak implementácia smerovania je vytváraná pomocou VTY SS. Takéto využitie onePK zvyšuje odozvu celkového systému. Priemerná odozva systému, ktorá bola nameraná¹ pri využití VTY SS je 420 ms.

Udalosti

K registrovaniu udalostí (events) na zariadenia sú využívané časti konfiguračného súboru. Pri registrácii týchto udalostí sa definuje takzvaný poll interval. Tieto udalosti na zariadení sú registrované ako asynchronne. Tento interval sa využíva na definovanie času, za ktorý zariadenia skontrolujú počítadla. Čo sa týka rozhraní (interfaces), zariadenia na modifikáciu priemernej dĺžky počítania počítadla rozhrania môžu využívať príkaz:

```
load-interval <(30-300)s>
```

Prototyp je konzolového typu. Nutný vstup prototypu je validný konfiguračný súbor. Voliteľné nastavenie je poll interval (sekundy a milisekundy). Implicitná hodnota poll intervalu má prednastavenú hodnotu 50 ms.

¹Proces merania bol uskutočnený na jednom zariadení príkazom show clock. V závislosti od príkazov sa mení odozva systému zariadenia.

Kapitola 5

Experimenty

V tejto kapitole sú uvedené experimenty s prototypom. Prvá časť sa zaobrá výkonnostným porovnávaním smerovacieho protokolu EIGRP s PBR a prototypom Gosia. Výsledky sú značne ovplyvnené použitím VTY SS v neprospech prototypu. Ďalším prvkom, ktorý ovplyvňuje chovanie prototypu je nastavenie poll intervalu. Druhá časť experimentov poskytuje emuláciu reálnej komunikácie. V obidvoch experimentoch sú menené nastavenia jednotlivých hodnôt udalostí v konfiguračnom súbore. Prototyp v týchto experimentoch využíva implicitnú hodnotu poll intervalu.

Nasledujúce riadky vysvetľujú a upresňujú význam nastavení v experimentoch. Hodnoty príkazov `rx_speed_per_s_max`, `tx_speed_per_s_max` a `reliability` definujú zmeny smerovania t.j. nájst' inú voľnú cestu pre dátá s najmenšou prioritou alebo dané dátá budú označené, aby neboli smerované v globálnej smerovacej tabuľke.

Hodnoty príkazov `rx_speed_per_s_min` a `tx_speed_per_s_min` definujú zmeny smerovania tak, že nainštalujú nesmerované dátá z globálnej smerovacej tabuľky.

Zmenou hodnoty poll intervalu je možné modifikovať čas, za ktorý rozhranie skontroluje svoje počítadlá. Takáto vlastnosť môže spôsobiť zmene chovania smerovania. V tomto prípade, v závislosti od stavu reálnej rýchlosť pretekajúcich dát rozhraním. Inými slovami povedané, ak sa nastaví príliš veľký poll interval, tak krátkodobé nárazové dátá neovplyvnia smerovací proces.

Pre lepšie priblíženie riešenej problematiky je načrtnuté chovanie jednotlivých príkazov. Existuje topológia T, v ktorej jednotlivé smerovače S sú zapojené do kruhu. Pričom k dvom ľubovoľným smerovačom S je pripojený zdroj dát Z a príjemateľ dát P. Na každom rozhraní R smerovačov S je nakonfigurovaný poll interval T_p s udalosťami U_a a U_z . Udalosť U_a značí príkazy: `rx_speed_per_s_max`, `tx_speed_per_s_max`, `reliability` s hodnotou U_{ah} a udalosť U_z `rx_speed_per_s_min` a `tx_speed_per_s_min` s hodnotou U_{zh} . Ak Z generuje prioritné dátá $P_{d_{1..n}}$ o celkovej rýchlosťi Z_v a hodnota rozhrania R ľubovoľného smerovača S pri intervale čítania T_p definovanej udalosti, v intervale definovanom U_a alebo U_z , tak sa vykoná akcia. V prípade, že hodnota Z_v je väčšia ako interval definovaný príkazmi v U_{ah} , tak najmenej prioritné dátá P_{d_n} budú vymazané v smerovačoch S zo smerovania. V prípade, že existuje možnosť začlenenia do smerovania, tak budú dané prioritné dátá začlenené pri využití inej cesty rozhraniami smerovačov S. Ak sa jednotlivým dátam $P_{d_{min}}$ nepodarí začleniť do smerovania, tak budú ozačené ako X.

Ak hodnota Z_v je menšia, ako hodnota intervalu U_{zh} , tak prioritným dátam $P_{d_{max}}$ označenými ako X je vyhľadávaná cesta, a následne sú opäť začlenené do smerovania v topológii T.

Pre kontinuálne opakovanie experimentov s danými softvérovým vybavením je nutné

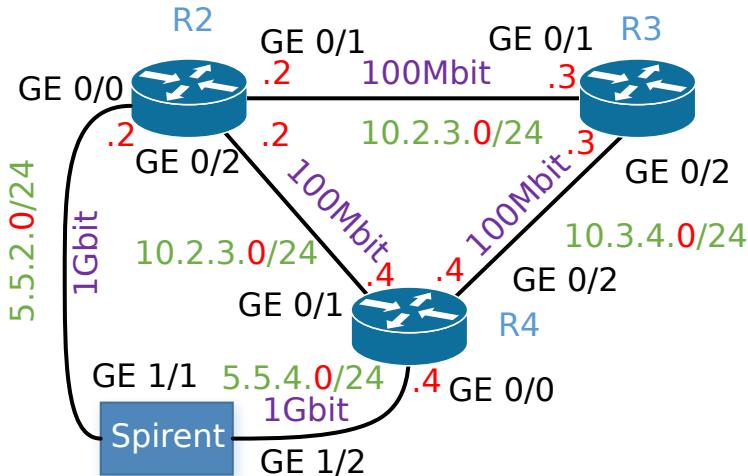
reštartovať smerovače ISR G2 2911. Tabuľka 5.1 zobrazuje hardvérové a softvérové vybavenie, na ktorom boli vykonávané experimenty. V sekcii A.1 je uvedená konfigurácia zariadení pre prácu s onePK. V prílohe B sú zobrazené topológie a kompletný príklad konfigurácie konfiguračného súboru pre funkciu prototypu.

Požiadavky	Popis
Prepínače	1 x Cisco Catalyst 2960
Smerovače	3 x Cisco ISR G2 2911, 1 x Cisco ISR G1 2811
Operačný systém ISR G2 2911	Cisco IOS Software, C2900 Software (C2900-UNIVERSALK9-M), Experimental Version 15.4(20131213:031344) [surf-onep_ca_pi23_throttle-nightly 144]
Kontrolér	Notebook, Intel Core i3 2 GHz, 8 GiB RAM
SDK kontroléra	Cisco onePK 1.1.0 python SDK, Controlled Availability
Emulácia štandardného sietového chovania	4 x PC minimálne so 100 Mbit Ethernet NIC
Hardvérový generátor	Spirent SPT-2000A

Tabuľka 5.1: Použité hardvérové a softvérové prostriedky k experimentom.

5.1 Porovnanie EIGRP, PBR a Gosia

Táto sekcia porovnáva výkonnostné parametre smerovania medzi EIGRP, PBR a prototypom Gosia. In-band topológia merania je zobrazená na obrázku 5.1. K experimentom bol primárne využívaný hardvérový generátor dát Spirent SPT-2000A.



Obr. 5.1: Inband meracia topológia.

Smerovací protokol EIGRP bol nakonfigurovaný tak, aby umožňoval využiť viaceré vektory smeru cesty k cieľovej destinácii, takzvaný load balancing. Táto konfigurácia bola docielená na zariadení¹ príkazom:

```
variance 2
```

Keďže implicitné správanie EIGRP pri topológií uvedenej v obrázku 5.1 nevyužíva load balancing do destinácie 5.5.4.0/24, je nutné modifikovať správanie smerovacieho protokolu. Toto správanie bolo modifikované na rozhraní² príkazom:

```
delay 5
```

Aby bolo možné vhodne využiť load balancing, tak v tomto prípade sa využíva load balancing typu per packet. Správanie zariadení³ na load balancing per packet bolo dosiahnuté príkazom na rozhraní:

```
ip load-sharing per-packet
```

k overeniu správania load balancing-u zariadení bol využitý príkaz:

```
show ip cef <route> internal
```

5.1.1 Výkonnostné porovnanie EIGRP s PBR

Popis experimentu

Cieľom experimentu bolo ukázať, že rozdiel medzi výkonom tradičného smerovania voči PBR je zanedbateľný. V tomto experimente sú demonštrované výkonnostné schopnosti smerovania zariadenia (ISR G2 2911). Dôvod pre takýto experiment je ten, že PBR vyžaduje väčšiu výpočtovú kapacitu ako tradičné smerovanie pomocou smerovacieho protokolu EIGRP. V tomto experimente bola využívaná záťaž 250 Mbit/s po dobu 10 sekúnd. Pri generovaní dát boli vyžívané dva toky, ktoré ekvivalentne zaťažovali zariadenia v topológii. Zdroj zaťaženia bol port GE 1/1 na hardvérovom generátore Spirent SPT-2000A. Zariadenia (ISR G2 2911) boli nakonfigurované tak, aby umožnili generovaným tokom využívať viaceré vektory smeru. Experimenty spočívali v generovaní rámcov rozličných veľkostí a meraní ich stratovosti.

Výsledky experimentu

V tabuľke 5.2 sú zobrazené výsledky výkonnostného experimentu PBR a tradičného smerovania.

Veľkosť rámcov (B)	EIGRP		PBR	
	Prijaté rámce	Strata (%)	Prijaté rámce	Strata (%)
1024	239799	19.8	239799	19.8
1280	192643	19.8	192643	19.8
1518	162884	19.8	162884	19.8

Tabuľka 5.2: Výsledky výkonnostného merania smerovacieho protokolu EIGRP s PBR.

¹Smerovač R2.

²Rozhraní, ktoré priamo spojujú smerovače R3 a R4.

³Smerovač R2.

Zhodnotenie experimentu

Z výsledkov experimentu je možné vyčítať, že stratovosť dát pri použití týchto technológií je ekvivalentná. To je spôsobené tým, že záťaž 250 Mbit/s nie je dostatočné veľká, aby mala vplyv na smerovanie zariadení technológiou PBR. Preto je možné usúdiť, že záťaž 250 Mbit/s nemá vplyv na výsledky ďalších experimentov.

5.1.2 Gosia poll interval #1

Popis experimentu

Cieľom experimentu je meranie stratovosti dát pri rozličnej veľkosti rámcov a to na základe hodnoty poll intervalu a hodnôt udalostí.

Rozhrania na zariadeniach boli nastavené príkazom:

```
load-interval 30
```

V konfiguračnom súbore udalosti boli nastavené na každom⁴ rozhraní na hodnoty:

```
"tx_speed_per_s_max": "35000000",
"rx_speed_per_s_max": "35000000",
"tx_speed_per_s_min": "10000000",
"rx_speed_per_s_min": "10000000",
"reliability" : "50"
```

V tomto experimente boli generované 3 toky o spoločnej veľkosti zaťaženia 250 Mbit/s, po dobu 10 sekúnd.

Výsledky experimentu

V tabuľke 5.3 sú zobrazené výsledky porovnania stratovosti dát na základe poll intervalu.

Veľkosť rámcov (B)	Poll interval		
	50 ms	500 ms	
1024	119900	59.9	119901
1280	156725	34.8	176450
1518	149170	26.5	149171

Tabuľka 5.3: Výsledky výkonostného merania prototypu Gosia na základe zmeny poll intervalu.

Zhodnotenie experimentu

Na základe výsledkov experimentu je možné usúdiť, že zmena prijatých dát zmenou poll intervalu pri takomto nastavení udalosti je minimálna. Predpokladom zmeny pri výchylke 500 ms je fakt, že zariadenia nestihli dostatočne rýchlo reagovať vystavenému zaťaženiu. Jeden z hlavných dôvodov je ten, že použitím VTY SS sa rapídne zvyšuje odozva celého systému. Ďalším dôvodom je skutočnosť, že počítadla na rozhraniach nereflektujú okamžitý status rýchlosťi dát prechádzajúcich cez rozhranie.

⁴Okrem priamo pripojených rozhraní k hardvérovému generátoru Spirent.

5.1.3 Gosia poll interval #2

Popis experimentu

Tento experiment demonštruje chovanie smerovania pri zmene nastavenia udalosti. V experimente sú porovnávané hodnoty stratovosti rámcov na základe nastavenia poll intervalu. V porovnaní s predchádzajúcim prípadom boli použité iné parametre nastavenia udalosti rozhraní.

V konfiguračnom súbore boli udalosti nastavené na káždom⁵ rozhraní v nasledujúcich hodnotách:

```
"tx_speed_per_s_max": "30000000",
"rx_speed_per_s_max": "30000000",
"tx_speed_per_s_min": "20000000",
"rx_speed_per_s_min": "20000000",
"reliability" : "50"
```

Výsledky experimentu

V tabuľke 5.4 sú zobrazené výsledky porovnania stratovosti dát na základe poll intervalu.

Veľkosť rámcov (B)	Poll interval			
	50 ms		500 ms	
1024	Prijaté rámce	Strata (%)	Prijaté rámce	Strata (%)
1024	116626	61.0	119901	59.9
1280	29698	87.6	96322	59.9
1518	81442	59.9	135475	33.3

Veľkosť rámcov (B)	Poll interval	
	123 ms	
1024	Prijaté rámce	Strata (%)
1024	219677	26.6
1280	96322	59.9
1518	139668	31.2

Tabuľka 5.4: Výsledky výkonostného merania prototypu Gosia na základe zmeny poll intervalu.

Zhodnotenie experimentu

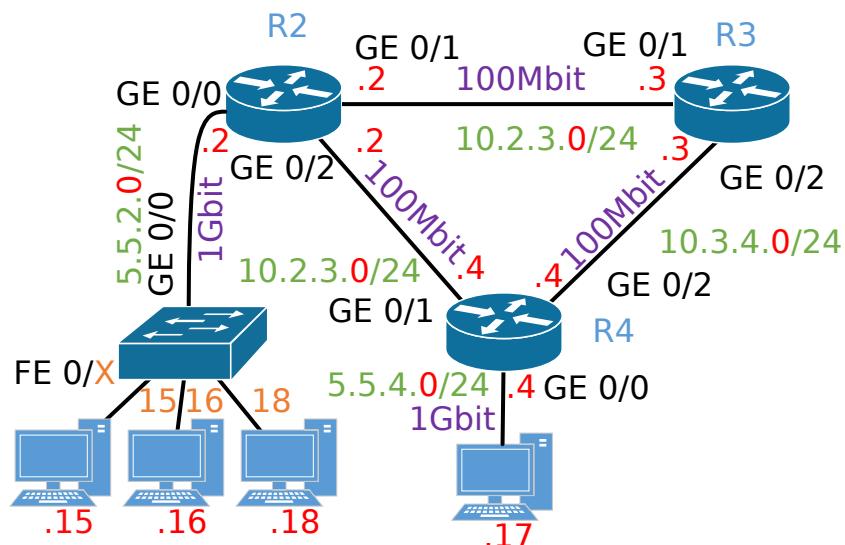
Na výsledku predchádzajúceho, ale aj tohto experimentu má najväčší podiel použitie VTY SS. Z týchto výsledkov je možné usúdiť, že pri nastavení poll intervalu na hodnotu 123 ms vzniká najnižšia stratovosť dát. Je nutné si uvedomiť, že stratovosť je v korelácii od nastavení prototypu a chovania jednotlivých dát.

⁵Okrem priamo pripojených rozhraní k hardvérovému generátoru Spirent.

5.2 Emulácia reálnej komunikácie

Experimenty v tejto sekcii sa snažia emulovať reálnu prevádzku. Dôvodom na takéto meranie je realita, že v predchádzajúcich experimentoch hardvérový generátor nezohľadňoval strátovosť dát pri generovaní ďalších dát. Na obrázku 5.2 je zobrazená topológia, na ktorej boli vykonávané experimenty. Komunikácia existovala medzi hostami 5.5.2.X a 5.5.4.17. Prenosová rýchlosť celkovej komunikácií bola obmedzená na hodnotu 250 Mbit/s. Takáto limitácia bola vykonávaná na prepínači Cisco Catalyst 2960. Prototyp Gosia využíval poll time, ktorý bol nakonfigurovaný na hodnotu 50 ms. Rozhrania na zariadeniach boli nastavené príkazom:

```
load-interval 30
```



Obr. 5.2: Inband meracia topológia.

Jednotlivé informácie o tokoch boli merané na stanici s IP adresou 5.5.4.17. Na niektorých závislostiach sú zobrazené vertikálne čiary (S_x), ktoré reprezentujú niektoré straty spôsobené zmenou smerovania.

5.2.1 Gosia #1

Popis experimentu

Experiment overuje chovanie prioritizovania dát pri využití viacerých priroritizovaných vektorov smeru. Pri inicializácii prototypu Gosia boli všetkým dátovým tokom nastavené rovnaké vektory smeru. V experimente sú porovnávané chovania jednotlivých tokov na základe veľkosti TCP okna. V konfiguračnom súbore boli udalosti nastavené na každom⁶ rozhraní na hodnoty:

```
"tx_speed_per_s_max": "50000000",
"rx_speed_per_s_max": "50000000",
"tx_speed_per_s_min": "45000000",
"rx_speed_per_s_min": "45000000",
"reliability": "50"
```

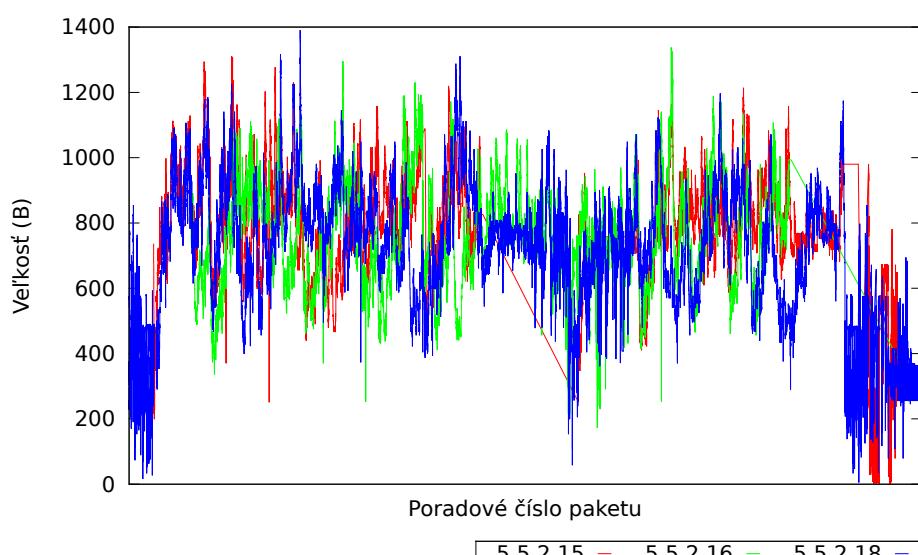
V experimente boli generované 3 toky o spoločnej maximálnej veľkosti zaťaženia 250 Mbit/s. Toky boli generované pomocou protokolu SCP. Každému z nich bolo umožnené využívať všetky vektory smeru. Najvyššiu prioritu mal tok so zdrojovou IP adresou 5.5.2.18, a najnižšiu prioritu tok so zdrojovou IP adresou 5.5.2.15.

Výsledky experimentu

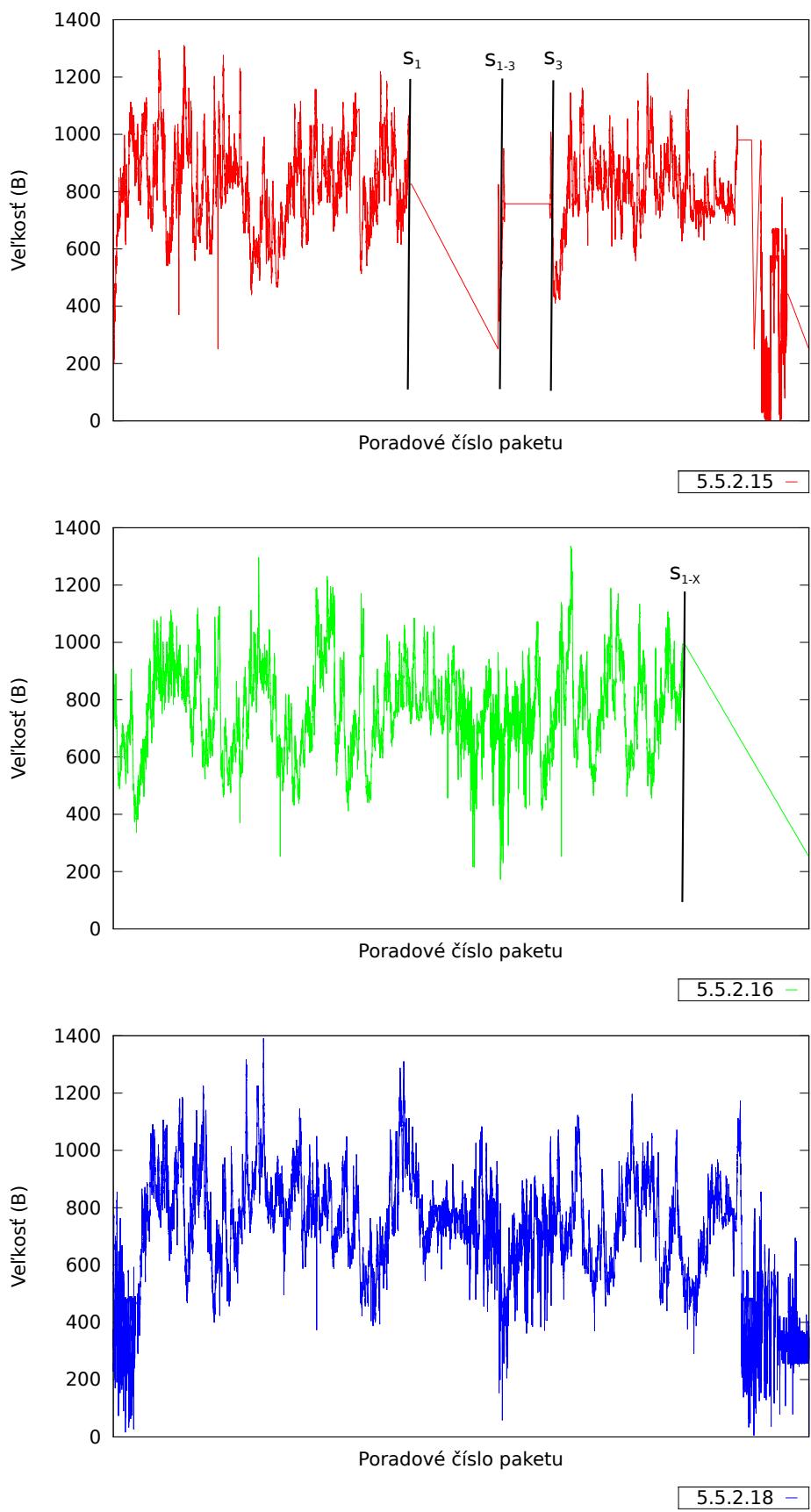
V závislosti 5.3 sú zobrazené výsledky veľkosti TCP okna a tieto výsledky sú vytvárané pomocou programu Wireshark s použitým filtrom:

```
ip.src == 5.5.4.17 and ip.dst == 5.5.2.X and not tcp.analysis.flags
```

Kde X je číslo rozhrania na prepínači v uvedenej topológii.



⁶Okrem priamo pripojených rozhraní k hostom.



Obr. 5.3: Využitie všetkých ciest, pre všetky toky.

Zhodnotenie experimentu

Zo závislosti je možné určiť stavy, kde jednotlivé toky pri využití rovnakých vektorov smeru medzi sebou súperia. Najviac prioritnému toku bolo umožnené udržiavať kvalitatívnu úroveň na úkor menej prioritných tokov.

5.2.2 Gosia #2

Popis experimentu

Experiment sa venuje pozorovaniu správania menej prioritných tokov voči viac prioritnému toku, pri neumožnení využitia prioritnému toku viacerých vektorov smeru cesty. V eksperimente sú porovnávané chovania jednotlivých tokov na základe veľkosti TCP okna. V konfiguračnom súbore boli udalosti nastavené na každom⁷ rozhraní na hodnotu:

```
"tx_speed_per_s_max": "60000000",
"rx_speed_per_s_max": "60000000",
"tx_speed_per_s_min": "45000000",
"rx_speed_per_s_min": "45000000",
"reliability": "50"
```

V eksperimente boli generované 3 toky o spoločnej maximálnej veľkosti zaťaženia 250 Mbit/s. Toky boli generované pomocou protokolu SCP. Toku so zdrojovou IP adresou 5.5.2.18 bolo umožnené využívať iba jeden vektor smeru. Ostatným tokom bolo umožnené využívať všetky vektory smeru. Pri inicializácii smerovania, každý tok využíval rovnaký vektor smeru cesty. Najvyššiu prioritu mal tok so zdrojovou IP adresou 5.5.2.18, a najnižšiu prioritu tok so zdrojovou IP adresou 5.5.2.15.

Výsledky experimentu

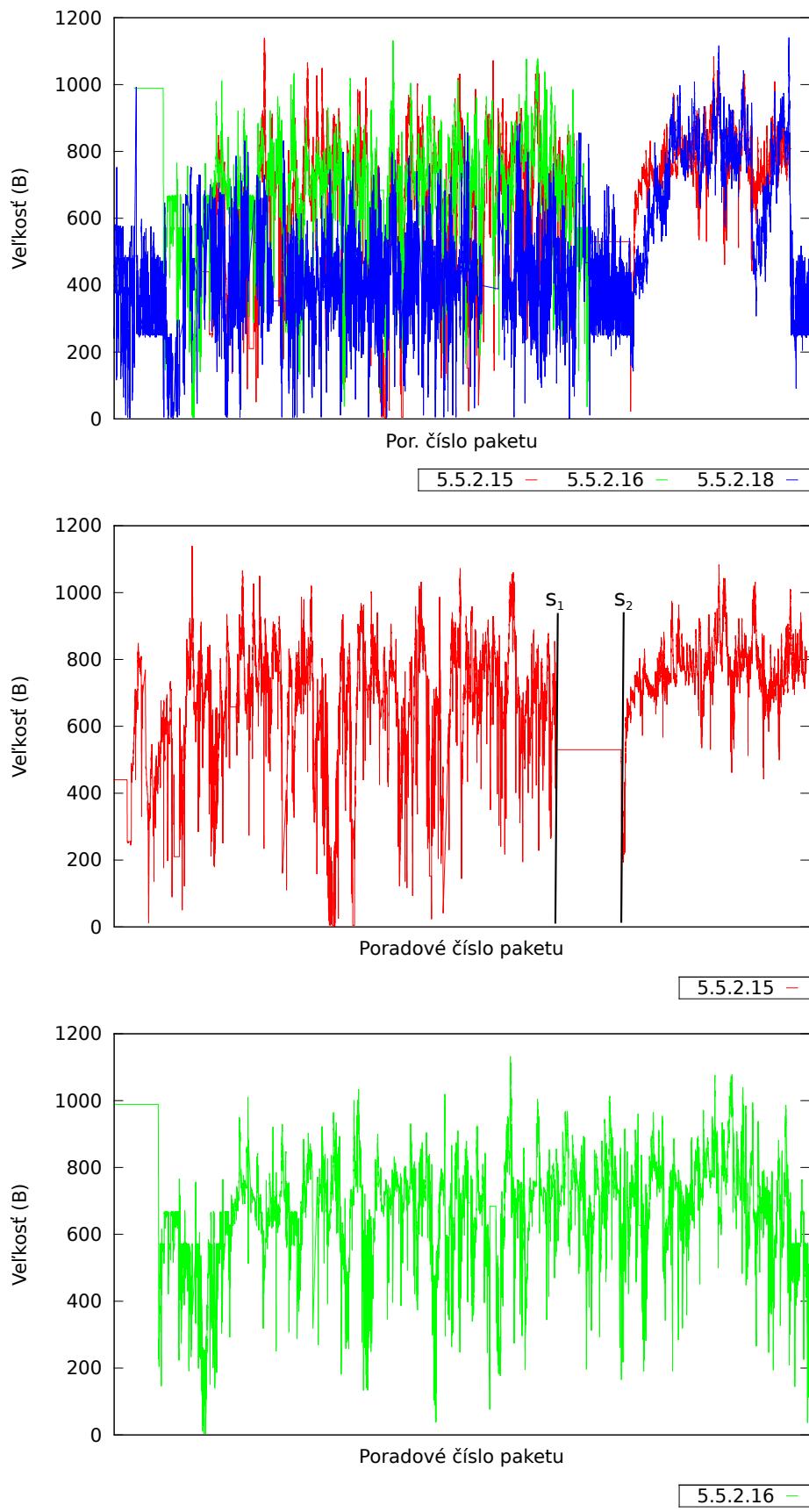
V závislosti 5.4 sú zobrazené výsledky veľkosti TCP okna a tieto výsledky sú vytvárané pomocou programu Wireshark s použitým filtrom:

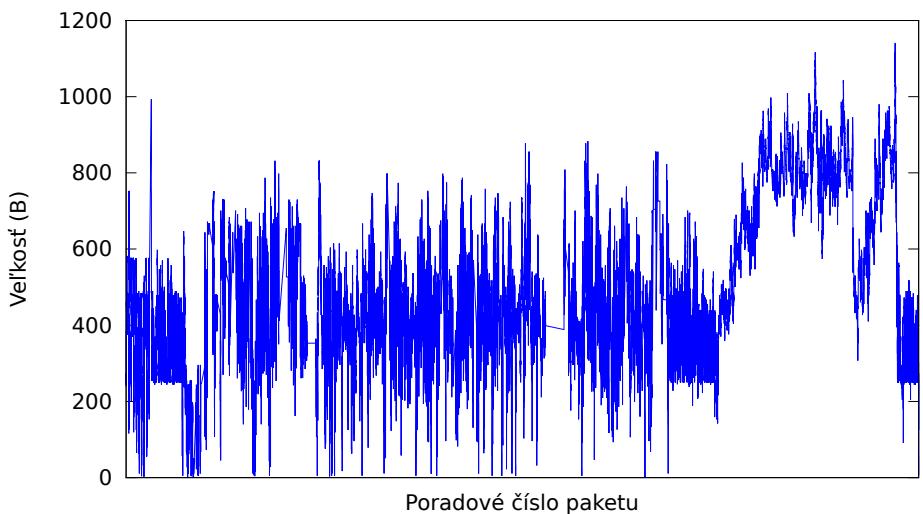
```
ip.src == 5.5.4.17 and ip.dst == 5.5.2.X and not tcp.analysis.flags
```

Kde X je číslo rozhrania na prepínači v uvedenej topológii.

⁷Okrem priamo pripojených rozhraní k hostom.

Obr. 5.4: Prioritný tok jedna cesta, menej prioritné toky všetky cesty.





5.5.2.18 —

Zhodnotenie experimentu

Najviac prioritnému toku je udržiavaný predom nastavený vektor smeru. Jednotlivé medzery v najviac prioritnom toku sú spôsobené pomalou reakciou zariadenia na prekročenia rýchlosť rozhrania zariadenia. V niektorých prípadoch menej prioritné toky využívali rovnaký vektor smeru, ako najviac prioritný tok a dané zariadenia nereagovali dostatočne presne na vzniknutú skutočnosť. Príkladmi pre skutočnosť je rýchlosť pretekajúcich dát na rozhrani, alebo nastavenie príliš malej hodnoty poll intervalu.

5.2.3 Gosia #3

Popis experimentu

Tretí experiment sa zaobera meraním správania jednotlivých tokov. Kde pri inicializácii najviac prioritný tok využíval iný vektor smeru cesty ako menej prioritné toky.

V eksperimente sú porovávané chovanie jednotlivých tokov na základe veľkosti TCP okna. V konfiguračnom súbore boli udalosti nastavené na každom⁸ rozhraní na hodnoty:

```
"tx_speed_per_s_max": "40000000",
"rx_speed_per_s_max": "40000000",
"tx_speed_per_s_min": "35000000",
"rx_speed_per_s_min": "35000000",
"reliability": "50"
```

V eksperimente boli generované 3 toky o spoločnej maximálnej veľkosti zaťaženia 250 Mbit/s. Toky boli generované pomocou protokolu SCP. Najvyššiu prioritu mal tok so zdrojovou IP adresou 5.5.2.18, a najnižšiu prioritu tok so zdrojovou IP adresou 5.5.2.15.

Výsledky experimentu

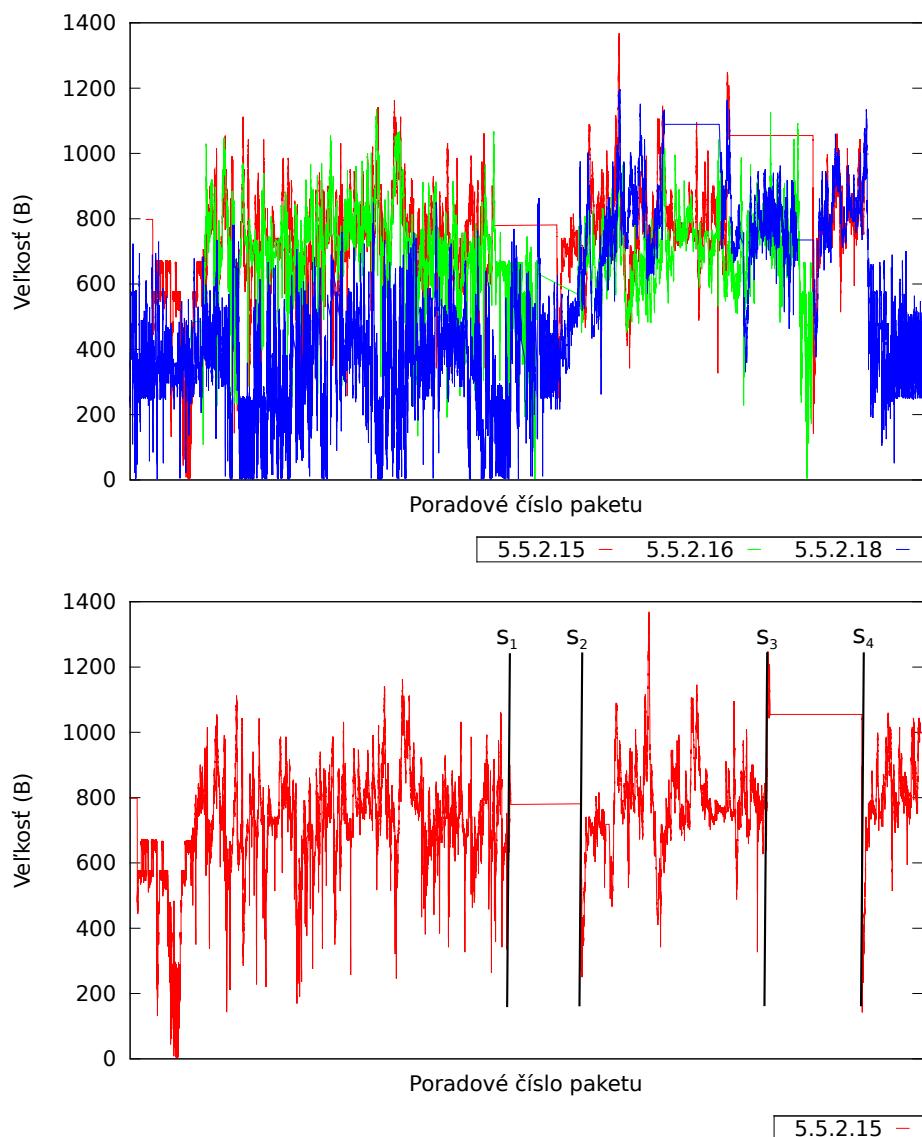
V závislosti 5.5 sú zobrazené výsledky veľkosti TCP okna na čase a tieto výsledky sú vytvárané pomocou programu Wireshark s použitým filtrom:

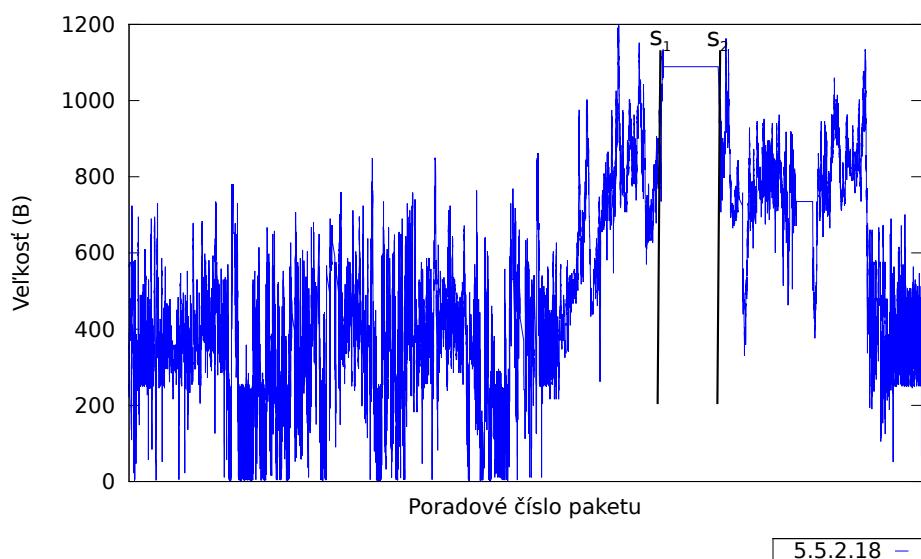
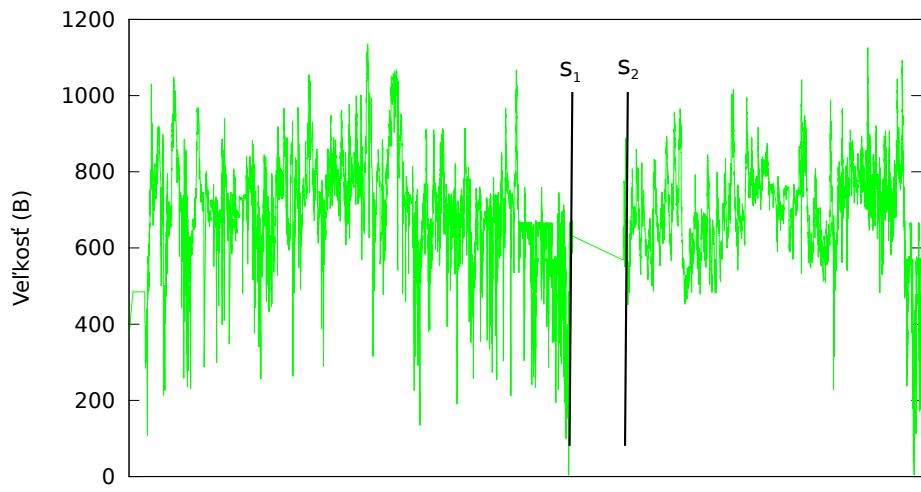
```
ip.src == 5.5.4.17 and ip.dst == 5.5.2.X and not tcp.analysis.flags
```

⁸Okrem priamo pripojených rozhraní k hostom.

Kde X je číslo rozhrania prepínača na uvedenej topológii.

Obr. 5.5: Inicializácia ciest tokov odlišná, prioritný tok jedna cesta, menej prioritné toky všetky cesty.





Zhodnotenie experimentu

Vo výsledkoch experimentu sú zobrazené hodnoty, na ktorých je vidieť, že najmenej prioritný tok dostáva najmenej priestoru na komunikáciu. Ako v predchádzajúcich experimentoch, tak aj v týchto, pri prekročení definovaných hodnôt má najvačší podiel VTY SS.

5.2.4 Gosia #4

Popis experimentu

Cieľom experimentu bolo meranie chovania tokov pri využití jedného vektora smeru cesty. Tok so zdrojovou IP adresou 5.5.2.16 bol neskôr generovaný ako ostatné toky. V experimente sú porovnávané chovania jednotlivých tokov na základe veľkosti TCP okna. V konfiguračnom súbore boli udalosti nastavené na každom⁹ rozhraní na hodnoty:

⁹Okrem priamo pripojených rozhraní k hostom.

```

"tx_speed_per_s_max": "50000000",
"rx_speed_per_s_max": "50000000",
"tx_speed_per_s_min": "30000000",
"rx_speed_per_s_min": "30000000",
"reliability": "50"

```

V tomto experimente boli generované 3 toky o spoločnej maximálnej veľkosti zaťaženia 250 Mbit/s. Toky boli generované pomocou protokolu SCP. Všetkým tokom bolo umožnené využívať iba jeden (rovnaký) vektor smeru. Jednotlivé toky boli generované s väčším one-skorením ako v predchodzích prípadoch. Najvyššiu prioritu mal tok so zdrojovou IP adresou 5.5.2.18, a najnižšiu prioritu tok so zdrojovou IP adresou 5.5.2.15.

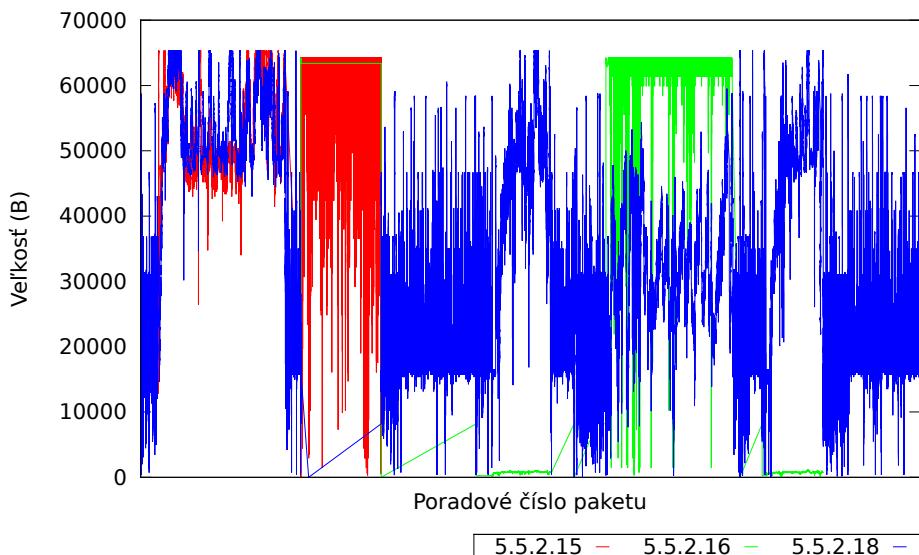
Výsledky experimentu

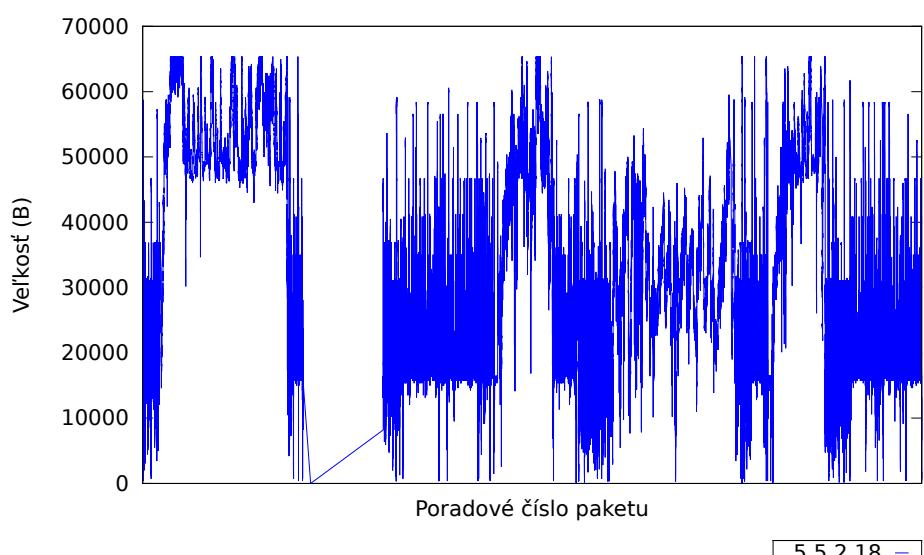
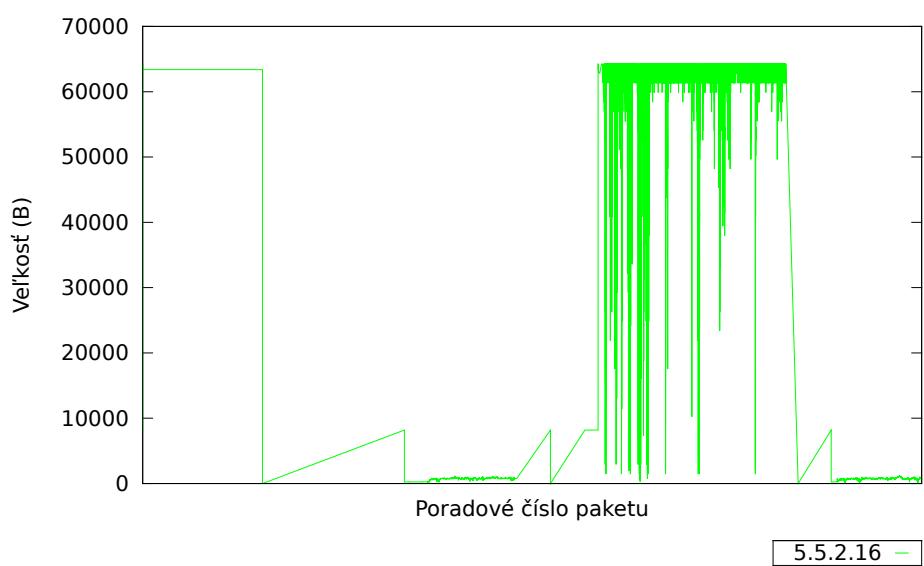
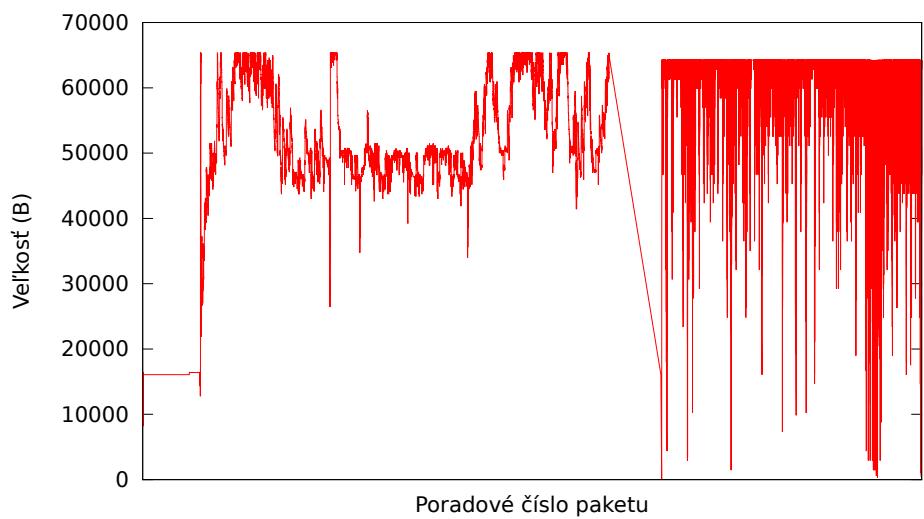
V závislosti 5.6 sú zobrazené výsledky veľkosti TCP okna, tieto výsledky sú vytvárané pomocou programu Wireshark s použitým filtrom:

```
ip.src == 5.5.4.17 and ip.dst == 5.5.2.X and not tcp.analysis.flags
```

Kde X je číslo rozhrania prepínača na uvedenej topológii.

Obr. 5.6: Všetky toky využívajú jednu cestu.





Zhodnotenie experimentu

Z výsledkov experimentu je možné usúdiť, že tok so zdrojovou IP adresou 5.5.2.15 v prvej tretine závislosti predbehol prioritný tok 5.5.2.18. Za túto skutočnosť sú čiastočné zodpovedné udalosti s daným nastavením hodnoty poll intervalu.

5.3 Zhodnotenie

Prvá časť tejto kapitoly je venovaná experimentom na harvérovom generátore. Experimenty v tejto časti demonštrujú výkonnostné zmeny pri použití iného poll intervalu a hodnôt udalostí.

V druhej časti experimenty boli hlavne sústredené na meranie veľkosti TCP okna. Dôvodom na využitie TCP okna je fakt, že rýchlosť zmeny veľkosti TCP okna typicky znamená stratu dát. V tomto prípade inštalovaním novej cesty, alebo zahadzovaním dát. Male výchylky veľkosti TCP okna tokov sú spôsobené implicitným chovaním zariadenia.

Prototyp využíva k svojej činnosti smerovania VTY SS a práve táto vlastnosť zvyšuje celkovú odozvu systému. Odozva je príčinou väčších medzier komunikácie jednotlivých tokov.

Chovanie jednotlivého správania tokov je aj vo veľkej miere ovplyvňované samotnou konfiguráciou ciest a udalostí. Nastavenie hodnôt udalostí a dĺžku poll intervalu nie je jednoduché určiť. Nevhodné nastavenie týchto hodnôt má za následok nevyužitie potenciálu infraštruktúry a tieto nastavenia sú závislé od typu prenašaných dát.

Kapitola 6

Záver

V bakalárskej práci som sa venoval návrhu a implementácii prototypu pre dynamickú zmenu smerovania dát pomocou technológie Cisco onePK. Jednotlivé chovanie dátových tokov sú vytvárané jednoduchým konfiguračným jazykom.

Vyššie zmienený prototyp bol implementovaný na experimentálnych verziach nástrojov, ktoré v dobe tvorby neumožňovali efektívny implementačný proces a efektívne využitie prostriedkov zariadení.

Výsledkom práce je prototyp Gosia, ktorý umožňuje definovaným prioritným dátam využívať viaceré cesty do destinácií. Tieto cesty sú dynamicky vyberané na základe stavu infraštruktúry.

Chovanie prototypu bolo demonštrované experimentmi, tie boli prevažne sústredené na meranie veľkosti TCP okna. Výsledky experimentov dokazujú, že využívanie jednej cesty pri takomto dynamickom smerovaní rapídne zvyšuje stratovosť dát. Túto stratovosť je možné upraviť vhodným nastavením prototypu.

Prototyp Gosia umožňuje pri návrhu a používaní dátových infraštruktúr zohľadňovať kvalitatívne požiadavky rôznorodých aplikácií.

Keďže technológie SDN sú v ranom štádiu vývoja, rozšírenia sú závislé na budúcom smerovaní týchto technológií. Zaujímavým rozšírením by bolo využitie súbežného posielania dát po nezávislých cestách pri využití smerovacieho protokolu, ktorý by zohľadňoval kvalitatívne požiadavky jednotlivých definovaných dát.

Literatúra

- [1] OpenFlow Switch Specification, Version 1.0.0 (Wire Protocol 0x01). 2009.
URL <http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf>
- [2] Cisco onePK C API Reference. 2013.
URL <https://developer.cisco.com/media/onePKCAPI/index.html>
- [3] Cisco onePK Technical Overview. 2013.
URL <https://developer.cisco.com/site/tech/networking/sdn/onepk-developer/overview/technical-overview/index.gsp>
- [4] onePK Design Guidelines Version 1.0.0, Controlled Availability Release. 2013.
URL <https://developer.cisco.com/media/onePKDesignGuidelines/GUID-1C3A8615-9BFB-4C2A-B227-CF27DA9AE93B.html>
- [5] OpenFlow Switch Specification, Version 1.4.0 (Wire Protocol 0x05). 2013.
URL <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>
- [6] PfR Technology Overview. 2013.
URL http://docwiki.cisco.com/wiki/PfR:Technology_Overview
- [7] Chao, H.; Liu, B.: *High Performance Switches and Routers*. Wiley, 2007, ISBN 978-0-4701-1394-3.
- [8] Cisco Systems: The Zettabyte Era - Trends and Analysis. 2013.
URL http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.html
- [9] HP: HP OpenFlow Protocol Overview Technical Solution Guide. 2013.
URL http://h17007.www1.hp.com/docs/networking/solutions/sdn/devcenter/03_-_HP_OpenFlow_Technical_Overview_TSG_v1_2013-10-01.pdf
- [10] Nadeau, T.: Interface to the Routing System (IRS). 2012.
URL <http://forums.juniper.net/t5/0ccupy-SDN-and-Programmability/Interface-to-the-Routing-System-IRS/ba-p/158704>
- [11] Nadeau, T.; Gray, K.: *SDN: Software Defined Networks*. O'Reilly Media, 2013, ISBN 978-1-4493-42230-2.
- [12] Nichols, K.; Blake, S.; Baker, F.; aj.: RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Technická zpráva, IETF, 1998.
URL www.ietf.org/rfc/rfc2474.txt

- [13] Open Networking Foundation: OpenFlow Management and Configuration Protocol (OF-Config 1.1.1). 2013.
URL <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1-1-1.pdf>
- [14] Schmid, S.; Suomela, J.: Exploiting Locality in Distributed SDN Control. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, New York, NY, USA: ACM, 2013, ISBN 978-1-4503-2178-5, s. 121–126, doi:10.1145/2491185.2491198.
URL <http://doi.acm.org/10.1145/2491185.2491198>
- [15] Wang, Z.: *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann series in networking, Morgan Kaufmann, 2001, ISBN 9781558606081.
- [16] Xiao, X.; Ni, L. M.: Internet QoS: A big Picture. *IEEE Network*, ročník 13, 1999: s. 8–18.
URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.712&rep=rep1&type=pdf>
- [17] Zhao, W.; Olshefski, D.; Schulzrinne, H.: Internet Quality of Service: an Overview. Technická zpráva, 2000.
URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.3060&rep=rep1&type=pdf>

Dodatok A

Konfigurácie

A.1 Konfigurácie zariadenia

V nasledujúcej časti je zobrazená nutná konfigurácia zariadenia s podporou onePK 1.1.0. Predpokladá sa použitie vygenerovaného certifikátu.

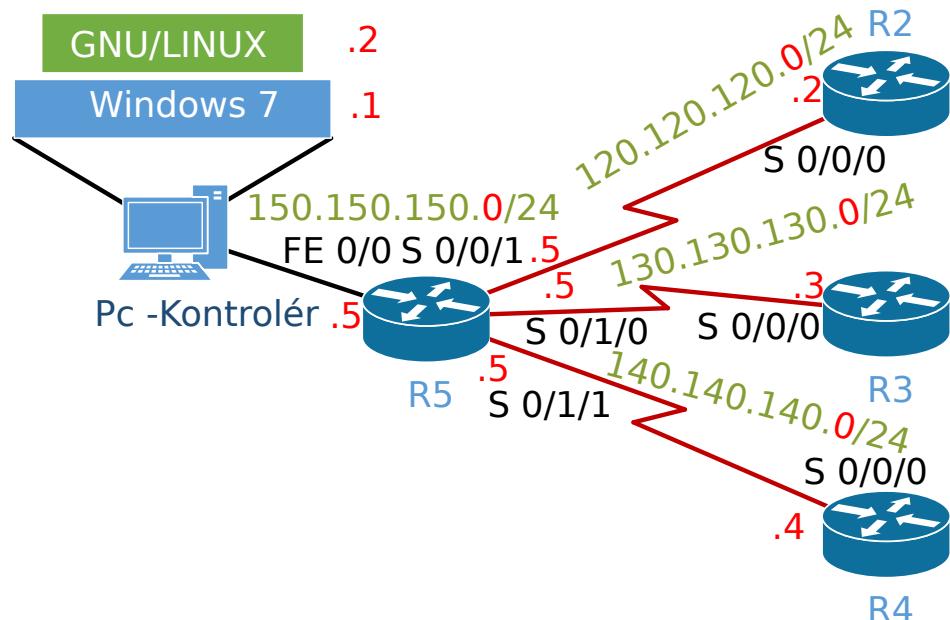
```
crypto pki import demoTP pkcs12 [URI] password [PASS]
username [LOGIN] password [HESLO]
username [LOGIN] privilege 15
onep
transport type tls local-cert demoTP disable-remotecert-validation
service set vty
```

Kde [URI] je URI, kde je uložený certifikát vo formáte pkcs12, užívateľské meno a heslo, ([LOGIN], [PASS]), sú využívané pri autentifikovaní aplikácie. Nastavenie komunikácie pomocou TLS a podpora VTY SS.

Dodatok B

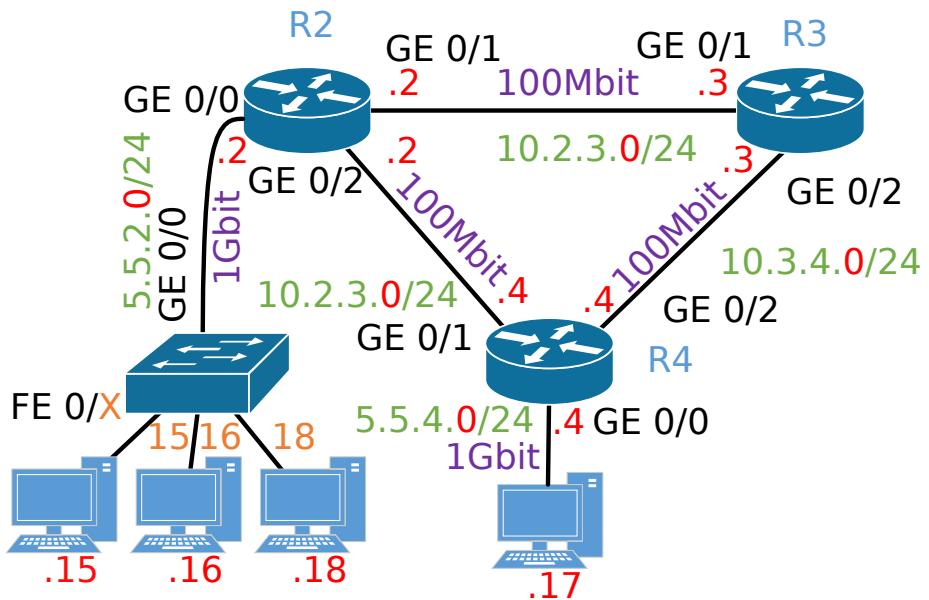
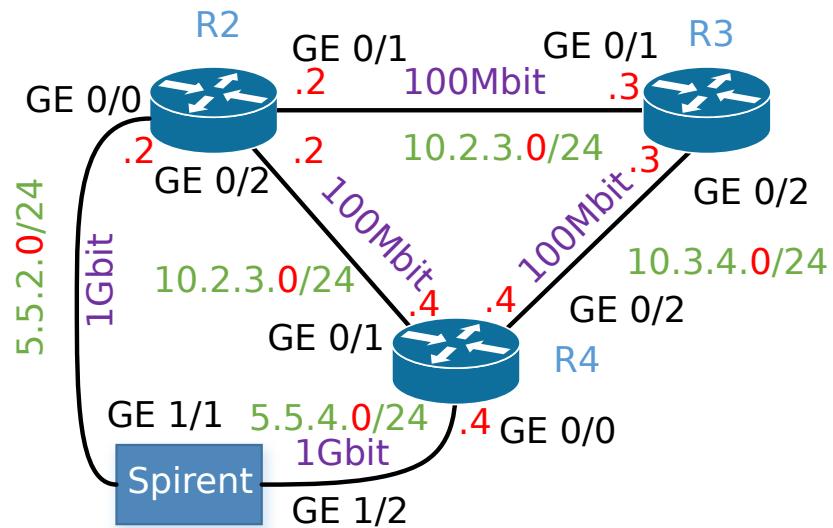
Príklad použitia

B.1 Out of Band topológia



Obr. B.1: Out of band riadiaca topológia.

B.2 In band topológia



Obr. B.2: Inband meracia topológia.

B.3 Príklad konfiguračného jazyka

```
{  
    "devices": {  
        "router": {  
            "1": {  
                "ipv4": "120.120.120.2",  
                "username": "cisco1",  
                "password": "cisco1",  
                "transport_type": "TLS",  
                "root_cert": "/home/cisco/ca.pem",  
                "client_cert_path": "",  
                "client_key_path": "",  
                "key_phrasses": ""  
            },  
            "2": {  
                "ipv4": "130.130.130.3",  
                "username": "cisco1",  
                "password": "cisco1",  
                "transport_type": "TLS",  
                "root_cert": "/home/cisco/ca.pem",  
                "client_cert_path": "",  
                "client_key_path": "",  
                "key_phrasses": ""  
            },  
            "3": {  
                "ipv4": "140.140.140.4",  
                "username": "cisco1",  
                "password": "cisco1",  
                "transport_type": "TLS",  
                "root_cert": "/home/cisco/ca.pem",  
                "client_cert_path": "",  
                "client_key_path": "",  
                "key_phrasses": ""  
            }  
        }  
    },  
    "topology": {  
        "1": {  
            "connected_to": {  
                "2": {  
                    "via": {  
                        "interface": ["GigabitEthernet0/1"]  
                    }  
                },  
                "3": {  
                    "via": {  
                        "interface": ["GigabitEthernet0/2"]  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        }

    }

}

"2": {
    "connected_to": {
        "1": {
            "via": {
                "interface": ["GigabitEthernet0/1"]
            }
        },
        "3": {
            "via": {
                "interface": ["GigabitEthernet0/2"]
            }
        }
    }
},
"3": {
    "connected_to": {
        "2": {
            "via": {
                "interface": ["GigabitEthernet0/2"]
            }
        },
        "1": {
            "via": {
                "interface": ["GigabitEthernet0/1"]
            }
        }
    }
}
},
"access_list": {
    "1": {
        "1": {
            "access": "permit",
            "src": "5.5.2.18",
            "src_prefix": "32",
            "dst": "5.5.4.17",
            "dst_prefix": "32",
            "proto": "all",
            "dscp": "000000"
        },
        "router": ["1", "2", "3"]
    },
    "2": {

```

```

        "1": {
            "access": "permit",
            "src": "5.5.4.17",
            "src_prefix": "32",
            "dst": "5.5.2.18",
            "dst_prefix": "32",
            "proto": "all",
            "dscp": "000000"
        },
        "router": ["1", "2", "3"]
    }
},
"gosia_routing": {
    "1": {
        "1": {
            "via": ["1", "3"]
        },
        "2": {
            "via": ["1", "2", "3"]
        }
    },
    "2": {
        "1": {
            "via": ["3", "1"]
        }
    },
    "3": {
        "1": {
            "via": ["1", "3"]
        }
    }
},
"gosia_interface": {
    "router": {
        "1": {
            "GigabitEthernet0/1": {
                "tx_speed_per_s_max": "50000000",
                "rx_speed_per_s_max": "50000000",
                "tx_speed_per_s_min": "30000000",
                "rx_speed_per_s_min": "30000000",
                "reliability": "50"
            },
            "GigabitEthernet0/2": {
                "tx_speed_per_s_max": "50000000",
                "rx_speed_per_s_max": "50000000",
                "tx_speed_per_s_min": "30000000",
                "rx_speed_per_s_min": "30000000",
                "reliability": "50"
            }
        }
    }
}

```

```
    },
    "2": {
        "GigabitEthernet0/1": {
            "tx_speed_per_s_max": "50000000",
            "rx_speed_per_s_max": "50000000",
            "tx_speed_per_s_min": "30000000",
            "rx_speed_per_s_min": "30000000",
            "reliability": "50"
        }
    }
}
```

Dodatok C

Obsah CD

Priložený dátový optický nosič obsahuje nasledujúce zložky:

- Thesis
- Code

Dodatok D

Zoznam skratiek

Skratka	Popis
AAA	Authentication, Authorization and Accounting
ACE	Access Control Entry
ACL	Access Control List
AF	Assured Forwarding
API	Application Programming Interface
BGP	Border Gateway Protocol
CAM	Content Adressable Memory
CDP	Cisco Discovery Protocol
CS	Class Selector
DoS	Denial of Service
DPI	Deep Packet Inspection
DSCP	Differentiated Service Code Point
ECMP	Equal Cost Multipath
ECN	Explicit Congestion Notification
EEM	Embedded Event Manager
EF	Expedited Forwarding
EIGRP	Enhanced Interior Gateway Routing Protocol
I2RS	Interface to the Routing System
IP	Internet Protocol
IP SLA	Internet Protocol Service-level agreement
IPC	Inter-process communication
ISO/OSI	International Organization for Standardization/ Open Systems Interconnection model
L2	Layer 2
MAC	Media Access Control
NB	NorthBound
NETCONF	Network Configuration
NGN	Next Generation Network
OF	OpenFlow
onePK	one Platform Kit

Skratka	Popis
OSPF	Open Shortest Path First
PBB	Provider Backbone Bridges
PBR	Policy Based Routing
PfR	Performance Routing
QoS	Quality of Service
RED	Random Early Detection
RIP	Routing Information Protocol
RPC	Remote Procedure Call
RSVP	Resource ReSerVation Protocol
SB	SouthBound API
SDK	Software Development Kit
SDN	Software Defined Networking
SDX	Software Defined Internet Exchange
SNMP	Simple Network Management Protocol
SS7	Signalling System 7
SSH	Secure SHell
TCP	Transmission Control Protocol
TELNET	TELEcommunications NETwork
ToS	Type of Service
TTL	Time To Live
UDP	User Datagram Protocol
VLAN	Virtual Local Area Networks
WFQ	Weighted Fair Queue
WRED	Weighted Random Early Detection
XNC	eXtensible Network Controller