Meranie dostupnej šírky pásma v softvérovo riadených sieťach

(Availble Bandwidth Measurement in Software Defined Networks)

Jakub Jasaň
Fakulkta informatiky a informačných technológií
Slovenská Technická Univerzita
Slovenská republika, 841 04 Bratislava
Email: xjasanj@stuba.sk

Miloslav Slížik
Fakulkta informatiky a informačných technológií
Slovenská Technická Univerzita
Slovenská republika, 841 04 Bratislava
Email: xslizikm@stuba.sk

***Abstract*— Softvérovo riadené siete (SDN) sú stúpajúcov paradigmou, ktorej predpokladom je renovovať počítačové siete. S oddelením dátovej a logickej časti siete, a s prínosom komunikáčných rozhraní medzi vrstvami, SDN umožnuje programovateľnosť celej siete, čo by mohlo priniesť rýchlu inováciu v tejto oblasti. SDN koncept sa ukázal byť úspešný v cloudových a dátových centrách. V našom článku sme sa zamerali na porovnanie výsledkov dostupnej šírky pásma v sieti, ktorú merali aj v opísanom článku[1]. Z naších výsledkov sme zistili, že metóda, ktorá bola použitá v uvedenom článku nie je úplne dôverihodná, nakoľko naše výsledky sa značne líšili pri použití rovnakých nástrojov.**

***Keywords*— Softvérovo riadené siete (SDN), dostupná šírka pásma, Floodlight, Mininet**

I. ÚVOD

V dnešnej dobe sú počitačové siete všadeprítomné. Každý bežný deň sme takmer neustále pripojení k internetu, to isté platí aj počas pracovnej dobe, keďže v mnohých pracových odvetviach je prístup k internetu nevyhutný. Rôzne požiadavky na sieť dospeli k tomu, že IP siete sú veľmi komplexné na vytvorenie a udržanie. Softvérovo riadené siete (SDN) nám ponúkajú riešenie v podobe rôznych vlastností. Najvýznamnejšie z nich sú oddelenie dátovej časti od kontrolnej, kontrolná časť je ostránená zo sieťových zariadení (napr. switche) a vonkajšie aplikácie dokážu riadiť sieť pomocou mechanizmou, ktoré nám ponúka SDN kontroler. V článku, z ktorého sme vychádzali sa zamerali na meranie jednej časti Kvalít Služieb (Quality of Service, QoS) a to meranie dostupnej šírky pásma v sieti, nakoľko problém s optimalizáciou priepustnosti v sieti sa zaoberá málo prác.
Hlavným prínosom tejto práce bolo overiť postup merania a následné výsledky východzej práce. V prvej časti práce sme zanalyzovali prostriedky, ktoré sme použili pri meraní a implementácií. Druhá časť práce sa zaoberá samotnou implementáciou a metódami merania a nakoniec v tretej časti sú podrobne opísané testovacie scenáre a výsledky.V závere sme zhodnotili výsledky a porovnania.

II. ANALÝZA

V tejto časti práce sa budeme zaoberať prostriedkami a metódami, ktoré sme využili na meranie dostupnej šírky pásma. V tabuľke (tabuľka 1) je opísaný hardvér a softvér, použitý pri meraniach. Na simuláciu SDN siete sme použili Mininet, a ako kontroler pre SDN sme použili Floodlight a pomocou jeho Rest API sme získavali informácie o dostupnej šírke pásma.

Host CPU	Intel Core i5 3360M
Pamäť hosta	16GB
Virtualizácia	VirtualBox 5.1
Guest OS	16.04 64-bit
Konfigurácia VM	2 CPU jadrá, 2GB pamäť
Verzia Mininetu	2.2
Verzia Floodlightu	1.2

Tabuľka 1. Použitý hardvér a softvér pri meraní

A. Mininet

Mininet je emulátor siete, ktorá vytvára virtuálnych hostov, prepínače (switch), kontrolery (controller) a linky. Mininet hostovia sa spúšťajú na štandardnom Linuxovom sieťovom softvéri, a jeho prepínače podporujú OpenFlow pre flexibilitu vlastného routovania a Softvérovo Riadené Siete[2].

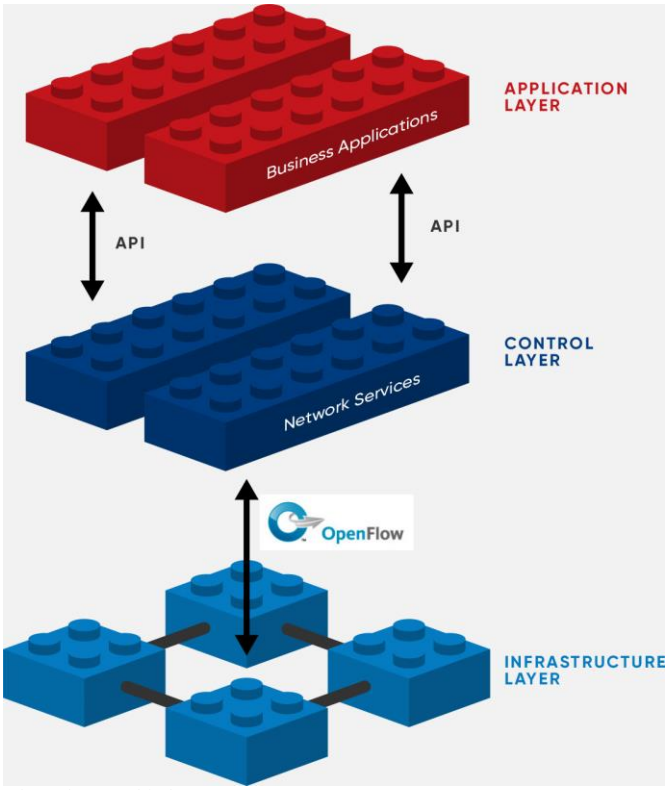
Mininet podporuje výskum, vývoj, štúdium, tvorbu prototypov, testovanie, debugovanie a každú inú úlohu, ktorá by bola nápomocná pri sieťovom experimentovaní na vlastných počítačoch[2].

Mininet poskytuje[2]:

- Jednoduché a nenáročné testovacie prostredie siete pre vývoj OpenFlow aplikácií
- Umožnuje viacerím vývojárom súbežne a nezávisle pracovať na rovnakej topológií
- Umožnuje testovanie komplexných topológií bez prepájania fyzickej siete
- Obsahuje CLI pre debugovanie a spúštanie testov na sieti

B. *Softvérovo riadené siete (SDN)*

Softvérovo riadené siete sú vznikajúca architektúra, ktorá je dynamická, ovládateľná, lacná a adaptívna. Tieto vlastnosti z nej robia ideálnu pre aplikácie, ktoré vysoko zaťažujú sieť a táto záťaž je dynamická. Táto architektúra (obrázok 1) oddeluje riadenie siete a funkcie forwardovania umožnujú riadeniu siete, aby bola priamo programovateľná. OpenFlow protocol je fundamentálnym elementom pre tvorbu SDN riešení[3].



Obrázok 1. Architektúra SDN [3]

SDN architektúra je [3]:

- ***Priamo programovateľná***

Riadenie siete je priamo programovateľné vďaka oddeleniu od forwardovacích funkcií

- ***Agilná***

Oddelenie riadenia od forwardovania nám umožní dynamicky prispôsobovať premávku siete podľa naších potrieb

- ***Centralizovaná***

Logika siete je centralizovaná v softvérovo riadených kontroleroch, ktoré udržujú celkový pohľad na siet, ktorý sa zobrazuje aplikáciam v sieti ako jeden logický prepínač (switch)

- ***Programovo konfigurovateľná***

SDN umožnuje konfigurovať, udržiavať, zabezpečiť a optimalizovať sieť rýchlo pomocou dynamických a automatizovaných SDN programov, ktoré si môžeme aj vytvoriť aj my.

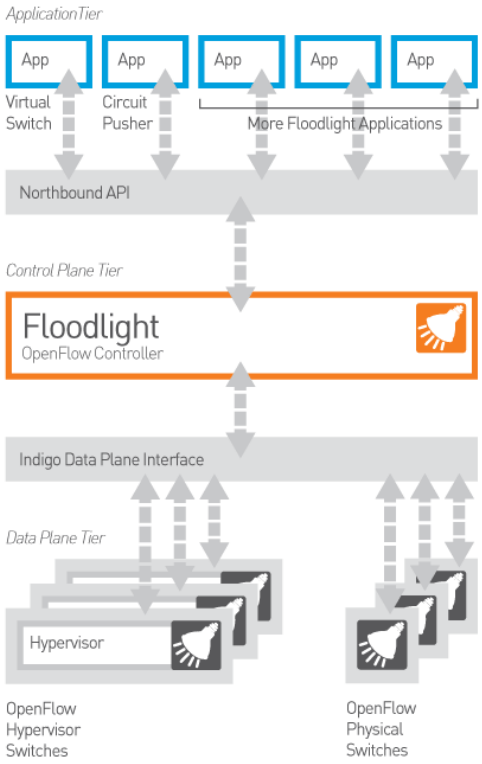
C. *Floodlight*

Floodlight Open SDN kontroler je enterprise-class, Apache licencovaný, na Jave postavený OpenFlow kontroler. Je podporovaný komunitou vývojárov vrátane inžinierov z Big Switch Networks [4][5].

OpenFlow je otvorený štandard, ktorý je riadený Open Networking Foundation. Špecifikuje protokol cez, ktorý je možné meniť správanie sieťových zariadení cez definované sady forwarding inštrukcií. Floodlight bol navrhnutý tak, aby dokázal pracovať s rastúcim počtom prepínačov, routrov, virtuálnych prepínačov a dostupných bodov ktoré podporujú OpenFlow štandard [4].

Vlastnosti Floodlightu [4]:

- Ponúka modulárny načítavací systém, ktorý uľahčuje rozšírenie a vylepšenie
- Jednoduché nastavenie s minimálnym počtom dependencies
- Podporuje širokú škálu virtuálnych a fyzických OpenFlow prepínačov
- Navrhnutý na rýchlu prevádzku – je multithreadový od základov
- Podpora pre OpenStack [6] cloud orchestration platform

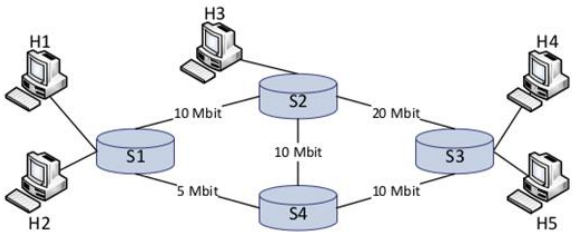


Obrázok 2. Architektúra Floodlight [4]

III. IMPLEMENTÁCIA

Program na meranie dostupnej šírky pásma je implementovaný v jazyku python a počíta s tým, že všetky linky v sieti sú obojsmerné.

Cieľom programu je merať dostupnú šírku pásma pre dva scenáre. Prvý scenár je pokiaľ je routovanie v sieti fixné. V druhom prípade hľadáme najlepšiu dostupnú cestu zo všetkých možných od jedného hosta k druhému.



Obrázok 3. Testovacia topológia siete

V rámci prípravy testovacieho scenáru bolo potrebné vytvoriť v mininete testovaciu topológiu (obrázok 2), pomocou python skriptu. V testovacej topológii je obmedzená priepustnosť liniek pomocou Linux Traffic Control, analogicky k pôvodnému článku. Takisto rovnako ako v pôvodnom článku[1], program pre zistenie potrebných informácii o sieti dopytuje Floodlight SDN controller cez REST API.

Prvým problémom, na ktorý sme narazili bolo, že Floodlight nepreberá informáciu o obmedzení priepustnosti linku cez Traffic Control.
Prvý krok výpočtu dostupnej šírky pásma je pre obidva scenáre rovnaký. Zisťujeme kapacitu pre všetký hrany v sieti. Overiť si to je možné cez REST API query: $ *curl -X GET <controller ip:8080>/wm/core/switch/all/port-desc/json*, ktorá vráti pre všetký linky rovnakú kapacitu, aj keď v sieti majú niektoré linky reálne obmedzenú kapacitu. Tento problém sme nevedeli vyriešiť, tak sme dostupnú šírku pásma pre konkrétne linky vložili do programu na pevno.
Následne chceme zistiť aktuálne zaťaženie linky a to tak, že dopytujeme cez REST API na každej linke counter jedného switchu/portu. Konkrétne nás zaujímajú hodnoty RX a TX bytes. Floodlight poskytuje rovno hodnoty RX-bits-per-second a TX-bits-per-second + čas ku ktorému sú tieto hodnoty platné. Sčítame hodnoty RX a TX bits per second. Tým dostaneme aktuálnu záťaž na všetkých linkoch v bitoch za sekundu. Použitím vzorca na výpočet dostupnej šírky pásma: Dostupná šírka pásma = kapacita linku - aktuálna záťaž, zistíme pre všetky linky aktuálnu dostupnú šírku pásma. V tomto bode sa na základe voľby argumentu pri spustení programu začne vykonávať buď prvý, alebo druhý scenár.

Druhý scenár, hľadanie najlepšej dostupnej cesty v grafe spočíva v spustení upraveného Dijkstrovho algoritmu v cykle, čím sa vyhľadá cesta s najväčšou šírkou pásma pre všetky kombinácie switchov. Dijkstrov algoritmus je implementovaný pomocou python knižnice na minimálnu haldu. Keďže potrebujeme nie minimálne, ale maximálne hodnoty dostupnej šírky pásma, tak aktuálne hodnoty dostupnej šírky pásma pre každý link vynásobíme -1, čím sa docieli efekt "zmeny" minimálnej haldy na maximálnu.

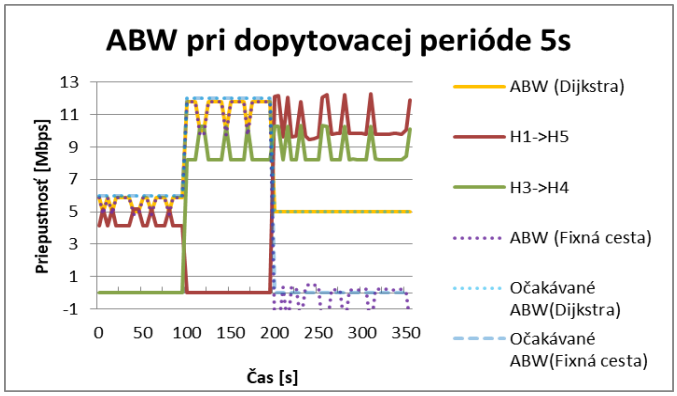
Prvý scenár, zisťovanie maximálnej šírky pásma pre fixné cesty v sieti sa skladá z viacerých krokov. Najprv zisťujeme body pripojenia k sieti pre všetkých hostov, čím zistíme konkrétny switch a port ku ktorému sú pripojený. Ide to cez dopytovanie REST API príkazom :
$ curl -X GET <controller ip:8080>/wm/device/ .
Ďalší krok je zistenie aktuálnych tokov v sieti, to sa vykonáva cez dopytovanie REST API :*$curl –X GET <controller ip:8080>/wm/core/switch/all/flow/json/*. Ignorujeme toky, ktoré generuje controller, ako odosielanie LLDP paketov, pretože sú tak malé, že pre výpočet dostupnej šírky pásma sú irelevantné a zbytočne by predlžovali dobu výpočtu dostupnej šírky pásma na fixovaných cestách. Zo zoznamu tokov zisťujeme, aké cesty má controller pre tieto toky a to cez dopytovanie REST API : *curl -X GET <controller ip:8080>/wm/routing/path/<src-dpid>/<src-port>/<dst-dpid>/<dst-port>/json/* . Následne prechádzame v cykle tieto cesty, kde sú momentálne toky a vzorcom : dostupná šírka pásma pre cestu = minimálna šírka pásma všetkých linkov na ceste , zistíme dostupnú šírku pásma pre všetky cesty.

IV. VÝSLEDKY

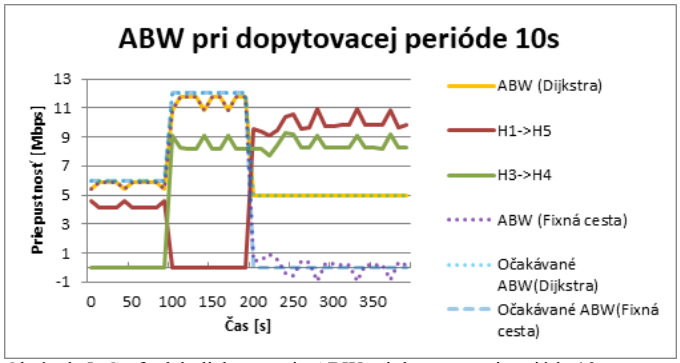
V topológií siete (obrázok 3), ktorú sme vytvorili v Mininete, sme použili ako SDN kontroler Floodlight pre naše testovnia. Mininet sme spúštali vo virtuálnom stroji na host serveri pomocou VirtualBoxu a Floodlight priamo na host serveri. Na testovanie sme si vytvorili štyri switche (S1,S2,S3,S4), päť hostov a dva testovacie scenáre a to meranie ABW na fixovaných cestách a meranie ABW na najlepšej možnej ceste z hľadiska priepustnosti. Pre oba scenáre sme spravili tri rôzne merania, ktoré sa líšili v dopytovacej perióde na switch (2, 5 a 10 sekúnd) z ktorého sme získavali rx a tx hodnoty, ktoré nám priamo udávali prijaté a odoslané bity za sekundu na porte.
Na to, aby sme dokázali merať dostupnú šírku pásma sme potrebovali použiť program na generovanie prevádzky.
Ako prvú možnosť sme vyskúšali D-ITG, pretože ho používali aj vo vybranom článku, z ktorého sme vychádzali, avšak pri našom testovaní sme narazili na problém, keď D-ITG neposielalo približne 10% paketov, ktorých sme mu zadali aby posielal. Táto odchylka vytvárala príliš veľké rozdiely pri dopytovaní, a tak sme sa rozodli použiť iperf, ktorý generoval prevádzku presnejšie.

Oba scenáre boli testované na prevádzke s konštantným bit rate-om po dobu štyristo sekúnd. Ako prvá UDP prevádzka, s veľkosťou 4Mbps, sa vykonala medzi H1 a H5 po dobu sto sekúnd. Po tejto dobe sa H1 uspal (teda negeneroval žiadnu prevádzku) na ďalších 100 sekúnd. Následne, po uplinutí 100 sekúnd, sa znovu obnovila UDP prevádzka na tejto ceste ale s veľkosťou 10Mbps. Paralelne s tým, ako sa H1 uspal začal H3 posielať UDP prevádzku o veľkosti 8Mbps do H4 až do

Obrázok 4. Graf odchyliek merania ABW pri dopytovacej perióde 5s



Obrázok 4. Graf odchyliek merania ABW pri dopytovacej perióde 10s



Obrázok 5. Graf odchyliek merania ABW pri dopytovacej perióde 10s

konca testovania. Obrázok 4 a 5 nám znázorňujú grafy týchto prevádzok a dostupnú šírku pásma (ABW) pre meranie na fixných cestách a pomocou Dijkstrovho algoritmu, ktorý nám odmeral ABW podľa najlepšej možnej ceste z hľadiska priepustnosti. Z grafov je možné si všimnúť, že ABW pomocou Dijkstrovho algoritmu nie je konštantné do momentu, kým H1 a H3 tvoria prevádzku súčasne. Toto je spôsobené tým, že cesty medzi S1,S2 a S2,S3 sa natoľko zaplnia, že alternatívna cesta cez S4 nám ponúka väčšiu ABW a keďže po tejto ceste nejde žiadna prevádzka, je táto hodnota konštantná, a síce 5Mbps.

Výsledky prvého a druhého scenára, sú znázornené v grafoch na obrázkoch 4 a 5. Môžeme si všimnúť, že ABW je rovnaké pre oba scenáre po dobu, kým sa alternatívna cesta cez S4 nestane lepšou. V prvom scenári túto cestu nevyužívame, pretože meriame ABW pre fixnú cestu S1 -> S2 -> S3. V grafe je možné si všimnúť, že hodnoty ABW kolíšu viac pri väčšej dopytovacej perióde. Toto je spôsobené tým, že pri väčšej dopytovacej perióde priemerujeme hodnoty v dlhšom čase, zatial čo rozdiel aproximácií timestampov ostáva rovnaký, čiže vytvára menší relatívny efekt. Tak isto v oboch grafoch je znázornená očakávaná hodnota ABW pre oba scenáre, aby sme vedeli vypočítať chybovosť našeho merania a porovnali s hodnotami, ktoré sú prezentované z východzieho článku. Toto kolísanie hodnôt si tak isto môžeme všimnúť pre meranie prevádzky v sieti, kde keď sme chceli prevádzku o veľkosti 10Mbps tak nám counter na switchi vrátil hodnoty v rozmedzí od 9,5Mbps až po 12 Mbps. Tieto odchylky sú zachytené v tabuľke 2.

Dopytovacia perióda v sekundách

	2	5	10	10(100ms)	10(10ms)
H1->H5	31.25%	7.66%	4.86%	5.28%	7.44%
H3->H4	31.81%	8.05%	4.62%	5.61%	5.61%
ABW(Dijkstra)	12.27%	2.76%	2.23%	3.56%	4.21%
ABW(Fixed)	26.56%	4.85%	4.47%	4.72%	5.58%

Tabuľka 2. Priemerné hodnoty relatívnych odchyliek od očakávaných hodnôt

Namerané hodnoty zachytené v tabuľke 2 nám potvrdzujú trend zhoršovania výsledkov pri zvyšovaní frekvencie dopytovania sa na prepínač. K meraniam sme aj pridali meranie keď sme umelo vytvorili oneskorenie v dopytovaní sa na switch (100 a 10 ms). V tomto prípade sme ale zaznamenali opačný trend aký dosiahli v článku, a to keď sme znížili oneskorenie pri dopytovani tak sa nám zhoršili výsledky. Zvláštnosťou je, že pri oneskorení 100ms sme dosiahli horšie výsledky ako keď sme nemali žiadne oneskorenie, a keď sme ho znížili tak sa odchýlka zvýšila. Tento fenomén pripisujeme nepresnosti Floodlightu, ktorý nám pomocou REST API vrátil hodnoty priepustnosti na portoch na switchi.

V. ZÁVER

V tejto práci sme otestovali spôsob merania dostupnej šírky pásma, aký bol prezentovaní v článku [1]. Najväčším problémom pre testovaní bolo pre nás slabo zdokumetovaný spôsob merania v uvedenom článku. Autori vôbec neuviedli, akým spôsobom merali ABW pre uvedené tri scenáre v štvrtej kapitole, a tak sme otestovali dva spôsoby, ktoré boli uvedené v kapitole tri, a to meranie ABW pre fixné cesty a pre najlepšie cesty pomocou Dijkstrovho algoritmu. Naše výsledky sa značne odlišujú od výsledkov, ktoré dosiahli autori, aj keď trend zhoršovania sa výsledkov pri vyššej frekvencií dopytovania sa na counter prepínača overil. Pri meraní sme narazili na chybu vo floodlighte, keď pri znížení dopytovacej periódy na 2s sa približne raz za 5 meraní stalo, že floodlight hlásil dvojnásobnú hodnotu prijatých a odoslaných bitov za sekundu na countri switchu. Tieto veľké rozdiely vo výsledkoch mohli byť aj spôsobené tým, že my sme hodnotu prevádzky na porte brali priamo cez REST API Floodlightu, zatiaľ čo autori článku použili *FlowStatsReq* OpenFlow správu a z nej si pomocou vzorca počítali ABW v čase, aj keď si nemyslíme, že by to vytvorilo rádovo veľké rozdiely a je ťažko uveriteľné, že by sme touto metódou dosiahly tak presné výsledky (v najlepšom prípade 0.06% odchýlka).

ODKAZY

[1] Péter Megyesi, Alessio Botta, Giuseppe Aceto, Antonio Pescapè, and Sándor Molnár. 2016. Available bandwidth measurement in software defined networks. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (SAC '16). ACM, New York, NY, USA, 651-657. DOI: https://doi.org/10.1145/2851613.2851727

[2] Mininet Team. *Mininet Overview*. 2017. http://mininet.org/overview/

[3] Open Networking Foundation. *Software-Defined Networking (SDN) Definition.* 2017. https://www.opennetworking.org/sdn-definition/

[4] Project Floodlight. *Floodlight*. 2017. http://www.projectfloodlight.org/floodlight/

[5] Big Switch Networks. 2017. https://www.bigswitch.com/

[6] OpenStack. 2017. https://www.openstack.org/