Zabezpečenie WAN sieti voči nečakaným udalostiam, aplikáciou SDN technológie

Michal Škuta
Fakulta informatiky a informačných technológií
Slovenská Technická Univerzita
Bratislava, Slovenská republika
Email: michal.skuta@gmail.com

Jaroslav Lišiak
Fakulta informatiky a informačných technológií
Slovenská Technická Univerzita
Bratislava, Slovenská republika
Email: jaroslav.lisiak@gmail.com

***Abstract*—Tento dokument vznikol za účelom overenia článku „An experimental feasibility study on applying SDN technology to disaster-resilient wide area networks“[1]. Autori článku sa pokúsili overiť použitie SDN technológie, ako prostriedku, ktorý rýchlejšie obnoví WAN sieť pri nečakanej udalosti v podobe výpadku spojenia. Tento dokument ma za úlohu ozrejmiť spôsob akým sme overovali výsledky autorov článku. Taktiež ma za úlohu zdokumentovať simuláciu a testovanie. Na záver porovnať naše výsledky simulácie s výsledkami z článku.**

Keywords—SDN; mininet; POX; RYU; OpenFlow; SINET; iPerf; bwm-ng;

I. ÚVOD

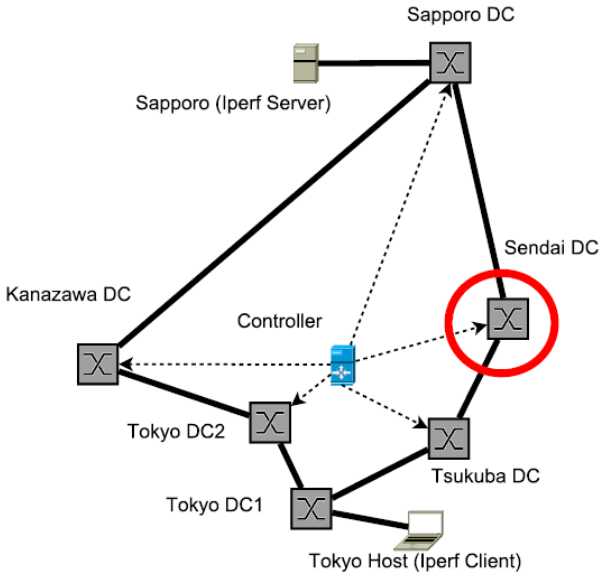
Článok opisuje výhody používania SDN sieti v prípade vyskytnutia sa nečakanej pohromy (búrky, tsunami). Znefunkčnenie dôležitého prvku v hlavnom uzle siete (backbone) dôsledkom nečakanej udalosti, by mohlo mať veľmi kritické následky. Autori preto experimentálne otestovali aplikovanie SDN technológie v backbone. Overovali správanie siete pri nečakaných výpadkoch dôležitých komponentov v backbone a následné presmerovanie toku dát. Ukázali efektivitu SDN technológie a rýchlosť reakcie pri nečakaných udalostiach v sieti. Overili reakčnú dobu medzi kontrolerom a prvkami siete. Overili taktiež end-to-end komunikáciu pri kritickom scenári s použitím SDN technológie a bez nej.

II. ANALÝZA

Práca je rozdelená na tri samostatné testovania. Prvé testovanie sa zaoberá reakčným časom, ktorý je potrebný na presmerovanie toku dát na switchi bez pomoci kontroléra. Na vytvorenie softvérového switchu je použitá aplikácia OpenvSwich, ktorá implementuje protokol OpenFlow. Protokol OpenFlow umožňuje použitie skupín(group). Vytvorená grupa je typu FastFailover. Použitá grupa slúži na zisťovanie stavu portu zariadenia. Vďaka nej vie switch okamžite reagovať na náhle zmeny v stave portu (on/off), bez pomoci kontroléra. V prípade ak je jeden z portov vypnutý, alebo stratí konektivitu (spadne linka), switch okamžite presmeruje tok dát na druhý nastavený port. V prvej časti práce autori počítajú čas od doručenia posledného paketu na linke, ktorá bola prerušená až po čas kedy bol prijatý prvý paket na druhej linke. Autori vykonali 50 takýchto testov a ich cieľom bolo dosiahnuť čas na obnovu siete menej ako 50 ms.

Druhá časť práce sa zaoberá výpočtom oneskorenia medzi kontrolérom a SDN zariadením v simulovanej backbone sieti. V prvom kroku na základe počtu zariadení v danej sieti a predom určenej rýchlosti toku dát medzi nimi vypočítali priemernú a najhoršiu dobu odozvy. V ich simulovanej sieti poupravili rýchlosť toku dát, tak aby ju prispôsobili, prostriedkom ktoré mal simulátor siete k dispozícií. V danej topológii menili počet kontrolérov a merali doby odozvy pri použití rôznych počtov kontrolérov a zároveň menili umiestnenie a pripojenia jednotlivých kontrolerov. S vyšším počtom kontrolérov v sieti klesala doba odozvy. Pri použití 4 a 5 kontrolerov nebol rozdiel v dobe odozvy až taký výrazný.

Posledná, tretia časť sa zaoberá tokom dát cez backbone sieť z pohľadu používateľa. Autori vytvorili simuláciu SINET siete v Japonsku. Topológia je vyobrazená na obrázku nižšie. Na tejto schéme simulovali výpadky spojenia medzi datacentrami, ktoré nastali v roku 2013. Následne sledovali čas potrebný na konvergenciu siete a obnovenie toku dát v skonvergovanej sieti. Sledovali tok dát počas simulácie výpadkov a zisťovali aký vplyv by mal výpadok spojenia v simulovanej SDN sieti na koncového používateľa.

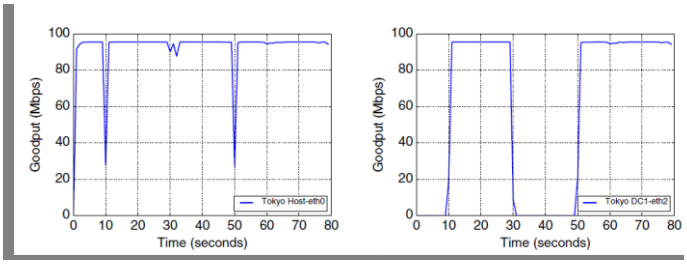


Obr. 1 Testovaná topológia

III. NÁVRH

V našom návrhu plánujeme zrealizovať topológiu z poslednej časti článku. Autori sa snažili vytvoriť WAN sieť podobnú reálnej siete SINET3 v Japonsku. Autori článku taktiež špecifikujú podrobný priebeh simulácie a teda presný čas trvania simulácie a približné časy prerušení spojenia medzi uzlami siete. Túto simuláciu taktiež plánujeme zopakovať a budeme pozorovať správanie sa siete z pohľadu koncového užívateľa. Užívateľ je pripojený do tejto siete a snaží sa komunikovať so serverom umiestneným v tejto sieti. Nato aby mohli komunikovať je potrebné aby existovala cesta medzi serverom a klientom. V prípade prerušenia spojenia už táto cesta nemusí správne fungovať (byť kompletná) a je potrebné vypočítať novú cestu komunikácie medzi serverom a klientom. Takéto hľadanie novej cesty bude užívateľ vnímať ako spomalenie komunikácie so serverom (prípadne pozastavenie ak by sa cesta hľadala dlhší čas).

Autori článku prezentujú tieto informácie v grafe, kde na osi x je znázornený celý čas simulácie a na osi y je dosiahnutá rýchlosť na vybraných portoch (port klienta smerom k sieti a port uzla, cez ktorý bude viesť alternatívna cesta pri prerušení spojenia). Na obrázku nižšie sú znázornené tieto grafy. Podobne ako autori článku vytvoríme grafy a overíme či získané výsledky nášho testovania sa budú zhodovať s výsledkami, ktoré sú prezentované v článku. Prípadne rozdiely medzi našim testovaním a výsledkami z článku sa budeme snažiť podrobne analyzovať a vyhodnotiť.



Obr. 2 Namerané výsledky

Analyzovaný článok sa nezameriava na aplikovanie SDN technológii do WAN siete, ale hodnotí či dokáže súčasná technológia dostatočne rýchlo reagovať na nečakané udalosti, ktoré by mali za následok stratu dát. Nato aby sme mohli zopakovať a overiť testovanie z daného článku však musíme aplikovať SDN technológie do WAN sieti. Autori článku vytvorili modul pre SDN kontrolér POX, ktorý zabezpečuje linkovú a sieťovú vrstvu pre zariadenia a taktiež zabraňuje vzniku slučiek v sieti. V našom prípade si topológiu predstavíme v dvoch rôznych prípadov.

V prvom prípade sa sieťové uzly budú správať ako prepínače. Podobne ako pri prepínačoch tu budeme používať Spanning tree protocol na zamedzenie slučiek, ktoré by vznikali pri broadcastových rámcoch. Hneď tu sa ukazuje nevýhoda a to je blokovanie určitých spojení (na odstránenie slučiek) a tým zníženie priepustnosti siete. Ďalší problém je pomalá konvergencia siete a tým pádom dlhé výpadky pri zmene topológie. Problém pomalej konvergencie by sme chceli ďalej analyzovať a nájsť pomocou kontroléra riešenia na urýchlenie konvergencie. Serveri a klienti musia byt v rovnakej IP sieti a nepotrebujú smerovač na vzájomnú komunikáciu.

V druhom prípade sa sieťové uzly budú správať podobne ako smerovače. Každý uzol bude mať svoj unikátny identifikátor a zoznam sieti k nemu pripojených. Kontroler bude poznať kompletnú topológiu a pomocou dijkstru vypočíta pre každý uzol cesty do všetkých ostatných uzlov. V prípade zmeny topológie sa na kontroleri len znova spusti výpočet všetkých ciest medzi uzlami. V tomto prípade je potrebné aby klient a server boli v rôznych IP sieťach, tento návrh používa len sieťovú vrstvu.

V prípade oboch topológii je možné prerušenia spojenia hneď oznamovať kontroléru openflow správou typu PortStatus. Na túto správu vie kontrolér rýchlo reagovať a prepočítať nové cesty (alebo odblokovať zablokované cesty). Oba spôsoby implementácie SDN sieti v našej topológii porovnáme z hľadiska používateľa a porovnáme z údajmi od autorov článku.

IV. IMPLEMENTÁCIA

A. *Použité technológie*

1) Mininet

Mininet slúži na vytváranie realistických virtuálnych sieti. Umožňuje simuláciu SDN sieti, všetkých komponentov tejto siete a ich prepojenia. Mininet podporuje protokol OpenFlow, ktorý taktiež použijeme pri vypracovaní tohto projektu.

2) Open vSwitch

Open vSwitch je viacvrstvový virtuálny switch, licencovaný pod open source Apache 2.0 licenciou. V našom prípade použitý aj v aplikácii mininet.

3) OpenFlow

Definuje komunikačný protokol v SDN sieťach, ktorý umožňuje SDN kontroleru priamo komunikovať s SDN zariadeniami.

4) SDN kontrolér

SDN kontrolér je základným prvkom SDN sieti, získava dáta z jednotlivých zariadení a na základe týchto dát vykonáva základnú riadiacu logiku nad týmito zariadeniami.

5) Spanning Tree Protocol

STP je protokol, ktorý v linkovej vrstve modelu OSI slúži na riešenie problémov so slučkami. Používa BPDU správy na vymieňanie si informácii o prepínačoch a na dohodnutie sa na jednotnej kostre grafu. V súčasnosti sa nahradzuje rýchlejšími alternatívami (napr.RSTP).

6) Dijkstra

Dijkstrov algoritmus je jedným zo základných algoritmov teórie grafov, jeho primárnym využitím je hľadanie najkratšej cesty v hranovo-ohodnotenom grafe. Tento graf pozostáva z množiny vrcholov, orientovaných hrán a funkcie, ktorá zobrazuje množinu hrán do množiny reálnych čísel. Typovo ide o algoritmus najkratšej cesty z jedného vrcholu do ostatných, najčastejšie však do jedného konkrétneho.

B. *Modifikácia kontroléra*

Autori vo svojej implementácií používali POX kontrolér, ktorý je stručne opísaný v predchádzajúcej kapitole. Do POX kontroléra autori doimplementovali vlastný modul, ktorý mal za úlohu smerovanie na 2. a 3. vrstve ISO OSI modelu a zabezpečoval bezslučkovosť.

Nakoľko však autori nezverejnili zdrojové kódy ich implementácie, bolo potrebné implementovať vlastnú funkcionalitu do kontroléra. My sme sa však rozhodli pre použitie RYU kontroléra, nakoľko sme s týmto kontrolérom už mali predchádzajúce skúsenosti. Pre kontrolér RYU môžeme nájsť veľké množstvo vzorových príkladov kódu, pomocou ktorých môžeme postupovať pri vlastnej implementácii. Do tohto kontroléra sme implementovali funkcionalitu opísanú v nasledujúcich kapitolách. Prídavnú funkcionalitu sme písali v programovacom jazyku Python a používali sme vývojové prostredie PyCharm od spoločnosti JetBrains. Implementovanú funkcionalitu sme rozdelili do dvoch súborov. Jeden súbor zabezpečoval smerovanie na základe MAC adries, teda na druhej vrstve ISO OSI modelu a ďalší súbor smerovanie pomocou IP adries na 3 vrstve IOS OSI modelu. Doimplementovaná funkcionalita sa skladala zo zabezpečenia spracovania nasledujúcich požiadaviek:

1) *Prijatie správy o zmene stavu rozhrania*

Pri prijatí správy od zariadenia na ktorom bol zmenený stav portu, kontrolér v uloženej topológii (grafe) odstráni spojenie medzi konkrétnymi zariadeniami. Zabezpečí sa tak, že v prípade ak sa bridge následne opýta kam smerovať, kontrolér má aktuálnu topológiu a vie cez ktoré spojenia nie je možné smerovať. V prípade ak kontrolér dostane správu, že niektoré z vypnutých rozhraní je obnovené, tak do topológie pridá spojenie medzi príslušnými zariadeniami.

2) *Zariadenie nevie kam smerovať dáta*

Pri spúšťaní zariadenia, má dané zariadenie základnú flow tabuľku. Po prihlásení sa do siete sa v tejto tabuľke vytvorí záznam / pravidlo. Toto pravidlo zabezpečuje, že v prípade ak SDN zariadenie (bridge) nevie kam má smerovať dáta opýta sa kontroléra. Takáto správa sa nazýva „PacketIn“ správa. Po prijatí tejto správy kontrolérom, kontrolér vypočíta kam sa má ďalej paket smerovať a odošle zariadeniu správu, kam ma paket poslať. Táto správa sa nazýva „PacketOUT“. Zároveň však kontrolér odošle aj FlowMode správu. Tú zariadenie spracuje v podobe vytvorenia nového pravidla do Flow tabuľky. Takto sa zabezpečí, že v prípade ak na zariadenie podobný paket, zariadenie bude vedieť kam ho ďalej odoslať bez potreby pýtania sa kontroléra.

V prípade, ak ani samotný kontrolér nevie kam sa má daný paket odoslať ďalej (nevie ako sa dostať k cieľu), rieši túto situáciu nasledujúco:

a) Ak sa jedná o layer2

Kontrolér povie zariadeniu, nech daný paket odošle broadcastom na všetky ostatné rozhrania, okrem toho, ktorým daný paket prijal.

b) Ak sa jedná o layer3

Kontrolér potrebuje zistiť umiestnenie cieľovej IP adresy. Z uloženej topológie zistí, ktoré zariadenie má podsieť z ktorej je cieľová IP adresa. Následne zariadeniu, ktoré sa pýta na cestu povie, nech pošle ARP request správu na zariadenie, ktoré ma pripojenú príslušnú podsieť.

3) *Potreba bezslučkovej topologie*

Pri implementovaní vlastnej funkcionality do kontroléra, bolo potrebné zabezpečiť danú sieť voči prípadným slučkám. To je dôležité najmä pri smerovaní na druhej vrstve, kde využívame posielanie broadcastových správ. V prípade ak by sme mali v našej sieti neošetrené slučky, vznikol by tzv. broadcast storm. Pre zabezpečenie bezslučkových spojení v topológií, kontrolérom blokujeme konkrétne rozhrania na konkrétnych zariadeniach.

Nakoľko kontrolér pozná celú topológiu siete, je možné nad daným grafom siete vytvoriť minimum spanning tree. Jedná sa o graf najkratších spojení medzi všetkými vrcholmi v sieti , bez toho aby v danom grafe vznikali slučky. Výsledný minimum spanning tree porovnáme s grafom celej našej topológie a identifikujeme rozhrania (spojenia), ktoré je potrebné vypnúť (prerušiť) aby sme docielili požadovanú topológiu bez slučiek. Príslušné rozhrania následne deaktivujeme.

Veľkou výhodou smerovania pomocou IP adries je, že nemusíme blokovať nadbytočné spojenia. V prípade ak bridge nevie kam má dáta smerovať opýta sa na ďalší postup kontroléra. Nakoľko kontrolér pozná celú topológiu siete, vypočíta najkratšiu cestu medzi zdrojovým a cieľovým zariadením. Po vypočítaní správnej cesty, kontrolér pošle príslušnému zariadeniu správu, kam ma daný paket odoslať a zároveň mu odošle aj informáciu, nech si do svojej „flow table“ uloží záznam s danými smerovacími informáciami, aby pri nasledujúcom smerovaní vedel kam má paket odoslať, bez potreby kontroléra.

Pre simuláciu TCP komunikácie sme použili aplikáciu IPerf. Jedná sa o rovnakú aplikáciu, akú použili autori článku. Táto aplikácia simuluje TCP komunikáciu medzi klientskym zariadením a serverom vo virtuálnej sieti. Umožňuje nastaviť veľké množstvo parametrov, my v tejto aplikácií nastavujeme iba dobu, počas ktorej má táto aplikácia simulovať TCP komunikáciu.

Na monitorovanie priepustnosti požadovaných sieťových prepojení používame aplikáciu bwm-ng. Taktiež aj v tomto prípade sa jedná o rovnakú aplikáciu, akú používali autori článku pri ich simulácií. Pri spúšťaní tejto aplikácie nastavujeme rozhranie, na ktorom bude aplikácia monitorovať priepustnosť, frekvenciu, ako často bude aplikácia monitorovať stav priepustnosti a typ štatistiky, ktorú chceme. Autori článku uvádzajú meranú priepustnosť ako „goodput“, no aplikácia bwm-ng slúži na monitorovanie priepustnosti (bandwidth). Preto sme sa rozhodli v našich výsledkoch používať názov priepustnosť.

Pre jednoduché spustenie nášho prototypu sme poskytli skripty, ktoré uľahčujú samotné testovanie. O priebehu testovania, ako sa spúšťa a ako prebieha informuje nasledujúca kapitola.

Grafové algoritmy

Pre prácu s grafovými algoritmami sme sa rozhodli použiť knižnicu NetworkX, ktorá je dostupná online[1]. Jedná sa o knižnicu pre Python. Pomocou tejto knižnice zisťujeme minimal spanning tree pre danú topológiu, aby sme zabránili tvorbe slučiek.

V. TESTOVANIE

Aplikácie použité pre simuláciu TCP komunikácie a monitorovanie stavu priepustnosti spojení v sieti, sú opísané v predchádzajúcej kapitole. Presný postup ako spustiť celú virtuálnu sieť nájdeme na našom github-e [2].

Pred samotným spustením mininetu je potrebné mať na zariadení nainštalované požadované programy, ktoré sú nutné pre správne fungovanie celej virtuálnej siete. Pre jednoduché nainštalovanie požadovaných programov je potrebné spustiť súbor ***deploy-scripts\check_ubuntu_dependecies.sh.*** Jedná sa o súbor obsahujúci shell príkazy, ktoré doinštalujú chýbajúce programy potrebné pre správny chod programu.

Pri testovaní implementovanej funkcionality má používateľ na výber medzi manuálnym spôsobom testovania a automatickým.

A. *Manuálne testovanie*

Pri manuálnom testovaní je potrebné vytvoriť samotnú topológiu spustením príkazu ***sudo python mininet-scripts/runner3.py***. Tento príkaz slúži na spustenie mininetu s požadovanou topológiou. Pre správne fungovanie kontroléra je potrebné spustiť kontrolér až po spustení mininetu príkazom ***ryu-manager --observe-links ryu/simple_switch_nx3.py***. Tieto príkazy slúžia na smerovanie na 3. vrstve. Pre zabezpečenie prepínania na 2. vrstve je potrebné zadať nasledujúce príkazy:

1) ***sudo python mininet-scripts/runner.py***
2) ***ryu-manager --observe-links ryu/simple_switch_nx.py***

Na zariadení predstavujúcom server spustíme Iperf aplikáciu príkazom ***iperf –s***. Na koncovom zariadení spustíme TCP komunikáciu so serverom spustením príkazu ***iperf -c 192.168.10.10 -i 1 -t 1000***. Následne je možné simulovať prerušenie spojení medzi jednotlivými zariadeniami. V našom prípade použijeme príkazy: ***link s3 s4 down*** a ***link s3 s4 up*** a jeho rôzne variácie.

B. *Automatické testovanie*

Automatické testovanie sa spúšťa zadaním príkazu ***sudo python mininet-scripts/runner3_auto.py***, pre otestovanie komunikácie na 3. vrstve.

Pre automatické testovanie na 2. vrstve spustíme príkaz ***sudo python mininet-scripts/runner_auto.py***.

[1] https://networkx.github.io/

[2] https://github.com/aks-2017/semestralne-zadania-semestralne-zadanie-xskuta-xlisiak

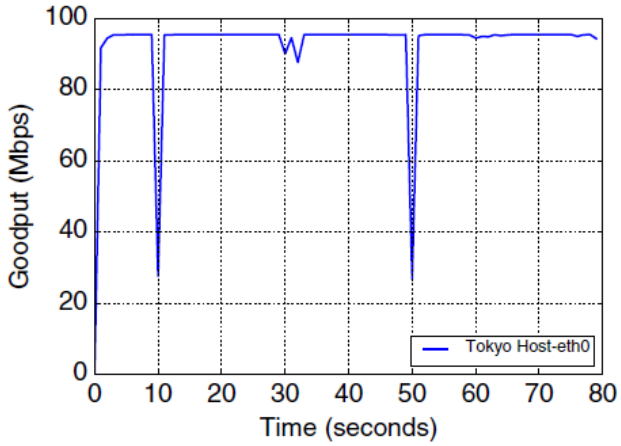
C. *Výstup testovania*

Výstup automatického testovania je ukladaný do súborov v adresári */mininet-scripts/logs/*. Po ukončení testovania sa vygenerujú tri samostatné *.csv súbory obsahujúce výsledky monitorovania priepustnosti príslušných spojení. Na základe týchto súborov následne môžeme vytvoriť graf priepustnosti na jednotlivých linkách. Z vygenerovaných súborov je potrebné odstrániť text „total“ aby sa výsledky dali ľahšie spracovať do podoby grafu.

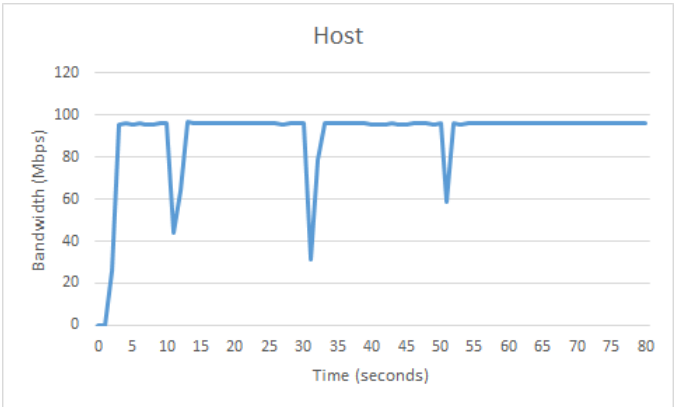
VI. VÝSLEDKY MERANÍ

Testovanie sme vykonávali viac-násobne. Výsledky jedného z testovaní sme previedli do grafov znázornených nižšie.

Na obrázku Obr. 3 vidíme výsledky simulácie autorov zobrazené v grafe. Jedná sa priepustnosť medzi zariadeniami ***Tokyo Host*** a ***Tokyo DC1*** na obrázku Obr. 1. Na obrázku Obr. 4 vidíme priepustnosť spojenia medzi rovnakými zariadeniami, ale jedná sa o výsledky našej simulácie.

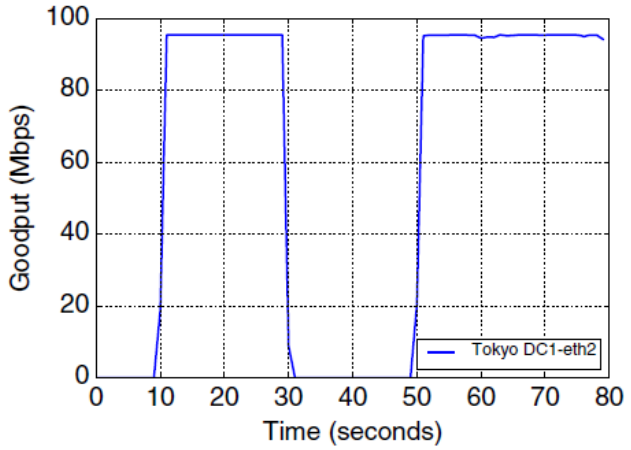


Obr. 3 Výsledky simulácie autorov článku (konconé zariadenie)

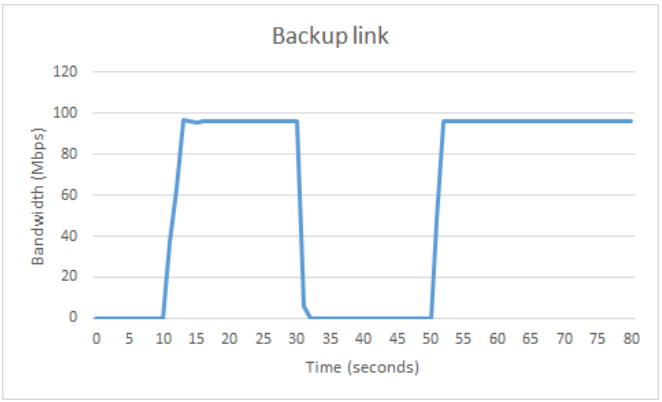


Obr. 4 Výsledok našej simulácie (koncové zariadenie)

Obrázok Obr. 5 predstavuje priepustnosť spojenia na záložnej linke. Konkrétne medzi zariadeniami ***Tokyo DC1*** a ***Tokyo DC2*** podľa obrázku Obr. 1. Na obrázku Obr. 6 sú výsledky z monitorovania priepustnosti v našej simulácií.

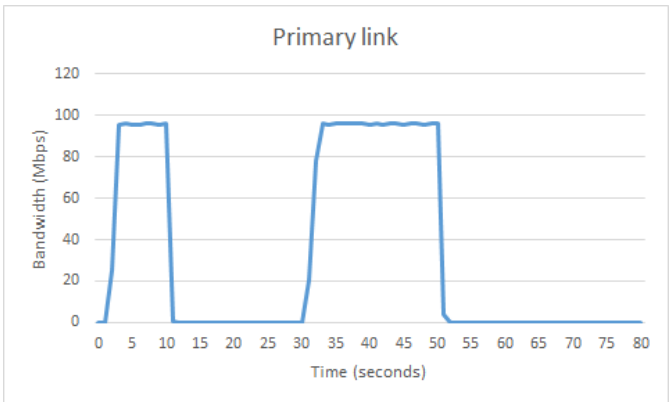


Obr. 5 Výsledky simulácie autorov článku (záložná linka)



Obr. 6 Výsledok našej simulácie (záložná linka)

Na Obr. 7 vidíme priepustnosť na hlavnej komunikačnej linke. Jedná sa o spojenie medzi zariadeniami ***Tokyo DC1*** a ***Tsukuba DC*** podľa obrázku Obr. 1.



Obr. 7 Výsledky našej simulácie (hlavná komunikačná linka)

VII. ZÁVER

Cieľom tejto práce bolo porovnať výsledky simulácie autorov článku s našimi výsledkami. Ako môžeme v predchádzajúcej kapitole vidieť výsledky našej simulácie sú veľmi podobné výsledkom simulácie autorov článku. Najväčší rozdiel vo výsledkoch je dobré viditeľný na obrázkoch Obr. 3 a Obr. 4. V 30. sekunde je obnovená prerušená linka a pri komunikácii sa začne používať pôvodná, preferovaná cesta. Autori článku v danom čase zaznamenali menšie zníženie priepustnosti spojenia, v našom prípade bola priepustnosť výraznejšie ovplyvnená. Tento menší rozdiel môže byť spôsobený použitím rozdielneho kontroléra, nakoľko autori použili POX kontrolér a my sme sa rozhodli pre RYU kontrolér.

LITERATÚRA

[1] NGUYEN, Kien a Shigeki YAMADA. An experimental feasibility study on applying SDN technology to disaster-resilient wide area networks. *Annals of Telecommunications* [online]. 2016, **71**(11-12), 639-647 [cit. 2017-11-28]. DOI: 10.1007/s12243-016-0502-2. ISSN 0003-4347. Dostupné z: http://link.springer.com/10.1007/s12243-016-0502-2