Emulation of Dynamic Adaptive Streaming over HTTP with Mininet

Maroš Hrobák

Slovenská Technická Univerzita, Fakulta Informatiky a Informačných Technológií

Bratislava, Slovensko

xhrobak@is.stuba.sk

Matúš Kováč

Slovenská Technická Univerzita, Fakulta Informatiky a Informačných Technológií

Bratislava, Slovensko

xkovacm8@is.stuba.sk

***Abstrakt* — Video streaming sa v dnešnej dobe stáva viac a viac populárnym. Dynamické adaptívne streamovanie cez HTTP doručuje mediálny obsah používateľovi vždy s najvyšším možným bitratom cez sieť s rozličnými hodnotami bandwidthu. Tento článok sa zaoberá dodávaním MPEG-DASH obsahu cez virtuálne emulačné prostredie Mininet a reálnu sieť. V článku porovnávame naše namerané výsledky s výsledkami dosiahnutými v hlavnom článku[6] . Navyše chceme dokázať univerzálnosť Mininetu, tým, že sme si zvolili iné prvky siete a potvrdiť, že toto prostredie môže slúžiť ako podklad pre návrh nových algoritmov dynamického adaptívneho streamovania.**

Kľúčové slová — Dynamic Adaptive Streaming, Mininet, video, SDN

I. ÚVOD

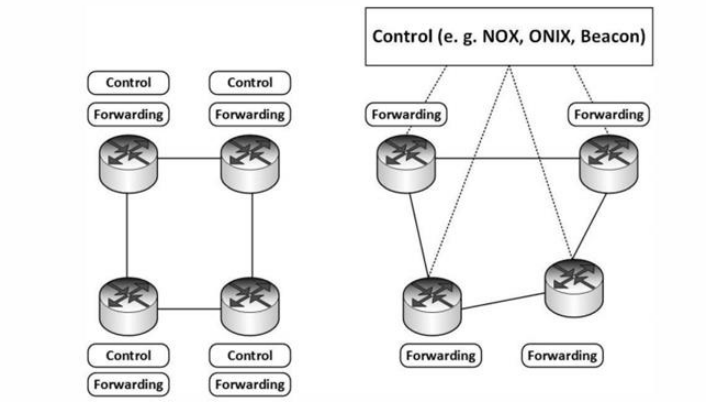
„*Technológie pre streamovanie videa sa nepretržite rozvíjajú a to najmä vďaka potrebe držať krok so stále rastúcim množstvom video obsahu na internete. Streamovanie videa cez protokol HTTP sa stalo široko rozšíreným. Mnohé veľké korporácie ako Adobe, Microsoft alebo Apple, už dlhšiu dobu využívajú svoje vlastné systémy na dodanie dát cez protokol HTTP ako HTTP Dynamic Streaming, Smooth Streaming alebo HTTP Live streaming. Všetky tieto technológie spadajú do kategórie Adaptive bitrate streaming, ktorá umožňuje dynamickú zmenu kvality videa, vzhľadom na aktuálny bitrate klienta. Rozhodovanie o zmene kvality majú na starosti viaceré algoritmy. Dynamické adaptívne streamovanie cez HTTP, známe tiež ako MPEG-DASH sa stalo štandardom v roku 2012 a používa sa hlavne v live streamingových systémoch, ako sú YouTube, Netflix, Twitch*.“ [6]

V kapitole II. sa nachádza analýza softvérovo definovaných sietí, technológie dynamického adaptívneho streamovania a potrebných náležitostí pre vypracovanie tohto článku. Zvyšok článku sa zaoberá návrhom a implementáciou experimentu na potvrdenie univerzálnosti emulačného prostredia Mininet. Kapitoly V. a VI ponúkajú detailný pohľad na testovanie a zhodnotenie výsledných zistení.

II. ANALÝZA

A. *Softvérovo definované siete*

Inicializovať, kontrolovať, meniť a manažovať správanie siete dynamicky cez rozhranie.



Obrázok 1. Tradičná sieť (vľavo) a SDN (vpravo) [6]

Tradičné hardvérové siete nezodpovedajú neustále sa meniacim potrebám v oblasti výpočtovej techniky a úložiska v prostrediach dátových centier a poskytovateľov služieb. SDN (Software-Defined Networking) poskytuje lepšie možnosti v takých situáciách, kde mnohé vlastnosti vyžadujú flexibilnejší a dynamickejší prístup. Vlastnosti SDN sú [10]:

- Priamo programovateľná – sieťová kontrola je priamo programovateľná pretože je oddelená od smerovacích funkcií
- Agilná – oddelenie kontroly od smerovania umožňuje administrátorom dynamicky prispôsobovať sieťovú premávku aby spĺňala meniace sa potreby
- Centrálne manažovateľná – sieťová inteligencia je centralizovaná v SDN kontroléroch, ktoré udržujú celkový pohľad na sieť.

- Programovateľná – umožňuje rýchlo meniť konfiguráciu, manažovať, zabezpečiť alebo optimalizovať sieť rýchlo cez automatické SDN programy, ktoré si môžu ľudia písať sami, pretože nie sú proprietárne

Oproti tradičnej sieti je SDN sieť riadená kontrolérom, ktorý má niekoľko rovín. Medzi základné roviny patria [10]:

- Riadiaca rovina ovláda to kam sa budú posielať správy
- Dátová rovina už len posiela podľa toho, čo mu riadiaca rovina povie

B. Prostredie Mininet

Mininet je sieťový emulátor, ktorý umožňuje vytvárať koncové zariadenia, prepínače, smerovače, a linky medzi nim. Mininet host sa správa rovnako ako reálna mašina a je možné sa naň pripojiť pomocou SSH. Čo sa týka OpenFlow kontrolérov, Mininet je veľmi flexibilný a umožňuje pridať do simulácie množstvo typov kontrolérov. Siete Mininetu pracujú so skutočným kódom vrátane štandardných sieťových aplikácií Unix / Linux, ako aj na skutočnom jadre Linuxu a sieťovom zásobníku. Hlavné funkcie nástroja Mininet sú [1]:

- poskytuje jednoduché a lacné sieťové testovacie médium pre vývoj aplikácií nad protokolom OpenFlow
- umožňuje viacerým súbežným vývojárom pracovať nezávisle na tej istej topológii
- podporuje regresné testy na úrovni systému, ktoré sú opakovateľné a ľahko zabalené
- umožňuje zložité testovanie topológie bez nutnosti prepojenia fyzickej siete
- podporuje ľubovoľné vlastné topológie a obsahuje základnú sadu parametrizovaných topológií
- je použiteľný mimo krabice bez programovania a tiež poskytuje priamu a rozšíriteľnú Python API pre vytváranie a experimentovanie so sieťami
- služba Mininet poskytuje jednoduchý spôsob, ako dosiahnuť správne správanie systému (a v rozsahu podporovanom vaším hardvérom, výkonom) a experimentovať s topológiami.

Mininet kombinuje mnohé z najlepších funkcií emulátorov a špecializovaných simulátorov. V porovnaní s prístupmi založenými na úplnej virtualizácii systému Mininet vyniká v [1]:

- rýchlejšom bootovaní
- umožňuje spravovať stovky hostiteľov a prepínačov
- poskytuje väčšiu šírku pásma
- Jednoducho sa inštaluje

V porovnaní s hardvérovými testovacími doskami, Mininet vyniká v [1]:

- je rýchlo rekonfigurovateľný a reštartovateľný

V porovnaní so simulátormi, Mininet vyniká v [1]:

- beží na reálnom, nemodifikovanom kóde vrátane aplikačného kódu
- ľahko sa pripája k reálnym sieťam

C. Kontrolér

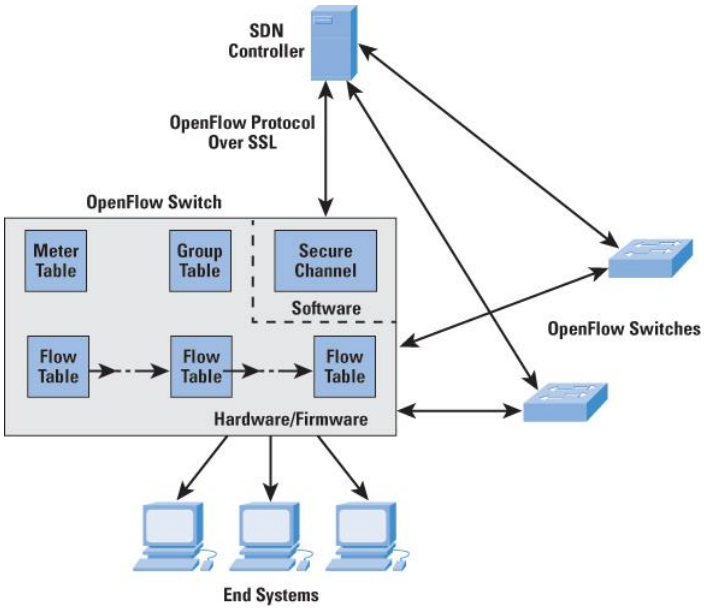
OpenFlow kontrolér je typ SDN kontroléra, ktorý používa OpenFlow protokol. OpenFlow kontrolér používa OpenFlow protokol aby spojil a konfiguroval sieťové zariadenia, ako smerovače a prepínače, na nájdenie najlepšej cesty v premávke. SDN kontroléry teda uľahčujú riadenie siete a dokážu riadiť celú komunikáciu medzi aplikáciami a zariadeniami, tak aby sa čo najefektívnejšie upravil tok premávky, tak ako je v danom momente potrebné. Keď nie je riadiaca rovina implementovaná vo firmvéri, ale je implementovaná v softvéri, administrátori dokážu manažovať sieť oveľa jednoduchšie, viac dynamicky a s lepšou granularitou. OpenFlow kontrolér teda tvorí centrálnu kontrolnú jednotku v sieti, ktorá riadi všetku premávku, všetky zariadenia v sieti vykonávajú akcie tak ako od nich požaduje kontrolér a podporuje OpenFlow. [5]

Ryu poskytuje veľmi silný pomer medzi jeho výhodami a nevýhodami, v prospech výhod. Ryu si stavia medzi jej hlavne piliere výhod tri charakteristiky: kvalita kódu, funkcionalita a použiteľnosť. Podporuje niekoľko protokolov, pre správu sieťových zariadení, tými sú napr. Netconf, OF-config, no primárne OpenFlow vo verziách až po aktuálnu 1.5. Je napísaný v jazyku Python. [4]

D. Protokol Openflow

Je protokol medzi kontrolérom a sieťovými zariadeniami, ako aj nástroj pre špecifikáciu logickej štruktúry siete. Pracuje v rámci TCP (Transmission Control Protocol), kde by mal počúvať na porte 6653. Môže vykonať rôzne akcie [2]:

- pridávanie, zmena alebo vyraďovanie paketov, podľa vopred definovaných pravidiel a akcií
- smerovanie akceptovaných paketov prepínačom
- neakceptované pakety sú smerované do kontroléra, ktorý môže:
    - zmeniť pravidlá smerovacej tabuľky na jednom alebo viacerých prepínačoch
    - nastaviť nové pravidlá, aby tak predišiel veľkej komunikácií medzi prepínačom a kontrolérom



Obrázok 2. Openflow prepínač [2]

Obrázok popisuje približnú komunikáciu v rámci OpenFlow protokolu [2]:

- SDN kontrolér komunikuje s prepínačmi, ktoré sú kompatibilné s OpenFlow, pomocou OpenFlow protokolu bežiaceho cez SSL (Secure Sockets Layer)
- každý prepínač sa pripojí k zariadeniam cieľového používateľa, ktoré sú zdrojmi a cieľmi paketových tokov
- každý prepínač má niekoľko tabuliek, implementovaných hardvérom alebo firmvérom, ktoré sa používajú na riadenie toku cez prepínače

E. Prepínač

Preposielacie zariadenie v SDN sa nazýva forwarder a pozostáva z dvoch častí [3]:

- prietokovej tabuľky (flow table) obsahujúcej záznam a akciu na prijímanie aktívnych tokov
- abstrakčnej vrstvy, ktorá bezpečne komunikuje s riadiacou jednotkou (kontrolérom) o nových vstupoch, ktoré sa v danom momente nenachádzajú v tokovej tabuľke.

OpenFlow prepínač pozostáva z jednej alebo viacerých tabuliek tokov, ktoré ukladajú záznamy pre vyhľadanie alebo presmerovanie paketov. [3]

Po príchode paketov na OpenFlow prepínač sa hlavička paketu extrahuje a porovná s políčkom zhody (match field) z tabuľky tokov. Ak sa zhoduje hlavička paketu s políčkom v tabuľke, tak prepínač použije príslušnú sadu inštrukcií spojenú s daným tokom. V prípade, že sa nenájde žiadna zhoda s tabuľkou tokov, tak nasledujúca akcia prepínača bude závisieť od inštrukcií definovaných v tabuľke chýbajúcich tokov (table-miss flow entry), napr. zahodenie paketu. [3]

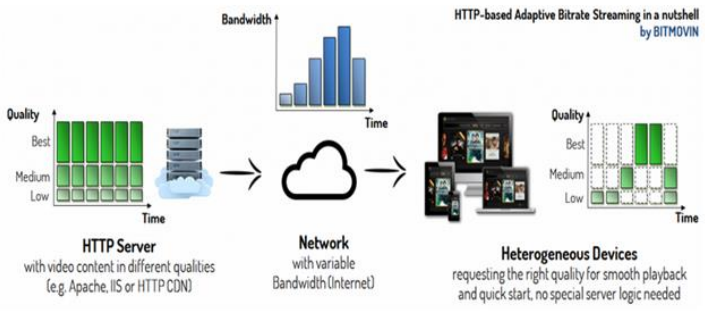
F. Dynamické adaptívne streamovanie

Hlavná myšlienka Dynamic adaptive streaming (MPEG-DASH) je rozdeliť súbor s mediálnym obsahom na segmenty, ktoré môžu byť enkódované na rôznych bitratoch. Segmenty sú poskytnuté na webovom servery a môžu byť stiahnuté cez HTTP (GET request). Segmenty sú referencované pomocou URL (definované v RFC3986). Segmenty sa ďalej môžu deliť aj na menšie subsegmenty. Možné obsahy segmentov [8] [9]:

- SegmentBase - najzákladnejšia reprezentácia
- SegmentList – obsahuje zoznam SegmentURL
- SegmentTemplate – vyskladáva zoznam segmentov

Na vysvetlenie vzťahov medzi segmentami, MPEG-DASH zadefinovalo Media Presentation Description (MPD). MPD je XML súbor, ktorý reprezentuje rôzne vlastnosti mediálneho obsahu a jednotlivé segmenty pre každú vlastnosť. Táto štruktúra prezentuje spojenie medzi segmentami a bitratom. [8] [9]

Na obrázku je možné vidieť princíp MPEG-DASH. Server rozdelí mediálny obsah na segmenty podľa kvality na Low, Medium a Best. Klient podľa aktuálneho bandwidthu a CPU kapacity vyberá medzi kvalitou mediálneho obsahu. [8] [9]



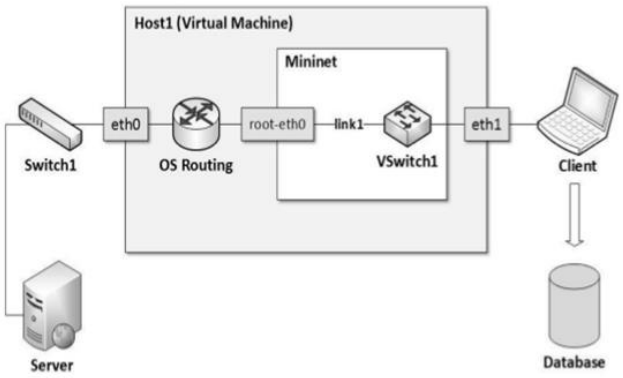
Obrázok 3. Princíp fungovania MPEG-DASH [8]

III. NÁVRH

A. Topológia

V našom prevedení budeme používať podobnú topológiu, aká bola uvedená v článku. Bude obsahovať server, virutálny switch, klientské PC, PC s VM v ktorej bude bežať Mininet. Pre jednoduchosť riešenia bude použitý jeden rozsah IP adries:

- adresy v rozsahu-192.168.1.0/24
- adresy v rozsahu-192.168.2.0/24



Obrázok 4. Návrh topológie [6]

B. Technológie

Porovnanie použitých technológií v článku a návrhu našich technológií:

Tabuľka 1. Porovnanie technológií

Použitá technológia v článku	**Naša navrhnutá technológia**
Server Apache HTTP	Server Nginx
Klientský PC (Win 8, I7, 8GB RAM)	Klientský PC (Linux Ubuntu, I5, 8GB RAM)
Prehrávač - Bitmovin	Prehrávač – VLC
Kontrélér - nespomenutý	Kontrolér - Ryu
Codec - nespomenutý	Codec - H.264
Virtuálny switch - OpenVSwitch	Virtuálny switch - OpenVSwitch
Mininet verzie 2.2.1	Mininet verzie 2.2.2
Databáza - MySQL	Databáza - PostgreSQL

Rozhodli sme sa od článku odlíšiť inou voľbou HTTP servera, prehrávača, klientského PC, voľbou vlastného kontroléra, codecom, verziou Mininetu a inou databázou. V prípade zistenia vzájomnej nekompatibility alebo nekompatibility s technikou MPEG-DASH skúsime použiť iné kompatibilné zariadenia alebo technológie. Tým by sme chceli dokázať univerzálnosť prostredia Mininet aj v novšej verzií s použitím iných technológií.

C. Princíp testovania

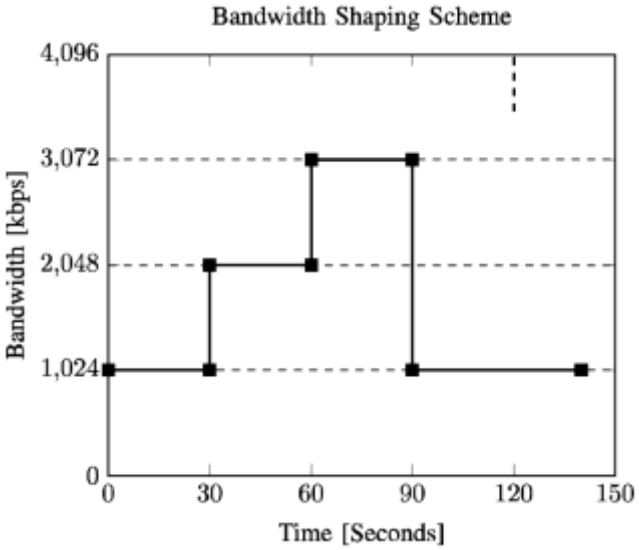
Testovať budeme podobným princípom, ako je použitý v článku. Video v codecu H.264 bude streamované pomocou VLC media player s využitím techniky MPEG-DASH. Bandwidth bude menený cez framework Mininetu minievents, ktorý umožňuje meniť bandwidth a iné vlastnosti dynamicky.

Na nasledujúcom obrázku môžeme vidieť ukážku JSONu, ktorý bude meniť bandwidth na linke medzi virtuálnym switchom s1 a s2.

```
{
    "time": 0,
    "type": "editLink",
    "params": {
      "src": "s1",
      "dst": "s2",
      "bw": 1
    }
},
{
    "time": 31,
    "type": "editLink",
    "params": {
      "src": "s1",
      "dst": "s2",
      "bw": 2
    }
},
{
    "time": 61,
    "type": "editLink",
    "params": {
      "src": "s1",
      "dst": "s2",
      "bw": 3
    }
},
{
    "time": 91,
    "type": "editLink",
    "params": {
      "src": "s1",
      "dst": "s2",
      "bw": 1
    }
}
```

Obrázok 5. Ukážka JSONu [6]

Každý test bude trvať 120 sekúnd a každých 30 sekúnd budeme na linke meniť bandwidth na hodnoty 1024 kb/s, 2048 kb/s a 3072 kb/s, podľa nasledujúceho grafu.



Obrázok 6. Graf zmeny bandwidthu v čase [6]

Budeme sledovať zmeny hodnoty bitratu v plynúcom čase.

Bude vykonaných 20 testov s nasledujúcimi parametrami:

- dĺžka testu - 120 sekúnd
- zmena bandwidthu - každých 30 sekúnd
- hodnoty bandwidthu - 1024 kb/s, 2048 kb/s, 3072 kb/s, 1024 kb/s
- video s codecom H.264

Všetky testy budú následne vyhodnotené a porovnané s výsledkami z článku. Prípadné zhody alebo rozdiely budú zdokumentované.

IV. Implementácia

Pre voľbu serveru sme sa rozhodli pre server Nginx a nie pre Apache. Oba tieto servery sú veľmi podobné čo sa týka dokumentácie, výkonu, škálovateľnosti či bezpečnosti. Avšak rozhodli sme sa pre Nginx kvôli tomu, že je modernejší, prišiel nám ľahší na konfiguráciu a nie je taký komerčný ako Apache server. Server je nainštalovaný na počítači pod OS Linux. Zo servera sme vysielali dva druhy streamu MPEG DASH a HSL, ktoré sú si veľmi podobné. Stream mal kvalitu low a hd všetky rozoznateľné voľným okom.

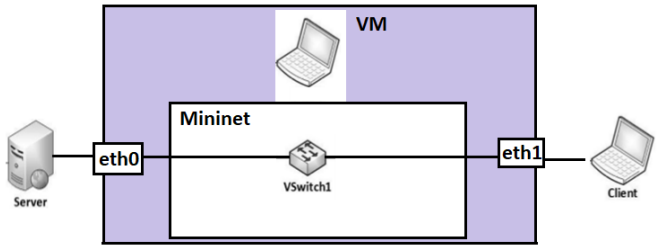
Klientský PC s OS Linux, procesorom I5 a 8GB RAM prijíma stream cez prehrávač VLC. VLC prehrávač má neoficiálnu verziu 3.0.0. (build 12.8.2017), ktorá podporuje MPEG-DASH streaming. Ako druhú alternatívu, sme stream sledovali pomocou programu ffplay, v ktorom sme však neboli schopní sledovať aktuálny bitrate videa. Prehrávač ffplay taktiež nie je schopný dynamicky meniť kvalitu videa. Preto sme ho používali iba na kontrolné testy.

Mininet verzie 2.2.2. beží vo VM na strednom počítači s dvoma sieťovými rozhraniami. Na jedno rozhranie je pripojený server a na druhé klient. V Mininete je možnosť spustiť dve topológie, v jednej sa nachádza iba jeden virtuálny switch a kontrolér, v druhej sa nachádzajú dva virtuálne switche a kontrolér. Zámerom vytvorenia dvoch topológií bolo lepšie porovnávanie získaných výsledkov. V prípade topológie 1, sa bandwidth mení na klientskom počítači pomocou skriptu dynamicky. V prípade dvoch switchov v mininete sa bandwidth upravuje na linke v mininete medzi týmito dvoma switchmi dynamicky cez minievents spomínaný vyššie v článku, táto verzia je viac podobná článku. Obe rozhrania sú namapované do VM ako bridged, sú v promiskuitnom móde a sú pripojené káblovo.

IP adresy sú pridelené nasledovne:

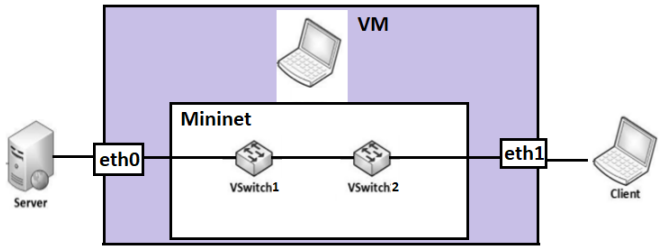
- Klient 192.168.1.1/24
- eth1 192.168.1.2/24
- eth0 192.168.1.3/24
- Server 192.168.1.4/24

Výsledná implementovaná topológia 1 vyzerá nasledovne:



Obrázok 7. Implementovaná topológia 1

Výsledná implementovaná topológia 2 vyzerá nasledovne:



Obrázok 8. Implementovaná topológia 2

Rozhodli sme sa využiť len jednu podsieť, pretože funkčnosť streamu nezávisí od viacerých sietí. Reálny switch sme taktiež nepoužili pretože sme žiadny nemali k dispozícii. Nahradili sme ho však druhým virtuálnym switchom v Mininete v prípade topológie 2, aby sme sa viac priblížili podmienkam z pôvodného článku.

Bandwidth na linke medzi klientom a virtuálnym switchom sa mení pomocou skriptu na klienstkom PC každých 30 sekúnd na dané hodnoty rovnako ako v článku. Bandwidth v prípade topológie č. 2, sme menili pomocou frameworku MiniEvents, ktorý nám umožnil dynamickú zmenu bandwidthu na linke v Mininete. Bitrate v plynúcom čase sme sledovali z prehrávača VLC a hodnoty sme zapisovali. Výsledné hodnoty sme nasledovne porovnávali s výsledkami z hlavného článku[6].

V. Testovanie

Komunikáciu v sieti sme vždy testovali pingom z klientského počítača na server. Testovanie sa vždy začalo spustením streamu zo servera, následne sme vo VLC alebo ffplay poslali požiadavku na získanie streamu zadaním cesty:

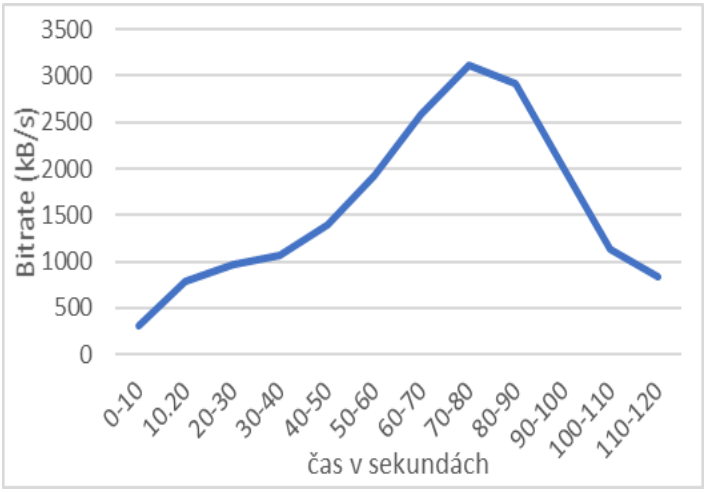
http://192.168.1.4 /dash/xxx2.mpd

http://192.168.1.4/hls/xxx2.m3u8

Celkovo sme vykonali 20 testov po 120sekúnd na oboch topológiách a technikách streamovania (DASH, HLS). Všetky hodnoty bitratu, ktoré nám dovoloval prehrávač VLC zaznamenať, so zmenami bandwidthu sme zaznamenávali. Výsledky môžete vidieť v tabuľke č.2. Pri menení bandwidthu na klientskom rozhraní za použití topológie č. 1 (obrázok 7) boli výsledky nemerateľné, kvôli beta verzii prehrávača VLC, ktorý často vypadával a refreshoval stream. Výsledky obsiahnuté v tabuľke sú merané pri topológii č.2 (obrázok 8), za použitia vyššie spomínaného frameworku MiniEvents.

Tabuľka 2. Priemerné bitrate

Čas [s]	Bitrate [Kb/s]	Bandwidth [Kb/s]
0-10	312	1024
10-20	786	1024
20-30	974	1024
30-40	1076	2048
40-50	1401	2048
50-60	1922	2048
60-70	2580	3076
70-80	3112	3076
80-90	2923	3076
90-100	2001	1024
100-110	1132	1024
110-120	840	1024



Obrázok 9 Hodnoty bitratu s plynúcim časom

V porovnaním s hodnotami z článku sú naše hodnoty podobné. Na tabuľke č.2 môžeme vidieť zmeny hodnôt bitratu v plynúcom čase. Hodnoty ukázali, že schopnosť cachingu v prípade zmien bandwidthu mala za následok neaktuálnu hodnotu bitratu v nasledujúcich 10 sekundách. Až po 10 sekundách sa hodnoty bitratu dostali do rozmedzia pod nastavenú hodnotu bandwidthu na linke.

VI. Zhodnotenie

Po implementácií a testovaní riešenia potvrdzujeme schopnosť Mininetu doručovať MPEG DASH a HSL obsah. Tiež je Mininet vhodným nástrojom na simulovanie sietí a je dobre prepojiteľný aj s reálnymi zariadeniami. Mininet sa ukázal aj ako univerzálny nástroj pretože dokáže korektne pracovať aj s využitím rozličných zariadení v sieti.

Chceli by sme však poukázať najmä na prehrávač VLC, ktorý nie je vhodným nástrojom na monitorovanie hodnôt videa a tiež jeho funkčnosť pri dopytovaní sa na stream v sieti bola často oneskorená a neaktuálna. Prehrávač ffplay fungoval oveľa lepšie a neboli s ním problémy, mohli sme v ňom dynamicky ručne meniť kvalitu videa pomocou kláves ‚c' a ‚v'. Nemohli sme však v ňom sledovať bit rate.

Prehrávač VLC nie je vhodným nástrojom na sledovanie dynamického adaptívneho streamu, kvôli vnútornému cachingu, ku ktorému nie je uvedenej veľa dokumentácie. Veľa krát streamované video vo VLC odrážalo kvalitu v závislosti od hodnoty bitratu streamovaného obsahu, nie podľa bandwidthu na linke. Streamované video sa spúšťalo v náhodných časoch nezávisle od času ako streamoval server a taktiež stále pokračoval v prehrávaní aj napriek tomu, že stream už nebežal.

S Mininetom, frameworkom MiniEvents a serverom Nginx sme nemali väčšie problémy, ich konfigurácia a dokumentácia boli v kvalitnom stave.

Referencie

[1] Mininet Overview, 2017, http://mininet.org/overview/

[2] Software-Defined Networks and OpenFlow - The Internet Protocol Journal, Volume 16, No. 1, 2013, https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html

[3] Mendoca M., Astuto B., Obraczka K., Turletti T.: Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,2013

[4] Kei Ohamura, 2013, NTT, https://osrg.github.io/ryu/slides/LinuxConJapan2013.pdf

[5] SDNCentral LLC 2017, https://www.sdxcentral.com/sdn/definitions/sdn-controllers/openflow-controller/

[6] Zabrovskiy A., Kuzmin E., Petrov E., Fomichev M., Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT), 2016 18th Conference of, Petrozavodsk State University, Petrozavodsk, Russia

[7] Software Defined Networking: Design and Deployment, Patricia A. Morreale, James M. Anderson, 2014

[8] Encoding Intelligence™ 2016, https://www.encoding.com/mpeg-dash/

[9] Christopher Mueller 21.4.2015, https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/

[10] Ed Tittel, https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html