FRI-2G.302-1-CSN-11

A COMPARATIVE ANALYSIS OF SOFTWARE DEFINED NETWORKING CONTROLLERS [11]

Eng. Jordan Raychev
Department of Telecommunications,
"Angel Kanchev" University of Ruse
Tel.: 082 888 353 / +359884979934
E-mail: jraychev@uni-ruse.bg

Abstract: *Software defined networking (SDN) is an emerging paradigm that plays an important role into networks within the networking research field. The SDN technology provides a possibility to decouple the control plane from the physical plane of networking devices, while traditional telecommunication equipment encompasses both the control and data planes into a single device. All of the decoupled control function are implemented in a single centralized server called a SDN controller, making it the only device that could control the network as a whole entity. The centralized management approach has proven its benefits over the past several years. On good example about that is the server virtualization technology. Like the SDN technology, the server virtualization allows server administrators to manage tens even hundred virtual machines from a single instance called a hypervisor.*

The objective of this paper is to show how the two technologies can co-exists in a single environment and how they can benefits from each other. In an addition to that a comparative analysis is going to be conducted in order to show how different SDN instances (controllers) handle the network traffic and how network resources are affected by that. In order to achieve the objective, a virtual test laboratory is going to be built on top of type one hypervisor. The virtual test laboratory is going to be comprised of several components – a management platform that serves as a single point of management, a control plane (a virtual instance of a SDN controller that has the ability to adapt to changes in network behavior) and a data plane (a virtual representation of a software-defined networking data plane). The objective of the comparative analysis is to show how different SDN controller behave under various types of load – change in traffic volume by a single device, change in traffic volume from multiple sources and so on.

Keywords: *Comparative SDN analysis, Network virtualization, Server virtualization, Software-defined networking*

ВЪВЕДЕНИЕ

Софтуерно дефинираните (СД) мрежи представлява ново появила се парадигма, която играе важна роля в сферата на телекомуникационните мрежи от ново поколение. СД технологията създава възможността контролните функции на мрежовите устройства в мрежите от ново поколение да бъдат прехвърлени към централизиран управляващ сървър, наричан софтуерно дефиниран контролер или накратко само контролер [1]. Посредством този метод на отделяне на контролната от физическата равнина се създава възможност мрежите от ново поколение да бъдат управлявани като обща единица независимо от мрежовите устройства, които изграждат физическата равнина. Централизираното управление на ресурси е доказало своите предимства много преди появата на софтуерно дефинираните мрежи. Подобна технология, която извлича ползите от централизираното управление на ресурсите е сървърната виртуализация. Така, както СД мрежите и технологията за сървърна виртуализация позволява множество виртуални машини да бъдат управлявани през централизиран портал познат още като хипервайзор.

В настоящият доклад ще бъде представена методика, която показва как тези две технологии могат да си взаимодействат и да извлекат предимствата една от друга. Посредством изградената СД платформа ще бъде направен и сравнителен анализ между най-утвърдените СД контролерите. Целта на сравнителният анализ е да предостави информация за това как различните СД контролери управляват трафика и колко пълноценно се използват мрежовите ресурси от тези контролери. За постигане на горе посочената цел ще бъде създадена виртуална лаборатория за тестове, чиято основа ще бъде изградена върху

[11]

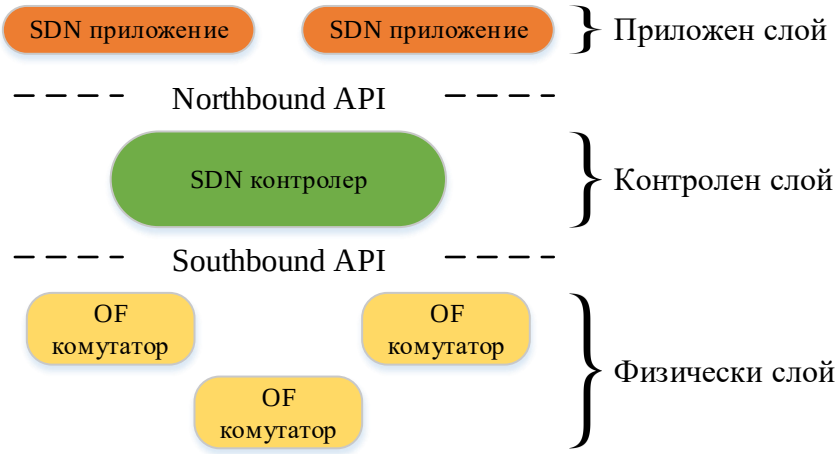
хипервайзор от първи тип (хипервайзора, който ще бъде използван е VMWare ESXi версия 6.5). Виртуалната лаборатория ще бъде съставена от няколко основни компонента – слой за управление чрез, който администратора ще има възможност да контролира както СД контролери, така и устройствата, които те управляват, контролна равнина, която представлява виртуална инстанция на всеки един от изследваните СД контролери и физическа равнина емулираща реална телекомуникационна мрежа.

ИЗЛОЖЕНИЕ

Анализ на структурата и функционалностите на софтуерно дефинираните мрежи

Както вече бе споменато, софтуерно дефинираните мрежи представялват новопоявила се парадигма в сферата на телекомуникационните мрежи, чиято основна цел е да предостави нови възможност за управление на непрекъснато увеличаващия се мрежови трафик. Основният недостатък на традиционните телекомуникационни мрежи се изразява в това, че те не притежават необходимите функционалности да покрият нуждите на съвременните приложения и непрекъснато увеличаващия се брой техни потребители. Именно това налага преосмисляне и въвеждане на промяна в телекомуникационните мрежи и съответно появата на софтуерно дефинираната технология [2].

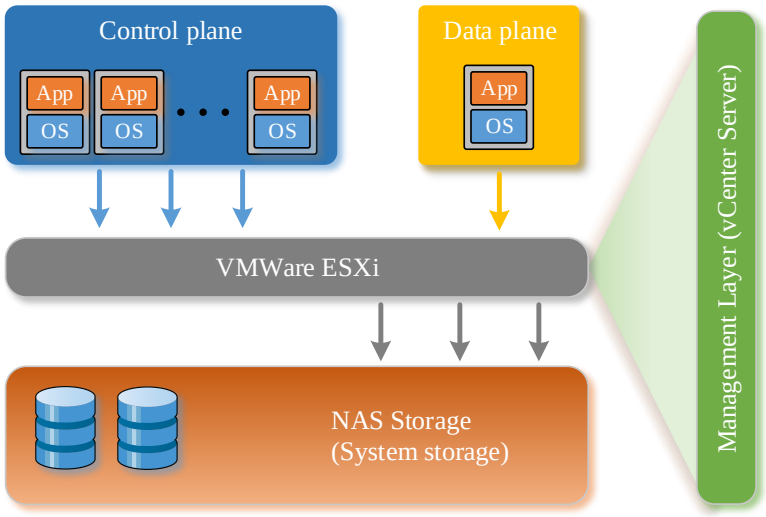
Софтуерно дефинираните мрежи се появяват за пръв път в университетската среда, като тяхната цел е тестването и внедряването на нови мрежови функционалности традиционните компютърни мрежи. Внедряването на нови функции се оказва изключително сложен и трудоемък процес, който в някой случаи се оказва дори невъзможен. Именно това поражда и появата на първите мрежи от ново поколение. Разликата между традиционните компютърни мрежи и ново появилите се СД мрежи се състои в това, че контролните функции на СД мрежите биват изнесени към специализиран сървър (най-често физически или логически централизиран), нарина още СД контролер. Неговата функционалност е от огромно значение, тъй като в сферата на СД мрежите, контролера е единственото устройство, което има поглед върху физическата топология на мрежата и отговаря за правилното комутиране на мрежовия трафик през физическата равнина. Изнасянето на контролните функции към централизиран сървър, създава предпоставка за поява на логическо разделяне на мрежите от ново поколение, като най-общо те биват разделяни на два основни слоя – контролен слой (контролна равнина) в която се намира СД контролера/ите и физическа равнина, която бива изградена от мрежови комутатори. Важна особеност тук е, че ново появилите се устройства изграждащи физическия слой много често притежават минимални контролни функции, а много често са напълно не функциониращи без да бъдат съпровождани от съответния управляващ ги контролер. Архитектурата на СД мрежи [3] е допълнена от два допълнителни слоя – приложен слой, който се състои от набор от СД приложения притежаващи контрол върху част от мрежовите ресурси и слой за управление, чиято основна цел е да координира мрежовите ресурси между отделните мрежови елементи. Основната архитектура на СД мрежите е представена на фиг. 1. Освен четирите основни слоя за които бе споменато по-горе, архитектурата на СД мрежи добавя и два допълнителни елемента, а именно приложно програмните интерфейси свързващи отделните слоеве. Първият приложно програмен интерфейс (Southbound API – Application Programming Interface) е т.нар OpenFlow (OF) интерфейс [4], който представлява основния комуникационен канал между всеки OF комутатор и управляващия го контролер. Посредством този канал, управляващия контролер управлява съответните комутатори, които са в неговата област и получава редица известия от страна на комутатора. Другият приложно програмен интерфейс (Northbound API), както се вижда от фиг. 1., е връзката между СД контролер и съответстващите му приложения. Тази връзка често бива илюстрирана единствено с цел по-добре онагледяване на отделните слоеве изграждащи СД мрежи, т.е. СД контролера и неговите приложения могат да бъдат разгледани като обща единица.



Фиг. 1. Опростена архитектура на софтуерно дефинирана мрежа

Методика за изграждане на виртуална лаборатория за изследване на софтуерно дефинираните мрежи посредством виртуализационни инструменти

Анализът на архитектурата на софтуерно дефинираните мрежи показва, че те основно биват изградени от два слоя – контролен слой и слой за достъп. СД контролер намиращ се в контролната равнина най-често бива реализиран като свободен софтуерен инстурмент, което позволява инсталирането му на устройства с x86 архитектура. Някой от по-известните контролери са - Floodlight [5], Open Daylight [6], ONOS (Open Network Operating System) [7], Ryu [8], Beacon [9] и други. Всеки контролер се характеризира със своите предимства и недостатъци относно методите за управление на мрежовия трафик, пропускателната му способност, наличието на графичен интерфейс и други. Всички тези елементи ще бъдат разгледани при извършване на сравнителния анализ между контролерите. Другият основен елемент при СД мрежи е слоя за достъп. Съществуват два основни метода за изграждане на физическата равнина при СД мрежите – посредством комутатори поддържащи OpenFlow протокола или чрез тяхното емулиране в софтуерна среда.

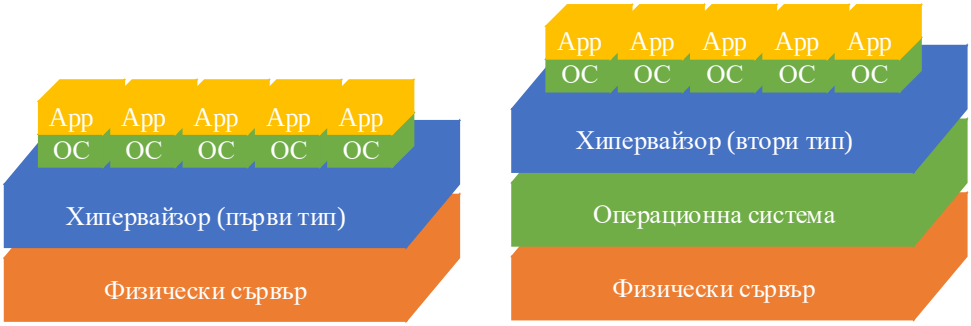


Фиг. 2. Виртуална лаборатория за изследване на СД контролери

Тъй като целта на настоящият доклад е да бъде направен сравнителен анализ между някой от най-често използваните СД контролери емулирането на физическата равнина е най-подходящия подход за постигане на поставената цел. Софтуерната среда Mininet [10] позволява да бъдат емулирани множество OpenFlow комутатори в различни по сложност топологии – от опростена линейна топология с няколко комутатора до сложни топология с йерархична структура, намиращи най-често приложение при центровете за съхранение на

данни или корпоративните мрежи. Тъй като и двата основни слоя изграждащи СД мрежите могат да бъдат реализирани посредством техния софтуерен еквивалент, това позволява те да бъдат реализирани посредством инструментите за сървъра виртуализация, която от своя страна добавя и допълнителен слой за управление на ресурсите между тях.

Архитектурата на виртуалната лаборатория за изследване на софтуерно дефинирани мрежи е представена на фиг. 2. Всеки един от изследваните контролери е реализиран на отделен виртуален сървър с цел избягване на евентуални конфликти, които биха възникнали между отделните контролери. Слоя за достъп също е реализиран като отделен виртуален сървър, които се свързва към съответния управляващ контролер по време на изследването.



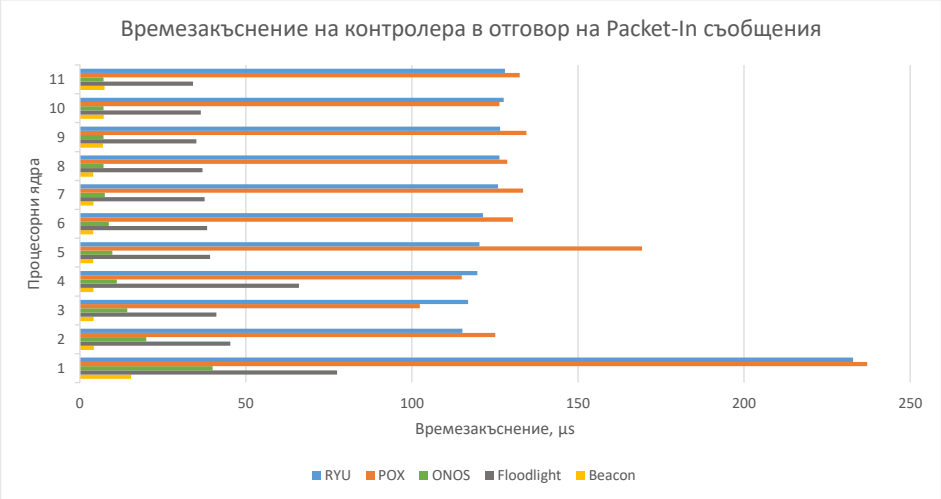
Фиг. 3. Сравнение между хипервайзор от първи тип (в ляво) и хипервайзор от втори тип (в дясно)

Средата за виртуализация на компонентите изграждащи виртуалната лаборатория представлява хипервайзор от първи тип и по-конкретно VMWare ESXi 6.5. За разлика от хипервайзорите от втори тип, които работят върху операционна система (ОС) (Windows, Linux или MacOS), е че хипервайзорите от първи тип могат да бъдат инсталирани директно върху сървъра върху хардуера на сървъра, а не върху отделна ОС. На фиг. 3 са представени разликите между двата вида хипервайзори. Ефективността и разпределянето на достъпа до хардуерните ресурси, сигурността и други само малка част от предимствата поради, които хипервайзор от първи тип е избран за реализацията на виртуалната лаборатория.

Анализ на производителността на софтуерно дефинираните контролери

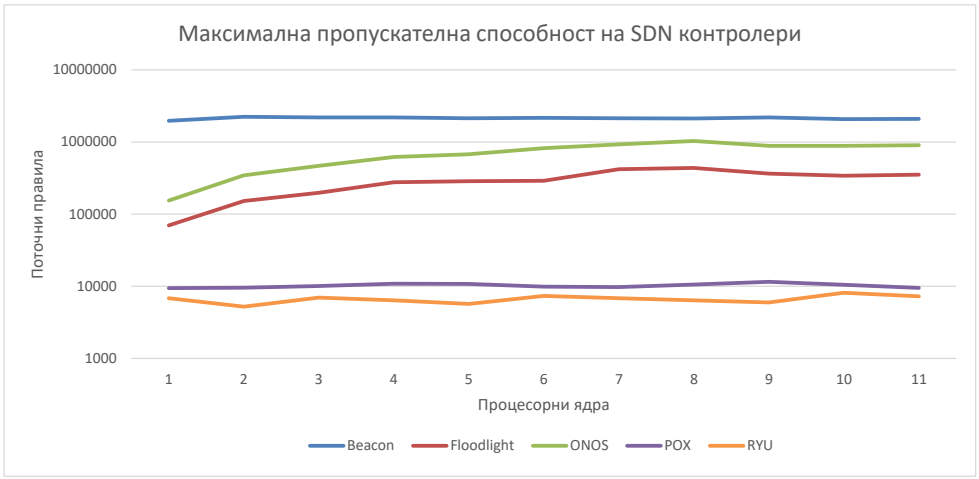
Анализът на производителността на СД контролери е осъществен на три етапа – при първия етап се изследва времезакъснението при отговор на постъпващи заявки към контролера, като в същото време се отчита и средното натоварване на процесора; втория етап от анализа е посветен върху изследване на максималната пропускателна способност на дадения контролер спрямо броя на процесорните ядра с които разполага всеки един от контролерите; третия етап от анализа бива изследвана пропускателната способност на контролерите спрямо диаметъра на физическата равнина (броя на OpenFlow комутаторите).

Както вече бе споменато по-горе, анализа на производителността на СД контролери е разделен на три основни етапа. При първият и вторият етап е необходимо да бъде анализирано времезакъснението за отговор на съответния контролер и неговата пропускателна способност. За осъществяването на тези два етапа от анализа, управляващите контролери не е задължително да бъдат физически/логически свързани към OpenFlow комутатори, тъй като е възможност входящите (Packet-In) съобщения, които се генерират от комутаторите при нормална работа на мрежата да бъдат симулирани чрез подходяш софтуерен инструмент, като Cbench [11]. Cbench дава възможност да бъдат генерирани множество на брой Packet-In съобщения (подобен тип съобщения биват генерирани от OpenFlow комутаторите, когато изпращат запитване към управляващия ги контролер) в отговор на които пропускателната способност и времезакъснението за отговор на контролера може да бъде изчислено.



Фиг. 4. Времезакъснение на контролера при отговор на Packet-In съобщения

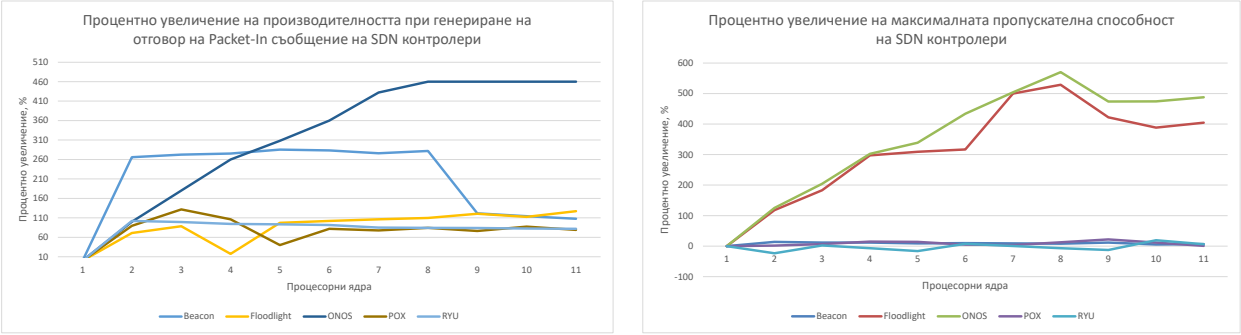
На фиг. 4 е представено времезакъснението на пет основни СД контролера в отговор на постъпващите Packet-In съобщения от слоя за достъп. От графиката се забелязва, че два от контролерите (RYU и POX) имат най-голямо времезакъснение при отговор на Packet-In заявки, което до голяма степен не зависи от броя на процесорните ядра, които се заделят за всеки от контролерите.



Фиг. 5. Максимална пропускателна способност на SDN контролери

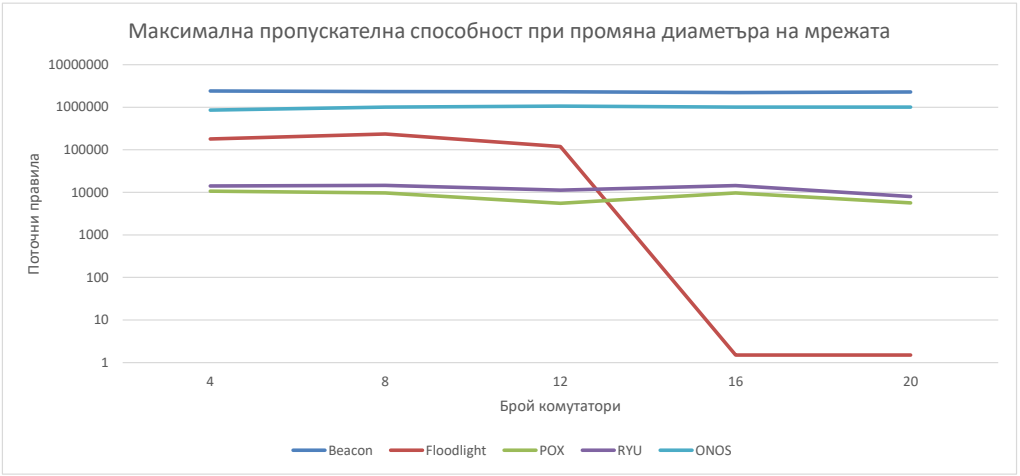
Това се дължи на факта, че и двата контролера са написани на програмен език Python [12], който не използва ефективно многонишковия модел за програмиране. В Python се използва механизъм за глобално заключване на интерпретатора синхронизиращ нишките, които биват използвани по такъв начин, че в даден момент от време да съществува само една главна нишка в изпълнение. Това значително намаля ефективността при използване на многонишкови системи и съответно увеличава времезакъснението за отговор на СД контролери. Аналогична е ситуацията при анализа на максималната пропускателна способност на контролерите представена на фиг. 5. Анализът на резултатите на двата контролера (RYU и POX) показват, че броя на процесорните ядра (нишките) не указват влияние върху максималната пропускателна способност на контролерите. За разлика от тях, резултатите от анализа на контролерите програмирани на езици като Java [13] показват значително по-добри резултати, както при изследването на времезакъснението в отговор на Packet-In съобщения, така и при и изследването на максималната пропускателна способност, фиг. 5. И трите контролера (ONOS, Beacon и Floodlight) показват аналогични резултати по време на проведените изследвания, като при всеки един от тях резултатите се увеличават с увеличаването на процесорните ядра (нишки) на контролера. Важна особеност, която трябва

да бъде подчертае тук, е че резултатите на контролерите са най-добри в диапазона на 7-8 процесорни ядра/нишки, като увеличението на производителността е представено на фиг. 6.



Фиг. 6. Процентно увеличение на производителността на SDN контролери

На фиг. 7 е представен резултата при анализа на максималната пропускателна способност на СД контролери при промяна диаметъра на мрежата, която е под техния административен домейн. От графиката ясно се вижда, че броя на OpenFlow комутаторите до голяма степен не влияе върху общата производителността на СД контролери. Изключение прави единствено Floodlight контролера, при който се наблюдава отказ на услугите след включване на повече от 12 комутатора към него. Необходимо е да се отбележи, че отказа на услуги на Floodlight контролера не се наблюдаваше при всеки един от опитите по време на изследването, но по време на по-голямата част от тях.



Фиг. 7. Максимална пропускателна способност при промяна диаметъра на мрежата

ИЗВОДИ

Софтуерно дефинираните мрежи представляват нова технология, която има за цел да замести традиционните телекомуникационни мрежи с нова, по-надеждна, програмируема система за управление на мрежите от ново поколение. За постигане на целта, контролната и физическата равнина биват логически разделени, като същевременно цялата логика бива насочена към централизиран сървър наричан софтуерно дефиниран контролер. Доброто познаване на процесите изпълняващи се от контролера, а също така и анализа на неговата производителност са от голямо значение за изграждане на по-надеждни мрежи от ново поколение.

Настоящият доклад има за цел да представи някой от основните особености на софтуерно дефинираните мрежи и да анализира производителността на някой от най-известните СД контролери. Резултатите от извършения анализ показват, че времезакъснението на контролерите в отговор на Packet-In съобщения генерирани от слоя за достъп (физическия слоя), до голяма степен не зависи от процесорните ядра/нишки на

съответния контролер, докато максималната пропускателна способност на контролерите изцяло зависи от техния брой. От извършеният анализ се забелязва, че непрекъснатото увеличение на процесорните ядра не увеличава общата производителност на контролера, а в някой случай даже тя намалява. Този момент е най-силно изразен при ONOS и Floodlight контролери. Освен анализа на общата производителност на контролерите, бе изследвано и тяхното поведение при увеличаване на диаметъра на мрежата, която те администрират. За всички контролери освен Floodlight, не бе забелязвано изменение в поведението на контролера при увеличаване на броя на OpenFlow комутаторите.

От извършеният анализ в настоящият доклад може да бъде заключено, че архитектурата и програмния език на който са написани контролерите определят до голяма степен тяхната производителност. За контролери които използват Python програмен език производителността на контролерите не зависи от увеличаването на хардуерните им ресурси. За разлика от тях, контролерите използващи програмен език Java показват увеличаване в производителността при увеличаване на броя на процесорните им ядра/нишки. Трябва да бъде отбелязано, че при достигане на 8 броя на процесорните ядра/нишки, пропускателната способност на контролера остава постоянна, т.е. не се наблюдава увеличаване, а в някой случай се наблюдава и нейното понижаване.

ACKNOWLEDGMENTS

This paper reflects the results received during the implementation of Project №2018-FEEA-02, funded by the Scientific and Research Fund of the University of Ruse “Angel Kanchev”. The study was financially supported by the University of Ruse “Angel Kanchev” contract №BG05M2OP001-2.009-0011-C01, "Support for the development of human resources for research and innovation at the University of Ruse “Angel Kanchev”", which is funded with support from the Operational Program "Science and Education for Smart Growth 2014 - 2020" financed by the European Social Fund of the European Union.

REFERENCES

[1] Open Networking Foundation, Software-Defined Networking: The New Norm for Networks, ONF White Paper, April 13, 2012.

[2] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Turner, J. (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.

[3] Open Networking Foundation, SDN Architecture, Issue 1, June, 2014

[4] Open Networking Foundation, OpenFlow Switch Specification version 1.5.1 (Protocol version 0x06), March 26th, 2015

[5] http://www.projectfloodlight.org/floodlight/

[6] https://www.opendaylight.org/

[7] https://onosproject.org/

[8] https://ryu.readthedocs.io/en/latest/

[9] https://openflow.stanford.edu/display/Beacon/Home.html

[10] http://mininet.org/

[11] https://github.com/mininet/oflops/tree/master/cbench

[12] https://www.python.org/

[13] https://www.oracle.com/java/