UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI

Şcoala Doctorală de Electronică, Telecomunicații și Tehnologia Informației

Decizie nr. 973 din 08-12-2022

REZUMAT TEZĂ DE DOCTORAT

Ing. Teodor-Pantelimon Tivig

OPTIMIZAREA PLANULUI DE CONTROL ÎN REȚELE DEFINITE PRIN SOFTWARE

COMISIA DE DOCTORAT

Prof. Dr. Ing. Ion MARGHESCU Univ. Politehnica din Bucureşti	Preşedinte
Prof. Dr. Ing. Eugen BORCOCI Univ. Politehnica din Bucureşti	Conducător de doctorat
Prof. Dr. Ing. Sorin ZOICAN Univ. Politehnica din Bucureşti	Referent
Prof. Dr. Ing. Virgil DOBROTĂ Univ. Politehnica din Cluj-Napoca	Referent
Prof. Dr. Ing. Florin ALEXA Univ. Politehnica din Timișoara	Referent

BUCUREŞTI 2023

Cuprins

Cuprins ii
Capitolul 1 6
Introducere 6
1.1 Prezentarea domeniului tezei de doctorat 7
1.2 Scopul tezei de doctorat 7
Capitolul 2 ..... **Error! Bookmark not defined.**
Starea actuală a domeniului SDN și problemele deschise 9
2.1 Probleme întâmpinate cu rețelele clasice IP 9
2.2 Rețele definite de software 11
2.2.1. Controler Ryu 12
2.3 O analiză critică a problemei amplasării multi-controlerului în rețele SDN de arie largă 13
2.3.1 Limitarea unei soluții de control unic 13
2.3.2 Analize și discuții privind aspectele de plasare dinamică și statică a controlerului SDN 13
2.3.3 Analiză critică la problemele deschise privind CPP 14
Capitolul 3 15
Calitatea serviciilor în rețelele definite prin software 15
3.1 Relația dintre SDN și QoS 16
3.2 Experimente de evaluare a performanței pentru traficul video și VoIP cu controlerul RYU și cadrul Mininet 17
Capitolul 4 17
4.1 Prototiparea aplicației pentru redirecționarea pachetelor 18
4.1.2 Managerul de evenimente pentru comutatoarele nou înregistrate 18
4.1.2 Adăugarea de intrări pentru fluxuri în comutator 19
4.1.3 Construirea mesajului de modificarea a fluxului 19
4.1.6 Analiza pachetului și verificarea integrității mesajului 20
Capitolul 5 21
Optimizarea Planului de Control în Tehnologia SDN 21
5.1 Problema de plasare a multicontrolerelor SDN 22
5.2. Problema de plasare a mai multor controlere în rețelele SDN cu arie largă 23
5.3 Contribuții pentru crearea unui plan de control scalabil în tehnologia SDN 23

5.3.1 Soluția propusă și rezultate 24
Capitolul 6 26
Concluzii 26
6.1 Rezultate Obținute 27
6.2 Sumar al contribuțiilor originale 27
6.3. Publicații personale 28
6.3.1 Lista Articolelor Publicate/Acceptate pentu publicare 28
6.3.4 Lista Proiectelor 30
Colaborator pentru conducerea temelor de dizertatie la Master. 31
6.4 Obiective de viitor 31
Referințe 31

Capitolul 1

Introducere

În ultimii ani, în contextul dezvoltării rețelelor mari de telecomunicații, a fost introdus conceptul de Software Defined Networks (SDN), care aduce în prim plan rețelele formate din plan de control (controler SDN) și noduri de transmitere a traficului de date.

„SDN este văzut ca o nouă paradigmă, care încearcă să construiască un model de rețea elastic, adaptabil și programabil prin decuplarea planului de control, de planul de date". Acest model, cu ajutorul programării, permite inovare într-un mod agil, care nu a mai fost cunoscut anterior. Nu în ultimul rând, paradigma de rețea, care promite să înlăture specificul unui producător de echipamente de rețea, dar și cerințele specifice unui administrator de rețea, prin transformarea dispozitivelor comutatoare de rețea, în dispozitive simple de redirecționare a pachetelor, care pot fi programabile printr-o interfață, plasând în același timp, planul de control al fiecărui dispozitiv de rețea, într-un modul centralizat logic [1].

SDN se bazează pe controlul rețelelor centralizate logic. Această centralizare, datorită SDN-ului, aduce câteva provocări care au legătură cu controlul distribuit, în rețelele tradiționale TCP/IP.

În tehnologia rețelelor de calculatoare, una dintre cele mai puternice mișcări o reprezintă tendința de a crea aplicații dezvoltate de utilizator, deoarece utilizatorii nu mai sunt priviți ca și consumatori pasivi, ci pot reprezenta un vector al dezvoltării de aplicații. Utilizatorii pot fi reprezentați de anumite departamente ale companiilor sau chiar companii într-un mediu partajat și doresc să își personalizeze aplicațiile și serviciile. În acest context, aplicațiile se referă la serviciile și funcțiile de rețea: balansarea traficului pentru optimizarea resurselor de rețea, rutare și chiar securitatea. Pentru a putea implementa aceste aplicații, rețelele trebuie să se schimbe, să devină flexibile, adaptabile cerințelor utilizatorilor, prin aplicarea cerințelor ca servicii. Acestă mișcare a indus conceptualizarea și adoptarea de noi modele de rețea, cum ar fi Sofware Defined Networks [2], [3] și Network Function Virtualization [4], [5].

“Conceptul de Network Function Virtualization (NFV) este atât un concept, cât și o tehnologie puternică, în continuă dezvoltare. La nivel de concept, NFV propune dezagregarea software de hardwareul pe care rulează, prin realizarea la nivelul softwareului a cât mai multe funcții de rețea, care în mod tradițional au fost construite cu ajutorul hardwareului și softwareului, special concepute pentru a lucra în tandem.

Evoluția și tendința actuală se concentrează pe o utilizare cât mai intensă și într-o legătură strânsă a tehnologiilor SDN și NFV.

O soluție pentru eficientizarea rețelelor mari este reprezentată de dezvoltarea rețelelor multicontroler, cu o geografie bine definită. Avantajele rețelei multicontroler sunt din punct de vedere a traficului, atât prin posibilitatea de împărțire între controlere a acestuia, cât și prin preluarea la nevoie a traficului unui controler de către celelalte controlere, acesta din urmă reprezentând redundanța controlerelor in rețelele SDN. Unul din dezavantajele majore a acestor rețele este dată de latența care intervine la comunicarea dintre controlere și noduri, atunci când se realizează comunicarea inter-controlere. De asemenea, latența apare și la comunicarea inter-controlere. Soluția intens studiată pentru a minimiza dezavantajele rețelelor SDN este dată de amplasarea optimă a controlerelor, cu o vedere de ansamblu asupra rețelei.

1.1 Prezentarea domeniului tezei de doctorat

Domeniul principal al acestei teze este reprezentat de focusarea pe tehnologia SDN și studiul cât și contribuțiile personale la o problemă care este încă deschisă: sincronizarea stării rețelei cu mai multe controlere în cazul defectării controlerului master.

Pe baza cadrului software de dezvoltare al controlerului Ryu și al emulatorului de rețea Mininet, domeniul tezei se concentrează în principal pe optimizarea planului de control, în rețelele definite prin software-SDN. Pe baza textului, care identifică provocările, teza creează experimente și contribuții personale pentru concepte, configurații software și integrează baze de date distribuite, pentru consistența stării rețelei. În plus de cele prezentate, rețelele 5G, NFV și instrumentele software sunt introduce, de asemenea în studiu.

1.2 Scopul tezei de doctorat

Punctul de plecare al acestei teze a fost experiența mea profesională coroborata cu interesul pentru tendințele analizate care se întrevedeau la orizont în SDN, virtualizare, NFV și cloud computing.

Mai mult oportunitatea de cercetare științifică pentru a contribui la o tehnologie, care introduce automatizarea în domeniul rețelisticii, domeniul în care activez profesional, pentru a rezolva una dintre provocările actuale și anume de a configura manual, mulțimea de echipamente existentă într-o rețea de furnizor de servicii și eliminarea anumitor sarcini, care se execută manual, ambele supuse la erori din partea factorului uman mi-a dat șansa de a cunoaște, a lucra și a contribui la optimizarea planului de control în rețelele definite prin software-SDN.

1.3 Conţinutul tezei de doctorat

Această teză este organizată după cum urmează:

Capitolul 2 „Starea actuală a domeniului și problemele deschise” prezintă stadiul actual al rețelelor IP clasice, provocările și limitările acestei arhitecturi descentralizate, precum și domeniul rețelelor definite software, dar și **propria contribuție** la identificarea problemelor deschise în SDN.

Capitolul 3 „Calitatea serviciului în rețelele definite prin software” prezintă caracteristicile QoS specifice rețelelor definite prin software și diverse cercetări care au tratat subiectul QoS în arhitectura SDN, dar și experimente care implică parametrii utilizați la evaluare calității serviciului.

Capitolul 4 „Implementări de aplicații pentru controlere SDN” se focusează pe prezentarea componentelor cadrului Ryu și explicația detaliată a evenimentelor specifice, cu ajutorul cărora este posibil crearea de aplicații de management și control, dar și propria contribuție pentru elaborarea unei aplicații pentru direcționarea traficului.

Capitolul 5 „Optimizarea planului de control în tehnologia SDN” se focusează pe prezentare și testarea unui mecanism propriu, care creează un plan de control scalabil și tolerant la defecțiuni fiind o posibilă soluție la problema prezenței unui singur controler în rețeaua definită prin software.

Capitolul 6 „Concluzii” - sumarizează principalele lecții învățate pe parcursul activității doctorale

Capitolul 2

Starea actuală a domeniului SDN și problemele deschise

În acest capitol sunt prezentate provocările întâmpinate în rețelele IP clasice. Deasemenea acest capitol este dedicat înțelegerii arhitectura rețelelor definite prin software (SDN) împreună cu o analiză critică a amplasării multi-controlerelor în tehnologia SDN utilizată în rețelele dispersate pe arii geografice mari. Acestă analiză a fost utilizată ca punct de start pentru pentru arhitectura propusă în capitolul 5.

Acestă abordare critică a plecat de la observația noastră că există relativ putine ceretări care tratează problema amplasării controlerului SDN (CPP) cu accent pe un singur criteriu de performață dar și inexistența unor mecanisme pentru controlere SDN de tip sură deschisă de a realege un nou master controler în scenariul cand controlerul master curent se defectează.

2.1 Probleme întâmpinate cu rețelele clasice IP

Rețelele tradiționale utilizează o cantitate mare de resurse umane (operatori de rețea) și echipamente (routere, switchuri și echipamente intermediare). Acesta este motivul pentru care rețelele tradiționale prezintă aceste provocări, în timp ce operatorii de rețea trebuie să configureze protocoalele, să utilizeze politici de nivel înalt pentru a configura; prin utilizarea comenzilor de configurare se răspunde astfel diferitelor tipuri de evenimente din rețea, adaptându-se permanent la condițiile în schimbare [6].

Rețelele tradiționale sunt integrate vertical, ceea ce înseamnă că planul de control (factorul de decizie privind traficul în rețea) și planul de date (expeditorul traficului de rețea conform deciziei planului de control) sunt strâns integrate în interiorul rețelei.

După cum se vede în Figura 2.1 [7], rețelele sunt formate din planuri arborescente: plan de date, plan de control și planuri de management. Planul de date corespunde dispozitivelor de rețea utilizate pentru transmiterea pachetelor, planul de control corespunde protocoalelor care elaborează tabelele de transmitere, planul de management include atât serviciile software [8], cât și politicile de rețea. Serviciile software sunt folosite pentru a supraveghea planul de control, iar politicile de rețea definite în planul de management sunt folosite pentru a transmite date în acord cu planul de control [4].

Ultimele două planuri menționate sunt localizate pe dispozitivele de rețea. Încorporarea acestora pe aceleași dispozitiv și strânsa legătură dintre ele impune o flexibilitate limitată rețelelor tradiționale IP, deoarece aceste două planuri sunt integrate vertical, rezultând o arhitectură descentralizată. Aceste caracteristici au fost importante pentru proiectarea rețelei Internet; de asemenea important în construirea rețelelor a fost reziliența și performața acestora, pentru rata de linie și densitatea porturilor [9].

Rețelele tradiționale pot conține frecvent configurații greșite și erori asociate (mai mult de 1000 de routere BGP) [10], un singur dispozitiv configurat greșit producând pierderi de pachete, bucle de redirecționare, căi de redirecționare neintenționate și încălcări ale contractului [7].

Rețelele IP tradiționale sunt foarte descentralizate, cu planuri de control și date strâns cuplate, caracteristici care au fost importante pentru proiectarea Internetului. De asemenea, importantă în construirea rețelei a fost rezistența rețelei și performanța acesteia, pentru a crește rapid rata de linie și densitățile portului [7].

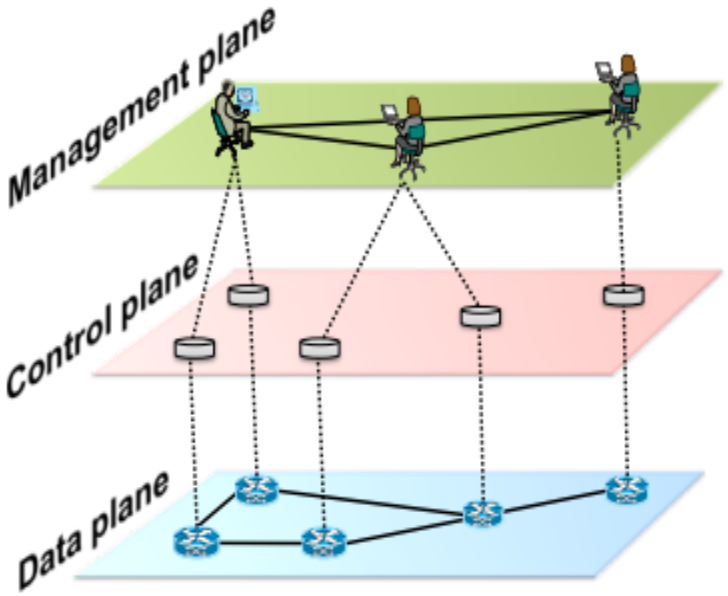

Figura 2. 1. *Vedere stratificată pe planuri a funcționalității rețelei* [7]*, [Copyright © 2015, IEEE]*

2.2 Rețele definite de software

Termenul SDN reprezintă o paradigmă recentă în rețelele de calculatoare. Acesta a devenit mai bine cunoscut cu apariția protocolului OpenFlow în anul 2008 [9] [11]. Separarea planurilor de control și de date ale echipamentelor de rețea a luat naștere de prin anii 1980 [9], [12]. Încercările timpurii de a realiza rețele programate au existat de prin anii 1997 [9], [13].

Rețelele definite software (SDN) prezintă o arhitectură în care planul de date și planul de control sunt separate sau decuplate. Arhitectura SDN cuprinde trei elemente sau straturi principale, care sunt: stratul de aplicație, stratul de control și stratul de redirecționare, așa cum se prezintă în Figura 2.2 [7].

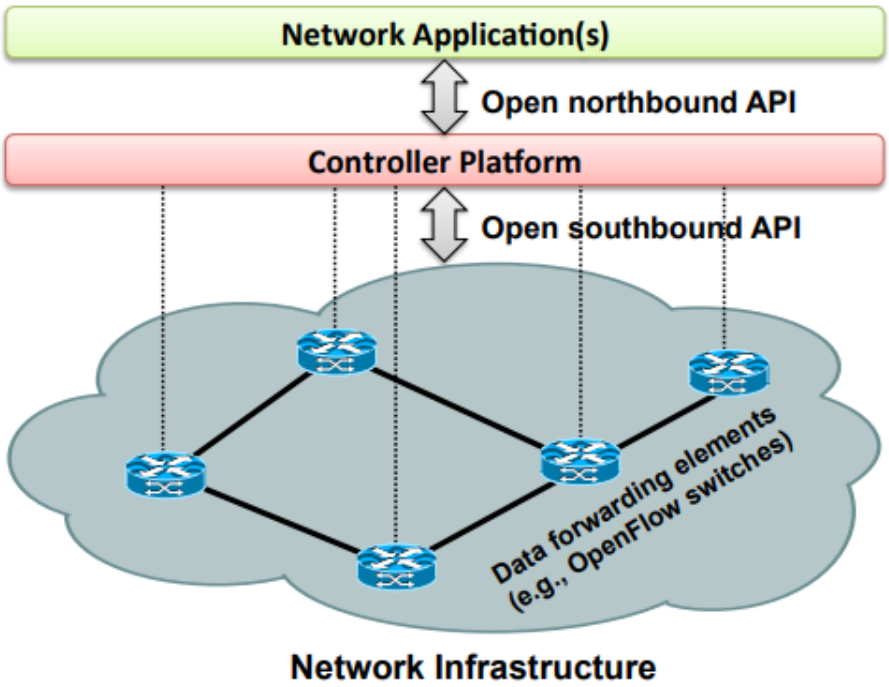

Figura 2. 2. *Vedere simplificată asupra arhitecturii definite software* [7]*, [Copyright © 2015, IEEE]*

O arhitectură a unei rețele definite software are patru atribute [7]:

- Planurile de control și date sunt decuplate. Dispozitivele de rețea reprezintă doar elementele de comutare, fără a mai încorpora decizii de control.
- Deciziile de redirecționare sunt bazate pe fluxuri. Un flux este un set de câmpuri de valori al pachetului, compus din un criteriu de potrivire și un set de instrucțiuni. În rețelele definite prin software, un flux este definit ca un set de pachete între o sursă și o destinație cu aceleași politici de serviciu [14], [15].

- Controlul logic este denumit controler SDN sau sistem de operare de rețea (NOS). Un NOS este un software care rulează pe un server și programează dispozitivele de comutare pe baza logicii centralizate și pe o vedere abstractă a rețelei.
- Programabilitatea rețelei reprezintă o caracteristică principală a SDN, oferită de aplicațiile software ale planului de management.

2.2.1. Controler Ryu

Controlerul ales pentru această teză este Ryu. În arhitectura SDN, Ryu este descris ca un software bazat pe componente și open source. Pentru implementarea Ryu se folosește limbajul de programare Python, oferind gratuit instrumentele sale software, care conțin API-uri explicite.

Arhitectura controlerului Ryu este prezentată în Figura 2.6 [16].

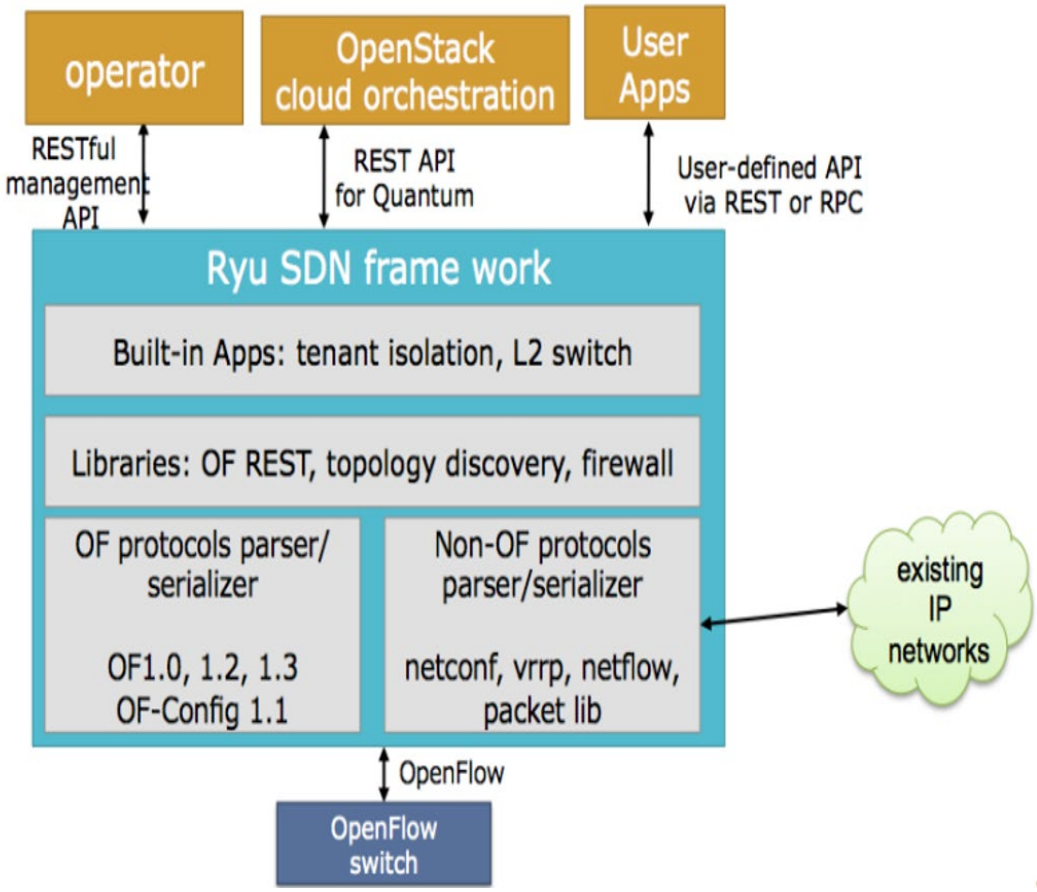

Figura 2. 3. *Arhitectura controlerului Ryu [16]*

2.3 O analiză critică a problemei amplasării multi-controlerului în rețele SDN de arie largă

”În această secțiune comparăm complexitatea diferitelor abordări CPP, prin analiză critică, utilizând două criterii largi, adică maparea statică și elastică a controlerului la comutatoare, având ca scop evidențierea lacunelor potențiale de cercetare. Criteriile de evaluare specifice SDN vor fi: capacitate, scalabilitate și echilibru de încărcare. Multe studii de cercetare CPP au considerat o mapare statică între comutatoare și controlere. Cu toate acestea, în rețelele reale, încărcarea de trafic este dinamică, cu cerințe mari pentru lățimea de bandă; prin urmare, unele studii au concluzionat recent că maparea statică nu este cea mai bună abordare, pentru a îndeplini Acordurile privind Nivelurile de Servicii (SLA), care reglementează multe servicii de rețea.

2.3.1 Limitarea unei soluții de control unic

„Primele arhitecturi SDN au utilizat un singur controler pentru a controla întreaga infrastructură. Între timp, cercetătorii au ajuns la concluzia că un singur controler nu poate face față cererii de debit mare de date din partea comutatoarelor, din cauza capacității limitate a controlerului [17].

De asemenea, un singur controler este un singur punct de defecțiune; consecința este că, fără controler, dispozitivele de redirecționare nu pot învăța cum să trateze pachetele nou sosite. Pentru atenuarea acestor probleme, a fost propus un design arhitectural modern, folosind o schemă multi-controler. În noua schemă [7] există majoritatea conceptelor de master, slave și peer controler; orice comutator se poate conecta la un singur master și la mai multe controlere peer sau slave. Un controler de tip peer este unul care are acces deplin la comutatoare și este un controler de egalitate cu un alt controler având același rol”.

2.3.2 Analize și discuții privind aspectele de plasare dinamică și statică a controlerului SDN

„Amplasarea controlerului are două aspecte strategice: dinamică și statică. În general, multe lucrări propun metode care au un număr cunoscut de controlere și apoi rezolvă optim CPP”.

”Mai ales în rețelele statice și de dimensiuni medii, amplasarea optimă a controlerului se realizează în consecință, luând în considerare doi parametri: tipul de topologie și capacitatea controlerului, [18]. În această secțiune, încercăm să realizăm o analiză critică a problemei CPP în cazul în care condițiile rețelei se schimbă.”

2.3.2.1 Abordare statică pentru amplasare controlerului SDN

„Multe studii au încercat să optimizeze timpul de procesare a fluxului pe interfețele de rețea ale controlerului, îmbunătățind astfel timpul de răspuns al controlerelor în timp ce alti cercetători [19] au folosit de asemenea și modelul model de stocare a mesajelor în memorii tampon, dar au impus că numărul de fluxuri procesate de controlor nu va depăși capacitatea controlerului.

În schimb, alte studii au descoperit că partiționarea rețelei în subdomenii va crește numărul de controlere și configurarea fluxului va scădea la jumătate. Părerea noastră este că orice abordare pentru îmbunătățirea timpului de configurare este valoroasă pentru CPP, pe care o vom dezbate în continuarea secțiunii”.

2.3.3 Analiză critică la problemele deschise privind CPP

„Până în prezent, problema selectării migrării pentru un comutator sau controler nu a fost rezolvată complet în niciuna dintre lucrările citate; nu există încă o abordare clară pentru alegerea unui comutator țintă sau un controler. Dacă există mai multe controlere subutilizate, nu există un mecanism clar care ar trebui să fie selectat. Există câteva studii care iau în considerare eficiența costurilor de migrare pentru un cluster de comutatoare. Această lipsă de informații ar trebui extinsă în cercetările viitoare.

Dezvoltarea unei metode de previziune a comportamentului traficului cu scopul de a estima starea viitoare în care traficul este imprevizibil este o problemă de cercetare încă deschisă. Acest lucru este important pentru a lua decizia de migrare a comutatoarelor.”

Capitolul 3

Calitatea serviciilor în rețelele definite prin software

Odată cu dezvoltarea rapidă a SDN, cercetătorii din mediul academic și din industrie au încercat să obțină furnizarea QoS, pentru mai multe aplicații de rețea în prezent. Diverse studii, care au abordat subiectul QoS, au descris una sau mai multe dintre categoriile, în care QoS poate fi plasat ca un factor avantajos al arhitecturilor SDN. Aceste categorii pot fi transformate într-o schemă simplă (Figura 3.1 [20]), fiecare dintre cele 7 elemente, reflectă o provocare pentru QoS în SDN care a fost studiată în cercetări variate, pentru a aduce o soluție la una dintre cele 7 probleme particulare, în furnizarea QoS:

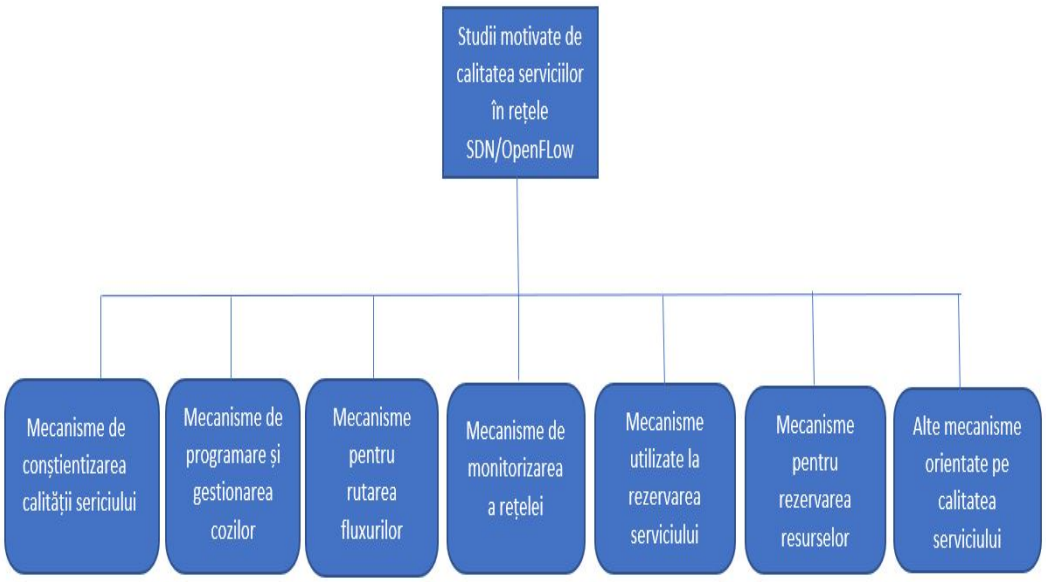

Figura 3. 1. Organizarea studiilor bazate pe calitatea serviciilor în rețelele SDN/OpenFlow, modificat după [20]

3.1 Relația dintre SDN și QoS

Scopul cheie al QoS este de a asigura clasificarea traficului, care rezultă din utilizarea parametrilor QoS [20]:

- lățimea de bandă;
- întârziere;
- jitter;
- pierderea pachetelor.

Relația dintre mecanismele SDN și furnizarea calității serviciului este enumerată mai jos [20]:

- separarea planului de date și a planului de control pentru rețele, care îmbunătățește controlerul de rețea, din perspectiva controlului rețelei;
- aplicațiile de rețea nu sunt forțate să se ocupe de configurații de nivel scăzut, ale dispozitivelor din planul de date și sunt furnizate cu vedere abstractă a rețelei, de către controlere;

  Tabelul 3.1 [20] prezintă avantajele SDN-ului în corelație cu cele 7 categorii de furnizare QoS.

Tabelul 3.1. Principalele caracteristici ale SDN, care sunt utilizate în lucrările chestionate și relația lor cu cele 7 categorii de furnizare QoS, modificat după [20]

Organizarea Caracteristicilor SDN	Arhitectură	Rutare	Management			Modelare	
	Mecanism de rutare a calității serviciului inter-domeniu	Mecanisme de rutarea a fluxurilor multimedia	Mecanisme de rezervarea resurselor	Mecanisme de monitorizarea retelei	Mecanisme pentru programarea si gestionarea cozilor	Mecanisme de constinetizarea a calității serviciului	Alte mecanisme de constientizare a calității serviciului
Actualizare dinamică a regulilor de flux	•	•	•		•	•	•
Monitorizarea traficului			•	•	•	•	•
Configurarea cozii		•			•	•	•
Redirectionare bazată pe flux	•	•				•	
Analiza traseului fluxului	•		•	•			•
Analiza pachetului/ fluxului	•	•	•	•	•	•	•

- controlerele pot obține vizualizarea globală a rețelei și stărilor acesteia.
- SLA-urile si politicile de control pot fi specificate de un administrator la un nivel de abstractizare mai înalt, fără a fi nevoie de configurare pe fiecare dispozitiv de redirecționare.
- setul de politici și de asemenea, diferitele clase de flux sunt nerestricționate permițând reglajul fin bazat pe nevoile utilizatorului;

- prin urmare, regulile pot fi definite pe flux (dacă este necesar), iar controlerul are sarcina de a le aplica, în mod corespunzător diferitelor elemente de rețea.

3.2 Experimente de evaluare a performanței pentru traficul video și VoIP cu controlerul RYU și cadrul Mininet

Această secțiune prezintă experimentele personale SDN , al căror scop este de a crea diferite scenarii pentru a efectua teste de performanță, la nivel de nod. Testele utilizează următoarele criterii de evaluare: lățimea de bandă, întârzierea și lățimea de bandă pentru trafic de date și VoIP. Experimentele utilizează instrumentul software Iperf, emulatorul de rețea Mininet și controlerul SDN Ryu.

Statisticille de trarafic sunt utile pentru a obține informații despre congestionarea rețelei cu scopul de a le trimite ulterior controlerului pentru a decide cum să rezolve necesarul de lățime de bandă pentru fluxurile concurente de la abonatii conectați la rețea.

Capitolul 4

Implementări de Aplicații pentru Controlere SDN

Această secțiune continuă lucrările din Capitolul 3, cu o secțiune despre prototiparea unei aplicații de control pentru controlerul Ryu. Secțiunea curentă implementează, analizează, explică și testează logica unui comutator care efectuează o redirecționare a pachetelor utilizând criterii de nivelul 3 al stivei OSI.

Aceste componente software prezintă API, care fac posibilă creearea de aplicații de control într-o manieră personalizată, instruind controlerul Ryu cum utilizatorul dorește sa uilizeze echipamentul.

Aplicațiile pentru controlerul Ryu sunt scripturi Python, iar controlerul va configurea echipamentul de rețea utilizând protocolul Open Flow sau alt protocol de pe interfața de sud.

4.1 Prototiparea aplicației pentru redirecționarea pachetelor

Aplicația propusă va implementa funcționalități precum transferul pachetelor de la o interfață de rețea la altă interfață spre adresa de destinație, stocarea acestora în memoria tampon a echipamentului și efectuarea altor acțiuni specific pachetelor. Codul pentru aplicația propusa se bazează pe aplicația *simple_switch.py* [18], pe care vom construi pentru crearea funcționalitatea aplicației propuse.

„În această etapă, aplicația va inițializa datele care vor fi utilizate în întreaga infrastructură. Pentru a implementa caracteristica cheie a inteligenței echipamentului va fi utilizat managerul *ryu.base.app_ ,* care mostenește caracteristicile comutatorului de Nivel 2 OSI”.

```
clasa Layer3forwarding(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_5.OFP_VERSION]
    def __init__(self, *args, **kwargs):
            super(Layer3forwarding).__init__(*args,**kwargs)
                    self.mac_to_port = {}
```

4.1.2 Managerul de evenimente pentru comutatoarele nou înregistrate

„Managerul de evenimente specifică acțiunile care trebuie întreprinse pentru un anumit eveniment, cum ar fi recepționarea unui pachet. Următoarele două linii instanțiază acțiunile care urmează să fie executate. Decoratorul *set_ev_cls* se utilizează pentru menționarea clasei de eveniment. De fiecare dată când comutatorul generează un eveniment, care ajunge la controler, acesta trebuie să răspundă cu un mesaj corespunzător. Structura *ofp_event.EventOFPSwitch Features + Message_name* reprezintă argumentul, numelui clasei de eveniment sau dispecerul pentru managerul de evenimente.

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
        defswitch_features_handler(self, ev):
```

„Informațiile primite de comutator vor fi decodificate folosind următoarea structură de date de cod:”

```
datapath = ev.msg.datapath
ofproto = calea datelor.ofproto
parser = datapath.ofproto_parser
potrivire = parser.OFPMatch()
actiuni=[parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,dep
roto.OFPCML_ NO_BUFFER)]
self.add_flow(datapath, 0, match, actions)
```

4.1.3 Adăugarea de intrări pentru fluxuri în comutator

„Funcția *add_flow* adaugă o condiție de potrivire, instrucțiuni și de asemenea timpul efectiv și nivelul de prioritate de intrare pentru pachetul vizat în fiecare intrare nouă de flux. Această funcție este invocată de fiecare dată ca răspuns la un anumit mesaj.”

```
def add_flow(self, datapath, priority, match, actions, buffer_id=None):
ofproto = calea datelor.ofproto
parser =datapath.ofproto_parser
inst=[parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
```

4.1.4 Construirea mesajului de modificarea a fluxului

„Mesajele de modificare a fluxului (*FlowMods*) indică căii de date să adauge, să șteargă sau să modifice un flux în tabelul său de flux.

```
if buffer_id:
   mod=parser.OFPFlowMod(datapath=datapath,buffer_id=buffer_id,
                        prioritate=prioritate,potrivire=potrivire,
                        instrucțiuni=inst)
                else:
                        mod=parser.OFPFlowMod(datapath
                        =datapath,priority=priority,match
                        =potrivire, instrucțiuni=inst)
                        datapath.send_msg(mod)
```

4.1.5 Analiza pachetului și verificarea integrității mesajului

„Controlerul Ryu are propria bibliotecă de gestionare a pachetelor încorporată, care conține regulile pentru tratarea mesajelor de notificare *Packet-In*.

```
    @set_ev_cls(ofp_event.EventOFPPacketIn,MAIN_EXPEDITOR)
        def _packet_in_handler(self, ev):
    if ev.msg.msg_len < ev.msg.total_len:
                        self.logger.debug("packet truncated:only %s of %s bytes",
v.msg.msg _len)
    msg = ev.msg
    datapath = msg.datapat
    ofproto = calea datelor.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']
    msg = ev.msg
    datapath = msg.datapat
    ofproto = calea datelor.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']
    pkt =packet.Packet(msg.data)
        eth =pkt.get_protocols(ethernet.ethernet)[0]
    if eth.ethertype == ether_types.ETH_TYPE_
                        LLDP:
                        return
    dst = eth.dst
    src = eth.src
    dpid = datapath.id
    self.mac_to_port.setdefault(dpid, {})
```

Metadatele necesare pentru pachet sunt înregistrate în scopul depanării.

```
    self.mac_to_port[dpid][src] = in_port
    if dst în self.mac_to_port[dpid]:
                out_port = self.mac_to_port[dpid][dst
        else:
                out_port = ofproto.OFPP_FLOOD
    actions = [parser.OFPActionOutput(out_port)]”
    if out_port != ofproto.OFPP_FLOOD:
                dacă L3.ethertype==ether_types.ETH_TYPE_IP:ip
                        =pkt.
                        obține _ protocol(ip4.ipv4)
```

srsip=ip.src
dstip=ip.dst
potrivire=parser.OFMatch(eth_type=ether_

Capitolul 5

Optimizarea Planului de Control în Tehnologia SDN

Acest capitol expune principalele contribuții științifice pentru crearea unui plan de control scalabil tolerant la defecțiuni care să reprezinte o soluție potențială la problema unui singur controler care reprezintă un singur punct de defecțiune în rețelele definite prin software. Mecanismul pentru a crea un plan de control scalabil utilizează Redis și Zookeeper. Simulările software sunt realizate cu controlerul Ryu și emulatorul Mininet.

Obiectivele generale ale acestui capitol sunt:

- Prezentarea problemei de amplasare a mai multor controlere SDN și problemele sale intermediare:
    - numărul de controlere SDN pentru funcționarea optimă a rețelei;
    - și unde să se plaseze controlerele în rețea
- Propunerea unui scenariu științific, precum și o soluție practică pentru asigurarea unui plan de control scalabil în tehnologia SDN.
- Componentele funcționale și instrumentele utilizate sunt următoarele:
- Python, Ryu, Mininet, Zookeeper și Redis.

5.1 Problema de plasare a multicontrolerelor SDN

Pentru a rezolva problema de plasare a mai multor controlere în tehnologia SDN, trebuie să se îmbunătățească mai multe valori, cum ar fi latența dintre controler și elementele de redirecționare și între controlere dar și reziliența și fiabilitatea. De asemenea, luând în considerare latența pentru arhitecturile de rețea reale, alegerea optimă în scenarii fără eșec este evaluarea spațiului global al soluției, cu calcule offline.

Arhitectura planului de control al rețelelor definite prin software a fost proiectată cu un singur controler sau cu controlere multiple aceste fiind clasificate la rândul lor fie în arhitecturi fizice sau logice centralizate sau arhitecturi fizice sau logice distribuite așa cum se prezintă în Figura 5.1.

Într-o arhitectură multicontroler, controlerele lucrează simultan pentru obținerea unei performanțe și scalabilitate rezonabilă. O arhitectură SDN multi-controler poate avea forme și aspecte multiple, cu segmente de rețea împărțite în arhitecturi logice sau fizice centralizate sau distribuite și segment plate și ierarhice.

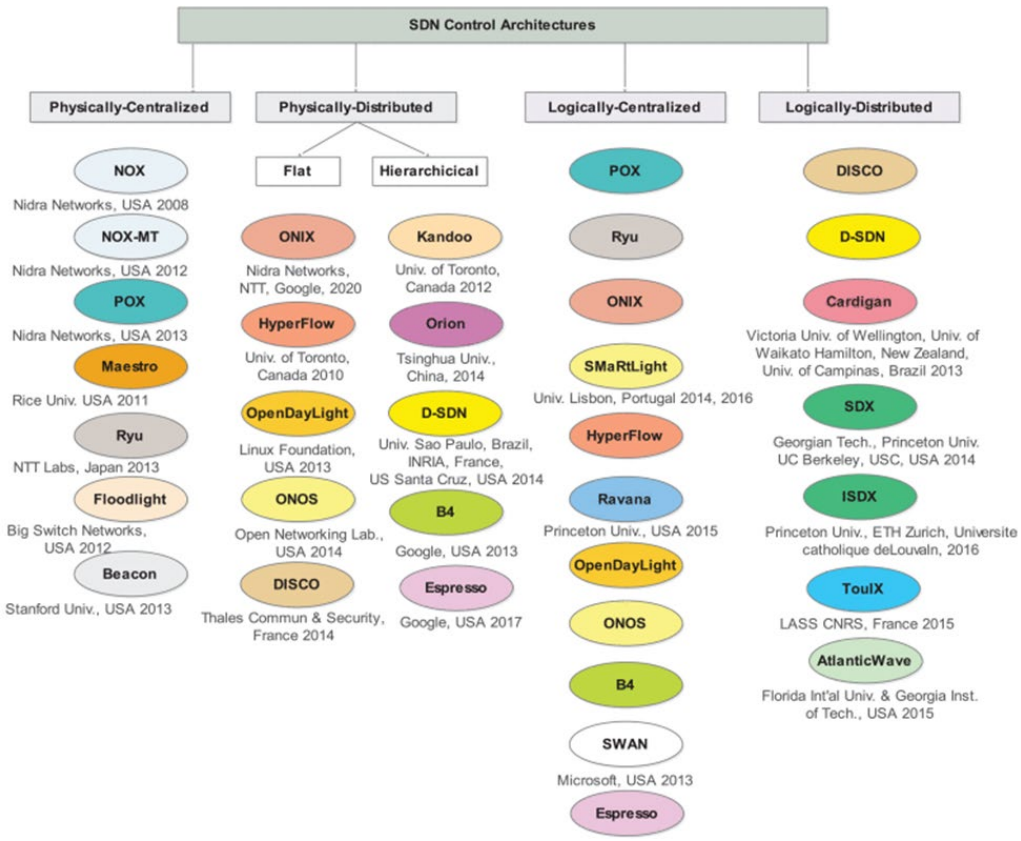

Figura 5. 1. *Clasificarea arhitecturilor planului de control SDN [21], [Copyright © 2018, IEEE]*

5.2. Problema de plasare a mai multor controlere în rețelele SDN cu arie largă

Această secțiune este menită să introducă o serie de constângeri cât și obiectivele pentru amplasarea multicontrolerelor în rețele SDN cu arie extinsă cât și de asemenea, provocări care pot aparea în implementarea lor pentru realizarea obiectivelor. De asemenea, în rețelele SDN cu suprafață mare este necesară utilizarea algoritmilor pentru a rezolva problema de plasare a controlerului rețelelor SDN cu suprafețe mari multi-obiective [22].

Constrângerile utilizate la amplasarea controlerelor includ:

- capacitățile controlerelor;
- sarcinile comutatoarelor.

Obiectivele aplicabile pentru rețelele SDN la scară largă [22],

- latența dintre comutatoare și controlere;
- latența inter-controler;
- echilibrarea sarcinii;
- defecțiunile nodurilor, controlerelor și legăturilor.

5.3 Contribuții pentru crearea unui plan de control scalabil în tehnologia SDN

În această secțiune, se prezintă un mecanism pentru crearea uni plan de control în arhitectura SDN.

Ideea principală a acestei teze este că încearcă să creeze și să prezinte o testare live a unui mecanism robust, care să reprezinte o soluție potențială la problema existenței unui punct de defecțiune în arhitectura SDN.

Mecanismul prin care controlerele SDN care gestionează rețeaua au o bază topologică simetrică, în scenariul când controlerul principal eșuează să mai funcționeze este realizat cu Redis și Zookeeper. Mecanismul permite ca mai multe controlere Ryu să continue să își exercite funcțiile asupra rețelei într-o manieră tolerantă la evenimente de rețea care pot scoate controlerul master din funcționare, prin intermediul unui serviciu centralizat numit Redis care menține informația despre starea rețelei iar Zookeeper este utilizat pentru alegerea controlerului master.

5.3.1 Soluția propusă și rezultate

In această secțiune se prezintă un mecanism tolerant la erori pentru a reduce punctul unic de defecțiune în rețeaua SDN și în consecință, soluția propusă va crea un grad ridicat de disponibilitate a serviciului și va crește fiabilitatea și scalabilitatea.

Dintr-o perspectivă de nivel înalt acestă arhitectura constă din următoarele componente:

- Controlere Ryu;
- Baza de date Redis pentru distribuirea informațiilor topologice pentru controlerul Ryu;
- Zookeeper utilizat pentru detectarea defecțiunilor și alegerea controlerului master sau slave;
- Mininet pentru construirea topoligiei;
- Limbajul Python utilizat la definirea topologiei și la interacțiunea componentelor prezentate mai sus.

Soluția propusă constă din trei controlere Ryu individuale care rulează în trei terminale, fiecare controler utilizând numere de porturi diferite. Asfel controlerul numărul 1 ascultă pe portul 6633, controlerul numărul 2 ascultă pe portul 6634 iar controlerul numărul 3 ascultă pe portul 6635. Controlerele vor fi utilizate în fiecare dintre cele trei roluri de egaliate, de master sau slave. În al patrulea terminal, va rula o topologie liniară de bază cu două comutatoare și fiecare comutator individual va fi conectat la cele trei controlere conform cu Figura 5.2 prin rularea comenzii: *sudo mn --controller=remote, ip=127.0.0.1:6633 -- controler= remote, ip= 127.0.0.1:6634 controler=remote, ip=127.0.0.1:6635*

În simularea controlerului în modul Equal

Comutatoarele sunt conectate după cum se poate observa din Figura 5.2 și toate controlerele vor avea roluri egale. Aceasta înseamnă că toate controlerele vor primi o notificare duplicată cu mesajul *PACKET_IN*, pentu fiecare pachet nou pe care îl primește un comutator. Având toate controlerele cu același rol, consecința fiind că numărul de pachete se va înmulți cu numărul de controlere prezente în topologie. Avantajul este că redundanța planului de control este realizată în cazul eșecului oricărui controler, iar pierderea de pachete rezultată este zero. Dezavantajul acestei configurații este scalabilitatea limitată, fiecare controler primește același mesaj *PACKET_IN*, aceeași încărcare. Graficul din Figura 5.3 afișează numărul de pachete transmise, pachete primite și pachete duplicate atunci când controlerele rulează în rolul equal.”

```
[ivtg@ubuntu] ~/Documents/CUDAS_2021/Articol_2_Comp/Using multiple controllers$ sudo ovs-vsctl show
38ad1481-d652-496d-a6bb-ff546ec1049a
    Bridge s1
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        Controller "ptcp:6654"
        Controller "tcp:127.0.0.1:8853"
            is_connected: true
        Controller "tcp:127.0.0.1:7754"
            is_connected: true
        fail_mode: secure
        Port s1
            Interface s1
                type: internal
        Port s1-eth2
            Interface s1-eth2
        Port s1-eth1
            Interface s1-eth1
    Bridge s2
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        Controller "tcp:127.0.0.1:8853"
            is_connected: true
        Controller "ptcp:6655"
        Controller "tcp:127.0.0.1:7754"
            is_connected: true
        fail_mode: secure
        Port s2-eth2
            Interface s2-eth2
        Port s2
            Interface s2
                type: internal
        Port s2-eth1
            Interface s2-eth1
    ovs_version: "2.13.3"
```

Figura 5. 2. *Evaluarea tipului de conectivitate a comutatoarelor cu toate cele trei controlere [23]*

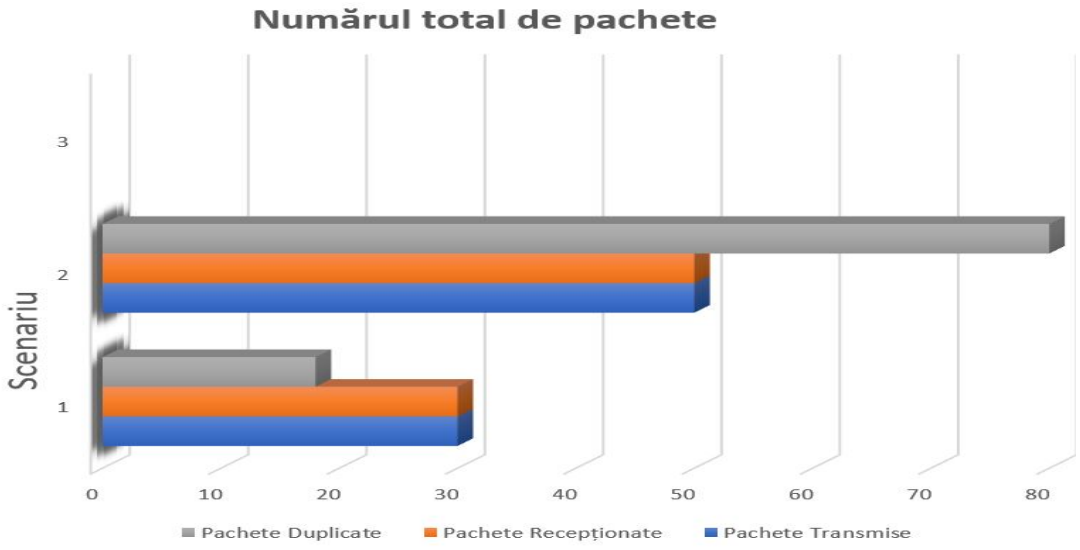

Figura 5. 3*. Statistici pentru pachetele transmise, primite și duplicate [23]*

Simularea rolului Master și Slave

Scenariul configurației master și slave va avea primul controler pornit cu rol de master, iar restul, cele două controlere funcționând cu rol de slave. Cele două controlerele care funcționează în rolul „SLAVE” nu primesc mesajul PACET_IN, ci doar controler numărul unu care are rolul de master, primește notificarea

PACKET_IN de la toate comutatoarele. După ce este verificată funcționalitatea planului de control reprezentat de primul controler testul continuă prin oprirea controlerului principal folosind comanda CTRL+C în terminalul acestuia, comandă care declanșează realegerea controlerului master și slave. Orice controler rămas poate fi ales ca și master.

Al doilea controler devine master (conform unui criteriu aleatoriu, care pote fi modificat de utilizator), acesta recepționează toate informațiile topologice din baza de date Redis, primind notificarea PACKET_IN deoarece are rolul master și controlerul numărul trei primesc rolul de SLAVE. Prin inițierea unui test ping folosind comanda *pingall* de la prima gazdă a topologiei se evaluează funcționalitatea planului de control după realegere.

În pasul următor, se pornește controlerul numărul unu care a fost oprint anterior lansând comanda: *ryu-manager --ofp-tcp-listen-port 6633 master_slave.py* și preia rolul de slave, datorită faptului că controlerul master a fost ales anterior.

După aceea, vom opri controlerul principal, controlerul numărul 2 și oricare dintre cele două controlere devine controlerul master. Se testează funcționalitatea controlerului care obține rolul de master (controlerul numarul 3) dupa care se repornește controlerul numarul doi care preia rolul de slave. În ultimul pas controlerul numărul 3 este oprit, declanșând asfel procesul de realegere a controlerului master.

Asfel prin pași succesivi de pornire și oprire a unuia dintre cele trei controlere cu rol master, controlerul care preia rolul master dupa defectarea (oprirea) controlerului master curent, se testează dacă mecanismul propus care integrează baza de date Redis, furnizează infomațiile topologice într-un timp optim către noul controler ales master la fiecare defecțiune simulată a controlerului cu rol master.

Capitolul 6

Concluzii

Tehnologia SDN aduce avantaje dar și provocări majore ca: rețelele care au un singur controler care poate impacta rețeaua devenind inutilizabilă; dar și la nivelul rețelelor dispersate pe arii geografice mari care necesită prezența multi-controlerelor prin: interconectarea acestor pentru realizarea unui plan de control rezistent la defecțiuni prin furnizarea stării rețelei controlerului care preia controlul în caz de defectarea controlerului master dar și foarte important este amplasare acestor controlere pentru satisfacerea parametrilor specifici cu puternic impact asupra calității serviciilor dar și a scalabilității deopotrivă.

Această teză propune o revizuire a lucrărilor prezente în literatura de specialitate pentru a identifica linii directoare privind contribuții pentru optimizarea planului de control atât pentru a identifica o soluție optimă pentru amplasare multi-controlerelor dar și realizării comunicație la nivelul planului de date pentru transferul stării rețelei în scenariul cât controlerul care gestionează rețeaua se defectează, dar nu a identificat o soluție completă.

6.1 Rezultate Obținute

Capitolul 1, prezintă teza, motivația acesteia și structura. Domeniul tezei se concentrează foarte puternic pe planul de control al tehnologiei SDN, dezvoltând experimente personale pentru calitatea serviciului (QoS), realizarea de aplicații pentru controlerul Ryu cu ajutorul limbajului de programare Python.

Cadrele software utilizate sunt Ryu și Mininet.

Capitolul 2, se focusează pe starea domeniului SDN și problemele deschise, provocările și stadiul rețelelor clasice. Prezintă propriile contribuții privind identificare provocărilor și problemele deschise aparținând tehnologiei SDN, dar și evidențierea aspectelor critice, care au fost prezentate la conferința ICCNE 2015.

Capitolul 3, este complet dedicat pentru prezentarea calități serviciilor arhitecturii SDN la un nivel înalt și experimentelor efectuate.

Capitolul 4, rezultatele sunt bazate pe cadrul software al controlerului Ryu și pe evenimentele software specializate, utilizate la crearea de aplicații. Există un exemplu de o aplicație bazată pe direcționarea traficului unde codul utilizat este contribuție.

Capitolul 5 este în întregimea sa opera autorului din perspective teoretice și practice. Atenția principală se bazează pe planul de control al rețelelor definite prin software și principalelor provocări care pot impacta scalabilitatea și disponibilitatea controlului dar și calitatea serviciului. Datorită acestor provocări a fost propus un mecanism și un scenariu propriu pentru un plan de control scalabil care totodată evită problema prezenței unui singur controler în topologia SDN.

Capitolul 6 este alocat pentru concluzii și perspective.

6.2 Sumar al contribuțiilor originale

Mai jos se află o descriere succintă a contribuțiilor personale prezentate în această teză.

1. Colectarea a puncte diferite de vedere și încercarea să identifice și să sublinieze aspecte critice în legătură cu tehnologia 5G.

Contribuții pot fi identificate în: Capitolul 2.6, Anexa A.3, Publicații [1-ICCNE].

2. Analiză critică privind problema amplasării multi-controlerului în rețele SDN de arie extinsă.
   Contribuții pot fi identificate în: Capitolul 2.7 Publicații [2-COMMS].
3. Experimente privind evaluarea performanțelor pentru traficul de voce și video cu controlerul Ryu și cadrul Mininet.
   Contribuții pot fi identificate în: Capitolul 3.8 Publicații [3-Buletinul Științific]
4. Analizează, explică componentele cadrului Ryu, precum și evenimentele create prin software pentru a fi utilizate la crearea de aplicații de gestionare și control. Propune o alicație pentru direcționarea traficului cu explicații și pașii necesari realizării acesteia.
   Contribuții pot fi identificate în: Capitolul 4.12 Publicații [4-ISNCC].

5. Un alt experiment propune un scenariu pentru crearea unui plan de control scalabil, tolerant la defecțiuni care încearcă să rezolve problema prezentei unui singur controller SDN, pentru controlere cu arhitectura software deschisă.
   Contribuții pot fi identificate în: Capitolul 5.3 Publicații [4-ISNCC].

6.3. Publicații personale

6.3.1 Lista Articolelor Publicate/Acceptate pentu publicare

1. „Pantelimon-Teodor Tivig, Silviu-Andrei Lazăr, **5G Mobile Technology between Requirements and Real Life Deployment**, The 2end International Conference on Communication and Network Engineering, 2015".
   Publicat în: International Journal of Future Computer and Communication (IJFCC).
   Abstracting/Indexing by Google Scholar, Engineering & Technology Digital Library, and Crossref, DOAJ, Electronic Journals Library, EI (INSPEC, IET).
   Locația conferinței: Amsterdam, Olanda.
   Data conferinței: 10-11 Decembrie.

2. „Pantelimon-Teodor Tivig, Borcoci Eugen, **Software Defined Networking Using Mininet for Agile Testing Network Scenarios,** The 12th International Conference on Computer Science and Information Technology (ICCSIT 2019), 2019"

Publicat în: **Journal of Communications (**JCM, ISSN: 1796-2021Online; 2374-4367 Print). Abstracting/Indexing: SCOPUS; DBLP; ULRICH's Periodicals Directory; etc.
Locația conferinței: Barcelona, Spania.
Data conferinței: 18-20 Decembrie.

3. Pantelimon-Teodor Tivig, Borcoci Eugen, **Critical Analysis of Multi-Controller Placement Problem in Large SDN Networks**, 13th International Conference on Communications (COMM), 2020.
   Publicată și indexat: IEEE Xplore, Conferint **Abstracting/Indexing:** ISI database.
   Locația conferinței: București, România.
   Data conferinței:18-20 Iunie.
   ISSN: 1796-2021Online

4. Pantelimon-Teodor Tivig, Borcoci Eugen, Alexandru Brumaru, **Layer 3 Forwarder Application - Implementation Experiments Based on Ryu SDN Controller**, International Symposium on Networks, Computers and Communications (ISNCC), 2021.
   Publicat în: IEEE Xplore digital library effective 2021-11-25, DBLP and Thomson Reuters.
   Indexat în: IEEE BDI
   Locația conferinței: Dubai – Emiratele Arabe Unite.
   Data conferinței: 31 Octombrie – 2 Noiembrie.
   DOI: 10.1109/ISNCC52172.2021.9615685

5. Pantelimon-Teodor Tivig, Borcoci Eugen, Alexandru Brumaru, **Performance evaluation experiments for video and voip traffic with Ryu controller and mininet framework, Buletin Științific – Universitatea Politehnică București, 2022.**
   Acceptat și publicat.
   Numărul de depunere 12260.
   Indexed in ISI și SCOPUS.

6. Pantelimon-Teodor Tivig, Borcoci Eugen, Alexandru Brumaru, **Performance Evaluation Use Cases For Video And Voip Traffic Using Ryu Controller And Mininet Framework, Buletin Științific – Universitatea Politehnică București, 2022.**
   Acceptat și publicat.
   Numărul de depunere 12261
   Indexed in ISI și SCOPUS.

7. Pantelimon-Teodor Tivig, Alexandru Brumaru, Serban Georgica Obreja, „**Creating Scalable Distributed Controlplane In SDN To Rule Out The Single Point Of failure”**, The 14th International Conference on Communications, 2022.
   Locația conferinței: București, România.
   Data conferinței: 16-18 Iunie.
   Publicat și Indexată în: Conferința IEEE BDI.
   DOI: 10.1109/COMM54429.2022.9817181

8. Pantelimon-Teodor Tivig, Eugen Borcoci, „**Performance Assessments For SDN Control Plane Into Distinct Network Topologies”**, The 30th International Conference on Software, Telecommunications and Computer Networks.
   Publicat și Indexat în: IEEE Xplore, Scopus, și DBLP.
   Data conferinței: 22 - 22 Septembrie 2022 - Split, Croația.
   DOI:10.23919/SoftCOM55329.2022.9911385

9. Pantelimon-Teodor Tivig, Eugen Borcoci, „ **Dynamically Offloading The SDN Control Plane In Large Area Networks By Condition Aware Migration Of Forwarding Devices”**, 2022 25th International Symposium on Wireless Personal Multimedia Communications.
   Publicat și Indexat în: IEEE Xplore
   Data conferinței: 30 Octombrie - 2 Noiembrie 2022 - Herning, Danemarca.
   DOI:10.1002/dac.3101

10. Pantelimon-Teodor Tivig, Eugen Borcoci, Frank Y. Li, Indika Anuradha Mendis Balapuwaduge, Marius Vochin, „**Sicing the 5G Core Network Based on SDN Ryu Controller for Everything as a Service**”, în curs de elaborare.

6.3.4 Lista Proiectelor

Cunoaștere, inovare și dezvoltare prin burse doctorale - CID-Doc – 2014-1015

Dotare de investiții la laboratoarele „Ad Net Market Media pentru cercetare și dezvoltare în viitoarele comunicații mobile” – FUTURE-NET-LAB, finanțat de UE în cadrul Programul Operațional Competitivitate (POC), axa prioritară 1, acțiunea 1.1.1:.Infrastructuri mari de cercetare și dezvoltare, Noiembrie 2019.

A Massive MIMO Enabled IoT Platform with Networking Slicing for Beyond 5G IoV/V2X and Maritime Services (SOLID-B5G), Ianuarie 2022.

Increasing the innovative competitiveness of SC Ad Net Market Media SRL through initial innovation investments in order to achieve a SmartDelta technology platform, within a newly established unit for the realization of CD activities in effective collaboration, Februarie 2022.

Colaborator pentru conducerea temelor de dizertatie la Master.

6.4 Obiective de viitor

Intenția pentru viitor este de a aduce contribuții în aria de interes a SDN și de asemenea în domeniu NFV, care este puternic integrat cu rețele definite prin software, dar și în tehnologii tangențiale Cloud computer și Machine learning.

Din perspectivă academică, posibilitatea de a intra în ierarhia personalului didactic, este luată în considerare.

Referințe

[1] Feamster, H. Kim and N., „Improving network management with software defined networking," *Communications Magazine, IEEE,* vol. vol. 51, nr. no. 2, p. pp. 114–119, 2013.

[2] Qiang Duan, Nirwan Ansari, and Mehmet Toy, „Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Network,* 2016.

[3] N. Omnes, M. Bouillon, G. Fromentoux, and O. le Grand, „A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges," *International Conference on Intelligence in Next Generation Networks,* p. pp. 64–69, Mar. 2015.

[4] ETSI Industry Specification Group (ISG) NFV, 2014b. GS NFV-MAN 001 - V1.1.1: Network Functions Virtualisation (NFV); URL:, Management and Orchestration. Technical Report., „http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf," [Interactiv].

[5] ETSI Industry Specification Group (ISG) NFV, a. ETSI GS NFV-EVE 005 V1.1.1: Network Functions Virtualisation (NFV); Ecosystem; URL:, Report on SDN Usage in NFV Architectural Framework. Technical Report. ETSI., „http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_nfv-eve005v010101p.pdf.," [Interactiv].

[6] Bruno Nunes Astuto, Marc Mendonça, Xuan Nam Nguyen, Katia Obraczka, Thierry Turletti, „A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," 2013.

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, „Software-Defined Networking: A Comprehensive Survey," *in Proceedings of the IEEE,* vol. vol. 103, nr. no. 1, pp. pp. 14-76, Jan. 2015.

[8] Presuhn, R., „Version 2 of the Protocol Operations for the SimpleNetwork Management Protocol (SNMP)," RFC 3416 (INTERNET STANDARD), Internet Engineering Task Force," Dec. 2002. [Interactiv]. Available: http://www.ietf.org/rfc/rfc3416.txt.

[9] CIARAN EGAN, Dr. Marco Ruffini, „End-to-end Capacity Reservation in Software Defined Networks," *A dissertation submitted in fulfillment of the requirements for the degree: Integrated Masters In Computer Science at the School Of Computer Science & Statistics, The University of Dublin,* May 2016.

[10] Balakrishnan, N. Feamster and H., „Detecting BGP configuration faults with static analysis," *in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, ser.NSDI'05. Berkeley, CA, USA: USENIX Association,* vol. Volume 2, p. pp. 43–56, 2005.

[11] McKeown, Nick et al., „OpenFlow: Enabling Innovation in Campus Networks," In: SIGCOMM Comput. Commun. Rev. 38.2, pp. 69–74. ISSN: 0146-4833, 2008. [Interactiv]. Available: http://doi.acm.org/10.1145/1355734.1355746.

[12] Sheinbein, D. and R. P. Weber, „800 Service Using SPC Network Capability," *In: The Bell System Technical Journal 61.7,* 1982.

[13] Tennenhouse, D.L. et al., „A survey of active network research," *In: Communications Magazine, IEEE 35.1, pp. 80–86. ISSN: 0163-6804. DOI: 10.1109/35.568214.,* 1997.

[14] P. Newman, G. Minshall, and T. L. Lyon, „Ip switching—atm under ip," *IEEE/ACM Trans. Netw., vol. 6, no. 2, pp. 117–129,* Apr.1998.

[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, „NOX: towards an operating system for networks," *Comp. Comm. Rev.,* 2008.

[16] [Interactiv]. Available: https://nsrc.org/workshops/2014/nznog-sdn/raw-attachment/wiki/WikiStart/Ryu.pdf.

[17] Srivastava, I.S. A. K. Singh and S., „A survey and classification of controller placement problem in sdn," *International Journal of Network Management,* 2018.

[18] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, „Applying nox to the datacenter," *in*

HotNets, 2009.

[19] D. Zeng, C. Teng, L. Gu, H. Yao, and Q. Liang, „Flow setup time aware minimum cost switch-controller association in software-defined networks," *in Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conferenceon. IEEE, pp. 259–264,* 2015.

[20] Murat Karakus, Arjan Durresi, „Quality of Service (QoS) in Software Defined Networking (SDN): A Survey," J. Netw. Comput. Appl. 80, C, pp. 200–218., February 2017. [Interactiv]. Available: https://doi.org/10.1016/j.jnca.2016.12.019.

[21] B. Isong, RRS Molose, AM Abu-Mahfouz și N. Dladlu, „Comprehensive Review of SDN Controller Placement Strategies," *în IEEE Access, vol. 8, p. 170070-170092,* 2020.

[22] V. Ahmadi, M. Khorramizadeh, „An adaptive heuristic for multi-objective controller placement in software-defined networks," *Computers & Electrical Engineering. 66. 10.1016/j.compeleceng.2017.12.043.,* 2017.

[23] P.-T. Tivig, A. Brumaru, S. G. Obreja, „Creating Scalable Distributed Control Plane In SDN To Rule Out The Single POINT Of Failure," *COMMS,* 2022.

[24] B. Eugen, A. Tudor, M. Vochina , „Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes," *The Fifteenth International Conference on Networks ICN 2016,* February 2016.