UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE POLITEHNICA BUCUREȘTI

Şcoala Doctorală de Electronică, Telecomunicații și Tehnologia Informației

Decizie nr. 93 din 05-10-2023

REZUMAT TEZĂ DE DOCTORAT

Ing. Cosmin Andrei CONȚU

MANAGEMENTUL RESURSELOR ÎN SOLUȚII DE VIRTUALIZARE PENTRU REȚELE ȘI SERVICII

RESOURCE MANAGEMENT IN VIRTUALIZATION SOLUTIONS FOR NETWORKS AND SERVICES

COMISIA DE DOCTORAT

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

BUCUREŞTI 2023

Cuprins

1. Introducere 4
   1.1. Prezentarea domeniului de doctorat 5
   1.2. Scopul tezei 6
   1.3. Conţinutul tezei 6
2. Utilizarea ML în network slicing 8
   2.1. Concepte și instrumente 8
   2.2. Biblioteci și funcții 9
      2.2.1. Biblioteci ML cu sursă deschisă 9
      2.2.2. Funcții de pierdere și optimizare 10
   2.3. Experiment de compararea a funcțiilor de optimizare 10
   2.4. ML în network slicing 11
3. Cadrul de Lucru SONATA 12
   3.1. Arhitectură 12
   3.2. Comparație cu modelul ETSI NFV 13
   3.3. Experimente cu emulatorul SONATA 14
4. Cadrul de Lucru Open Source MANO 15
   4.1. Arhitectură 15
      4.1.1. Componente funcționale 16
   4.2. Paralelă cu SONATA 16
   4.3. Compararea performanţelor sistemelor NFV MANO cu sursă deschisă 17
   4.4. Cazuri de utilizare 18
   4.5. Detectarea unui prag optim al alarmei folosind ML 18
      4.5.1. Generarea unei alarme 18
      4.5.2. Componenta Predictor 19
   4.6. Implementări 20
5. Platforma de automatizare 21
   5.1. Arhitectură 21
   5.2. Interfața grafică 22
   5.3. Experimente 23
6. Concluzii 24
   6.1. Rezultate obţinute 24
   6.2. Contribuţii originale 25

6.3. Lista lucrărilor originale ……..... 26
6.3.1. Lista proiectelor de cercetare ……..…..... 29
6.4. Perspective de dezvoltare ulterioară ……..... 29
Bibliografie ……..... 30

Capitolul 1

Introducere

În industria telecomunicațiilor există o cerere foarte mare de bandă de transmisie mobilă (broadband) datorată în mare parte de necesității de livrare a conținutului video de calitate foarte înaltă. Rețelele 5G vin în întâmpinarea acestei probleme, încercând, în același timp, să susțină dezvoltarea mai multor industrii, oferind infrastructura necesară dezvoltării și utilizării de aplicații care au drept cerințe o latență foarte mică și o fiabilitate cât mai mare a rețelei. Pentru a putea satisface toate aceste servicii și cerințe, rețelele 5G au nevoie de o flexibilitate ridicată la nivel de arhitectură [1].

Virtualizarea Funcțiilor de Rețea (Network Functions Virtualization – NFV) reprezintă un concept de arhitectură în dezvoltare. Acesta are ca scop rezolvarea unor provocări și limitări existe în industria telecomunicațiilor, precum numărul mare de mașini hardware patentate, dedicate pentru servicii specifice, costuri ridicate de investiție (Capital Expenditure – CAPEX) și operaționale (Operational Expenditure – OPEX), consumul mare de energie electrică și problemele la nivel de spațiu de depozitare a mașinilor hardware precum și lipsa flexibilității sau a interoperabilității [2][3].

În prezent, NFV reprezintă și o tehnologie suport în domenii de tip Cloud sau Edge Computing. NFV decuplează mașinile hardware de funcțiile de rețea care rulează pe acestea, folosind mașini hardware generice și rulând funcțiile de rețea prin intermediul unor mașini virtuale instalate pe aceste echipamente generice. Bazându-se pe tehnologii de virtualizare, NFV permite o dezvoltare și implementare mult mai rapidă a serviciilor ce conțin funcții de rețea care pot fi virtualizate (în comparație cu serviciile compuse din funcții de rețea fizice). Funcțiile de rețea virtualizate (Virtual Network Functions -VNFs) pot fi definite, instanțiate, implementate sau mutate, folosind aceeași infrastructura. Acestea pot fi create, modificate sau șterse fără a fi nevoie de o prezență fizică în locul în care se află mașinile hardware pe care rulează aceste funcții. Un alt avantaj este acela al reducerii costurilor de tip CAPEX și OPEX datorită industriei IT în creștere care permite dezvoltarea de software. De asemenea, se poate obține o micșorare a consumului de energie electrică prin intermediul unui plan de management și migrare a mașinilor virtuale.

Rețele Definite prin Software (Software Defined Networking – SDN) reprezintă o tehnologie complementară cu NFV. Conceptul SDN este acela al

separării planului de control de planul de date / al utilizatorului. Acest concept permite o flexibilitate mare, programabilitate și o abstractizare a rețelei, îmbunătățind funcțiile de management și control. Cu toate ca SDN și NFV sunt două tehnologii complet independente una față de cealaltă, acestea pot fi folosite împreună pentru a crea sisteme puternice și flexibile în domeniul telecomunicațiilor. Dintr-un punct de vedere general, aceste două tehnologii pot fi considerate ca fiind ortogolane, întrucât SDN separă planul de control de cel de date iar NFV poate fi utilizat atât în planul de date cât și în cel de control pentru a implementa diverse funcții virtuale de rețea.

Conceptul de Network Slicing 5G se bazează pe virtualizare și programabilitate, permițând o modularitate a asigurării resurselor de rețea, adaptate la diverse cerințe de sistem la nivel de lățime de bandă, latență, mobilitate, etc. [4-8]. Dintr-o pespectivă generală, un "slice" (Network Slice – NSL) reprezintă un grup logic administrat, alcătuit din subseturi ale resurselor, funcții fizice și/sau virtualizate de rețea, plasate, din punct de vedere arhitectural într-un Plan de Date (Data Plane – DPl), un Plan de Control (Control Plane – CPl) și un Plan de Management (Management Plane – MPl). Slice-ul este programabil și are abilitatea de a-și expune capabilitățile utilizatorilor. Tehnologiile NFV [9-11] și SDN pot fi folosite împreună [12] pentru a administra și controla mediul 5G dintr-un slice, într-un mod flexibil și programabil.

În sistemele 5G, un operator poate diviza fizic traficul de rețea, poate împărți o rețea sau poate împărți resursele mai multor rețele combinate. Această flexibilitate le permite operatorilor să selecteze caracteristicile dorite care sunt solicitate de un client, cum ar fi volumul de trafic sau densitatea conexiunii. După cum a fost definit de proiectul de parteneriat de a treia generație (3rd generation partnership project - 3GPP) [13], există trei tipuri generale de slice-uri de rețea, fiecare corespunzând unui caz de utilizare 5G. Slice-uri de bandă largă mobilă îmbunătățită (enhanced mobile broadband - eMBB), care pot oferi viteze mari de transfer de date și o întârziere moderată. Slice-uri de comunicații ultra-fiabile cu întârziere redusă (ultra-reliable low latency communications - URLLC) care asigură transmisii de foarte mare fiabilitate cu o întârziere extrem de redusă. Slice-uri de comunicații masive de tip mașină (massive machine-type communications - mMTC) care susțin conexiunile de mare densitate la Internetul lucrurilor (Internet of things – IoT).

1.1 Prezentarea domeniului tezei de doctorat

Având în vedere dezvoltarea foarte mare a industriei de telecomunicații și implementarea rețelelor 5G din ultimii ani, paradigme precum SDN și NFV au putut fi exploatate, oferind soluții viabile la probleme curente ale industriei. Folosind aceste tehnologii, conceptul de network slicing poate aduce mari beneficii precum modularitatea asigurării resurselor de rețea și alocarea unor subseturi ale acestora în grupuri logice, adaptate în funcție de cerințele clienților la nivel de latență, lățime de bandă, etc.

O etapă foarte importantă în virtualizarea unei rețele o reprezintă orchestrarea și administrarea acesteia. Din acest motiv, cadrele SONATA și Open Source MANO au fost studiate în această lucrare și utilizate ulterior în experimente.

Alte tehnologii prezentate în această lucrare sunt cele de dezvoltare a unei platforme web (librăria React), standardul de comunicație REST API precum și algoritmi de machine learning.

1.2 Scopul tezei de doctorat

Această lucrare își propune prezentarea paradigmelor NFV și Network Slicing și diverse cazuri de utilizare ale acestora. Studiul a două cadre de lucru NFV precum SONATA și OSM completează studiul acestor paradigme. Pentru o mai bună înțelegere a modului de funcționare a acestor cadre, diverse experimente au fost realizate.

După analizarea rezultatelor obținute în urma acestor experimente, unul dintre cele două cadre de lucru a fost ales pentru a fi integrat într-un sistem autonom care să permită configurarea inteligentă de noi slice-uri prin intermediul algoritmilor de "machine learning" (ML). Sistemul autonom a fost integrat într-o platformă prin intermediul căreia s-a realizat un nou lanț de servicii, permițând realizarea de experimente în vederea validării acestei implementări. Acest sistem autonom precum și platforma care îl cuprinde reprezintă contribuții originale ale autorului, atât la nivel de arhitectură, cât și la nivel de implementare și testare.

Utilizând această platformă automatizată, implementarea unui slice prin lanțul de servicii Open Source MANO – manager de infrastructură virtuală devine mult mai rapidă și mai ușoară. Adăugând blocul de configurare inteligentă, se optimizează slice-urile atât la nivel de implementare și instalare pe infrastructura potrivită, cât și modificarea adaptivă a pragurilor de alarme și scalabilitate.

Soluția propusă în această lucrare își propune să îmbunătățească fluxul de lucru actual al administrării slice-urilor prin mecanisme automate care să poată aloca resurse în mod optim și să adapteze pragurile de alarmă și scalare în vederea optimizării funcționării slice-urilor.

1.3 Conţinutul tezei de doctorat

Lucrarea este împărțită în șase capitole. Primul capitol reprezintă o introducere în domeniul tezei de doctorat iar în ultimul sunt prezentate concluziile acestei lucrări. Celelalte capitole sunt prezentate în continuare.

Capitolul 2 conține o prezentare pe larg a conceptelor și instrumentelor de machine learning utilizate în cazuri specifice slice-urilor. Conceptele prezentate vor sta la baza capitolelor următoare. De asemenea, este realizată o comparație a funcțiilor

de optimizare cu ajutorul unui experiment precum și o trecere în revistă a altor proiecte care studiază utilizarea machine learning în network slicing.

Capitolul 3 prezintă cadrul de lucru SONATA, plecând de la o trecere în revistă a altor proiecte asemănătoare și continuând cu arhitectura sa detaliată și o comparație a acesteia cu modeul ETSI NFV. Mai departe, o serie de experimente utilizând emulatorul SONATA sunt prezentate.

Capitolul 4 tratează cadrul de lucru Open Source MANO începând cu arhitectura acestuia. O paralelă între acest cadru de lucru și SONATA este trecută în revistă pentru a pune în evidență arhitectura mai complexă a Open Source MANO. O analiză a performanțelor platformelor cu sursă deschisă este realizată și diferite cazuri de utilizare și proiecte care utilizează OSM sunt prezentate. În continuare este propusă o îmbunătățire a arhitecturii la modulului de monitorizare prin detectarea unui prag optim al alarmei cu ajutorul algoritmilor ML iar în final sunt prezentate o serie de implementări împărțite în diverse scenarii.

Capitolul 5 este dedicat Platformei de Automatizare. Aceasta este o componentă nouă adăugată unui lanț de servicii existent (Open Source MANO și un manager de infrastructură virtuală) în vederea îmbunătățirii procesului de implementare și administrare a slice-urilor. Este prezentată arhitectura acestei platforme și rolul său în noul lanț de servicii, precum și interfața grafică a acesteia împreună cu elementele constitutive. O serie de experimente ce acoperă diverse scenarii sunt prezentate în ultima parte a capitolului.

Capitolul 6 conține concluziile acestei lucrări în care sunt prezentate rezultatele obținute în cadrul multiplelor experimente efectuate, contribuțiile originale ale autorului și aportul acestora în vederea îmbunătățirii sistemelor existente. De asemenea, sunt prezentate listele de lucrări și proiectele ale autorului, precum și perspectivele de dezvoltare ulterioară are contribuțiilor acestuia.

Capitolul 2

Utilizarea ML în network slicing

Scopul acestui capitol este prezentarea și înțelegerea tehnologiilor de machine learning și aplicabilitatea lor în domeniul network slicing. Experimentul realizat a facilitat alegerea unei funcții de optimizare și realizarea unui model de predicție care a fost utilizat în capitolele următoare.

ML este o ramură a inteligenței artificiale (Artificial Intelligence – AI) ce permite mașinilor să învețe și să își îmbunătățească performanțele în timp, folosind algoritmi în continuă evoluție și date de intrare (date de antrenament, date din viața reală). În timp ce în programarea tradițională, rezultatul va fi obținut pe baza datelor de intrare și a algoritmului (regulilor) deja definit, algoritmii ML reprezintă un proces adaptiv care este capabil să genereze un rezultat prin învățarea din experiența anterioară.

2.1 Concepte și instrumente

Tehnicile ML sunt utilizate astăzi într-un număr foarte mare de domenii. Un exemplu reprezentativ de aplicație ML este recunoașterea imaginilor, care este capabilă să determine tipul unei imagini pe baza unui set de date de antrenare și poate ajunge până la recunoașterea unor modele specifice, cum ar fi obiecte sau persoane. Predicțiile sunt alte cazuri frecvente de utilizare a ML și sunt aplicate în diverse domenii, de la predicția prețurilor bunurilor imobiliare până la predicția modelelor de trafic. Recomandările de produse, recunoașterea vocală, detectarea programelor dăunătoare sau prelucrarea limbajului natural sunt alte cazuri de utilizare bine cunoscute [14]. Algoritmii și tehnicile ML sunt din ce în ce mai mult utilizate în rețelele și în operațiunile de gestionare și control al serviciilor (rezervarea și alocarea resurselor, prognoza traficului, gestionarea mobilității etc.).

O clasificare a metodelor de ML ar putea fi:

Învățare supravegheată (Supervised learning - SML) - aceste metode prezic una sau mai multe variabile dependente, pe baza unor date etichetate (inițial).

Învățarea semi-supravegheată (Semi-supervised learning - SSML) reprezintă cazurile în care nu toate datele sunt etichetate.

Învățare nesupravegheată (Unsupervised learning - UML) - mașina caută structura în seturile de date (neetichetate).

Învățarea prin consolidare (Reinforcement learning - RL) este a patra metodă de învățare majoră în ML, alături de învățarea supravegheată, nesupravegheată și semisupravegheată. Cu toate acestea, modelul RL nu are nevoie de multe date pentru a fi antrenat. Acesta învață structuri prin recompensarea pentru comportamentele dorite și pedepsirea pentru cele rele.

Învățare profundă (Deep learning - DL) - utilizează o serie de straturi de unități de procesare neliniare pentru extragerea și transformarea caracteristicilor. Modelele DL pot fi supravegheate, semisupravegheate sau nesupravegheate (sau o combinație a celor trei). DL se bazează pe rețele neuronale (NN), care imită modul de funcționare al creierului uman [15].

RL are probleme în cazurile cu spații de stare mari, deoarece trebuie să traverseze fiecare stare și să obțină o funcție de valoare sau un model pentru fiecare pereche stare-acțiune într-un mod direct și explicit. Au fost elaborate propuneri de eșantionare doar a unor părți din stări și apoi de aplicare a NN pentru a antrena o funcție sau un model suficient de precise. Rezultatul este Învățarea Profundă prin Consolidare (Deep Reinforcement Learning – DRL), care expune o stabilitate suficientă a performanțelor. Pe scurt, DRL este o combinație de RL și DL, unde RL definește obiectivul, iar DL oferă mecanismul.

2.2 Biblioteci și funcții

ML este o ramură a inteligenței artificiale (Artificial Intelligence – AI) ce permite mașinilor să învețe și să își îmbunătățească performanțele în timp, folosind algoritmi în continuă evoluție și date de intrare (date de antrenament, date din viața reală). În timp ce în programarea tradițională, rezultatul va fi obținut pe baza datelor de intrare și a algoritmului (regulilor) deja definit, algoritmii ML reprezintă un proces adaptiv care este capabil să genereze un rezultat prin învățarea din experiența anterioară.

2.2.1 Biblioteci ML cu Sursă Deschisă

Există numeroase biblioteci ML cu sursă deschisă disponibile, care sunt utilizate în prezent în dezvoltarea aplicațiilor. Aceste biblioteci sunt capabile să furnizeze modele predefinite și posibilități extinse de personalizare.

Scikit-learn [16] este una dintre cele mai populare și robuste biblioteci ML care oferă o gamă largă de instrumente și modele pentru dezvoltarea ML în Python. PyTorch [17] este un cadru cu sursă deschisă care este utilizat în mod frecvent în dezvoltarea învățării profunde, cum ar fi recunoașterea imaginilor sau procesarea limbajului. Natural Language Toolkit (NLTK) [18] este o platformă importantă pentru dezvoltarea de aplicații Python de procesare a limbajului natural. Acesta oferă o gamă

largă de resurse pentru recunoașterea vocală, codificare, clasificarea etc. TensorFlow [19] este o bibliotecă ML cu sursă deschisă, dezvoltată de Google, axată pe învățarea profundă și pe proiecte de machine learning la scară largă. TensorFlow oferă o gamă largă de modele ML/DL și algoritmi implementați în diferite limbaje de programare, cum ar fi Python sau Javascript.

2.2.2 Funcții De Pierdere Și Optimizare

Pentru a putea începe antrenarea unui model, acesta trebuie mai întâi să fie creat prin asamblarea unuia sau mai multor straturi într-un model. După crearea și asamblarea acestuia ca model secvențial, modelul trebuie compilat. În etapa de compilare, modelul primește două funcții: o funcție de pierdere care este responsabilă pentru măsurarea diferenței dintre predicții și rezultatul așteptat și o funcție de optimizare care urmărește să reducă pierderea prin ajustarea valorilor interne până când obține cea mai precisă formă posibilă pentru model. Ambele funcții sunt apelate în timpul etapei de antrenare pentru a calcula pierderea pentru fiecare etapă și pentru a o îmbunătăți.

În timp ce eroarea medie pătratică (Mean Squared Error) [20] este cea mai utilizată funcție de pierdere, există mai multe funcții de optimizare care pot fi utilizate.

Algoritmul de gradient adaptiv (Adagrad) [21] este o metodă de optimizare cu coborâre a gradientului în loturi mici care adaptează rata de învățare prin includerea cunoștințelor din experiența anterioară. Estimarea adaptivă a momentului (Adaptive Moment Estimation - Adam) [22] este un algoritm de optimizare a coborârii gradientului care utilizează algoritmul momentului pentru a accelera algoritmul de coborâre a gradientului prin utilizarea mediei ponderate exponențial a gradienților și a algoritmului de învățare adaptivă de propagare a mediei pătratice (RMSProp) care utilizează media mobilă exponențială. RMSProp [23] funcționează în același mod ca și Momentum prin creșterea ratei de învățare, folosind pași mai mari pentru direcțiile care converg mai repede. Diferența dintre cele două optimizatoare constă în modul în care sunt calculați gradienții.

2.3 Experiment de Comparare a Funcțiilor de Optimizare

Pentru a compara funcțiile de optimizare, a fost luat în considerare rezultatul unui scurt experiment realizat în [24]. Utilizând un mic set de date cu valori x și y care pot fi descrise de ecuația liniară 1.1:

$$y = 2x + 40 \tag{1.1}$$

Utilizând un algoritm ML, ar trebui să se găsească aceeași ecuație între valorile x și y. Valorile x vor reprezenta datele de intrare pentru modelul ML, în timp ce valorile y vor fi etichetele sau rezultatele pe care modelul ar trebui să le prezică. După pregătirea datelor (și împărțirea lor în date de antrenare și de testare), modelul va fi creat cu un singur strat și un singur neuron (deoarece trebuie rezolvată o ecuație liniară simplă). Modelul va fi asamblat și compilat cu o funcție de pierdere și o funcție de optimizare. Pentru a putea compara funcțiile de optimizare, modelul va fi antrenat cu aceeași funcție de pierdere, aceleași date de antrenament și același număr de epoci, dar cu o funcție de optimizare diferită de fiecare dată. Ca ultimă etapă, acuratețea modelului va fi măsurată cu ajutorul unui parametru de performanță (Figura 2.1).

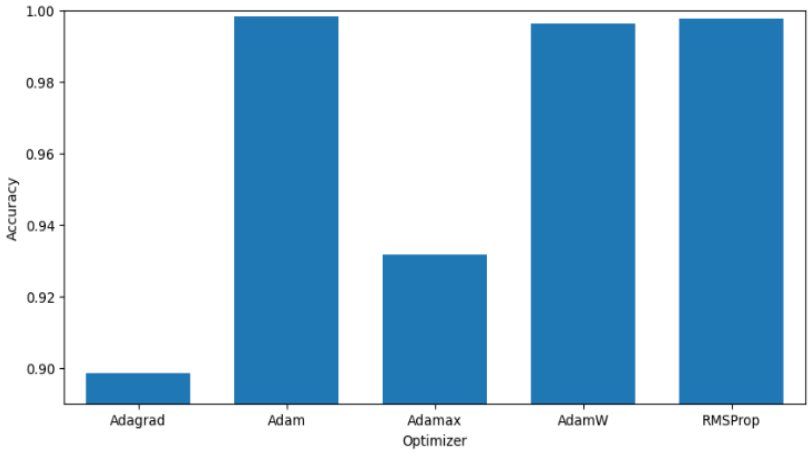

Figura 2.1 *Rezultatele comparării funcțiilor de optimizare*

Rezultatele obținute [25] indică faptul că funcția de optimizare Adam a avut cea mai bună precizie (0,9981), urmată de RMSProps (0,9975) și AdamW (0,9963). Un pic mai puțin precise au fost Adamax (0,9317) și Adagrad (0,8986). Pe baza acestor rezultate și având în vedere că, dintre funcțiile de optimizare cu o precizie calculată de peste 99%, Adam are cel mai mic timp de calcul, optimizatorul Adam a fost ales pentru a fi utilizat în continuare în lucrarea de față.

2.4 ML în network slicing

Există o mulțime de studii și implementări existente ale tehnicilor de machine learning în domeniul network slicing. Studiile acoperă mai multe aspecte legate de modul de aplicare a algoritmilor ML în cazuri de utilizare a network slicing, cum ar fi prezicerea potențialelor amenințări, alocarea resurselor și prognozarea traficului.

Capitolul 3

Cadrul de Lucru SONATA

În acest capitol a fost studiată capacitatea cadrului SONATA de a orchestra și administra funcții virtuale de rețea într-un mediu NFV aliniat la modelului ETSI. Efectuarea experimentelor din cadrul acestui capitol au realizat un punct important de început în vederea înțelegerii ciclului de viață al unui serviciu de rețea precum și comunicația dintre un orchestrator și un manager virtual de rețea.

Proiectul EU H2020 SONATA: Service Programming and Orchestration for Virtualized Software Networks [26] își propune dezvoltarea unui cadru NFV care să asigure dezvoltatorilor externi (terți) un model de programare și un set de instrumente pentru dezvoltarea de servicii virtualizate integrate cu un sistem de orchestrare. SONATA oferă posibilitatea dezvoltatorilor software de a obține un timp scăzut de lansare pe piață a unui serviciu de rețea (Network Service – NS), de a optimiza și a reduce costurile de implementare și de a micșora timpul de integrare a rețelelor software în industria telecomunicațiilor.

Din perspectiva SONATA, OpenStack este o platformă complementară. Dezvoltatorii SONATA au nevoie de acces la o instanță funcțională de OpenStack pentru a-i utiliza funcționalitatea de manager de infrastructură virtuală pentru a rula serviciilor din platforma de servicii.

O altă opțiune pentru dezvoltatorii de servicii SONATA, este aceea de a utiliza emulatorul SONATA pentru a dezvolta și testa diverse servicii de rețea înlănțuite în diverse scenarii. Emulatorul SONATA oferă interfețe asemănătoare cu cele din OpenStack pentru a permite unui orchestrator (SONATA, Open Source MANO) să controleze managerul de infrastructură virtuală emulat.

3.1 Arhitectură

Arhitectura generală a cadrului SONATA (Figura 3.1), care este construit pe baza modelului de arhitectură al NFV MANO oferit de ETSI, este constituită din următoarele componente [27]:

- Platforma de Servicii (Service Platform – SP)
- Setul de Dezvoltare Software (Software Development Kit – SDK)
- Cataloage

Platforma de servicii (Figura 3.2) recepționează pachetele de servicii create prin intermediul setului de dezvoltare software și este responsabilă de plasarea, implementarea, provizionarea, scalarea și gestionarea serviciilor pe infrastructurile cloud existente. Componenta responsabilă de prelucrarea cererilor primite și transmise este modulul portar (gatekeeper). Platforma de servicii este responsabilă și de furnizarea de feedback direct către setul de dezvoltare software cu privire la serviciile implementate, cum ar fi, de exemplu, date de monitorizare a unui serviciu sau a componentelor sale. De aseemenea, aceasta a fost proiectată având în vedere posibilitatea de a o personaliza complet, oferind astfel atât flexibilitate cât și control operatorilor și dezvoltatorilor.

O imagine mai detaliată a arhitecturii cadrului SONATA și modul în care comunică componentele între ele se poate observa în Figura 3.4.

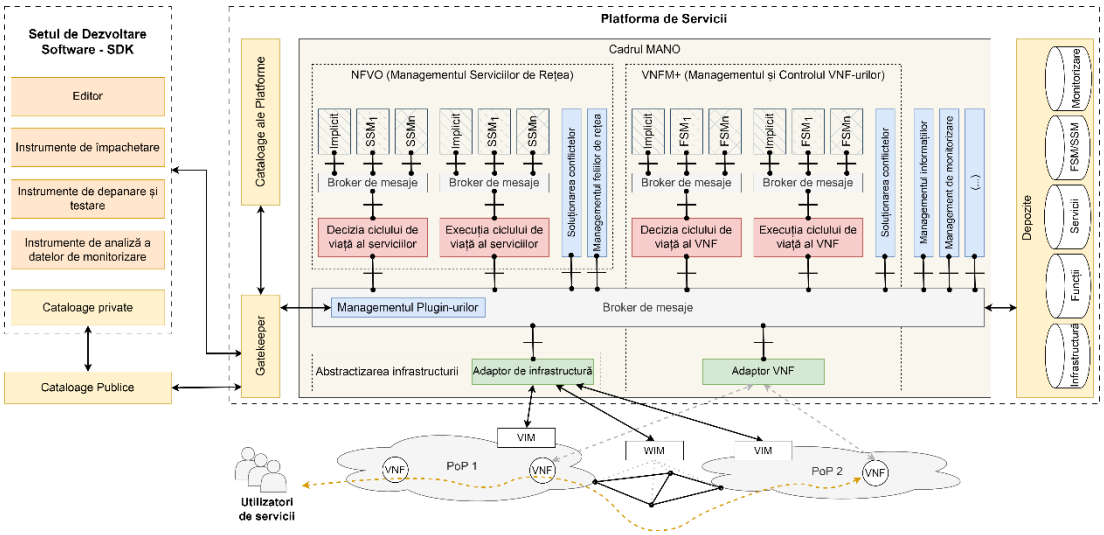

Figura 3.4 *Arhitectura detaliată a cadrului SONATA [28].*

3.2 Comparație cu modelul ETSI NFV

Cadrul MANO este elementul principal al platformei și asigură managementul pentru întregul ciclu de viață al serviciilor de rețea complexe. Acesta are aceleași elemente constitutive precum și blocul ETSI-MANO (orchestratorul NFV și managerul VNF).

Atât în SONATA cât și în modelul ETSI, managerul infrastructurii virtuale controlează și administrează resursele virtualizate de calcul, stocare și rețea în domeniul infrastructurii operatorului. Un manager de infrastructură virtuală poate controla un anumit tip sau mai multe tiprui de resurse din cadrul infrastructurii NFV.

De asemenea, față de modelul ETSI-MANO, în SONATA se mai află și componenta gatekeeper care este responsabilă de validarea serviciilor de rețea încărcate în platformă sub formă de pachete, prin intermediul unei medieri între operațiuni de dezvoltare și operaționale [29].

Așadar, platforma de servicii urmează modelul ETSI NFV dar, în același timp, adaugă noi extensii proprii care ușurează suportul pentru clienți multiplii prin intermediul slicing-ului resurselor care pot fi alocate exclusiv unui client specific.

Comparativ cu modelul ETSI, SONATA a adăugat setul de dezvoltare software ca o componentă foarte importantă din cadrul arhitecturii propuse. Acest set vine în ajutorul dezvoltatorilor terți care doresc să dezvolte servicii complexe care să conțină multiple funcții virtualizate de rețea, cu o varietate de instrumente software dar și cu un suport pentru implementarea și administrarea serviciilor de rețea create pe diferite platforme de servicii SONATA.

3.3 Experimente cu emulatorul SONATA

Aceste experimente au reprezentat un prim pas al înțelegerii modului de funcționare a unui orchestrator și a ciclului de viață al unei funcții virtuale de rețea. Pornind de la topologii simple precum comunicația dintre două mașini tip client, până la topologii cu funcții înlănțuite (client, router, firewall, etc.), aceste experimente au permis o bună înțelegere a virtualizării și managementului funcțiilor de rețea într-un mediu NFV. Aceste contribuții au stat la baza experimentelor ulterioare din cadrul acestei lucrări.

Topologiile utilizate în cadrul acestor experimente sunt reprezentate drept rețele emulate care utilizează recipiente (containers) Docker [30] ca instanțe pe care să ruleze funcții virtuale de rețea.

Experimentele realizate:

- **Topologie cu mașini virtuale simple de tip client**: Obiectivul acestui experiment este acela de a crea două mașini virtuale de tip client și de a testa comunicația dintre cele două.
- **Topologie cu server HTTP**: Obiectivul acestui experiment este acela de a crea o mașină virtuală de tip server HTTP și o mașină virtuală de tip client pentru a testa funcționalitatea serverului virtual HTTP.
- **Topologie cu firewall virtual**: Obiectivul acestui experiment este acela de a crea o mașină virtuală de tip firewall și două mașini virtuale de tip client pentru a testa funcționalitatea mașinii virtuale de tip firewall, care va permite sau bloca un anumit tip de trafic dintre cele două mașini client.
- **Topologie cu routere virtuale**: Obiectivul acestui experiment este acela de a crea o rețea de mașini virtuale de tip router care va dirija traficul dintre trei mașini virtuale de tip client aflate în trei rețele diferite.
- **Topologie cu funcții virtualizate de rețea înlănțuite**: Obiectivul acestui experiment este acela de a crea o topologie care să instanțieze o înlănțuire de diverse funcții virtualizate de rețea. Mașinile virtuale utilizate în această topologie vor fi de tip client, router, firewall, server proxy și server HTTP.

Capitolul 4

Cadrul de Lucru Open Source MANO

În acest capitol a fost studiată arhitectura cadrului OSM comparativ cu cadrul SONATA, precum și performanțele acestuia. Pentru o mai bună înțelegere a modului de funcționare a OSM, o serie de implementări cu diverse scenarii au fost realizate. De asemenea, a fost prezentată o propunere de îmbunătățire a arhitecturii utilizând algoritmi de ML în vederea detectării unui prag optim al alarmei.

Open Source MANO (OSM) este un proiect sprijinit de către ETSI care își propune dezvoltarea unui pachet software de management și orchestrare aliniat cu arhitectura NFV prezentată de ETSI. OSM este bazat pe un model multistrat, unde fiecare strat conține un serviciu compus din serviciile straturilor inferioare. Folosind această abordare, OSM dorește să ofere o mai bună integrare a serviciilor cu un efort minim. Acest lucru este obținut prin utilizarea unui model de informații (Figura 4.1) bazat pe modelul NFV oferit de ETSI, care este capabil să modeleze și să automatizeze ciclul de viață al funcțiilor de rețea.

4.1 Arhitectură

Din punct de vedere arhitectural [31], OSM este format din mai multe module. Fiecare dintre acestea are roluri specifice și este decuplat unul de celălalt (Figura 4.2). NBI este un modul important care consumă informațiile de la alte module. NBI oferă o comunicare prin intermediul API-urilor de tip transfer de stare de reprezentare (Representational State Transfer – REST), care este consumată de diferiți clienți, cum ar fi clientul CLI. CLI este un client OSM furnizat ca parte a instalării. În plus, există și un client de GUI care permite utilizarea directă a OSM. Alte tipuri de clienți care pot interacționa cu NBI sunt sistemele de asistență pentru operațiuni și afaceri (OSS/BSS). Clientul de VIM va fi responsabil de gestionarea resurselor (calcul, spațiu de stocare și rețea) ale infrastructurii NFV.

.

4.1.1 Componente funcționale

Principalele componente ale sistemului sunt LCM, modulul de Configurare și Abstractizare VNF (VNF Configuration and Abstraction – VCA) și Orchestratorul de Resurse (Resource Orchestrator – RO). Atunci când se face o nouă cerere de instanțiere, NBI înregistrează în baza de date MongoDB o intrare care precizează că trebuie creată o nouă instanță și transmite controlul către LCM, care este responsabil de operațiunile de instanțiere și asigură orchestrarea de la un capăt la altul. În primul rând, în cazul în care implementarea se face pe un manager de infrastructură, LCM va interacționa cu acel VIM pentru a crea rețele și a asigura conectivitatea între mașinile virtuale din acel VIM. De asemenea, acesta interacționează cu controlerele SDN care sunt, ca VIM, componente externe cu care OSM interacționează prin intermediul modulului RO, și implementează acele mașini virtuale.

După instalarea software-ului OSM pe un server local sau la distanță, trebuie să se stabilească o conexiune cu un VIM.

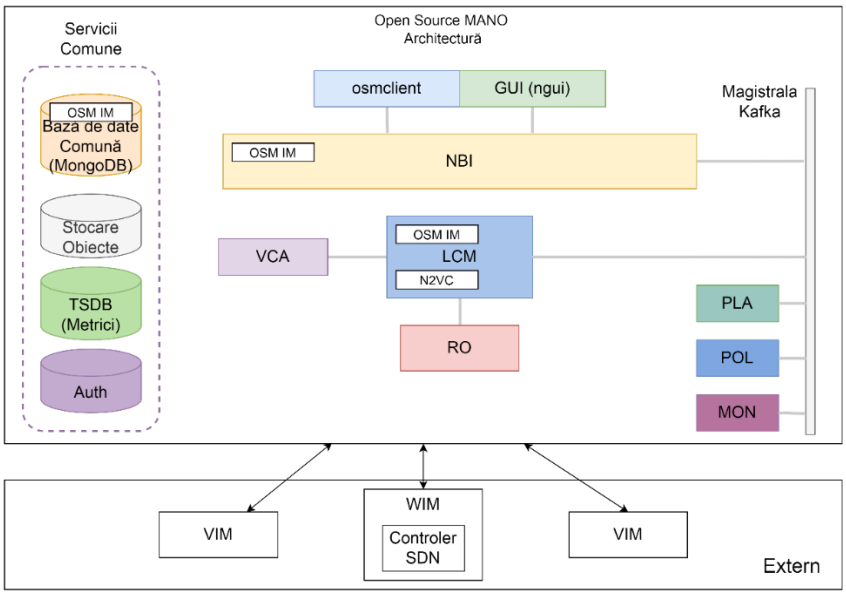

Figura 4.2 *Arhitectura OSM. [31].*

4.2 Paralelă cu SONATA

Acest sub-capitol prezintă o perspectivă selectivă asupra unor lucrări conexe dedicate dezvoltării și orchestrării de servicii în rețele virtualizate și relația acestora cu arhitectura OSM. De asemenea, prezintă și cadrul SONATA care face parte din cadrul OSM.

SONATA se află într-o relație directă cu OSM datorită integrării emulatorului SONATA din modulul SDK în OSM DevOps, care a făcut parte din a treia versiune a

OSM. Acest emulator asigură o integrare mai ușoară cu stive MANO datorită API-urilor sale, care seamănă cii API-urile oferite de Openstack [32].

Componenta SP din cadrul SONATA poate fi considerată ca o alternativă la OSM. Aceasta conține elemente precum un modul portar, un cadru MANO, un manager de slice-uri, abstractizarea infrastructurii, cataloage, depozite, un manager de politici, un manager de Acord la Nivel de Servicii (Service Level Agreement – SLA), manager de monitorizare și un portal.

SONATA este mai mult un colaborator al OSM decât un competitor, având diferite componente care sunt deja compatibile cu cadrul OSM sau care sunt ușor de integrat cu acesta. De asemenea, SONATA este menționată în diverse scenarii în lucrări publicate ale OSM. Un tip de scenariu este cel în care operatorul dorește să coopereze cu un alt operator pentru a furniza serviciul de rețea. De exemplu, acesta poate transfera furnizarea infrastructurii sau a unui VNF specializat către un alt operator. Implicația este că OSM are nevoie de o arhitectură care să permită orchestrarea orchestratorilor, iar proiectul SONATA contribuie la acest lucru prin cadrul său MANO [33].

4.3 Compararea performanțelor sistemelor NFV MANO cu sursă deschisă

Proiectele de tip MANO cu sursă deschisă precum Open Network Automation Platform (ONAP) [34], OSM, Open Baton [35], Cloudify [36], OPNFV [37], se află în diferite stadii de dezvoltare constantă. Printre aceste proiecte mai proeminente se numără Open Network Automation Platform (ONAP) și Open Source MANO (OSM), care au câștigat multă atenție din partea comunității operatorilor, în special datorită patronajului unor mari operatori din spatele dezvoltării acestora. De exemplu, ONAP, care este dezvoltată sub egida Linux Foundation, este susținută în principal de AT&T, în timp ce OSM este condusă de Telefonica și este dezvoltată de grupul Open Source Group (OSG) al ETSI, recent înființat.

Atât ONAP, cât și OSM se află în diferite stadii de lansare, dar sunt departe de a fi complete sau stabile. Ambele își propun să ofere un cadru integrat NFV MANO, dar urmează direcții foarte diferite în ceea ce privește arhitectura și implementarea. Datorită faptului că acestea sunt dezvoltate relativ recent, există foarte puține informații și experiență disponibile în ceea ce privește capacitățile funcționale și operaționale ale acestor platforme și nivelul de pregătire tehnologică. În plus, realizarea unei analize comparative a performanțelor sistemelor MANO reprezintă în sine o provocare. Acest lucru se datorează faptului că, spre deosebire de alte entități de rețea tradiționale, care au Indicatori Cheie de Performanță (Key Performance Indicators – KPI) bine definiți pentru a evalua performanța, nu există KPI stabiliți și bine definiți pe baza cărora să poată fi evaluată performanța unui sistem MANO.

4.4 Cazuri de utilizare

Comunitatea OSM reunește o mulțime de proiecte de cercetare care implementează orchestrarea NFV prin utilizarea stivei OSM. Cele mai multe dintre aceste proiecte sunt orientate spre 5G și folosesc Openstack ca VIM.

Proiectul 5GCity [38] construiește o platformă care permite utilizarea infrastructurii de tehnologie a informației și comunicațiilor a unui oraș în medii de tip cloud-to-edge. 5GTango [39] este un proiect 5GPPP care oferă o programabilitate flexibilă a unei rețele 5G cu mai multe componente, cum ar fi un SDK orientat spre NFV, o platformă de stocare VNF/NS cu mecanisme de validare și verificare și o platformă de servicii modulare. Proiectul Metro-Haul are ca scop proiectarea rețelelor metropolitane (agile, programabile și cu un consum de energie și costuri reduse) care ar trebui să fie compatibile cu 5G, incluzând proiectarea nodurilor metropolitane complet optice, inclusiv capacități de stocare și de calcul [40]. Proiectul MATILDA implementează un cadru operațional de servicii 5G E2E, axat pe ciclul de viață al aplicațiilor pregătite pentru 5G și al serviciilor de rețea 5G (proiectare, dezvoltare și orchestrare) atât pe infrastructura virtuală, cât și pe cea fizică, prin utilizarea unui model de programabilitate unificat [41].

4.5 Detectarea unui prag optim al alarmei folosind ML

Componenta de monitorizare (Figura 4.4) este responsabilă pentru colectarea de metrici ale Unității de Implementare Virtuală (Virtual Deployment Unit – VDU) furnizate de VIM, de metrici specifice VNF prin intermediul charm-urilor Juju și de metrici ale infrastructurii. De asemenea, aceasta se ocupă de stocarea valorilor acestor metrici în TSDB din Prometheus și gestionează și evaluează alarmele. Modulul MON conține trei module: un server, un colector și un evaluator [42].

Modulul POL (Figura 4.11) a fost proiectat pentru procesul de autoscalare și se ocupă de ascultarea sau configurarea alarmelor, de trimiterea mesajelor de scalare și de apelarea de metode tip webhooks atunci când sunt îndeplinite politicile de alarmă [42].

Evaluatorul MON va evalua pragurile de măsurare specifice, iar apoi POL va lua măsurile necesare, cum ar fi autocalibrarea. Ori de câte ori se declanșează o alarmă, MON va genera o notificare și o va trimite prin intermediul magistralei Kafka pentru ca alte componente, cum ar fi POL, să poată consuma mesajul respectiv.

4.5.1 Generarea unei alarme

Descriptorul de alarmă face, de asemenea, parte dintr-un descriptor VNF și specifică ce metrici ar trebui monitorizate, care dintre ele ar trebui să fie deja definite în lista de monitorizare, pragurile care trebuie monitorizate și metoda de tip webhook care va fi invocată.

Arhitectura actuală permite utilizatorilor OSM să aleagă singuri pragurile de alarmă, fără a avea niciun fel de informații sau cunoștințe despre care ar trebui să fie

valoarea corespunzătoare unui prag. Acest lucru nu ar trebui să reprezinte o problemă pentru utilizatorii experimentați, dar pentru utilizatorii noi sau pentru cazurile de utilizare noi, acest lucru ar putea ridica unele probleme în alegerea unui prag de alarmă adecvat.

Pentru a depăși acest neajuns, în această lucrare a fost propusă o îmbunătățire a arhitecturii. Mai exact, se introduce în modulul de monitorizare o a patra componentă denumită Predictor (prezentat în Figura 4.13), care aplică un algoritm de machine learning pentru a prezice un prag de alarmă adecvat pe baza datelor existente din baza de date comună (MongoDB).

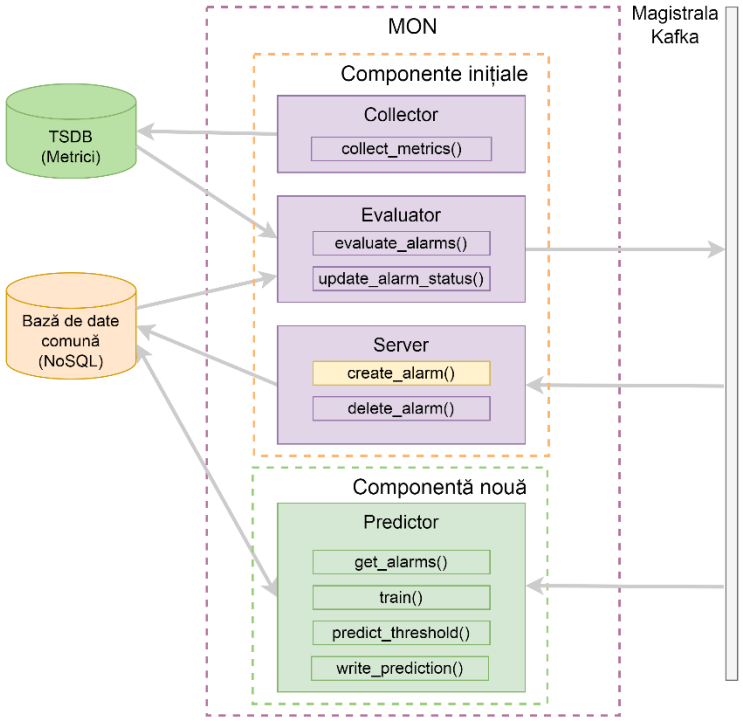

Figura 4.13 *Propunere de proces cu componenta Predictor integrată.*

4.5.2 Componenta Predictor

Componenta Predictor va fi responsabilă de generarea de sugestii pentru pragurile de alarmă și pentru a prezice cea mai bună valoare a pragului pentru cazuri de utilizare specifice.

Această nouă componentă va conține patru procese principale:

- get_alarms() care va interoga baza de date comună și va prelua toate alarmele stocate.
- train() care va fi un proces în fundal programat pentru antrenarea modelului de predicție la intervale de timp specifice.
- predict_threshold() este procesul care va apela funcția de predicție a modelului pentru a obține valoarea prezisă.
- write_prediction() este procesul final care este responsabil pentru scrierea pragului prezis în baza de date comună.

Componenta Predictor va asculta pe magistrala Kafka pentru noua cerere de creare a alarmei (topic: alarm_request; key: create_alarm_request) și, ori de câte ori acest mesaj este citit, va declanșa începerea procesului predict_threshold().
Deoarece Predictorul adaugă o valoare de prag prezisă la înregistrarea de alarmă nou creată, sunt necesare unele modificări la nivelul modelului din baza de date comună. Modelul de alarmă va conține un câmp suplimentar numit "prag_propus" de tip float. Această modificare se reflectă, de asemenea, în schema magistralei de mesaje în care este implicată structura de alarmă. În acest fel, un prag de alarmă îmbunătățit poate fi stabilit automat, independent de experiența anterioară a utilizatorului OSM, îmbunătățind alte procese OSM, cum ar fi repararea automată (auto-healing) și asigurând o mai bună fiabilitate.

4.6 Implementări

Atunci când se creează noi NS care conțin VNF personalizate, este necesar să se urmeze anumiți pași pentru a obține un serviciu de rețea adecvat și funcțional, gata de implementare. O bună practică atunci când se creează noi servicii de rețea și funcții de rețea virtuale este de a crea diagrame pentru ambele înainte de a începe să se creeze descriptorul asociat acestora.

O serie de implementari au fost prezentate care acoperă următoarele scenarii:

- NS cu un singur VNF
- NS cu multiple VNF și VDU
- NS cu configurare de tip Day0
- Slice de rețea

Capitolul 5

Platforma de automatizare

În acest capitol a fost prezentată o platformă de automatizare care permite îmbunătățirea și simplificarea procesului de implementare a slice-urilor de rețea prin OSM. Au fost trecute în revistă arhitectura acesteia, interfața grafică și modul ei de operare cât și experimente care au acoperit diverse scenarii, evidențiind funcționalitățile platformei.

Pe măsură ce aplicațiile web au evoluat de la simple pagini web la sisteme foarte complexe, bibliotecile și cadrele de lucru au fost îmbunătățite pentru a oferi dezvoltatorilor web soluții cât mai calitative. Aplicațiile web moderne plasează, de obicei, logica pe partea de client în loc de server și preiau date în mod dinamic de la un API al serverului [43].

React este o bibliotecă JavaScript cu sursă deschisă dezvoltată de către Facebook și utilizată pentru proiectarea de interfețe pentru utilizator [44]. Biblioteca a fost publicată în 2013 și astăzi se numără printre cele mai populare instrumente utilizate pentru dezvoltarea web de front-end.

5.1 Arhitectură

OSM oferă posibilitatea de a crea slice-uri de rețea prin utilizarea pachetului lor software care necesită un VIM pentru a crea mașinile virtuale necesare. Comunicarea dintre OSM și VIM se realizează prin trimiterea de mesaje prin intermediul protocolului de transfer hipertext (Hypertext Transfer Protocol - HTTP) către API-urile VIM-ului. În ceea ce privește procesul de creare de slice-uri de rețea cu ajutorul OSM, există trei modalități diferite:

- Prin UI,
- prin comenzi CLI,
- Prin OSM NBI API.

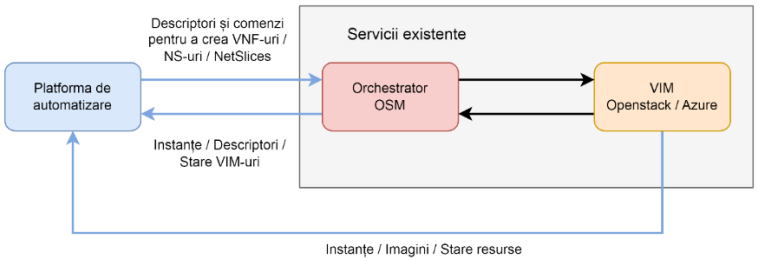

Figura 5.1 *Fluxul de lucru al platformei de automatizare.*

Platforma de automatizare este o aplicație web dezvoltată cu biblioteca ReactJS, care utilizează apeluri HTTP pentru a comunica cu OSM și cu Openstack. Interfața sa este stilizată prin intermediul bibliotecii Material UI.

Arhitectura (Figura 5.2) acestei aplicații web conține mai multe componente și servicii. Fiecare componentă reprezintă implementarea grafică a fiecărei pagini afișate în interfața cu utilizatorul. Serviciile au următoarele roluri:

- Autentificare: inițiază comunicarea inițială atât cu OSM, cât și cu Openstack pentru a obține token-uri de acces care vor fi folosite de celelalte două servicii atunci când vor solicita diferite informații.
- Serviciul OSM: se ocupă de toate solicitările către OSM NBI pentru a prelua toate datele necesare.
- Serviciul Openstack: se ocupă de toate cererile către Openstack API, care va transmite datele solicitate, cum ar fi lista de imagini disponibile etc.

5.2 Interfața grafică

În cele ce urmează, este prezentat modul de utilizare și funcționare a platformei de automatizare pentru diferite scenarii. Această interfață oferă utilizatorului o serie de acțiuni precum: crearea unei unui slice nou de rețea, afișarea instanțelor de slice-uri, afișarea șabloanelor de slice-uri sau afișare VIM-uri disponibile.

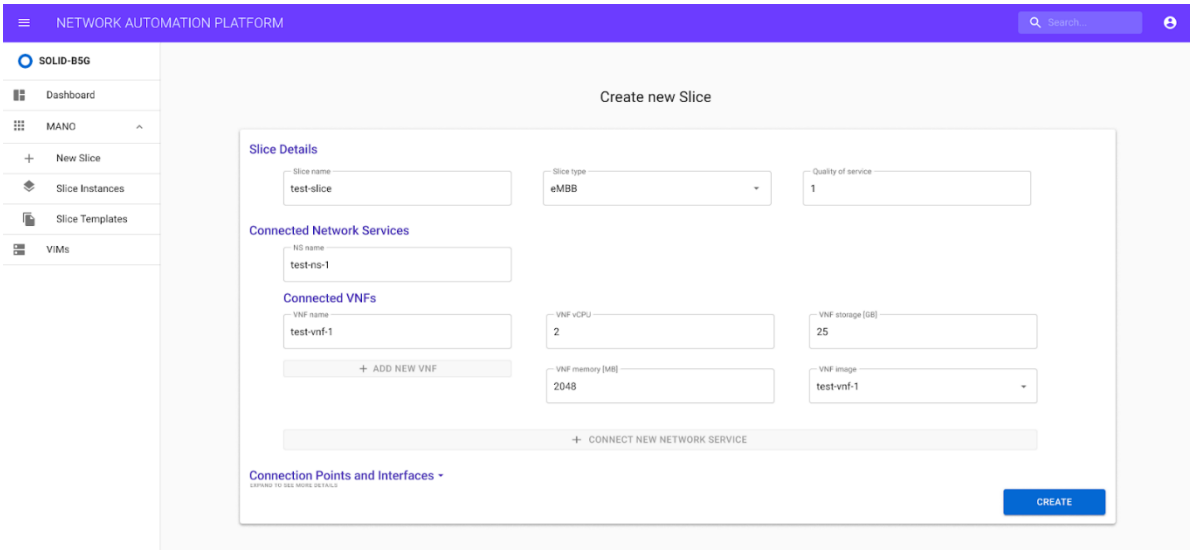

Figura 5.3 *Pagina Slice de Rețea Nou a Platformei de Automatizare.*

5.3 Experimente

Pentru a realiza implementarea automată a slice-ului, este necesară configurarea mai multor parametri (procesare, stocare, memorie, imagine etc.) pentru fiecare VNF și pentru fiecare NS conținute de acest slice. Acesta este un pas important pentru a dezvolta o platformă capabilă să automatizeze și să simplifice procesul de implementare a unui slice prin intermediul unui sistem de MANO cu sursă deschisă (OSM) [45].

De asemenea, crearea unui slice se poate realiza și pornind de la un caz de utilizare generic. Fiecare serviciu de rețea din slice va fi preconfigurat cu ajutorul scripturilor cloud-init [46] pe mașinile instanțiate. Instanțierea va avea loc pe un VIM specific, potrivit pentru tipul respectiv de slice de rețea.

O nouă funcționalitate a fost integrată în platforma de automatizare pentru a oferi o mai bună alocare a resurselor. Componenta Resource Prediction (RP) este capabilă să optimizeze un slice prin furnizarea de valori prognozate pentru anumiți parametri, cum ar fi CPU, memorie, stocare. Modelul utilizat a fost cel prezentat în secțiunea 2.3, cu un strat dens cu un singur neuron care a fost compilat cu o funcție de pierdere de eroare medie pătratică și cu optimizatorul Adam cu rată de învățare de 0,1, modelul fiind antrenat timp de 1000 de epoci. Setul de date a fost construit prin utilizarea datelor anterioare de la slice-uri implementate anterior prin intermediul platformei de automatizare. Chiar dacă dimensiunea setului de date nu este foarte mare, fiecare nouă implementare de slice va adăuga noi date la setul de date, îmbunătățindu-l. Antetele setului de date sunt: Slice_Type, Use_Case, NS_Name, VNF_Name, vCPU, Memory și Storage. În prezent, antrenarea este inițiată manual, dar o funcție de antrenare programată va fi adăugată ca pas următor.

Au fost efectuate trei experimente distincte pentru a testa funcționalități diferite ale platformei:

- **Slice de rețea predefinită pentru transmisiune video în flux de înaltă definiție**: slice-ul a fost creată pornind de la un caz de utilizare, așadar, pentru VNF-uri, conținutul util va conține și scriptul cloud-init care va instala serverul media necesar în cazul de transmisiune în flux de tip ultra HD.
- **Slice de rețea specializată pentru calcul de înaltă performanță** – un slice de rețea specializată care să susțină simulări paralele pentru probleme de mari dimensiuni la scară și viteză, oferind calcul de înaltă performanță.
- **Slice de rețea de tip eMBB cu parametri optimizați prin ML** - pentru a testa componenta RP, slice-ul general de tip eMBB a fost ales ca scenariu, deoarece setul de date actual conține în mare parte date de implementare a slice-ului eMBB, iar modelul ar trebui să producă valori mai precise.

Capitolul 6

Concluzii

În cadrul acestei lucrări au fost studiate paradigmele NFV și Network Slicing precum și diverse cazuri de utilizare ale acestora. Acest studiu a continuat prin utilizarea și evaluarea a două cadre de lucru MANO cu ajutorul cărora au fost realizate diverse experimente. În urma evaluării acestora, cadrul OSM a fost ales pentru a continua studiul și a-l integra într-o nouă platformă de automatizare a slice-urilor de rețea.

De asemenea, au fost prezentate avantaje ale tehnicilor de ML și posibilitatea de utilizare a acestora în scenarii de network slicing. Pe această bază, a fost propusă o contribuție arhitecturală la nivelul modulului de monitorizare OSM, care va spori eficiența și flexibilitatea acestuia. Noul sub-modul propus va fi implicat în fluxul de lucru al alarmelor, generând o recomandare a valorii pragului de alarmă pentru fiecare parametru pentru care se configurează o alarmă. Valoarea recomandată a pragului de alarmă este generată prin utilizarea unui algoritm de predicție ML.

Același tip de algoritm este utilizat într-un bloc de predicție a resurselor care este implementat și testat într-o platformă de automatizare. Prin intermediul acestui nou bloc, ori de câte ori un utilizator va începe procesul de creare a unui nou slice prin intermediul platformei de automatizare, pentru fiecare tip de slice sau caz de utilizare, valorile recomandate vor fi afișate ca spații de intrare, permițându-i utilizatorului să configureze valori mai bune ale resurselor pe baza datelor de desfășurare a slice-urilor anterioare.

Utilizând această platformă automatizată, implementarea unui slice prin lanțul de servicii Open Source MANO – VIM devine mult mai rapidă și mai ușoară. Adăugând blocul de predicție a resurselor, se optimizează slice-urile atât la nivel de implementare și instalare pe infrastructura potrivită, cât și modificarea adaptivă a pragurilor de alarme și scalabilitate. Datorită capacității pe care o are platforma de a prelua date despre slice-urile și serviciile instanțiate din OSM și Openstack, operatorii de servicii vor avea o vedere mai bună asupra acestor slice-uri și servicii înainte de implementarea slice-ului.

Soluțiile propuse în această lucrare își propun să îmbunătățească fluxul de lucru actual al administrării slice-urilor de rețea prin mecanisme automate care să poată aloca resurse în mod optim și să adapteze pragurile de alarmă și scalare în vederea optimizării funcționării slice-urilor de rețea.

6.1 Rezultate obţinute

Primul capitol reprezintă o introducere în domeniul tezei de doctorat precum și a

scopului și conținutului acesteia.

Capitolul 2 conține o prezentare pe larg a conceptelor și instrumentelor de machine learning utilizate în cazuri specifice slice-urilor de rețea, o comparație a funcțiilor de optimizare cu ajutorul unui experiment precum și o trecere în revistă a altor proiecte care studiază utilizarea machine learning în network slicing. Acest capitol a avut ca rezultat o mai bună înțelegere a tehnologiilor de machine learning și aplicabilitatea lor în domeniul network slicing. Rezultatele experimentale au facilitat alegerea unei funcții de optimizare și realizarea unui model de predicție care a fost utilizat în capitolele următoare.

Capitolul 3 prezintă cadrul de lucru SONATA, plecând de la o trecere în revistă a altor proiecte asemănătoare și continuând cu arhitectura sa detaliată și o comparație a acesteia cu modeul ETSI NFV. Mai departe, o serie de experimente utilizând emulatorul SONATA sunt prezentate. Rezultatele acestui capitol au fost înțelegerea ciclului de viață al unui serviciu de rețea precum și comunicația dintre un orchestrator și un manager virtual de rețea prin intermediul experimenteleor, reprezentând un punct important de început a înțelegerii modului de funcționare a unui cadru de lucru MANO.

Capitolul 4 tratează cadrul de lucru Open Source MANO începând cu arhitectura acestuia. O paralelă între acest cadru de lucru și SONATA este trecută în revistă pentru a pune în evidență arhitectura mai complexă a Open Source MANO. O analiză a performanțelor platformelor cu sursă deschisă este realizată și diferite cazuri de utilizare și proiecte care utilizează OSM sunt prezentate. A fost propusă o îmbunătățire a arhitecturii la nivelul modulului de monitorizare prin detectarea unui prag optim al alarmei cu ajutorul algoritmilor ML iar în final realizate implementări împărțite în diverse scenarii. Ca rezultate ale acestui capitol, pot fi menționate înțelegerea arhitecturii OSM, care a condus la o propunere de îmbunătățire a arhitecturii utilizând algoritmi de ML în vederea detectării unui prag optim al alarmei.

Capitolul 5 introduce platforma de automatizare. Aceasta este o componentă nouă adăugată unui lanț de servicii existent (Open Source MANO și un manager de infrastructură virtuală) în vederea îmbunătățirii procesului de implementare și administrare a slice-urilor. Este prezentată arhitectura acestei platforme și rolul său în noul lanț de servicii, precum și interfața grafică a acesteia împreună cu elementele constitutive. O serie de experimente ce acoperă diverse scenarii sunt prezentate în ultima parte a capitolului. Rezultatele capitolului sunt definirea arhitecturii platformei de automatizare, implementarea acesteia și adăugarea de blocuri de automatizare și predicție a resurselor. Implementarea platformei a fost validată prin efectuarea unor experimente care au acoperit diverse scenarii, evidențiind funcționalitățile acesteia.

6.2 Contribuţii originale

Contribuţiile originale rezultate pe parcursul activității de cercetare din cadrul stagiului doctoral sunt următoarele:

1. Studiu sintetic al paradigmelor NFV și Network Slicing [6.3 – 2];

2. Studiu sintetic al conceptelor și instrumentelor ML aplicabile în domeniul network slicing [6.3 – 1];

3. Rularea experimentului de comparare a funcțiilor de optimizare și interpretarea rezultatelor acestuia [6.3 – 1];

4. Studiu sintetic al componentelor arhitecturale ale cadrului de lucru SONATA [6.3 – 6,7,10];

5. Realizarea unei paralele între cadrul de lucru SONATA și modelul ETSI NFV [6.3 – 6,7,10];

6. Realizarea a multiple experimente utilizând emulatorul SONATA [6.3 – 6,7,10];

7. Studiu sintetic al componentelor arhitecturale ale cadrului de lucru OSM [6.3 – 19];

8. Compararea cadrului OSM cu SONATA la nivel de arhitectură și cu ONAP la nivel de performanță [6.3 – 19];

9. Propunerea unui nou sub-modul de detectare a unui prag optim al alarmei cu ajutorul algoritmilor ML în cadrului arhitecturii modulului de monitorizare OSM [6.3 – 1];

10. Definirea arhitecturii platformei de automatizare [6.3 – 3];

11. Proiectarea interfeței grafice a platformei de automatizare [6.3 – 3];

12. Implementarea platformei de automatizare ca aplicație web [6.3 – 2];

13. Implementarea și integrarea unui bloc de predicție a resurselor în platforma de automatizare [6.3 – 1];

14. Validarea funcționalităților platformei de automatizare prin intermediul unor experimente [6.3 – 1,2];

6.3 Lista lucrărilor originale

Pe parcursul activității de cercetare din cadrul stagiului doctoral, autorul a publicat un număr de 14 lucrări științifice legate de domeniul tezei de doctorat, 6 ca prim autor și 6 indexate în IEEEXplore. 5 rapoarte științifice au fost de asemenea adăugate în lista de mai jos. Pe parcursul stagiului doctoral, autorul a fost implicat și în două proiecte de cercetare.

1. **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin, Frank Y. Li and Alexandru Aloman, *Machine Learning-based Monitoring in Network Slice Creation and Resource Management*, IEEE Access, SUBMITTED (**ISI)**

2. **Cosmin Conțu**, Andra Ciobanu, Eugen Borcoci, Marius-Constantin Vochin, Indika A. M. Balapuwaduge, Silviu Topoloi and Razvan-Florentin Trifan, *Deploying Use Case Specific Network Slices Using An OSM Automation Platform*, 25th International Symposium on Wireless Personal Multimedia Communications (WPMC) 2022, Herning, Denmark, October 30 – November 2, 2022 (**IEEEXplore**);

3. **Cosmin Conțu**, Andra Ciobanu, Eugen Borcoci, Marius-Constantin Vochin, Frank Y. Li, *An Automation Platform for Slice Creation using Open Source MANO*, Proceeding of The 14th International Conference on COMMUNICATIONS, COMM 2022, Bucharest, ROMANIA, June 16-18, 2022, ISBN: 978-1-6654-9485-4 (**ISI, IEEEXplore**);

4. **C. Conțu**, I. Cioarcă, M. Ene, L. Nichifor, *Study on Optimal Convolutional Neural Networks Architecture for Traffic Sign Classification Using Augmented Dataset*, Proceeding of The 13th International Conference on COMMUNICATIONS, COMM 2020, Bucharest, ROMANIA, June 18-20, 2020, ISBN: 978-1-7281-5611-8 (**ISI, IEEEXplore**);

5. Andra-Isabela-Elena Ciobanu, **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin, and Frank Y. Li, *Optimal Service Placement with QoS Monitoring in NFV and Slicing Enabled 5G IoT Networks*, 2021 IEEE Globecom Workshops (GC Wkshps): IEEE IoST-5G&B: 2nd Workshop on Recent Trends of Internet of Softwarized Things - 5G & B, Madrid, Spain, 7–11 December 2021 (**IEEEXplore**);

6. A. Țapu, **C. Conțu**, E. Borcoci, *Multiple Chained Virtual Network Functions Experiments with SONATA Emulator*, Proceeding of The 12th International Conference on COMMUNICATIONS, COMM 2018, Bucharest, ROMANIA, June 14-16, 2018, ISBN: 978-1-5386-2350-3 (**ISI, IEEEXplore**);

7. A. Țapu, **C. Conțu**, E. Borcoci, *Network Function Virtualization Experiments using SONATA Framework*, Proceeding of The International Symposium on Advances in Software Defined Networking and Network Functions Virtualization SOFTNETWORKING 2018, Athens, GREECE, April 22-26, 2018, ISBN: 978-1-61208-625-5 (**Best Paper Award**);

8. A. Ciobanu, **C. Conțu**, E. Borcoci, *Study on Use-Cases of Open Source Management and Orchestration Framework in 5G Projects*, Proceeding of The Nineteenth International Conference on Networks 2020, Lisbon, Portugal, February 23-27, 2020, ISBN: 978-1-61208-770-2 (**Best Paper Award**);

9. A. Ciobanu, **C. Conțu**, E. Borcoci, *Charms and Virtual Network Functions Primitives Experiments using Open Source MANO Framework*, Proceeding of The Sixteenth Advanced International Conference on Telecommunications AICT 2020, Lisbon, Portugal, September 27 – October 1, 2020, ISBN: 978-1-61208-802-0 (**Best Paper Award**);

10. **C. Conțu**, A. Țapu, E. Borcoci, *Virtual Network Function Use Cases Implemented on SONATA Framework*, International Journal on Advances in Networks and Services, issn 1942-2644 vol. 11, no. 3 & 4, year 2018, 103:112;

11. E. Borcoci, A. Ciobanu, **C. Conțu**, *Layered Network Domain Resource Management in Multi-domain 5G Slicing Environment*, Proceeding of Advanced Information and Communication Technologies-2019, Nice, France, July 28-August 2, 2019, ISBN: 978-1-61208-727-6;

12. E. Borcoci, **C. Conțu**, A. Ciobanu, *5G Slicing Management and Orchestration Architectures - Any Convergence?,* Proceeding of The Eleventh International Conference on Advances in Future Internet AFIN 2019, Nice, France, October 27-31, 2019, ISBN: 978-1-61208-747-4;

13. E. Borcoci, **C. Conțu**, A. Ciobanu, *On Heterogeneity of Management and Orchestration Functional Architectures in 5G Slicing*, International Journal on Advances in Internet Technology, vol 13 no 1 & 2, year 2020, 83-96;

14. **Cosmin Conțu**, Eugen Borcoci, Marius-Constantin Vochin and Frank Y. Li, *Automating Network Slices in Open Source MANO*, Proceedings of the Doctoral Symposium on Electronics, Telecommunications, & Information Technology SD-ETTI 2023, October 4-6, 2023, Bucharest **(DBLP)**

15. Eugen Borcoci, Andra Țapu, **Cosmin Conțu**, *D1.2 Virtual Evolved Packet Core*, ORANGE Romania – UPB cooperation project "5G Technologies", 2018;

16. **C. Conțu**, *Experimente de Virtualizare a Funcțiilor de Rețea utilizând Framework-ul SONATA*, Raport Științific Nr. 1 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2018;

17. **C. Conțu**, *Cazuri de Funcții Virtualizate de Rețea Înlănțuite Implementate utilizând Framework-ul SONATA*, Raport Științific Nr. 2 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Decembrie 2018;

18. **C. Conțu**, *Cazuri de Funcții Virtualizate de Rețea Implementate pe Infrastructură Fizică utilizând Framework-ul Open Source Management and Orchestration (MANO),* Raport Științific Nr. 3 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2019;

19. **C. Conțu**, *Managementul Funcțiilor Virtualizate de Rețea și Implementarea unui "Slice" de Rețea Utilizând Framework-ul Open Source Management and Orchestration (MANO) – Studii de caz*, Raport Științific Nr. 4 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Decembrie 2019;

20. **C. Conțu**, *Despre Eterogenitatea Administrării și Orchestrării a Arhitecturilor Funcționale în Felierea Rețelelor 5G*, Raport Științific Nr. 5 (nepublicat), Universitatea Politehnica din Bucureşti, Romania, Iunie 2020;

6.3.1 Lista proiectelor de cercetare

1. **Februarie – Mai 2018**: *SLICENET: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks* (colaborare între Universitatea Politehnica din Bucureşti și Orange Romania);

2. **Noiembrie 2020 – Decembrie 2023**: *A Massive MIMO Enabled IoT Platform with Networking Slicing for Beyond 5G IoV/V2X and Maritime Services*;

6.4 Perspective de dezvoltare ulterioară

Platforma de automatizare prezentată în această lucrare a reușit să simplifice procesul de implementare a unei slice de rețea și să ofere operatorului un control și o perspectivă mai bune asupra ciclului de viață al slice-ului de rețea.

Ca perspective de dezvoltare ulterioară, atât calitatea cât și cantitatea seturilor de date o sa necesite îmbunătățire, faza de instruire a blocului de predicție a resurselor va fi automatizată, iar platforma de automatizare va fi extinsă cu mai multe caracteristici. De asemenea, actualizări periodice vor fi necesare pentru a menține compatibilitatea cu noile versiuni ale OSM prin colaborarea cu comunitatea OSM, unde organizația autorului este înregistrată ca Participant oficial.

O altă direcție de dezvoltare a platformei este aceea de a-i mări compatibilitatea cu mai multe tipuri de VIM deja compatibile cu OSM (Amazon Web Services, Microsoft Azure și Google Cloud Platform).

Nu în ultimul rând, vor fi studiate noi cazuri de utilizare ale algoritmilor ML în cadrul platformei și se vor implementa noi blocuri pentru a susține aceste cazuri.

Bibliografie

[1] 5G PPP Architecture Working Group, *View on 5G Architecture*, Version 1.0, July 2016. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf [Online]

[2] NFV White paper: *Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1*. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf, 2018 [Online]

[3] R. Mijumbi et al., *Network function virtualization: State-of-the-art and research challenges*, IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 236-262, 1st Quart. 2016

[4] 5GPPP Architecture Working Group, *View on 5G Architecture*, Version 3.0, June, 2019, Available: https://5gppp.eu/wp-content/uploads/2019/07/5G-PPP-5GArchitecture-White-Paper_v3.0_PublicConsultation.pdf, [Online].

[5] J. Ordonez-Lucena et al., *Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges*, IEEE Communications Magazine, 2017, pp. 80-87, Citation information: DOI 10.1109/MCOM.2017.1600935.

[6] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, *Network Slicing in 5G: Survey and Challenges*, IEEE Communications Magazine, May 2017, pp. 94-100

[7] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, *Network Slicing & Softwarization: A Survey on Principles, Enabling Technologies & Solutions*, IEEE Communications Surveys & Tutorials, March 2018, pp. 2429-2453.

[8] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, *On Multi-domain Network Slicing Orchestration Architecture & Federated Resource Control*, Available: http://mosaic-lab.org/uploads/papers/3f772f2d-9e0f-4329-9298-aae4ef8ded65.pdf, 2019 [Online].

[9] ETSI GS NFV 002, *NFV Architectural Framework*, V1.2.1, December, 2014

[10] ETSI GS NFV-IFA 009, *Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options*, Technical Report, V1.1.1, July, 2016

[11] ETSI GR NFV-IFA 028, *Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains*, Technical Report, V3.1.1, January, 2018.

[12] ONF TR-526, *Applying SDN Architecture to 5G Slicing*, April 2016

[13] 3GPP, "System architecture for the 5G System (5GS)," TS 23.501, R17, v17.5.0, Jun. 2022.

[14] *An Introduction to Machine Learning*,. Available: https://monkeylearn.com/machine-learning/ 2023. [Online]

[15] *Deep learning vs. Machine learning vs. Artificial Intelligence* Available: https://www.javatpoint.com/deep-learning-vs-machine-learning-vs-artificial-intelligence/, 2023. [Online].

[16] *scikit-learn Machine Learning in Python,* Available: https://scikit-learn.org/stable/, 2023. [Online].

[17] *PyTorch,* Available from: https://pytorch.org/, 2023. [Online].

[18] *NLTK :: Natural Language Toolkit,* Available: https://www.nltk.org/, 2023. [Online].

[19] *TensorFlow,* Available: https://www.tensorflow.org, 2023. [Online].

[20] *Mean squared error,* Available: https://en.wikipedia.org/wiki/Mean_squared_error, 2023. [Online].

[21] *AdaGrad,* Available: https://www.databricks.com/glossary/adagrad, 2023. [Online].

[22] *Intuition of Adam Optimizer,* Available: https://www.geeksforgeeks.org/intuition-of-adam-optimizer/, 2023. [Online].

[23] *A Look at Gradient Descent and RMSprop Optimizers,* Available: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b/, 2023. [Online].

[24] *Build your first Machine Learning Model using TensorFlow,* Available: https://techwithshadab.medium.com/build-your-first-machine-learning-model-using-tensorflow-d61b9b2b7d5e/, 2023. [Online].

[25] C. Conțu, I. Cioarcă, M. Ene, L. Nichifor, *Study on Optimal Convolutional Neural Networks Architecture for Traffic Sign Classification Using Augmented Dataset*, Proceeding of The 13th International Conference on COMMUNICATIONS, COMM 2020, Bucharest, ROMANIA, June 18-20, 2020, ISBN: 978-1-7281-5611-8

[26] S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris, *Sonata: Service programming and orchestration for virtualized software networks*, in 2017 IEEE International Conference on Communications Workshops (ICC Workshops), May 2017, pp. 973–978

[27] Sevil Draxle et al., *SONATA: Service Programming and Orchestration for Virtualized Software Networks*, 2017 IEEE International Conference on Communications Workshops (ICC Workshops).

[28] SONATA. *D2.2 Architecture Design*. Available: http://sonata-nfv.eu/sites/default/files/sonata/public/contentfiles/pages/SONATA_D2.2_Architecture_and_Design.pdf, [Online].

[29] *The SONATA Gatekeeper*, Available: http://sonata-nfv.eu/sites/default/files/sonata/public/content-files/article/SONATA_Gatekeeper_SDNWorld_3.pdf, 2018 [Online].

[30] *Docker - Build, Ship, and Run Any App, Anywhere*. Available: https://www.docker.com/, 2018 [Online].

[31] Gerardo García de Blas, *OSM architecture*, Available: https://osm-download.etsi.org/ftp/osm-11.0-eleven/OSM12-hackfest/presentations/OSM%2312%20Hackfest%20-%20OSM%20architecture.pdf, 2023. [Online].

[32] *Research – OSM Public Wiki*, Available from: https://osm.etsi.org/wikipub/index.php/Research#Sonata, 2019, [Online].

[33] A. Ciobanu, C. Conțu, E. Borcoci, *Study on Use-Cases of Open Source Management and Orchestration Framework in 5G Projects*, Proceeding of The Nineteenth International Conference on Networks 2020, Lisbon, Portugal, February 23-27, 2020, ISBN: 978-1-61208-770-2

[34] ONAP, *ONAP Architecture Overview*, Available from: https://www.onap.org/architecture, 2019, [Online].

[35] G. A. Carella and T. Magedanz, *Open baton: A framework for virtual network function management and orchestration for emerging software-based 5G networks*, IEEE Softwarization, July 2016.

[36] Cloudify, *Cloudify Orchestration Project Portal*, Available from: https://cloudify.co/, 2019, [Online].

[37] OPNFV, *Open Platform for NFV (OPNFV) Project Portal*, Available from: https://www.opnfv.org/, 2019, [Online].

[38] *5GCity – A distributed cloud & radio platform for 5G Neutral Hosts*, Available: https://www.5gcity.eu/, 2022, [Online].

[39] *5GTANGO*, Available: https://www.5gtango.eu/ 2022, [Online].

[40] *METRO-HAUL 5G Project*, Available: https://metrohaul.eu/ 2022, [Online].

[41] *MATILDA*, Available: https://www.matilda-5g.eu/ 2022, [Online].

[42] Subhankar Pal, *HD4.3 Closed-Loop Operations: Adding Auto-Scaling & Alerting to VNFs*, Available: https://osm-download.etsi.org/ftp/osm-8.0-eight/OSM-MR9-hackfest/presentations/OSM-MR%239%20Hackfest%20-%20HD4.3%20-%20Closed-Loop%20Operations.pdf, 2023. [Online].

[43] D. Johansson, *Building maintainable web applications using React*, Available: https://www.divaportal.org/smash/get/diva2:1415320/FULLTEXT01.pdf, 2022, [Online].

[44] *React – A JavaScript library for building user interfaces*, Available: https://reactjs.org/ 2022, [Online].

[45] ETSI, *Open source MANO*, Available: https://osm.etsi.org/, 2022. [Online].

[46] Canonical, *Cloud init – The standard for customising cloud instances,* Available: https://cloud-init.io/, 2022. [Online].