UNIVERSITATEA "POLITEHNICA" DIN BUCUREȘTI

ȘCOALA DOCTORALĂ ETTI-B

Nr. Decizie ..... din .....

TEZĂ DE DOCTORAT

Contribuții la dezvoltarea și implementarea rețelelor definite prin programe soft

Contributions to the development and implementation of Software-Defined Networks

Doctorand: **Ing. Liviu-Alexandru STANCU**

Conducător de doctorat: **Prof. Dr. Ing. Simona HALUNGA**

COMISIA DE DOCTORAT

Președinte	**Prof. Dr. Ing. Gheorghe BREZEANU**	de la	**Univ. "Politehnica" din Bucureşti**
Conducător de doctorat	**Prof. Dr. Ing. Simona HALUNGA**	de la	**Univ. "Politehnica" din Bucureşti**
Referent		de la	**Univ. "Politehnica" din Bucureşti**
Referent		de la	**Univ. "Politehnica" din Bucureşti**
Referent		de la	**Univ. "Politehnica" din Bucureşti**

București 2017

Mulțumiri

Cuprins

Lista tabelelor

Lista figurilor

Lista abrevierilor

DVM	Default Values Mediator
ForCES	Forwarding and Control Element Separation
HAL	Hardware Abstraction Layer
IoT	Internet of Things
MPLS	Multi-Protocol Label Switching
NAT	Network Address Translation
NCP	Network Control Point
NETCONF	Network Configuration Protocol
NVP	Network Virtualization Platform
ODL	OpenDaylight
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OT	Optical Transport
OVSDB	Open vSwitch Database Management
PoC	Proof of Concept
POF	Protocol-Oblivious Forwarding
POSIX	Portable Operating System Interface
ROFL	Revised Open Flow Library
SDN	Software-Defined Networking
TR	Technical Recommendation
VLAN	Virtual Local Area Network
WLAN	Wireless Local Area Network
WT	Wireless Transport
WTE	Wireless Transport Emulator
xDPd	eXtensible Datapath Daemon

Capitolul 1

Introducere

1.1 Prezentarea domeniului tezei de doctorat

În urma avansurilor tehnologice recente în toate domeniile, în general și în domeniul calculatoarelor și al telecomunicaţiilor, în particular, a apărut nevoia de a redefini arhitectura reţelelor de comunicaţii, din cauza faptului că reţelele tradiţionale au început să îşi arate limitele. În zilele noastre, există o tendinţă de a interconecta totul, cu ajutorul unor tehnologii care permit acest lucru, cum ar fi conceptul de *Cloud Computing*, mobilitatea, sau idei mai noi, cum ar fi *Internetul Tuturor Lucrurilor* - IoT (Internet of Things) sau sistemele de comunicaţii de generaţia a cincea - 5G. Aceste noi abordări au nevoie, pe lângă o lăţime de bandă crescută, de o reţea mai simplă și agilă, unde se facilitează inovarea. Reţelele definite prin software - SDN (Software-Defined Networking), reprezintă o nouă paradigmă care a apărut în industria reţelisticii, pentru a mitiga dezavantajele pe care reţelele tradiţionale le-au dovedit.

Tehnologia SDN nu este încă matură și nu a pătruns în toate tipurile de reţele de comunicaţii. Este prezentă în campusuri universitare, sau în centre de date, însă se încearcă introducerea acesteia în toate aspectele unei reţele de comunicaţii, cum ar fi transportul de date optic - OT (Optical Transport), transportul de date fără fir - WT (Wireless Transport) sau noduri de interconectare ale Internetului. Aceste încercări presupun muncă de standardizare și demonstraţii de concept - PoC (Proof of Concept), pentru prezentarea avantajelor pe care această nouă paradigmă de oferă, până când tehnologia se va maturiza și va fi adoptată de toată industria reţelisticii.

1.2 Scopul tezei de doctorat

Această lucrare îşi propune să prezinte noua paradigmă apărută în industria reţelisticii, SDN, împreună cu avantajele pe care această abordare le poate aduce dacă ar fi aplicată în toate aspectele unei reţele de comunicaţii, punând accent pe reţelele de transport de date fără fir. Autorul îşi va prezenta activitatea de cercetare, constând în unelte software care pot fi folosite ca simulatoare de echipamente de

transport de date fără fir, ce expun interfeţe folosite în tehnologia reţelelor definite prin software.

Aceste unelte software au fost folosite cu succes în procesul de standardizare SDN, care încă se desfăşoară în cadrul ONF (Open Networking Foundation), facilitând testarea modelelor informaţionale ce se dezvoltă în contextul reţelelor definite prin software și uşurând dezvoltarea și testarea aplicaţiilor SDN care fac parte din acest ecosistem. Simulatorul rezultat în urma acestei cercetări, în forma sa finală, poate emula o întreagă reţea de echipamente de transport de date fără fir, care expun interfeţe specifice SDN. El poate fi folosit de către dezvoltatorii de produse software SDN pentru reţele de transport de date fără fir, eliminând nevoia acestora de a deţine astfel de echipamente scumpe pentru a-și putea testa aplicaţiile. Poate fi folosit și de către operatorii care vor sa lanseze această tehnologie în reţelele de producţie, pentru a simula consecinţele instalării unor noi aplicaţii anterior lansării, fără a afecta reţeaua.

1.3 Conţinutul tezei de doctorat

Lucrarea este împărţită în opt capitole, primul prezentând domeniul abordat în teză, iar ultimul fiind dedicat concluziilor. În continuare va fi prezentat, pe scurt, conţinutul fiecărui dintre celelalte capitole.

Capitolul 2 introduce domeniul reţelelor definite prin software, plecând de la istoria sa și nevoia pentru care această nouă paradigmă a apărut. Apoi va fi ilustrată activitatea de standardizare în acest domeniu, inclusiv demonstraţiile de concept conduse de către ONF, în particular de către grupul WT, care va duce la maturizarea soluţiei și adoptarea acesteia pe scară largă, în toate aspectele unei reţele. Tot în acest capitol se va evidenţia și prezenţa SDN în contextul reţelelor actuale.

Cel de-al 3-lea capitol pune accent pe prezenţa SDN în reţelele de transport de date fără fir. Sunt prezentate modelele informaţionale dezvoltate în cadrul ONF în acest context, având rolul de recomandări tehnice - TR (Technical Recommendation): *TR-532, Microwave Information Model* și *TR-512, Core Information Model*. Ulterior se vor da detalii despre NETCONF (Network Configuration Protocol), care este protocolul de bază pentru reţelele de transport de date fără fir, în contextul SDN. În următoarea secţiune se vor compara câteva cadre software cu sursă deschisă, ce oferă facilitatea unui server NETCONF. Pe baza acestei comparaţii s-a ales unealta software care va face parte din simulatorul propus în lucrare. Capitolul va fi încheiat de o prezentare a arhitecturii demonstraţiilor de concept organizate de grupul WT din ONF, ce va ajuta la înţelegerea necesităţii unui astfel de simulator.

Capitolul 4 prezintă prima versiune a simulatorului, numită *Mediatorul cu valori implicite* - DVM (Default Values Mediator), folosită în cel de-al doilea PoC. Se vor prezenta, pe rând, arhitectura și implementarea, iar apoi se va evidenţia folosirea acestui simulator în contextul demonstraţiilor de concept.

Următorul capitol, 5, descrie cea de-a doua versiune a DVM, abordând aspecte despre arhitectura, implementare și folosire in cadrul celui de-al treilea PoC. În plus, se va evidenţia încercarea de a integra acest simulator cu o soluţie de comutator

software, LINC, folosit în SDN, în special în cadrul reţelelor de transport optic de date, prezentând avantajele și dezavantajele date de această abordare.

Capitolul 6 descrie ultima și cea mai avansată versiune a simulatorului reţelelor de transport de date fără fir - WTE (Wireless Transport Emulator), prezentând arhitectura, detaliile implementării și folosirea acestuia în cea de-a patra demonstraţie de concept a grupului WT din cadrul ONF.

Capitolul 7 ilustrează rezultatele obţinute în urma acestei cercetări și propune discuţii pe baza simulatorului implementat. În primul rând, această soluţie este evaluată din punctul de vedere al resurselor consumate și al extensibilităţii pe care o oferă. Apoi, se compară simulatorul cu alte soluţii care există în momentul de faţă în contextul SDN. Ulterior, se prezintă câteva cazuri de utilizare, propuse în cadrul grupului WT din ONF, care pot fi demonstrate cu ajutorul simulatorului, eliminând nevoia unor echipamente de transport de date fără fir.

Capitolul 2

Introducere în reţelele definite prin software

În zilele noastre, reţelele de comunicaţii au devenit complexe și greu de administrat și configurat. De asemenea, numărul dispozitivelor mobile a crescut considerabil, alături de conţinutul pe care acestea îl accesează. Aceste lucruri au dus la evidenţierea limitărilor pe care reţelele tradiţionale le presupun. Chiar dacă nu toate ideile ce stau la baza ei sunt noi, datorită unui context favorabil, acestea, împreună cu alte noi idei, au dus la apariţia paradigmei SDN în industria reţelisticii.

Această nouă tehnologie nu a ajuns încă la maturitate și la adoptarea pe scară largă, în toate aspectele reţelelor, însă eforturile considerabile care se fac în activităţile de standardizare și crearea de ecosisteme SDN vor duce la această adoptare. După cum este evidenţiat și în [1], se tinde către crearea unor reţele care pot fi programate prin software, crescând astfel flexibilitatea și agilitatea lor.

2.1 Istoria și evoluția SDN

2.1.1 Istoria SDN

Reţelele definite prin software îşi au originile în activitatea și ideile din cadrul proiectului OpenFlow, început la universitatea Stanford, în jurul anului 2009. Multe dintre conceptele și ideile folosite în SDN au evoluat însă în ultimii 25 de ani și acum îşi găsesc locul în această nouă paradigmă, care îşi propune să schimbe modul în care reţelele sunt proiectate și administrate.

SDN reprezintă o arhitectură nouă de reţea, în care starea de dirijare a planului de date este administrată de un plan de control distant, decuplat de cel de date. Reţelele definite prin software sunt definite ca fiind o arhitectură de rețea ce se bazează pe următoarele 4 concepte, conform [2]:

1. Decuplarea planurilor de date și de control;

2. Deciziile de dirijare se bazează pe fluxuri de date, nu pe adresa destinaţie;

3. Logica de control se mută într-o entitate externă, un echipament de control SDN (care are un sistem de operare de reţea);

4. Reţeaua este programabilă prin aplicaţii software care rulează peste sistemul de operare de reţea și care interacţionează cu echipamentele din planul de date.

Reţelele definite prin programe software au apărut ca o nevoie, pentru a oferi posibilitatea inovaţiei în cadrul administrării reţelelor și pentru a uşura introducerea de noi servicii. Aceste nevoi nu sunt însă noi, ele mai fiind studiate și în trecut, însă abia acum, prin SDN, pot fi satisfăcute într-un mod viabil, care să nu implice schimbări majore în infrastructura reţelelor deja existente.

Istoria SDN poate fi împărţită în trei etape, fiecare influenţând această nouă paradigmă prin conceptele propuse, aşa cum este evidenţiat în [3]:

1. Reţelele active, care au introdus funcţiile programabile în reţea, sporind gradul de inovaţie (mijlocul anilor 1990 – începutul anilor 2000);

2. Separarea planurilor de date și de control, care a condus la dezvoltarea de interfeţe deschise între planurile de date și de control (aproximativ 2001 – 2007);

3. Dezvoltarea protocolului OpenFlow și a sistemelor de operare de reţea, care reprezintă prima adoptare pe scară largă a unei interfeţe deschise, făcând separarea planurilor de control și de date extensibilă și practică.

Rețelele active

Reţelele active reprezintă reţele în care comutatoarele pot efectua anumite calcule sau operaţii asupra pachetelor de date. Reţelele tradiţionale nu pot fi considerate programabile. Reţelele active au reprezentat un concept radical asupra controlului unei reţele, propunând o interfaţă de programare care să expună resurse în noduri individuale de reţea și care să susţină construirea de funcţionalităţi specifice, care să fie aplicate unui subset de pachete care tranzitează acel nod.

Motivul principal pentru care reţelele active au apărut a fost cel de accelerare a inovației. La momentul respectiv, introducerea unui nou concept, serviciu sau tehnologie, într-o reţea de mari dimensiuni, cum ar fi Internet-ul, putea dura până la zece ani, de la faza de prototip până la implementare. Se dorea ca nodurile active din reţea să permită ruterelor/comutatoarelor să descarce servicii noi în infrastructura deja existentă. În acelaşi timp, aceste noduri active ar fi putut coexista fără probleme în aceeaşi reţea cu vechile dispozitive. Au existat două tipuri de abordări în cadrul reţelelor active, în funcţie de modelul ales pentru programarea rețelei:

- *Modelul încapsulat* – unde codul care trebuia executat în cadrul nodurilor active era transportat în bandă, în pachetele de date; fiecare pachet de date conţinea cod care trebuia rulat;

- *Modelul comutatoarelor programabile* – codul care trebuia executat în cadrul nodurilor active era stabilit prin mecanisme din afara benzii. Execuţia programelor era determinată de antetul pachetului.

Rețelele active nu au ajuns niciodată să fie implementate pe scară largă, din mai multe cauze: momentul de timp la care au apărut acestea nu a fost potrivit; la acel moment nu aveau o aplicabilitate clară, deoarece nu apăruseră încă centrele de date sau infrastructurile de tip cloud; implementarea reţelelor active avea nevoie și de suport hardware, care nu era tocmai ieftin, ceea ce a constituit încă un dezavantaj.

Chiar dacă rețelele active nu au ajuns să fie implementate pe scară largă, câteva idei au fost preluate în cadrul rețelelor definite prin programe software:

- **Funcţii programabile în rețea, care să faciliteze inovaţia.** În motivaţia introducerii rețelelor definite prin software se acuză dificultatea inovației în rețelele de producţie. Rețelele active foloseau programarea planului de date, în timp ce în SDN se programează atât planul de control cât și cel de date.

- **Virtualizarea rețelei și capacitatea de a demultiplexa programe soft pe baza antetului pachetelor.** Rețelele active au dezvoltat un cadru arhitectural care să permită funcționarea unei astfel de platforme, având drept componente de bază un sistem de operare comun (al nodurilor), un set de medii de execuţie și un set de aplicații active, care oferă de fapt un serviciu capăt-la-capăt.

- **Atenţia la aparatele de rețea și la felul în care funcţiile acestora sunt compuse.** În cercetarea din cadrul rețelelor active se vorbea despre nevoia unificării gamei largi de funcţii oferite de aparatele de rețea într-un cadru programabil sigur.

Separarea planurilor de date și de control

Rețelele, încă de la început, au avut planurile de date și de control integrate. Acest lucru a dus la câteva dezavantaje: îngreunarea sarcinilor de administrare a rețelei, de depanare a configurării rețelei sau de controlul/prezicerea comportamentului de dirijare.

Primele inițiative de separare a planurilor de control și de date datează din anii 1980. La acel moment, cei de la AT&T propuneau renunţarea la semnalizarea în bandă și introducerea unui Punct de Control al Rețelei - NCP (Network Control Point), realizând astfel separarea planurilor de control și de date. Această modificare a înlesnit accelerarea inovației în rețea, prin posibilitatea de introducere rapidă de noi servicii și a furnizat noi metode de a îmbunătăţi eficienţa, printr-o vedere de ansamblu asupra rețelei oferită de punctul de control al rețelei. Există și inițiative mai recente care și-au propus separarea planurilor de control și de date, cum ar fi ETHANE [4], NOX [5], OpenFlow [6]. Acestea au ca avantaj faptul că nu au nevoie de modificări substanţiale în echipamentele de dirijare, ceea ce înseamnă că pot fi adoptate mai ușor de către industria rețelelor.

Ideile preluate în rețelele definite prin programe software din cercetarea care propunea separarea planurilor de control și de date sunt următoarele:

- **Control logic centralizat care folosește o interfață deschisă către planul de date.** O interfață deschisă către planul de date, care să permită

inovația în aplicațiile software din planul de control a fost propusă de activitățile de cercetare din cadrul ForCES [7]. Însă, această interfață nu a fost adoptată de marile companii furnizoare de echipamente de rețea, astfel că aceasta nu a fost implementată pe scară largă.

- **Administrarea stărilor distribuite.** Controlul logic centralizat al rețelei a atras alte provocări, cum ar fi administrarea stărilor distribuite. Un echipament de control logic centralizat trebuie reprodus pentru a face față defectării acestuia. Însă această reproducere poate duce la stări de inconsistență între copiile echipamentului de control. Aceste probleme apar și în cadrul SDN, în contextul echipamentelor de control distribuite.

Protocolul OpenFlow și sistemele de operare de rețea

Înainte de apariţia protocolului OpenFlow, ideile care stau la baza rețelelor definite prin programe software aveau parte de o contradicţie între viziunea unor rețele complet programabile și pragmatismul care ar fi permis lansarea în rețele reale. Protocolul OpenFlow a găsit un echilibru între aceste două obiective, prin posibilitatea de implementare pe comutatoarele deja existente în rețele (suport hardware deja existent) și prin implementarea mai multor funcții decât predecesorii săi. Chiar dacă, bazându-se pe suportul hardware deja existent, și-a asumat anumite limitări, protocolul OpenFlow a fost astfel imediat pregătit pentru lansare în rețelele de producție.

Inițial, s-a dorit ca protocolul OpenFlow să fie implementat pe rețelele din campusuri studențești, pentru a putea fi conduse experimente pe arhitectura de rețea, într-un mediu care să permită cercetarea.

După ce protocolul OpenFlow a avut succes în aceste rețele din campusuri, a început să ia amploare în alte domenii, cum ar fi centrele de date. S-a dovedit a fi mai eficient din punct de vedere al costurilor angajarea de ingineri care să dezvolte aplicații software sofisticate, de control al rețelei, decât cumpărarea de echipamente care să suporte aceleași facilități în mod proprietar.

Ideile care apar în SDN, derivate din cercetarea pentru dezvoltarea protocolului OpenFlow sunt următoarele:

- - **Generalizarea funcţiilor și echipamentelor de rețea.** Ruterele clasice folosesc, în principiu, IP-ul destinație pentru a dirija traficul. În schimb, protocolul OpenFlow poate defini comportamentul de dirijare în funcție de orice set din treisprezece antete diferite de pachet. Astfel, acest protocol unifică mai multe tipuri de echipamente de rețea, care diferă doar prin câmpurile din antetul pachetelor pe care le folosesc la dirijare și tipul de acțiuni pe care le efectuează.

- **Viziunea unui sistem de operare de rețea.** Spre deosebire de rețelele active, care propuneau un sistem de operare la nivelul nodurilor de rețea, cercetarea din cadrul OpenFlow a condus la noțiunea de sistem de operare de rețea. Acesta oferă o împărțire a rețelei în trei niveluri: un plan de date cu o interfață deschisă, un nivel de administrare a stărilor, care are ca responsabilitate menținerea unei vederi consistente asupra stării rețelei și o logică de control, care sa efectueze diferite operații, în funcție de vederea sa asupra rețelei.

- **Tehnici de administrare a stărilor distribuite.** Separarea planurilor de date și de control a dus la provocări privind administrarea stărilor. E nevoie de existența mai multor echipamente de control, pentru performanța, extensibilitatea și siguranța rețelei, însă acestea trebuie să conlucreze ca un singur echipament logic de control.

2.1.2 Evoluția SDN

Motivația apariției SDN

Explozia numărului de dispozitive mobile și a conținutului pe care acestea îl accesează, apariția serviciilor de tip cloud, precum și virtualizarea serverelor au condus la reexaminarea arhitecturilor de rețea de către industria rețelisticii [8]. Astfel, s-au găsit limitări ale rețelelor tradiţionale și, împreună cu nevoile determinate de evoluția tehnologiei s-a ajuns la concluzia că o nouă paradigmă în rețelistică este necesară: rețelele definite prin software.

Îndeplinirea cerințelor pieței în momentul de față, cu ajutorul arhitecturilor de rețea tradiționale, este aproape imposibilă. Costurile operaționale pentru o astfel de rețea sunt foarte mari, din cauza faptului că echipamentele de rețea trebuie administrate individual în momentul implementării de noi politici sau din cauza faptului că echipamentele de rețea care provin de la producători diferiţi trebuie controlate diferit. Pe lângă cele operaționale și costurile de capital au crescut pentru o rețea, din cauza nevoii așa numitor aparate de rețea care trebuie introduse pentru asigurarea securităţii rețelei, sau pentru a putea efectua operații de inginerie de trafic asupra rețelei respective. Așa cum este ilustrat în [8], printre limitările din rețelele tradiționale care au condus la nevoia apariției acestei noi paradigme amintim:

- **Complexitatea** – aceasta duce la stagnarea rețelelor. În momentul de față, rețelele sunt privite ca seturi discrete de protocoale care conectează utilizatorii, într-un mod sigur, indiferent de distanţă, viteza conexiunilor sau topologiile folosite. Aceste protocoale sunt definite punctual, pentru a rezolva o anumită problemă și fără a beneficia de abstractizări fundamentale, ceea ce le face complexe. Pentru adăugarea sau mutarea unui echipament de rețea, administratorii acesteia trebuie să reconfigureze mai multe entități, atât hardware cât și software, folosind aplicații de administrare și trebuie să ia în considerare mai mulți factori, printre care topologia rețelei, modelul echipamentului, versiunea de software etc. Astfel, această complexitate a rețelelor tradiționale duce la o evoluţie lentă a acestora, pentru a scădea riscul întreruperii serviciilor oferite de rețea. Acest lucru duce și la incapacitatea unei adaptări dinamice la modificările de trafic, aplicații și cereri ale utilizatorilor.

- **Inconsistenţa politicilor de rețea** – într-o rețea de producție, pentru implementarea unor politici care să fie valabile în întreaga rețea, este nevoie de configurarea a mii de dispozitive. Tot din cauza complexităţii rețelelor tradiționale, aplicarea unor politici consistente pentru acces, securitate sau calitatea serviciilor este foarte dificilă.

- **Probleme de extensibilitate** – ideea de a garanta mai multă bandă decât poate oferi o conexiune (over-subscription), având la bază modele de trafic predictibile, nu mai este o soluţie în rețelele de la momentul actual. În centrele mari de date, care se bazează pe virtualizare, modelele de trafic sunt foarte dinamice și, implicit, greu de anticipat. Furnizorii foarte mari de servicii, cum ar fi Google sau Facebook, au nevoie de *rețele hiperscalabile*, care să poată asigura o conectivitate cu performanțe ridicate și un cost scăzut între sutele de mii de servere fizice de care aceștia dispun. Acest lucru implică mii de echipamente de rețea și o astfel de extensibilitate este imposibil de oferit printr-o configurare manuală a dispozitivelor rețelei.

- **Dependența de producători** – marile companii doresc un răspuns rapid la schimbările nevoilor de afaceri sau la cererile clienților. Însă, acest lucru este împiedicat de ciclul produselor al vânzătorilor de echipamente, care poate fi chiar de câțiva ani. De asemenea, operatorii de rețea sunt limitați în a își personaliza rețeaua după bunul plac, din cauza lipsei de standarde și de interfețe deschise ale echipamentelor de rețea.

Majoritatea rețelelor tradiționale sunt rețele ierarhice, bazate pe niveluri de comutatoare Ethernet dispuse într-o structură de tip arbore. Această arhitectură era potrivită pentru modelul de calcul de tip client-server. Însă, o astfel de arhitectură statică nu mai este potrivită pentru corporațiile din zilele noastre, unde este nevoie de putere de calcul și de stocare dinamice în centrele de date. Printre ideile promotoare ale paradigmei rețelelor definite prin software se numără:

- **Modele dinamice de trafic** – o data cu apariția marilor centre de date, modelele de trafic s-au schimbat radical. Dacă înainte, majoritatea aplicațiilor erau de tip client-server și majoritatea comunicației era realizată între client și server, aplicațiile mai noi accesează mai multe servere și baze de date, ceea ce implică o rafală de trafic de tip *est-vest* între diferite mașini, până ca datele să ajungă la utilizator, într-un tradițional model de trafic *nord-sud*. În același timp și modelele de trafic ale utilizatorilor se schimbă, aceștia dorind acces la resurse și aplicații de pe orice tip de dispozitiv, de oriunde și la orice oră.

- **Nevoia unui acces flexibil la resursele IT** – angajații încearcă tot mai mult să folosească dispozitive mobile personale, cum ar fi telefoanele inteligente, tabletele sau laptopurile, pentru a accesa rețeaua întreprinderilor. Administratorii rețelelor sunt astfel nevoiți să permită accesul acestor dispozitive și, în același timp, să protejeze datele întreprinderilor, proprietatea intelectuală și să îndeplinească mandatele de conformitate.

- **Dezvoltarea serviciilor de tip cloud** – marile companii au apelat la servicii de tip cloud, atât publice cât și private, ducând la o creștere masivă a acestui tip de servicii. Companiile doresc acum să poată accesa aplicații, infrastructura și alte resurse IT la cerere și la alegere. Pentru a se putea implementa acest tip de cereri, este nevoie de o extensibilitate a puterii de calcul, a puterii de stocare

și a resurselor de rețea și este preferabil ca aceasta să poată fi făcută dintr-un punct comun și utilizând instrumente comune.

- **Nevoia de lățime de bandă mai mare** – volumele foarte mari de date din ziua de astăzi necesită procesare paralelă masivă pe mii de servere interconectate, care au nevoie de conexiuni directe. Creșterea acestor volume de date înseamnă și nevoia creșterii capacității rețelelor. Operatorii acestor centre de date au ingrata sarcină de a crea o rețea în acel centru de date care să fie extensibilă până la o dimensiune inimaginabilă și având grijă să nu se piardă conectivitatea între oricare două elemente de rețea.

Rețelele definite prin programe soft se dovedesc a fi foarte potrivite și în contextul apariției IoT, satisfăcând exact nevoile pe care acesta le are: creșterea lățimii de bandă, configurarea dinamică a rețelei, arhitectură de rețea simplificată care să faciliteze inovația [9].

Introducere în SDN

Rețelele definite prin software reprezintă o nouă paradigmă în arhitecturile de rețea și au la bază patru idei principale: (i) decuplarea planurilor de date și de control, (ii) deciziile de dirijare sunt luate pe baza fluxurilor de date, în loc de adresa destinație, (iii) planul de control se mută într-o entitate logică externă, numită echipament de control SDN, care rulează un sistem de operare de rețea și (iv) rețeaua este programabilă prin aplicații software care rulează în sistemul de operare de rețea și interacţionează cu echipamentele de dirijare.

Astfel, rețelele definite prin software pot fi definite cu ajutorul a trei abstractizări, conform [2], cum se poate observa în Figura 2.1:

- Abstractizarea dirijării;
- Abstractizarea distribuţiei;
- Abstractizarea specificărilor.

În mod ideal, *abstractizarea dirijării* reprezintă permiterea oricărui comportament de dirijare dorit de aplicațiile software din rețea (cu ajutorul planului de control), fără a fi nevoie de cunoașterea de detalii despre capabilitățile hardware ale infrastructurii existente. Un exemplu pentru o astfel de abstractizare este protocolul OpenFlow.

Abstractizarea distribuției se referă la faptul că aplicațiile SDN nu ar trebui să cunoască problemele stărilor distribuite din rețea, transformând problemele unui plan de control distribuit, cum era în rețelele tradiționale, într-un plan de control logic centralizat. Acesta este realizat printr-un nivel comun de distribuţie, în SDN fiind reprezentat de sistemul de operare de rețea. Acesta are două mari funcții: instalarea comenzilor de control pe echipamentele de dirijare și colectarea de informaţii despre starea planului de date, pentru a putea oferi programelor software o vedere de ansamblu asupra rețelei.

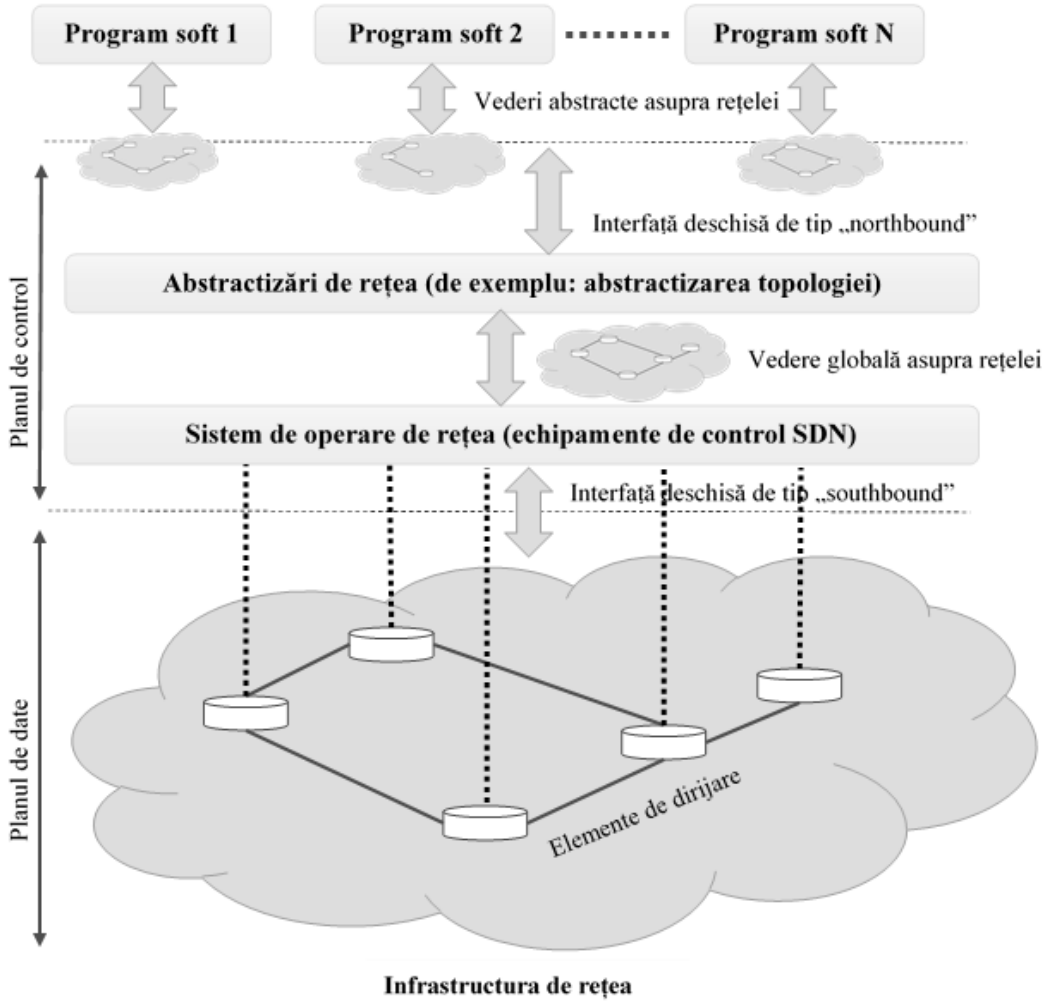

Figure 2.1: Arhitectura SDN și abstractizările fundamentale [2]

Abstractizarea specificărilor reprezintă capabilitatea unui program software din rețea de a exprima un anume comportament al acesteia fără a fi responsabil personal și de implementarea acestui comportament. Acest lucru se poate realiza prin soluții de virtualizare, precum și cu ajutorul limbajelor de programare de rețea.

Arhitectura SDN poate fi privită ca o structură cu mai multe niveluri, având fiecare funcțiile sale specifice. Unele niveluri sunt necesare în orice implementare SDN, în timp ce altele pot fi prezente doar în anumite implementări particulare. Aceste niveluri sunt ilustrate în Figura 2.2 și vor fi prezentate în continuare.

Nivelul 1: Infrastructura Infrastructura pentru o rețea definită prin programe software este compusă, la fel ca în cazul rețelelor tradiționale, din echipamente de rețea (comutatoare, rutere, echipamente de transport de date etc.). Însă, diferenţa majoră constă în faptul că, în SDN, echipamentele de rețea sunt simple elemente de dirijare de trafic, inteligenţa lor fiind mutată în echipamentele de control SDN și în

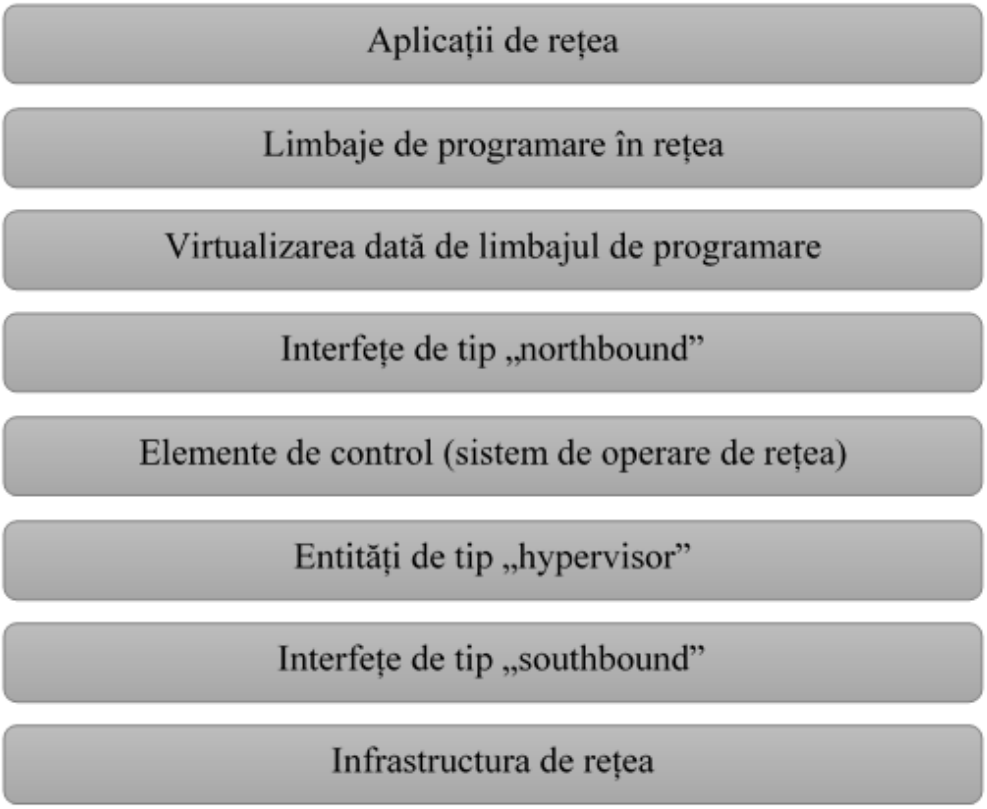

Figure 2.2: Nivelurile rețelelor definite prin software

aplicațiile software. O altă diferenţă importantă este aceea că echipamentele care constituie infrastructura de rețea trebuie să implementeze interfețe standard (cum ar fi OpenFlow), care să asigure compatibilitatea comunicației și a configurărilor, dar și interoperabilitatea cu alte echipamente, atât din planul de control cât și din cel de date, indiferent de producător. Acest lucru era destul de dificil în rețelele tradiționale, din cauza multitudinii de interfețe proprietare.

În paradigma SDN, elementele care alcătuiesc infrastructura rețelei nu mai iau deciziile de dirijare pe baza adresei destinație, ca în cazul rețelelor tradiționale, ci pe baza unor fluxuri de date. Presupunând o rețea bazată pe protocolul OpenFlow, elementele de dirijare se bazează pe o secvenţă de tabele de fluxuri, unde fiecare intrare din tabel conţine: (i) o regulă de potrivire, (ii) acțiunile care trebuie executate asupra pachetelor care se potrivesc regulii și (iii) contoare care să menţină o statistică despre pachetele care s-au potrivit. Prioritatea regulilor este dată de ordinea din tabelul de fluxuri. Acțiunile posibile asupra pachetelor pot fi: dirijarea acestora către un port de ieșire, dirijarea către echipamentul de control, aruncarea pachetelor, trimiterea către secvența normală de prelucrare, trimiterea către următorul tabel de fluxuri etc. Regulile de potrivire care se pot aplica pachetelor se pot baza pe mai multe câmpuri

din antetul pachetului, cum ar fi câmpuri de nivel doi (Ethernet), MPLS, de nivel 3 (IPv4/v6), câmpuri de nivel 4 (TCP/IP) etc., în aproape orice fel de combinație.

Nivelul 2: Interfețele de tip *southbound* Interfețele de tip *southbound* fac legătura între echipamentele de dirijare și echipamentele de control din rețea, îndeplinind astfel una dintre funcțiile de bază ale rețelelor definite prin programe soft: separarea planurilor de date și de control.

Acest tip de interfețe este foarte bine privit de către industrie. Prin standardizarea acestora se va permite construirea de rețele cu echipamente care să poată proveni de la mai multi producători, promovând astfel interoperabilitatea.

Cel mai acceptat și utilizat standard pentru acest tip de interfețe este OpenFlow. Acesta propune specificaţii comune pentru implementarea canalului de comunicaţie dintre echipamentele de dirijare și cele de control. Protocolul OpenFlow furnizează trei surse de informații pentru sistemul de operare de rețea: (i) mesaje bazate pe evenimente sunt trimise de echipamentele de dirijare către cele de control în momentul în care apare o schimbare a legăturii de date sau a unui port; (ii) statistici despre fluxul de date sunt generate în echipamentele de dirijare și trimise către echipamentele de control; (iii) mesaje care conţin și pachete de date sunt trimise de către echipamentele de dirijare către cele de control în momentul în care nu ştiu cum să trateze un anumit tip de flux de date sau din cauza faptului că există o acțiune explicită de tipul *trimite la echipamentul de control* în tabela de fluxuri. Aceste tipuri de informații sunt esențiale pentru furnizarea de detalii despre fluxurile de date sistemului de operare de rețea.

Deşi este cel mai utilizat protocol, OpenFlow nu este singura interfață de tip *southbound* pentru rețelele definite prin programe soft. Există și alte propuneri pentru acest tip de interfețe, dintre care amintim: NETCONF, ForCES (Forwarding and Control Element Separation), OpFlex, POF (Protocol-Oblivious Forwarding), OVSDB (Open vSwitch Database Management), ROFL (Revised Open Flow Library), HAL (Hardware Abstraction Layer), OpenState etc.

Nivelul 3: Entitățile de tip *hypervisor* de rețea Entităţile de tip *hypervisor* reprezintă o soluție software, firmware, sau hardware care creează și rulează mașini virtuale. Acestea permit unor mașini virtuale distincte să partajeze aceleași resurse hardware. Astfel au apărut noi modele de afaceri și tehnologii, cum ar fi aplicațiile de tip cloud, unde fiecare utilizator poate avea resursele sale virtuale, de la puterea de calcul până la spaţiul de stocare. Din păcate, acest model nu a putut fi aplicat și pentru resursele de rețea, acestea fiind configurate în continuare în mod static și independent una față de cealaltă.

Există două cerinţe ale aplicațiilor de la rețea: spaţiul de adresare și topologia rețelei. Volume diferite de muncă necesită diferite servicii și topologii de rețea, însă acestea sunt greu de oferit de către o singură topologie fizică. De asemenea, aplicațiile care rulează într-un mediu virtualizat funcționează în același spațiu de adresare ca infrastructura fizică. Astfel, mașinile virtuale nu pot migra în locuri arbitrare, deoarece schema de adresare este fixă și greu de modificat.

Soluția la această problemă ar fi ca rețeaua să ofere la rândul ei virtualizare, din punctul de vedere al topologiei și al spațiului de adresare. Apoi, fiecare aplicație care rulează într-o mașină virtuală ar avea posibilitatea să configureze atât nodurile de calcul cât și rețeaua, în același timp. Acest lucru nu este posibil prin tehnologiile actuale, cum ar fi VLAN (Virtual Local Area Network) - domeniu virtualizat de nivel 2, NAT (Network Address Translation) - spațiu de adresare IP virtualizat și MPLS (Multi-Protocol Label Switching) - rute virtuale, deoarece rețeaua nu se poate reconfigura într-un mod global, fiind nevoie ca fiecare element de rețea să fie reconfigurat individual.

Rețelele definite prin software încearcă să ofere posibilitatea de virtualizare a rețelei, prin această entitate de tip *hypervisor*. Există mai multe propuneri în acest sens, dintre care amintim: FlowVisor, NVP (Network Virtualization Platform), FlowN, RadioVisor, IBM SDN VE, OpenVirteX, xDPd (eXtensible Datapath Daemon) etc.

Nivelul 4: Echipamentele de control / Sistemul de operare de rețea Sistemele de operare tradiționale oferă abstractizări pentru accesarea resurselor hardware, pentru administrarea accesului concurent la resurse și pentru a oferi mecanisme de securitate și protecție. În contrast, rețelele au fost administrate și configurate până acum cu ajutorul unor instrucțiuni de nivel jos, specifice fiecărui echipament și cu sisteme de operare de rețea proprietare (cum ar fi IOS de la Cisco sau JunOS de la Juniper). Acestea nu furnizează abstractizări care să ofere, într-un mod transparent, funcționalități de bază.

În cadrul SDN se încearcă găsirea unor soluții în acest sens, printr-un control logic centralizat, oferit de un sistem de operare de rețea. Acesta va trebui să ofere abstractizări, care să fie apoi folosite de dezvoltatorii de aplicații software. Printre serviciile oferite de sistemul de operare de rețea ar trebui să se regăsească funcționalități generale, cum ar fi: descoperirea dispozitivelor de rețea, distribuirea configuraţiei rețelei, starea rețelei sau informații despre topologia rețelei.

Echipamentul de control este elementul critic din arhitectura SDN, deoarece el va trebui să interpreteze aplicațiile dezvoltate pentru administrarea rețelei și să genereze configurația acesteia pe baza politicilor definite acolo. Există deja o multitudine de echipamente de control propuse pentru a fi utilizate în arhitectura SDN, diferite în foarte multe aspecte. Unul dintre cele mai importante, care diferențiază echipamentele de control, este tipul arhitectural folosit: centralizat sau distribuit.

Un echipament de control centralizat reprezintă o entitate care administrează toate dispozitivele din planul de date al rețelei. În mod natural, acesta poate fi privit ca un punct unic de defectare a rețelei și ar putea aduce limitări în extensibilitatea acesteia, deoarece este posibil ca un singur echipament de control să nu fie suficient pentru administrarea unui număr mare de echipamente de dirijare. Printre propunerile de astfel de echipamente de control se numără: Maestro, Beacon (care poate administra un număr foarte mare de fluxuri de date, undeva la 12 milioane de fluxuri pe secundă, conform [10]), NOX-MT, sau Floodlight. Acestea se bazează pe mai multe fire de execuție și pe paralelismul oferit de arhitecturile de calculatoare cu mai multe nuclee.

Pe de altă parte, un sistem de operare de rețea distribuit poate rezolva problema extensibilităţii rețelei, fiind potrivit pentru orice tip de rețea. Această distribuţie poate fi realizată printr-un grup de echipamente de control care să se afle în același loc, sau printr-un set de dispozitive distribuite fizic în mai multe locuri. Această ultimă variantă ar putea fi mai potrivită pentru prevenirea diferitelor defecţiuni logice sau fizice. Câteva exemple de astfel de echipamente de control distribuite: Onix, HP VAN SDN, PANE, HyperFlow, ONOS (Open Network Operating System) [11] sau ODL (OpenDaylight) [12], SMaRtLight etc.

Însă, în cazul echipamentelor de control distribuite, apare o problemă destul de importantă: consistenţa datelor. Toate echipamentele de control ar trebui să citească aceeaşi valoare imediat după ce aceasta a fost scrisă, pentru a evita cazul când dispozitivele de control au vederi diferite asupra rețelei. Majoritatea propunerilor oferă doar o *consistență scăzută*, ceea ce înseamnă că, după ce o valoare a fost scrisă într-un nod de control, valoarea se va reflecta, *la un moment dat*, în toate nodurile de control. Acest lucru implică o perioadă de timp în care dispozitivele de control au viziuni diferite asupra rețelei. Există și propuneri de echipamente de control care oferă o *consistență ridicată* (Onix și SMaRtLight), care garantează citirea aceleiaşi valori de către orice nod de control, imediat după scrierea acesteia.

Există și situaţii unde o abordare hibridă ar fi cea mai potrivită, o arhitectură în care să existe grupuri de echipamente de control într-o parte de rețea și dispozitive de control distribuite în alte locuri din rețea.

Nivelul 5: Interfețe de tip *northbound* Interfețele de tip *northbound*, împreună cu cele de tip *southbound* constituie cele mai importante două abstractizări din arhitectura rețelelor definite prin software. Dacă cele din urmă asigură comunicaţia dintre echipamentele de control și dispozitivele din planul de date al rețelei, fiind astfel mai mult orientate spre hardware, interfeţele de tip *northbound* alcătuiesc, în mare parte, un ecosistem software. Încă nu există un astfel de tip de interfețe care să fie acceptat la scară largă, cum este OpenFlow în cazul celor te tip *southbound*, însă acest lucru se va întâmpla probabil, pe măsură ce paradigma SDN se va maturiza și cazurile de utilizare ale acestui tip de arhitectură de rețea se vor contura mai exact.

Este nevoie ca acest tip de interfețe să fie deschise și standardizate, astfel încât să se asigure interoperabilitatea și portabilitatea programelor soft pe mai multe tipuri de dispozitive de control. Un exemplu în acest sens îl constituie NOSIX [13]. Aceasta este o propunere, care poate fi comparată cu standardul POSIX (Portable Operating System Interface) din sistemele de operare și care oferă abstractizări ce garantează independenţa față de limbajul de programare și dispozitivul de control folosite. Dintre celelalte propuneri de interfețe de tip *northbound* amintim: RESTCONF, SFNet, Pyretic, NetCore, Frenetic, Nettle etc.

Nivelul 6: Virtualizarea dată de limbajul de programare Există două caracteristici principale ale soluţiilor de virtualizare date de limbajele de programare: permiterea mai multor nivele de abstractizare, concomitent cu oferirea proprietăţilor dorite, cum ar fi protecţia și abilitatea de a exprima modularitatea.

Metodele de virtualizare pot permite, de exemplu, diferite vederi asupra unei singure infrastructuri fizice de rețea. Un *comutator virtual* ar putea reprezenta o combinație de mai multe dispozitive de dirijare. Acest mod de lucru simplifică sarcinile dezvoltatorilor de aplicații, care nu trebuie să ţină cont individual de elementele de rețea care alcătuiesc acel *comutator virtual*. Dezvoltarea și implementarea unor aplicații de rețea complexe este simplificată cu ajutorul acestor abstractizări.

O altă formă de virtualizare dată de limbajul de programare o reprezintă împărţirea statică a rețelei în bucăţi. Acest lucru se face de către compilator, bazat pe definiţiile date de nivelul aplicație. După compilare va rezulta un program unitar de control, care are deja implementate definiţiile pentru împărțirea rețelei în bucăţi și comenzile de configurare a acestora. În acest caz nu mai este nevoie de entitatea *hypervisor* care să administreze dinamic bucăţile de rețea.

Există diverse propuneri pentru astfel de soluții de virtualizare, cum ar fi: Pyretic, Splendid, libNetVirt, FlowVisor, IBM SDN VE etc.

Nivelul 7: limbaje de programare în rețea Limbajele de programare au evoluat de la limbaje mașină, specifice hardware-ului, cum era limbajul *assembly* pentru arhitecturile x86, până la limbaje de nivel înalt, cum ar fi Java sau Python. În același mod, limbajele de programare folosite în programarea rețelelor evoluează de la OpenFlow (echivalentul limbajului *assembly*) la limbaje de nivel înalt, cum ar fi Pyretic, Procera, NetCore, Frenetic etc.

Aceste limbaje de programare de nivel înalt oferă câteva avantaje în contextul rețelelor definite prin software: facilitează dezvoltarea virtualizării rețelei, creează abstractizări care simplifică programarea elementelor de dirijare, promovează modularizarea software și reutilizarea codului în planul de control și chiar îmbunătăţesc dezvoltarea și inovația prin crearea unor medii de lucru mai productive.

Există două tipuri de paradigme de programare în contextul SDN: cea declarativă, care este cea mai răspândită și paradigma imperativă, care este reprezentată doar prin limbajul Pyretic. Paradigma declarativă reprezintă o abordare în care rețelei i se spune ce tip de comportament să aibă și aceasta se va configura (luând singură decizii) astfel încât să îndeplinească acea cerinţă. Paradigma imperativă se referă la situația în care programatorul îi spune rețelei cum să facă ceva și rezultatul va fi cel așteptat, fără ca aceasta să ia propriile decizii.

Scopul SDN este ca, în final, să ofere facilități de administrare a rețelelor pe baza infrastructurii definite anterior. Cu ajutorul progreselor din domeniul limbajelor de programare de nivel înalt se va facilita crearea unui ecosistem propice pentru dezvoltarea aplicațiilor SDN.

Nivelul 8: Aplicațiile de rețea Aplicațiile software vor reprezenta cea mai importantă parte a rețelelor definite prin software, deoarece acestea vor implementa logica de control, care va fi translatată în comenzi ce vor fi instalate pe dispozitivele din planul de date.

Majoritatea aplicațiilor software din cadrul SDN se încadrează într-una din următoarele cinci categorii, conform [2]: ingineria traficului, securitate și fiabilitate, măsurători și monitorizare, reţelistica centrelor de date și mobilitate și microunde.

Programele software din categoria mobilitate și microunde își propun să faciliteze implementarea și administrarea rețelelor fără fir, cum ar fi rețelele locale fără fir - WLAN (Wireless Local Area Network) sau rețelele de telefonie mobilă. Un plan de control distribuit, cum există în momentul de față în rețelele fără fir, nu este optim pentru implementarea mecanismelor de transfer dintre celule, pentru micșorarea interferențelor, pentru alocarea resurselor radio, pentru administrarea spectrului limitat de frecvenţe etc. Aceste probleme sunt adresate în SDN și se pot rezolva mai uşor cu ajutorul acestei noi paradigme.

2.2 Standardizarea SDN

2.3 SDN în contextul reţelelor actuale

Capitolul 3

SDN în contextul reţelelor de transport fără fir

3.1 Modelul informaţional pentru microunde - ONF TR-532

Istoria reţelelor definite prin software.

3.2 Modelul informaţional de bază - ONF TR-512

SDN și reţelele actuale..

3.3 Protocolul NETCONF

Standardizare: ONF, etc..

3.4 Alegerea unui cadru pentru serverul NETCONF

Standardizare: ONF, etc..

3.5 Arhitectura demonstraţiilor de concept WT SDN

Standardizare: ONF, etc..

Capitolul 4

Mediatorul cu valori implicite - prima versiune

4.1 Arhitectura

SDN și reţelele actuale..

4.2 Implementarea

Istoria reţelelor definite prin software.

4.3 Folosirea în contextul demonstraţiilor de concept

Standardizare: ONF, etc..

Capitolul 5

Mediatorul cu valori implicite - a doua versiune

5.1 Arhitectura

SDN și reţelele actuale..

5.2 Implementarea

Istoria reţelelor definite prin software.

5.3 Folosirea în contextul demonstraţiilor de concept

Standardizare: ONF, etc..

5.4 Integrarea cu LINC

Standardizare: ONF, etc..

Capitolul 6

Simulatorul reţelelor de transport de date fără fir

6.1 Arhitectura

SDN și reţelele actuale..

6.2 Implementarea

Istoria reţelelor definite prin software.

6.3 Folosirea în contextul demonstraţiilor de concept

Standardizare: ONF, etc..

Capitolul 7

Rezultate și discuţii

7.1 Evaluarea soluţiilor propuse

SDN și reţelele actuale..

7.2 Comparaţie între WTE și alte abordări

Istoria reţelelor definite prin software.

7.3 Demonstrarea cazurilor de utilizare cu ajutorul WTE

Standardizare: ONF, etc..

Capitolul 8

Concluzii

8.1 Rezultate obţinute

Istoria reţelelor definite prin software.

8.2 Contribuţii originale

SDN și reţelele actuale..

8.3 Lista contribuţiilor originale

Standardizare: ONF, etc..

8.4 Perspective de dezvoltare ulterioară

Standardizare: ONF, etc..

Bibliografie

[1] Thomas D Nadeau and Ken Gray. *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies.* " O'Reilly Media, Inc.", 2013.

[2] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[3] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.

[4] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking control of the enterprise. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 1–12. ACM, 2007.

[5] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[6] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[7] Evangelos Haleplidis, Jamal Hadi Salim, Joel M Halpern, Susan Hares, Kostas Pentikousis, Kentaro Ogawa, Weiming Wang, Spyros Denazis, and Odysseas Koufopavlou. Network programmability with forces. *IEEE Communications Surveys & Tutorials*, 17(3):1423–1440, 2015.

[8] OME Committee et al. Software-defined networking: The new norm for networks. *Open Networking Foundation*, 2012.

[9] Zhijing Qin, Grit Denker, Carlo Giannelli, Paolo Bellavista, and Nalini Venkatasubramanian. A software defined networking architecture for the internet-of-things. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE, 2014.

[10] David Erickson. The beacon openflow controller. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 13–18. ACM, 2013.

[11] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6. ACM, 2014.

[12] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, pages 1–6. IEEE, 2014.

[13] Andreas Wundsam and Minlan Yu. Nosix: A portable switch interface for the network operating system. Technical report, TR-12-013, Oct. 1, 2012, XP055120641,(URL: https://www. icsi. berkeley. edu/pubs/techreports/ICSI—TR-12-013. pdf) Section 3, Figure 2, 2012.