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

BBF	BroadBand Forum
BEEP	Blocks Extensible Exchange Protocol
CLI	Command Line Interface
CORBA	Common Object Request Broker Architecture
DVM	Default Values Mediator
ETC	Ethernet Container
ETH	Ethernet MAC Layer
ETSI	European Telecommunications Standards Institute
ETY	Ethernet Physical Layer
ForCES	Forwarding and Control Element Separation
HAL	Hardware Abstraction Layer
I2RS	Interface to the Routing System
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IRTF	Internet Research Task Force
ITU-T	International Telecommunications Union - Telecommunications Standardization Sector
LLDP	Link Layer Discovery Protocol
MAC	Medium Access Control
MEF	Metro Ethernet Forum
MIMO	Multiple-Input Multiple-Output
MPLS	Multi-Protocol Label Switching
MPLS-TP	Multi-Protocol Label Switching Transport Profile
MWPS	Microwave Physical Section
MWPS-TTP	Microwave Physical Section - Trail Termination Point
MWS	Microwave Section
MWS-TTP	Microwave Section - Trail Termination Point
NAT	Network Address Translation
NCP	Network Control Point

NE	Network Element
NETCONF	Network Configuration Protocol
NVP	Network Virtualization Platform
OCh	Optical Channel
ODL	OpenDaylight
ODU	Optical Data Unit
OIF	Optical Interface Forum
OMS	Optical Multiplex Section
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OPS	Optical Protection Switch
OSPF	Open Shortest Path First
OT	Optical Transport
OTS	Optical Transmission Section
OTU	Optical channel Transport Unit
OTWG	Open Transport Working Group
OVSDB	Open vSwitch Database Management
PoC	Proof of Concept
POF	Protocol-Oblivious Forwarding
POSIX	Portable Operating System Interface
PTP	Precise Time Protocol
REST	Representational State Transfer
ROFL	Revised Open Flow Library
RPC	Remote Procedure Call
SDN	Software-Defined Networking
SDNRG	Software-Defined Networking Research Group
SDO	Standards Developing Organization
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SPRING	Source Packet Routing in Networking
SSH	Secure Shell
TLS	Transport Layer Security
TR	Technical Recommendation
TS	Technical Specification
UML	Unified Modeling Language
VLAN	Virtual Local Area Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WT	Wireless Transport
WTE	Wireless Transport Emulator

xDPd	eXtensible Datapath Daemon
XML	Extensible Markup Language
XPIC	Cross Polarization Interference Cancellation
YANG	Yet Another Next Generation

Capitolul 1

Introducere

1.1 Prezentarea domeniului tezei de doctorat

În urma dezvoltării 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 toate echipamentele, cu ajutorul unor tehnologii care permit acest lucru, cum ar fi arhitectura *Cloud Computing*, de a crește mobilitatea tuturor aplicațiilor, de a defini concepte și aplicații 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 lărgime de bandă crescută, de o reţea mai simplă și agilă, care să faciliteze inovarea. Reţelele definite prin software - SDN (Software-Defined Networking), reprezintă o nouă paradigmă care a apărut în industria reţelisticii, pentru a contracara dezavantajele pe care reţelele tradiţionale le-au dovedit.

Tehnologia SDN nu a ajuns încă la maturitate și nu a pătruns în toate tipurile de reţele de comunicaţii existente. Este prezentă în campusuri universitare, sau în centre de date, însă se încearcă introducerea acesteia în alte 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 cercetare, standardizare și demonstraţii de concept - PoC (Proof of Concept), pentru prezentarea avantajelor pe care această nouă soluție le 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 introdusă anterior, 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 al 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 soluții 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, greu de administrat și configurat. De asemenea, numărul dispozitivelor mobile a crescut considerabil, precum și 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 reprezintă 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 în scopul de 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, fără a implica 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 o schimbare de concept radicală asupra controlului unei reţele, propunând o interfaţă de programare care să expună resurse în noduri individuale de reţea și 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 și să implementeze 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, prin introducerea unei vederi 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 stăteau 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 la nivelul comutatoarelor 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 în rețelele din campusuri studențești, pentru a putea fi conduse experimente în 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 imagini consistente asupra stării rețelei și o logică de control, care sa efectueze diferite operații, în funcție de imaginea 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 sistem 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 identificat limitări ale rețelelor tradiţionale și, împreună cu nevoile determinate de evoluția tehnologiei s-a ajuns la concluzia că o nouă abordare 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 acele 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 duce la creșterea complexităţii lor. 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, de asemenea, 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 pentru 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 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 lărgime 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 complexa 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 lărgimii 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 necesita cunoașterea detaliilor legate de 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

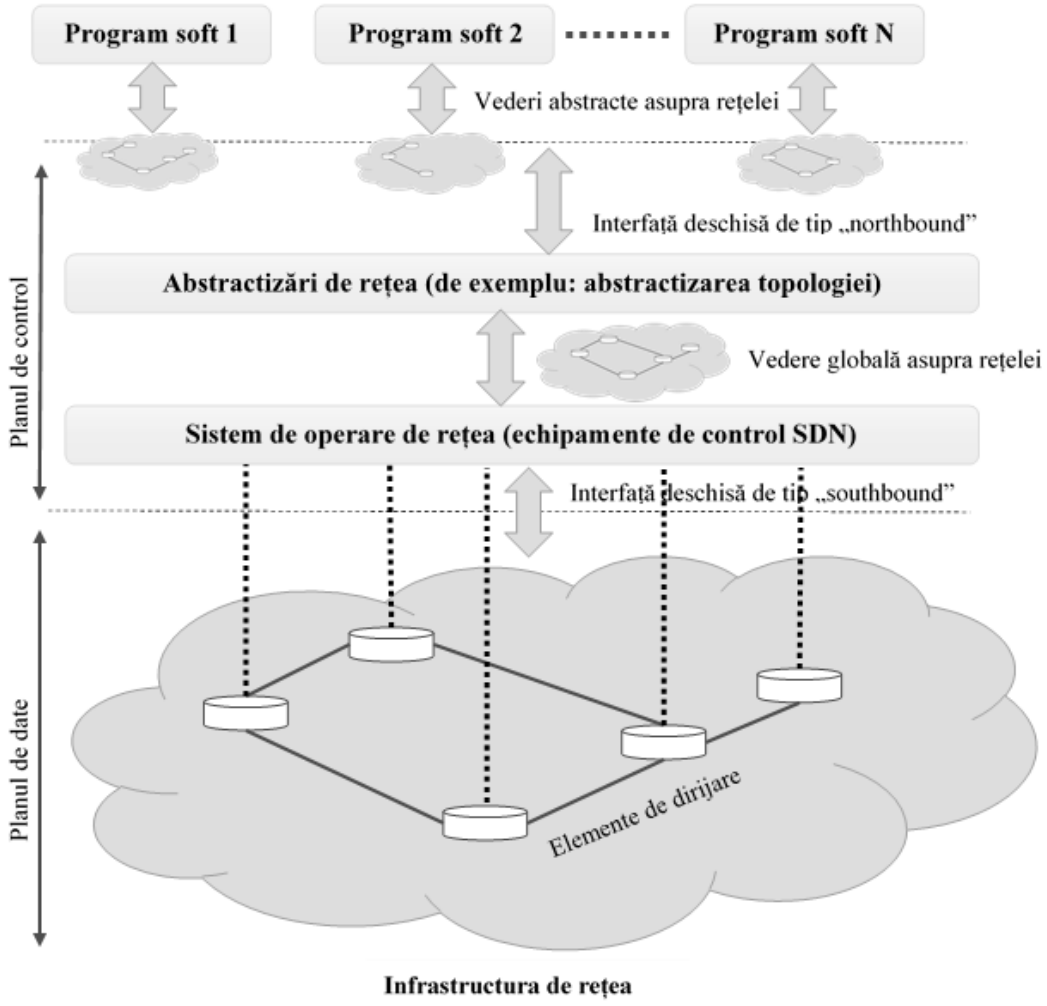

Fig. 2.1: Arhitectura SDN și abstractizările fundamentale [2]

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.

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 [2].

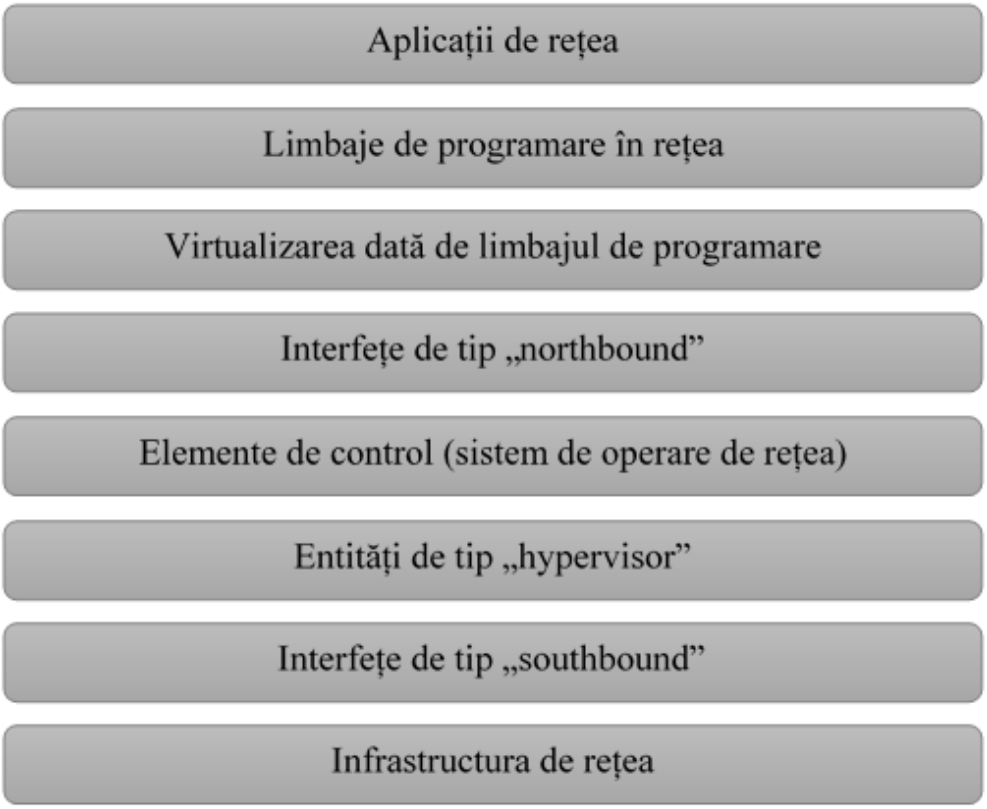

Fig. 2.2: Nivelurile rețelelor definite prin software

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 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 [6]. 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 asociere, (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, care 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 fluxurile de date, care sunt generate în echipamentele de dirijare și trimise către echipamentele de control; (iii) mesaje care conţin și pachete de date și 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. [7, 10, 11].

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 [2, 12, 13].

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, HyperFlex, xDPd (eXtensible Datapath Daemon) etc. [14–18].

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. Prin 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 [2].

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 [19–21].

Un echipament de control centralizat reprezintă o entitate care administrează toate dispozitivele din planul de date al rețelei [8]. 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 [22]), 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 [23], HP VAN SDN, PANE, HyperFlow [24], ONOS (Open Network Operating System) [25] sau ODL (OpenDaylight) [26], SMaRtLight [27] 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 de până acum 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) [23, 27], 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 [28]. 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 [29]. Dintre celelalte propuneri de interfețe de tip *northbound* amintim: RESTCONF, SFNet, Pyretic, NetCore, Frenetic, Nettle etc. [30–33].

Nivelul 6: Virtualizarea dată de limbajul de programare Există două caracteristici principale ale soluţiilor de virtualizare date de limbajele de programare: permiterea mai multor niveluri 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 imagini 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 secţiuni. 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 funcțiile 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. [14, 32, 34, 35].

Nivelul 7: limbaje de programare în rețea Limbajele de programare au evoluat de la limbaje mașină, specifice hardware-ului, cum era limbajul de asamblare 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 de asamblare) la limbaje de nivel înalt, cum ar fi Pyretic, Procera, NetCore, Frenetic etc. [32, 33, 36].

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 cea 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 acţioneze 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 la nivelul dispozitivelor 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, mobilitate și rețele fără fir.

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

Activitățile de cercetare și standardizare din jurul SDN au loc pe două planuri importante. Pe de o parte, există organizaţiile care dezvoltă standarde - SDO (Standards Developing Organization), care sunt formate din reprezentanţi ai industriei, ai academiei, sau alte entități și au ca scop dezvoltarea de specificaţii sau recomandări tehnice, în contextul SDN care să fie folosite de toată industria rețelisticii. Autorii din [37] amintesc astfel de organizații: ONF, IETF (Internet Engineering Task Force), ETSI (European Telecommunications Standards Institute), ITU-T (International Telecommunications Union - Telecommunications Standardization Sector), IEEE (Institute of Electrical and Electronic Engineers).

Pe de altă parte, există asociații sau comunități de oameni, în general care fac parte din industrie, ale căror rezultate ale cercetării candidează pentru a deveni standarde. Aceste rezultate sunt, de obicei, implementări cu sursă deschisă ce vor fi folosite ulterior în industrie. Exemple de astfel de comunități sunt prezente în [38,39]: OpenDaylight (activităţi ce se desfăşoară sub patronajul fundaţiei Linux), MEF (Metro Ethernet Forum), BBF (BroadBand Forum), OIF (Optical Interface Forum).

Activitățile de standardizare ale SDN sunt foarte importante, asigurându-i acestei tehnologii o evoluţie stabilă și aducând-o la o maturitate care îi va permite adoptarea pe scară largă în industria rețelisticii, mitigând astfel dezavantajele rețelelor tradiționale.

Sunt mai multe planuri pe care se lucrează pentru standardizarea SDN. Unul dintre aceste planuri este cel al interfeţelor de tip *southbound*, care fac legătura între echipamentele de dirijare și echipamentele de control SDN. Protocolul OpenFlow este un exemplu în acest sens, însă nu este singurul protocol capabil să facă legătura între planurile de date și de control. În ultimul timp se pune foarte mare accent pe NETCONF ca o alternativă pentru a configura echipamentele care dirijează traficul, după cum se poate observa în [10, 40, 41]. Astfel apare nevoia de a dezvolta modele informaţionale care să abstractizeze echipamentele din planul de date și care să poată fi folosite de NETCONF pentru a configura dispozitivele. Și planul interfeţelor de

tip *northbound* are un rol important în standardizare, deoarece poate oferi un punct de plecare comun pentru dezvoltatorii de aplicații SDN. De exemplu, în cadrul ONF există un grup care se ocupă cu activităţi de standardizare în această direcţie. Un alt plan este reprezentat de implementările software, cu sursă deschisă (*open-source*), care se crează în acest context. Așa cum evidenţiază și autorii din [42, 43], aceste implementări sunt foarte importante prin ecosistemele care apar ca urmare a activităţii *comunităţilor open-source*.

O importanţă foarte mare în cadrul acestor activităţi o au și demonstraţiile de concept. Acestea au capacitatea de a demonstra avantajele pe care SDN le aduce nu doar la un nivel teoretic, ci într-un mod practic, propunând diferite cazuri reale de utilizare a acestei tehnologii și aplicând-o, într-un mod restrâns, unor topologii formate din echipamente reale. Scopul acestora este, pe de o parte, de a adăuga un plus de valoare activităţilor de standardizare și de a testa și proba rezultatele acestor activităţi. Pe de altă parte, aceste demonstraţii de concept au scopul de a atrage atenţia și altor entități din industrie și a duce la înlesnirea adoptării acestei tehnologii pe scară largă.

2.2.1 Open Networking Foundation

ONF este o organizaţie non-profit formată din peste două sute de membri care fac parte din industrie, academie, sau institute de cercetare, ce are ca obiectiv accelerarea adoptării SDN pe scară largă in industria rețelisticii prin dezvoltarea de standarde deschise și de ecosisteme software cu sursă deschisă. ONF a apărut ca urmare a activităţii de cercetare din jurul protocolului OpenFlow din cadrul Universităţii Stanford. Dintre membrii cei mai importanți amintim operatori de rețele, precum AT&T, Google, Facebook, Verizon, Deutsche Telekom sau Telefonica, producători de echipamente, cum ar fi Cisco, Ericsson, Huawei, Intel sau NEC și reprezentanţi ai unor universităţi cunoscute, ca Stanford sau Princeton.

Activitățile din cadrul ONF sunt împărţite în mai multe zone de interes, conform [44]:

- *Operatori*. Această zonă se ocupă de mai multe aspecte, precum SDN în contextul sistemelor *Carrier Grade* (rețelele de telecomunicații foarte sigure, cu o disponibilitate de peste 99,999% și recuperare în caz de defectare de sub 50 de milisecunde), în centre de date, în întreprinderi sau aspecte legate de migrarea serviciilor dintr-o rețea tradiţională într-o rețea definită prin software.

- *Servicii*. Se ocupă de proiecte care permit existența aplicațiilor și serviciilor de rețea care au la bază tehnologia SDN. Astfel, există mai multe grupuri de lucru care analizează arhitectura SDN și modul în care aceste principii se aplică în imaginea de ansamblu a unei rețele de comunicaţii, un model informaţional *de bază*, care să reprezinte piatra de temelie cu ajutorul căreia să se dezvolte alte modele informaţionale, specializate pentru anumite aplicații sau tehnologii, interfeţele de tip *northbound*, pentru a facilita dezvoltarea aplicațiilor SDN, probleme de securitate pe care această nouă tehnologie le poate întâmpina.

- *Specificaţii*. Zonă care are ca responsabilitate publicarea tuturor specificaţiilor și recomandărilor tehnice create de ONF. Acestea includ protocoalele OpenFlow, OF-Config dar și alte interfețe standard care se dezvoltă pentru tehnologii de transport de diferite tipuri (optic, fără fir). Există și un proiect care se ocupă de testare și interoperabilitate, scopul acestuia fiind accelerarea adoptării protocolului OpenFlow prin certificări și promovarea interoperabilităţii între diferiţi producători de echipamente [45].

- *Piaţă*. Această zonă se concentrează pe educarea comunităţii SDN în legătură cu valoarea pe care standardele ONF le oferă rețelelor definite prin software și promovarea adoptării unor astfel de rețele definite prin software cu sursă deschisă. Aceste obiective sunt îndeplinite prin publicaţii, evenimente care se organizează sau demonstraţii care să arate comunităţii cazuri reale de utilizare.

Există trei tipuri de publicaţii care reies din activitățile ONF: (i) *specificaţii*, care includ toate standardele care definesc un protocol, modelul informaţional, funcționalități și documente despre cadrele asociate; publicaţiile normative astfel rezultate se numesc Specificaţii Tehnice - TS (Technical Specification) și sunt supuse unor procese de licenţiere; (ii) *recomandări*, care includ consideraţii arhitecturale, cazuri de utilizare, analiza cerințelor, terminologie; aceste documente referă documente normative ONF, dar nu necesită licenţiere și pot fi utilizate în mod liber, având rolul de Recomandări Tehnice - TR; (iii) *publicaţii*, reprezentând documente care conţin informații ce ajută în procesele de lansare a SDN în producție, rezumate ale soluţiilor, studii de caz sau cărţi albe (*white papers*); aceste documente nu au caracter normativ și pot fi folosite în mod liber.

În continuare vor fi enumerate câteva astfel de documente produse de către ONF: *OpenFlow Switch Specification Ver. 1.5.1* (TS-025), care descrie cerinţele unui comutator logic ce suportă protocolul OpenFlow; *OpenFlow Management and Configuration Protocol 1.2 (OF-Config 1.2)* (TS-016), document ce descrie motivația, cerințele, scopul și specificaţiile protocolului OF-Config; *Conformance Test Specification for OpenFlow Switch Specification V1.3.4* (TS-026), definind cerinţele și procedurile de test care determină conformitatea unui comutator cu specificațiile protocolului OpenFlow 1.3.4; *Core Information Model (CoreModel) 1.2* (TR-512), care prezintă modelul informațional de bază, pe care celelalte modele informaționale dezvoltate în ONF se bazează; *Microwave Information Model* (TR-532), reprezentând modelul informațional ce descrie echipamentele de transport de date fără fir. Aceste ultime două documente vor fi detaliate în următorul capitol, fiind baza dezvoltării și implementării simulatoarelor propuse în această lucrare.

Un rol important în adoptarea SDN de către operatori și în diseminarea rezultatelor din cadrul ONF îl au demonstraţiile de concept. Acestea adună laolaltă operatori, producători de echipamente și integratori de servicii, cu scopul de a demonstra recomandările produse de activitatea de cercetare. În contextul rețelelor de transport de date fără fir, proiectul WT, care face parte din grupul OTWG (Open Transport Working Group), a terminat cu succes patru astfel de demonstraţii de concept [46–48].

Prima astfel de demonstraţie a avut loc în octombrie 2015, în Madrid și a fost organizată de Telefonica, împreună cu Universitatea Carlos III. Scopul acesteia a fost

de a extinde protocolul OpenFlow cu atribute specifice echipamentelor de transport de date fără fir. Astfel, interfaţa de tip *southbound* folosita a fost OpenFlow, în timp ce echipamentul de control SDN a fost ONOS. Au fost prezentate două cazuri de utilizare: (i) pornirea/oprirea unei interfeţe radio în funcţie de nivelul de trafic ce trebuie transmis de către echipament, economisind astfel energie în momentele în care nivelul de trafic în reţea nu este foarte ridicat; (ii) schimbarea căilor de date într-un ruter, în cazul în care se pierd pachete pe legătura radio, ca urmare a unor schimbări meteorologice (simulate prin folosirea unui atenuator variabil pe legătura radio).

A doua demonstraţie de concept a avut loc în aprilie 2016, la München și a fost organizată de Telefonica. Spre deosebire de prima demonstraţie, în cea de-a doua s-a folosit drept interfaţă *southbound* protocolul NETCONF, iar echipamentul de control SDN a fost ODL. Ca model informaţional YANG pentru configurarea echipamentelor s-a ales un model pentru microunde simplificat (conţinea un număr limitat de atribute). Acesta a fost dezvoltat de către grup și apoi implementat de către echipamentele de la diferiţi producători. Au fost demonstrate mai multe cazuri de utilizare folosind aplicaţii SDN care se bazează pe acel model: detectarea și configurarea de noi echipamente, detectarea și corecţia (printr-o acţiune a operatorului) diferenţelor între configuraţia curentă și cea planificata, detecţia și vizualizarea reţelei de transport configurate, primirea, afişarea și stocarea evenimentelor și alarmelor din reţea.

Cea de-a treia demonstraţie de concept s-a desfăşurat în octombrie 2016 și a avut loc în centrul de cercetare WINLAB de la Universitatea Rutgers, fiind organizat de AT&T. S-au ales aceeaşi interfaţă *southbound* și acelaşi echipament de control SDN, scopul acestei demonstraţii fiind de a proba utilitatea întregului model informaţional pentru microunde (conţinea toate atributele pe care le poate avea un echipament de transport de date fără fir). S-a implementat și modelul de bază (*Core Model*), dezvoltat de alt grup din cadrul ONF. Au fost demonstrate aplicaţiile dezvoltate pentru cea de-a doua demonstraţie, dar și două noi aplicaţii: una care administrează spectrul, prin compararea și configurarea frecvenţelor planificate și cele configurate pe echipamente, realocându-le în caz că nu se potrivesc; cealaltă, denumită automatizarea în bucla închisă, demonstra un răspuns simplist la anumiţi factori declanşatori (interni, externi sau temporali).

A patra demonstraţie de concept a avut loc în iunie 2017, la Bonn și a fost organizată de Deutsche Telekom. S-au folosit mai multe modele informaţionale: modelul pentru microunde, modelul de bază, un model Ethernet simplificat și un model pentru sincronizare - pentru PTP (Precise Time Protocol), plecând de la un model dezvoltat de ITU-T. Astfel, s-au putut demonstra mai multe cazuri de utilizare: administrarea echipamentelor care transportă date fără fir, administrarea indicatorilor de performanță ai echipamentelor, administrarea puterii echipamentelor, administrarea echipamentelor capabile Ethernet, recalcularea căilor de trafic Ethernet într-o rețea sau administrarea echipamentelor care suportă PTP.

2.2.2 IETF

Un alt SDO care prestează activităţi de standardizare în domeniul rețelelor definite prin software este IETF. Această organizaţie, în general, oferă standardele de bază folosite în Internet. Obiectivul lor declarat este îmbunătăţirea funcţionării Internetului prin dezvoltarea de documente tehnice relevante și de înaltă calitate care să ghideze proiectarea, utilizarea și administrarea Internetului.

Doar puține din grupurile de lucru din cadrul IETF se ocupă cu activităţi care au legătură cu tehnologia SDN. Unul dintre acestea a dezvoltat ForCES, inițiativă ce separa planurile de date și de control ale echipamentelor [49], cu câțiva ani înainte ca protocolul OpenFlow, mult mai cunoscut, să propună asta într-un mod care să permită o adoptare mai rapidă în industrie.

Alt grup de lucru se ocupă de *interfaţa către sistemul de rutare* - I2RS (Interface to the Routing System). Acesta își propune să dezvolte interfețe pentru ca aplicațiile SDN să poată accesa sistemul de rutare al rețelei. Se doreşte ca beneficiarii I2RS să fie aplicații de administrare, echipamente de control SDN sau aplicații de utilizator care au cereri specifice la adresa rețelei. Acestea ar permite informaţiilor, politicilor și parametrilor operaţionali să fie introduşi sau extraşi din sistemul de rutare, fără a afecta consistenţa datelor și coerenţa infrastructurii de rutare. Această abordare poate fi privită ca un rival al protocolului OpenFlow, cu mențiunea că se aplică doar de la nivelul 3 în sus, în stiva OSI.

În interiorul grupului de lucru SPRING (Source Packet Routing in Networking) se discută un control al rutării care să permită indicarea unei căi de dirijare încă de la sursa de trafic [37], calcularea acesteia putând fi făcută atât centralizat, cât și distribuit. Tot în cadrul IETF se dezvoltă și protocolul NETCONF, însă acesta va fi detaliat într-o secţiune viitoare.

Parte din IETF este și IRTF (Internet Research Task Force). Aceasta este o organizaţie paralelă, care se ocupă mai mult de partea de cercetare, sprijinind proiectele care se crede că vor fi benefice comunităţii Internetului, dar care nu sunt încă pregătite pentru implementare. Aici există un grup de lucru care investighează tehnologia SDN din mai multe perspective, cu scopul de a recunoaşte atât abordări ce pot fi definite și lansate pe termen scurt, dar și provocări viitoare: SDNRG (Software-Defined Networking Research Group). Sunt de interes probleme ca extensibilitatea soluţiei, abstractizări sau limbaje de programare ce pot fi folosite în contextul SDN. Se doreşte ca acest grup să ofere un spațiu comun tuturor cercetătorilor cu interes în acest domeniu.

2.3 SDN în contextul reţelelor actuale

Chiar dacă nu este o tehnologie ajunsă la maturitate în toate aspectele unei rețele, paradigma SDN este deja folosită în rețele de producție de anumite tipuri. Autorii din [50] evidenţiază faptul că evoluția SDN este susţinută de către trei mari porţiuni ale industriei rețelisticii: furnizorii serviciilor de tip *cloud* (care utilizează mari centre

de date), furnizorii de servicii de telecomunicaţii și întreprinderile, așa cum se poate observa și în Figura 2.3.

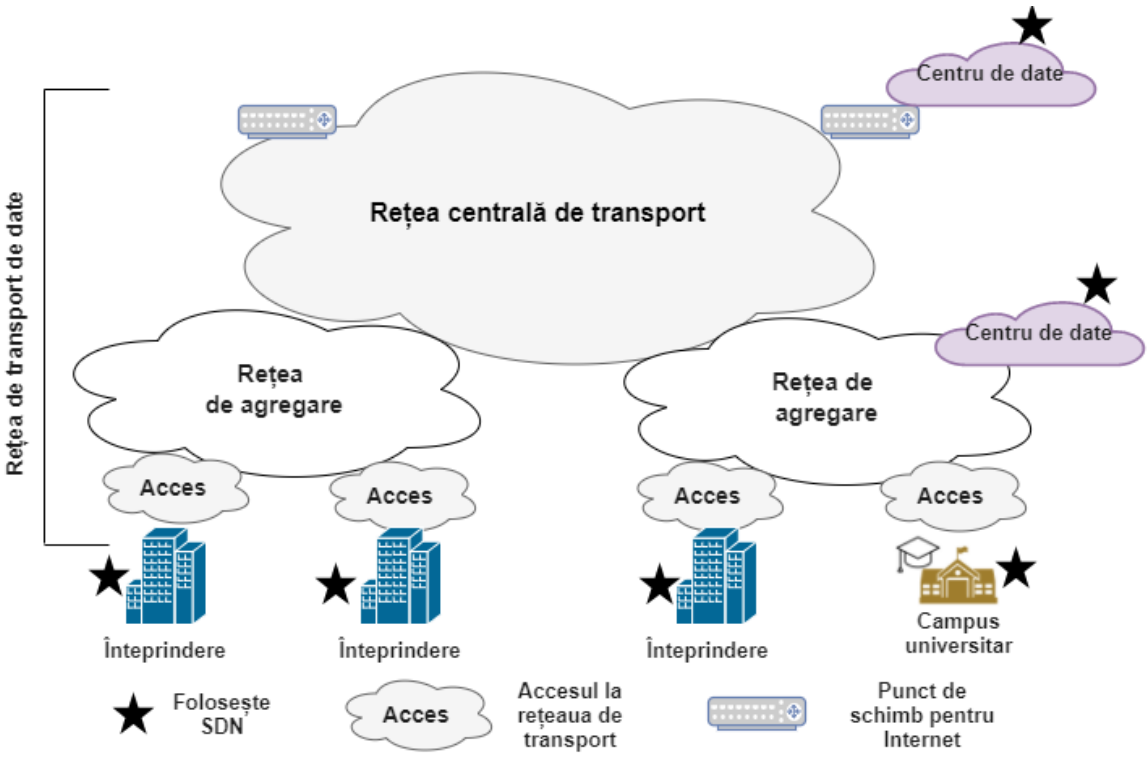

Fig. 2.3: Implementarea SDN în contextul rețelelor actuale [50]

Astfel, SDN este implementat deja cu succes în rețelele unor campusuri universitare (Universitatea Stanford, de exemplu, dat fiind faptul că această paradigmă își are originile acolo), în mari centre de date sau în întreprinderi. În cazul rețelelor de transport, încă se duc activităţi de standardizare care să permită operatorilor rețelelor de telecomunicaţii să implementeze această tehnologie în cel mai scurt timp, beneficiind astfel de avantajele pe care aceasta le oferă (cel mai important, din punctul lor de vedere, fiind reducerea costurilor).

Această secţiune va prezenta în continuare utilizarea SDN la momentul actual, în mai multe contexte: în rețelele din marile centre de date, în rețelele de tip hibrid, care combină rețelele tradiționale cu SDN sau prin studii de caz, implementări sau experimente efectuate pe teren (în rețele de producție).

2.3.1 SDN în centrele de date

Centrele de date reprezintă grupări de calculatoare cu putere mare de procesare și capacitate mare de stocare. Această idee nu este una nouă, fiind prezentă de câteva decenii, însă, în timp, au evoluat caracteristicile calculatoarelor, precum și numărul lor într-un centru de date. În zilele noastre acestea au ajuns sa conţină mii sau chiar zeci de mii de mașini fizice (servere) [51].

Creșterea numărului de servere și a puterii de stocare, combinate cu creșterea vitezei și a lărgimii de bandă din rețea a dus la necesitatea de a stoca din ce în ce mai multă informație în centre de date tot mai mari. În mod natural, acest lucru va însemna combinarea centrelor de date pentru a forma altele mai mari. Pe lângă acest aspect, se pune foarte mare accent în ultimul timp pe tehnologii de virtualizare, care permit unui server rularea mai multor mașini virtuale, pentru diferite aplicații sau servicii ale utilizatorilor.

A fost creat astfel un mediu dinamic, atât în cadrul centrelor de date, cât și între acestea. După cum este prezentat și în [52], operații precum prevenirea sau recuperarea în caz de dezastru, sau echilibrarea încărcării severelor au nevoie de o creștere a traficului între centrele de date, lucru ce duce la o administrare mai complexă a acelei rețele.

Centrele de date sunt folosite și în cazul oferirii de servicii de virtualizare de servere. Operaţiunile într-un astfel de mediu implică lucrul cu mașini virtuale și, de cele mai multe ori, necesitatea migrării acestora între diferite mașini sau centre de date. Asta înseamnă că diferite aplicații sau servicii trebuie să aibă o imagine proprie asupra rețelei dintre servere, care este doar o vedere la nivel logic, în comparaţie cu rețeaua fizică existentă. Așa cum este prezentat și în [1], crearea de astfel de rețele logice se poate face si fără SDN, prin metode folosite și în rețelele tradiționale, cum ar fi cu ajutorul unor VLAN-uri. Însă, în cazul centrelor de date foarte mari, acestea pot fi insuficiente. Cum ele pot fi exprimate într-un pachet Ethernet doar prin 12 biţi, numărul maxim de rețele virtuale este 4096. Apare astfel nevoia unor alte metode, care sunt însă mai complex de configurat.

O metodă pentru crearea de rețele logice care să fie prezentate unor aplicații sau servicii este chiar folosirea tehnologiei SDN, făcând abstracţie de rețeaua fizică. În [52–55] se prezintă diferite metode care pot fi aplicate în astfel de cazuri, sau care chiar au fost demonstrate.

Soluția SDN a fost chiar aplicată în rețele de producție. Așa cum se explică în [56], centrele de date ale Google folosesc deja această soluție, cu ajutorul protocolului OpenFlow. Există două laturi ale rețelei de arie largă - WAN (Wide Area Network) folosită de Google: o parte legată la Internet, care conţine traficul utilizatorilor și o parte internă, care transportă traficul dintre centrele de date. Cele două au caracteristici diferite ale traficului și, implicit, nevoi diferite. În acea rețea internă a fost implementat SDN, prin protocolul OpenFlow. Avantajele evidenţiate de Google includ o viziune de ansamblu asupra rețelei, un răspuns mai rapid la erori, timp mai scurt de implementare, actualizări mai rapide ale software-ului rețelei, fără pierdere de trafic sau existența unui mediu de test de mare fidelitate. Chiar dacă în 2012, atunci când a fost implementată această tehnologie în rețea, protocolul OpenFlow era încă la început și nu avea toate facilităţile pe care le oferă astăzi, Google a arătat încă de atunci că tehnologia SDN poate fi folosită, aducând foarte multe avantaje, în multe cazuri de utilizare din industria rețelisticii.

2.3.2 SDN în rețelele hibride

Rețelele hibride reprezintă o combinație între rețelele tradiționale și SDN, având scopul de a combina avantajele aduse de ambele, cât timp tehnologia SDN nu este încă destul de matură și prezintă unele dezavantaje. Un alt motiv pentru apariția acestor rețele hibride este imposibilitatea de a schimba rețelele de producție într-un timp foarte scurt, în același timp asigurând și funcționarea acestora în parametrii agreaţi.

Autorii din [57] prezintă oportunităţile și provocările aduse de o astfel de abordare. Ei propun patru modele ale rețelelor hibride: (i) rețele hibride bazate pe topologie, (ii) rețele hibride bazate pe servicii, (iii) rețele hibride bazate pe clase și (iv) rețele hibride integrate. Avantajele prezentate de aceste modele hibride sunt flexibilitatea, robusteţea, extensibilitatea, costuri mai mici ale implementării, date de faptul ca nu toată rețeaua este actualizată. Pe de altă parte, această abordare vine și cu dezavantaje, reprezentate de nevoia de a administra paradigme eterogene și garantarea faptului că interacţiunea dintre ele este benefică.

În [58–60] se propun sisteme care să permită o implementare incrementală a SDN în rețele de tip hibrid pentru rețele ale întreprinderilor și ale furnizorilor de servicii. Autorii evidenţiază faptul că implementarea directă, într-o rețea de producție a acestei noi tehnologii este imposibilă și ar conduce atât la probleme de implementare, cât și la probleme operaționale în rețea. Ei susţin că implementarea SDN în puncte cheie ale rețelelor va aduce avantajele acestei tehnologii, fără nevoia de a înlocui toată rețeaua de la început. Pentru a determina aceste puncte cheie, autorii propun un *planificator al implementării*, care să analizeze topologia de rețea, împreună cu informaţiile istorice despre trafic și constrângeri de resurse.

Lucrarea [61] prezintă o altă abordare pentru aceste rețele hibride. Autorii propun un echipament de control al rețelei care să poată interacţiona atât cu echipamentele care folosesc un control centralizat, cât și cu cele care au un control distribuit. Controlarea echipamentelor vechi din rețea, care nu suportă protocolul OpenFlow se face chiar cu ajutorul acestuia. Astfel, dacă se vrea alterarea fluxului de date dintr-un echipament vechi, autorii propun folosirea unei extensii a OpenFlow, numită *LegacyFlowMod*, care să instruiască un echipament OpenFlow să trimită un alt pachet special către echipamentul vechi, astfel încât acesta să își schimbe tabela de dirijare.

Autorii din [62, 63] susţin de asemenea că o soluție hibridă este mai potrivită. Propun astfel folosirea unor rețele care să combine SDN cu protocolul OSPF (Open Shortest Path First), păstrând avantajele date de acest protocol vechi de rutare, care a fost folosit cu succes câteva zeci de ani și, în același timp, aducând și câteva din avantajele propuse de SDN.

2.3.3 Studii de caz. Implementări. Experimente în rețele de producție

Există numeroase studii de caz sau experimente în rețele de producție care demonstrează utilitatea SDN și faptul că această tehnologie poate fi implementată cu succes în rețelele din zilele noastre.

În [64] se prezintă soluţia implementată în rețeaua spitalului *Kanazawa University*, din Japonia. S-a folosit tehnologia SDN, cu ajutorul protocolului OpenFlow pentru a uşura munca de administrare a rețelei spitalului, care devenea din ce în ce mai complexă, pe măsură ce noi echipamente medicale trebuiau adăugate. Rețeaua, fiind operată de personalul spitalului, era vulnerabilă la erorile provocate de greşelile umane. Trebuia asigurată izolarea între anumite departamente ale spitalului, în unele cazuri, astfel că administrarea rețelei devenea greoaie. Cu ajutorul SDN a fost creată o soluție în care infrastructura de rețea a devenit mai uşor de administrat, prin aplicații software care rulează deasupra echipamentului de control SDN, permiţând departamentelor crearea de rețele virtuale proprii, asigurând în același timp conectivitate între departamente. Astfel, rețeaua spitalului a devenit stabilă și pregătită pentru dezvoltările rapide care apar în industria medicală în zilele noastre.

Lucrarea [65] ilustrează folosirea SDN în cadrul unei rețele optice de transport a unui furnizor de servicii prin Ethernet. Autorii au implementat această tehnologie pentru oferirea de servicii în acea infrastructură de rețea. Obiectivele urmărite au fost simplificarea administrării rețelei și furnizarea de trafic, în funcție de serviciul solicitat (de exemplu, lățime de bandă la cerere).

În [66] se analizează posibilitatea folosirii tehnologiei SDN în rețelele Ethernet industriale. Autorul evidenţiază punctele în care această tehnologie s-ar putea aplica în rețelele industriale și avantajele pe care o astfel de implementare le-ar aduce.

Autorii din [67] urmăresc evoluția implementărilor SDN pornind de la nivelul unei rețele dintr-un simplu laborator din cadrul Universităţii Stanford și până la rețele de nivel naţional. Cu ajutorul acestor studii se evidenţiază compromisurile care apar în implementarea acestei noi tehnologii în rețele de producție. De exemplu, pentru o simplă rețea a unui campus universitar, un singur server poate susţine tot planul de control al rețelei, incluzând echipamentul de control și diferitele aplicații care rulează acolo pentru administrarea acesteia. Primele cazuri de utilizare demonstrate cu ajutorul SDN au fost rutarea pe cea mai scurtă cale de nivel 2, învăţarea adreselor de nivel 2 - MAC (Medium Access Control), descoperirea topologiei cu ajutorul LLDP (Link Layer Discovery Protocol) sau colectarea de statistici de la comutatoarele din rețea. Aceste experimente au subliniat câteva aspecte importante pentru performanţelor rețelelor SDN: timpul necesar configurării fluxurilor de date în rețea, numărul de intrări în tabelele de fluxuri ale comutatoarelor. De asemenea, este important ca un comutator să fie hibrid, adică să suporte atât protocolul OpenFlow, cât și pe cele tradiționale. O altă fază a experimentelor conduse de aceşti autori a constat în împărțirea infrastructurii de rețea în mai multe rețele logice destinate diferitelor aplicații.

Toate aceste experimente și studii de caz prezentate au avut la bază protocolul OpenFlow, ducând la evoluția acestuia. Acest lucru nu trebuie să ne ducă însă cu gândul că tehnologia SDN înseamnă doar OpenFlow. După cum se va vedea în capitolele următoare, în special în cazul rețelelor de transport de date prin microunde, există și alte protocoale cu ajutorul cărora se pot aduce principiile SDN în rețea.

Capitolul 3

Unelte SDN în contextul reţelelor de transport de date fără fir

Așa cum a fost prezentat în capitolul anterior, o mare parte din cercetarea și implementările SDN se bazează pe protocolul OpenFlow. Acesta, însă, nu se poate aplica în orice aspect al unei rețele (de exemplu în rețelele de transport de date fără fir). În [46] s-a demonstrat faptul că se poate extinde protocolul OpenFlow pentru a cuprinde atribute specifice echipamentelor de transport de date fără fir, însă s-a ajuns la concluzia că este totuși nevoie de un model informațional care să abstractizeze astfel de echipamente pentru a facilita administrarea acestora prin aplicații software.

Astfel, grupurile de lucru din ONF au formulat recomandări pentru astfel de modele informaționale care să poată fi aplicate în acest context. În Martie 2015 a fost publicată de către ONF prima versiune (1.0) a modelului informațional de bază, TR-512, [68], apoi în Noiembrie 2015 versiunea 1.1 [69], ca în Septembrie 2016 să fie publicată versiunea curentă, 1.2, purtând numele TR-512.1 [70]. Modelul informațional de bază este doar un schelet, care poate fi folosit în toate tipurile de rețele de transport, indiferent de natura acestora. Pentru rețelele de transport de date fără fir, ONF a publicat în Decembrie 2016 și modelul informațional pentru microunde [71], pentru abstractizarea echipamentelor din acest tip de rețele, care este integrat cu TR-512.1.

În următoarele secţiuni aceste modele vor fi detaliate, pentru a putea mai apoi înțelege arhitectura simulatoarelor dezvoltate în această lucrare. Apoi va fi prezentat protocolul NETCONF și modul în care utilizează aceste modele informaționale, precum și alegerea unui cadrul software cu sursă deschisă care să implementeze un server pentru acest protocol. În finalul capitolului se va prezenta arhitectura demonstrațiilor de concept desfășurate în cadrul ONF, pentru o înțelegere mai bună a motivației din spatele creării simulatoarelor care fac scopul acestei lucrări.

3.1 Modelul informaţional de bază - ONF TR-512.1 (*Core Model*)

Modelul informațional de bază, *CoreModel* reprezintă o recomandare făcută de grupul *Information Modeling* din cadrul ONF. Aceasta propune un model care să descrie resursele din planul de date al unei rețele de transport, indiferent de tehnologia folosită, cu scopul de a fi folosit în activitățile de control și administrare.

Un model informațional descrie lucrurile dintr-un domeniu, în ceea ce priveşte obiectele, proprietăţile lor (reprezentate prin atribute) și relaţiile dintre acestea [68]. Scopul dezvoltării unui astfel de model este acela de a fi folosit de către echipamentele de control SDN, pentru administrarea automată a unor astfel de rețele de transport, în concordanţă cu arhitectura SDN. Echipamentul de control va prezenta cu ajutorul acestui model viziunea sa asupra rețelei către clienţii acestuia (care pot fi aplicații software sau alte echipamente de control).

CoreModel propune obiecte de bază care reprezintă planul de date al unei rețele, care sunt însă independente de tehnologia folosită pentru transportul datelor. Acestea pot fi apoi folosite pentru dezvoltarea de modele informaționale specifice pentru anumite tehnologii (de exemplu tehnologii fără fir - microunde sau unde milimetrice, tehnologii optice, etc.). Modelul conţine și obiecte care pot fi folosite în aplicații specifice, însă toate acestea sunt independente de protocoalele care ar putea fi folosite în planul de control. El este propus în limbajul UML (Unified Modeling Language) și se bazează și pe alte modele informaționale, propuse de alte organizaţii care dezvoltă standarde.

O vedere de ansamblu simplificată, folosind UML se poate vedea în Figura 3.1. Blocurile relevante pentru simulatoarele dezvoltate vor fi detaliate în paragrafele următoare. Nu se vor detalia toate obiectele care alcătuiesc modelul informațional de bază deoarece, așa cum este sugerat în recomandarea ONF, modelul poate fi redus și simplificat, în funcție de nevoile pentru care este folosit.

3.1.1 Obiectul *NetworkElement (NE)*

Obiectul *NetworkElement* reprezintă un element de rețea - NE (Network Element), adică, în cazul SDN, un echipament de dirijare din planul de date, sau, în cazul în care există virtualizare, un element virtual de rețea vizibil în interfaţa care folosește acea virtualizare.

Pentru o interfață directă între echipamentul de control SDN către un echipament de rețea, obiectul *NetworkElement* delimitează domeniul de control pentru resursele echipamentului, cum ar fi încapsularea folosită, multiplexarea, demultiplexarea, funcțiile asociate operaţiilor de administrare și de mentenanţă, etc. De asemenea, obiectul *NetworkElement* defineşte domeniul spaţiilor de adresare pentru identificarea obiectelor care reprezintă resursele conţinute de echipamentul respectiv.

În cazul în care se folosesc metode de virtualizare, obiectul *NetworkElement* reprezintă un element virtual de rețea. Asocierea unui element virtual cu unul real este responsabilitatea echipamentului de control SDN. Cu ajutorul interfeţei de tip

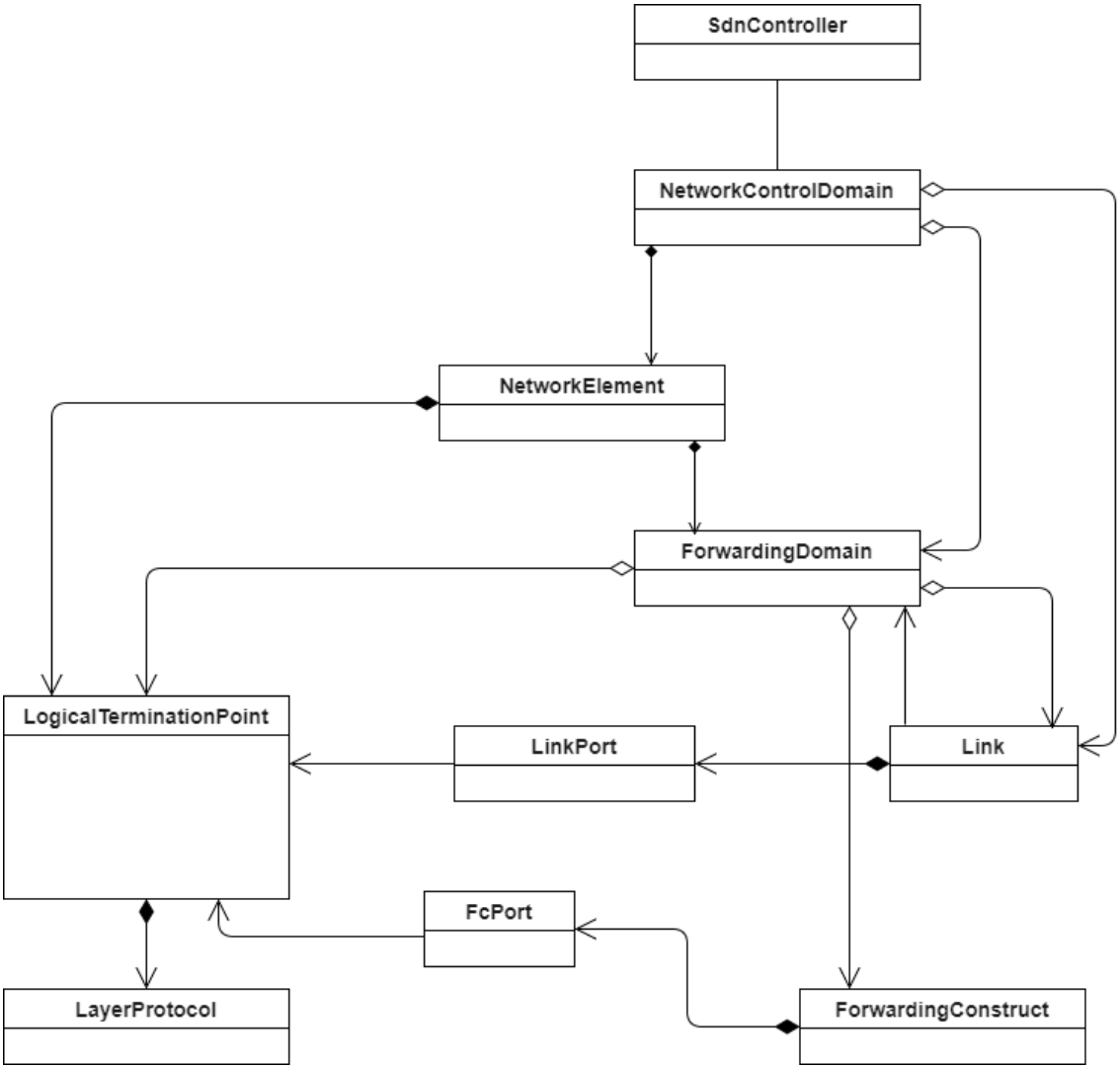

Fig. 3.1: Reprezentare UML simplificată a *CoreModel*

southbound acesta poate crea sau şterge în mod dinamic astfel de obiecte pentru a oferi diferite vederi asupra rețelei, în funcție de nevoile aplicațiilor care se află deasupra echipamentului de control.

3.1.2 Obiectele *LogicalTerminationPoint (LTP)* și *LayerProtocol (LP)*

Obiectul *LogicalTerminationPoint* - punct logic de terminaţie - cuprinde terminaţiile, adaptările sau funcțiile asociate operaţiilor de administrare și de mentenanţă ale unuia sau mai multor niveluri de transport. Prin natura sa, acest obiect suportă toate tipurile de protocoale, inclusiv cele pentru comutare de pachete sau pentru comutare de circuite. Fiecare nivel de transport este reprezentat de o instanţă a unui obiect *LayerProtocol*, instanţă care poate fi folosită pentru a controla terminaţiile sau

funcțiile de administrare și mentenanţă ale nivelului respectiv, sau pentru adaptarea (încapsularea sau multiplexarea semnalului client).

Dacă relația client-server între resursele echipamentului reprezentate de obiectele *LTP* și *LP* are un grad de asociativitate de 1:1 și este imuabilă, atunci un obiect *LTP* poate conține mai multe obiecte *LP*, de niveluri diferite de transport. Altfel, obiectele *LP* de pe niveluri diferite trebuie să aibă asociate obiecte *LTP* diferite.

Scopul acestor obiecte este acela de a oferi suport din perspectiva controlului și a administrării fără a fi nevoie de a defini atribute specifice unei anumite tehnologii, permițând astfel extinderea modelului fără a depinde de proprietățile tehnologiei respective, oferind flexibilitate sporită.

Un atribut important al obiectelor *LP* este reprezentat de către *layerProtocolName*. Acesta reprezintă nivelul de transport al obiectului și poate avea următoarele valori, conform [70]:

- Nivel 0: OPS (Optical Protection Switch), OTS (Optical Transmission Section), OMS (Optical Multiplex Section), OCh (Optical Channel);
- Nivel 1: OTU (Optical channel Transport Unit), ODU (Optical Data Unit);
- Nivel 2: Carrier Ethernet: ETY (Ethernet Physical Layer), ETH (Ethernet MAC Layer); MPLS-TP (Multi-Protocol Label Switching Transport Profile);
- Proprietăţi specifice nivelului de transport asociate cu obiectul *LP*.

3.1.3 Obiectul *ForwardingConstruct (FC)*

Obiectele *ForwardingConstruct (FC)* sunt folosite pentru a realiza dirijarea informației caracteristice nivelului de transport dat de obiectul *LP* și oferă posibilitatea de a permite dirijarea între două sau mai multe obiecte *LTP*. Astfel, obiectele *FC* sunt independente de nivelul de transport folosit și suportă orice formă de pachete sau circuite.

Asocierea între *FC* și *LTP* se face prin port-uri, în care fiecare dintre acestea are un rol în contextul obiectului *FC*. Dirijarea traficului între port-urile asociate se face în funcție de tipul de obiect *FC*. Un astfel de obiect poate fi asociat unui singur obiect *FD*. Ele pot fi definite recursiv (un obiect *FC* poate fi parte din alt obiect *FC*), însă la cel mai mic nivel de recursivitate acesta reprezintă de fapt o legătură în matricea de comutatoare a elementului de rețea.

Obiectele *FC* pot fi folosite pentru a reprezenta orice fel de conexiune, cum ar fi punct la punct, punct la multi-punct sau multi-punct la multi-punct.

3.1.4 Obiectul *FC Port*

Obiectele *FC Port* sunt folosite, așa cum a fost prezentat anterior, la asocierea dintre obiectele *FC* și *LTP*. Dirijarea traficului între aceste obiecte se face conform tipului de *FC*. De exemplu, *FC Port* poate reprezenta un punct protejat (de încredere) sau un punct care protejează (de rezervă), în cazul în care rolul obiectului *FC* este unul de protecție.

3.1.5 Obiectul *ForwardingDomain (FD)*

ForwardingDomain - domeniul de dirijare - este un obiect care modelează componenta topologiei care permite dirijarea pachetelor între diferite puncte ale resurselor echipamentului de rețea (reprezentate prin alte obiecte, de tipul *LogicalTermination-Point*). Lista punctelor logice de terminaţie care pot fi folosite de către un domeniu de dirijare este parte a acestui obiect. *ForwardingDomain* poate conţine zero sau mai multe obiecte de tip *ForwardingConstruct*, indiferent de nivelul la care se află acestea (Ethernet, MPLS, optic, etc.).

Acest domeniu de dirijare oferă contextul pentru crearea, modificarea sau ștergerea obiectelor de tip *ForwardingConstruct*. Obiectul *ForwardingDomain* dintr-un element de rețea poate reprezenta comutatorul sau gruparea de comutatoare din echipamentul de dirijare.

3.2 Modelul informaţional pentru microunde - ONF TR-532 (*Microwave Information Model*)

Modelul informațional pentru microunde [71] a apărut in Decembrie 2016 ca o recomandare formulată de grupul OTWG din cadrul ONF. Scopul acestuia este de a modela un echipament de transport de date fără fir, pentru a putea fi folosit de echipamentele de control SDN, în încercarea de a asigura o independenţă față de producătorii de echipamente. Chiar dacă este denumit *model informațional pentru microunde*, acesta poate fi aplicat fără probleme nu numai echipamentelor ce funcționează în spectrul microundelor, ci și echipamentelor care funcționează în benzi de frecvenţă mai înalte (lungimi de undă milimetrice), care încep să își facă tot mai mult simţită prezenţa în rețelele actuale de transport.

TR-532 este de fapt o extensie specifică tehnologiei WT a modelului informațional de bază, versiunea 1.2 (TR-512.1). Legătura cu *CoreModel* se face prin extinderea clasei de obiecte *LayerProtocol*. Astfel, *MicrowaveModel* conţine şase pachete condiţionale caracteristice tehnologiilor folosite pentru transport, care au în nume extensia **_Pac*:

- *MW_AirInterface_Pac*;
- *MW_AirInterfaceDiversity_Pac*;
- *MW_PureEthernetStructure_Pac*;
- *MW_HybridMWStructure_Pac*;
- *MW_EthernetContainer_Pac*;
- *MW_TdmContainer_Pac*

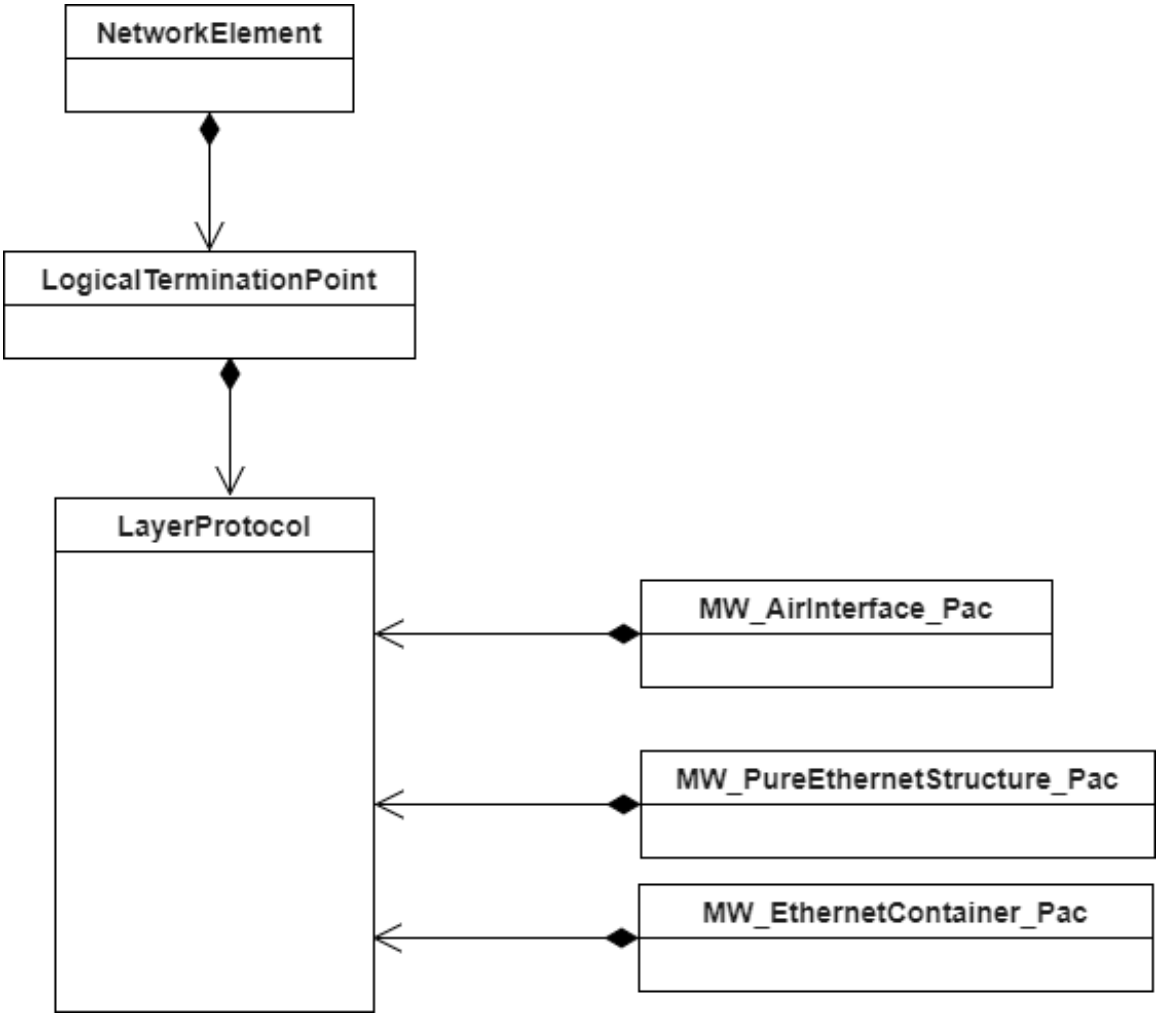

Fig. 3.2: Reprezentare UML simplificată a *MicrowaveModel* și legătura acestuia cu *CoreModel*.

O vedere de ansamblu simplificată a acestui model, în limbajul UML, care conține doar obiectele relevante pentru simulatoarele dezvoltate, împreună cu legătura acestuia cu *CoreModel* este ilustrată în Figura 3.2.

În următoarele paragrafe se vor detalia obiectele acestui model care sunt importante din punctul de vedere al simulatoarelor dezvoltate în această lucrare.

3.2.1 Obiectul *MW_AirInterface_Pac*

Obiectul *MW_AirInterface_Pac* reprezintă o interfață radio fizică a unui echipament. Este denumit în recomandare ca *punct de terminaţie a traseului secţiunii fizice de microunde* - MWPS-TTP (Microwave Physical Section - Trail Termination Point), astfel că nivelul de transport al obiectului *LayerProtocol* asociat este MWPS (Microwave Physical Section). O reprezentare simplificată în limbajul UML a *MW_AirInterface_Pac* se poate observa în Figura 3.3.

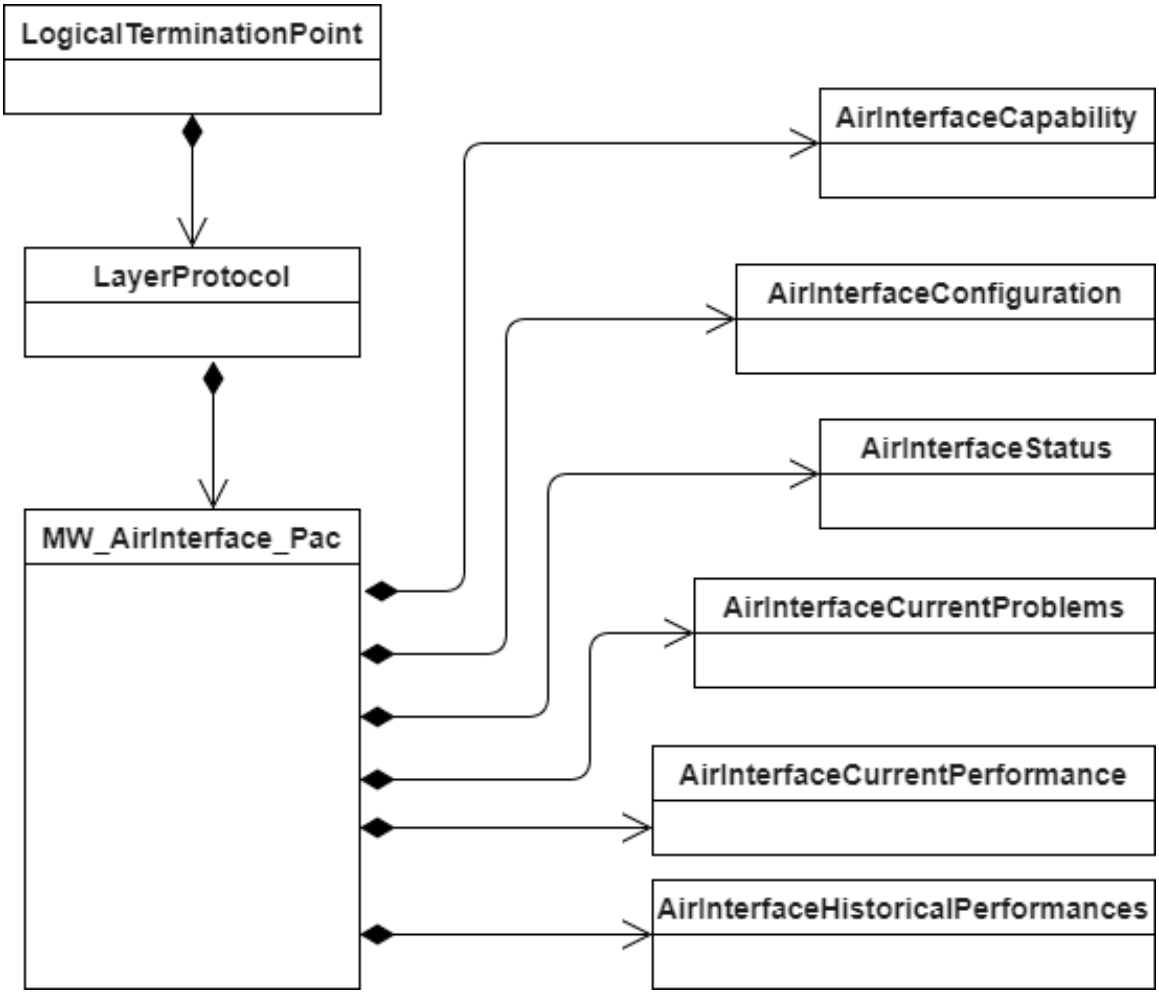

Fig. 3.3: Reprezentare UML simplificată a obiectului *MW_AirInterface_Pac*.

Acest obiect conţine alte câteva obiecte care modelează caracteristice unei interfețe radio fizice, cum ar fi: (i) capabilităţi ale modemului și ale transmiţătorului interfeţei radio asociate (de exemplu modulaţiile suportate pentru transmisie, valorile posibile ale puterii de transmisie, intervalul de frecvenţe suportate de transmiţător sau de receptor, alarmele expuse de interfață, suportul interfeţei pentru modulaţie adaptivă, etc.), (ii) parametrii configurabili ai interfeţei radio (de exemplu numele interfeţei, lărgimea de bandă a canalului de transmisie/de recepţie, frecvenţele folosite pentru transmisie/recepţie, puterea de transmisie, intervalul pentru modulaţia care poate fi folosită, diferite alte caracteristici configurabile ale interfeţei, cum ar fi XPIC (Cross Polarization Interference Cancellation), MIMO (Multiple-Input Multiple-Output), criptarea datelor, etc.), (iii) parametrii care descriu starea interfeţei la un anumit moment de timp (de exemplu frecvenţele actuale de transmisie/recepţie, nivelurile actuale de putere a semnalului de transmisie/recepţie, modulaţia actuală folosită, raportul semnal-zgomot măsurat de către modem, temperatura actuală a unităţii radio, etc.), (iv) problemele actuale ale interfeţei radio (adică alarmele care apar pe interfață

la un moment dat), (v) valorile actuale ale parametrilor de performanţă a interfeţei și (vi) valorile istorice ale parametrilor de performanţă a interfeţei.

3.2.2 Obiectul *MW_PureEthernetStructure_Pac*

Obiectul *MW_PureEthernetStructure_Pac* este o reprezentare logică a unei interfețe radio capabilă să transporte doar trafic Ethernet. Este denumit în recomandare ca *punct de terminaţie a traseului secţiunii microunde* - MWS-TTP (Microwave Section - Trail Termination Point), astfel că nivelul de transport al obiectului *LayerProtocol* asociat este MWS (Microwave Section). Acest obiect este reprezentat într-un mod simplificat, în limbajul UML, în Figura 3.4. Asocierea cu o interfață fizică radio se face la nivelul *CoreModel*, printr-o relație de tip client-server.

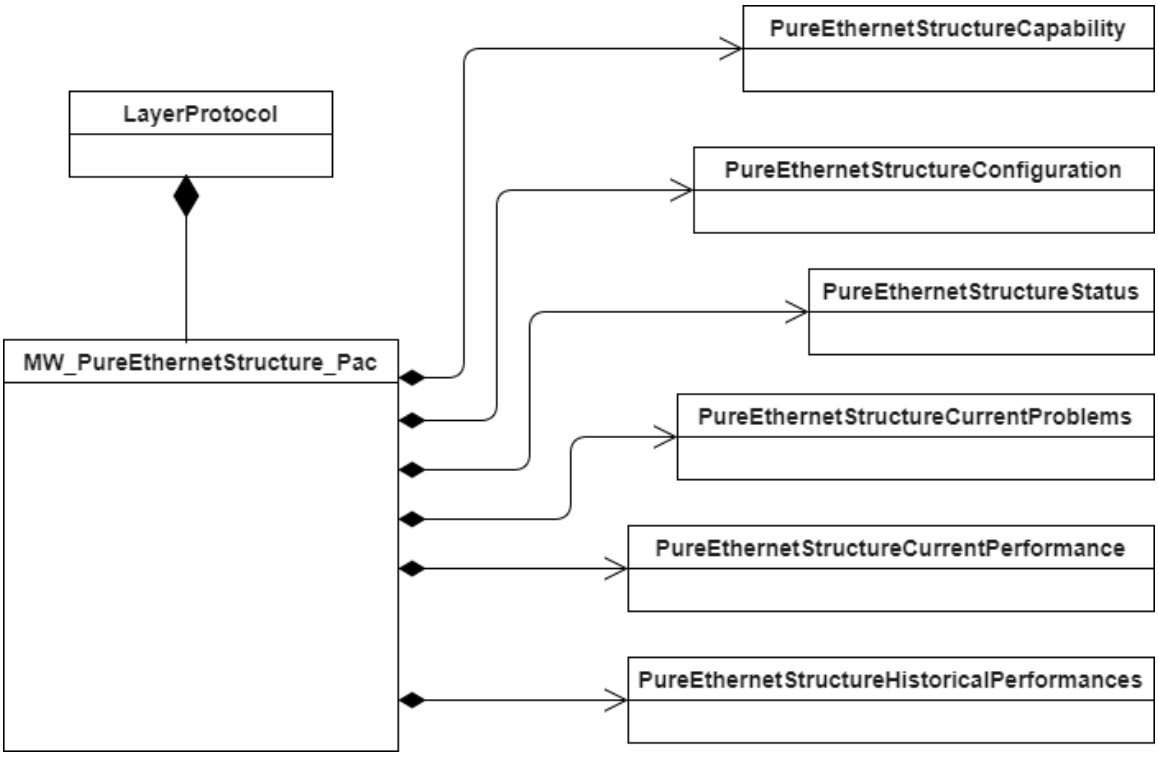

Fig. 3.4: Reprezentare UML simplificată a obiectului *MW_PureEthernetStructure_Pac*.

Structura obiectelor conţinute de către *MW_PureEthernetStructure_Pac* este similară cu cea a obiectului *MW_AirInterface_Pac*. Conţine obiecte care reprezintă (i) capabilitățile acestei interfețe logice (de exemplu alarmele aplicabile ei sau identificatorul structurii respective, care poate fi folosit de alte obiecte), (ii) parametrii configurabili ai interfeţei logice (de exemplu severitatea alarmelor pe care această interfață le expune), (iii) parametrii care descriu starea interfeţei logice la un anumit moment de timp, (iv) problemele actuale ale interfeţei logice, (v) valorile actuale ale parametrilor de performanţă a interfeţei logice și (vi) valorile istorice ale parametrilor de performanţă a interfeţei logice.

3.2.3 Obiectul *MW_EthernetContainer_Pac*

Obiectul *MW_EthernetContainer_Pac* reprezintă de asemenea o interfață logică și este denumit în recomandare *punct de terminaţie a conexiunii unui client de microunde*, pentru un semnal Ethernet client. Practic, este un interfață logică ce are rol de container pentru traficul Ethernet care este transmis de echipament prin radio. În raport cu obiectul *LayerProtocol* acesta are un nivel de transport denumit ETC (Ethernet Container).

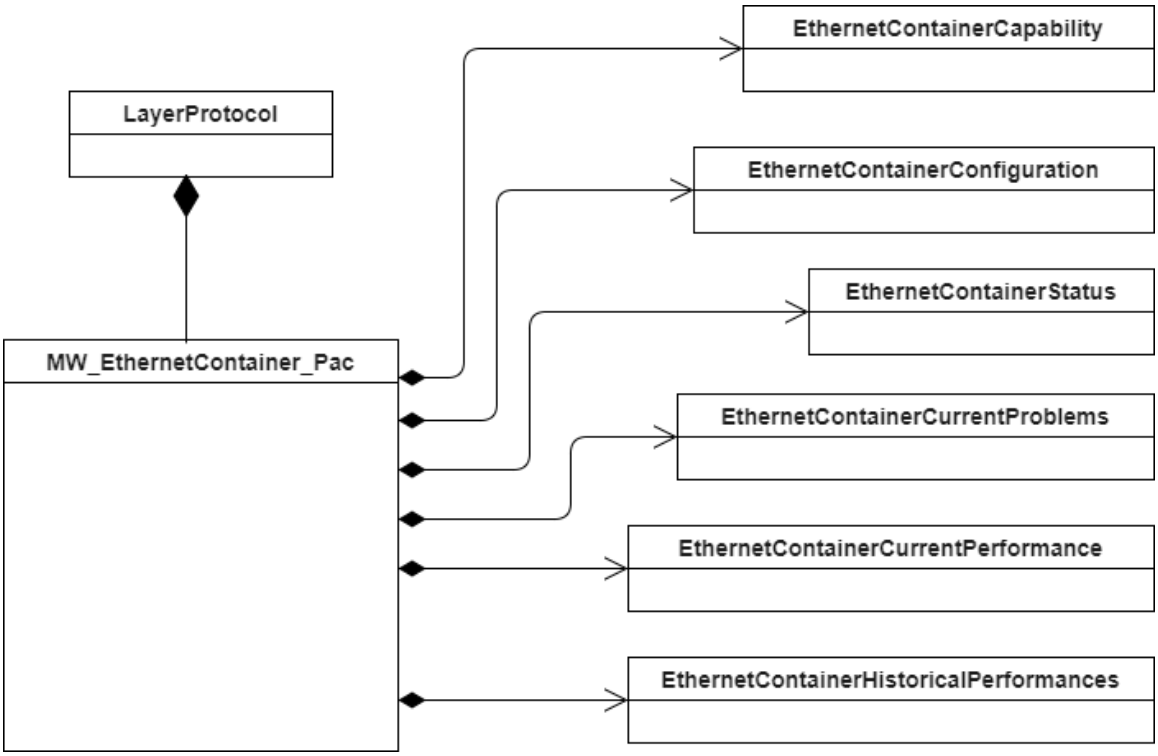

Fig. 3.5: Reprezentare UML simplificată a obiectului *MW_EthernetContainer_Pac*.

Și în cazul obiectului *MW_EthernetContainer_Pac* se păstrează aceeaşi structura a obiectelor pe care le conţine, ca în cazul celorlalte două obiecte detaliate anterior. Astfel, acesta prezintă obiecte care reprezintă (i) capabilitățile containerului (de exemplu dacă există compresie la diferite niveluri, sau criptare a datelor sau alarmele pe care această interfață le expune), (ii) parametrii configurabili ai containerului (de exemplu un identificator al containerului, identificatoarele segmentelor folosite pentru a transporta traficul Ethernet asociat acestui container, etc.), (iii) parametrii care descriu starea containerului, (iv) alarmele la momentul actual de timp pe care containerul le raportează, (v) valorile actuale ale parametrilor de performanţă a containerului și (vi) valorile istorice ale parametrilor de performanţă a containerului. O reprezentare grafică simplificată în limbajul UML a obiectului *MW_EthernetContainer_Pac* poate fi găsită în Figura 3.5.

3.3 Protocolul NETCONF și limbajul YANG

Network Configuration Protocol - NETCONF - este un protocol de administrare a echipamentelor de rețea care oferă posibilitatea de a instala, manipula și şterge configuraţii ale echipamentelor de rețea. A fost prima oară definit în RFC 4741 [72], în decembrie 2006, ca mai apoi să fie revizuit în RFC 6241 [73], în iunie 2011. A apărut ca o nevoie a industriei rețelisticii, care până atunci utiliza alte modalităţi de administrare, precum CLI (Command Line Interface), SNMP (Simple Network Management Protocol) sau CORBA (Common Object Request Broker Architecture) [74].

Chiar dacă tehnicile folosite până acum pentru administrarea echipamentelor sunt utilizate de mai bine de douăzeci de ani, acestea și-au dovedit limitările. De exemplu, prima abordare, cea folosind interfețe cu linie de comandă, implică o foarte mare dependenţă de producătorii de echipamente [75], în timp ce protocolul SNMP este folosit îndeosebi pentru colectarea de alarme și de valori ale parametrilor de performanţă de la echipamente și mai puţin pentru configurarea acestora. NETCONF încearcă să diminueze aceste limitări propunând o abordare nouă și inovativă. În acest scop a fost dezvoltat și limbajul YANG (Yet Another Next Generation), în RFC 6020 [76], care să modeleze datele folosite de acest protocol.

În următoarele secţiuni se vor detalia modul de funcţionare a protocolului NETCONF, punând accent pe aspectele relevante din punctul de vedere al simulatoarelor dezvoltate în această lucrare și limbajul de modelare YANG.

3.3.1 NETCONF

Protocolul NETCONF defineşte un mecanism simplu de administrare a echipamentelor de rețea, prin care parametrii configurabili pot fi ceruţi dispozitivelor, manipulaţi și apoi retrimiși pentru configurarea echipamentelor. Cu ajutorul acestuia dispozitivele pot expune interfețe de programare care să poată fi utilizate de aplicații software pentru administrare.

Mecanismul propus de NETCONF constă în utilizarea paradigmei apelurilor de proceduri la distanţă - RPC (Remote Procedure Call). Un server rulează pe fiecare dispozitiv, aşteptând conexiuni de la clienţi prin canale securizate și folosind protocoale orientate spre conexiune, precum SSH (Secure Shell), TLS (Transport Layer Security), BEEP (Blocks Extensible Exchange Protocol), SOAP (Simple Object Access Protocol) [77]. Clienţii își codează acel apel de procedură la distanţă cu ajutorul limbajului XML (Extensible Markup Language), iar serverul răspunde printr-un mesaj codat cu același limbaj. În timpul stabilirii conexiunii între un client și un server NETCONF, acesta din urmă își expune capabilitățile pe care le are, astfel încât clientul va fi informat asupra acestora, având posibilitatea de a-și schimba comportamentul pentru a profita de caracteristicile serverului.

NETCONF propune folosirea a două tipuri de date: (i) date care descriu starea unui dispozitiv, adică valorile parametrilor operaţionali sau valori de contorizare a parametrilor de performanţă, toate acestea fiind valori care pot fi doar citite din

echipamente și (ii) date configurabile pe dispozitiv, care pot fi atât scrise cât și citite. Pentru manipularea acestora se oferă diferite operații, cele mai importante fiind:

- *get* - această operaţie permite unui client să ceară unui dispozitiv atât date de stare, cât și parametri configurabili. Evident, se pot aplica filtre pentru a întoarce doar anumiţi parametri dintre cei pe care îi expune echipamentul;
- *get-config* - această operaţie oferă unui client posibilitatea să ceară unui server doar parametri configurabili, excluzându-i pe cei de stare;
- *edit-config* - aceasta este o operaţie prin care un client poate schimba valorile unor parametri configurabili pe un dispozitiv.

Pentru stocarea acestor date pe echipament, protocolul NETCONF propune la nivel conceptual un loc în care să se stocheze și prin care să se acceseze informaţia, denumit *datastore*. Reprezentarea acestuia este la alegerea fiecărui dispozitiv, putând fi implementat prin fişiere, baze de date, locaţii de memorie flash, etc. Există trei tipuri de astfel de locuri pentru stocarea datelor:

- *startup datastore* - acest tip reprezintă un *datastore* ce poate conţine valori implicite ale parametrilor configurabili ai dispozitivului de rețea și poate fi încărcată în echipament în momentul inițializării acestuia. El este prezent doar în dispozitivele care suportă această separare între configurația curentă, care rulează în echipament și configurația care se încarcă în momentul inițializării;
- *running datastore* - acest tip reprezintă toţi parametrii configurabili activi la momentul curent, care sunt prezenţi în dispozitiv. Acest tip există întotdeauna pe un echipament;
- *candidate datastore* - acest tip reprezintă o copie a tuturor parametrilor configurabili ai unui dispozitiv. Modificarea acestora nu influenţează configurația curentă a dispozitivului, însă aceasta se poate aplica, prin copierea cu ajutorul unei tranzacţii, înlocuind *running datastore*. Nu toate tipurile de echipamente suportă această capabilitate.

NETCONF propune și un mecanism prin care să protejeze accesul concurent la scrierea parametrilor configurabili ai dispozitivelor. Astfel, un client poate bloca o parte sau chiar toată *running datastore* cât timp execută operații prin care schimbă valorile acestor parametri, oferind astfel protecție datelor pe care le modifică. Dispozitivul însă trebuie să se asigure că aceste valori nu pot fi modificate în același timp prin alte căi, cum ar fi SNMP sau CLI.

Acest protocol oferă și posibilitatea serverelor NETCONF să trimită informări către clienţi cu privire la anumite evenimente care se petrec în dispozitiv. Spre deosebire de celelalte mesaje, aceste notificări sunt inițiate de către server și sunt trimise tuturor clienților și-au exprimat dorinţa de a le primi (prin trimiterea către server a unui mesaj de abonare).

3.3.2 YANG

Yet Another Next Generation - YANG este un limbaj de modelare a informaţiei dezvoltat specific pentru protocolul NETCONF. Acesta descrie atât datele de configuraţie și de stare pe care un dispozitiv le poate expune pentru a fi folosite de către protocol, cât și apelurile de proceduri la distanţă sau notificările. A apărut în octombrie 2010 ca RFC 6020, fiind dezvoltat de IETF.

În momentul apariției NETCONF, dat fiind faptul că se baza pe limbajul XML, soluţia naturală pentru definirea modelului de date folosit de protocol era utilizarea soluţiilor XML existente pentru modelarea informaţiei, precum schemele XML sau Relax NG [77]. Aceste soluții aveau însă dezavantajele de a fi greu de folosit și de a avea o lizibilitate redusă.

Astfel, un nou limbaj a fost dezvoltat: YANG. Au fost considerate mai multe aspecte în dezvoltarea acestuia: lizibilitatea, o abordare orientată pe obiecte și o oarecare similaritate cu limbajele de programare. YANG îndeplineşte aceste condiţii, fiind folosit pentru a descrie ierarhii de noduri, care pot reprezenta notificări, apeluri de proceduri la distanţă sau parametri de stare sau de configuraţie și pot fi folosite de operaţiile NETCONF. Informaţiile sunt stocate în modele YANG și precum în limbajele de programare, un model poate include date din alt model, oferind astfel posibilitatea de a crea seturi de modele de date generice și reutilizabile [78]. Acestea descriu atât nodurile într-un mod concis și clar, cât și interacţiunile dintre ele [79].

YANG descrie informaţia într-un mod ierarhic, astfel că un fiecare nod are, pe de o parte, un nume și pe de altă parte o valoare sau un set de noduri copil. Se pot descrie și constrângeri ce pot fi aplicate asupra apariției sau valorii unor noduri, bazându-se pe prezenţa sau valoarea altor noduri ale ierarhiei.

Există mai multe tipuri de noduri definite în limbajul YANG. Cele relevante din punctul de vedere al simulatoarelor implementate în această lucrare sunt:

- *grouping* - acesta este, așa cum o sugerează și numele, un nod care reprezintă o grupare de noduri. După ce este definit el poate fi utilizat în același sau în alte module sau sub-module;
- *list* - acest tip de nod YANG defineşte o listă de noduri, iar intrările în această listă sunt distinse prin noduri care reprezintă cheia intrării respective;
- *typedef* - este folosit pentru definirea unui tip de date care poate fi utilizat ulterior de alte noduri;
- *rpc* - este folosit pentru modelarea apelurilor de proceduri la distanţă, prin definirea numelui procedurii și a parametrilor de intrare și de ieșire;
- *notification* - acest tip de nod se folosește pentru descrierea unei notificări NETCONF pe care un server o poate genera, prin modelarea conținutului acesteia;
- *leaf* - nodurile frunză reprezintă nivelul cel mai jos al ierarhiei si descriu un parametru al dispozitivului, care poate fi de stare (poate fi doar citit) sau de configurare (poate fi și scris și citit).

Așa cum a fost prezentat anterior, ONF dezvoltă modelele informaționale pe care le recomandă cu ajutorul limbajului UML, care este mult mai general si mai puţin specializat decât YANG. Însă, pentru a putea fi folosite într-un mod facil, ONF a dezvoltat și o unealtă software care să transforme modelele din limbajul UML în limbajul YANG, împreună cu o recomandare despre cum această transformare ar trebui făcută [80].

3.4 Alegerea unui cadru software pentru serverul NETCONF

Există numeroase soluții care lucrează cu protocolul NETCONF, atât pe partea de client, cât și pe cea de server. O listă a acestora este menţinută de către grupul de lucru NETCONF și poate fi găsită online [81]. Conţine și soluții software proprietare, dar și soluții cu sursă deschisă. Pentru implementarea simulatoarelor prezentate în această lucrare au fost considerate trei opţiuni de implementare a unui server NETCONF, cu sursă deschisă: *Netopeer*, *OpenYuma* și *NETCONF Test Tool* (unealtă oferită de proiectul ODL). O comparaţie între acestea va fi prezentată în continuare, justificând astfel alegerea de a folosi una dintre ele în simulatoare [82].

3.4.1 Netopeer

Netopeer este o soluție ce se bazează pe librăria *libnetconf*, oferind atât o implementare pentru server, cât și una pentru client. Această librărie este una cu sursă deschisă, implementată în limbajul C, ce oferă o implementare a protocolului NETCONF [83]. Este o soluție care poate fi personalizată, oferind numeroase posibilităţi pentru implementările de server și de client și suportă toate caracteristicile protocolului NETCONF.

Netopeer oferă câteva unelte, de exemplu pentru a facilita integrarea modelelor informaționale YANG în module ale serverului, denumită *netopeer-manager*, sau pentru a configura caracteristicile serverului NETCONF, *netopeer-configurator*. Orice model de date YANG poate fi adăugat ca un modul al serverului, însă acesta trebuie prelucrat înainte. Astfel, fişierul **.yang* este transformat de către această soluție în fişiere pe care serverul le poate recunoaşte, inclusiv un fişier **.c*, care conţine un schelet de cod C, reprezentând așa-numite funcții de apel invers (*callback functions*) ce pot fi implementate pentru ca serverul să ofere comportamentul dorit, în raport cu modelul YANG folosit. Apoi, codul C este compilat, rezultând o bibliotecă partajată (*shared library*) care poate fi utilizată de către codul de bază al serverului.

3.4.2 OpenYuma

OpenYuma este o soluție software care se bazează pe proiectul *Yuma*, care a devenit proprietar în anul 2011. Propune de asemenea implementări pentru server și client NETCONF, scrise în limbajul C, oferind chiar posibilitatea de a încorpora acest cod în dispozitive al căror software folosește tot limbajul C.

OpenYuma are o filosofie asemănătoare cu *Netopeer*, oferind unelte pentru transformarea modelelor YANG în cod C schelet, care să fie apoi implementat pentru ca serverul NETCONF să ofere facilităţile propuse. Unealta propusă de acest cadru software se numeşte *yangudmp* și transformă fişierele **.yang* în fişiere **.h* și **.c*, conţinând, la fel ca în cazul *Netopeer*, funcții de apel invers ce trebuie rescrise.

Codul C obţinut după transformarea modelelor YANG se compilează, rezultând tot o bibliotecă partajată care să poată fi folosită de către codul de cază al serverului. Aceasta poate fi încărcată în momentul inițializării serverului sau chiar în mod dinamic, în timp ce acesta rulează.

3.4.3 Netconf Test Tool

Netconf Test Tool este un cadru software oferit în cadrul proiectului OpenDaylight. Este o soluție simplă, care nu poate fi personalizată foarte mult, oferind doar o implementare Java pentru un server NETCONF. Aceasta este folosită de proiectul ODL pentru a-și testa interfața de tip *southbound* care implementează protocolul NETCONF.

Scopul acestei soluții este puțin diferit de al celorlalte, deoarece *Netconf Test Tool* nu își propune oferirea unui cadru software pentru un server NETCONF care apoi să poată fi integrat cu echipamentele de rețea, ci oferirea unei soluții simple și rapide care să încarce un model YANG specific, cu scopul de a-l testa. Cu toate acestea, acest software a fost considerat pentru comparaţie, deoarece și scopul simulatoarelor este de a testa modelele YANG și de a crea topologii specifice rețelelor de transport de date fără fir, expunând modelele informaționale descrise anterior.

3.4.4 Comparaţie între soluţiile care oferă server NETCONF

Soluţiile descrise anterior au fost evaluate atât prin compararea documentaţiei relevante pe care acestea o pun la dispoziţie, cât și prin experimente practice care consideră diferite scenarii. Primul criteriu care poate fi considerat este limbajul pe programare în care aceste cadre software sunt implementate: *Netopeer* și *OpenYuma* sunt scrise în limbajul C, pe când *Netconf Test Tool* este o implementare Java.

Un alt criteriu pentru evaluare constă în abilitatea serverului de a încărca în mod direct (dinamic sau în momentul inițializării) modele YANG. Această posibilitate este oferită doar de implementarea Java. Celelalte două soluții au nevoie de o fază premergătoare de procesare,transformând fişierele **.yang* în **.c*. *Netconf Test Tool* poate încărca modelul YANG doar în momentul inițializării serverului, dintr-un director anume, pe când celelalte cadre software pot încărca acest model, după ce a fost procesat, în mod dinamic.

Un alt subiect pentru comparaţie este dat de tipurile de locuri de stocare a datelor (*datastore*) propuse de protocolul NETCONF suportate de implementările serverelor. *Netopeer* și *OpenYuma* folosesc fişiere XML în care stochează informaţiile și suportă toate cele trei tipuri de *datastore* propuse de NETCONF: *startup*, *candidate* și *runnning*. Cealaltă soluție software, *Netconf Test Tool* oferă posibilitatea de a utiliza doar *running datastore* stocând valorile parametrilor în variabile de execuţie,

acestea pierzându-se în momentul în care serverul este oprit. O diferenţă importantă apare în acest context, între cele două soluții implementate în C, *OpenYuma* oferind o flexibilitate mai mare. În cazul *Netopeer*, atunci când serverul se iniţializează, încarcă valorile parametrilor din *startup datastore* și *running datastore* în memorie. Apoi, începe să analizeze valorile din *startup datastore*, comparându-le cu valorile corespunzătoare atributelor din *running datastore*. Dacă valorile nu sunt egale, sau valoarea din *running datastore* nu există (însemnând prima utilizare a serverului), atunci serverul copiază valoarea din *startup datastore* și apelează funcţia de apel invers asociată parametrului de configurare. Prin această abordare severul se asigură ca nu există inconsistenţe între *startup datastore* și *running datastore* și, mai mult, dacă acest server este conectat la un echipament real de rețea, prin apelarea funcţiei asociate parametrului, dispozitivul va fi configurat astfel încât valorile din server să reflecte valorile de pe echipament. *OpenYuma* are o abordare mai flexibilă, permiţând dezvoltatorilor să altereze *running datastore* în timpul inițializării modulului, fără a implica *startup datastore*.

O altă caracteristică importantă a serverelor care poate fi comparată este abilitatea de a genera notificări NETCONF. Toate cele trei soluții oferă notificări. În cazul *Netconf Test Tool* este ceva simplist, acestea fiind declanşate printr-o comandă NETCONF și trimise dintr-un fişier XML care le conţine. *OpenYuma* oferă câte o funcție cu apel invers pentru fiecare notificare pe care o găseşte în momentul procesării modelului YANG. *Netopeer* nu are această abilitate în mod implicit, însă suportul este oferit de *libnetconf*.

Din perspectiva funcţiilor cu apel invers generate în momentul procesării modelului YANG putem compara doar cele două soluții implementate în limbajul C, deoarece *Netconf Test Tool* nu oferă astfel de caracteristici. Soluția *Netopeer* oferă posibilitatea de a alege care dintre parametrii configurabili vor avea o funcție cu apel invers asociată, în timp ce *OpenYuma* va genera o astfel de funcție pentru fiecare din parametrii configurabili pe care îi găseşte în modelul YANG. Pentru informaţiile de stare *Netopeer* generează o singură funcție cu apel invers care este apelată pentru orice operaţie NETCONF de tip *get* care ajunge la server, iar *OpenYuma* generează câte o astfel de funcție pentru fiecare parametru de stare.

Din punctul de vedere al abilităţii de a configura portul pe care serverul ascultă, toate soluțiile oferă această posibilitate. *Netconf Test Tool* are nevoie de un port de plecare, incrementând apoi numărul portului pentru celelalte instanţe ale serverului. Și celelalte soluții oferă posibilitatea de a schimba portul pe care serverul ascultă. Din perspectiva rulării mai multor instanţe de server pe aceeaşi mașină, atât *Netconf Test Tool* cât și *OpenYuma* oferă această posibilitate, diferitele instanţe folosind porturi diferite. *Netopeer*, pe de altă parte, nu permite rularea mai multor instanţe pe aceeaşi mașină.

Un alt criteriu pentru compararea acestor cadre software este dat de suportul pentru mai multe fire de execuție, adică abilitatea serverului de a permite conectarea mai multor clienţi în același timp. *Netconf Test Tool* oferă această posibilitate. *OpenYuma* oferă de asemenea acest suport, cu menţiunea că accesul concurent la resursele comune trebuie rezolvat în funcțiile cu apel invers care vor fi implementate de către

Tabelul 3.1: Comparaţie a caracteristicilor oferite de cadrele software considerate.

Criteriile de comparație	**Soluții servere NETCONF**		
	Netopeer	***OpenYuma***	***Testtool***
Limbajul de programare	C	C	Java
Încărcarea modelelor YANG brute	Nu	Nu	Da
Încărcarea dinamică a modulelor în server	Da	Da	Nu
NETCONF *datastore*	toate	toate	*running*
Suport pentru notificări	da	da	da
Port configurabil	da	da	da
Mai multe instanțe de server	nu	da	da
Mai multe conexiuni în același timp	da	da	da
Capabilități pentru depanare	jurnalizare	jurnalizare	jurnalizare

Tabelul 3.2: Compararea practică a cadrelor software considerate

Scenariul experimentat	***Netopeer***	***OpenYuma***	***Testtool***
Procesarea **.yang* în **.c*	lnctool	yangdump	N/A
Reprezentarea *datastore* în cadrul serverului	fişier XML	fișier XML	variabile de execuție
Încărcarea datelor în faza de iniţializare a serverului	*startup*	flexibilă, orice se poate suprascrie	N/A
Implementarea notificărilor NETCONF	*libnetconf*	funcție cu apel invers oferită	fișier XML
Funcții pentru parametrii configurabili	câte una per atribut	câte una per atribut	N/A
Funcții pentru parametrii de stare	doar una, globală	câte una per atribut	N/A
Conexiuni concurente de la clienți	da	suportă, trebuie implementat	da

dezvoltatori, nefiind rezolvat de către cadrul software. În cazul *Netopeer*, acest acces concurent este rezolvat din faza de proiectare a serverului și este oferit de *libnetconf*.

Și capabilitățile de depanare oferite ar putea constitui un criteriu de comparaţie, însă toate soluţiile se bazează pe fişiere de tip jurnal, oferind mai multe niveluri de jurnalizare.

Comparaţia bazată pe lucrurile descrise în documentaţie este rezumată în Tabelul 3.1, în timp ce un sumar al comparaţiei bazată pe experimentare se găseşte în Tabelul 3.2.

3.5 Arhitectura demonstraţiilor de concept WT SDN

Așa cum a fost amintit anterior, proiectul *Wireless Transport* din grupul Open Transport Working Grup din cadrul ONF se ocupă cu cercetare legată de SDN în contextul rețelelor de transport de date fără fir. După cum este prezentat și în [84, 85], acest tip de rețele sunt o parte importantă atât din perspectiva rețelelor curente, cât și din cea a rețelelor viitorului, precum cele 5G.

După efectuarea unei demonstraţii de concept [46] care folosea protocolul OpenFlow pentru a arăta capabilitățile SDN în rețelele de transport de date fără fir, observând dificultăţile pe care le presupune lucrul cu OpenFlow în acest context, grupul a trecut la dezvoltarea unui model informațional care să abstractizeze dispozitivele ce fac parte din aceste rețele, model care a fost descris anterior.

Apoi, alte trei demonstraţii de concept au avut loc, descrise în [47, 48, 86], care au urmărit evoluția acestui model informațional. Chiar dacă în cel de-al doilea PoC s-a folosit un model de date mult simplificat, iar în cel de-al patrulea a fost folosită cea mai bună și matură variantă a sa, arhitectura acestor demonstraţii a fost similară.

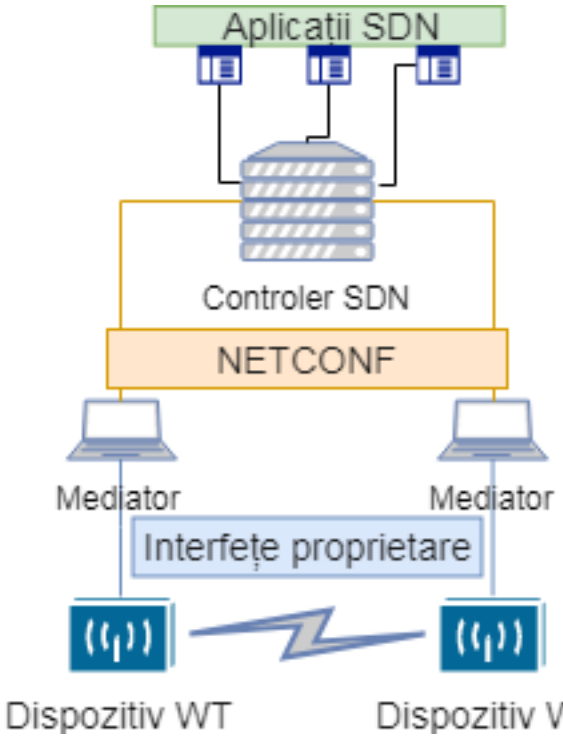

Fig. 3.6: Arhitectura demonstraţiilor de concept WT.

După cum se poate observa în Figura 3.6, în arhitectura acestor demonstraţii apare nevoia unui nivel intermediar, de mediator. Acest mediator este reprezentat de o aplicație software care expune o interfață NETCONF către nord, unde se conectează echipamentul de control SDN, folosind modelul informațional descris de TR-532. Rolul mediatorului este de a traduce comenzile NETCONF care vin de

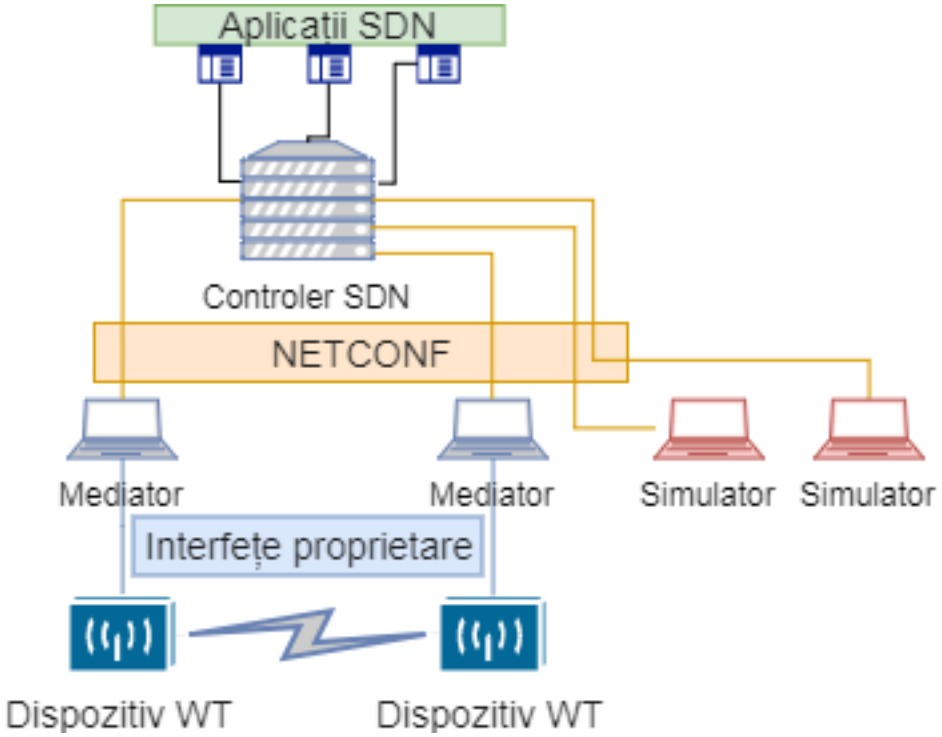

Fig. 3.7: Poziţionarea simulatoarelor în arhitectura demonstraţiilor de concept WT.

la controler în comenzi pe care dispozitivul de rețea le poate înţelege (de exemplu SNMP, CLI sau REST (Representational State Transfer)). Această nevoie apare din cauza faptului că echipamentele din rețelele actuale, care au fost folosite și pentru demonstraţiile de concept, nu suportă încă noul model informațional în mod nativ, deoarece abia a fost dezvoltat. Cel mai probabil, în viitor această nevoie va dispărea, echipamentele putând să încadreze *Microwave Model* în aplicaţia software proprie de control.

Poziţionarea simulatoarelor în arhitectura acestor demonstraţii de concept este ilustrată în Figura 3.7. Astfel, ele pot fi folosite de către dezvoltatorii de aplicații SDN pentru testarea aplicațiilor care utilizează interfaţa NETCONF ce implementează modelul informațional TR-532, eliminând nevoia acestora de a deţine dispozitive pentru transportul de date fără fir și mediatorul asociat acestora.

Capitolul 4

Mediatorul cu valori implicite (DVM) - 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 (DVM) - 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 LINC-WE. Integrarea cu *mininet*

Standardizare: ONF, etc..

Capitolul 6

Simulatorul reţelelor de transport de date fără fir (WTE)

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] Lei Zhou, Ligang Dong, and Rong Iin. Research on ForCES Configuration Management Based on NETCONF. *Information Technology Journal*, 13(5):904–911, 2014.

[11] ONF. TS-016, OpenFlow Management and Configuration Protocol. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf`, 2014. [Online].

[12] Chunyi Peng, Minkyong Kim, Zhe Zhang, and Hui Lei. VDN: Virtual machine image distribution network for cloud data centers. In *INFOCOM, 2012 Proceedings IEEE*, pages 181–189. IEEE, 2012.

[13] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan J Jackson, et al. Network Virtualization in Multi-tenant Datacenters. In *NSDI*, volume 14, pages 203–216, 2014.

[14] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. FlowVisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 1:132, 2009.

[15] Aditya Gudipati, Li Erran Li, and Sachin Katti. Radiovisor: A slicing plane for radio access networks. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 237–238. ACM, 2014.

[16] Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe, Guru Parulkar, Elio Salvadori, and Bill Snow. OpenVirteX: Make your virtual SDNs programmable. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 25–30. ACM, 2014.

[17] M Sune, A Köpsel, V Alvarez, and T Jungel. xDPd: eXtensible DataPath Daemon. *EWSDN, Berlin, Germany*, 2013.

[18] Andreas Blenk, Arsany Basta, and Wolfgang Kellerer. HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 397–405. IEEE, 2015.

[19] Advait Dixit, Fang Hao, Sarit Mukherjee, TV Lakshman, and Ramana Kompella. Towards an elastic distributed SDN controller. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 7–12. ACM, 2013.

[20] Dan Levin, Andreas Wundsam, Brandon Heller, Nikhil Handigol, and Anja Feldmann. Logically centralized?: state distribution trade-offs in software defined networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 1–6. ACM, 2012.

[21] Yury Jimenez, Cristina Cervello-Pastor, and Aurelio J Garcia. On the controller placement for designing a distributed SDN control layer. In *Networking Conference, 2014 IFIP*, pages 1–9. IEEE, 2014.

[22] 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.

[23] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *OSDI*, volume 10, pages 1–6, 2010.

[24] Amin Tootoonchian and Yashar Ganjali. HyperFlow: A distributed control plane for OpenFlow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pages 3–3, 2010.

[25] 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.

[26] 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.

[27] Fábio Botelho, Alysson Bessani, Fernando Ramos, and Paulo Ferreira. Smartlight: A practical fault-tolerant SDN controller. *arXiv preprint arXiv:1407.6062*, 2014.

[28] 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.

[29] Blaise Barney. POSIX threads programming. *National Laboratory. Disponível em:¡ https://computing. llnl. gov/tutorials/pthreads/¿ Acesso em*, 5, 2009.

[30] Andy Bierman, Martin Bjorklund, and Kent Watsen. RESTCONF protocol. 2017.

[31] Kok-Kiong Yap, Te-Yuan Huang, Ben Dodson, Monica S Lam, and Nick McKeown. Towards software-friendly networks. In *Proceedings of the first ACM asia-pacific workshop on Workshop on systems*, pages 49–54. ACM, 2010.

[32] Joshua Reich, Christopher Monsanto, Nate Foster, Jennifer Rexford, and David Walker. Modular SDN Programming with Pyretic. *Technical Reprot of USENIX*, 2013.

[33] Christopher Monsanto, Nate Foster, Rob Harrison, and David Walker. A compiler and run-time system for network programming languages. In *ACM SIGPLAN Notices*, volume 47, pages 217–230. ACM, 2012.

[34] Cole Schlesinger, Alec Story, Stephen Gutz, Nate Foster, and David Walker. Splendid isolation: Language-based security for Software-Defined Networks. In *Proc. of Workshop on Hot Topics in Software Defined Networking*, 2012.

[35] Daniel Turull, Markus Hidell, and Peter Sjödin. libNetVirt: the Network Virtualization Library. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5543–5547. IEEE, 2012.

[36] Andreas Voellmy, Hyojoon Kim, and Nick Feamster. Procera: a language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 43–48. ACM, 2012.

[37] Fabian Schneider, Takashi Egawa, Sibylle Schaller, Shin-ichiro Hayano, Marcus Schöller, and Frank Zdarsky. Standardizations of SDN and its practical implementation. *NEC Technical Journal, Special Issue on SDN and Its Impact on Advanced ICT Systems*, 8(2), 2014.

[38] Joel M Halpern. Standards collisions around SDN. *IEEE Communications Magazine*, 52(12):10–15, 2014.

[39] David Meyer. The Software-Defined-Networking Research Group. *IEEE Internet Computing*, 17(6):84–87, 2013.

[40] Attila Csoma, Balázs Sonkoly, Levente Csikor, Felicián Németh, Andràs Gulyas, Wouter Tavernier, and Sahel Sahhaf. Escape: Extensible service chain prototyping environment using mininet, click, netconf and pox. *ACM SIGCOMM Computer Communication Review*, 44(4):125–126, 2015.

[41] Antonio Felix, Nuno Borges, H Wu, Michael Hanlon, Martin Birk, and Alexander Tschersich. Multi-layer sdn on a commercial network control platform for packet optical networks. In *Optical Fiber Communications Conference and Exhibition (OFC), 2014*, pages 1–3. IEEE, 2014.

[42] Ying-Dar Lin, Dan Pitt, David Hausheer, Erica Johnson, and Yi-Bing Lin. Software-defined networking: Standardization for cloud computing's second wave. *Computer*, 47(11):19–21, 2014.

[43] Christian Esteve Rothenberg, Roy Chua, Josh Bailey, Martin Winter, Carlos NA Corrêa, Sidney C de Lucena, Marcos Rogério Salvador, and Thomas D Nadeau. When open source meets network control planes. *Computer*, 47(11):46–54, 2014.

[44] ONF. Technical Communities Overview. https://www.opennetworking.org/technical-communities, 2017. [Online].

[45] ONF. TR-539, OpenFlow Controller Benchmarking Methodologies, version 1.0. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-539_OpenFlow_Controller_Benchmarking_Methodologies_v1.pdf.

[46] ONF. Wireless Transport SDN Proof of Concept White Paper. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/ONF_Microwave_SDN_PoC_White_Paper%20v1.0.pdf, Sep 2015. [Online].

[47] ONF. Wireless Transport SDN Proof of Concept 2 Detailed Report. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Wireless_Transport_SDN_PoC_White_Paper.pdf, Jun 2016. [Online].

[48] ONF. Third Wireless Transport SDN Proof of Concept White Paper. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Third-Wireless-Transport-SDN-Proof-of-Concept-White-Paper.pdf, Dec 2016. [Online].

[49] Avri Doria, J Hadi Salim, Robert Haas, Horzmud Khosravi, Weiming Wang, Ligang Dong, Ram Gopal, and Joel Halpern. Forwarding and control element separation (ForCES) protocol specification. Technical report, 2010.

[50] Rodolfo Alvizu, Guido Maier, Navin Kukreja, Achille Pattavina, Roberto Morro, Alessandro Capello, and Carlo Cavazzoni. Comprehensive survey on T-SDN: Software-defined Networking for Transport Networks. *IEEE Communications Surveys & Tutorials*, 2017.

[51] Paul Goransson, Chuck Black, and Timothy Culver. *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2016.

[52] ONF. OpenFlow-Enabled Cloud Backbone Networks Create Global Provider Data Centers. https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-cloud-backbone-networks.pdf, Nov 2012. [Online].

[53] ONF. SDN Security Considerations in the Data Center. https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf, Oct 2013. [Online].

[54] Jiaqiang Liu, Yong Li, and Depeng Jin. SDN-based live VM migration across datacenters. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 583–584. ACM, 2014.

[55] Raul Muñoz, Ricard Vilalta, Ramon Casellas, Ricardo Martinez, Thomas Szyrkowiec, Achim Autenrieth, Víctor López, and Diego López. Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks. *Journal of Optical Communications and Networking*, 7(11):B62–B70, 2015.

[56] Google. Inter-Datacenter WAN with centralized TE using SDN and OpenFlow, White Paper. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/csgooglesdn.pdf`, 2012. [Online].

[57] Stefano Vissicchio, Laurent Vanbever, and Olivier Bonaventure. Opportunities and Research Challenges of Hybrid Software Defined Networks. *ACM SIGCOMM Computer Communication Review*, 44(2):70–75, 2014.

[58] David Ke Hong, Yadi Ma, Sujata Banerjee, and Z Morley Mao. Incremental deployment of SDN in hybrid enterprise and ISP networks. In *Proceedings of the Symposium on SDN Research*, page 1. ACM, 2016.

[59] Dan Levin, Marco Canini, Stefan Schmid, Anja Feldmann, et al. Toward transitional SDN deployment in Enterprise Networks. *Proceedings of the Open Networking Summit (ONS)*, 2013.

[60] Marco Canini, Anja Feldmann, Dan Levin, Fabian Schaffert, and Stefan Schmid. Panopticon: Incremental Deployment of Software-Defined Networking. In *ACM Symposium on SDN Research*, 2016.

[61] Cheng Jin, Cristian Lumezanu, Qiang Xu, Zhi-Li Zhang, and Guofei Jiang. Telekinesis: Controlling Legacy Switch Routing with OpenFlow in Hybrid Networks. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, page 20. ACM, 2015.

[62] Marcel Caria, Tamal Das, and Admela Jukan. Divide and conquer: Partitioning OSPF networks with SDN. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 467–474. IEEE, 2015.

[63] Marcel Caria and Admela Jukan. Link Capacity Planning for Fault Tolerant Operation in Hybrid SDN/OSPF Networks. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[64] NEC. Kanazawa University Hospital, Case Study. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/cs-nec.pdf`, 2012. [Online].

[65] Sarvesh Bidkar, Ashwin Gumaste, Puneet Ghodasara, Saurabh Hote, Anirudha Kushwaha, Geetha Patil, Shivprasad Sonnis, Rishav Ambasta, Braja Nayak, and Peeyush Agrawal. Field trial of a software defined network (SDN) using carrier ethernet and segment routing in a tier-1 provider. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 2166–2172. IEEE, 2014.

[66] György Kálmán. Applicability of Software Defined Networking in Industrial Ethernet. In *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, pages 340–343. IEEE, 2014.

[67] Masayoshi Kobayashi, Srini Seetharaman, Guru Parulkar, Guido Appenzeller, Joseph Little, Johan Van Reijendam, Paul Weissmann, and Nick McKeown. Maturing of OpenFlow and Software-defined Networking through deployments. *Computer Networks*, 61:151–175, 2014.

[68] ONF. TR-512, Core Information Model (CoreModel), version 1.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Core_Information_Model_V1.0.pdf`, Mar 2015. [Online].

[69] ONF. TR-512, Core Information Model (CoreModel), version 1.1. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/ONF-CIM_Core_Model_base_document_1.1.pdf`, Nov 2015. [Online].

[70] ONF. TR-512.1, Core Information Model (CoreModel), version 1.2. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-512_CIM_(CoreModel)_1.2.zip`, Sep 2016. [Online].

[71] ONF. TR-532, Microwave Information Model, version 1.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-532-Microwave-Information-Model-V1.pdf`, Dec 2016. [Online].

[72] IETF. NETCONF Configuration Protocol [RFC 4741]. `https://tools.ietf.org/html/rfc4741`, Dec 2006. [Online].

[73] IETF. Network Configuration Protocol (NETCONF) [RFC 6241]. `https://tools.ietf.org/html/rfc6241`, Jun 2011. [Online].

[74] James Yu and Imad Al Ajarmeh. An empirical study of the NETCONF protocol. In *Networking and Services (ICNS), 2010 Sixth International Conference on*, pages 253–258. IEEE, 2010.

[75] Khalid Elbadawi and James Yu. Improving network services configuration management. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[76] IETF. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) [RFC 6020]. `https://tools.ietf.org/html/rfc6020`, Oct 2010. [Online].

[77] Huang Ji, Bin Zhang, Guohui Li, Xuesong Gao, and Yan Li. Challenges to the new network management protocol: NETCONF. In *Education Technology and*

Computer Science, 2009. ETCS'09. First International Workshop on, volume 1, pages 832–836. IEEE, 2009.

[78] Emmanuel Nataf and Olivier Festor. End-to-end YANG-based configuration management. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 674–684. IEEE, 2010.

[79] Huiyang Cui, Bin Zhang, Guohui Li, Xuesong Gao, and Yan Li. Contrast analysis of NETCONF modeling languages: XML Schema, Relax NG and YANG. In *Communication Software and Networks, 2009. ICCSN'09. International Conference on*, pages 322–326. IEEE, 2009.

[80] ONF. TR-531, UML to YANG Mapping Guidelines, version 1.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-531_UML-YANG_Mapping_Guidelines_v1.0.pdf`, Sep 2016. [Online].

[81] NETCONF Working Group. NETCONF Implementations, Wiki. `https://trac.ietf.org/trac/netconf/wiki`, 2017. [Online].

[82] Alexandru Stancu, Alexandru Vulpe, George Suciu, and Eduard Popovici. Comparison between several open source Network Configuration protocol Server implementations. In *Communications (COMM), 2016 International Conference on*, pages 173–176. IEEE, 2016.

[83] Radek Krejci. Building NETCONF-enabled network management systems with libnetconf. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 756–759. IEEE, 2013.

[84] Dudu Bercovich, Luis M Contreras, Yoram Haddad, Ariel Adam, and Carlos J Bernardos. Software-Defined Wireless Transport Networks for Flexible Mobile Backhaul in 5G Systems. *Mobile Networks and Applications*, 20(6):793–801, 2015.

[85] Carlos J Bernardos, Antonio De La Oliva, Pablo Serrano, Albert Banchs, Luis M Contreras, Hao Jin, and Juan Carlos Zúñiga. An Architecture for Software Defined Wireless Networking. *IEEE wireless communications*, 21(3):52–61, 2014.

[86] ONF. Fourth Wireless Transport SDN Proof of Concept White Paper. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Third-Wireless-Transport-SDN-Proof-of-Concept-White-Paper.pdf`, Aug 2017. [Online].