Universitatea Politehnica, Bucuresti
Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Tema de curs **Retele de Calculatoare**

Software Defined Networking la nivel de retea

Studenti: BUSCA Tudor Stefan
TOADER Lucian Alexandru
DINICA Mihai Andrei

Grupa: 442A

Ianuarie,2016

CUPRINS

Capitolul 1: SDN – Software Defined Networking

I.Concept

SDN-ul (Software Defined Networking – Rețele definite prin sofware) este o abordare a rețelelor de calculatoare ce permite administratorilor de rețea să gestioneze serviciile de rețea prin abstractizarea funcționalității la nivel înalt.[1.1]

SDN-ul schimbă deja rețeaua de centre de date, dar în viitor ar putea redefini alte părți ale rețelei precum și profesia de inginer de rețea. [1.1]

Conceptul oferă flexibilitate operatorilor de rețea și centrelor de date în administrarea echipamentelor de rețea. Acest metodă este implementată cu ajutorul aplicațiilor software care rulează pe servere externe . Conceptul oferă customizări majore a operațiilor de rețea prin renunțarea la dispunerea tradiționala a stivei de rețea. Astfel se sporește accesabilitatea și implicit flexibilitatea . [1.1]

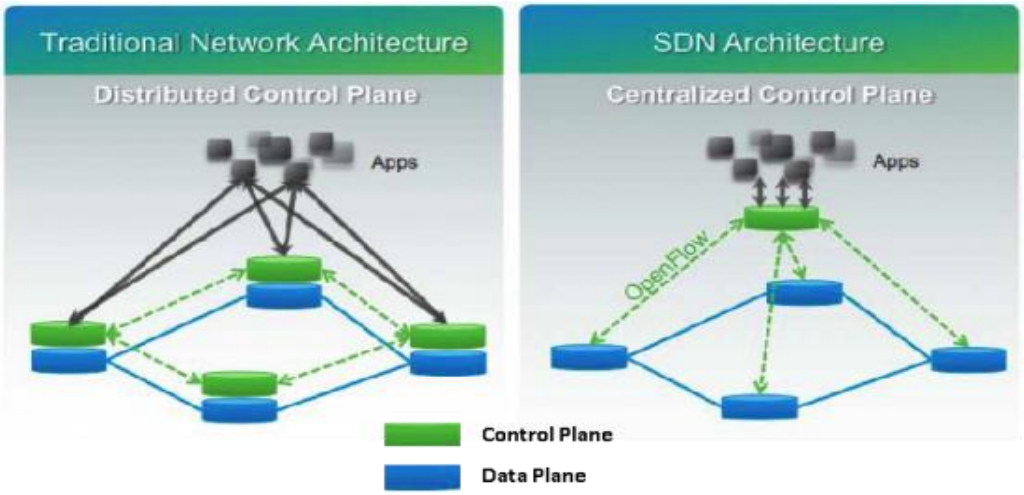

Figura 1 [1.1]

SDN-ul urmărește separarea nivelului de date de către nivelul de control , astfel întreaga rețea să poata fi administrată de către un controller. [1.1]

Software Defined Networking este o arhitectură în curs de dezvoltare dinamică, flexibilă , rentabilă și adaptabilă , făcând-o ideală pentru natura dinamică a aplicațiilor din ziua de azi. Această arhitectură decuplează controlul rețelei și funcțiile de expediere permițând blocului de control să devină direct programabil și infrastructurii de bază să fie abstractizată în concordanță cu aplicațiile și serviciile actuale. Principiul necesită o anumită metoda pentru ca planul de control să comunice cu planul de date. Un astfel de mecanism, OpenFlow, este un element fundamental pentru construirea de soluții SDN. [1.1]

Arhitectura SDN este :

- Direct programabilă - controlul rețelei nu mai depinde de direct de funcțiile de expediere;
- Agilă - sustragerea controlului din expediere permite administratorilor să ajusteze dinamic traficul de rețea pentru a satisface nevoia de schimbare;
- Gestionată la nivel central : Network Intelligence (Inteligența/Logica de rețea) este centralizată în controllerele SDN care mențin o imagine de ansamblu;
- Programabilă : SDN permite administratorilor de rețea să configureze, gestioneze, asigure și optimizeze resursele de rețea foarte repede prin intermediul programelor dinamice SDN;
- Bazată pe standarde deschise și furnizori neutrii: Când este implementat prin standarde deschise, SDN-ul simplifică proiectarea și operarea rețelei, deoarece instrucțiunile sunt furnizate de controllerele SDN în loc de protocoale specifice ale furnizorilor; [1.1]

Open Networking Foundation (ONF) este grupul cel mai implicat în crearea și standardizarea SDN. ONF este o organizație dedicată promovării și adoptării SDN.Aceasta dorește implementarea prin standarde deschise și crede că acestea sunt necesare pentru evoluția industriei de rețele. Ca o parte a eforturilor sale de a face SDN o realitate comercială care să raspundă la nevoile clienților, ONF dezvoltă standarde deschise

precum OpenFlow® Standard și OpenFlow® Configuration and Management Protocol Standard. [1.2]

OpenFlow® Standard este primul și singurul standard cu furnizor neutru ce se axează pe comunicațiile interfețelor dintra stratul de control și straturile de expediere din arhitectura SDN. Grupurile de lucru din cadrul SDN deschid noi orizonturi în dezvoltarea unor soluții interoperabile prin colaborarea cu experții mondiali cu privire la conceptele și standardele din cadrul SDN și OpenFlow. [1.2]

II.Istorie

Originile SDN-ului au început la scurt timp după ce Sun Microsystems a lansat Java în 1995. [1.3]

Una dintre primele și cele mai notabile proiecte SDN a fost Geoplex, un proiect dezvoltat de către gigantul AT&T. Membrii proiectului de la AT&T Labs Geoplex au echivalat interefețele de rețea și aspectele dinamice ale limbajului Java cu un mijloc de a implementa rețele intermediare. [1.3]
"Geoplex nu este un sistem de operare și nici nu încearcă să concureze cu unul. Este o rețea middleware (intermediară) care utilizează unul sau mai multe sisteme de operare ce rulează pe calculatoare conectate la Internet.Geoplex este o platformă de servicii care gestionează rețele și servicii online.Geoplex mapează toate activitățile IP ale rețelei într-unul sau mai multe servicii" au declarat Michah Lerner, George Vanecek, Nino Vidovic, și Dado Vrsalovic , membri ai proiectului . [1.3]

Un alt capitol important din istoria SDN îl reprezintă WebSprocket și implicit creatorul sau Mark Medovich. În 1998, Medovich a proiectat un nou sistem de operare pentru rețele și un model de rulare cu structură obiect-orientată care poate fi modificat de către un compilator sau class loader (încarcător de clase) în timp real. Cu această abordare , aplicații puteau fi scrise în Java folosind ca moșteniri clase din kernelul WebSprocket .Platforma WebSprocket a fost proiectată astfel încât dispozitivele să aibă capacitatea de a instanța stiva de rețea și protocoalele ca mai multe fire de execuție.Ulterior s-a dezvoltat și ideea de Supranet . Supranetul reprezintă fuziunea dintre lumea fizică și cea digitală , supranumit astăzi si Internet Of Things. [1.4]

În 2000, WebSprocket a fost selectată ca fiind una dintre cele mai de top tehnologii emergente din lume, ca în anul 2001 să dezvolte alături de Ericsson primul switch software comercial . [1.4]

În aprilie 2001 , a avut loc primul test al SDN-ului de către Universitatea Ohio și OARnet.Testul a fost un succes după cum spune și Pankaj Shah – Managing Director,OARnet :

" Am asistat la primul pas de succes în realizarea rețelelor interoperabile prin implementarea Serverului Supernet Transaction ! "[1.5]

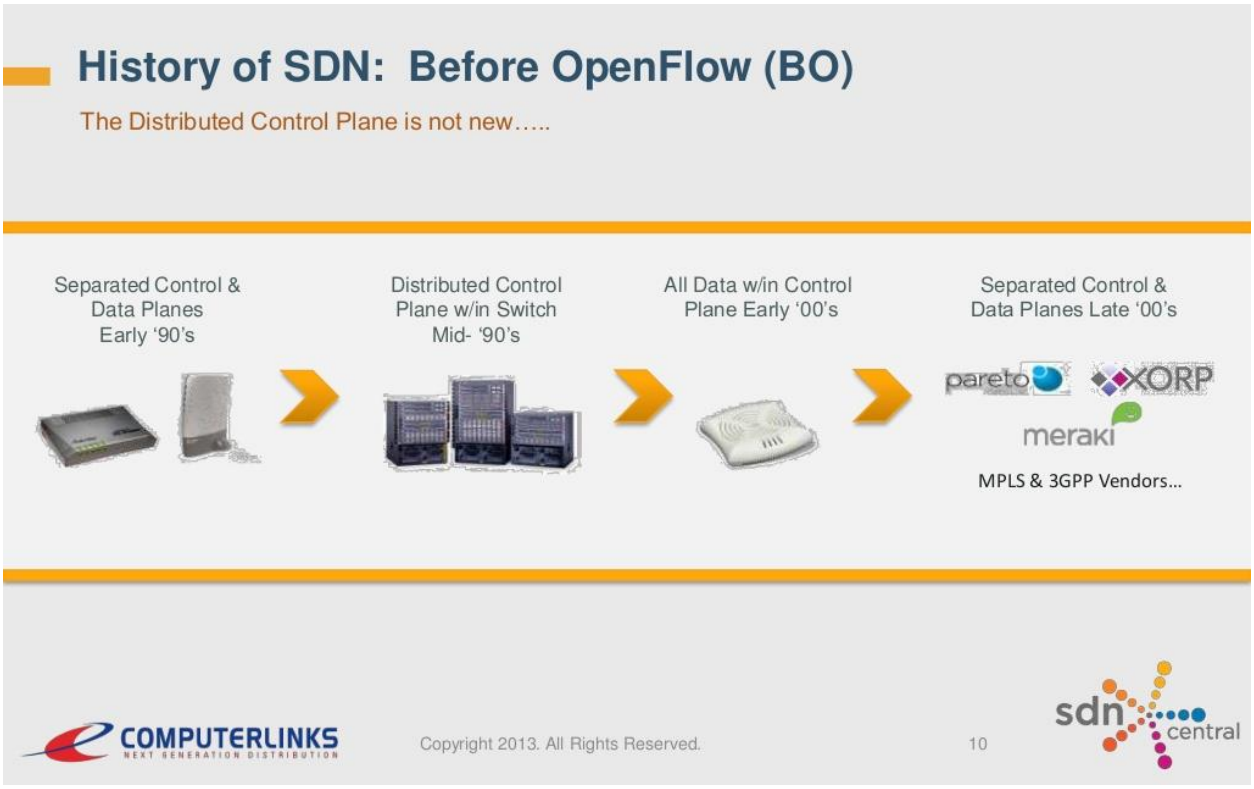

Figura 2 [1.5]

III.Arhitectură

Arhitectura unei rețele definite prin software este definită de către următoarele componente:

1. SDN Application (SDN App)
2. SDN Controller
3. SDN Datapath
4. SDN Interfaces :
    - Control to Data-Plane Interface (CDPI)
    - Northbound Interface (NBI)

Arhitectura SDN este organizată pe 3 niveluri : - aplicație;
- control;
- infrastructură;

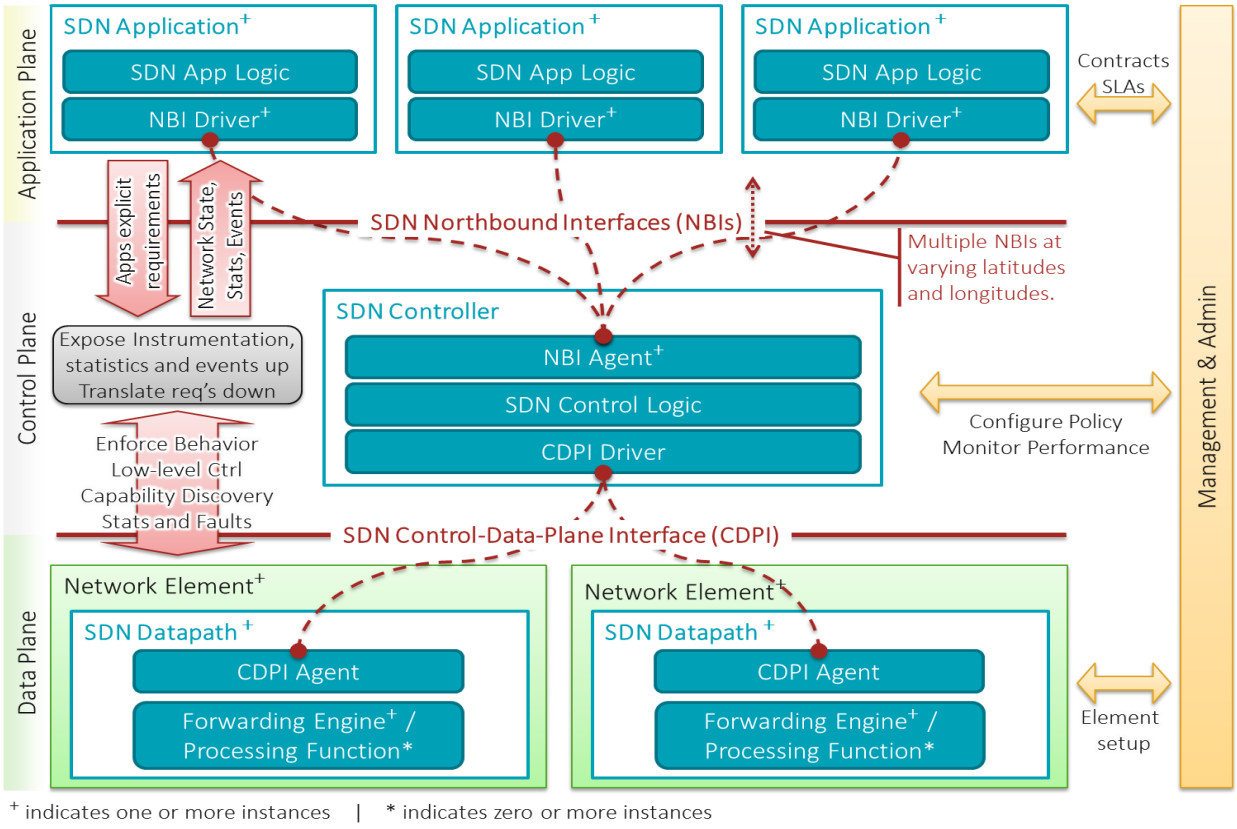

Figura 3 [1.5]

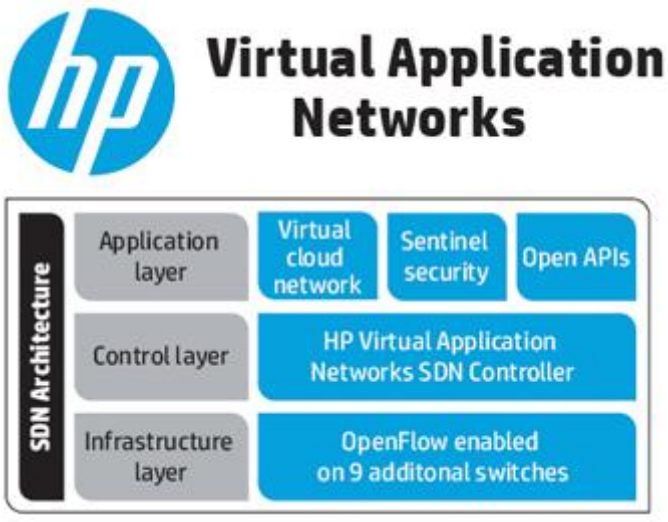

Figura 4. [1.5]

Nivelul de aplicație este constituit din totalitatea aplicațiilor care ajung la utilizatori.Exemplu: Aplicație pentru videoconferințe sau management-ul relațiilor cu clienții.[1.5]

Nivelul de control se referă la funcționalitatea ce permite aplicațiilor să execute eficient și în siguranță, precum firewall-uri și/sau protecția împotriva DDoS. [1.5]

Infrastructura este alcătuită din echipamentele în sine, aici regăsindu-se o gamă diferită de modalități de gestionare a acestora în funcție de necesitatea și scopul rețelei. [1.5]

1.*SDN Application* (SDN App)

SDN Applications sunt programe care în mod explicit și direct comunică cerințele lor de rețea și comportamentul de rețea dorit controllerului SDN prin intermediul NBI-Northbound interface. În plus, pot folosi o imagine de ansamblu a rețelei pentru scopurile lor decizionale interne. O aplicație SDN este alcătuită dintr-o parte logică (SDN Application Logic) și unul sau mai multe drivere SDN. Aplicațiile SDN pot expune singure un alt strat de control abstractizat, astfel oferind interfețe de nivel mai înalt agenților NBI.

2.*SDN Controller*

Controllerul SDN reprezintă "creierul" rețelei și este o entitate logică centralizată responsabilă de traducerea cerințelor care vin din stratul superior (SDN Application) pentru stratul inferior (SDN Datapath) și furnizează o vedere abstractă a rețelei ce conține statistici și evenimente pentru SDN App. [1.5]

Platforma unui controller SDN conține în mod obișnuit o colecție de module care pot efectua diferite sarcini de rețea , precum inventarierea dispozitivelor din rețea sau colectarea statisticilor. Se pot introduce extensii pentru a îmbunătății funcționalitatea și pentru a susține capabilități mai avansate precum rularea algoritmilor de analiză și orchestrarea unor noi reguli în rețea. Două dintre cele mai cunoscute protocoale folosite de controllerul SDN pentru a facilita comunicarea cu switch-uri sau routere

sunt OpenFlow și OVSBD. Aceste două protocoale se bazează pe ideea de centralizare a deciziilor de expediere. [1.5]

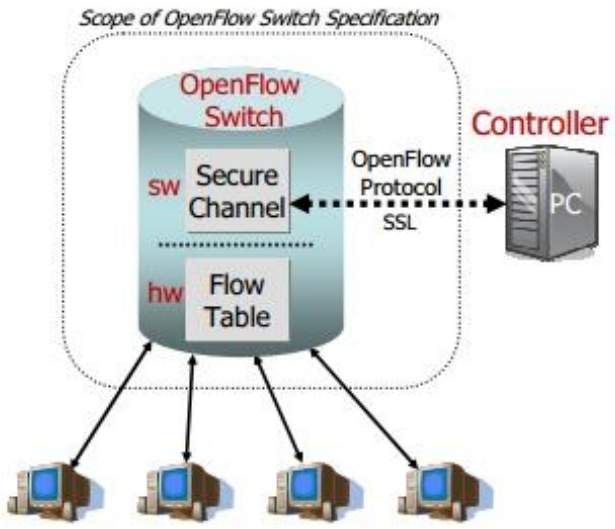

Figura 5. [1.5]

Tipuri de switch-uri:

-Switch pur SDN - toate funcțiile de control ale unui switch tradițional (protocoalele de routare sunt folosite în crearea bazei de expediere a informației) sunt rulate de un controller central , funcționalitatea switch-ului fiind în strânsă concordanță cu planul de date

-Switch-ul hibrid - este un switch pe care rulează, simultan, protocoalele de routare tradiționale și tehnologiile SDN). Diferența între un switch hibrid și unul pur SDN este că în cazul de față administratorul de rețea poate interveni de fiecare dată când este nevoie pentru a configura controllerul SDN cu scopul de a controla anumite fluxuri de trafic (protocoalele tradiționale rulând în background).

-Rețea hibridă – este rețeaua care conține ambele tipuri de switch-uri lucrând în strânsă concordanță. [1.5]

3.*SDN Datapath*

Este un dispozitiv logic de rețea care expune vizibilitate și control asupra expedierilor anunțate și a capacităților de procesare. Reprezentarea logică poate cuprinte o parte sau toate resursele substratului fizic. Un SDN Datapath este format dintr-un agent CDPI și un set de unul sau mai multe

motoare de control al traficului, precum și funcții de procesare a traficului.Unul sau mai multe SDN Datapath poate fi conținut într-un singur element fizic de rețea sau o combinație fizică de resurse de comunicare ce acționează ca o unitate. [1.5]

Un SDN Datapath poate fi definit asupra mai multor elemente fizice de rețea.Această definire logică nu prescrie sau include maparea logică-fizică,gestionarea resurselor partajate,interoperabilitatea cu o rețea non-SDN,nici funcționalitatea de prelucrare a datelor care pot include funcții L47 [1.5]

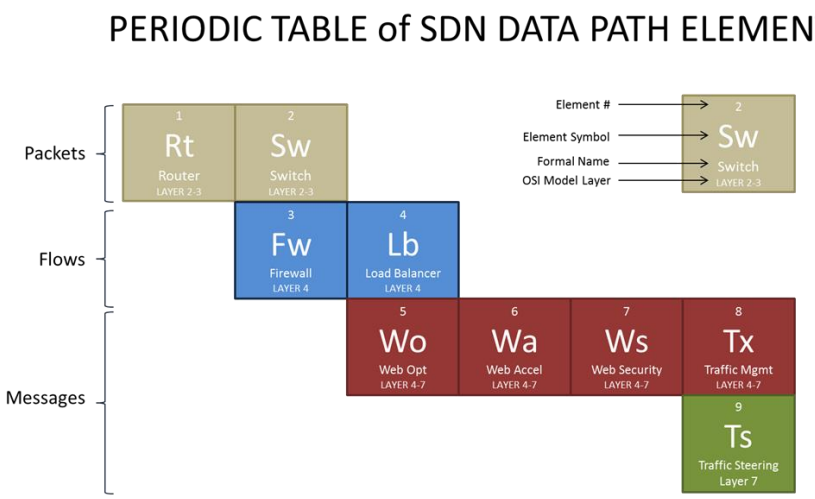

Figura 6[1.5]

4.*SDN Interfaces*

- SDN CDPI este interfața dintre un controler SDN și SDN Datapath, ce oferă controlul asupra tuturor operațiilor de expediere , operații de expediere , statistici de raportare și notificarea evenimentelor. Valoarea SDN-ului constă în speranța că CDPI-ul este implementat deschis, fără furnizori și interoperabil. [1.5]

- SDN NBI este o interfață dintre SDN App și un controler SDN și de obicei oferă o vedere abstractă a rețelei și permite exprimarea directă a comportamentului și a cerințelor de rețea. Aceasta se poate întâmpla la orice nivel de abstractizare (latitudine) și peste seturi diferite de funcționalitate (longitudine). [1.5]

Capitolul 2. OpenFlow

1) Descriere

Open Networking Foundation (ONF), o organizație dedicată promovării și adoptării SDN, definește OpenFlow ca prima interfață standard de comunicație între nivelurile de control și de expediere ale unei arhitecturi SDN. Acesta permite accesarea directă și manipularea planului de expediere al unui switch sau al unui router, atât fizic, cât și virtual. [2.1]

Într-o rețea convențională, switchurile trateaza atât rutarea de nivel înalt (calea de control) precum si expedierea pachetelor(calea de date). OpenFlow permite controlerelor de rețea să determine calea unor pachete în cadrul unei rețele de switchuri, controlerele fiind distincte de switchuri. Această separare a controlului de expediere permite un control al traficului mult mai sofisticat decât este fezabil folosind liste de control al accesului și protocoale de rutare. De asemenea, OpenFlow permite ca switchuri de la diferiți producători, care adeseori au interfețe si limbaje de scripting proprietare, să fie administrate de la distanță folosind un singur protocol deschis. [2.1]

OpenFlow este plasat peste TCP și prescrie utilizarea TLS (Transport Layer Security). Controlerele trebuie să asculte pe portul TCP 6653 după switchuri care vor să deschidă o conexiune.

2) Componente

Protocolul OpenFlow poate fi împărțit în patru componente:

- nivelul de mesaje
- automatul de stări
- interfața de sistem
- configurarea
- modelul de date

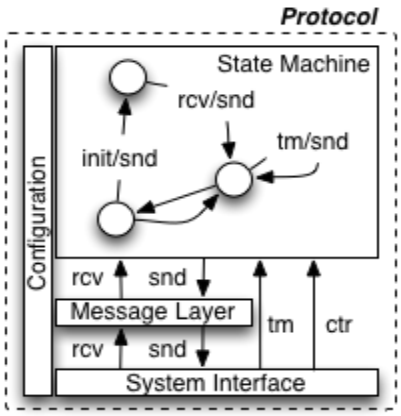

Figura 1. Protocolul OpenFlow [2.5]

2.1) Nivelul de mesaje

Acesta reprezintă nucleul stivei de protocol și definește structura validă și semantica pentru toate mesajele. Un nivel de mesaje tipic are funcțiile de construire, copiere, comparare, printare și manipulare a mesajelor. [2.5]

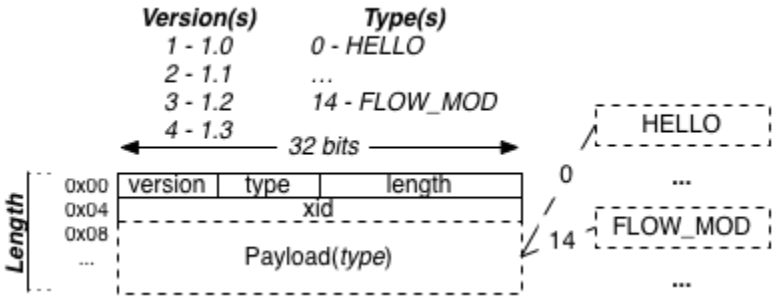

Figura 2. Structura unui mesaj [2.5]

Fiecare mesaj OpenFlow începe cu aceeași structură a headerului. Această structură fixă servește trei roluri, independente de versiunea de OpenFlow folosită:

- *câmpul de versiune* (indică versiunea de OpenFlow care a emis mesajul)
- *câmpul de lungime* (indică locul din fluxul de octeți la care se termină mesajul începând de la primul octet al headerului)

- *xid-ul sau identificatorul de tranzacție* (reprezintă o valoare unică, utilizată pentru a realiza corespondența între cereri și răspunsuri)

Câmpul de tip indică tipul de mesaj prezent și modul în care este interpretată încărcătura și este dependent de versiunea de OpenFlow folosită. [2.2]

2.2) Automatul de stări

OpenFlow are un model de automat de stări simplu. Aproape toate mesajele din cadrul acestui protocol sunt asincrone, fără a necesita o stare pentru a fi tratate. Totuși, procedura de stabilire a unei conexiuni implică negocierea de versiune și capabilități, ce trebuie făcută înainte ca oricare alte mesaje să fie transmise. De aceea, se izolează această procedură de stabilire a conexiunii în principal în automatul de stări. [2.4]
Automatele de stări sunt:

- controler, conexiune principală
- controler, conexiune auxiliară (începând cu versiunea 1.3)
- switch, conexiune principală
- switch, conexiune auxiliară (începând cu versiunea 1.3)

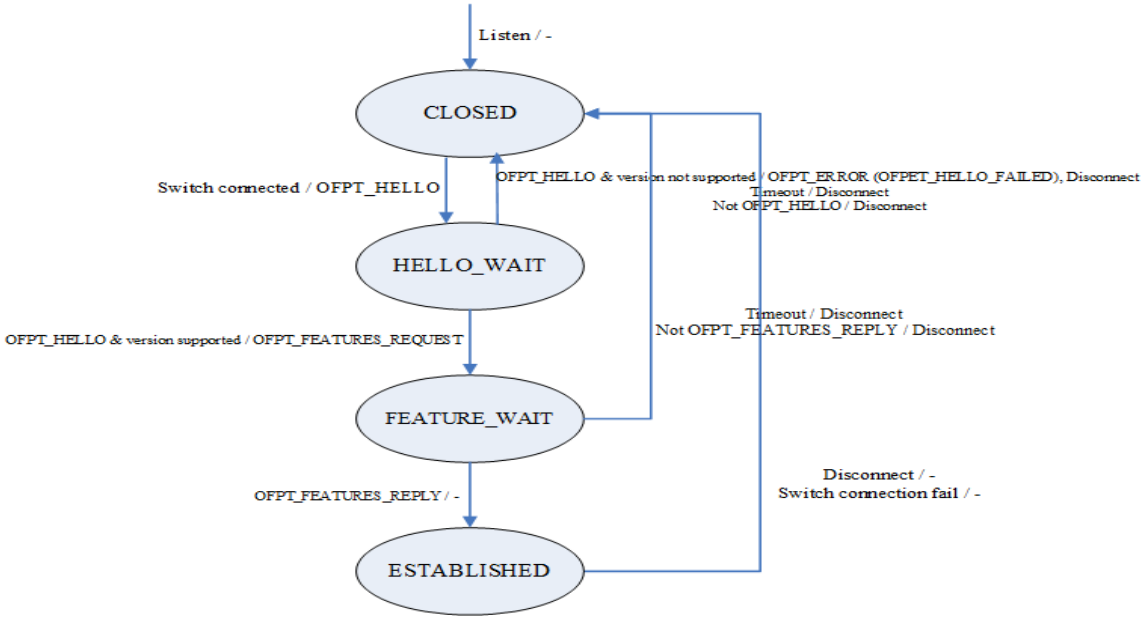

Figura 3. Controler, conexiune principală [2.5]

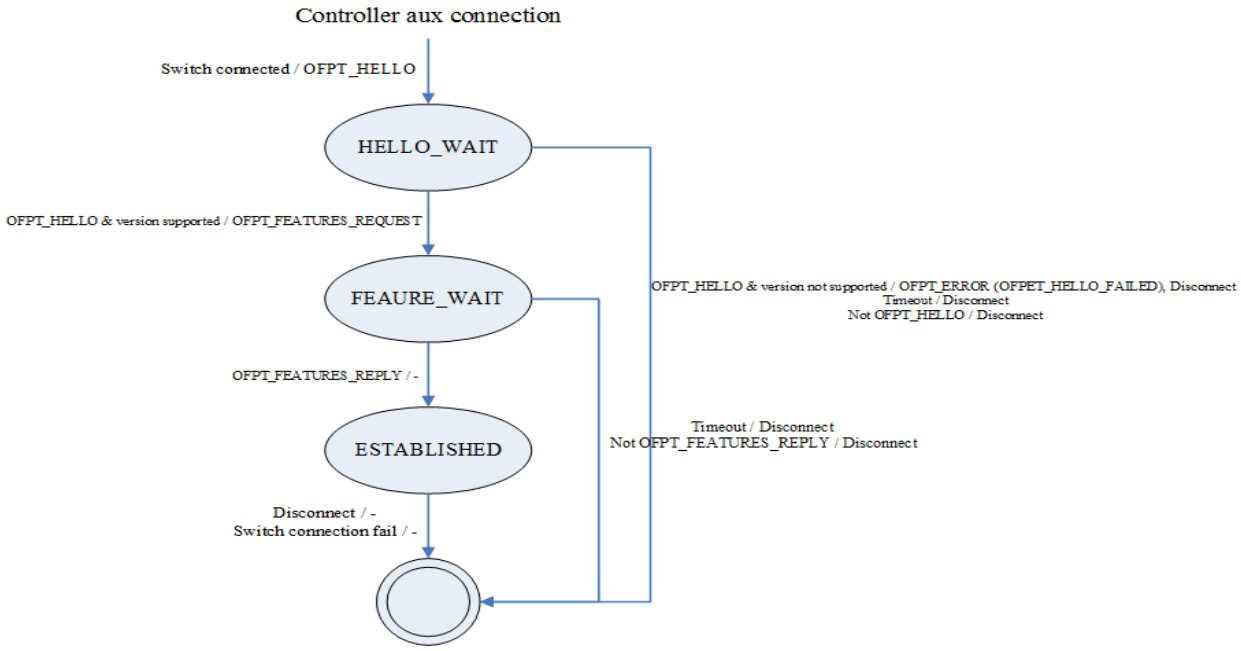

Figura 4. Controler, conexiune auxiliară [2.5]

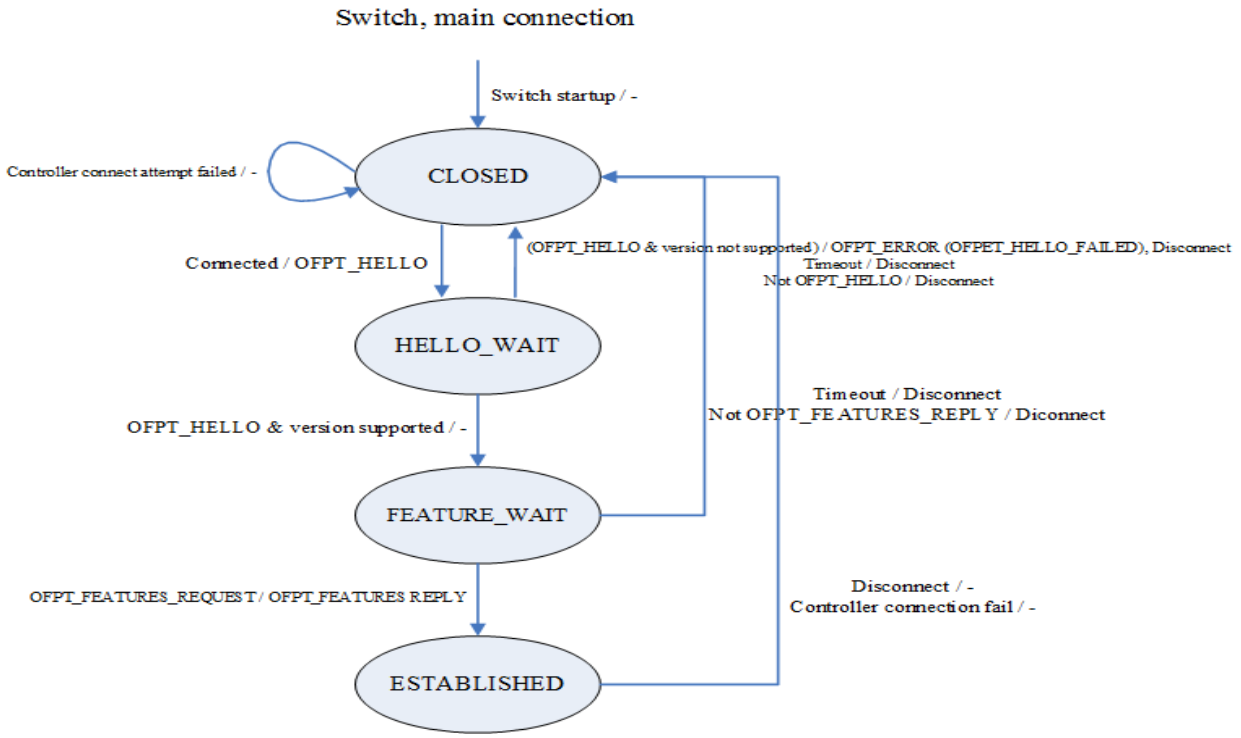

Figura 5. Switch, conexiune principală [2.5]

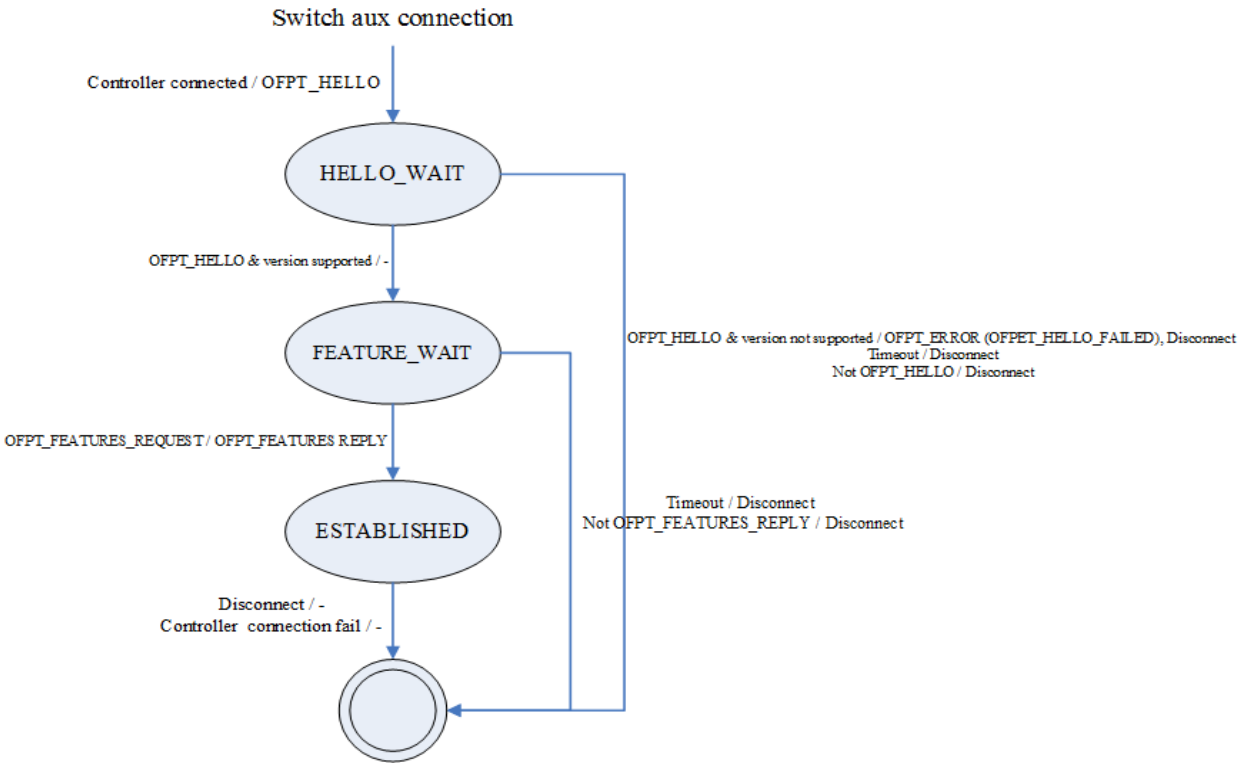

Figura 6. Switch, conexiune auxiliară [2.5]

Stabilirea conexiunii:

Secvența de stabilire a conexiunii între switch si controler constă din două faze:

1. *Negocierea versiunii*

Reprezintă procedura prin care aplicații diferite ajung la consens cu privire la aplicația care va fi utilizată. Până în momentul de față, există 5 versiuni lansate ale protocolului OpenFlow: 1.0, 1.1, 1.2, 1.3 și 1.4. Unele produse pot suporta numai un subset din cele 5 versiuni. Negocierea versiunii între switch si controler determină care dintre cele 5 versiuni ale protocolului OpenFlow va fi folosită pentru mesajele ulterioare. Este de asemenea posibil ca negocierea versiunii să determine faptul că versiunile dintre switch si controler nu sunt compatibile și nu mai este necesară comunicația. [2.3]

Procedura de negociere a versiunii începe imediat după ce conexiunea de nivel jos (TCP / TLS) este stabilită. Ambele capete ale conexiunii trimit un mesaj de tip “Hello” unul către celălalt. După recepționarea mesajului “Hello”, dispozitivul determină care versiune este cea negociată. Pentru

versiuni OpenFlow înainte de v1.3.1, versiunea negociată este cea mai veche versiune suportată atât de switch, cât și de controler. Pentru versiuni OpenFlow după v1.3.1, versiunea negociată este cea mai recentă versiune suportată atât de switch, cât și de controler. Negocierea se termină cu success dacă ambele capete suportă versiunea negociată și se termină cu eroare dacă oricare capăt nu suportă versiunea negociată sau oricare capăt nu recepționează mesajul “Hello”. [2.2]

Dacă negocierea se termină cu success, automatul de stări intră în următoarea fază, cea de descoperire a capabilităților. În caz contrar, conexiunea de nivel jos este încheiată și automatul de stări este resetat la starea inițială.

Sunt prezentate diagramele pentru toate situațiile posibile în cadrul unei negocieri:

- pentru o negociere normală:

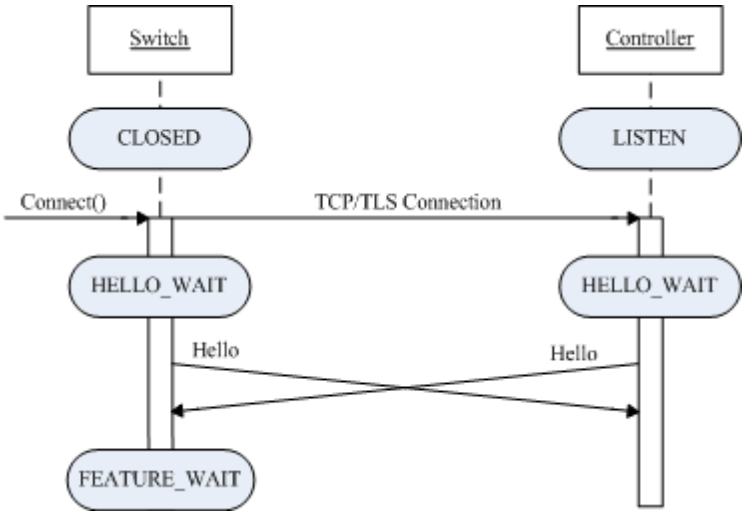

Figura 7. Negociere normală [2.5]

- pentru o negociere în care unul din capete nu suportă versiunea negociată:

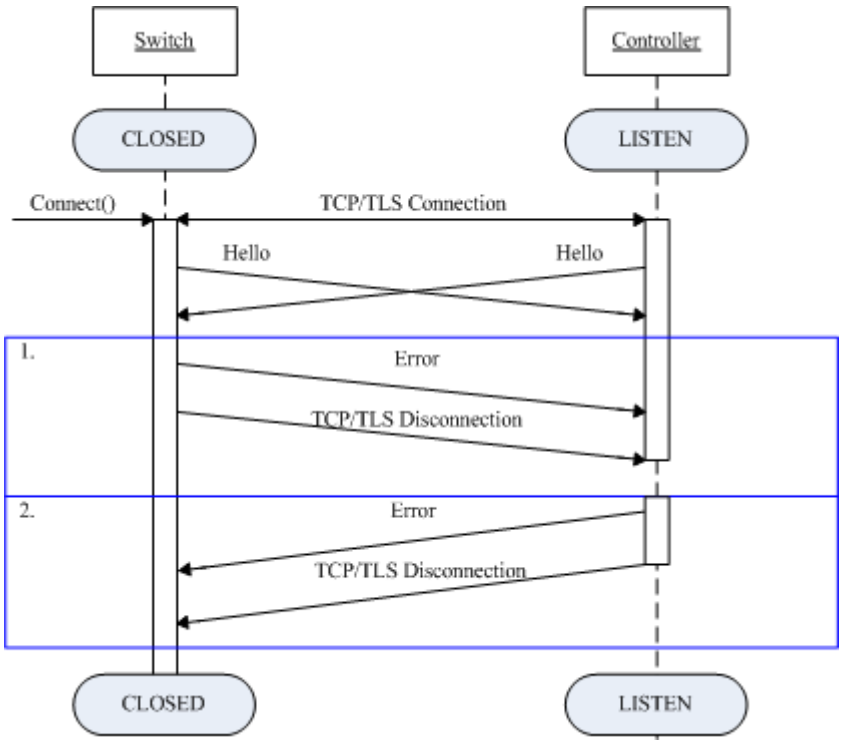

Figura 8. Negociere în care unul din capete nu suportă versiunea negociată [2.5]

- pentru o negociere în care unul dintre capete nu primește transmisia de la capătul corespondent:

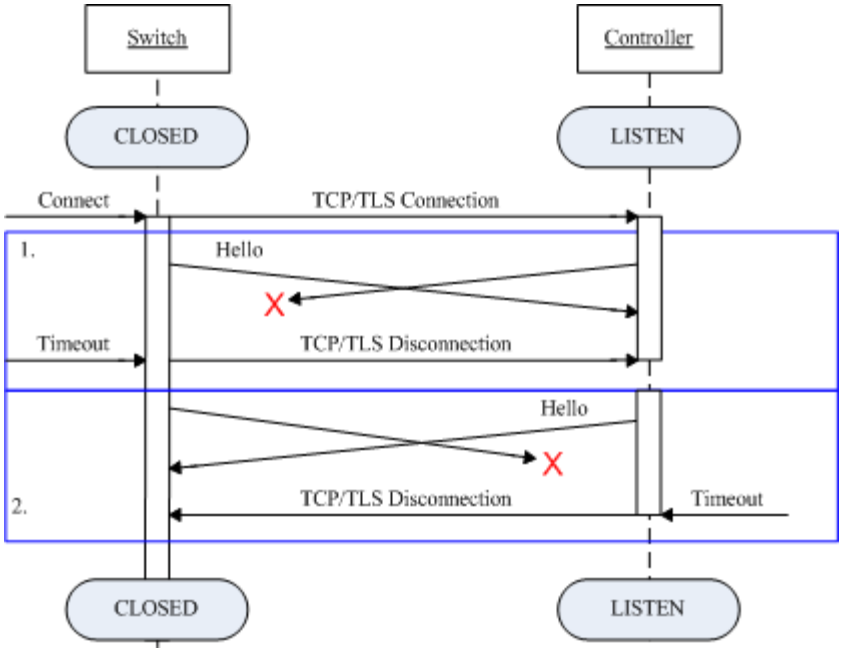

Figura 9. Negociere în care unul din capete nu primește transmisia de la capătul corespondent [2.5]

2. Descoperirea capabilităților

Scopul descoperirii capabilităților este acela de face cunoscute controlerului capabilitățile switchului. Este specifică tuturor versiunilor de OpenFlow și poartă numele de “Handshake”.
În procedura de decoperire a capabilităților, controlerul trimite un mesaj de tip Features Request către switch și primește un mesaj de tip Features Reply de la switch, care conține capabilitățile switchului. Dacă mesajul de tip Features Reply nu este recepționat de către controler, atunci acesta deconectează conexiunea de nivel jos de la switch. [2.1]
Sunt prezentate diagramele pentru situațiile posibile în cadrul descoperirii de capabilități:

-pentru o descoperire de capabilități normală:

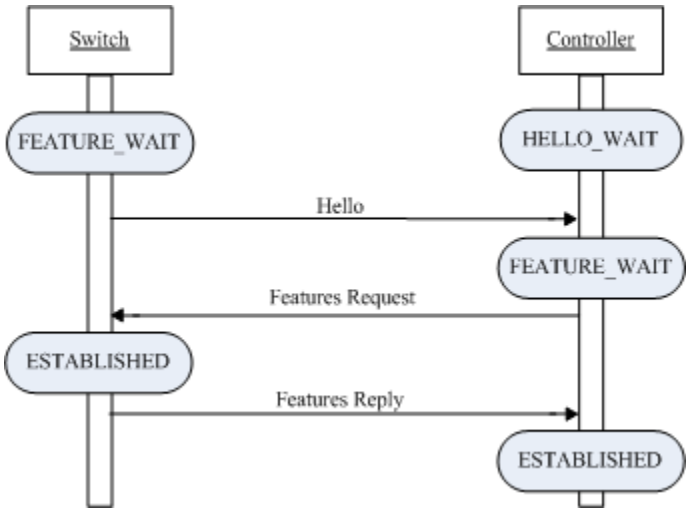

Figura 10. Descoperire de capabilități normală [2.5]

-pentru o descoperire de capabilități eșuată:

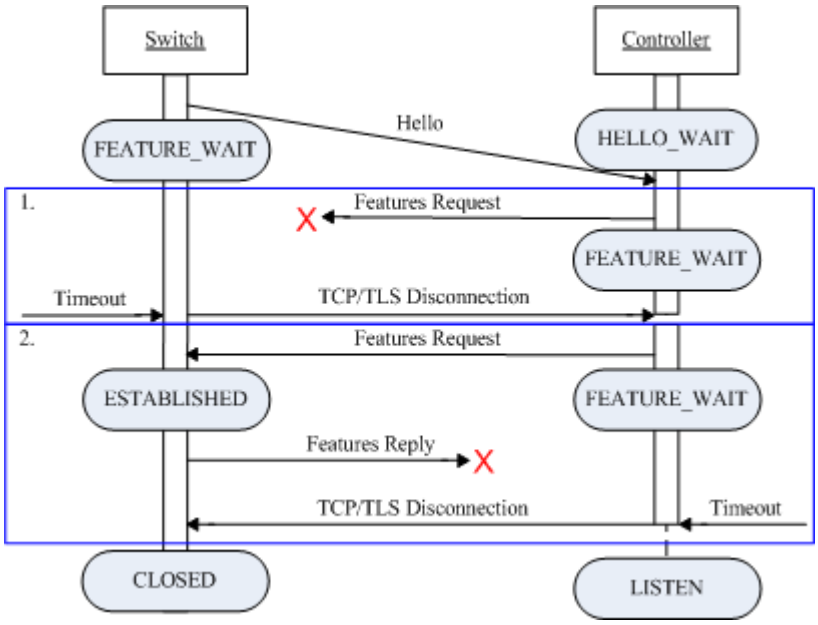

Figura 11. Descoperire de capabilități eșuată [2.5]

2.3) Interfața de sistem [2.4]

Reprezintă partea protocolului OpenFlow care se bazează pe și invocă alte componente din sistem. Există următoarele interfețe de sistem:

1. *Interfața TCP/TLS.* Această interacționează cu protocoalele de nivel jos (TCP sau TLV) din stiva protocolului și oferă transmisie fiabilă flux-orientată între switch și controler, care este folosită pentru transmisia oricărui mesaj OpenFlow între cele două.
2. *Interfața agentului de switch.* Aceasta interacționează cu kernelul de sistem al unui switch OpenFlow și expediază mesaje care au fost validate, de la controler, la kernelul switchului pentru procesările corespunzătoare. Acceptă de asemenea mesajele asincrone ale switchului și le expediază către stiva OpenFlow pentru procesare și transmisie către controler.
3. *Interfața aplicațiilor controler.* Aceasta interacționează cu aplicațiile controler care rulează peste stiva OpenFlow. Acceptă mesaje trimise către switch de la aplicația controler și le expediază către

stiva OpenFlow pentru procesare și transmisie. Expediază de asemenea mesaje recepționate de la stiva OpenFlow către aplicația controler.

4. *Interfața de configurare.* Aceasta permite operatorului sistemului să configureze stiva OpenFlow. Protocolul OpenFlow are un număr de parametri configurabili de către operatorul de sistem înainte sau în timpul utilizării. Aceștia sunt setați sau modificați prin interfața de configurare.

2.4) Configurarea

1. *Limbajul de configurare*

Acesta este proiectat pentru a avea o interfață simplă de configurare a controlerului și a switchurilor. Limbajul este implementat folosind un compilator care verifica validitatea sintaxei. Structurile principale din limbaj sunt legături între etape si identificatori. Tipurile din limbaj sunt predefinite, cu literali pentru IPv4, IPv6, întregi, timp, protocoale, stringuri și versiuni ale protocolului OpenFlow. Acești literali pot fi combinați în tipuri predefinite care formează etape. Aceste etape includ Realm, Authentification, Authorization, Initialization dar pentru un agent de switch, Realm este înlocuit de Connector. Aceste etape sunt proiectate pentru a rula în ordinea specificată de simbolul “->” din limbaj. Fiecare etapă descrie comportamentul pe care un controler trebuie să-l urmeze atunci când realizează sau acceptă noi conexiuni, sau descrie comportamentul conexiunii pentru un agent de switch. Avantajul utilizării unui compilator și a unui limbaj robust de configurare este acela că se poate verifica daca această configurare este validă și pot fi implementate capabilități avansate de configurare. [2.2]

2. *Utilitarul de configurare*

Limbajul de configurare este suportat de un utilitar care verifică gramatica și apoi generează comenzi care pot fi rulate pentru a configura controlerul și agentul de switch. Utilitarul rulează în 4 etape: tokenizarea, parsarea, elaborarea și evaluarea. Etapa de parsare verifică dacă inputul este

formatat corespunzător folosind o gramatică destinată pentru a fi simplu de scris și citit. Etapa de elaborare verifica dacă tot inputul corespunde tipurilor definite de limbaj, prevenind intrări eronate. După această etapă, garanții se pot face cu privire la tipurile de valori conținute de setări. În final, inputul este evaluat și transformat în comenzi ce pot fi rulate de controler sau de agentul de switch. [2.1]

2.5) Modelul de date [2.4]

Specificațiile OpenFlow descriu un pipeline abstract de procesare a pachetelor și o interfață pentru manipularea acestui pipeline. Totuși, majoritatea dezvoltatorilor de software sunt interesați doar de modelul de date. Acesta este un set relațional de structuri care descriu capabilitățile, starea de configurație și statisticile pentru fiecare abstracție OpenFlow. Un model de date independent de versiuni și împărțit în trei, este următorul:

1. *Capabilități.* Fiecare abstracție are un set de capabilități care descriu comportamente posibile. Cu versiuni ulterioare de OpenFlow, majoritatea capabilităților sunt opționale. Astfel, aplicațiile trebuie să interogheze un switch cu privire la capabilitățile sale, înainte de a face presupuneri cu privire la configurațiile posibile. Modelul de date prezintă capabilitățile rezultate ale unui switch specific.
2. *Configurație.* Fiecare abstracție are o stare de configurație care guvernează interacțiunea cu pachetele. Starea de configurație poate fi scrisă sau citită pentru a determina configurațiile existente sau pentru a actualiza o configurație ce rulează.
3. *Statistici.* Abstracțiile au uneori capacitatea de a înregistra statistici cu privire la comportamentul lor. Dacă aceste capabilități sunt suportate de switchul țintă, atunci statisticile asociate ale abstracției pot fi citite din modelul de date.

3) Beneficii [2.3]

Tehnologiile SDN bazate pe OpenFlow permit industriei IT să adreseze natura dinamică și de bandă largă a aplicațiilor moderne, adaptează rețelele către cerințele de business mereu schimbătoare și reduc semnificativ complexitatea operațiilor și administrării.

Printre beneficii sunt:

1. *Controlul centralizat al mediilor multi-producător*. Software-ul de control SDN poate controla orice rețea ce folosește OpenFlow, indiferent de producător, incluzând switchuri, routere și switchuri virtuale. În loc să administreze grupuri de dispozitive de la producători individuali, se pot utiliza arhitecturi și instrumente de management bazate pe SDN pentru a lansa, configura și actualiza cu rapiditate dispozitive în cadrul întregii rețele.
2. *Complexitate redusă prin automatizare*. Prin OpenFlow, se permite un framework flexibil de automatizare si administrare de rețea, ceea ce face posibilă dezvoltarea de instrumente care automatizează multe sarcini care sunt executate manual în prezent. Aceste instrumente de automatizare reduc overheadul operational, reduc instabilitatea rețelei introdusă de eroarea operatorului și suportă modelele emergente precum IT-as-a-Service.
3. *Rată mai mare de inovare*. OpenFlow accelerează inovația în business prin permiterea operatorilor de rețea să programeze rețeaua în timp real pentru a întruni diverse cerințe ale business-ului și ale utilizatorilor pe măsură ce acestea apar.
4. *Fiabilitate și securitate crescută*. SDN-ul permite industriei IT să definească configurații de nivel înalt ce sunt apoi traduse în cadrul infrastructurii prin OpenFlow. O arhitectură bazată pe OpenFlow elimină nevoia de a configura individual dispozitive din rețea de fiecare dată când sunt schimbări, ceea ce reduce probabilitatea de apariție a căderilor de rețea cauzate de inconsistențe în configurație.
5. *Un control mai granular al rețelei*. Modelul de control bazat pe flux al OpenFlow permite aplicarea de politici la un nivel foarte

granular, incluzând sesiunea, utilizatorul, dispozitivul și nivelurile aplicației, într-un mod puternic abstractizat și automatizat. Acest control permite operatorilor de cloud să suporte multi-tenancy în timp ce mențin izolarea traficului, securizarea, și administrarea resurselor în mod flexibil, atunci când clienții împart aceeași infrastructură.

6. *O experiență a utilizatorului îmbunătățită.* Prin centralizarea controlului rețelei și prin trimiterea informației de stare către aplicațiile de nivel înalt, o infrastructură bazată pe OpenFlow se poate adapta mai bine unor nevoi dinamice ale utilizatorilor. De exemplu, un operator ar putea introduce un serviciu video care oferă utilizatorilor săi premium cea mai bună rezoluție posibilă într-un mod transparent și automat. În prezent, utilizatorii trebuie să aleagă în mod explicit o setare de rezoluție, pe care rețeaua va fi sau nu capabilă să o suporte, ceea ce duce la întârzieri și întreruperi ce degradează experiența utilizatorului. Cu OpenFlow, aplicația video va putea să detecteze lărgimea de bandă disponibilă în timp real și ar putea ajusta automat rezoluția.

Capitolul 3. VXLAN(Virtual Extensible LAN)

Centrele de date au avut o crestere rapida a virtualizarii serverelor în ultimul deceniu, cu o creștere dramatică în agilitate. Virtualizarea rețelei este pasul următor evident - decuplarea rețelei virtuale din rețeaua fizică face mai ușoara gestiunea si automatizarea. [3.1]

O metodă comună folosita astăzi pentru a virtualiza rețelele de centrele de date este utilizarea de rețele de acoperire. O rețea de acoperire se află desupra retelei fizice, care permite utilizarea unei rețele virtuale de switch-uri, routere, firewall-uri, și așa mai departe. Aceasta decuplare a virtualului din nivelul fizic permite provizionarea programatica rapida de rețea pentru orice aplicație. [3.1]

Crearea unei rețele de suprapunere virtuală beneficiază, de asemenea, rețeaua fizica, care poate fi acum o rețea IP simpla care se ocupă numai de livrarea de pachete la destinații. O rețea de suprapunere adaugă simplitate, elasticitate și la scară de rețea fizică, un alt motiv prin care rețelele suprapuse câștigă popularitate. [3.1]

Figura 1 ilustrează o rețea de acoperire. Din perspectiva mașinii virtuale 1 și mașinii virtuale 2 (VM1 si VM2), traficul între ele este de a lua calea arătata de linia punctată, trecând prin dispozitivele tradiționale de rețea, cum ar fi switch-uri, routere, firewall-uri și, care sunt instanțiate în gazde. Cu toate acestea,traficul ia de fapt calea din partea de jos a figurii.[3.1]

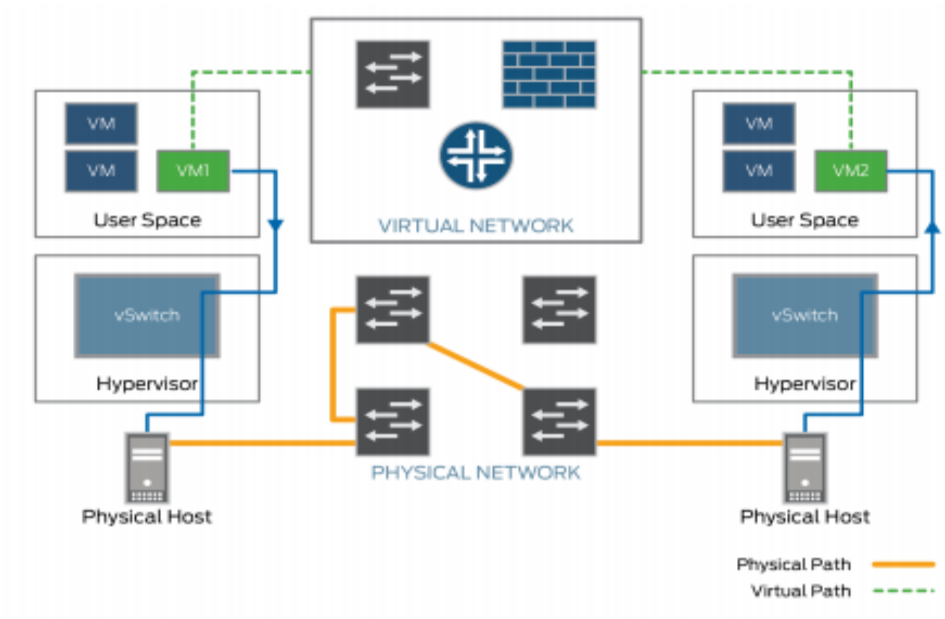

Figura 1. Ilustrarea unei retele de acoperire [3.1]

Un număr de protocoale de tunelare poate fi folosit pentru a crea suprapuneri în centrul de date - de exemplu, virtualizarea de rețea foloseste Generic Routing Encapsulation (NVGRE), Stateless Transport Tunneling (STT) si Virtual Extensible LAN (VXLAN). Deoarece VXLAN este cel mai frecvent utilizat, acest principiu se concentrează pe el. [3.2]

Protocolul VXLAN este documentat in Internet Engineering Task Force (IETF) RFC 7348. În plus față de sprijinirea virtualizarii rețelei centrului de date, protocolul VXLAN este, de asemenea, conceput pentru a răspunde nevoilor de date . [3.2]

Bazele VXLAN

Cel mai bun mod de a descrie VXLAN este că aceasta este o tehnologie de acoperire. VXLAN încapsulează cadre MAC ale Nivelului 2 in segmente UDP. Comunicarea se stabilește între două puncte ale capetelor de tunel numite Puncte finale ale tunelului virtual sau VTEPs. VTEPs încapsulează traficul mașinii virtuale într-un segment VXLAN și îl prezintă mașinii virtuale destinație cu pachetul originalul de Nivel 2. Acesta

poate fi de ajutor pentru a avea o privire asupra modului în care antetul de încapsulare este compus. [3.3]

VXLAN este un protocol de tunelare care încapsulează cadre de nivel 2 în pachete UDP de nivel 3, permițând crearea subretelelor virtualizate de nivel 2, sau segmente, cu deschidere catre nivelul 3 de retea. Înainte de examinarea pachetului încapsulat format, vom prezenta cele două concepte cheie VXLAN:[3.1]

§ Identificatorul de rețea VXLAN (VNI)

§ Punctele finale ale tunelului VXLAN (VTEP)

Identificatorul de rețea VXLAN

Fiecare subrețea sau segment , de nivel 2, este identificata în mod unic printr-un indentificator de rețea VXLAN (VNI). Ca și în cazul unui VLAN, masinile virtuale cu acelasi identificator pot comunica direct intre ele, în timp ce masinile virtuale cu identificatori diferiti au nevoie de un router pentru a comunica una cu cealălalta. [3.3]

VNI-ul a fost conceput pentru a răspunde nevoilor de creștere ale centrelor de date multi-chiriaș. Aceasta prevede:

- Creșterea scalei - Deși VNI îndeplinește o funcție similară cu ID-ul VLAN,VNI-ul are un avantaj foarte mare fata de ID-ul VLAN-ului: VNI-ul are o lungime de 24 biți, cu potential de a permite mai mult de 16 de milioane de segmente VXLAN. Id-ul VLAN de 12 biti prevede doar 4094 segmente utilizabile. Astfel, protocolul VXLAN poate sprijini segmentarea rețelei la scara cerută cu un număr mare de chiriași. [3.3]

- Creșterea ușurinței administrării - Într-o desfășurare VXLAN, o masina virtuala este unic identificata prin combinarea adresa MAC și VNI-ul acesteia. Două sau mai multe masini virtuale, poate avea , prin urmare ,aceeași adresă MAC, atâta timp cât acestea au diferite VNI-uri, care contribuie la simplificarea administrării rețelelor multi-chiriaș. [3.3]

Punctele finale ale tunelului virtual

Entitatea care efectuează încapsularea și decapsularea de pachete este reprezentata print-un punct final al tunelului virtual VXLAN (VTEP). Fiecare VTEP are două interfețe. Una dintre ele este o interfață de comutare cu care se confruntă masina virtuala gazdă și oferă comunicarea între masinile virtuale pe segmentul local de LAN. Cealalta este o interfață de nivel 3 de retea. [3.1]

Fiecare VTEP are o adresă IP unică care este utilizată pentru dirijarea pachetelor UDP între VTEP-uri. [3.1]

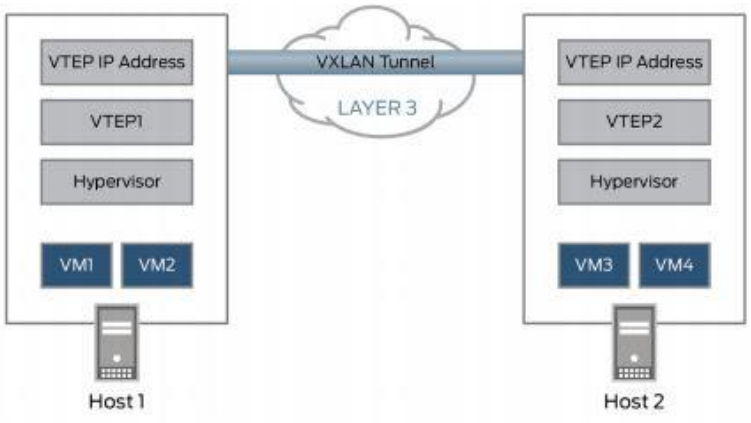

Figure 2 VTEPs in the Data Center

Figura 2 [3.1]

Așa cum se arată figura 2, atunci când VTEP1 primește un cadru Ethernet al VM1, adresat catre VM3, se foloseste VNI-ul și destinația MAC pentru a căuta în tabelul de expediere care este VTEP-ul pentru a putea trimite pachetul . VTEP1 adaugă apoi un antet VXLAN care conține un VNI al cadrului Internet, încapsulează cadrul intr-un pachet UDP de nivel 3, și ruteaza pachetul catre VTEP2 prin nivelul 3 de rețea . VTEP2 decapsuleaza cadrul Internet original și îl transmite catre VM3. [3.1]

Formatul pachetului VXLAN

Pachetul original de nivel 2 este încapsulat într-un antet VXLAN care include VNI asociate cu segmentele VXLAN care apartin mașinii virtuale. Pachetul rezultat este apoi înfășurat în pachetele UDP-> IP-> Ethernet pentru livrare finală cu privire la rețeaua de transport. Datorită acestei încapsulari VXLAN poate fi vazut ca un sistem de tunele cu gazdele ESX care compun punctele finale ale tunelului VXLAN Tunnel Endpoints (VXLAN Tunnel Endpoints =VTEP). VTEP sunt responsabile pentru încapsularea traficului mașinii virtuale in antetul VXLAN, precum și prezentarea mașinii virtuale destinație cu pachetul inițial L2.[3.1]

Figura 3 ilustrează formatul general al pachetului VXLAN.

Figure 3 VXLAN Packet Format

Figura 3. Formatul pachetului VXLAN [3.1]

Câmpurile cheie pentru pachetul VXLAN în fiecare dintre anteturile de protocol sunt[3.1]:

- ✓ *Antetul exterior MAC* - conține adresa MAC a VTEP sursă și adresa MAC a routerului următor . Fiecare router de-a lungul caii de pachete rescrie acest antet astfel încât adresa sursă este adresa MAC a router-ului și adresa destinație este adresa MAC a routerului urmator.
- ✓ *Antetul IP exterior* - Conține adresele IP ale VTEP-ului sursei și destinației.
- ✓ *Antetul UDP exterior* - Conține porturile UDP sursă și destinație:
- ✓ *Portul UDP Sursa* - In locul utilizarii acestui câmp pentru portul UDP sursă, protocolul il foloseste ca pe un identificator numeric pentru fluxul particular dintre VTEP-uri. Standardul VXLAN nu defineste

modul în care este derivat acest număr, dar, de obicei, VTEP-ul sursă calculează o combinație de câmpuri de nivel 2 si 3 sau antete de nivel 4.

- ✓ *Destinația portului UDP* - Portul UDP VXLAN. IANA alocă portul 4789 catre VXLAN.
- ✓ *Antetul VXLAN* - Contine 24 de biți VNI.
- ✓ *Cadrul original Internet* - Conține cadrul Internet original de nivel 2.

În total, încapsularea VXLAN adaugă între 50 și 54 de biti de informație la cadrul Internet original. Deoarece acest lucru poate duce la cadre Internet care depășesc implicit 1514 biti MTU, cele mai bune practici este de a pune în aplicare cadre colosale în întreaga rețea. [3.1]

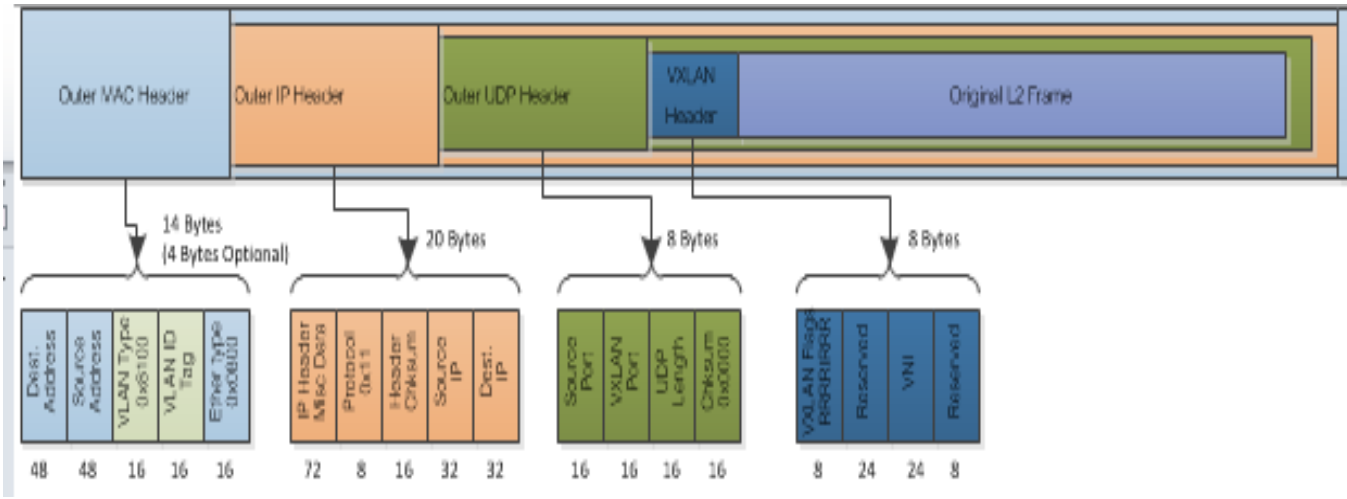

Figura 4. Formatul pachetului VXLAN [3.5]

În lumea reală , cu toate acestea , foarte puține centre de date au toate serverele lor virtualizate. Nevirtualizarea serverelor încă există - de exemplu , servere non- x86 (dispozitive UNIX și mainframe) , memoria (NAS ,iSCSI SAN) , și anumite baze de date și instante de calcul de înaltă performanță. Aceste dispozitive, de obicei, nu acceptă VXLAN și trebuie să continue să utilizeze segmente VLAN. [3.1]

Deci, cum pot dispozitivele nevirtualizate sa fie conectate la retea?

O modalitate este de a utiliza gateway-uri la marginea rețelei care acționează ca VTEPs , așa cum se arată în Figura 4. Gateway-ul VTEP mapeaza VLAN-uri la VXLAN-uri și se ocupa de încapsularea si decapsularea VXLAN astfel încât resursele nevirtualizate nu au nevoie sa suporte protocolul VXLAN. Acesta permite segmentelor VXLAN și VLAN să acționeze ca un domeniu de transmitere peste limita Nivelului 3. De exemplu , în ceea ce privește Serverul fizic 1 din figura 4, VM1 si VM2 se afla in acelasi segment VLAN . [3.4]

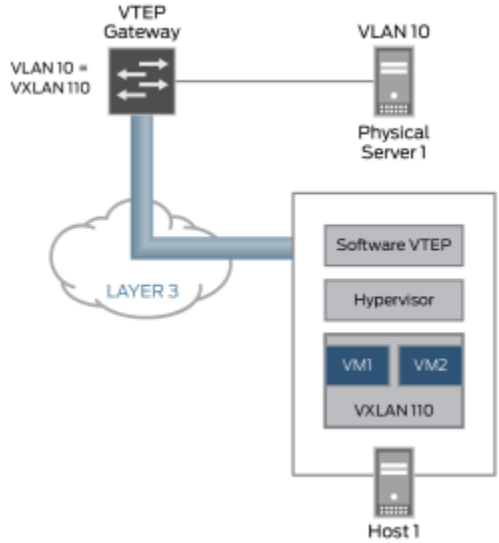

Figura 4. Serverul fizic 1 [3.1]

Un aparat software - de exemplu, un comutator instanță virtuală care rulează pe standardul hardware x86 - poate acționa ca un gateway VTEP. Pentru a satisface această cerință, sunt disponibile switch-urile care funcționează ca gateway-uri VTEP, de multe ori mentionate ca gateway-uri VTEP hardware. [3.4]

De exemplu, routere Juniper Networks MX Series, switch-uri QFX5100, și switch-uri EX9200, toate pot acționa ca gateway-uri VTEP. În plus, routere MX Series și switch-urile EX9200 pot ruta între VXLAN-uri diferite. [3.4]

Specificatiile actuale ale VXLAN nu includ un plan de control pentru a asigura un mecanism pentru VTEP, să distribuie ceea ce au descoperit despre adrese în reţea. Specificația, totuși, descrie un mecanism pentru fiecare STEP pentru a descoperi pe cont propriu ce adrese sunt în rețea prin intermediul monitorizarii pachetelor pe planul de date. Acest mecanism este similar cu modul în care switch-uri Ethernet afla adrese MAC: ori de câte ori un VTEP primește un pachet VXLAN, se înregistrează adresa IP a sursei VTEP, adresa MAC a VM, și VNI în tabelul de expediere VXLAN. În viitor, în cazul în care VTEP primește un cadru Internet să transmită acelei adrese MAC pe acel segment VNI, acesta stie sa incapsuleze acel cadru in pachetul VSLAN adresat acelui VTEP.[3.4]

Ce se întâmplă dacă VTEP primește un cadru Internet destinate unei VM pentru care nu se cunoaste adresa MAC?

Din nou, procesul VXLAN folosit pentru manipularea traficului necunoscut este similar cu modul în care se ocupa switch-urile cu traficul unicast necunoscut. Pentru a controla fluxul inutil, fiecare VNI este atribuit unui grup multicast. Ca un exemplu(Figura 5), să presupunem că VM1 din figura de mai jos vrea să trimită un pachet catre VM2 la adresa IP 192.168.0.11, dar nu are adresa MAC a lui VM2.

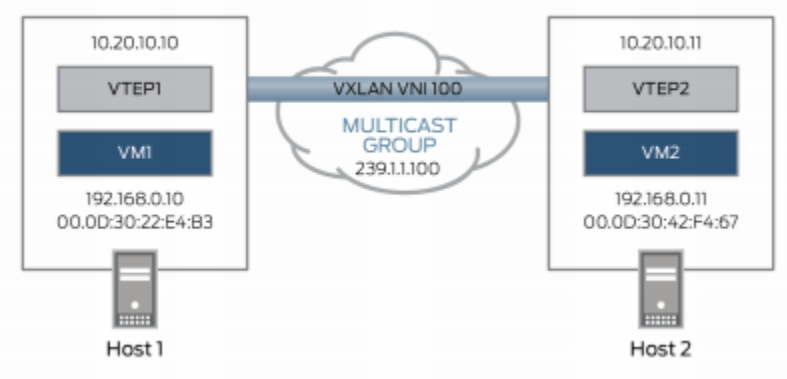

Figura 5. Exemplu grup multicast [3.1]

Procedeul de predare a acestui pachet unicast necunoscut este urmatorul:

1. VM1 trimite un pachet ARP care solicită adresa MAC asociată cu 192.168.0.11.

2. VTEP1 primește acest pachet ARP. Acesta încapsulează această cerere ARP într-un pachet multicast și abordează pachetul cu grupul multicast 239.1.1.100.

3. Toate VTEP-urile din grupul multicast 239.1.1.100 primesc pachetul. VNI-ul este decapsulat

și verificat în antetul VXLAN. În cazul în care VNI-ul pentru un segment local VXLAN este 100, VTEP-urile transmite pachetul inițial ARP pentru acel segment VXLAN.

În caz contrar, ele abandoneaza pachetul.

VTEP-urile adăuga, de asemenea maparea adresei IP a VTEP1 la adresa MAC a VM1 la tabelele VXLAN locale:

VNI	MAC Address	VTEP Address
100	00:0D:30:22:E4:B3	10.20.10.10

4. Când VM2 primește pachetul ARP din VTEP2, ea răspunde cu adresa lor MAC.

5. VTEP2 încapsulează răspunsul într-un pachet IP unicast și îl trimite catre VTEP1.

6. VTEP1 receptioneaza pachetul ARP, il decapsuleaza, și-l transfera catre VM1.

VTEP1 stochează acum maparea adresei IP VTEP2 și adresa MAC VM2 în tabelul de mapare VXLAN:

VNI	MAC Address	VTEP Address
100	00:0D:30:42:F4:67	10.20.10.11

În acest moment, toate adresele MAC relevante au fost învățate, și VM1 si VM2, în viitor, pot comunica direct prin intermediul unicast. [3.4]

De obicei, pentru a reduce fluxul inutil de pachete, administratorii atribuie fiecarui VNI propriul grup multicast. Cu toate acestea, nu există nicio obligație ca fiecare VNI sa aiba propriul grup multicast; se pot atribui multiple VNI-uri pentru un singur grup multicast. Izolarea segmentelor VXLAN este menținută în acest caz, deoarece VTEP verifică întotdeauna VNI-ul înainte de a trimite pachetele decapsulate segmentului VXLAN.[3.1]

Cerinte fizice ale retelei

Într-o implementare a acoperirii VXLAN, cerința de bază care sta la baza rețelei fizice este ca aceasta să fie dirijata, Layer 3 de rețea. Construirea unui centru de date de retea pe partea de sus a unui Layer 3 de rețea are urmatoarele avantaje:

- ✓ trebuie să susțină un singur protocol de rutare, făcând rețeaua mai stabila.
- ✓ nu trebuie să se bazeze pe STP să conveargă topologia – in schimb, protocolul de rutare trebuie sa se ocupe de asta. [4]

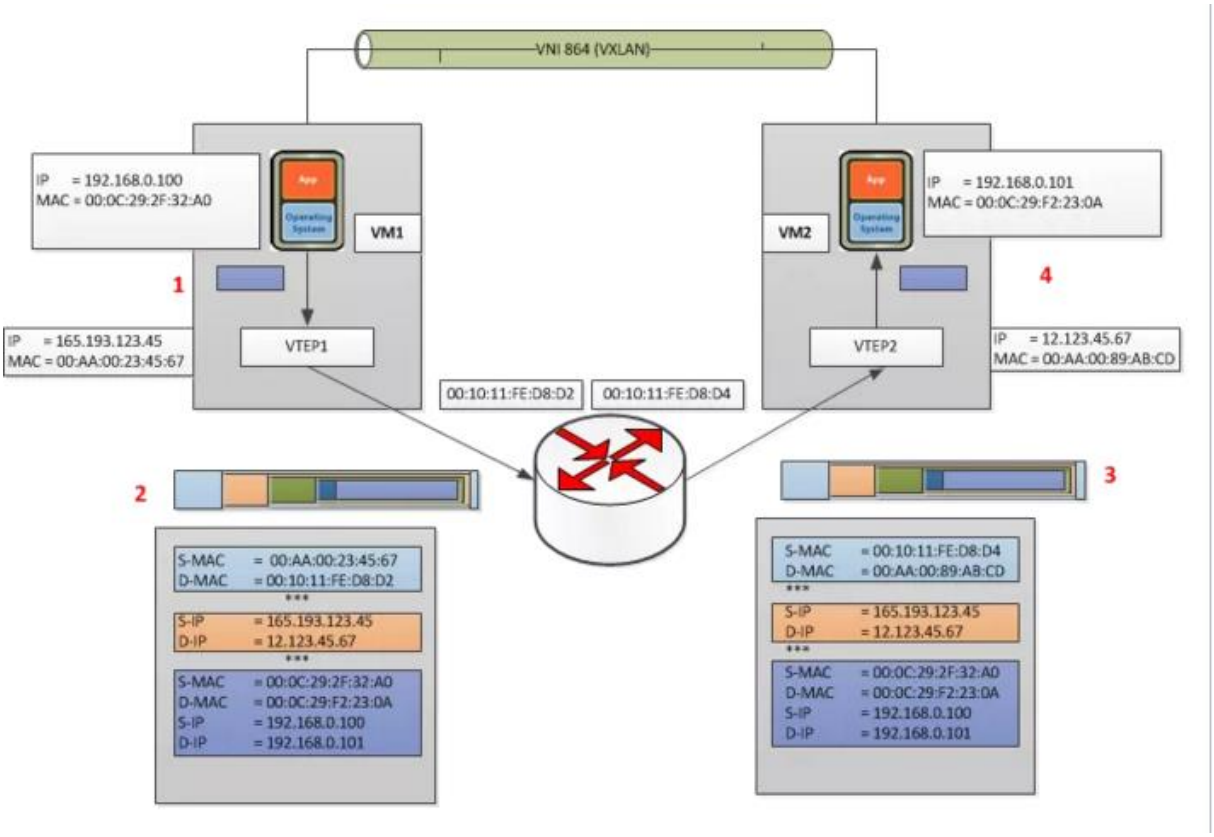

Figura 6. Comunicatia Masina Virtuala- Masina Virtuala[3.6]

Când VM1 vrea să trimită un pachet catre VM2, are nevoie de adresa MAC a VM2. Acesta este procesul care trebuie urmat[3.4]:

VM1 trimite un pachet ARP care solicită adresa MAC asociată cu 192.168.0.101

Acest ARP este încapsulat prin VTEP1 într-un pachet multicast la gruparea multicast asociată cu VNI 864

Toate VTEPs vad pachetul multicast si adauga asocierea VTEP1 și VM1 la tabelele sale VXLAN

VTEP2 primește pachetul multicast decapsulat și trimite broadcast-ul original pe porturile asociate cu VNI 864

VM2 vede pachetul ARP și răspunde cu adresa MAC

VTEP2 încapsulează răspunsul ca un pachet IP unicast și il trimite înapoi la VTEP1 folosind rutarea IP

VTEP1 decapsuleaza pachetul și il transfera catre VM1.

In acest moment VM1 cunoaște adresa MAC a VM2 si poate trimite pachete catre el așa cum se arată în Figura de mai sus:

VM1 trimite pachetul IP la VM2 de la adresa IP 192.168.0.100 la 192.168.0.101

VTEP1 ia pachetul și-l încapsulează prin adăugarea următoarelor antete:

Antetul VXLAN cu VNI = 864

Antet standard de UDP și stabilește suma de control UDP la 0x0000, și portul de destinație fiind portul desemnat VXLAN IANA. Cisco N1KV este in prezent folosind portul ID x8472.

Antetul IP standard cu destinația fiind adresa Ip a VTEP2 și Protocolul 0x011 pentru pachetul UDP utilizat pentru livrare.

Antetul standard MAC cu adresa MAC următoare. În acest caz, este interfața routeruluiscu adresa MAC 00: 10: 11: FE: D8: D2, care va utiliza rutarea IP pentru a trimite-l la destinație.

VTEP2 primeste pachetul deoarece are adresa lui MAC ca destinație. Pachetul este decapsulat și cauta un pachet VXLAN din cauza portului destinație UDP. În acest moment, VTEP va căuta grupurile de porturi aferente pentru VNI 864 găsite în antetul VXLAN. Se va verifica apoi că obiectivul, VM2 în acest caz, este permis să accepte cadre pentru VNI 864 din cauza că este membru al grupului și trece pachetul daca a trecut de verificare[3.4].

VM2 primește pachetul și se ocupă cu el ca orice alt pachet IP. [3.4]

Calea de întoarcere pentru pachete de la VM2 la VM1 ar urma aceeași rută IP prin router pe drumul de intoarcere.[3.6]

Compatibilitatea cu rețelele tradiționale

În timp ce coexistența unei rețele virtuale pe bază de VXLAN cu rețelele tradiționale este asigurata din cauza înființării sale privind standardele Internet existente, compatibilitatea cu rețele tradiționale necesită capacități suplimentare. Acest lucru este valabil mai ales atunci când gazdele dintr-o rețea virtuală VXLAN trebuie să comunice cu gazdele dintr-o rețea non-VXLAN, cum ar fi unul bazat pe VLAN. [3.7]

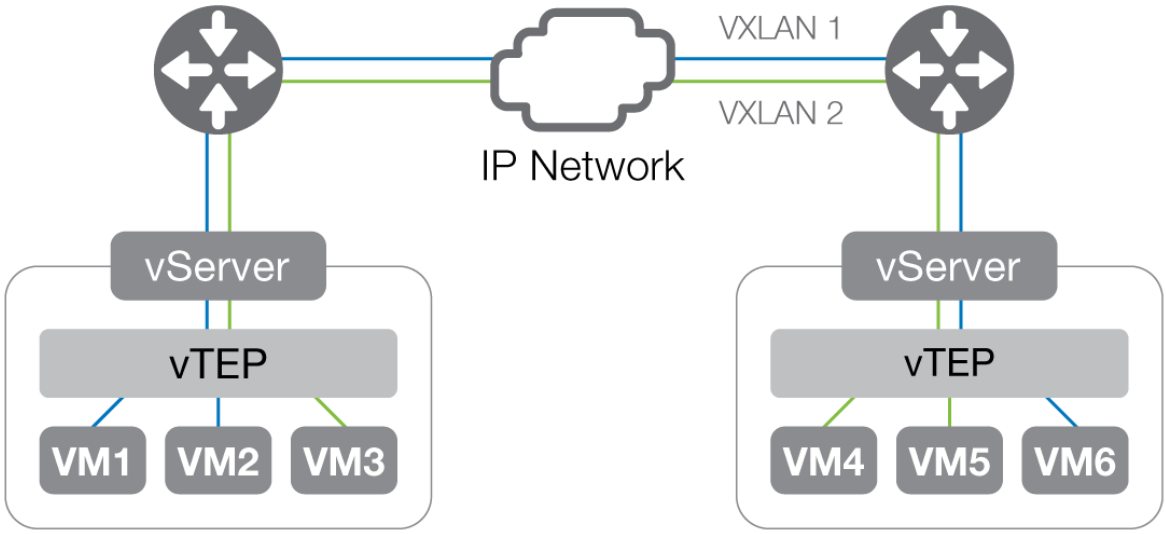

Figura 7. VXLAN [3.7]

În arhitectura de rețea virtuală, VXLAN, încapsularea se face pe gazda dintre rețeaua virtuală a mașinii virtuale(NIC) și portul său logic pe switch-ul virtual asociat. Acest punct final este transmis ca un punct final al tunelului virtual (vTEP). Tunelurile oferă izolare de trafic și vTEP oferă o măsură de protecție, permițând numai trafic cu ID-ul corespunzător segmentului VXLAN pentru a traversa rețeaua logică.

În timp ce acest model poate coexista cu o rețea IP traditionala, aceasta nu prevede compatibilitate cu gazdele dintr-un segment tradițional

de rețea IP. Compatibilitatea necesită un gateway VXLAN pentru a comunica segmentele VXLAN și VLAN și permite traficul pentru a traversa ambele rețele.

Exemplu(Configurarea VXLAN pe routere din seria MX)

În acest exemplu, VLAN este configurat pentru a rula pe un domeniu implicit. Interfețele VTEP sunt configurate la adresa loopback, și grupuri VLAN sunt configurate sub domenii cu VXLAN activat. Interfețe sunt configurate pentru VLAN sîncapsulare, și IRB este activat. Protocoalele OSPF și PIM sunt configurate pentru a facilita rutarea unicast și multicast. Sasiul este configurat pentru GRES și servicii IP îmbunătățite. [3.8]

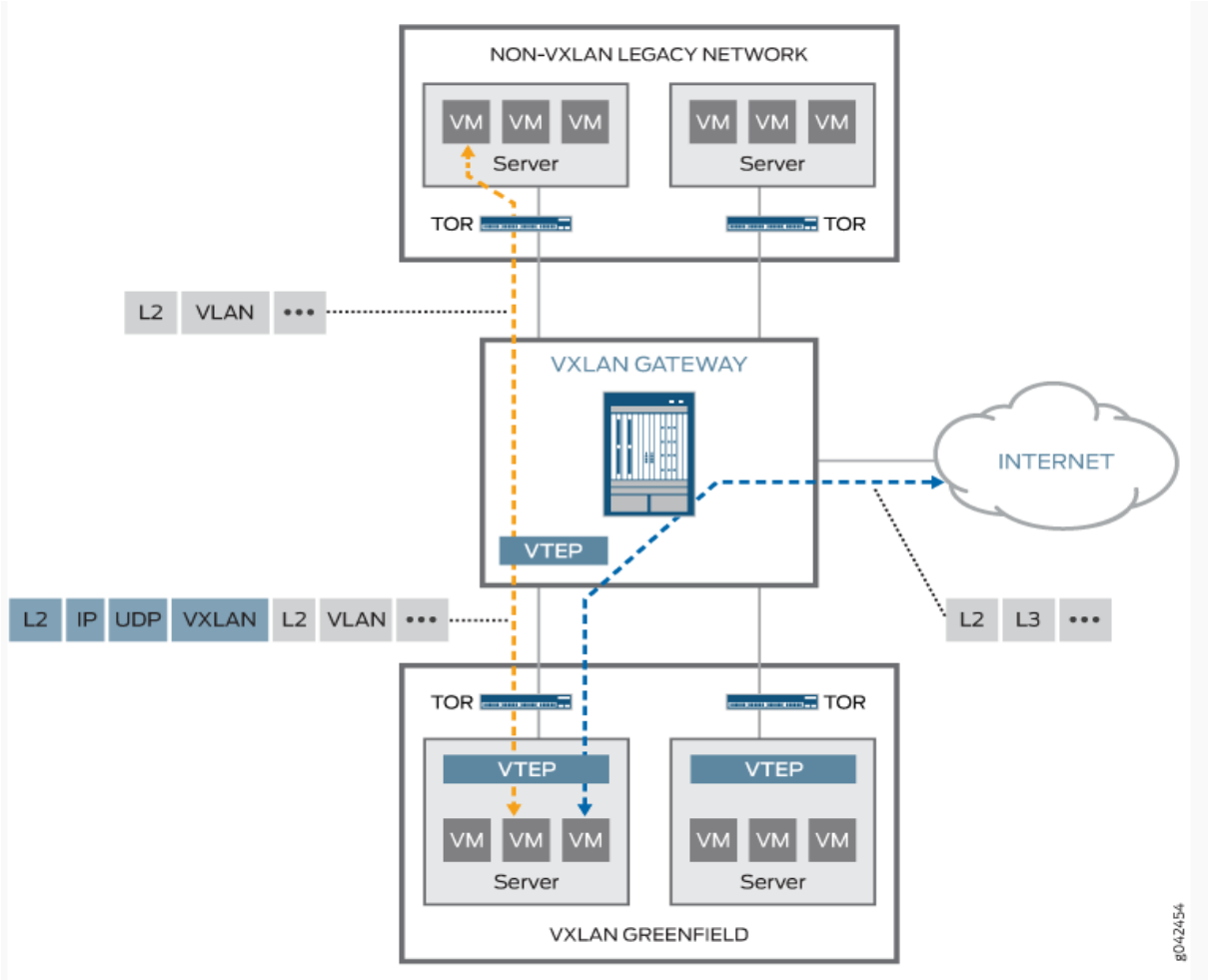

Figura 8. Topologia VXLAN[3.8]

Configurare rapidă CLI

```
set switch-options vtep-source-interface lo0.0
set bridge-domains vlan-5 vxlan vni 100
set bridge-domains vlan-5 vxlan multicast-group 239.1.1.1
set bridge-domains vlan-5 vlan-id 100
set bridge-domains vlan-5 routing-interface irb.0
set bridge-domains vlan-5 interface xe-1/0/0.0
set bridge-domains vlan-6 vxlan vni 200
set bridge-domains vlan-6 vxlan multicast-group 239.1.1.1
set bridge-domains vlan-6 vlan-id 200
set bridge-domains vlan-6 routing-interface irb.1
set bridge-domains vlan-6 interface xe-2/0/0.0
set interfaces xe-1/0/0 vlan-tagging
set interfaces xe-1/0/0 encapsulation flexible-ethernet-services
set interfaces xe-1/0/0 unit 0 encapsulation vlan-bridge
set interfaces xe-1/0/0 unit 0 vlan-id 100
set interfaces xe-2/0/0 vlan-tagging
set interfaces xe-2/0/0 encapsulation flexible-ethernet-services
set interfaces xe-2/0/0 unit 0 encapsulation vlan-bridge
set interfaces xe-2/0/0 unit 0 vlan-id 200
set interface irb unit 0 family inet address 5.5.5.1/24
set interface irb unit 1 family inet address 6.6.6.1/24
set interfaces lo0 unit 0 family inet address 3.3.3.3/32
set protocols ospf area 0.0.0.0 interface ge-8/3/8.0
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-0/1/3.0
set protocols ospf area 0.0.0.0 interface ge-8/3/2.0
set protocols pim rp static address 10.2.1.3
set protocols pim interface lo0.0 mode bidirectional-sparse
set protocols pim interface ge-8/3/8.0 mode bidirectional-sparse
set protocols pim interface xe-0/1/3.0 mode bidirectional-sparse
set protocols pim interface ge-8/3/2.0 mode bidirectional-sparse
set chassis redundancy graceful-switchover
set chassis aggregated-devices ethernet device-count 10
set chassis fpc 1 pic 0 tunnel-services bandwidth 10g
set chassis network-services enhanced-ip
```

Configurarea VXLAN(prezentare pas cu pas a codului de mai sus) [3.8]

Următorul exemplu arată cum se seteaza o configurație de bază cu VXLAN domenii implicite și opțiuni de comutare. Pentru a configura VXLAN pe un router MX Series, se folosesc urmatorii pași:

1. Configurarea interfetei VTEP sub comutare opțiuni pentru switch-ul .

user@router# set switch-options vtep-source-interface lo0.0

2. Setarea unui grup numit VLAN-VLAN 5 și setarea Identificatorului de retea VXLAN (VNI) la 100.

user@router# set bridge-domains vlan-5 vxlan vni 100

3. Configurarea grupului de adrese multicast VLAN-5 pentru VXLAN.

user@router# set bridge-domains vlan-5 vxlan multicast-group 239.1.1.1

4. Setarea ID-ului VLAN la 100 pentru VLAN-5.

user@router# set bridge-domains vlan-5 vlan-id 100

5. Configurarea puntii integrate și de rutare (IRB) pentru VLAN-5.

user@router# set bridge-domains vlan-5 routing-interface irb.0

6. Alocarea interfației XE-1/0 / 0,0 lui VLAN 5.

user@router# set bridge-domains vlan-5 interface xe-1/0/0.0

7. Setarea unui grup VLAN numit -VLAN 6 și setarea Identificatorului de Retea VXLAN (VNI) la 200.

user@router# set bridge-domains vlan-6 vxlan vni 200

8. Configurarea grupului de adrese multicast VLAN-6.

user@router# set bridge-domains vlan-6 vxlan multicast-group 239.1.1.1

9. Setarea ID-ului VLAN la 100 pentru VLAN-6.

user@router# set bridge-domains vlan-6 vlan-id 200

10. Configurarea IRB pentru-VLAN 6.

user@router# set bridge-domains vlan-6 routing-interface irb.1

11. Alocarea interfației XE-2/0 / 0,0 la VLAN 6.

user@router# set bridge-domains vlan-6 interface xe-2/0/0.0

12. Configurarea VLAN pentru-xe 1/0/0.

user@router# set interfaces xe-1/0/0 vlan-tagging

13. Configurarea flexibil încapsulare servicii Ethernet la xe 1/0/0.

user@router# set interfaces xe-1/0/0 encapsulation flexible-ethernet-services

14. Configurarea încapsularii puntii VLAN pentru-xe 1/0/0 unitatea 0˙.

user@router# set interfaces xe-1/0/0 unit 0 encapsulation vlan-bridge

15. Setarea unitatii 0 VLAN ID-xe 1/0/0 la 100.

user@router# set interfaces xe-1/0/0 unit 0 vlan-id 100

16. Configurarea VLAN pentru-xe 2/0/0

user@router# set interfaces xe-2/0/0 vlan-tagging

17. Configurarea flexibil serviciu Ethernet încapsulare pe-XE 2/0/0.

user@router# set interfaces xe-2/0/0 encapsulation flexible-ethernet-services

18. Configurarea incapsularii VLAN pentru-xe 2/0/0 unitatea 0˙.

user@router# set interfaces xe-2/0/0 unit 0 encapsulation vlan-bridge

19. Setarea ID-XE 2/0/0 unitatea 0 VLAN ID la 200.

user@router# set interfaces xe-2/0/0 unit 0 vlan-id 200

20. Configurarea unitatii 0 IRB.

user@router# set interface irb unit 0 family inet address 5.5.5.1/24

21. Configurarea unitatii 1 IRB.

user@router# set interface irb unit 1 family inet address 6.6.6.1/24

22. Setarea adresei familiei INET pentru unitatea loopback 0.

user@router# set interfaces lo0 unit 0 family inet address 3.3.3.3/32

23. Configurarea OSPF pentru interfața ge-8/3 / 8.0.

user@router# set protocols ospf area 0.0.0.0 interface ge-8/3/8.0

24. Configurarea OSPF pentru interfata loopback.

user@router# set protocols ospf area 0.0.0.0 interface lo0.0

25. Configurarea OSPF pentru interfața-xe 0/1 / 3.0.

user@router# set protocols ospf area 0.0.0.0 interface xe-0/1/3.0

26. Configurarea OSPF pentru interfața ge-8/3 / 2.0.

user@router# set protocols ospf area 0.0.0.0 interface ge-8/3/2.0

27. Configurarea adresei statice pentru modul de interfață fizică (PIM) punctul de întâlnire (RP).

user@router# set protocols pim rp static address 10.2.1.3

28. Configurarea interfatei loopback a modului bidirecțional pentru protocolul PIM.

user@router# set protocols pim interface lo0.0 mode bidirectional-sparse

29. Setarea interfetei GE-8 / 3 / 8.0 la modul bidirecțional pentru protocolul PIM.

user@router# set protocols pim interface ge-8/3/8.0 mode bidirectional-sparse

30. Configurarea interfeței-xe 0/1 / 3,0 la modul bidirecțional pentru protocolul PIM.

user@router# set protocols pim interface xe-0/1/3.0 mode bidirectional-sparse

31. Setarea interfetei GE-8 / 3 / 2.0 la modul bidirecțional pentru protocolul PIM.

user@router# set protocols pim interface ge-8/3/2.0 mode bidirectional-sparse

32. Configurarea latimii de bandă pentru FPC 1 / PIC 0.

user@router# set chassis fpc 1 pic 0 tunnel-services bandwidth 10g

33. Activarea IP-ului pentru servicii de rețea.

user@router# set chassis network-services enhanced-ip

Rezultate

De la modul de configurare, confirmarea configurarilor se face introducând următoarele comenzi:

user@router# show switch-options

```
switch-options {
vtep-source-interface lo0.0;
}
```

user@router# show bridge-domains

```
bridge-domains {
vlan-5 {
vxlan {
vni 100;
multicast-group 239.1.1.1;
        }
vlan-id 100;
```

```
routing-interface irb.0;
interface xe-1/0/0.0;
          }
vlan-6 {
vxlan {
vni 200;
multicast-group 239.2.1.1;
           }
vlan-id 200;
routing-interface irb.1;
interface xe-2/0/0.0;
}
}
```

user@router# show interfaces

```
interfaces {
xe-1/0/0 {
vlan-tagging;
encapsulation flexible-ethernet-services;
unit 0 {
encapsulation vlan-bridge;
vlan-id 100;
          }
  }
xe-2/0/0 {
vlan-tagging;
encapsulation flexible-ethernet-services;
unit 0 {
encapsulation vlan-bridge;
vlan-id 200;
           }
    }
irb {
unit 0 {
family inet {
address 5.5.5.1/24;
                        }
                    }
unit 1 {
family inet {
```

```
address 6.6.6.1/24;
     }
                  }
   }
lo0 {
unit 0 {
family inet {
address 3.3.3.3/32;
      }
}
   }
}
```

user@router# show protocols ospf

```
area 0.0.0.0 {
interface ge-8/3/8.0;
interface lo0.0;
interface xe-0/1/3.0;
interface ge-8/3/2.0;
      }
```

user@router# show protocols pim

```
rp {
static {
address 10.2.1.3;
         }
   }
```

user@router# show chassis

```
redundancy {
graceful-switchover;
        }
aggregated-devices {
ethernet {
device-count 10;
  }
       }
fpc 1 {
pic 0 {
tunnel-services {
```

bandwidth 10g;
}
}
}
network-services enhanced-ip;

Verificare

Confirmarea că funcționează corect configurația.

- Verificarea accesibilității
- Verificarea VLAN

Verificarea accesibilității

Scop:

Verificare dacă rețeaua este în funcțiune cu interfețele corespunzătoare și rutele instalate.

user@router> show interfaces terse irb

Interface Admin Link Proto Local Remote

irb up up

irb.0 up up inet 5.5.5.1/24 multiservice

irb.1 up up inet 6.6.6.1/24 multiservice

user@router> ping 5.5.5.1/24

PING 5.5.5.1 (5.5.5.1): 56 data bytes

64 bytes from 5.5.5.1: icmp_seq=0 ttl=64 time=0.965 ms

64 bytes from 5.5.5.1: icmp_seq=1 ttl=64 time=0.960 ms

64 bytes from 5.5.5.1: icmp_seq=2 ttl=64 time=0.940 ms

^C

--- 1.1.1.1 ping statistics ---

3 packets transmitted, 3 packets received, 0% packet loss

round-trip min/avg/max/stddev = 0.940/0.955/0.965/0.011 ms

Interpretare:

Se utilizeaza comanda IRB **show interfaces terse irb** pentru a verifica dacă interfața IRB a fost configurata corect. Interfețele irb.0 și irb.1 ar trebui să afișeze adresele multiservice inet adecvate.

Utilizați comanda **ping** pentru a confirma că rețeaua este conectată la adresa multiservice IRB.

Verificarea VLAN[3.8]

Scop:

Verificarea că VXLAN functioneaza și protocoalele corespunzătoare sunt activate.

user@router> show interfaces vtep

Physical interface: vtep, Enabled, Physical link is Up

Interface index: 133, SNMP ifIndex: 575

Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: 1600, Speed: Unlimited

Device flags : Present Running

Interface flags: SNMP-Traps

Link type : Full-Duplex

Link flags : None

Last flapped : Never

Input packets : 0

Output packets: 0

Logical interface vtep.32768 (Index 334) (SNMP ifIndex 607)

Flags: Up SNMP-Traps Encapsulation: ENET2

VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.255.187.32, L2 Routing Instance: default-switch, L3 Routing Instance: default

Input packets : 0

Output packets: 0

user@router> show l2-learning vxlan-tunnel-end-point remote mac-table

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Logical system : <default>
Routing instance : default-switch
Bridging domain : vlan-5+100, VLAN : 100, VNID : 100
Bridging domain : vlan-6+200, VLAN : 200, VNID : 200

user@router> show l2-learning vxlan-tunnel-end-point source

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	10.255.187.32	lo0.0	0

L2-RTT	Bridge Domain	VNID	MC-Group-IP
default-switch	vlan-5+100	100	239.1.1.1
default-switch	vlan-6+200	200	239.1.1.1

Interpretare:

Utilizarea comenzii **show interface vtep** pentrua afișa informații despre configurația VXLAN.

Utilizarea comenzii **show l2-learning vxlan-tunnel-end-point remote mac-table** pentru a confirma faptul că grupurile domeniilor VLAN au fost configurate corect.

Utilizarea comenzii **show l2-learning vxlan-tunnel-end-point source** pentru a confirma adresele IP multicast pentru grupurile domeniilor VLAN.

Bibliografie:

Capitolul 1:

1.1) https://en.wikipedia.org/wiki/Software-defined_networking - caracteristicile arhitecturii SDN
1.2) https://www.opennetworking.org/about/onf-overview -prezentare ONF
1.3) http://dl.acm.org/citation.cfm?id=556322 – citat dat de membrii AT&T Labs Geoplex
1.4) http://david.choffnes.com/classes/cs4700sp15/papers/sdnhistory.pdf – istoria SDN
1.5) https://www.scribd.com/doc/237513189/Software-Defined-Network-SDN - citat Pankaj Shah

Capitolul 2:

2.1) *"Software-Defined Networking (SDN): The New Norm for Networks"* - Open Networking Foundation
2.2) *"OpenFlow: Enabling innovation in campus networks"* - Nick McKeown; et al. (April 2008). ACM Communications Review.
2.3) *" OpenFlow v1.4"* - Open Networking Foundation
2.4) http://searchsdn.techtarget.com/feature/OpenFlow-protocol-primer-Looking-under-the-hood
2.5) http://flowgrammable.org/sdn/openflow/

Capitolul 3:

3.1) http://www.juniper.net/techpubs/en_US/learn-about/LA_VXLANinDCs.pdf
3.2) „*Juniper Networks- Learn About VXLAN in Virtualized Data Center Networks*”
3.3) https://blogs.vmware.com/smb/2013/09/vxlan-what-it-is-components-that-make-it-work-and-benefits.html
3.4) Andrew Tanenbaum – "Computer Networks" editia V, Ed. Prentice Hall 2011

3.5)http://www.borgcube.com/blogs/wp-content/uploads/2011/11/VXLAN-Headers1.png)
http://www.borgcube.com/blogs/2011/11/vxlan-primer-part-1/
3.6) http://blog.scottlowe.org/2011/11/30/vxlan-and-layer-3-connectivity/
3.7) https://f5.com/resources/white-papers/vxlan-and-the-big-ip-platform
3.8)http://www.juniper.net/documentation/en_US/junos14.1/topics/example/vxlan-default-switch-example.html#jd0e764