Implementacija OPENSTACK u multitenant mrežnom okruženju

Nebojša Kuduz, Mihajlo Travar, Jelena Kuduz, Mišo Lazić
Telekomunikacije Republike Srpske a.d., Banja Luka
Nebojsa.Kuduz@mtel.ba, Mihajlo.Travar@mtel.ba, Jelena.Kuduz@mtel.ba, Miso.Lazic@mtel.ba

Sadržaj - **U radu je prikazan jedan način primjene OpenStack arhitekture u dizajnu javnog cloud servisa. Primjenjeni pristup omogućio nam je da sa ograničenim budžetom prikažamo koncepte softverski vođene infrastrukture. Implementacijom otvorenog softverskog steka dobili smo najisplativije rešenje za realizaciju centra podataka. OpenStack nam je omogućio da virtualizujemo pojedine mrežne funkcije kao što su rutiranje ili upravljanje softverskim svičevima bez kupovine dodatnog hardvera. U radu su prikazani napredni koncepti kreiranja javnog clouda, kreiranja multitenant mreža i primjer prekrivajućeg umrežavanja. Primjenom OpenStack kao rezultat smo dobili primjer modularne i fleksibilne arhitekture javnog klauda koja zadovoljava postojeće tehnološke zahtjeve internet provajdera.**

Ključne riječi- OpenStack, SDN, NFV

I. UVOD

Širenje internet saobraćaja i različiti cloud koncepti uslovaljavaju prisutnost raličitih arhitektura u postojećim centrima podataka. Većina telekomunikacionih kompanija posjeduje centre podataka sa tradicionalnom arhitekturom i potrebno je da prilagode svoje centre podataka najnovijim tehnologijama. Većina postojećih arhitektura, ukoliko ne pretrpi bitne izmjene, ne može da zadovolji potrebe zahtjevnih internet servisa i da se prilagodi potrebama novih. Izgradnja savremenog data centra predstavlja imperativ za telekomunikacione kompanije i IT stručnjake.

Virtuelizacija je postala standard u konfiguraciji centara podataka i njena primjena poboljšava konsolidaciju resursa i doprinosi uštedi i povećanju raspoloživosti sistema. Softverski definisane mreže SDN i otvorenost hardvera i softvera omogućili su da velike kompanije poput Amazona i Googla diktiraju i globalni pravac razvoja *cloud* infrastrukture. Za kompleksnu orkestraciju i rezervaciju resursa sve više se koriste metapodaci. Jedan od ključnih projekata koji treba da podrži ove koncepte je OpenStack.

II. OPENSTACK

OpenStack je javno dostupno rešenje za upravljanje cloud infrastrukturom, koje su kao projekat pokrenuli NASA i Rackspace 2010. godine. To je besplatna cloud platforma otvorenog koda koja podržava sve vrste cloud okruženja. Cilj projekta je da se poboljša i pojednostavi implementacija, postigne veća skalabilnost i da se omogući što veći skup funkcionalnosti u cloud okruženju. Cloud stručnjaci iz cijelog svijeta intenzivno doprinose razvoju OpenStack projekta.

OpenStack pruža *Infrastructure-as-a-Service* IaaS rješenje putem različitih međusobno povezanih komplementarnih servisa odnosno dijelova koje možemo smatrati zasebnim projektima i svaki od njih je dobio popularno kodno ime. Svaki servis odnosno dio posjeduje programski aplikativni interfejs koji olakšava integracije sa drugim dijelovima steka.

Taj niz povezanih projekata omogućuje da kontrolišemo velike skupove resursa za izračunavanje, skladištenje i umrežavanje resursa u centru podataka. *OpenStack*-om se upravlja preko kontrolne table koja administratorima pruža kontrolu i sredstvo kojim se korisnicima na jednostavan način isporučuju traženi resursi putem određenog mrežnog interfejsa.

Moduli OpenStack-a i njihove funkcije koje smo implementirali na infrastrukturi centra podataka su:

Nova - upravlja životnim ciklusom računarskih instanci u OpenStack okruženju. Ovaj projekat je srce sistema i odgovoran je za kreiranje, alociranje i prestanak rada virtualnih mašina po zahtjevu.

Neutron – isporučuje mrežu kao servis za ostale OpenStack module kao što je OpenStack Nova. Neutron pruža korisnički API koji omogućuje da definišemo mreže i dodatke mreža. Ima prilagodljivu arhitekturu koja podržava različite mrežne tehnologije i vodeće proizvođače mrežnih komponenti.

Swift – omogućuje pohranjivanje i preuzimanje nestrukturiranih objekata podataka i pristupa podacima putem RESTfull HTTP API-ja. Zasnovan je na replikacijskoj i skalarnoj arhitekturi koja je otporna na greške.

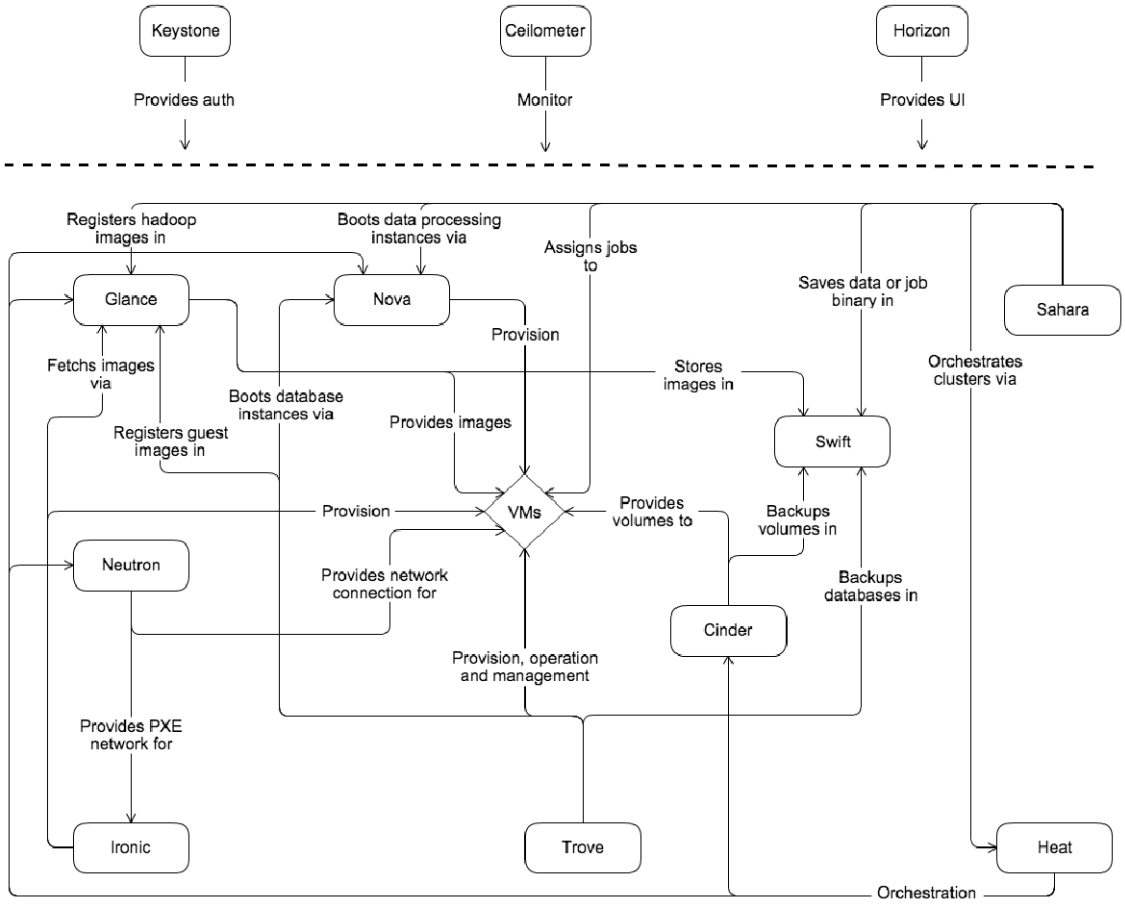
Cinder – omogućuje pohranu blokova podataka i pridruživanje blok uređaja instancama. Njegova arhitektura pojednostavljuje kreiranje i upravljanje uređajima za skladištenje i rad sa blok uređajima.

Keystone – omogućuje autentifikacijske i autorizacijske usluge za druge OpenStack projekte. Služi kao veza za sve ostale OpenStack servise.

Glance – služi za pohranjivanje i preuzimanje disk *image*-a virtuelnih mašina. Nova koristi *image* tokom kreiranja i dodjeljivanja instanci.

Ceilometar – prati i mjeri OpenStack cloud servise u svrhu naplate, vrednovanja, skalabilnosti i statistika.

Heat – Orkestrira OpenStack cloud servise i aplikacije



Slika 1. Dijagram glavnih servisa OpenStacka i veze između njih

Horizon – omogućuje veb bazirani *self-service* portal za interakciju sa ostalim OpenStack modulima, kao što su pokretanje instance, dodjela IP adrese, konfigurisanje kontrola pristupa.

Pokretanja virtuelne mašine ili instance uključuje mnogostruke interakcije između servisa. Sl. 1 [1] prikazuje dijagram arhitekture standardnog OpenStack okruženja.

Naša implementacija OpenStack-a sastoji se od više nezavisnih dijelova odnosno OpenStack servisa [2], [3]. Svi se autentifikuju preko zajedničkog servisa za identifikaciju. Pojedinačni servisi komuniciraju jedni s drugima putem javnog API-ja osim u dijelu gdje su potrebne posebne administratorske privilegije. OpenStack servisi se sastoje od nekoliko procesa. Svaki servis ima najmanje jedan API proces, koji sluša API zahtjeve, predprocesira ih i prosljeđuje drugim dijelovima servisa. Stvarni posao obavljaju različiti procesi unutar OpenStack servisa sa izuzetkom servisa za provjeru identiteta.

Za komunikaciju između procesa istog servisa koristili smo *Advanced Message Queuing Protocol*-AMQP broker poruka. Stanje servisa se pohranjuje u bazi podataka. Prilikom razvoja OpenStack okruženja mogli smo da biramo između nekoliko brokera poruka i baza. U ovom projektu korišteni su RabbitMQ i MariaDB.

Korinici mogu pristupiti OpenStack okruženju putem kontrolnog panela, CLI klijenta i alata koji komuniciraju putem API-ja. Za pristup aplikacijama omogućeno je nekoliko SDK. Svi ovi načini komunikacije se svode na REST API pozive ka OpenStack servisima.

III. DIZAJN I ARHITEKTURA MREŽE

U ovom poglavlju opisan je koncept korištenja Neutron servisa zasnovanog na OVS i VXLAN u našoj implementaciji mrežne komponente višečvorne OpenStack arhitekture.

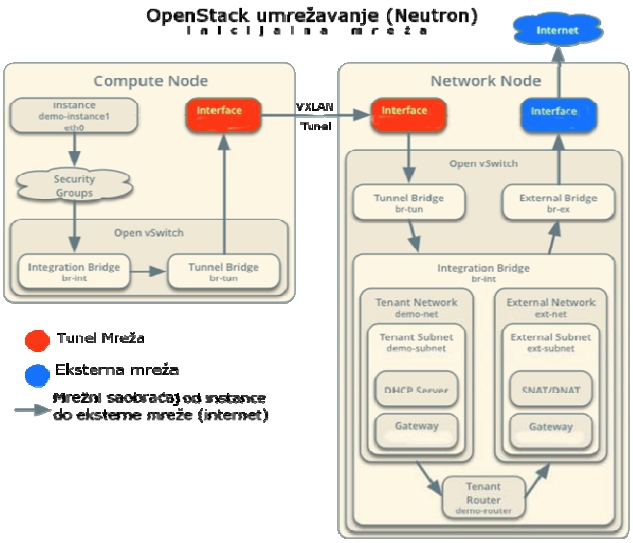
Mrežni koncept javnog *clauda* zahtjeva najmanje jednu spoljnu mrežu. Mreža nije samo softverski upravljiva već dijelom ima pogled i na spoljnu mrežu koja je dostupna izvan OpenStack okruženja. IP adrese spoljne mreže su dostupne svima pa smo DHCP onemogućili u spoljnoj mreži. Mrežna konfiguracija može da ima jednu ili više unutrašnjih mreža i to su SDN mreže koje direktno povezuju instance odnosno virtuelne mašine. Instancama interne mreže ili onima koje su u podmreži konektovanoj preko interfejsa zajedničkog rutera omogućili smo da pristupe drugim instancama te mreže.

Da bi se iz spoljne mreže pristupilo instancama i obrnuto, implementirali smo virtuelne rutere. Svaki ruter ima jedan *gateway* koji je veza sa spoljnom mrežom i više interfejsa koji su povezani na podmreže. Kao i u slučaju fizičkih rutera, podmreže mogu da pristupe instancama drugih podmreža koje su povezane na isti ruter i instance mogu da pristupe spoljnoj mreži preko rutera.

Omogućili smo da se IP adrese spoljne mreže pridruže portovima na unutrašnjoj mreži. Spoljne javne IP adrese pridružili smo VM portovima i na taj način smo omogućili da vanjski entiteti pristupaju instancama unutar mreže. Koncept umrežavanja podržava i sigurnosne grupe kojima smo omogućili administratorima da definišu *firewall* pravila unutar grupa. Instance mogu pripadati jednoj ili više sigurnosnih grupa koje blokiraju ili deblokiraju portove, opseg portova ili propuštaju određene tipove saobraćaja ka i od instanci.

Svaka instalacija mrežnog servisa koristi *core* dodatak i dodatak sigurnosnih grupa [4]. Pored toga, opciono smo omogućili implementaciju *Firewall-as-a-service* FWaaS, *Load-balancing-as-a-service* LBaaS i VPNaaS dodataka.

U OpenStack mrežnom servisu koristili smo dodatke i agente da bi podržali različite proizvođače mrežne infrastrukture i da bi se realizovali različiti zahtjevi sa ciljem dobijanja fleksibilne mrežne arhitekture. U dizajnu mrežne arhitekture koristili smo Neutron priključke za *Open vSwitch* i *Linux bridge*. Agenti koji su implementirani su *Layer-3* (L3) i *Layer-2* (L2). Iskoristili smo OpenStack mehanizme razmjene poruka da bi rutirali informacije između Neutrona i raznih agenata i da bi informacije o mreži i dodacima pohranili u bazu. Da bi instancama omogućili mrežni pristup, OpenStack umrežavanje odnosno Neutron uglavnom obavlja komunikaciju sa OpenStack računarskim servisom odnosno sa Nova, Sl. 2.



Slika 2. Komunikacija mrežnog i računarskog čvora

Najviše pažnje zauzeo je razvoj mrežnog koncepta i u prikazanom rješenju umrežavanje se sastoji od tri vrste čvorova. Servisni čvor omogućuje klijentima API mreže i obrađuje dolazne zahtjeve prije prosljeđivanja u red zahtjeva. Nakon inicijalne obrade, zahtjeve obrađuju drugi čvorovi. Servisni čvor hostuje mrežni servis i aktivne mrežne dodatke. Predstavljeno okruženje koristi kontroler čvor koji hostuje klijentske API-je i vrši komunikaciju sa ostalim servisima, pa kontoler čvor vrši ulogu servisnog čvora. Mrežni čvor upravlja najvećim dijelom OpenStack mreže. On hostuje DHCP, L3 i L2 agente, osnovene agente i agente metapodataka. Pored dodataka i priključaka koji zahtjevaju agente, mrežni čvor pokreće i instance priključenih agenata.

U prezentovanoj mreži ovaj čvor pokreće instance upravljačkih programa za *Open vSwitch* (OVS) i *Linux Bridge* agente. Računarski čvor hostuje same računarske instance. Da bi računarske instance povezali sa mrežnim servisom, na instancama sumo pokrenuli L2 agente. Kao i u svim drugim sistemima koji rukuju sa podacima potrebno je da se pokrenu i instance priključnih agenata.

OpenStack mreža koristi Linux mrežne prostore imena da bi se spriječile kolizije između fizičke mreže na mrežnom hostu i logičke mreže koju koriste virtuelne mašine. U predloženoj mreži koristili smo OpenStack umrežavanje odnosno Neutron umjesto tradicionalnog umrežavanja preko Nova servisa.

Neutron [5]–[7] onemogućuje kolizije između različitih logičkih mreža koje nisu izrutirane jedne prema drugima. Mrežni prostor, *network namespace*, je izolovano okruženje koje ima svoj sopstveni stek za umrežavanje. Mrežni prostor imena ima svoje mrežne interfejse, rute i iptables pravila. To je osobina Linux kernela koja omogućuje da grupa procesa ima svoj nezavisni mrežni stek (interfejse, ruting tabele, iptables pravila) koji se razlikuje od onog na hostu. Ono što je u linux *chroot jail* za fajlove, za mrežu je mrežni prostor imena. U našem primjeru kreirali smo na servisnom čvoru dva tipa mrežnih prostora imena da bi izbjegli kolizije na nivou podmreža.

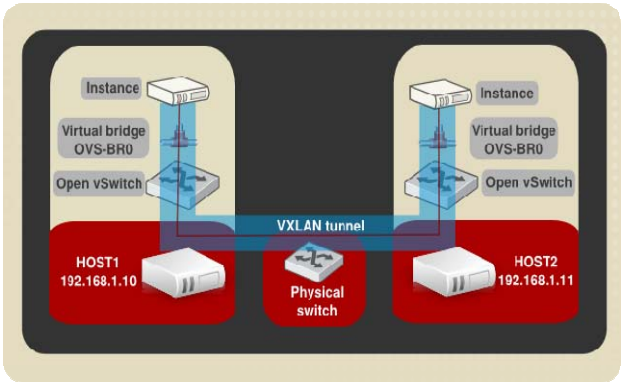
Prvi prostor imena je *qdhcp* koji sadrži *tap* interfejse i *dnsmasq* proces koji osluškuje na tap interfejsu i omogućuje DHCP servis za podmrežu. Primjena ovog mrežnog prostora imena omogućila nam je preklapanje IP adresa unutar podmreža mrežnog hosta. I za druge podmreže unutar hosta formirali smo ovaj prostor imena radi mogućeg preklapanja IP adresa i DHCP servisa. Drugi prostor imena je *qrouter* koji ima interfejse sa prefiksima *qr* i *qg* i odgovarajuće rute. *Qrouter* prostor imena odgovoran je za kreiranje logičkih rutera mreže.

Prije pokretanja bilo koje instance, kreirali smo virtuelnu mrežnu infrastrukturu koja uključuje kreiranje spoljne i tenant mreže. Prikazan je osnovni pregled mrežne arhitekture i način na koji saobraćaj putuje od instance do spoljne mreže odnosno interneta. Virtuelna mrežna infrastruktura sastoji se od spoljne i tenant mreže. Spoljna mreža omogućila je instancama pristup internetu. Predefinisano, pristup od instance dozvoljen je samo korištenjem NAT. Pristup sa interneta do pojedinih instanci omogućili smo korištenjem plivajućih *floating* IP adresa i podešavanjem odgovarajućih pravila sigurnosnih grupa. Vlasnik ove mreže je administrator budući da ona pruža spoljni pristup instancama za sve tenant mreže. Kreiranjem tenant mreža omogućili smo instancama interni mrežni pristup. Predloženom arhitekturom mreže izolovali smo ovaj tip mreže od drugih tenenta. Vlasnik ovog tipa mreže ne mora biti administrator jer služi da omogući međusobni pristup instancama unutar iste tenant mreže.

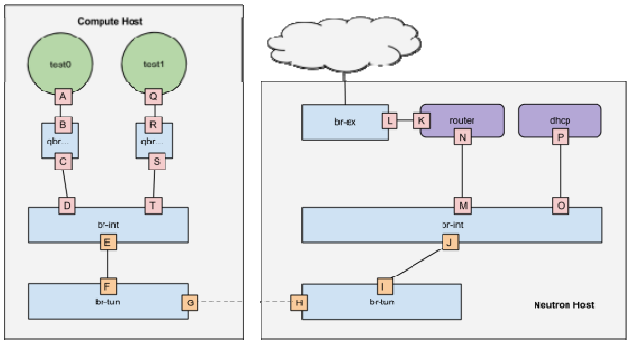
U projektu smo koristili dodatak modularnog sloja ML2, [8]. ML2 koristi OVS mehanizam odnosno agent za izgradnju virtuelnog okvira za instance. Pomoću njega smo definisali i OVS konfiguraciju kao i VXLAN tunele.

Prije nastanka ML2 svaki proizvođač morao je da razvije sve komponente za mrežni dodatke uključujući IPAM, pristup bazi podataka itd. Sa dodatkom modularnog sloja ML2 razdvojili smo glavne funkcionalnosti kao što su IPAM, upravljanje identitetima virtuelne mreže od specifičnih funkcija implementiranih od strane proizvođača. Njegovom primjenom pojednostavili smo implementaciju mrežnih dodataka. U ML2 terminologiji upravljanje mrežnim tipovima se naziva "tip upravljanja", a dio koji se tiče specifične implementacije nazvali smo "mehanizmom upravljanja". Implementacijom ML2 uključili smo podršku za upravljačke programe i za lokalne, flat, vlan, gre i vxlan tipove mreža i pojednostavili smo rad sa postojećim *Open vSwitch*, *Linux Bridge* i *HyperV* L2 agentima.

Primjenom tunela omogućili smo enkapsulaciju mrežnog saobraćaja između fizičkih hostova. Instance na ovaj način komuniciraju kao da djele istu L2 mrežu. Na Sl. 3 [9] su prikazane dvije instance koje su pokrenute na različitim hostovima i međusobno povezane sa vxlan tunelom.



Slika 3. Primjer vxlan tunela



Slika 4. Dijagram komunikacije u virtuelnoj mreži

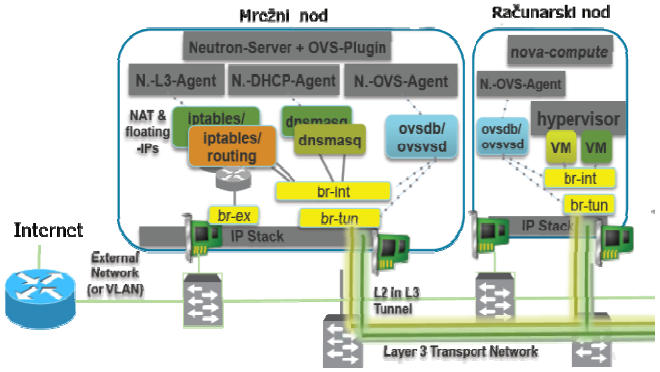
IV. KREIRANJE VIRTUELNOG SVIČA I MEĐUMREŽNA KOMUNIKACIJA

Izolaciju tenant mreža, provajderske mreže i upravljačke mreže postigli smo kreiranjem OVS i tri mosta [10]–[12]. To su *br-int*, *br-ex* i *br-tun*. Sa br-int mostom kreirali smo *Open vSwitch* upravljački mehanizam i iskoristili smo ga kao integracioni most na kom smo kreirali portove i uključili ih u topologiju virtuelnog sviča. Br-ex je OVS most koji smo iskoristili da povežemo fizičke portove, npr eth0, tako da javni IP saobraćaj ka tenant mreži može biti prihvaćen sa fizičke mrežne infrastrukture i interneta i biti izrutiran do mrežnih portova tenanta. Br-tun je tunel most koji smo koristili da međusobno povežemo OpenStack čvorove. Ovaj most smo iskoristili kako bi mrežni saobraćaj tenanta, korištenjem vxlan tenel protokola, proticao između svih računarskih čvorova na kojima su pokrenute instance tenant mreža. Na Sl. 4 [13] je logički prikaz mrežne komunikacije. Računarski čvor obuhvata mrežne instance A, B i C. Odlazni paket polazi sa eth0 virtuelne instance koji je konektovan na tap interfejs hosta. Tab uređaj je povezan na Linux most qbr. U idealnom slučaju TAP interfajs vnet0 je povezan direktno sa integracionim mostom br-int. Ovo nije moguće zbog načina na koji su implementirane OpenStack sigurnosne grupe. Uloga ovog logičkog interfejsa je da podrži primjenu sigurnosnih pravila. Drugi interfejs koji je povezan na most, *qvb*, povezuje tap interfejs sa integracionim mostom *br-int*. Integracioni most br-int (D, E) vrši označavanje paketa saobraćaja koji dolaze od i ka instanci. Interfejs *patch-tun* povezuje integracioni most sa mostom tunela *br-tun*. Most tunela računarskog čvora (F,G) prevodi označeni saobraćaj sa integracionog mosta u vxlan tunel. Prevod vlan id-a i tenel id-a vrše OpenFlow pravila koja su pokrenuta na *br-tun*. Inicijalno, ova pravila su odgovorna da se izvrši mapiranje između vlan id koji koristi integracioni most i tunel id koji koristi vxlan tunel. Saobraćaj stiže na mrežni čvor putem vxlan tunela vezanog za br-tun (H, I). Most ima tabelu protoka sličnu onoj na br-tun računarskog čvora. Integracioni most mrežnog čvora služi da poveže instance sa mrežnim servisima kao što su ruteri i dhcp server. Veza sa mostom tunela br-tun vrši se sa patch-tun interfejsom. Svaka tenant mreža za koju je omogućen dhcp ima dhcp server pokrenut na mrežnom kontroleru. Dhcp server je instanca *dnsmasq* koji je pokrenuta unutar mrežnog prostora imena. U predloženoj konfiguraciji realizovali smo dva mrežna prostora imena, *qdhcp* i *qrouter*. Neutron ruter (M, N) je mrežni prostor imena sa skupom ruting tabela i iptables pravila

pomoću kojih rutiramo saobraćaj između podmreža. U predloženoj konfiguraciji realizovali smo dva interfejsa, *qg* i *qr*. Sa *qg* smo povezali ruter sa *gateway*, a sa *qr* smo povezali ruter sa integracionim mostom. Implementacija *nat* tabela *netfilter*-a unutar ruterovog prostora imena poslužila nam je za pridruživanje plivajućih javnih IP adresa pojedinim instancama. Primjenom SNAT i DNAT pravila mapirali smo saobraćaj između javne plivajuće IP adrese i privatne adrese tenant instance. Spoljni saobraćaj (K, L) prolazi kroz *br-ex* i *qg* virtuelni interfejs koji je konektovan na *br-ex*.

V. IMPLEMENTACIJA MULTITENANT MREŽE

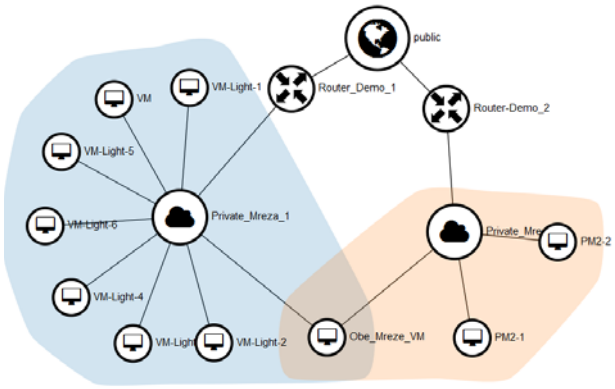
Implementirani stek sastoji se od tri čvora koji su fizički povezani L3 svičem na javnu mrežu. Budući da je u mreži prisutan spoljnji ruter koji igra ulogu *gateway*-a ka javnoj mreži, dodali smo interfejs kontrolera u most i time ostvarili spoljni pristup instancama unutar tenant mreže, Sl.5.



Slika 5. Logičke komponente uključene u mrežnu komunikaciju

Neutron smo implementirali na jednom servisnom čvoru, a druga dva fizička čvora smo virtualizovali i oni imaju ulogu računarskih čvorova. Za te potrebe iskoristili smo Intel arhitekturu čipova i servere odgovarajućih performansi prema OpenStack matrici kompatibilnosti. Čvorovi smo podigli sa RHEL v.7 operativnim sistemom, a radi lakše manipulacije sa komponentama steka, OS čvorova pokreću se direktno sa SAN uređaja. Kapacitet LUN na SAN-u odabrali smo tako da dobijemo dovoljan prostor i za servise skladišta blok uređaja i za skladišta objekata. Za javni opseg adresa odabrali smo CIDR /27 opseg što je dovoljno za predloženu topologiju mreže koja se sastoji od dva rutera, dvije tenant mreže, mrežnog i dva računarska čvora, Sl. 6.

U OpenStack orkestratoru, kreirali smo dvije tenant mreže koje su nam omogućile konektivnost unutar projekta. Ove mreže su predefinisano izolovane i nemaju pristup drugim projektima. Tenant mreže koje smo kreirali su vxlan tipa. Na čvorovima smo instalirali posljednju verziju *OpenStack-Liberty* korištenjem razvojne stek distribucije. Za vezu između tenant mreža kreirali smo vxlan tunele koji koriste mrežno prekrivanje *overlay* da bi podržali komunikaciju između instanci. Kreirali smo OpenStack rutere i time smo omogućili da saobraćaj prolazi izvan vxlan tenant mreže. Provajdersku mreža smo iskonfigurisali tako da je međusobno dijele novokreirane tenant mreže.



Slika 6. Topologija predložene mreže - izgled u Horizon

Neutron smo realizovali sa ML2 dodatkom preko koga smo implementirali *OVS* i *LinuxBridge* upravljačke mehanizme kao i vxlan upravljački tip. Tipovi Neutron agenata koji smo pokrenuli na servisnom, odnosno na mrežnom čvoru su: Loadbalancer agent, Open vSwitch agent, DHCP agent, Metadata agent i L3 agent. Na računarskim čvorovima smo pokrenuli Open vSwitch agent. Pored Neutrona i Nove, instalacija je obuhvatila i Keystrone, Swift, Ceph, Cinder, Horizon, Glance i Heat.

VI. ZAKLJUČAK

Budućnost IT centara podataka usko je povezana sa softverski vođenom infrastrukturom SLI, SDN (*Software Defined Networking*) i NFV (*Network Functions Virtualization*) tehnologijama. Ove tehnologije su usmjerene na potpunu virtuelizaciju centara podataka i kompletne infrastrukture provajdera. Otvorenost hardvera i softvera ujedno omogućuje i rapidno smanjenje troškova IT infrasterukture. OpenStack je projekat za razvoj *cloud* rješenja bilo kog tipa i pruža mogućnost za realizaciju potpuno virtuelizovanih centara podataka.

U radu je pokazano da primjena OpenStack NFV koncepta obezbjeđuje telekomunikacionim provajderima konsolidaciju mrežne IT infrastrukture. Predstavili smo jedan način primjene OpenStack projekta u mreži provajdera zasnovan na realizaciji Neutron servisa primjenom VXLAN u multitenant mreži. Predložena konfiguracija omogućuje da se arhitektura centra podataka prilagodi najnovijim NaaS tehnologijama. Ovaj koncept je primjenjen u radu i kao rezultat smo dobili primjer softverski upravljive infrastrukture.

Velike veb kompanije i koncepti poput OpenStack-a preuzimaju primat u arhitekturi savremenih centara podataka. Orjentacija ka pomenutim tehnologijama uveliko će uticati na budući razvoj i tržišnu poziciju telekomunikacionih operatera i provajdera internet servisa i aplikacija.

LITERATURA

[1] Official OpenStack Documentation - Kilo install guide, “http://docs.openstack.org/, published by OpenStack community, 2015.

[2] Kevin Jackson, “OpenStack Cloud Computing Cookbook - Second Edition”, published by Packt Publishing, 2013.

[3] Omar Khedher, “Mastering OpenStack”, published by Packt Publishing, 2015.

[4] James Denton, “Learning OpenStack Networking (Neutron)”, published by Packt Publishing, 2014.

[5] Sriram Subramanian, “OpenStack Networking”, published by Packt Publishing, 2015.

[6] Nebojša Kuduz, Mihajlo Travar, Željko Mlinarević, “Automatizacija, interoperabilnost i problemi standardizacije u CLOUD virtuelnim sistemima”, INFOTEH-JAHORINA Vol. 13, March 2014.

[7] Openstack community, “https://opensource.com/resources/openstack-tutorials, October 2015.

[8] Official OpenStack Documentation, “http://docs.openstack.org/, published by OpenStack community, 2015.

[9] Openstack manuals – RedHat customer portal, “https://access.redhat.com “

[10] Mirantis.Inc.,“https://www.mirantis.com/openstack-unlocked/tutorials/”, 2015.

[11] Openstack official site, “Collective blog aggregator with most news/tech blogs”, published by docs.openstack.org, 2015.

[12] Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, Joe Topjian, “OpenStack Operations Guide-Set Up and Manage Your OpenStack Cloud”, published by O'Reilly Media, April 2014.

[13] RedHat OpenStack community portal, OpenStack Neutron Architecture, "https://openstack.redhat.com/Networking_in_too_much_detail"

ABSTRACT

The paper presents one way of OpenStack architecture implementation in the public cloud service design. Applied approach has allowed us to present the concept of software driven infrastructure using a limited budget. Open software stack usage provided us the most economical solution for the data centre implementation. Derived configuration of OpenStack has enabled virtualization of particular network functions such as routing and software switches management without purchasing additional hardware. We have presented advanced concepts of creating a public cloud, multitenant networks and an example of overlay networks. By using the OpenStack we accomplished the flexible public cloud architecture model that meets the current technological requirements of Internet providers.

Keywords- OpenStack, SDN, NFV

IMPLEMENTATION OF OPENSTACK IN MULTITENANT NETWORK ENVIRONMENT

Nebojša Kuduz, Mihajlo Travar, Jelena Kuduz, Mišo Lazić