UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

MIHAJLO SAVIĆ

SISTEM ZA NADZOR RAČUNARSKE INFRASTRUKTURE KAO SERVISA

DOKTORSKA DISERTACIJA

BANJA LUKA, 2020.

UNIVERSITY OF BANJA LUKA
FACULTY OF ELECTRICAL ENGINEERING

MIHAJLO SAVIĆ

COMPUTER INFRASTRUCTURE AS A SERVICE MONITORING SYSTEM

DOCTORAL DISSERTATION

BANJA LUKA, 2020

INFORMACIJE O MENTORU I DISERTACIJI

Mentor: prof. dr Zoran Jovanović, redovni profesor, Elektrotehnički fakultet Beograd

Naslov doktorske disertacije: SISTEM ZA NADZOR RAČUNARSKE INFRASTRUKTURE KAO SERVISA

Rezime:

Disertacija predstavlja rezultat istraživanja u oblasti nadzora distribuirane i heterogene računarske infrastrukture sa posebnim osvrtom na računarsku infrastrukturu kao servis. Nakon opisa oblasti i pregleda dostupnih rješenja, izvršena je analiza zahtjeva koje mora da ispuni novi sistem za nadzor. Na osnovu podataka prikupljenih u toku VI-SEEM projekta koji je obuhvatao računarsku infrastrukturu u 16 država regiona, definisane su tri oblasti primjene sistema za nadzor: računarska infrastruktura kao servis, grid i računarstvo visokih performansi, te dijeljenje mrežne infrastrukture. Projektovan je i realizovan sistem za nadzor distribuirane heterogene računarske infrastrukture te je testiran za vrijeme trajanja projekta na regionalnoj infrastrukturi. Posebno značajan izlaz istraživanja je novi MIB (eng. *Management Information Base*) dokument koji podržava podatke iz svih analiziranih izvora. Nakon identifikovanja problema degradacije performansi u dijeljenim infrastrukturama projektovan je i sistem za korisnički nadzor dijeljenih infrastruktura u tri prethodno navedene oblasti. Sistem je upoređen sa srodnim rješenjima i izvršena je i analiza sigurnosnih aspekata. Imajući u vidu sve veći značaj i primjenu interneta stvari (IoT), dodatno je izvršena analiza i ove oblasti i dat je opis pristupa koji omogućava unificiran pristup nadzoru svih prethodno opisanih vrsta infrastruktura.

Ključne riječi: distribuirano računarstvo, računarske mreže, infrastruktura kao servis, nadzor, performanse računarskih sistema, SNMP, MIB

Naučna oblast: Tehnološke nauke

Naučno polje: Sistemski inženjering, računarska tehnologija

Klasifikaciona CERIF oznaka: T 120

INFORMATION ABOUT THE MENTOR AND THE DISSERTATION

Mentor: Prof. dr Zoran Jovanović, full professor, Faculty of Electrical Engineering Belgrade

Title of the dissertation: COMPUTER INFRASTRUCTURE AS A SERVICE MONITORING SYSTEM

Summary:

This dissertation is the result of the research in the field of monitoring of distributed and heterogeneous computer infrastructures with focus on infrastructure as a service environments. After the introduction to the field and overview of available technologies, the analysis of requirements for the new monitoring system was performed. On the basis of data collected in VI-SEEM project covering regional e-infrastructure spanning 16 countries, three areas of interest were defined: computer infrastructure as a service, grid and high performance computing, and shared computer networks. A new system for monitoring of distributed and heterogeneous computer infrastructure was designed, implemented and tested during the project on the regional infrastructure. Especially significant output of the research is Management Information Base (MIB) document that supports the information from all the analyzed data sources. After the issue of performance interference in shared infrastructures was identified, a system for client-side monitoring of shared infrastructures in three previously named areas was designed. The system was compared to similar available solutions and security aspects were analyzed. Due to the increasing importance and use of Internet of Things (IoT), an additional analysis of this area was also performed and an approach to unified monitoring of all abovementioned infrastructure types was formulated.

Keywords: distributed computing, computer networks, infrastructure as a service, monitoring, computer system performance, SNMP, MIB

Scientific field: Technological sciences

Research field: Systems engineering, computer technology

Classification CERIF mark: T 120

Mojoj porodici na nesebičnoj podršci i ljubavi

SADRŽAJ

SLIKE

TABELE

SKRAĆENICE

AMQP – Advanced Message Queuing Protocol

ASN.1 – Abstract Syntax Notation One

BDII – Berkeley Database Information Index

CC – Cloud Computing

CMP – Cloud Management Platform

CoAP – Constrained Application Protocol

CoRE – Constrained RESTful Environments

DBaaS – DataBase as a Service

ETSI – European Telecommunications Standards Institute

GOCDB – Grid Operations Centre Data Base

HTTP – Hypertext Transport Protocol

IaaS – Infrastructure as a Service

ICT – Information and Communication Technologies

IDS – Intrusion Detection System

IETF – Internet Engineering Task Force

IoT – Internet of Things

IP – Internet Protocol

IPS – Intrusion Prevention System

ITU – International telecommunication Union

ITU-T – ITU Telecommunication Standardization Sector

JMS – Java Message Service

KVM – Kernel-based Virtual Machine

LB – Logging and Bookkeeping

LDAP – Lightweight Directory Access Protocol

LXC – Linux Containers

M2M – Machine to Machine

MIB – Management Information Base

MONaaS – Monitoring as a Service

MQTT – Message Queueing Telemetry Transport

MQTT-SN – MQTT for Sensor Networks

NAS – Network Attached Storage

NAT – Network Address Translation

NFS – Network File System

NMS – Network Management System

OASIS – Organization for the Advancement of Structured Information Standards

OGF OCCI – Open Grid Forum Open Cloud Computing Interface

OID – Object Identifier

PaaS – Platform as a Service

RAID – Redundant Array of Independent Disks

RBAC – Role Based Access Control

REST – Representational State Transfer

RPC – Remote Procedure Call

RRD – Round-robin Database

SaaS – Software as a Service

SAN – Storage Area Network

SDN – Software Defined Network

SLA – Service Level Agreement

SMI – Structure of Management Information

SNMP – Simple Network Management Protocol

TCP – Transmission Control Protocol

TDBaaS – Time series DataBase as a Service

TLS – Transport Layer Security

UDP – User Datagram Protocol

UUID – Universally Unique Identifier

VM – Virtual Machine

VMM – Virtual Machine Monitor

VPN – Virtual Private Network

WMS – Workload Management System

XDR – External Data Representation

XML – Extensible Markup Language

1. UVOD

Brzim razvojem informacionih tehnologija u prethodnom periodu došli smo u situaciju da iako, makar teoretski, postoji i više nego dovoljno računarskih i drugih srodnih resursa za zadovoljavanje potreba korisnika infrastrukture, usljed neoptimalne raspodjele i upotrebe datih resursa, korisnici nisu u prilici da istu koriste na efikasan način. Dosadašnja praksa bila je da se računarski resursi koriste od strane pojedinaca ili manjih grupa korisnika za uzak skup potreba, bez dijeljenja resursa sa drugim korisnicima. Ovakav način razmišljanja vodi ka neoptimalnoj nabavci novih resursa koji neće biti efikasno iskorišćeni. U cijeloj priči nije zanemariv ni uticaj navedenih resursa i neophodne prateće opreme na potrošnju električne energije i ukupnu cijenu koštanja sistema.

Imajući navedeno u vidu ne iznenađuje činjenica da se u posljednje vrijeme sve više pažnje posvećuje iznalaženju rješenja za problem neefikasne upotrebe računarskih resursa. U oblasti naučno-istraživačkog rada je već godinama prisutan primjer upotrebe grid računarstva kao efikasnog modela dijeljenja resursa u cilju poboljšanja efikasnosti rada. Ideja grid računarstva je da omogući jednostavno, efikasno i sigurno dijeljenje i upotrebu raznorodnih računarskih resursa u formi zasnovanoj na otvorenim standardima na način koji bi eliminisao geografske i administrativne probleme [1]. Slično virtuelnim mašinama, grid tehnologija apstrahuje heterogenost infrastrukture i korisniku nudi pristup uniformnim servisima koji počivaju na stvarnim resursima. Krajnji cilj je univerzalni pristup računarskom resursu sa bilo koje lokacije na jednostavan, siguran, efikasan i nadasve transparentan način.

Sličan pristup, ali sa sopstvenim specifičnostima, je prisutan i kod upotrebe računarstva u oblaku (eng. *Cloud Computing* – CC), odnosno njegovog vida upotrebe infrastrukture kao servisa (eng. *Infrastructure as a Service* – IaaS) [2]. Ovaj pristup, iako je sam naziv nešto novijeg datuma, nije sam po sebi nov i vodi porijeklo još od šezdesetih godina prethodnog vijeka i začetaka višekorisničkih sistema, virtuelizacije i distribuiranog računarstva. Za razliku od grid računarstva u kojem je naglasak na omogućavanju pristupa računarskim resursima visokih performansi za naučno-istraživačke potrebe, kod IaaS pristupa je cilj omogućiti krajnjim korisnicima upotrebu dijeljenih resursa za izgradnju virtuelnih infrastruktura po potrebi. Na ovaj način je moguće postojeće resurse koristiti na efikasniji način uz zadržavanje komoditeta korisnika naviklih na

posjedovanje i ekskluzivnu upotrebu namjenski obezbijeđenih resursa. U skladu sa navedenim, jedna od definicija CC-a je i "skup mrežno povezanih resursa koji omogućuju skalabilnu, personalizovanu, dostupnu računarsku platformu po zahtjevu, pristupačnu na jednostavan i sveprisutan način uz garancije kvaliteta usluga" [2]. Kako je sam koncept doživio procvat u relativno kratkom i skorom vremenskom intervalu, nije iznenađujuće da postoji veliki broj, često međusobno nekompatibilnih, i modela i implementacija CC-a.

Važno je napomenuti da postoji mnogo različitih podjela CC-a od kojih je najčešća prema nivou apstrakcije koji nudi korisnicima. Na najnižem nivou se nalazi upotreba infrastrukture kao servisa. Ovaj nivo obezbjeđuje pristup fizičkim računarskim resursima poput servera i mrežne opreme, uz neophodne softverske komponente, u vidu servisa. Na ovaj način je omogućeno korisnicima da po potrebi kreiraju nove (virtuelne) računarske resurse, koji se nakon određenog vremena mogu proširiti, arhivirati, premjestiti na druge fizičke lokacije, ili uništiti ukoliko više nisu potrebni, bez ikakvih intervencija na postojećoj opremi.

Na višem nivou se nalazi upotreba platforme kao servisa (eng. *Platform as a Service* – PaaS). Ovaj koncept proširuje ponudu IaaS-a obezbjeđivanjem i unaprijed konfigurisane softverske platforme, na primjer u vidu skupa servisa, biblioteka ili baza podataka, koja služi kao osnova za razvoj i izvršavanje korisničkih aplikacija. Na ovaj način se u velikoj mjeri pojednostavljuje sam proces podrške razvoju i izvršavanju aplikacija jer se razvojnim timovima nudi standardizovana platforma za rad u svim dijelovima životnog ciklusa aplikacije.

Na vrhu ove podjele se nalazi upotreba softvera kao servisa (eng. *Software as a Service* – SaaS). Ideja je da se korisnicima ponudi predefinisani skup aplikacija koje je moguće na odgovarajući način povezati, konfigurisati i koristiti, bez potrebe za dodatnim programiranjem. Sam softver koji se nudi na ovaj način je po prirodi stvari vrlo fleksibilan i relativno jednostavno proširiv.

Postoji i pristup "bilo šta" kao servis (eng. *X as a Service* – XaaS) ali je sam koncept toliko širok da je od male praktične vrijednosti za ovo razmatranje. Jedna od glavnih karakteristika svih navedenih modela je tarifiranje koje se vrši na osnovu realno i transparentno izmjerene upotrebe servisa što može dovesti do bitno manjih troškova poslovanja ili većeg nivoa efikasnosti u radu. Iako su navedeni koncepti površno posmatrano jednostavni, sama realizacija najčešće uključuje veći broj komponenti koje moraju ispravno i pouzdano funkcionisati. Sama činjenica da se i

najjednostavnije infrastrukture – infrastrukture kao servis – sastoje minimalno od sistema za upravljanje identitetom, sistema za upravljanje pravima pristupa, sistemima za virtuelizaciju računarskih i mrežnih resursa, sistemima za pohranu podataka, sistemima za bilježenje upotrebe, itd, sugeriše da sam posao održavanja ovakvog sistema nije jednostavan posao [3].

1.1. FORMULACIJA PROBLEMA

Jedan od osnovnih preduslova za pravilno funkcionisanje ovako kompleksnih sistema je postojanje kvalitetnog sistema za nadzor upotrebe i pravilnog funkcionisanja dostupnih resursa. Kao što je ranije u tekstu navedeno, postoji veliki broj različitih praktičnih realizacija navedenih platformi za virtualizaciju, što je rezultovalo postojanjem većeg broja sistema za nadzor. Indikativno je da većina sistema za nadzor podržava samo određenu tehnologiju uz relativno malo pažnje posvećene integraciji u postojeće sisteme ili specifičnosti nadzora fizičkih uređaja. Na primjer, evidentno je da postojeći sistemi gotovo da nemaju podršku za najrasprostranjeniji standard za nadzor i upravljanje mrežnim uređajima (eng. *Simple Network Management Protocol* SNMP) [4], iako je to osnovni standard za nadzor podržan od strane praktično svih mrežnih uređaja. Razlog vjerovatno leži u i činjenici da formalizam i postojanje čvrstih standarda čine SNMP mnogo kompleksnijim u poređenju sa popularnim, fleksibilnim i jednostavnijim REST [5] API baziranim pristupima. Međutim, ukoliko želimo na uniforman način da sprovodimo nadzor heterogene, kompleksne infrastrukture, brzo ćemo se suočiti sa problemom međusobne nekompatibilnosti API-ja različitih platformi.

Sama organizacija računarske infrastrukture kao servisa se zasniva na upotrebi virtuelizovanih resursa koji se izvršavaju na realnim računarskim i mrežnim resursima. Kako je riječ o dijeljenom i dinamičkom okruženju, postoje brojni problemi pri pouzdanom identifikovanju komponenti sistema koje predstavljaju "uska grla" ili na drugi način degradiraju performanse kompletnog sistema. Jedan od jednostavnijih, ali ilustrativnih, problema je nemogućnost korisnika infrastrukture da ima uvid u realno iskorišćenje servera na kom se trenutno izvršava njegova virtuelna mašina [6]. Naime, iako korisniku virtuelna mašina daje informaciju da je virtuelni resurs slobodan, u realnom serveru je neka druga virtuelna mašina zauzela značajnu količinu resursa i na taj način smanjila performanse prve virtuelne mašine. Problem leži u činjenici da korisnicima nije

dozvoljeno, sa dobrim razlogom, da direktno pristupaju stvarnim podacima ni fizičkog servera, ni drugih virtuelnih mašina koje se izvršavaju na istom serveru i sa kojima dijele resurse.

Poseban problem predstavlja upotreba infrastrukture kao servisa u kompleksnim sistemima gdje se na višim nivoima koriste druge, same po sebi kompleksne, tehnologije kao što je grid računarstvo. Trend virtualizacije grid resursa je postao vrlo popularan u proteklom periodu i vjerovatno predstavlja budućnost grid računarstva u vidu skupa standardizovanih okruženja, slično platformi kao servisu, gdje grid segment predstavlja platformu kao servis. Imajući u vidu specifičnosti grid računarstva, kao i velike zahtjeve krajnjih korisnika za računarskim resursima visokih performansi, neophodno je obezbijediti nadzor svih relevantnih slojeva infrastrukture u cilju otkrivanja problema u funkcionisanju sistema. Trenutno postoje kvalitetno realizovane komponente za nadzor fizičkog [7] i grid servisnog segmenta [8], uz vrlo problematičan nedostatak standardizovanog sistema koji obezbjeđuje vezu između njih.

U trenutnoj situaciji se naročito ističu problemi kvalitetnog nadzora i fizičkih i virtuelizovanih resursa, problemi nadzora heterogenih sistema, kao i problem omogućavanja nadzora fizičkih uređaja od strane korisnika virtuelizovanih resursa.

1.2. CILJ I DOPRINOS ISTRAŽIVANJA

Osnovna hipoteza ove disertacije je:

Moguće je projektovati i realizovati efikasan sistem za nadzor računarske infrastrukture kao servisa, koja na uniforman način putem SNMP protokola omogućuje korisnicima uvid u stanje i fizičkih i virtuelnih elemenata infrastrukture.

Osnovni cilj disertacije je verifikacija postavljene hipoteze analizom, projektovanjem i realizacijom sistema za nadzor računarske infrastrukture kao servisa koji zadovoljava postavljene zahtjeve, što uključuje:

- Pregled dostupnih platformi koje omogućavaju upotrebu računarske infrastrukture kao servisa.
- Pregled relevantne literature u oblasti nadzora računarske infrastrukture kao servisa.

- Analizu dostupnih rješenja za nadzor računarske infrastrukture kao servisa uključujući i nadzor nižih slojeva infrastrukture.
- Analizu potreba i definisanje korisničkih zahtjeva za nadzor računarske infrastrukture kao servisa.
- Projektovanje i realizaciju sistema za nadzor računarske infrastrukture kao servisa.
- Verifikaciju realizovanog rješenja na realnoj infrastrukturi.

Očekivani doprinosi ove disertacije su:

- Analiza uticaja dijeljenih virtuelnih okruženja na performanse korisničkih virtuelnih resursa.
- Analiza potreba sistema za nadzor hibridnih, odnosno privatno-javnih, infrastruktura sa stanovišta korisnika infrastrukture.
- Analiza mogućnosti upotrebe SNMP standarda u navedenom sistemu za nadzor.
- Prijedlog realizacije sistema za nadzor koji uzima u obzir ranije navedene stavke.

1.3. PREGLED I SADRŽAJ RADA

Nakon uvoda i formulacije problema, slijedi poglavlje u kom su dati pregled dijeljenih infrastruktura i sistema za nadzor istih. Ovo poglavlje sadrži opise tri osnovne vrste dijeljenih infrastruktura od interesa za ovaj rad, te dodatnu analizu okruženja u oblasti interneta stvari. Poglavlje sadrži i pregled sistema za nadzor koji se mogu koristiti u okviru opisanih infrastruktura. U nastavku teksta je dat detaljniji opis SNMP (eng. *Simple Network Management Protocol*) standarda. U poglavlju su pored uvodnih razmatranja, dati i kratka istorija razvoja standarda, arhitektura standardnog sistema koji koristi SNMP, opis MIB (eng. *Management Information Base*) dokumenata, te je dat prikaz načina za proširenje funkcionalnosti SNMP agenta.

U trećem poglavlju je dat prikaz sistema za nadzor heterogene dijeljene računarske infrastrukture. Nakon uvoda slijedi opis regionalne infrastrukture koja je iskorišćena kao poligon za razvoj i testiranje sistema. Analizirani su izvori podataka podijeljeni u tri osnovne kategorije uz detaljan prikaz svakog od izvora. Slijedi prikaz organizacije podataka razvijene za potrebe sistema, te opis arhitekture razvijenog sistema za nadzor. Praktična realizacija sistema je opisana u narednoj

sekciji koja sadrži i opis svih komponenti sistema. Kako je jedan od osnovnih zahtjeva za sistem bio fleksibilnost, prikazan je i način alternativne upotrebe sistema kao veb baziranog portala testiranog na Horizon 2020 projektu VI-SEEM [9]. Poglavlje završava analizom performansi sistema u SNMP AgentX [10] režimu rada.

Četvrto poglavlje se bavi problemom degradacije performansi u dijeljenim infrastrukturama. Posebna pažnja je posvećena nadzoru infrastrukture sa korisničke strane uz opise tri osnovne vrste dijeljenih infrastruktura koje su analizirane u ovom radu. Dat je prikaz predloženog rješenja, analiza primjene za sve tri vrste infrastruktura, opis arhitekture sistema, analiziran je problem prikupljanja i prezentovanja podataka, dat je pregled sigurnosnih aspekata, kao i poređenje sa drugim srodnim rješenjima. Poglavlje završava eksperimentalnom validacijom realizovanog rješenja.

U petom poglavlju je dat primjer primjene sistema u oblasti interneta stvari uz opis specifičnosti datog okruženja. Dat je prikaz arhitekture sistema, načina komunikacije sa IoT uređajima i integracije u postojeće sisteme za nadzor.

Šesto poglavlje predstavlja zaključak u kom je dat osvrt na izloženi rad i doprinose disertacije, te su navedene smjernice za buduća istraživanja u oblasti.

2. PREGLED DIJELJENIH INFRASTRUKTURA I SISTEMA ZA NADZOR

Dijeljene infrastrukture su unijele revoluciju u moderni pristup računarstvu i umrežavanju računarskih resursa. Njihova upotreba je s jedne strane omogućila operaterima infrastruktura bolju iskorišćenost resursa, dok su korisnici sa druge strane dobili novu fleksibilnost mogućnosti definisanja iznajmljene infrastrukture po potrebi što dovodi do jednostavnijeg skaliranja i prilagođavanja dinamičnim uslovima modernog poslovanja [11]. Diaz, Martin i Rubio su proveli istraživanje u kojem su jasno identifikovali CC i IoT kao dva glavna nosioca budućeg razvoja u oblasti ICT-a [12]. Pregled dostupne literature u oblasti Industrije 4.0 proveden od strane Lua i saradnika je označio CC kao jedan od stubova na kojima počiva četvrta industrijska revolucija [13]. Sa povećanjem kompleksnosti i raznolikosti dostupnih tehnologija na kojima se zasniva ova revolucija, uloga jednostavnog i efikasnog operativnog pristupa od strane korisnika je postala od kritične važnosti. Klijentima je omogućena upotreba naprednih alata pomoću kojih mogu iskoristiti brojne prednosti novih tehnologija, uključujući i rješenja za nadzor infrastrukture. Ova rješenja se kreću u širokom spektru, od alata koji su sastavni dio ponuđenog proizvoda do eksternih sistema koji su ciljano namijenjeni za nadzor dijeljenih infrastruktura [14]. S druge strane, operateri infrastruktura imaju puni pristup svim relevantnim mjernim podacima vezanim za infrastrukturu, ali im je onemogućen puni uvid u korisničko okruženje jer je neophodno poštovati ograničenja proistekla iz prava privatnosti korisnika i povjerenja ključnog za uspjeh na otvorenom tržištu usluga.

2.1. INFRASTRUKTURA KAO SERVIS

Iako na prvi pogled mogu izgledati slično, razlika između virtuelizacije i IaaS je suštinska. Virtuelizacija prestavlja tehnologiju pomoću koje možemo kreirati efikasne izolovane kopije sistema koje se izvršavaju na jednom serveru, dok je ideja IaaS da budu okruženja koja omogućavaju apstrakciju, objedinjavanje, te dijeljenje računarskih resursa putem mrežne infrastrukture. Virtuelizacija predstavlja samo jednu važnu tehnologiju u upotrebi u okviru IaaS okruženja.

IaaS okruženja mogu biti bazirana na hardverskim, virtuelizovanim ili kontejnerskim resursima pomoću kojih je omogućeno dijeljenje lokalnih resursa putem mrežnih infrastruktura [2]. Posmatrano na ovaj način, na najnižem nivou se nalaze računarske mreže i serveri koje povezuju, iznad njih su sistemi za virtuelizaciju dok se na vrhu nalaze okruženja koja omogućavaju apstrakciju, objedinjavanje i dijeljenje resursa sa nižih nivoa na takav način da je moguća njihova efikasna upotreba od strane drugih entiteta na takav način da su obezbjeđivanje i skaliranje novih resursa brzi i jednostavni.

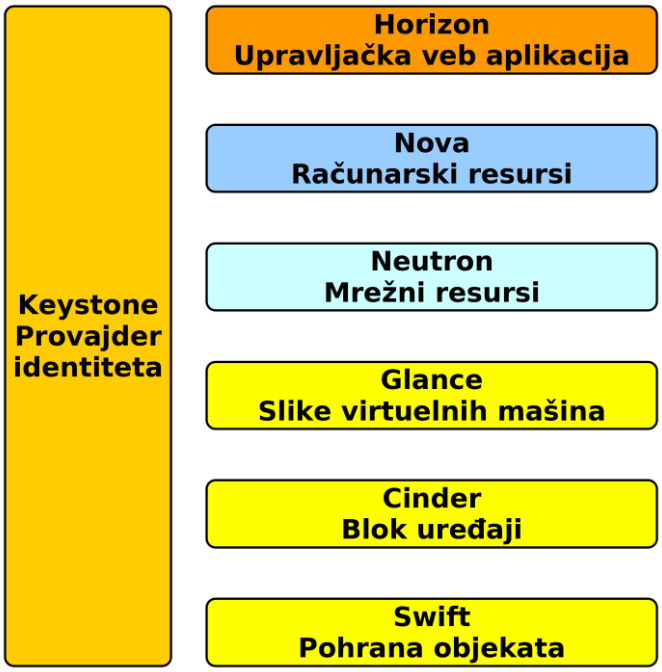
Još jedna ključna razlika se tiče načina upotrebe dodijeljenih resursa. Kod virtuelizacije je očekivana relativno dugoročna upotreba resursa koji skaliraju vertikalno, dok je u IaaS okruženju svaka pojedinačna instanca efemerne prirode, a skaliranje se vrži horizontalno dodavanjem novih instanci.

U nastavku poglavlje će biti dat opis trenutno najpopularnijih IaaS platformi otvorenog koda od interesa za oblast istraživanja obuhvaćenu ovim radom.

2.1.1. OPENSTACK

OpenStack [15] je besplatna softverska platforma otvorenog koda prvenstveno namijenjena upravljanju IaaS okruženjima i danas predstavlja najpopularniju platformu te vrste. Iako je OpenStack nastao kao proizvod saradnje NASA-e i RackSpace Hosting-a, danas je upravljanje u rukama neprofitne organizacije sa preko 500 članova pod nazivom OpenStack fondacija [16]. Princip razvoja je zasnovan na brzoj iteraciji rješenja što za posljedicu ima izdavanje nove verzije svakih šest mjeseci. Verzije osim numeričkih oznaka nose i imena koja počinju slovom u engleskom alfabetu koje odgovara verziji (npr. Austin, Bexar, Cactus, .., Rocky, Stein, Train).

Sam OpenStack se sastoji od velikog broja komponenti, trenutno 21, koje pokrivaju gotovo sve potrebe velikog broja korisnika IaaS platformi. Iako takav broj komponenti može izgledati zastrašujuće, za osnovni rad je potrebno svega nekoliko komponenti koje su prikazane na slici 2.1. i opisane u nastavku teksta [3].



Slika 2.1. Osnovne komponente OpenStack platforme

Komponenta sa kojom korisnici imaju naviše dodira je veb bazirani portal za upravljanje infrastrukturama *Horizon*. Komponenta pod nazivom *Nova* je zadužena za upravljanje hipervizorima koji se izvršavaju na serverskim čvorovima i obezbjeđuju računarske resurse za korisničke infrastrukture. Ova komponenta predstavlja srce sistema jer upravlja i rezervacijama i zauzećem svih dijeljenih resursa na nivou platforme. Nekada je *Nova* bila zadužena i za mrežne resurse, ali u modernim verzijama je uznapredovala dovoljno da zasluži posebnu komponentu imena *Neutron*. Rad sa mrežama je izuzetno fleksibilan i korisnicima je omogućeno da kreiraju proizvoljno kompleksne virtuelne mrežne topologije, dok administratori imaju mogućnost upotrebe VLAN-ova, SDN-ova, povezivanja sa IDS-ovima, VPN-ovima, itd. Virtuelne mašine koriste slike predefinisanih serverskih instalacija koje su dio usluga komponente pod imenom *Glance*. Upotreba ove komponente omogućava i migraciju VM između hipervizora bez zaustavljanja izvršavanja iste. Slike i drugi podaci se čuvaju u sistemu za pohranu objekata nazvanom *Swift*. *Swift* predstavlja skalabilno i redundantno skladište podataka koje replicira sve unose na višestruke uređaje i servere širom infrastrukture što omogućava upotrebu ekonomski isplativijeg hardvera jer su redundantnost i provjera integriteta podignuti na viši nivo. Ukoliko korisnik ima potrebu za brzim blok baziranim sistemom za pohranu podataka koristi se

komponenta *Cinder*. Ova komponenta podržava rad sa velikim brojem različitih platformi, od lokalnih fajlova do upotrebe velikih distribuiranih sistema kao što su Ceph, GlusterFS, GPFS, itd. Servis koji povezuje sve navedene komponente i koji pruža uslugu provajdera identiteta svim ostalim komponentama sistema je *Keystone.* Osim ove funkcionalnosti, servis održava i listu svih registrovanih servisa u okviru konkretne OpenStack instalacije.

OpenStack koristi sopstveni API [17] za sve operacije i ne pokušava da prati druge API-je ili standarde, iako postoje napori da se implementira podrška pristup OpenStack infrastrukturama putem drugih API-ja. Vrijedi napomenuti da postoje i obrnuta nastojanja da se OpenStack API koristi za sisteme koji nisu dio OpenStack familije servisa, kao što je Synnefo [18] koji koristi OpenStack API prema korisnicima iako je sama platforma bazirana na Ganeti klasterima [19]. Imajući u vidu da OpenStack nije jedinstveni proizvod nego skup od preko dvadeset samostalnih komponenti, razumljivo je da je sam proces instalacije i konfiguracije platforme komplikovan. Kako je želja svih uključenih strana da se olakša rad sa OpenStack platformom, pojavio se veliki broj alata namijenjenih instalaciji i konfiguracije potrebnih komponenti, od podrške za popularne alate za automatizaciju poput Chef [20], Puppet [21] i Juju [22], do namjenski kreiranih sistema kao što je Fuel [23]. Brz razvoj novih verzija, relativno kratak period podrške za svaku verziju i kompleksan proces prelaska na novije verzije su uslovili razvoj komercijalnih distribucija OpenStack platforme koje omogućavaju komercijalnim entitetima da OpenStack koriste po istim principima koji vrijede za generički poslovni softver. Važno je napomenuti da postoji i veliki broj upravljačkih programa koji omogućavaju integraciju OpenStack platforme sa komercijalnim proizvodima poput VMWare [24] ili IBM [25] rješenja.

Iako je sam OpenStack skup relativno nezavisnih projekata, većina koda je pisana u programskom jeziku Python [26] dok se za komunikaciju između komponenata koristi RabbitMQ [27] a kao server baze podataka MySQL [28] ili kompatibilno rješenje (MariaDB [29], Galera [30], itd).

2.1.2. OPENNEBULA

OpenNebula [31] je platforma za upravljanje heterogenim i distribuiranim računarskim IaaS okruženjima. Može da se koristi i kao rješenje za virtuelizaciju na nivou računarskog centra

iako je mnogo češći vid upotrebe puno IaaS okruženje. Predstavlja besplatan softver otvorenog koda i distribuirana je pod Apache licencom verzije 2. Iako je razvijena kao istraživački projekat, danas razvojem upravlja kompanija OpenNebula Systems koja nudi brojne komercijalne dodatke namijenjene poslovnim korisnicima.

Za razliku od OpenStack platforme, OpenNebula je jedinstveni sistem koji prati klasičnu klastersku arhitekturu sa čvorom namijenjenim za komunikaciju sa korisnikom i upravljanjem rada klastera hipervizora koji koriste neki od sistema za pohranu podataka, pri čemu su sve komponente povezane mrežom [32]. *Front-end* čvor osim veb portala takođe predstavlja i sistem za upravljanje, raspoređivanje i orkestraciju resursa i virtuelnih mašina, a zadužen je i za održavanje korisničkih virtuelnih mreža. Iako se formalno svi navedeni servisi vode na istom čvoru, u praktičnoj realizaciji je moguća instalacija na više servera prema potrebama operatera. Što se tiče izvršavanja virtuelnih mašina, OpenNebula podržava upotrebu KVM [33] i LXD [34] tehnologija, kao i komercijalnih rješenja poput VMWare [24] proizvoda, pri čemu serveri mogu biti grupisani u OpenNebula klastere sa ciljem jednostavnijeg i efikasnijeg rada. Kada govorimo o sistemu za pohranu podataka, OpenNebula podržava rad sa tri vrste skladišta podataka: sistemska skladišta koja čuvaju slike VM koje se trenutno izvršavaju, skladišta slika koja predstavljaju repozitorijum slika serverskih instalacija, te skladište fajlova za manje fajlove opšte namjene koji ne predstavljaju virtuelne diskove.

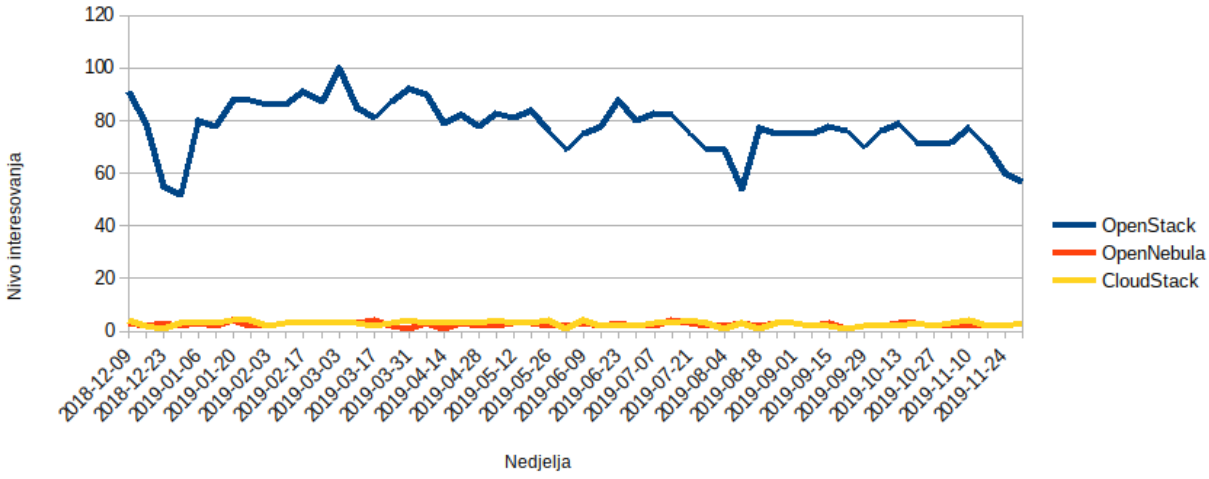
OpenNebula ima sopstveni API ali takođe podržava i druge načine pristupa kao što su Amazon EC2 [35] i OGF OCCI [36] što u velikoj mjeri pojednostavljuje rad sa hibridnim IaaS okruženjima jer je moguće koristiti isti API i alate za rad sa javnim i privatnim IaaS infrastrukturama. Dominantno je pisana upotrebom C++ i Ruby programskih jezika iako se u pojedinim komponentama koriste i brojni drugi jezici. Za komunikaciju između komponenti se koristi XML-RPC i RabbitMQ dok je kao server baze podataka odabran SQLite za manje instalacije i MySQL kao preporučeno rješenje.

2.1.3. CLOUDSTACK

CloudStack [37] je IaaS platforma koja je od početka projektovana na takav način da bude nezavisna od konkretnog hipervizora, da obezbijedi ugrađenu visoku dostupnost servisa i

virtuelnih mašina, te da bude otvorena za upotrebu različitima API-jima. Imajući navedeno u vidu nije čudno što CloudStack podržava širok spektar hipervizora, od LXC [34] kontejnera i KVM-a [33], preko XenServer [38] i VMWare [24] rješenja do Microsoft Hyper-V [39] sistema. Kada govorimo o API-ju, pored sopstvenog, CloudStack podržava i rad sa AWS (eng. *Amazon Web Services*) [35] kompatibilan API kao i OGF OCCI [36] interfejsom. Projekat je pisan dominantno u programskom jeziku Java i koristi RabbitMQ [27] za razmjenu poruka, te MySQL [28] kao server baze podataka.

Pored navedenih platformi postoji i veliki broj drugih projekata koji su u fazi mirovanja poput Eucalyptus [40] platforme, zasnovani su na ili su blisko vezani za neku od opisanih platformi (Rackspace [41], Mirantis [42]) ili više odgovaraju virtuelizacijskim projektima na nivou klastera (oVirt [43], OpenQRM [44], Proxmox [45], Ganeti [19]). Čak i pri navođenju ove tri platforme treba voditi računa o tome da je OpenStack mnogo popularnija platforma o čemu svjedoči i analiza Google trendova za prethodnih 12 mjeseci prikazana na slici 2.2.



Slika 2.2. Analiza Google trendova za OpenStack, OpenNebula i CloudStack IaaS platforme

Pregled oblasti proveden od strane kompanije Flexera [46] je obuhvatio analizu podataka prikupljene od 786 stručnjaka iz oblasti CC-a iz različitih vrsta kompanija i organizacija, ali dominantno iz regiona Sjeverne Amerike. U izvještaju je navedeno da je OpenStack trenutno u aktivnoj upotrebi u 28% kompanija, da još 13% testira upotrebu dok 8% ima planove za upotrebu u budućnosti. Upotreba CloudStack platforme je prisutna kod 14% ispitanika, 9% je trenutno

testira, dok 5% ima planove za upotrebu u budućnosti. Ako se ograničimo samo na poslovne korisnike, OpenStack je prisutan kod 36% ispitanika, uz testiranje kod 15% i planove za budućnost kod 6% kompanija, dok su vrijednosti za CloudStack 20%, 9% i 5% respektivno.

2.1.4. SISTEMI ZA NADZOR

Svaka od navedenih platformi posjeduje sopstveni sistem za nadzor dijeljenih resursa sa fokusom na prikupljanje i analizu podataka o upotrebi resursa, dok je nadzor nižih nivoa infrastrukture, te otkrivanje problema u tim okvirima ostavljen drugim generičkim alatima za nadzor.

Komponenta OpenStack platforme zadužena za prikupljanje podataka o upotrebi resursa, njihovo skladištenje i analizu, te djelovanje prema zadatim pravilima se naziva *Telemetry* [47]. Kako su zahtjevi postavljeni pred ovu komponentu izuzetno kompleksni i raznovrsni, sama komponenta se sastoji od tri odvojena projekta:

- *Ceilometer* – efikasno prikupljanje, normalizacija i obrada podataka prikupljenih od strane svih aktivnih OpenStack servisa.
- *Aodh* – omogućava automatizaciju postupanja po podacima ili dešavanjima prikupljenim od strane *Ceilometer* komponente.
- *Panko* – servis zadužen za kratkoročno i dugoročno pohranjivanje, indeksiranje i pretraživanje podataka krajnjih korisnika za potrebe naknadne kontrole i otkrivanja problema.

Osim navedenih projekata, veliki broj i OpenStack i eksternih servisa koristi podatke prikupljene od strane *Ceilometer*-a, kao što su *Horizon* za veb pristup sistemu, *Heat* za orkestraciju resursa, *Gnocchi* kao TDBaaS komponenta, *CloudKitty* za povezivanje podataka iz *Ceilometer*-a sa obračunom troškova, *Graphite Publisher* kao veza prema *Graphite* sistemu sa grafičko predstavljanje podataka, itd. Važno je napomenuti da su mnogi od projekata ili komponenti u fazi intenzivnog razvoja (*CloudKitty*), fazi reimplementacije (*Gnocchi*) ili pate od dugačkih perioda neaktivnosti (*Graphite Publisher*).

OpenNebula posjeduje sistem za nadzor koji kombinuje sisteme za prikupljanje podataka o statusu servera i virtuelnih mašina, kao i podatke o utrošku virtuelnih resursa. Prikupljanje podataka je realizovano upotrebom *collectd* [48] servisa koji putem UDP protokola prikuplja podatke sa svih trenutno aktivnih instanci u klasteru. Iniciranje prikupljanja podataka se vrši upotrebom *ssh* pristupa čvorovima klastera i pokretanjem beskonačne petlje izvršavanja testova i slanja rezultata na centralni *collectd* servis. Kako je komunikacija zasnovana na UDP protokolu moguće je ostvariti visoke performanse uz minimalan dodatni utrošak resursa ali po cijenu nepouzdanog prenosa podataka jer ne postoji garancija da će svaki mjerni podatak biti primljen i obrađen. Podrazumijevane nadzirane veličine su upotreba procesorskih, memorijskih i resursa za pohranu podataka.

Slično prethodno opisanim rješenjima, i CloudStack podržava rad sa standardnim kontrolnim pločama dostupnim administratorima i korisnicima, praćenje upotrebe resursa, ali i određeni stepen integracije u postojeće sisteme za nadzor kroz podršku za slanje poruka o događajima ili alarmima eksternim *Syslog* i SNMP servisima. Na ovaj način je moguće jednostavno uključiti taj vid podataka o nadzoru CloudStack infrastrukture u NMS i integrisati podatke sa mjerenjima dobijenim na druge načine.

Generički sistemi za nadzor umreženih uređaja, kao što su *Zabbix* [49], *Zenoss* [50] i *Nagios* [7], takođe mogu biti korišćeni za nadzor IaaS infrastruktura na višem nivou. Testovi koji inače prikupljaju podatke o zauzeću procesora ili mrežnom protoku na fizičkom uređaju su adaptirani da podatke prikupljaju iz API-ja IaaS okruženja i mogu biti korišćeni kao izvor sumarnih podataka na nivou klastera ili kompletne distribuirane infrastrukture. U studiji [51] koja je obuhvatila 12 opštih i 9 namjenskih IaaS sistema za nadzor izdvojeni su *Zabbix*, *Hyperic* [52] i *NimSoft* [53] kao rješenja koja su u datom trenutku ispunjavala najveći broj zahtjeva koje su autori studije definisali. Najčešće problem za analizirane sisteme je bilo postojanje centralnog servisa koji je ograničavao skaliranje sistema, ali i kreirao problem kritične tačke bez koje nije bilo moguće pravilno funkcionisanje sistema u cjelini [51]. Još jedan od izazova za sisteme za nadzor u IaaS platformama je i relativno slaba podrška za federacije, SLA i nadzor na višim nivoima infrastrukture [54]. Dodatni problem nastaje i kada je potrebno povezati podatke dobijene na višim nivoima infrastruktura sa mjernim podacima sa fizičkih uređaja i otkriti probleme vezane za degradaciju performansi o čemu će više riječi biti u poglavlju 4.

Interesantan pristup su odabrali autori *Monasca* sistema [55]. *Monasca* je, iako dio OpenStack projekata, relativno nezavisna od samog OpenStack okruženja i predstavlja MONaaS (eng. *Monitoring as a Service*) sistem namijenjen uopštenom nadzoru ali koji je moguće koristiti i za nadzor IaaS okruženja. *Monasca* ima prilično kompleksnu strukturu i sastoji se od 14 međusobno povezanih komponenti. Pretpostavka je takođe da se zbog kompleksnosti sistema vrši i nadzor svih komponenti sistema, po mogućnosti od strane eksternog servisa ili dodatne instance *Monasca* sistema namijenjene samo za nadzor osnovne instance, koja zauzvrat radi nadzor dodatne instance. Osnovne komponente su *Monasca Agent* koji prikuplja podatke (metrike, statistike i alarme) i publikuje ih pomoću *Monasca API* REST servisa koji kroz *Kafka* sistem za razmjenu poruka vrši njihovu distribuciju do svih potrebnih komponenti. Obrađeni podaci se čuvaju u TSDB poput InfluxDB ili *Cassandra* servisa u formatu pogodnom za daljnju analizu i vizualizaciju.

2.2. GRID I RAČUNARSTVO VISOKIH PERFORMANSI

Grid računarstvo predstavlja formu distribuiranog računarstva zasnovanog na standardima, otvorenim protokolima i interfejsima opšte namjene sa ciljem koordinisanja distribuiranih resursa i ostvarivanja kvaliteta usluga [1]. Osnovu grid računarstva najbolje opisuje pet ideja na kojima počiva: dijeljenje resursa na globalnom nivou, siguran pristup infrastrukturi, efikasna i uravnotežena upotreba resursa, smrt udaljenosti koja podrazumijeva jednak pristup bez obzira na to da li je servis lokalni ili udaljeni, te upotreba otvorenih tehnologija i standarda.

Kako su problemi koje rješava grid vrlo slični problemima sa kojima se susreću i federacije klastera visokih performansi, prirodno je bilo integrisati klastere u grid infrastrukture ili napraviti nove specifične grid bazirane infrastrukture namijenjene olakšanju rada sa klasterima visokih performansi.

U oba slučaja se radi o skupu centralnih servisa koji obezbjeđuju uniforman pristup distribuiranim resursima sakrivajući kompleksnost nižih slojeva infrastrukture od korisnika, servisa i aplikacija upotrebom midlvera i virtuelnih organizacija [56]. Sam sloj midlvera je najvažnija komponenta grid arhitekture i definiše logiku njenog rada upotrebom standardizovanih softverskih komponenti, biblioteka i protokola. Najpopularnija grid okruženja i alati su Globus [57], ARC [58], UNICORE [59] i EMI [60].

2.2.1. SISTEMI ZA NADZOR

Slično kao i u slučaju sistema za nadzor infrastruktura kao servisa, i u ovom slučaju je potrebno razdvojiti generičke sisteme za nadzor od namjenski projektovanih sistema koji uzimaju u obzir specifičnosti okruženja. Iako je i u ovom okruženju moguće koristiti generičke alate, oni su obično samo komponente u okviru kompleksnijih sistema za nadzor i njima se tipično delegira nadzor samo jednog on nižih slojeva infrastrukture ili se koriste samo kao izvršioci konkretnih test slučajeva dok se rezultati obrađuju na višim nivoima. Od sistema za nadzor Grid/HPC infrastruktura na višem nivou potrebno je izdvojiti Ganglia [61] i ARGO [8] sisteme.

Ganglia je distribuirani sistem za nadzor Grid-HPC infrastruktura u čijem je dizajnu posebna pažnja posvećena problemu skalabilnosti. Iako je u prvim verzijama naglasak bio na nadzoru na nivou klastera, razvojem grid računarstva i integracijom sa Globus skupom alata, Ganglia je evoluirala u sistem koji podržava nadzor generičkih distribuiranih računarskih okruženja na širokom spektru operativnih sistema. Umjesto kreiranja novih rješenja, autori Ganglia sistema su odabrali da iskoriste *open-source* prirodu Ganglia-e i srodnih projekata te za prenos podataka koriste XDR [62] a za skladištenje RRDtool [63]. Skalabilnost je ostvarena kroz dvije glavne karakteristike Ganglia sistema: niski troškovi rada na nivou čvora i upotreba *multicast* baziranog komunikacionog protokola. Osnovna arhitektura sistema je hijerarhijska i sastoji se od *gmond* procesa koji se odvijaju na nadziranim čvorovima i koji putem XDR/UDP šalju podatke *gmetad* čvorovima koji koriste XML i TCP za komunikaciju sa *gmetad* čvorovima na višim nivoima. Na ovaj način je omogućen i nadzor pojedinačnih servera u okviru klastera kao i klastera o okviru federacije.

ARGO je sistem za funkcionalni nadzor servisa namijenjen za upotrebu u istraživačkim infrastrukturama. Sama arhitektura sistema je slojevite prirode i sastoji se od skupa komponenti povezanih sistemom za razmjenu poruka. Sistem osim izvršavanja testova i praćenja statusa servisa podržava i rad sa deklarisanim periodima nedostupnosti, te dobavljanja topologije nadzirane infrastrukture iz eksternih izvora. Sam ARGO izvršava testove upotrebom Nagios sistema koji je automatski konfigurisan na osnovu podataka dobijenih od centralnih servisa. Sami testovi mogu imati različitu formu, od vrlo jednostavnih testova mrežne dostupnosti (npr. ping), preko ispitivanja važnosti X.509 sertifikata (npr. HTTPS) do kompleksnih testova koji se sastoje

od više faza (npr. kreiranje, pokretanje i uništavanje instance virtuelne mašine). Rezultati individualnih testova se čuvaju u bazi podataka i koriste za napredna izračunavanja na višim nivoima.

Najvažnija funkcionalnost ARGO sistema je vezana za izračunavanje dostupnosti i pouzdanosti svakog nadziranog servisa i krajnje tačke uz mogućnost prezentovanja izračunatih vrijednosti putem REST API-ja ili veb baziranog portala. Više informacija o upotrebi ARGO sistema je dostupno u poglavlju 3.2.5.

2.3. RAČUNARSKE MREŽNE INFRASTRUKTURE

Računarske mrežne infrastrukture su izuzetno dobro istražena i opisana oblast čija detaljnija analiza izlazi van okvira ovog rada. Kako računarske mreže predstavljaju najniži infrastrukturni sloj od interesa za ovaj rad, ipak je neophodno posvetiti zasluženu pažnju problemima koji se mogu reflektovati na više slojeve, ali i savremenim trendovima u ovoj oblasti. Problemi na mrežnom sloju kompleksne infrastrukture mogu biti izuzetno problematični za detekciju i pravilno lociranje, naročito kada je riječ o degradaciji performansi uzrokovanoj dijeljenom prirodom mrežnih infrastruktura što je detaljnije obrađeno u poglavljima 4.1.1. i 4.2.1.

Kada je riječ o samim tehnologijama u upotrebi u savremenim računarskim mrežama sa stanovišta ovog rada, iako postoji veliki spektar mogućnosti, tržište se u velikoj mjeri standardizovalo oko upotrebe Ethernet i IP mreža ili kompatibilnih rješenja za veliku većinu namjena. Brzim razvojem optičkih komunikacionih rješenja i padom cijena potrebne opreme i infrastrukture došli smo u situaciju u kojoj je moguće koristiti istu tehnologiju, čak i uređaje, i za lokalne mreže i za međudržavne linkove, kao što je slučaj u vezi Akademske mreže Republike Srpske na Akademsku mrežu Srbije i dalje u svijet. Standardizacija je donijela i pojednostavljenje nadzora opreme jer je moguće formirati jedinstveni model za cijelu klasu uređaja i pripadajućih interfejsa i na taj način olakšati rad i sistemima za nadzor i njihovim korisnicima.

Još jedan važan trend je sve češća upotreba centralizovanog sistema upravljanja korišćenjem softverski definisanih mreža (*Software Defined Network* - SDN) [64]. SDN predstavlja jedno od rješenja za probleme sa kojima se suočavaju tradicionalne mrežne arhitekture

zasnovane na decentralizaciji i kompleksnim modelima upravljanja saobraćajem pri čemu uređaji nemaju pristup informacijama o kompletnoj mreži već se sve završava na vrlo ograničenom vidokrugu. Mrežni uređaji u takvim mrežama su većinom samostalni u odlučivanju i moraju da se izbore sa sve većom količinom podataka koje treba obraditi na vrlo ograničenim dostupnim procesorskim i memorijskim resursima. Možda je najilustrativniji primjer veličina globalne BGP tabele koja je 1999. godine ima približno 50.000 unosa, 2009. godine malo manje od 300.000 dok danas ima preko 800.000 unosa [65]. Sličan trend je prisutan i na nivou lokalnih mreža, naročito u oblasti infrastrukture kao servisa, jer se povećanjem broja servera, virtuelnih mašina i virtuelizovanih mrežnih resursa stvara dodatno opterećenje na fizičke mrežne uređaje koji pored samog upravljanja saobraćajem moraju i da vode računa o rutama ili tabelama prosljeđivanja. Upotreba SDN uklanja taj teret sa mrežnih uređaja jer je sada željeno rutiranje i prosljeđivanje moguće izračunati upotrebom značajno većih centralizovanih resursa i mrežnim uređajima samo isporučiti gotove rezultate potrebne za njihov rad. Na taj način je moguće postići i veću efikasnost mreže uz upotrebu jednostavnijih i ekonomičnijih uređaja.

2.3.1. SISTEMI ZA NADZOR

Sistemi za nadzor mrežnih infrastruktura predstavljaju najstariji i najčešće korišćen koncept obrađen u ovom radu. Kako je riječ o vrlo dobro poznatoj oblasti koja je obrađena u velikom broju radova i knjiga, u ovom poglavlju će biti navedeni samo najznačajniji sistemi otvorenog koda za nadzor mreža.

Svakako najpoznatiji sistem je Nagios (Nagios Core) [7] koji omogućava nadzor raznovrsnih umreženih uređaja i servisa. Ima modularnu arhitekturu i veliki broj zvaničnih i nezvaničnih testova za testiranje praktično svih popularnih uređaja i servisa. Mane su prvenstveno vezane za relativno arhaičan proces konfigurisanja i neugledan veb bazirani interfejs. Icinga je sistem za nadzor razvijen od strane grupe autora Nagios-a i predstavlja pokušaj modernizacije korisničkog okruženja [66]. Važno je napomenuti da veliki broj i besplatnih i komercijalnih alata koristi Nagios ili njegove module uz izmjenu problematičnog korisničkog okruženja i jednostavniju integracije sa drugim sistemima.

Zabbix je još jedan primjer dobro poznatog sistema za nadzor mreža i servisa koji odlikuje upotreba veb interfejsa i izbjegavanje kompleksnih podešavanja za većinu tipičnih namjena [49]. Projektovan je prvenstveno za relativno manje infrastrukture do hiljadu nadziranih čvorova tako da je potrebno obratiti pažnju na performanse sistema pri nadzoru kompleksnijih sistema.

Osim navedenih sistema u upotrebi je veliki broj alata za nadzor mreža koji se kreću od jednostavnih sistema za pohranu mjernih podataka kao što je RRDtool [63], preko sistema prvenstveno namijenjenih grafičkoj prezentaciji prikupljenih podataka kao što su Cacti [67] i Grafana [68] do kompleksnih sistema poput OpenNMS-a [69]. Jedna zajednička odlika svih navedenih alata je podrška za SNMP. Od 46 sistema koji su bili obuhvaćeni preliminarnim istraživanjem samo dva sistema nemaju podršku za SNMP: Octopussy – namjenski razvijen servis za analizu sistemskih dnevnika, i FreeNATS – sistem usmjeren na testiranje mrežno dostupnih servisa bez ozbiljne podrške za rad sa mrežnim uređajima.

2.4. INTERNET STVARI

Internet stvari (eng. *Internet of Things* - IoT) [70] možda znači različite stvari različitim osobama, međutim malo je onih koji će osporiti da je u cilju ostvarivanja punih potencijala pametnih okruženja zasnovanih na IoT-u neophodno biti u stanju da se na efikasan način vrši nadzor svih komponenti koje čine IoT mogućim. Ako želimo da vršimo nadzor IoT infrastruktura, neophodno je nadzirati sve elemente infrastrukture, od najjednostavnijih senzora, preko komunikacionih mreža do virtuelizovanih servisa koji sirove mjerne podatke pretvaraju u korisne informacije. Nažalost, ne postoji jedan protokol koji bi sam pokrio sve slučajeve upotrebe sa kraja na kraj infrastrukture. Kada govorimo o nadzoru samog IoT okruženja, potrebno je obratiti posebnu pažnju na dva široko korišćena standarda: CoAP [71] i MQTT [72].

CoAP je u RFC 7252 definisan kao "specijalizovani veb transfer protokol za upotrebu na uređajima ograničenih mogućnosti i odgovarajućim mrežama (npr. niske snage i visokih gubitaka)" namijenjen za M2M aplikacije. Protokol je prvenstveno namijenjen za upotrebu na uređajima sa vrlo ograničenim procesorskim, memorijskim i mrežnim resursima. Baziran je na UDP protokolu i koristi prilagođeni podskup HTTP-a optimizovan za upotrebu u M2M okruženjima, prije svega kroz podršku za mogućnosti koje nisu prisutne u HTTP-u ali su od velike

važnosti u M2M primjenama kao što su multikast, otkrivanje topologije i asinhrona razmjena poruka [71]. Namjenski je razvijen na takav način da u eksploataciji koristi zanemarive procesorske resurse. Sa stanovišta obrade zahtjeva, CoAP je vrlo sličan HTTP-u, ali je ograničen samo na GET, POST, PUT i DELETE poruke koje odgovaraju istoimenim HTTP metodima. CoRE format linkova je opisan u RFC 6690 [73] i definiše unaprijed poznatu pristupnu tačku ("/.well-known/core") koja omogućava klijentu da pristupi listi svih dostupnih servisa i na taj način se može koristiti za otkrivanje okruženja, prikupljanje informacija o dostupnim resursima i druge srodne namjene. Trenutno se radi na proširenju pune podrške za CoAP na alternativnim transportnim protokolima kao što su TCP, P2P, WebSockets, ZigBee [74] i drugi. Ova proširena podrška omogućava širu primjenu CoAP protokola u većem broju mogućih scenarija prisutnih u IoT okruženjima.

MQTT protokol, kako je definisan od strane OASIS-a [72], je lak, jednostavan i otvoren klijent-server orjentisani *publish/subscribe* protokol za razmjenu poruka. Slično CoAP-u, namijenjen je za upotrebu u M2M okruženjima i na uređajima sa ograničenim resursima. Može da koristi TCP/IP protokol, ali i druge transportne protokole koji omogućavaju uređeni, dvosmjerni prenos bez gubitaka, kao što je ZigBee [74]. Posebna verzija MQTT-SN je razvijena za upotrebu u senzorskim mrežama koje čine uređaji sa izuzetno ograničenim resursima povezani vrlo nepouzdanim vezama, a koje je moguće integrisati upotrebom MQTT-SN Forwarder ili MQTT-SN Gateway pristupa [75].

MQTT koristi publish-subscribe obrazac u kom se klijenti (eng. *publisher*) povezuju na server koji ima ulogu brokera za prosljeđivanje poruka što im omogućava slanje poruka na odgovarajuća odredišta bez obzira na to ko prima poruke (eng. *subscriber*). Same poruke mogu biti filtrirane na osnovu nekog od atributa poruke od kojih je najznačajniji tema poruke (eng. *topic*). Same teme mogu biti hijerarhijski uređene sa ciljem bolje organizacije kompleksnih okruženja, na primjer "building1/room007/rack02/server27/temperature". Na ovaj način je omogućeno konzumentu poruke da odabere da prima samo poruke koje potiču iz uređaja u željenoj zgradi ili sve poruke određene vrste bez obzira na izvor. Kako se komunikacija vrši asinhrono, sama odredišta mogu postojati i u trenucima u kojima nema povezanih pošiljalaca niti primalaca poruka. Ovakav pristup omogućava upotrebu u okruženjima u kojima se krajnje tačke povremeno povezuju na sistem i u kojima ne postoji konstanta veza.

Iako se velika pažnja posvećuje uređajima i mrežama ograničenih mogućnosti, sama IoT okruženja se ne sastoje samo od takvih elemenata. Od velike važnosti za pravilno funkcionisanje IoT infrastruktura su i nadzor pozadinskih servisa koji se tipično izvršavaju u okruženjima koja nisu resursno ograničena, kao i mreža koje služe za povezivanje IoT uređaja sa tim servisima. Dodatna komplikacija leži u činjenici da su u savremenom okruženju i računarski i mrežni resursi virtuelizovani što dovodi do problema nadzora koji su opisani u prethodnom tekstu.

Kada govorimo o mrežnoj komponenti, važno je istaći da gotovo svi uređaji, osim eventualno namjenskih MQTT-SN elemenata, podržavaju upotrebu SNMP-a. Uređaji koji ga ne podržavaju su tipično neupravljivi i transparentni su sa stanovišta mreže. Virtuelizovani serveri koji izvršavaju pozadinske servise su pod kontrolom korisnika i na jednostavan način je moguće dodati podršku za SNMP ukoliko ona već i ne postoji. Kao što je ranije navedeno, problem pri nadzoru predstavlja nemogućnost pristupa podacima koji potiču sa stvarnog hardverskog uređaja jer je korisnik ograničen na pristup samo podacima koji postoje u virtuelizovanom okruženju.

2.5. SIMPLE NETWORK MANAGEMENT PROTOCOL

SNMP je zvanični dio IP skupa protokola definisanih od strane IETF-a [4] i definiše skup standarda za nadzor i upravljanje mrežnim infrastrukturama i umreženim uređajima. Sam standard je definisan kroz seriju RFC dokumenata koji propisuju sve segmente standarda, od skupova i organizacije podataka do protokola koji se koriste. Važno je napomenuti da aktuelna verzija SNMP-a ima status punog standarda u skupu IP protokola i nosi naziv STD0062. U doba kada postoji veliki broj novih ideja i sistema u fazi razvoja, ali i još veći broj ideja i sistema u fazi tihe smrti, jer su autori ili kompanije zaključili da postoje i profitabilniji pravci razvoja, od velike je važnosti postojanje stabilnih standarda, naročito u oblasti nadzora i upravljanja kao kritične komponente modernih informaciono-komunikacionih infrastruktura.

2.5.1. ISTORIJAT I RAZVOJ

Kako je SNMP pratio razvoj mrežnih infrastruktura i prilagođavao se promjenama u načinu njihove upotrebe, te razvoju novih sigurnosnih izazova, tako su definisane i različite verzije

standarda pri čemu je preporučljiva upotreba isključivo aktuelnih verzija standarda jer su sve prethodne verzije zastarjele. SNMP je kroz istoriju prošao kroz tri značajne verzije:

- Verzija 1 (SNMPv1) danas ima status istorijskog standarda koji nije pogodan za upotrebu u savremenim mrežnim okruženjima, ali usljed specifičnosti tržišta i dalje je široko podržan, naročito od strane jednostavnijih ili jeftinijih uređaja. Sigurnosni model je u potpunosti prevaziđen i ne propisuje nikakav vid kriptovanja podataka.
- Verzija 2 (SNMPv2) je donijela brojna poboljšanja, prije svega po pitanju performansi, ali i definisanjem mnogo naprednijeg sigurnosnog modela. Nažalost, usljed relativne kompleksnosti modela i nespremnosti tržišta da ga prihvati, u praksi je najzastupljenija varijanta SNMPv2c koja je zadržala prevaziđene sigurnosne mehanizme iz verzije 1.
- Verzija 3 (SNMPv3) je aktuelna verzija standarda i jedina čija se upotreba preporučuje. Pored poboljšanja performansi, sigurnosni model je bitno unaprijeđen i omogućava upotrebu mehanizama uporedivih sa drugim modernim protokolima.

2.5.2. ARHITEKTURA SNMP SISTEMA

SNMP je zamišljen kao standardni klijent-server protokol u kojem su glavni učesnici u komunikaciji administrativni čvor (eng. *master*) i upravljani uređaj (eng. *slave*) koji izvršava softversku komponentu agenta (eng. *agent*) namijenjenu za komunikaciju sa administrativnim čvorom. Agenti mogu obezbijediti jednostavan pristup podacima uređaja u svrhu nadzora, ali i omogućiti punu kontrolu nad uređajem kroz postavljanje određenih konfiguracionih i operativnih parametara uređaja.

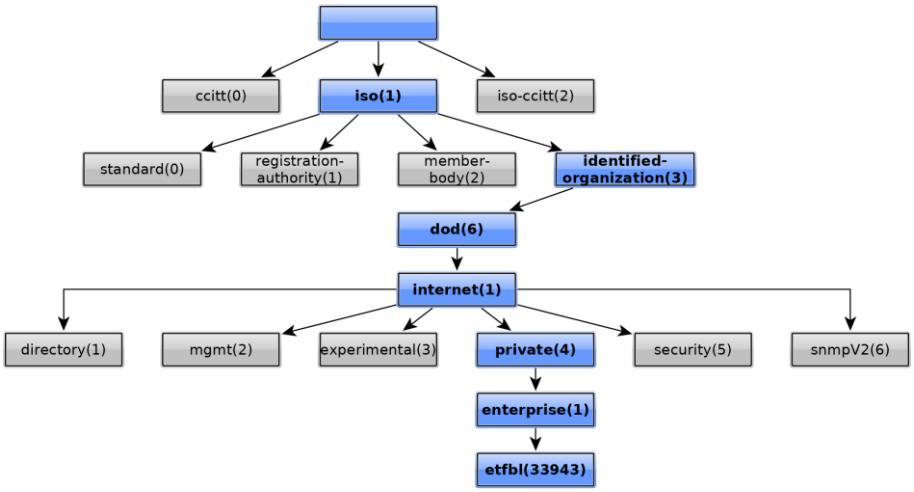
Iako je moguće koristiti i druge pristupe, komunikacija se obično odvija upotrebom UDP protokola preko portova 161 i 162 na strani agenta i administrativnog čvora, respektivno. Samu komunikaciju može inicirati administrativni čvor korišćenjem odgovarajućih GET ili SET operacija, ali je može inicirati i upravljani uređaj kroz TRAP ili INFORM operacije.

2.5.3. ORGANIZACIJA PODATAKA I MANAGEMENT INFORMATION BASE

Svi podaci kojima je moguće pristupiti u okviru SNMP-a su definisani u strogoj hijerarhijskoj strukturi sastavljenoj od virtuelnih baza podataka koje se nazivaju MIB (eng. *Management Information Base*) [76]. Svaki unos u hijerarhiji je jedinstveno identifikovan pomoću vrijednosti koja se zove OID (eng. *Object Identifier*). MIB stablo na vrhu ima jedan neimenovani korijen iz koga potiče prvi red grana dodijeljenih velikim međunarodnim organizacijama iz oblasti upravljanja standardima, kao što su ISO i ITU-T. Svaka od organizacija je zadužena za održavanja svog dijela globalnog stabla. Najinteresantnija grana za ovaj rad je .1.3.6.4.1 koja je predviđena za dodjelu pravnim licima kao što su proizvođači opreme ili razvojne institucije. Svaki entitet koji želi da dobije globalni identifikator u okviru ovog stabla mora proći proces registracije, što je Elektrotehnički fakultet Univerziteta u Banjoj Luci i učinio i dobio na upravljanje granu **.1.3.6.1.4.1.33943,** odnosno simbolički:

.iso.identified-organization.dod.internet.private.enterprise.etfbl

Slika 2.3. sadrži prikaz hijerarhije OID-a i označenu putanju od korijena do grane dodijeljene Elektrotehničkom fakultetu.



Slika 2.3. Primjer hijerarhije OID-a

Svi podaci predstavljeni MIB-om pripadaju jednoj od dvije moguće vrste podataka:

- Skalari – podaci koji definišu jednu instancu objekta.
- Tabele – nula ili više povezanih redova (instanci objekata) koji logički formiraju tabelu. Pristup je moguć na nivou jedne vrijednosti ili cijelog reda.

Kako je SNMP globalni opšti standard, problem različitosti u prezentaciji podataka je riješen upotrebom podskupa ISO standarda ASN.1 (eng. *Abstract Syntax Notation One*) [77] i SMI (eng. *Structure of Management Information*) [78]. Srodni upravljivi objekti u MIB-u su grupisani u jednu od tri grupe i koriste odgovarajuće makroe:

- MODULE-IDENTITY – definicije i opis semantike modula.
- OBJECT-IDENTITY – definicije objekata opisuju sintaksu i semantiku upravljivog objekta.
- NOTIFICATION-TYPE – definicije obavještenja opisuju sintaksu i semantiku obavještenja generisanih od strane agenta prema administrativnom čvoru.

2.5.4. PROŠIRENJE FUNKCIONALNOSTI AGENTA

Funkcionisanje SNMP-a je u velikoj mjeri zavisno od funkcionalnosti implementiranih u okviru agenta koji se izvršava na upravljanom uređaju. Sami agenti prema prirodi stvari imaju ograničene mogućnosti i ukoliko korisnik ima zahtjeve koji nisu trenutno podržani od strane agenta jedno moguće rješenje je proširenje funkcionalnosti agenta. U opštem slučaju se proširenje može izvesti na jedan od tri načina:

- Modifikacija agenta – ukoliko je korisniku dostupan izvorni kod agenta, moguće je pristupiti modifikaciji koda i implementirati nedostajuću funkcionalnost. Ovaj pristup nudi potencijalno najbolje performanse uz značajna ograničenja i pitanja vezana za izvodivost procedure modifikacije (Da li je izvorni kod dostupan? Kakvi su problemi vezani za pitanja intelektualne svojine? Da li će modifikovani agent funkcionisati na istovjetan način prema uređaju kao originalna verzija?).
- Upotreba eksternog programa – u ovom slučaju ne dolazi do modifikacije agenta već se dodatna funkcionalnost ostvaruje povremenim ili kontinualnim izvršavanjem eksternog programa koji obezbjeđuje podatke za originalnog agenta.

Potencijalni problemi su podrška za specifičnosti svakog od agenata i problem pravljenja univerzalno upotrebljivog rješenja.

- Upotreba SNMP AgentX protokola – upotreba *master-slave* odnosa agenata koji funkcionišu slično kao eksterni programi ali uz poštovanje svih standarda i mogućnost kreiranja univerzalnog rješenja podržanog od strane svih standardnih SNMP agenata. AgentX je plod rada IETF radne grupe pod nazivom SNMP *Agent Extensibility Working Group* koja je proizvela okvir za proširenja [10] i MIB dokument [79].

Sigurnosni aspekti vezani za upotrebu SNMP-a i proširenje funkcionalnosti agenta su dati u poglavlju 4.2.7. Sigurnosni aspekti.

3. NADZOR DISTRIBUIRANE RAČUNARSKE INFRASTRUKTURE

3.1. REGIONALNA INFRASTRUKTURA I VI-SEEM PROJEKAT

Region jugoistočne Evrope je u proteklih 15 godina obuhvaćen serijom projekata koji za cilj imaju uspostavu i razvoj regionalne e-infrastrukture koja obuhvata mrežne [80], računarske [81] i aplikativne [82] resurse dostupne naučnoistraživačkoj zajednici. Cilj trogodišnjeg Horizon 2020 projekta pod nazivom VI-SEEM (eng. *Virtual research environment for regional Interdisciplinary communities in Southeast Europe and the Eastern Mediterranean*) [9] bio je kreiranje virtuelnog istraživačkog okruženja povezivanjem infrastruktura prisutnih u 16 zemalja na tri kontinenta. Naglasak u ovom projektu je stavljen na pojednostavljenje upotrebe e-infrastruktura za multidisciplinarne korisnike koji pripadaju u tri osnovne oblasti: klimatologija, bionauke i kulturno nasljeđe. Da bi sami resursi bili upotrebljivi za krajnje korisnike, neophodan je i funkcionalan sistem za nadzor pravilnog funkcionisanja svih komponenti ovog složenog i distribuiranog sistema. Sama infrastruktura se sastoji od 14 superračunarskih centara sa preko 22800 procesorskih jezgara, 760000 GPGPU jezgara i 18000 Xeon Phi jezgara, deset IaaS klastera sa preko 14000 virtuelnih mašina, sedam grid klastera sa 3200 procesorskih jezgara, te trinaest klastera za skladištenje podataka ukupnog kapaciteta od 8000 TB diskova i 9800 TB trake. Ova infrastruktura i pripadajući servisi opslužuju veliki broj naučnika omogućujući im jednostavan i efikasan način za provođenje istraživanja, skladištenje i razmjenu podataka što je izuzetno važno sa stanovišta smanjivanja digitalnog procjepa prema razvijenim dijelovima kontinenta.

Savremena računarska infrastruktura može biti izuzetno kompleksne prirode, sastavljena od komponenti koje kombinuju više koncepata upotrebe računarskih resursa, od tradicionalnih centralizovanih računarskih klastera do distribuiranih okruženja zasnovanih na grid i računarstvu u oblaku. Servisi koji se izvršavaju na ovakvim infrastrukturama takođe pokrivaju široku lepezu opcija, od namjenski razvijenih servisa za jednu grupu korisnika, preko onih dijeljenih između više srodnih grupa do generičkih servisa koji se koriste kao gradivni elementi naprednijih rješenja. Kada se svemu navedenom doda i činjenica da se infrastruktura prostire preko velikog broja granica, i geografskih i administrativnih, jasno je da postojanje adekvatnog sistema za nadzor

kritično za sve učesnike projekta. Nakon prikupljanja podataka od operatera i korisnika projektne infrastrukture, formulisani su sljedeći zahtjevi za sistem za nadzor: sistem mora biti pristupačan i jednostavan za upotrebu i integraciju u druge sisteme, a podaci moraju reprezentativno odslikavati stvarno stanje infrastrukture i servisa.

U radu sa kompleksnim sistemima je od velike važnosti postojanje organizovanog načina rada sa topologijom i operativnim podacima vezanim za sistem, kao i postojanje okvira za izvršavanje individualnih testova za provjeru funkcionalne ispravnosti svakog od servisa. Na osnovu prethodnih istraživanja i iskustava vezanih za grid, HPC i IaaS infrastrukture [83], za bazične alate su odabrani GOCDB (eng. *Grid Operations Centre Data Base*) [84] i ARGO sistem za funkcionalni nadzor [8].

3.2. IZVORI PODATAKA

Kada govorimo o izvorima podataka, iste možemo podijeliti u tri osnovne grupe:

1. organizacija infrastrukture – topologija i statički podaci koji zavise od ljudskog faktora i sporo se mijenjaju u vremenu,
2. operativni resursi – podaci koji potiču od instanci koje čine infrastrukturu i nose informacije o postojećim resursima i njihovoj zauzetosti,
3. funkcionalni nadzor – podaci o trenutnom stanju servisa, te njihovoj dostupnosti i pouzdanosti.

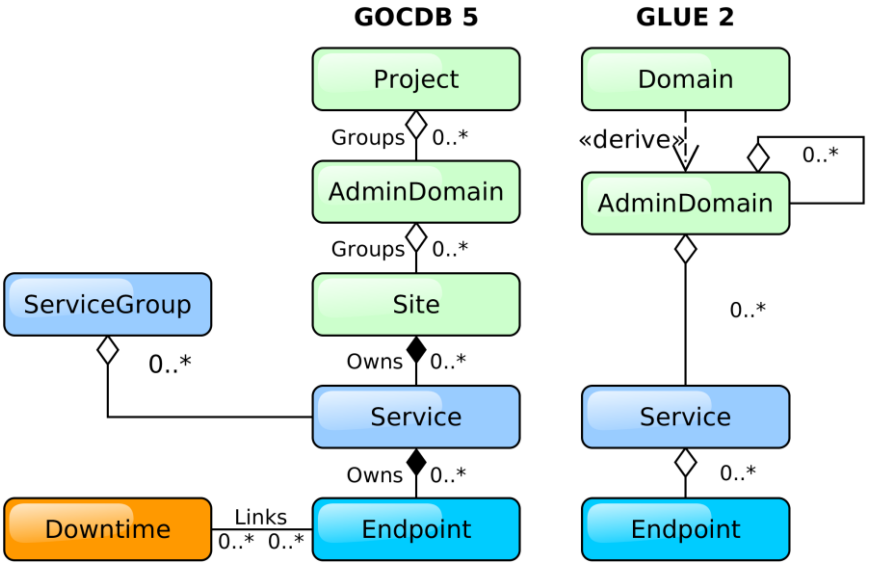
U nastavku teksta će biti dat detaljniji pregled svake grupe i tipičnih predstavnika relevantnih za ovaj rad.

3.2.1. ORGANIZACIJA INFRASTRUKTURE

Pri radu sa kompleksnim infrastrukturama od velike je važnosti posjedovanje kvalitetnih informacija o topologiji same infrastrukture, kao i o pojedinačnim pripadajućim komponentama, te ljudskim resursima vezanim za njih. Tipično rješenje za takve namjene je repozitorijum podataka poput GOCDB-a. GOCDB sadrži informacije o administrativnim domenima, državama, klasterima, čvorovima, servisima, aplikacijama, ali i operativne informacije o najavljenim

periodima nedostupnosti, te liste kontakata za svaki od resursa. Osnovni način pristupa informacijama je kroz veb portal koji koristi X.509 sertifikate, SAML2 ili kombinaciju korisničkog imena i pristupne lozinke za autentifikaciju korisnika. Autorizacija je zasnovana na RBAC modelu koji omogućava visok nivo fleksibilnosti jer organizacije ili pojedinačni projekti mogu samostalno upravljati pravima pristupa korisnika sistema.

Podaci su organizovani hijerarhijski sa projektom (eng. *Project*) na vrhu koji sadrži administrativne domene (eng. *Administrative domain*) koji sadrže klastere (eng. *Site*) koji sadrže servise (eng. *Service*) koji opet mogu sadržavati više krajnjih tačaka (eng. *endpoint*). Ovaj model blisko prati GLUE 2 [85] informacioni model koji opisuje D-AD-S-E (eng. *Domain-AdminDomain-Service-Endpoint*) hijerarhiju što je i ilustrovano na slici 3.1. Ovakav pristup olakšava integraciju sa drugim alatima koji su dijelovi distribuiranih računarskih infrastruktura ili midlvera kao što su gLite [86], ARC [58] i UNICORE [59].



Slika 3.1. GOCDB i GLUE 2 modeli

Pristup podacima je moguć i preko API-ja pod nazivom GOCDB PI (eng. *Programmatic Interface*) a koji koristi REST i XML. Kako podaci koji se čuvaju u GOCDB-u mogu sadržavati osjetljive ili lične podatke, definisana su tri stepena zaštite podataka:

- Nivo 1 – javni metodi koji ne rukuju osjetljivim ili ličnim podacima – bez ograničenja pristupa.

- Nivo 2 – zaštićeni metodi koju pristupaju ličnim podacima – korisnik mora imati validan X.509 sertifikat.
- Nivo 3 – privatni metodi koji imaju pristup sigurnosno osjetljivim podacima – korisnik mora imati validan i registrovan X.509 sertifikat.

Postojeći metodi mogu biti grupisani u četiri osnovne kategorije:

- Metodi administrativnog domena – pristup podacima o projektu, regionalnoj ili nacionalnoj organizaciji kroz zbirni ili individualni pristup.
- Metodi za rad sa klasterima – pristup zbirnim i individualnim podacima o klasterima, uključujući sigurnosne podatke.
- Metodi za rad sa servisima – pristup podacima o tipovima i grupama servisa, detaljne informacije o bilo kojoj instanci servisa.
- Metodi za rad sa najavljenim periodima nedostupnosti – kreiranje, ažuriranje i čitanje unosa vezanih za planirane periode održavanja sistema ili druge periode nedostupnosti registrovanih sistema.

Primjer upita koji vraća listu klastera u VI-SEEM projektu koji su registrovani u GOCDB instanci poziva metod *get_site_list* uz postavljen *scope* parametar na "*VI-SEEM*".

```
<results>
    <SITE ID="10" PRIMARY_KEY="8G0" NAME="ETFBL-CC01" COUNTRY="Bosnia and Herzegovina"
     COUNTRY_CODE="BA" ROC="VI-SEEM" SUBGRID="" GIIS_URL=""/>
…
    <SITE ID="8" PRIMARY_KEY="6G0" NAME="UVT" COUNTRY="Romania" COUNTRY_CODE="RO" ROC="VI-SEEM"
     SUBGRID="" GIIS_URL=""/>
</results>
```

3.2.2. GRID I HPC KLASTERI

Da bi klasteri bili dostupni kao dio distribuirane grid ili HPC infrastrukture neophodno je periodično publikuju informacije o sebi na centralne servise. Ovi centralni servisi predstavljaju instance BDII (eng. *Berkeley Database Information Index*) servera koji su zasnovani na LDAP (eng. *Lightweight Directory Access Protocol*) servisima koji implementiraju GLUE 2 model podataka. Na nivou svakog klastera postoji lokalna instanca servisa (eng. *site-bdii*) koja prikuplja

relevantne podatke iz statičkih i dinamičkih izvora. Statički izvori su prvenstveno konfiguracioni fajlovi i kontaktni podaci za servise u okviru lokalnog klastera. Dinamički izvori podataka su instance servisa čije stanje se mijenja u zavisnosti od aktivnosti klastera. Tako prikupljeni podaci se publikuju na centralni servis (eng. *top-bdii*) koji koriste drugi centralni servisi poput WMS (eng. *Workload Management System*) i LB (eng. *Logging and Bookkeeping*) servisa.

Centralni BDII serveri omogućavaju pristup podacima koji sadrže:

- Virtuelne organizacije – imena i opise.
- Statičke podatke o klasterima – imena, opise, lokaciju, kontaktne podatke, itd.
- Redove čekanja za računarske zadatke – pregled stanja raznih redova čekanja, broja zadataka po statusima, dostupne resurse, itd.
- Resurse za skladištenje podataka – ukupna, zauzeta i slobodna količina, prava pristupa, itd.

Kako su podaci organizovani prema GLUE 2 modelu, integracija sa podacima koji potiču iz GOCDB servisa je jednostavna.

3.2.3. IAAS KLASTERI

Kao što je ranije navedeno, postoji veliki broj različitih platformi za pružanje IaaS usluga. Svaka od platformi ima sopstveni API iako su primjetni napori usmjereni prema standardizaciji pristupa servisima, prije svega OCCI (eng. *Open Cloud Computing Interface*) razvijen od strane OGF (eng. *Open Grid Foruma*) [36]. Vrijedi istaći i istraživanja na temu automatizacije nadzora sistema koja je bazirana na upotrebama ontologija u oblasti IaaS-a [87]. Nažalost, trenutna situacije je takva da je daleko najrasprostranjeniji API u praktičnoj upotrebi OpenStack API tako da čak i sistemi koji ne koriste OpenStack za upravljanje resursima obezbjeđuju pristup preko OpenStack API-ja [18]. U okviru VI-SEEM projekta svi IaaS klasteri podržavaju neki vid pristupa pomoću OpenStack API-ja, uz podršku za bar *identity* i *compute* servise. Imajući navedeno u vidu, razvijeni sistem je testiran na OpenStack API kompatibilnim IaaS klasterima, ali ne postoji ništa u samom sistemu što je podređeno OpenStack okruženju ili što sprečava podršku za bilo koju drugu IaaS platformu jer je sam sistem razvijen da bude nezavisan od bilo koje konkretne platforme koja se koristi kao izvor podataka.

Kolektorska komponenta za OpenStack kompatibilne izvore podataka prikuplja sljedeće informacije sa odgovarajući API pristupnih tačaka:

- *Service catalogue* (eng. *keystone*) – podaci uključuju sve deklarisane servise koji se izvršavaju na klasteru.
- *Compute service* (eng. *nova*) – podaci uključuju podatke za svaki pojedinačni hipervizor, zbirne podatke za sve hipervizore, te podatke za vrste VM instanci i same VM instance.
- *Image service* (eng. *glance*) – podaci uključuju sve registrovane slike VM dostupne na klasteru.

3.2.4. VIRTUELIZACIJSKI KLASTERI

Ova vrsta izvora podataka može varirati u značajnoj mjeri od klastera do klastera jer postoji veliki broj raznorodnih rješenja za virtuelizaciju i isto tako velik broj različitih API-ja pa čak i rješenja koja nemaju definisan sopstveni javni API. Iako su primjetni napori [6] sa ciljem standardizacije [88] u realnim okruženjima je podrška za neki univerzalni standard praktično nepostojeća. Alternativno postoji više "univerzalnih" standarda promovisanih od različitih platformi koji su univerzalni samo ako se koriste isključivo proizvodi kompatibilni sa tom platformom.

Za potrebe razvoja sistema je odabrana Proxmox VE (eng. *Virtual Environment*) [45] platforma. Proxmox VE je platforma otvorenog koda namijenjena serverskoj virtuelizaciji koja podržava širok spektar tehnologija i načina upotrebe. Što se tiče same virtuelizacije, puna virtuelizacija je realizovana upotrebom KVM-a, dok je podrška za virtuelizaciju na nivou operativnog sistema omogućena kroz LXC. Prisutna je i podrška za upravljanje virtuelizovanim mrežnim resursima kao i velikim brojem tehnologija za lokalna, udaljena ili dijeljena skladišta podataka kao što su LVM, ZFS, iSCSI, Fiber Channel, NFS, GlusterFS, CEPH i DRBD. Sve navedeno čini Proxmox VE kvalitetnim izborom za predstavnika sistema za serversku virtuelizaciju. Integracija sa drugim sistemima je omogućena kroz REST API i interfejs dostupan iz komandne linije koji je pretežno namijenjen za automatizaciju rada sistema.

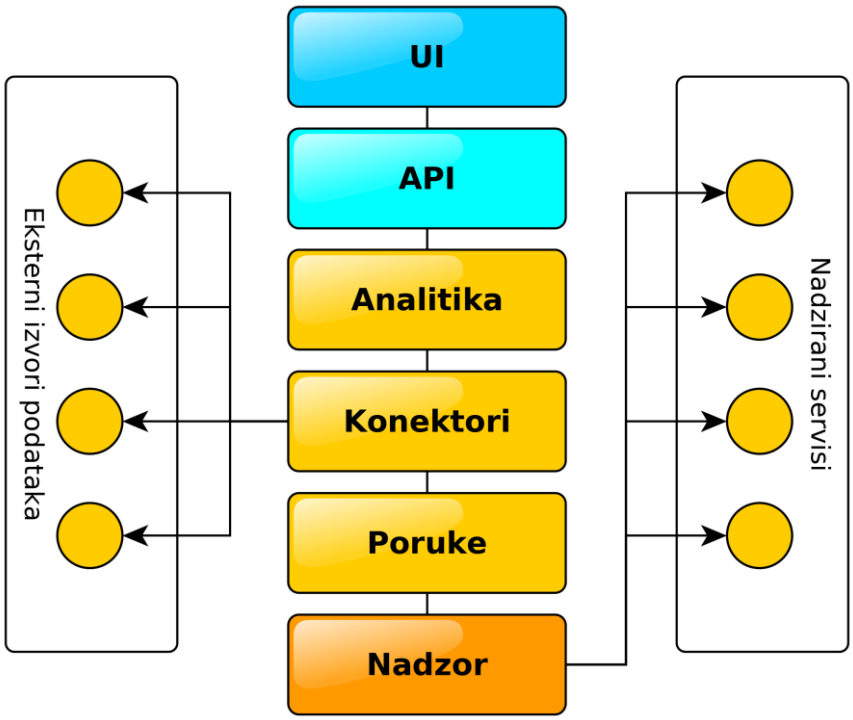
Kolektor podataka za Proxmox VE podržava prikupljanje sljedećih podataka:

- Podaci o pojedinačnom hipervizoru uključujući broj procesorskih jezgara, količinu radne i trajne memorije, itd.
- Sumarni podaci za sve hipervizore generisani na nivou klastera.
- Podaci o skladištima podataka – uključuju lokalne i dijeljene resurse.
- Podaci o svakoj pojedinačnoj VM koja se izvršava na klasteru.
- Podaci o vrstama instanci VM.
- Slike VM dostupne za upotrebu u okviru klastera.

3.2.5. FUNKCIONALNI TESTOVI – ARGO

ARGO [8] sistem za nadzor servisa predstavlja skalabilan i fleksibilan okvir za nadzor statusa, dostupnosti i pouzdanosti računarskih servisa. Modularna struktura sistema omogućava prilagođavanje i povezivanje sa eksternim sistemima što je bitna karakteristika za sistem namijenjen nadzoru širokog spektra servisa i različitim načinima upotrebe.

Arhitektura sistema, prikazana na slici 3.2, je organizovana u slojevima. Najniži sloj predstavlja sistem za direktni nadzor pristupnih tačaka servisa koji je baziran na Nagios [7] platformi. Za samo testiranje se koriste namjenski razvijeni testovi (eng. *probes*) i skup opcionih alata koji omogućavaju jednostavniju integraciju u distribuirano okruženje. Kako Nagios spada u red izuzetno popularnih sistema za nadzor, postoji veliki broj već napravljenih testova koje je moguće direktno koristiti ili ih prilagoditi konkretnim potrebama korisnika. Drugi sloj je zadužen za izračunavanje raspoloživosti i pouzdanosti sistema i sastoji se od sistema za prosljeđivanje poruka, konektora koji obezbjeđuju povezivanje sa eksternim sistemima i modulom za analitiku. Sama struktura je slabo spregnuta i omogućava izmjene i prilagođavanje svake pojedinačne komponente bez uticaja na ostatak sistema. Na vrhu se nalaze REST bazirani API i opcioni portal za pristup podacima.



Slika 3.2. Arhitektura ARGO sistema

ARGO API je REST HTTP API koji omogućava eksternim entitetima pristup podacima o trenutnom statusu, kao i o raspoloživosti i pouzdanosti nadziranih servisa. API podržava rad sa XML i JSON formatiranim podacima. Sistem kontrole pristupa je zasnovan na upotrebi ključa za pristup API-ju (eng. *x-api-key*) koji se šalje sa svakim zahtjevom u odgovarajućem polju zaglavlja zahtjeva. Podaci dostupni kroz API pokrivaju: projekte, izvještaje, administrativne operacije, metrike, statuse i rezultate. Za potrebe ovog rada dva najvažnija izvora podataka se tiču trenutnog statusa servisa (eng. *status*) i vrijednosti raspoloživosti i pouzdanosti za definisani vremenski period (eng. *results*).

Na primjer, da bi pristupili podacima o raspoloživosti i pouzdanosti servisa iz grupe CYI (eng. *Cyprus Institute*) za deveti mjesec 2019. godine potrebno je uputiti zahtjev na REST API pristupnu tačku na lokaciji `/api/v2/results/Critical/SERVICEGROUPS/CYI` i definisati parametre *start_time*, i *end_time* (vremenski period), te *granularity* (vrijednosti za cijeli mjesec). Odgovor API-ja će imati sljedeću formu:

```
{ "results": [
    { "endpoints": [
        { "type": "SERVICEGROUPS",
          "name": "CYI",
          "results": [
            { "uptime": "1",
              "timestamp": "2019-09",
              "unknown": "0",
              "reliability": "91.37",
              "downtime": "0",
              "availability": "91.37"
            } ] } ],
      "type": "PROJECT",
      "name": "VI-SEEM"
  } ]
}
```


3.2.6. FUNKCIONALNI TESTOVI – NAGIOS

Kako je sam ARGO zasnovan na Nagios sistemu, a često je u praktičnoj upotrebi prisutan samo Nagios bez viših slojeva, kreiran je i kolektor koji omogućava direktan pristup rezultatima generisanim od strane Nagios sistema. Kombinovanjem podataka iz GOCDB-a i Nagios-a moguće je generisati podatke potrebne za pravilan rad sistema a koji nisu prisutni u samom Nagios API-ju. Kolektor omogućava prikupljanje podataka o trenutnom statusu servisa, istoriji promjena statusa, te raspoloživosti i pouzdanosti servisa za proizvoljan period vremena.

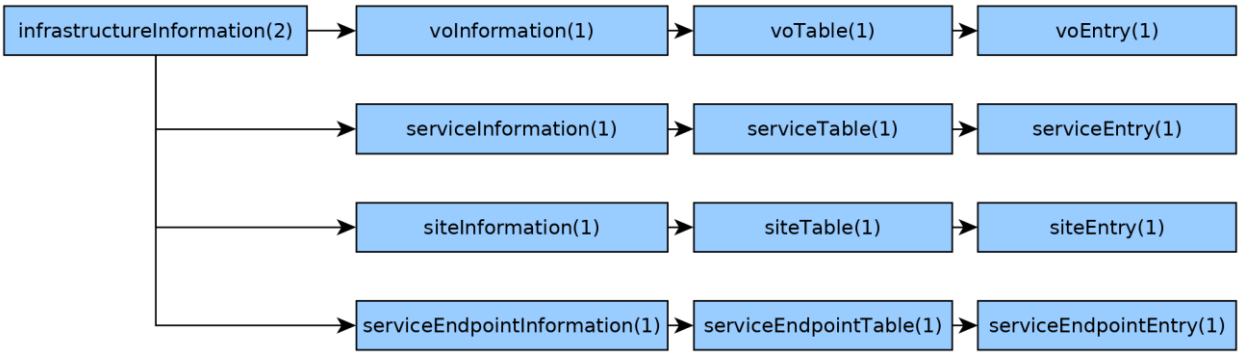
3.3. ORGANIZACIJA PODATAKA

Za potrebe razvoja sistema za nadzor distribuirane računarske infrastrukture razvijen je novi SNMP MIB koji omogućava efikasan i fleksibilan način rada sa prikupljenim podacima. Početni OID za ovaj MIB se nalazi u 1.3.6.1.4.1.33943 grani koja je dodijeljena na upravljanje Elektrotehničkom fakultetu Univerziteta u Banjoj Luci. Grana pod brojem 28 je određena za MIB za nadzor distribuirane računarske infrastrukture pod nazivom ETFBL-DCI-MIB. Imajući navedeno u vidu, OID na lokaciji 1.3.6.1.4.1.33943.28 će u nastavku teksta biti označavan sa "dci". Dizajn MIB dokumenta je urađen u skladu sa preporukama navedenim u SMIv2 (eng. *Structure of Management Information version 2*) i shodno tome se dokument sastoji od tri osnovne sekcije: *objects*, *events* i *compliances*. OID na lokaciji dci.1 je nazvan dciObjects a OID na lokaciji dci.3

dciCompliances. Svi podaci vezani za nadzor infrastruktura se nalaze u grani dciObjects koja se sastoji od sljedećih grana:

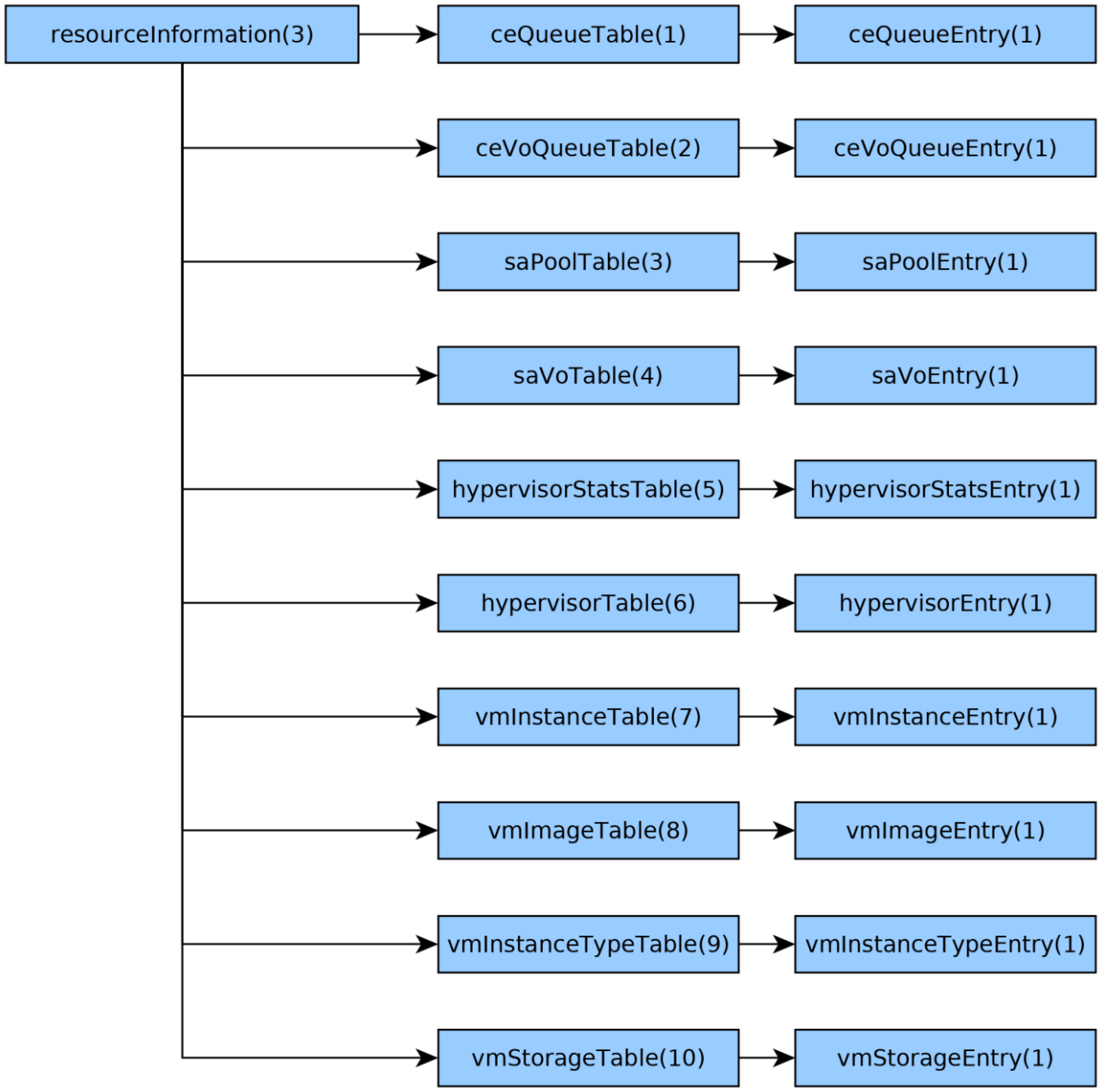
1. systemInformation – sadrži osnovne informacije o sistemu na kom se izvršava agent (naziv, opis i kontakt),
2. infrastructureInformation – informacije o topologiji infrastrukture,
3. resourceInformation – informacije o resursima dostupnim u infrastrukturi,
4. monitoringInformation – informacije o statusu, raspoloživosti i pouzdanosti nadziranih servisa.

Grana infrastructureInformation sadrži informacije o virtuelnim organizacijama (voInformation), servisima (serviceInformation), klasterima (siteInformation) i instancama servisa (serviceEndpointInformation) i prikazana je na slici 3.3.



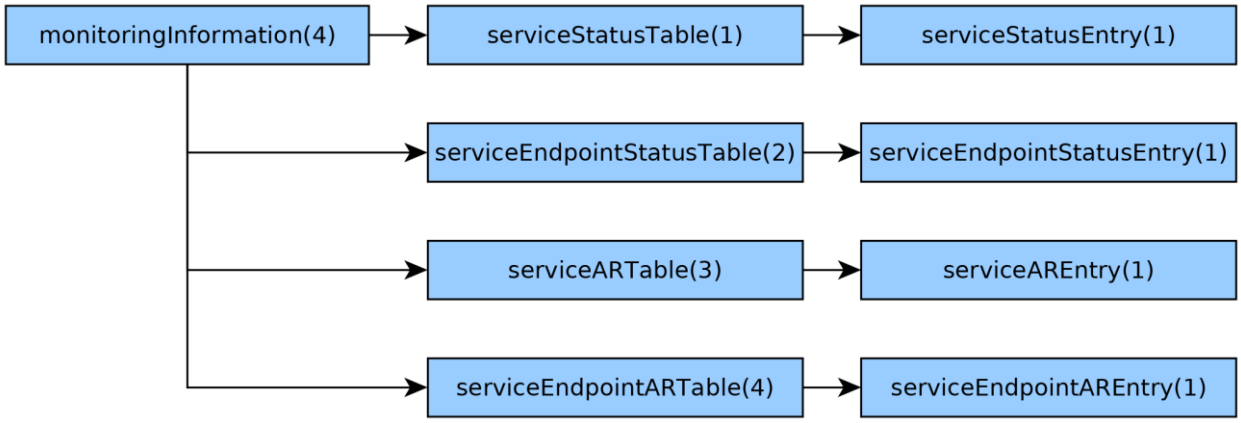
Slika 3.3. Dio MIB stabla – infrastructureInformation

Grana resourceInformation je prikazana na slici 3.4. i sadrži informacije o redovima čekanja za grid i HPC zadatke (ceQueue i ceVoQueue), skladištima podataka za grid i HPC (saPool i saVo), sumarne i pojedinačne podatke o hipervizorima (hypervisorStats i hypervisor), te informacije o instancama VM (vmInstance), slikama VM (vmImage), vrstama VM (vmInstanceType) i skladištima VM (vmStorage).



Slika 3.4. Dio MIB stabla - resourceInformation

Grana monitoringInformation je prikazana na slici 3.5. i sadrži informacije o trenutnim statusima servisa (serviceStatus), trenutnim statusima pojedinačnih pristupnih tački servisa (serviceEndpointStatus), dostupnosti i pouzdanosti servisa (serviceAR) i pristupnih tačaka servisa (serviceEndpointAR).



Slika 3.5. Dio MIB stabla - monitoringInformation

Moguće vrijednosti trenutnog statusa servisa i pristupnih tačaka servisa su definisane u tabelama 3.1. za servise i 3.2. za pristupne tačke servisa.

Tabela 3.1. Kod, naziv i opis statusa servisa

Kod statusa	**Naziv i opis statusa servisa**	
	Naziv	***Opis***
0	NA	Nepoznato ili neprimjenjivo
10	UP	Servis funkcioniše korektno
20	DOWN	Servis ne funkcioniše korektno
30	DEGRADED	Servis ima više instanci, makar jedna nije u UP status ali servis funkcioniše
40	PARTIAL	Servis ima više instanci, makar jedna nije u UP status ali servis funkcioniše u ograničenom kapacitetu
100	MAINT	Servis u najavljenom period održavanja

Tabela 3.2. Kod, naziv i opis statusa pristupne tačke servisa

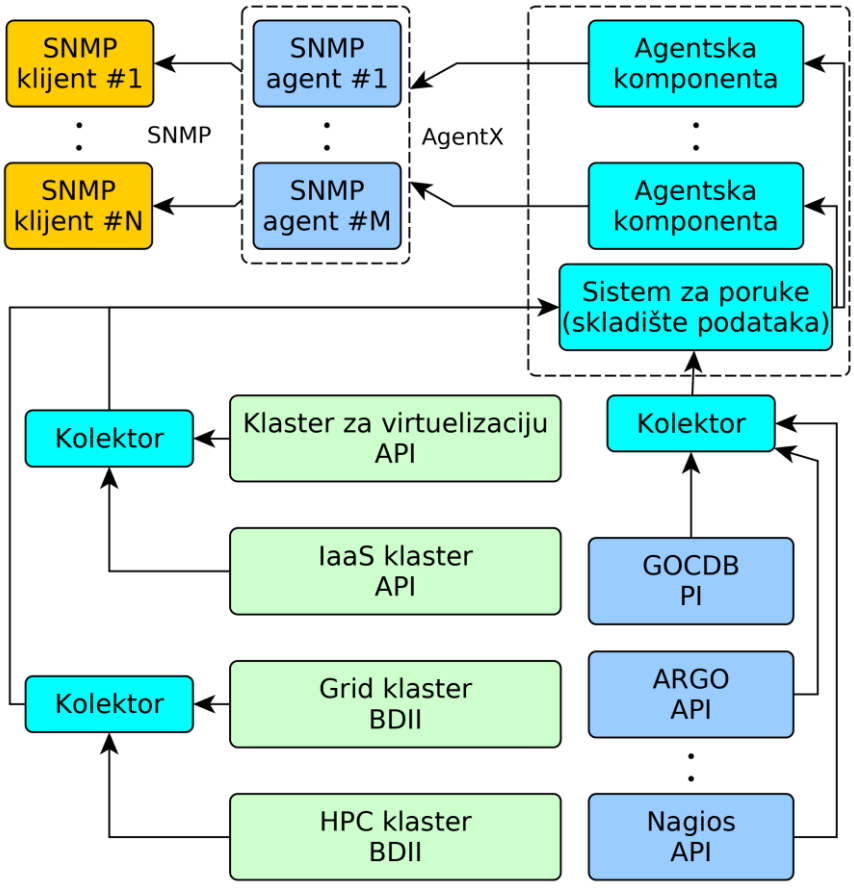
Kod statusa	Naziv i opis statusa pristupne tačke servisa	
	Naziv	*Opis*
0	NA	Nepoznato ili neprimjenjivo
10	OK	Tačka funkcioniše korektno
20	INFO	Tačka funkcioniše korektno ali postoje dodatne informacije u rezultatima testa
30	NOTE	Tačka funkcioniše korektno ali postoje važne napomene u rezultatima testa
40	WARN	Tačka prolazi kritične testove ali ne prolazi nekritične testove
50	ERROR	Minimalno jedan kritični test nije zadovoljen
60	CRIT	Minimalno jedan kritični test je u kritičnom status
100	MAINT	Tačka je u najavljenom periodu održavanja

Da bi MIB ispunjavao preporuke iz SMIv2 neophodno je kreirati i sekciju koja definiše koje je dijelove MIB-a neophodno implementirati, a koji su opcioni za implementaciju u okviru SNMP agenta. Sekcija koja definiše navedeno je vrste MODULE-COMPLIANCE i u ovom MIB-u je nazvana generalCompliance i definiše sljedeće:

- grupe systemInformationGroup, siteInformationGroup i voInformationGroup su obavezne u svakoj implementaciji,
- grupa ceInformationGroup je obavezna samo za sisteme koji nadziru *compute element* servere grid ili HPC klastera,
- grupa saInformationGroup je obavezna samo za sisteme koji nadziru *storage element* servere grid ili HPC klastera,
- grupa monitoringInformationGroup je obavezna samo ako sistem implementira funkcionalni nadzor infrastruktura i servisa i
- grupa hypervisorInformationGroup je obavezna samo ako se implementira nadzor resursa klastera koji koriste virtuelizaciju.

3.4. ARHITEKTURA SISTEMA ZA NADZOR

Kako je jedna od osnovnih ideja u toku razvoja bila da sistem bude fleksibilan i upotrebljiv u različitim namjenama, sam sistem je projektovan na takav način da može da radi i kao jedan monolitni server ali i kao potpuno distribuirano rješenje. Monolitni server treba da obezbijedi punu funkcionalnost u okviru jednog jednostavnog paketa pogodnog za manje instalacije ili za testiranje prije pune upotrebe. Distribuirani pristup omogućava slučajeve upotrebe u kojima se pojedine komponente nalaze u različitim administrativnim domenima i ne postoji centralna tačka otkaza sistema. Jednostavno dodavanje podrške za nove izvore podataka je takođe bilo od velike važnosti.



Slika 3.6. Arhitektura sistema

Da bi sistem zadovoljio sve navedene karakteristike dizajniran je kao slabo spregnuti modularni sistem baziran oko asinhronog prosljeđivanja poruka između komponenti sistema što omogućava heterogenu, distribuiranu, skalabilnu i proširivu prirodu sistema. Ovo je postignuto

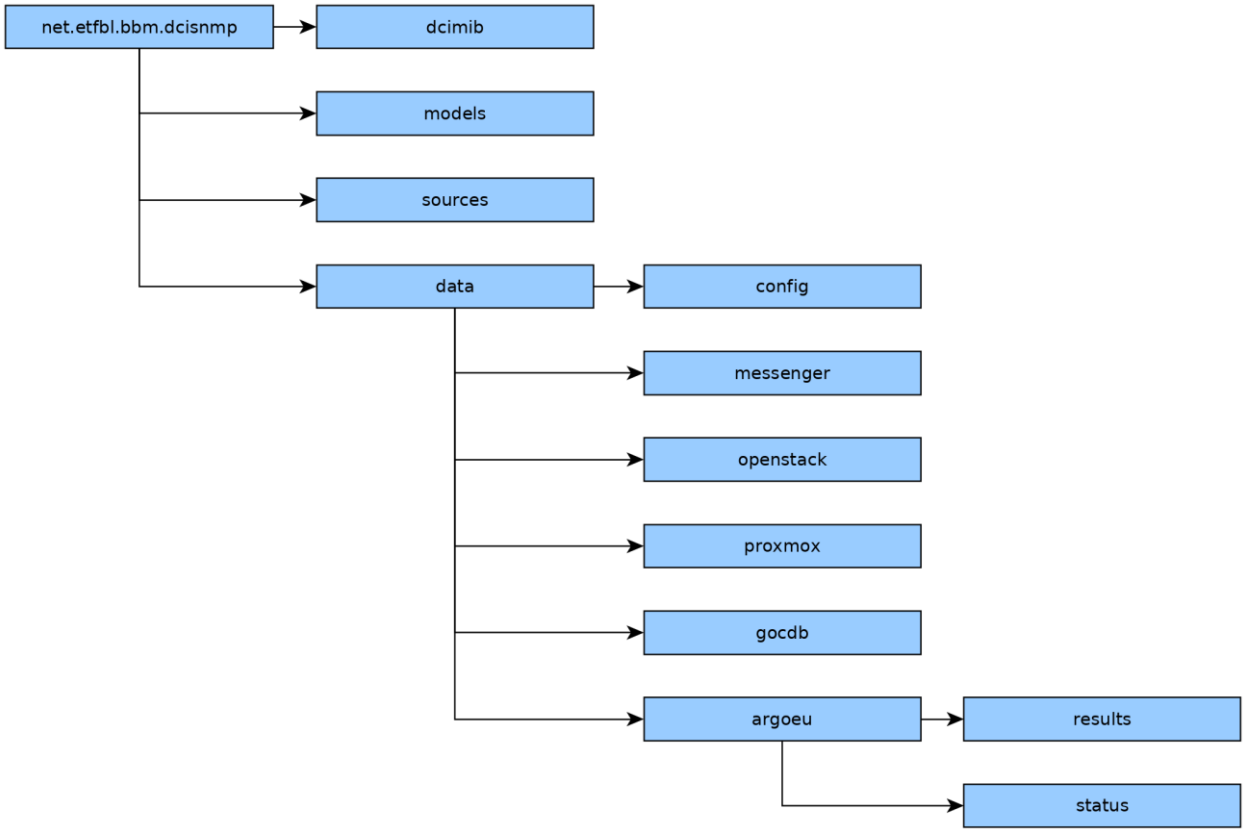
razdvajanjem logike u tri osnovna segmenta: kolektorske komponente, sistem za komunikaciju i agentske komponente. U zavisnosti od željenog načina rada, sistem za komunikaciju je ili interni sistem, u slučajevima monolitnog servera, ili eksterni sistem za prosljeđivanje poruka u slučaju distribuirane arhitekture. Generalizovana arhitektura sistema sa višestrukim kolektorskim i agentskim komponentama je prikazana na slici 3.6.

Bez obzira na odabranu arhitekturu, rad sistema je uvijek isti: kolektori prikupljaju podatke iz svih konfigurisanih izvora podataka, transformišu ih u odgovarajući format i šalju sistemu za komunikaciju dok agentske komponente primaju dolazne poruke na osnovu kojih generišu interno stanje sistema koje daju na uvid SNMP klijentima. Ovakav pristup omogućava upotrebu proizvoljnog broja instanci svake od komponenti prema konkretnim potrebama, bilo da se radi o horizontalnom skaliranju sa ciljem poboljšanja performansi i opsluživanja većeg broja krajnjih tačaka ili da je želja odvajanje kolektora prema administrativnim domenima ili kreiranje kompleksnih struktura u kojima se koristi rutiranje i filtriranje poruka prema zahtjevima implementacije. Sama konfiguracija sistema je smještena u XML fajl, ali može uključivati i dodatne fajlove ili izvore podataka prema potrebi.

3.5. PRAKTIČNA REALIZACIJA SISTEMA I ORGANIZACIJA KLASA

Praktična realizacija sistema je izvedena upotrebom programskog jezika Java, te većeg broja potrebnih biblioteka od kojih su najvažnije *jax*, *slf4j*, *gson*, *jackson*, *unirest*, *jms-api* i *amqp-client*. Klase za rad sa *jax* bibliotekom su generisane na osnovu MIB dokumenta pomoću programa *smidump*.

Slika 3.7. sadrži pregled osnovne organizacije paketa praktične realizacije sistema. Osnovni paket projekta je *net.etfbl.bbm.dcisnmp* i sadrži četiri paketa: *data, dcimib, models* i *sources*. U okviru osnovnog paketa su smještene samo osnovna klasa za pokretanje sistema nazvana *DCISNMP*, te pomoćna klasa *DCIUtil*. Paket *data* sadrži klasu *DCIData* koja predstavlja osnovnu strukturu podataka koju koriste sve ostale komponente sistema i organizacijom podataka prati razvijeni MIB. Ovaj paket takođe sadrži i osnovnu klasu naziva *DataSource* koju nasljeđuju sve klase koje rade sa izvorima podataka. Paketi definisani unutar *data* su: *config*, *messenger*, *openstack*, *proxmox*, *gocdb*, te *argoeu* koji ima dva paketa naziva *results* i *status*.



Slika 3.7. Osnovna organizacija paketa projekta

Osnovni princip rada sistema je sljedeći:

1. Sistem parsira konfiguracione fajlove i definiše osnovnu strukturu podataka pomoću *DCIData* klase.
2. Ukoliko je definisana AgentX veza sistem je uspostavlja i vrši se registracija svih potrebnih unosa iz paketa *data.dcimib*.
3. Sistem pokreće interni komunikacioni sistem ili se povezuje na eksterni u zavisnosti od konfiguracije.
4. Vrši se inicijalizacija svih konfigurisanih kolektora podataka upotrebom *factory* metoda *createDataSource* klase *DataSource* nakon čega svaki od kolektora počinje sa prikupljanjem podataka.
    a) Po prijemu podataka, kolektor ih transformiše u interni format i prosljeđuje sistemu za komunikaciju kao jednu veliku poruku ili kao niz pojedinačnih poruka u zavisnosti od konfiguracije.

b) *Messenger* instanca isporučuje poruke na odgovarajuće odredište.
c) Primljene poruke se parsiraju i na osnovu dobijenih podataka se ažuriraju strukture i podaci definisani *DCIData* klasom na osnovu klasa iz paketa *data.models*.

5. Proces se ponavlja do kraja rada sistema.

3.5.1. KOLEKTORSKE KOMPONENTE

Generalno posmatrano postoje dvije osnovne vrste kolektorskih komponenata: aktivne i pasivne. Aktivne vrše periodično ispitivanje izvora podataka (eng. *polling*) dok pasivne čekaju na prijem poruke iz odgovarajućeg sistema za prosljeđivanje poruka (eng. *listening*). Bez obzira na vrstu kolektora, svi su definisani u konfiguracionom XML fajlu kao krajnje tačke (eng. *endpoint*). Svaka tačka je definisana tipom, URL-om, opcionim pristupnim parametrima, periodom prikupljanja podataka kao i parametrima specifičnim za dati izvor podataka. Na primjer, jedan unos u konfiguraciji definiše da je tip izvora "LDAP", da je URL "ldap://bdii.my.tld:2170/Mds-vo-name=local,o=grid", te da se prikupljanje podataka vrši svakih 5 minuta. S druge strane, primjer pasivnog kolektora je unos sa tipom "JMS", URL-om "tcp://jms.etfbl.net:61616" i definisanim potrebnim pristupnim parametrima (korisničko ime i lozinka, teme i filteri, itd).

Svi kolektori su minimalno predstavljeni klasom u paketu *net.etfbl.bbm.dcisnmp.sources* koja nasljeđuje klasu *DataSource* i ponovo implementira metod *getData*, te obezbjeđuje konstruktor koji prima odredište za poruke i konfiguracione parametre kao ulaz, a naziv klase se završava na *Collector*. Na ovaj način je omogućeno jednostavno proširenje podržanih izvora podataka u budućnosti, dok su trenutno podržani sljedeći izvori podataka: GOCDB, LDAP/BDII, ARGO, Nagios, OpenStack i Proxmox VE.

3.5.2. KOMUNIKACIONI SISTEM

Centralna tačka cijelog sistema za nadzor je komunikacioni sistem. Kao što je ranije navedeno, sistem može koristiti ili interni komunikacioni sistem ili eksterni sistem za prosljeđivanje poruka. Interni sistem je relativno jednostavan i zasnovan je na redovima čekanja i ključevima za rutiranje. Kako se radi o rješenju koje se koristi kod upotrebe monolitnog servera,

svi podaci se nalaze u istom dijeljenom memorijskom prostoru, a sve komponente dijele isti format podataka, prilikom upotrebe internog sistema se izbjegava kreiranje poruka i njihovo ponovno parsiranje. Ovaj pristup obezbjeđuje najbolje performanse, ali samo u slučajevima da su zahtjevi ispunjivi upotrebom jednog monolitnog servera.

Alternativni pristup podrazumijeva postojanje eksternog AMQP servera koji je podržan od strane RabbitMQ klijentske biblioteke [89] ali je jednostavno proširiv i na druge sisteme za prosljeđivanje poruka. Poruke koje se šalju na eksterni server su serijalizovane u standardizovane JSON poruke i dodjeljuje im se odgovarajući ključ za rutiranje (ili drugi odgovarajući atribut) koji kasnije može biti korišten za selektivno filtriranje poruka.

Sami podaci mogu biti slani na dva načina: zbirno i pojedinačno. Zbirno slanje grupiše sve podatke koje je kolektor obradio u jednom prolazu i generiše jednu poruku koja ih sve sadrži. Na ovaj način je smanjena mogućnost preciznog rutiranja i filtriranja poruka ali se poboljšavaju performanse sistema. Zbirno slanje se uvijek koristi kod internog komunikacionog sistema. Individualno slanje poruka dijeli prikupljene podatke u kraće poruke koje je lako filtrirati i dostaviti samo zainteresovanim stranama. Na primjer, kod prikupljanja podataka iz GOCDB servisa, zbirno slanje će sve podatke o svim virtuelnim organizacijama, klasterima, servisima, itd, poslati u jednoj poruci, dok će individualno slanje generisati veliki broj poruka, po jednu za svaki pronađeni unos u sistemu. Na taj način strana koju interesuju samo klasteri i pristupne tačke servisa sa područja, na primjer, Rumunije, neće morati obrađivati i neželjene podatke. Pažljivim odabirom strategija slanja poruka moguće je ostvariti balans između performansi i fleksibilnosti jer strategija može biti definisana na nivou pojedinačnog kolektora ili izvora podataka.

3.5.3. AGENTSKE KOMPONENTE

Podatke koji su prikupljeni i obrađeni od strane kolektora komunikacioni sistem isporučuje agentskim komponentama. Da bi implementirali podršku za pristup podacima putem SNMP-a, odabran je AgentX protokol i implementacija Jasmin JAX [90]. Osnovni kod za Java klase je generisan upotrebom *smidump* alata iz paketa *net-snmp-tools*. Ovaj pristup ima brojne prednosti od kojih je najvažnija jednostavna i efikasna integracija u postojeća SNMP agentska okruženja jer administratori sistema mogu da zadrže sva rješenja koja su već u upotrebi a tiču se autentifikacije,

autorizacije, itd. Ovaj pristup je i agnostičan po pitanju konkretnog SNMP agenta čija se funkcionalnost proširuje, a kako je riječ o Java rješenju, sam sistem je nezavisan i od operativnog sistema na kojem se izvršava. Kako se AgentX komunikacija može obavljati na više načina, ovakav pristup je pogodan i za rješenja kod kojih se svi agenti izvršavaju na istom serveru ali i za rješenja koja predstavljaju distribuiranu mrežu agenata i podagenata. Jedna potencijalna mana leži u povećanoj kompleksnosti jednoserverskih rješenja jer je moguća degradacija performansi u odnosu na namjenski pisan SNMP agent. Ovaj teoretski problem je detaljnije obrađen u sekciji 3.6.

Nakon što agentske komponente prime podatke, na osnovu njih se računaju vrijednosti pogodne za upis u brojne skalarne i tabelarne unose koje su definisane MIB-om i detaljno su opisane u sekciji 3.3. Ovi podaci su kroz AgentX vezu prema SNMP agentima dostupni ovlaštenim korisnicima sistema.

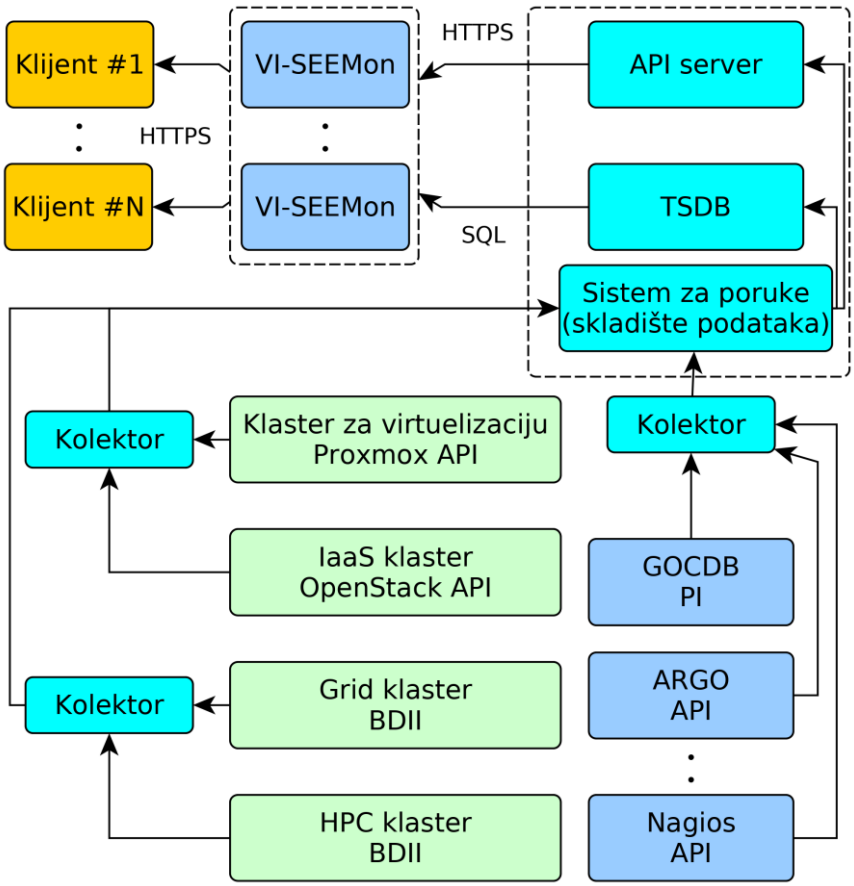
Kako je sistem projektovan i realizovan kao slabo spregnuti sistem, isti princip je primijenjen i na agentske komponente što znači da je moguće imati proizvoljan broj agentskih komponenti. Višestruke agentske komponente mogu se koristiti za horizontalno skaliranje i obradu većeg broja korisničkih zahtjeva u datom periodu ili za ograničavanje podataka koji su vidljivi klijentima određenih agenata, bilo iz praktičnih bilo iz administrativnih razloga.

Korektno funkcionisanje realizovanog sistema je verifikovano korišćenjem za potrebe nadzora kompletne infrastrukture VI-SEEM projekta u periodu dužem od godinu dana bez ikakvih zabilježenih problema u radu i sa podacima koji su odgovarali drugim relevantnim izvorima podataka u projektu (korisnička podrška, obračun potrošnje, lokalni sistemi za nadzor). Kako je Elektrotehnički fakultet Univerziteta u Banjoj Luci u okviru projekta imao ulogu i operatera infrastrukture kroz ETFBL-CC01 klaster, ali i korisnika infrastruktura drugih operatera, sistem je testiran za nadzor privatnih, javnih ali i hibridnih infrastruktura.

3.5.4. ALTERNATIVNI NAČINI PRISTUPA

Slaba sprega komponenata sistema za posljedicu ima i činjenicu da zadatak komunikacije prema korisnicima sistema za nadzor i prezentacije mjernih podataka nije ograničena samo na SNMP agente. Kao primjer proširivosti sistema realizovana je i veb komponenta pod nazivom

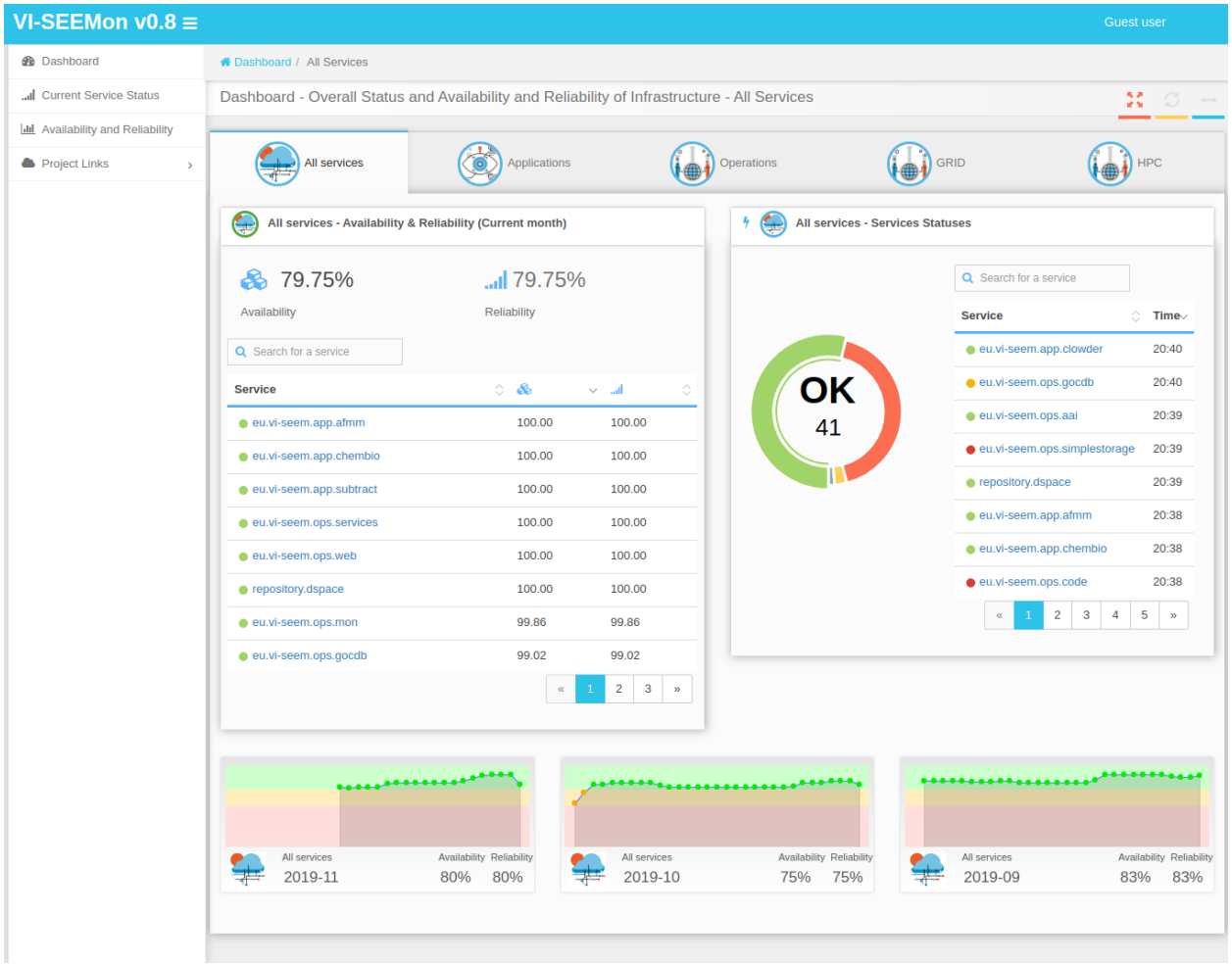
VI-SEEMon. Slika 3.8. prikazuje arhitekturu sistema prilagođenog upotrebi u veb baziranom okruženju. Kolektorske komponente i komunikacioni sistem su u potpunosti nepromijenjeni, a umjesto prethodno opisanih agentskih komponenti sistem podatke čuva u bazi podataka za vremenske unose TSDB (eng. *Time Series DataBase*), dok je pristup podacima sada realizovan putem REST API-ja. Da bi korisnici imali veći komfor u radu, pored API servera je razvijen i veb portal u vidu JavaScript SPA upotrebom React skupa biblioteka [91]. Autentifikacija korisnika je realizovana upotrebom SAML2 standarda i u okviru VI-SEEM projekta je koristila centralizovani provajder identiteta (IdP) dok je sam portal registrovan kao pružatelj usluga (SP).



Slika 3.8. Arhitektura sistema za veb bazirani pristup

Na narednim slikama su prikazani osnovni ekrani i opisana je logika rada veb portal za nadzor. Slika 3.9. prikazuje osnovni ekran portal (eng. *dashboard*) i sastoji se od više elemenata. Sa lijeve strane se nalazi meni koji predstavlja osnovni vid navigacije, dok desni, glavni dio sadrži dio za osnovne informacije, dio za brzo pozicioniranje u hijerarhiji (eng. *breadcrumbs*), te podatke

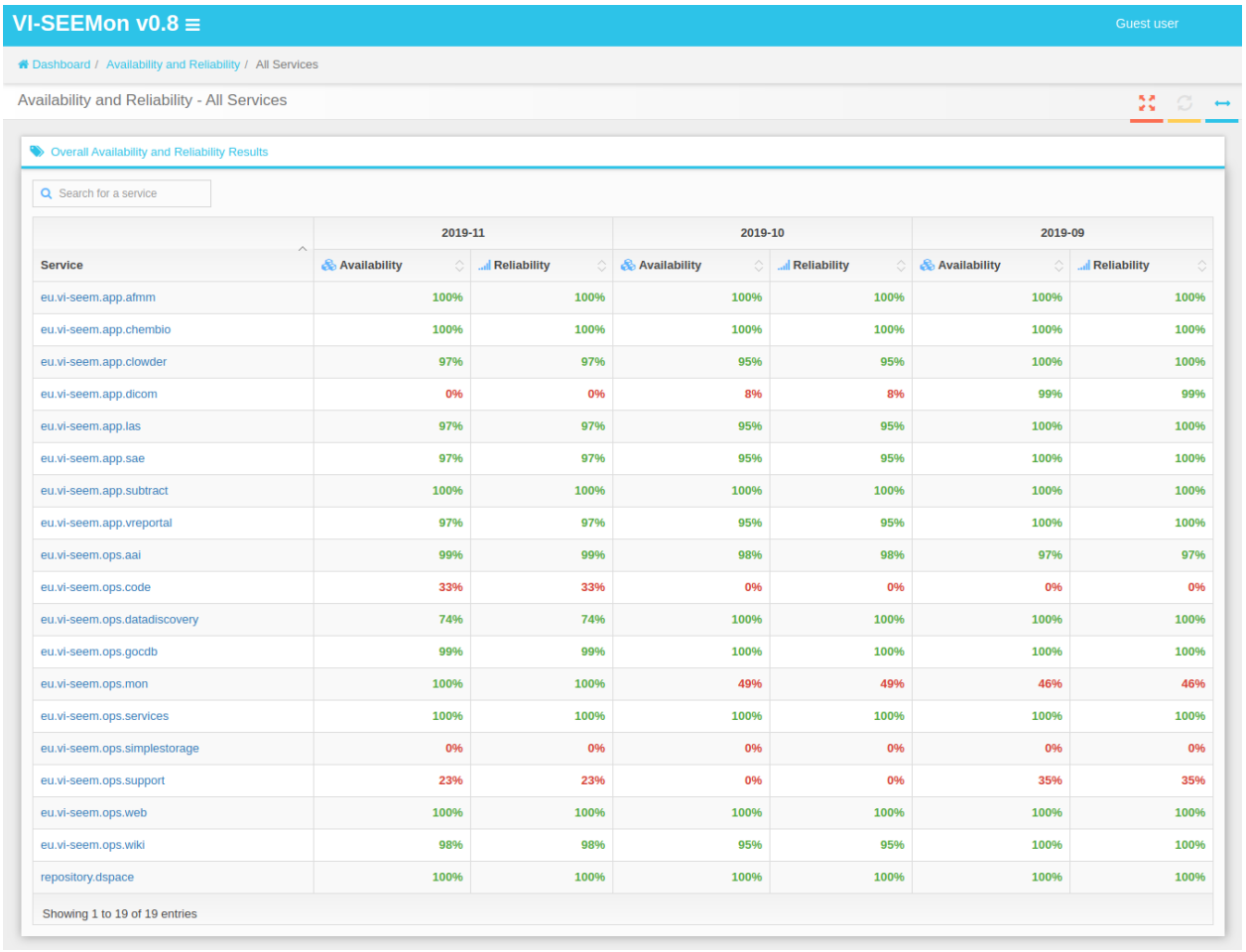
o status, dostupnosti i raspoloživosti instanci servisa. Taj dio interfejsa se sastoji od tri osnovne cjeline: tabele sa podacima o dostupnosti i raspoloživosti, tabele sa najnovijim rezultatima testova, te tromjesečnim prikazom raspoloživosti odabrane infrastrukture. Tabele je moguće pretraživati i sortirati prema željama korisnika, dok će odabirom nekog od unosa sistem preusmjeriti korisnika na odgovarajući ekran sa detaljnim informacijama o odabranoj instanci servisa.



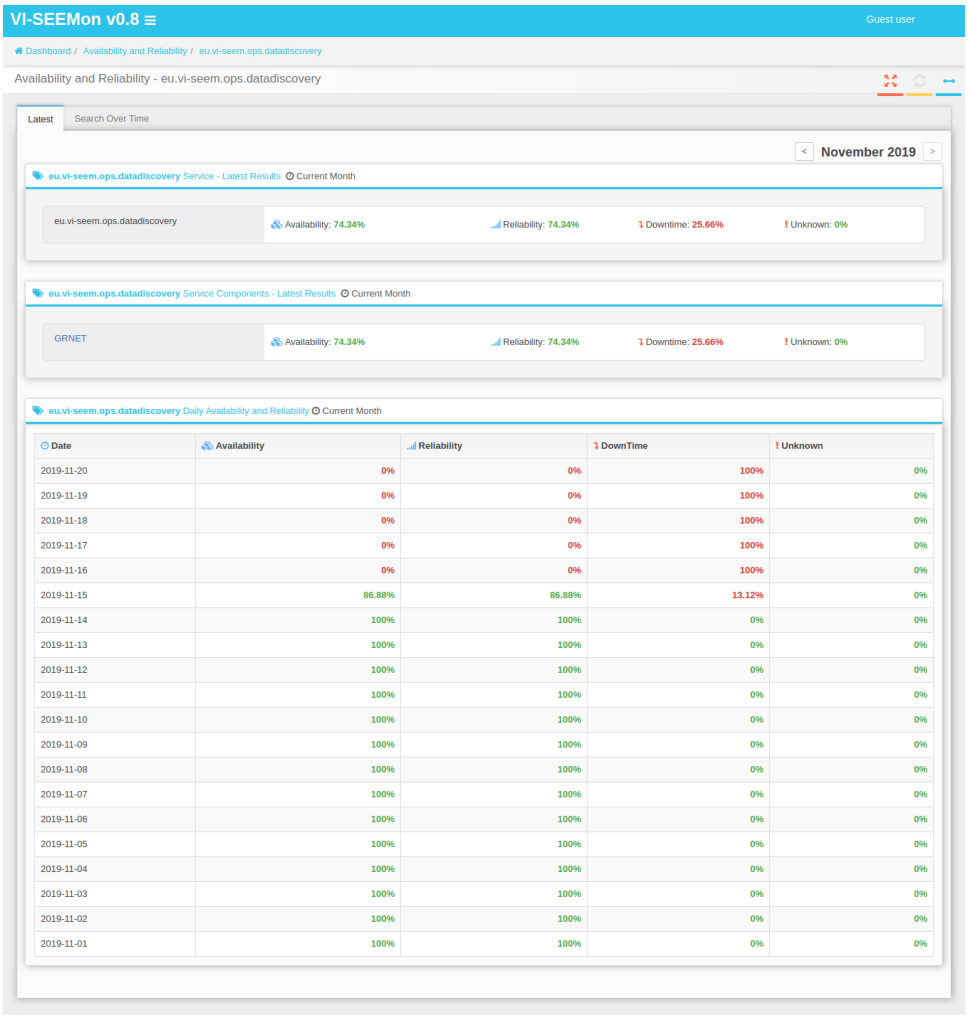
Slika 3.9. Osnovni ekran veb portal za nadzor

Pregled izračunatih dostupnosti i raspoloživosti je moguće pregledati sumarno, za sve instance, kao što je prikazano na slici 3.10. ili za pojedinačnu instancu što je prikazano na slici 3.11. Dostupnost predstavlja udio vremena u kojem je servis funkcionisano korektno, dok raspoloživost uzima u obzir i najavljene periode održavanja sistema. Na primjer, ukoliko je u periodu od 24 časa servis korektno funkcionisao 12 časova, 6 časova proveo u najavljenom periodu održavanja i 6 časova funkcionisao nekorektno, dostupnost bi iznosila 50% a raspoloživost

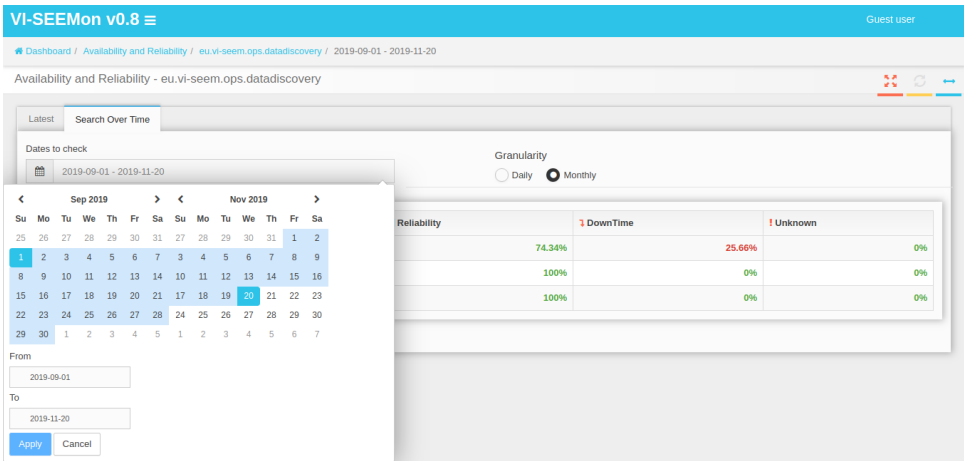
66,67%. Pri pregledu istorijskih vrijednosti za odabranu instancu, moguće je definisati proizvoljan interval i granularnost podataka (dnevna ili mjesečna) što je prikazano na slici 3.12.



Slika 3.10. Pregled dostupnosti i pouzdanosti instanci servisa za prethodna tri mjeseca

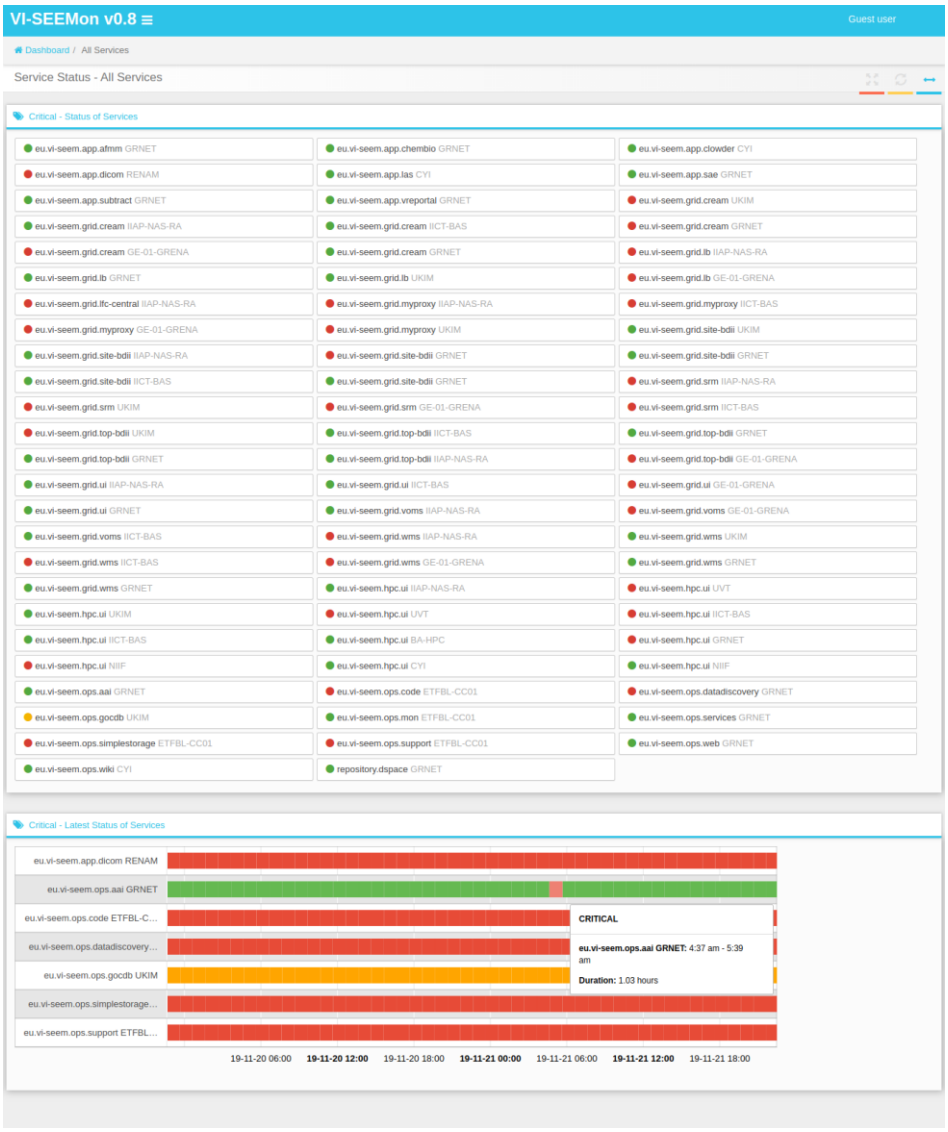


Slika 3.11. Detaljni prikaz dostupnosti i pouzdanosti za odabranu instancu servisa



Slika 3.12. Definisanje vremenskog intervala za prikaz dostupnosti i pouzdanosti

Slika 3.13. prikazuje ekran za prikaz informacija o statusu instanci servisa. U gornjem dijelu ekrana se nalazi lista svih instanci sa nazivom instance i klastera kojem je pridružena, te grafička predstava trenutnog statusa u vidu kruga određene boje (zelena – OK, narandžasta – WARNING, crvena – ERROR). Prelaskom pointera preko unosa moguće je dobiti detaljnije informacije o aktuelnom mjerenju za datu instancu. Donji dio ekrana služi za prikaz instanci koje u posmatranom periodu imaju minimalno jedan zabilježen pad testa. Pored imena instance i pridruženog klastera je iscrtana lenta vremena koja takođe poštuje prethodno definisanu šemu boja. Prelaskom pointera preko lente, moguće je dobiti detaljnije informacije o svakom od zabilježenih statusa, kao što je vidljivo na primjeru sa slike.

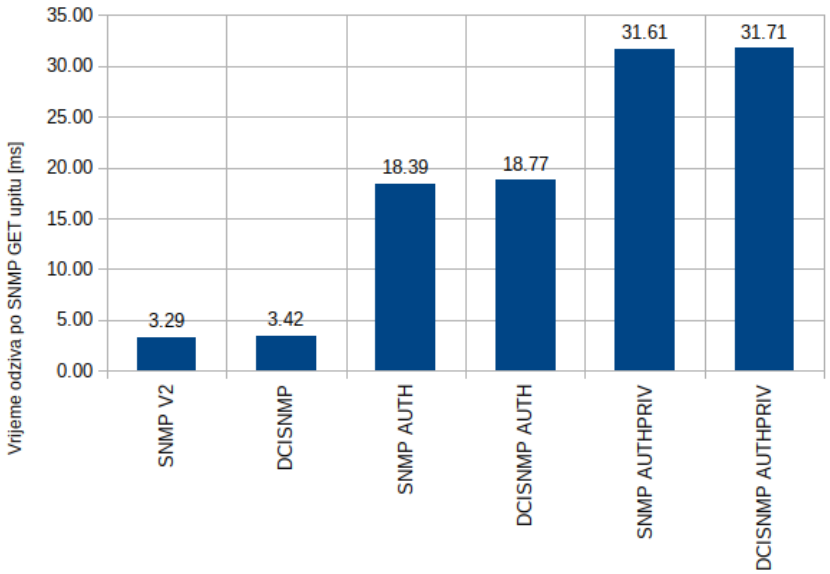


Slika 3.13. Prikaz trenutnog statusa svih instance servisa

3.6. PERFORMANSE SISTEMA

Da bi ustanovili opravdanost izbora AgentX protokola za proširenje funkcionalnosti SNMP agenta za potrebe rada sistema, sistem je testiran u realnom okruženju. Cilj testiranja je bio ispitati da li upotreba AgentX unosi značajnu degradaciju performansi u odnosu na razvoj samostalnog SNMP agenta.

U okviru testiranja su poređena vremena odziva opisanog sistema (DCISNMP) sa vremenima odziva Net-SNMP agenta (SNMPD) u scenariju koji predstavlja server koji koristi jednu nit i odgovara na niz nezavisnih SNMP GET upita klijenta. Mjereno je vrijeme potrebno za obradu 100 sekvencijalnih SNMP GET upita za SNMP u verziji 2 (SNMPv2), SNMP u verziji 3 bez enkripcije (AUTH) i SNMP u verziji 3 sa enkripcijom (AUTHPRIV). Testiranje je izvršeno na serveru HP ProLiant DL360G7 sa jednim Intel Xeon E5649 procesorom na 2.53 GHz sa šest 4 GB 1333 MHz DDR3 RAM modula. Softversko okruženje se sastojalo od operativnog sistema Ubuntu 16.04.3, linux kernela verzije 4.4.0-109 i Net-SNMP verzije 5.7.3+dfsg-1ubuntu4. Slika 3.14. je grafički prikaz dobijenih rezultata testiranja. Tabela 3.3. sadrži numeričke podatke o vremenu odziva po upitu, te standardne devijacije za sve testirane slučajeve.



Slika 3.14. Vrijeme odziva za SNMP GET upite za DCISNMP i SNMPD agente

Tabela 3.3. Vrijeme odziva i standardna devijacija po GET upitu

Vrsta upita	Vrijeme po upitu [ms]	St. dev.
SNMP V2	**3,29**	0,0738
DCISNMP	**3,42**	0,1135
SNMP AUTH	**18,39**	0,2514
DCISNMP AUTH	**18,77**	0,2002
SNMP AUTHPRIV	**31,61**	0,2132
DCISNMP AUTHPRIV	**31,71**	0,1663

Vremena odgovora po GET upitu za SNMPv2 su iznosila 3,29 ms za SNMPD i 3,42 ms za DCISNMP što iznosi povećanje od 3,95%. Vremena odgovora za AUTH pristup su iznosila 18,39 ms za SNMPD i 18,77 ms za DCI SNMP što predstavlja 2,07% veće vrijeme. Za AUTHPRIV su zabilježena vremena od 31,61 ms za SNMPD i 31,71 ms za DCISNMP ili povećanje od 0,32% pri upotrebi AgentX protokola u poređenju sa SNMP agentom pisanim u programskom jeziku C.

Pad performansi je prihvatljiv čak i u najgorem slučaju koji koristi zastarjelu verziju SNMP protokola, dok je za aktuelnu verziju sa uključenim preporučenim sigurnosnim mehanizmima pad performansi u granicama standardne devijacije.

4. KORISNIČKI NADZOR DIJELJENE INFRASTRUKTURE

Degradacija performansi predstavlja ozbiljan problem i za korisnike i za operatere dijeljenih infrastruktura. Jedno od mogućih rješenja je pravovremena detekcija degradacije i pravilno određivanje uzroka iste koje omogućava rekonfiguraciju sistema ili preraspodjelu resursa sa ciljem poboljšanja performansi. Iako i korisnici i operateri imaju isti cilj, eliminaciju degradacije performansi, način na koji pristupaju sistemu i količina podataka kojima raspolažu se znatno razlikuju. U ovom poglavlju je dat pregled tri osnovne vrste dijeljenih infrastruktura, te prikaz sistema koji ima za cilj zatvaranje procjepa u pristupu podacima između operatera i korisnika na način koji je prihvatljiv za sve strane.

Iako operateri infrastrukture imaju samo ograničen uvid u korisničko okruženje, Nemati i Dagenais su pokazali da je i bez kršenja korisničkih prava moguće procijeniti sa velikom preciznošću, trenutno stanje procesora u korisničkoj virtuelnoj mašini [92]. Ova mogućnost je od potencijalno velike važnosti za operatere jer olakšava detekciju kritičnih resursa i degradaciju performansi u korisničkom okruženju. S druge strane, korisnici dijeljenih infrastruktura su izloženi brojnim ograničenjima i nemaju slobodan pristup podacima koji potiču od stvarne fizičke infrastrukture. Oni su tipično ograničeni samo na podatke dostupne iz virtuelizovanog okruženja i nisu svjesni potencijalnih problema izazvanih od strane drugih korisnika iste infrastrukture. Imajući navedeno u vidu, lako je zaključiti da korisnici moraju riješiti mnogo kompleksnije probleme ukoliko žele da na pouzdan način utvrde tačnu lokaciju i razloge degradacije performansi u dijeljenim okruženjima. Studija provedena 2016. godine od strane Rodrigues-a i saradnika u oblasti CC okruženja je identifikovala nadzor nad nižim nivoima infrastrukture, virtuelizacijom i višekorisničkim resursima kao otvorena pitanja koja je potrebno riješiti u kontekstu nadzora IaaS okruženja [6]. Do sličnih zaključaka su došli i autori studije koja je analizirala 21 dostupan sistem za nadzor generičkih i CC infrastruktura [51]. Problem dolazi do izražaja u situacijama u kojima dolazi do degradacije performansi korisničkih resursa usljed dijeljene prirode istih. Ovaj fenomen je nazvan interferencija performansi. Kako operateri imaju pun nadzor nad uređajima koji čine fizičku infrastrukturu, za njih je moguće detektovati ovu pojavu pravilnim odabirom načina nadzora infrastrukture. Detekcija interferencije performansi može biti dodatno zakomplikovana kratkotrajnim poremećajima i činjenicom da različite vrste korisničkih poslova mogu primijetiti

efekte iste interferencije u spektru od zanemarivih do izuzetno značajnih poremećaja funkcionisanja sistema. U radu Pu-a i saradnika je data jedna takva analiza koncentrisana na mrežne performanse u virtuelizovanim računarskim okruženjima pod uticajem raznovrsnih virtuelnih mašina i tipova opterećenja izvršavanih na istom fizičkom serveru [93].

Chen i saradnici su kreirali sistem koji omogućava detekciju i predikciju interferencije performansi u virtuelizovanim okruženjima [94]. U radu je data i detaljna analiza uticaja različitih tipova opterećenja na performanse, od procesorski intenzivnih zadataka, preko mrežnih opterećenja do zadataka dizajniranih da opterete sistem za skladištenje podataka. Autori su pokazali da je izborom poboljšanog raspoređivača virtuelnih mašina moguće postići poboljšanja performansi i do 30% u test okruženju. Amri, Hamdi i Brahimi su ispitivali uticaj sistema za nadzor i detekciju interferencije performansi [95]. U radu su klasifikovali sisteme, između ostalog, i prema ciljnom tipu korisnika sistema te metodu detekcije ili predikcije interferencije. Svi analizirani sistemi namijenjeni korisnicima infrastrukture su koristili pristupe zasnovane ili na redovima čekanja ili na mašinskom učenju. Sistemi su prvenstveno ublažavali efekte interferencije ili kroz poboljšano raspoređivanje VM po serverima ili su detektovali interferenciju na osnovu odstupanja mjerenih performansi sistema u odnosu na dostupne istorijske podatke. Lee, Lee i Eom su pokazali da je pravilnim odabirom rješenja optimizovanog za određene klase problema interferencije performansi, kao što je sistem namijenjen za optimizaciju pristupa trajnoj memoriji, moguće smanjiti kašnjenje do 52% te poboljšati propusni opseg do 59,6% u uslovima interferencije [96].

Upotrebu CC okruženja za naučna izračunavanja su analizirali Exposito i saradnici uz glavni zaključak da je skalabilnost u najvećoj mjeri zavisna od mrežnih performansi virtuelizovanog okruženja [97]. Popularnost upotrebe raznih kontejnerskih tehnologija kao alternativa punoj virtuelizaciji je zasnovana na činjenici da upotreba ovih tehnologija tipično ima mnogo manje efekte na performanse korisničkih okruženja uz potencijalno manji rizik od interferencije performansi. Ruiz, Jeanvoine i Nussbaum su analizirali mogućnost upotrebe kontejnerskih tehnologija za računarstvo visokih performansi i naučna izračunavanja uz zaključak da je upotreba ovih tehnologija opravdana u navedenim oblastima [98]. Xavier i saradnici su pokazali da je čak i u tim uslovima moguća je degradacija performansi do 38% usljed interferencije performansi što ilustruje činjenicu da je problem interferencije izazvan dijeljenom prirodom resursa a ne puka posljedica neefikasne virtuelizacije [99].

Problem degradacije performansi u mrežnoj komponenti IaaS okruženja je analiziran od strane Guo-a i saradnika [100]. Autori u radu predlažu upotrebu kooperativnog pristupa iz teorije igara sa ciljem garantovanja određenog nivoa usluga za sve VM uključujući pravednu raspodjelu preostalog propusnog opsega. Gong, He i Li su pokazali da upotreba generičkih rješenje za optimizaciju mrežnih infrastruktura nije pogodna u IaaS okruženjima, te da je potrebno izvršiti odvajanje konstantnog od dinamičkog opterećenja mreže sa ciljem poboljšanja performansi iste [101].

4.1. KORISNIČKI NADZOR DIJELJENE INFRASTRUKTURE

Kao što je vidljivo iz prethodno navedenih izvora, obim istraživanja pokazuje ozbiljnost problema, ali je i jasno vidljivo da je velika većina napora usmjerena na operatere infrastruktura uz mali broj rješenja okrenutih prema korisnicima dijeljenih infrastruktura. Ovaj pristup je razumljiv jer operateri imaju slobodan pristup podacima potrebnim za korektnu detekciju interferencije performansi, ali i pristup alatima koji omogućavaju predikciju i otklanjanje efekata interferencije u određenim slučajevima. Sve ovo ostavlja korisnike u nezavidnom položaju jer su u potpunosti zavisni od operatera bez stvarnog načina da sami detektuju interferenciju u opštem slučaju. Da bi korisnici bili u stanju da samostalno detektuju interferenciju, neophodno je da imaju određeni nivo pristupa mjernim podacima koji potiču sa fizičkih uređaja koji čine dijeljenu infrastrukturu. Način na koji je to moguće realizovati je predstavljen u nastavku ovog poglavlja.

Nakon analize dostupne literature, poređenja sa prethodnim istraživanjem iz oblasti nadzora infrastruktura [102] i IoT-a [103], te prikupljanja podataka od korisnika i operatera dijeljenih infrastruktura u okviru VI-SEEM [9] i NI4OS-Europe [104] projekata identifikovane su tri sljedeće vrste infrastruktura od interesa za ovaj rad: mrežne infrastrukture, virtuelizovane računarske infrastrukture i računarske grid infrastrukture. Analiza svake od pobrojanih vrsta infrastruktura je data u narednim odjeljcima.

4.1.1. MREŽNE INFRASTRUKTURE

Jedan od čestih scenarija u savremenim računarskim mrežama je obezbjeđivanje veze za korisnike sa kraja na kraj mreže (eng. *port-to-port*). U tekstu će ti portovi biti označeni kao port

P1 i port P2. Korisnici imaju ograničeni pristup podacima sa terminalnih uređaja na svakom od krajeva bez direktnih znanja o topologiji ili operativnim detaljima mreže koja obezbjeđuje vezu između P1 i P2. Ova mreža će u daljnjem tekstu biti označena kao unutrašnja mreža. Imajući u vidu ograničeni uvid u stvarno stanje infrastrukture, jasno je da je problem pravilnog lociranja problema koji nije direktno izazvan od strane korisnika izuzetno težak. U toku istraživanja su definisana tri osnovna izvora problema:

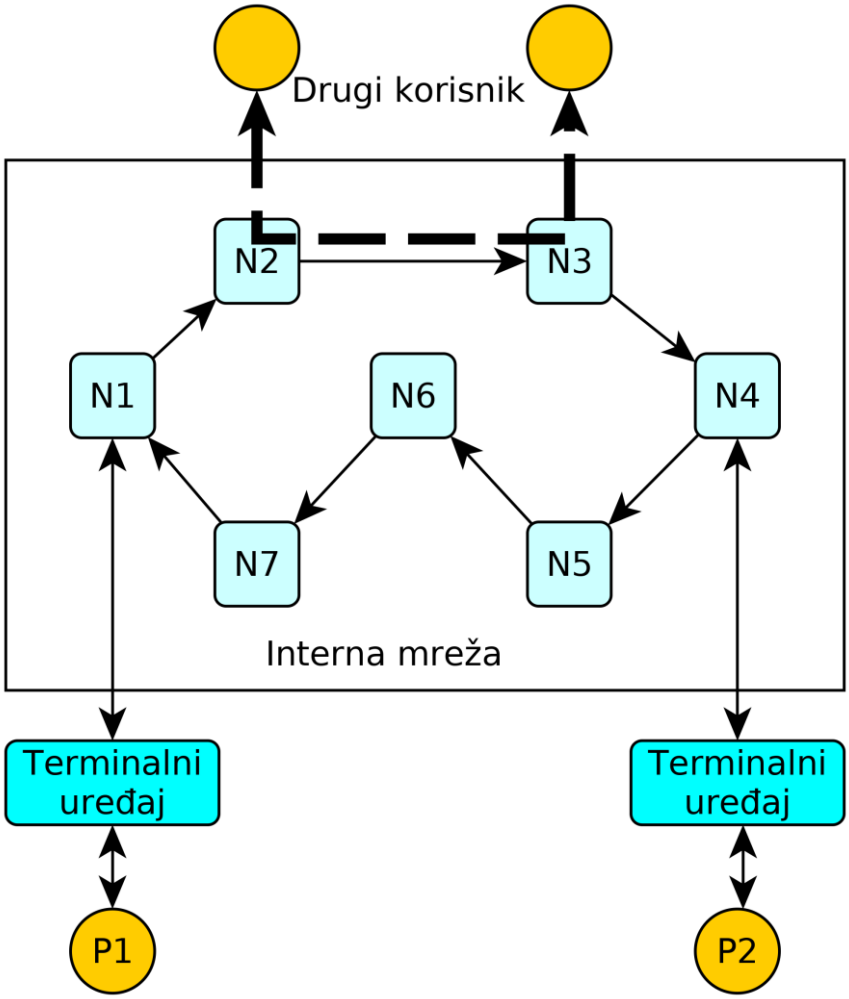
1. problemi izazvani dijeljenom prirodom infrastrukture i aktivnostima drugih korisnika,
2. problemi izazvani asimetričnim tokovima u unutrašnjoj mreži i
3. problemi izazvani dinamičkim promjenama topologije unutrašnje mreže.

Kako je unutrašnja mreža dijeljenog karaktera i tipično nema dovoljno resursa za simultano zadovoljavanje svih korisničkih zahtjeva (eng. *oversubscription* – prebukiranje resursa) nivo performansi koji je dostupan korisniku zavisi i od aktivnosti drugih uređaja koji koriste istu dijeljenu mrežnu infrastrukturu. Sam korisnik nema uvid u stanje unutrašnje mreže i lociranje izvora degradacije performansi nije trivijalan zadatak. Situacija može biti dodatno zakomplikovana kratkotrajnom prirodom i nepravilnim pojavljivanjem poremećaja izazvanih velikim protokom stranih podataka kroz mrežu.

Na slici 4.1. je predstavljen primjer degradacije performansi za korisnika izazvan velikim opterećenjem na mrežnim uređajima N2 i N3 usljed aktivnosti drugog korisnika. Kako korisnik nije u prilici da ima informacije o opterećenju uređaja N2 i N3, nije ni u stanju da pravilno locira problem.

Drugi identifikovani problem vezan za asimetričnu putanju između portova P1 i P2, te P2 i P1 je takođe ilustrovan na slici 4.1. Iako je za korisnika veza između portova P1 i P2 jednostavan direktan link, u stvarnosti ne postoje nikakve garancije o prirodi veze koja može biti izuzetno kompleksna. Na slici je predstavljen slučaj u kom veza između P1 i P2 ima putanju N1-N2-N3-N4 dok povratna veza P2 na P1 ima putanju N4-N5-N6-N7-N1. U kombinaciji sa prethodno opisanim uticajem drugih korisnika na performanse, moguć je kompleksan slučaj u kom je izuzetno teško ili praktično nemoguće korektno locirati problem na osnovu mjerenja koja potiču samo sa terminalnih uređaja.

Oba opisana problema mogu biti dodatno zakomplikovana promjenama u topologiji unutrašnje mreže što je vrlo česta pojava u modernim računarskim mrežama, a naročito je prisutna u softverski definisanim mrežama. Na primjer, povremeni problemi sa uređajem N5 mogu mijenjati putanju između čvorova P2 i P1 između alternativnih putanja N4-N5-N6-N7-N1 i N4-N3-N2-N1. Ovakve promjene putanja u unutrašnjoj mreži mogu unijeti degradaciju performansi za krajnjeg korisnika bez ikakvih, za korisnika vidljivih, promjena u mreži.

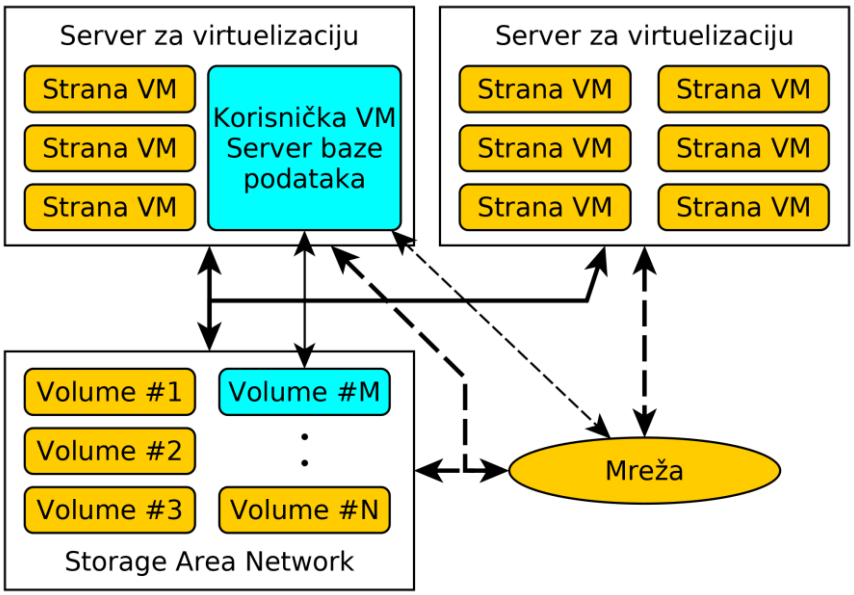


Slika 4.1. Uticaj drugih korisnika i asimetričnih putanja na performanse

Ukoliko bi korisnik imao direktan pristup nadzoru nad svim uređajima koji čine unutrašnju mrežu, bio bi u stanju da otkrije i locira problem. Ovakav pristup bi doveo do pojave velikog broja drugih problema, prije svega sigurnosne prirode, ali bi takođe zahtijevao od korisnika da bude u stanju da vodi računa o topologiji unutrašnje mreže i svim kompleksnim promjenama u njoj.

4.1.2. INFRASTRUKTURE ZASNOVANE NA VIRTUELIZACIJI RAČUNARSKIH RESURSA

Kada govorimo o dijeljenim računarskim okruženjima, gotovo uvijek je riječ o upotrebi virtuelnih mašina ili kontejnera koji se izvršavaju na fizičkom serveru u dijeljenom okruženju. Korisnik treba da bude u poziciji da na efikasan način vrši nadzor dijeljenog okruženja kao i da detektuje interferenciju performansi. Na slici 4.2. je dat primjer korisnika koji izvršava VM sa serverom baze podataka a koja trenutno ima primjetan pad performansi iako su i broj transakcija u sekundi i njihova priroda nepromijenjeni na bilo kakav značajan način. Za ovog korisnika je izuzetno teško otkriti pravi izvor problema, a to je druga VM na istom serveru koja trenutno koristi neuobičajeno velik udio dijeljenih resursa, bio to procesor, mreža ili lokalni disk. Takođe je moguće da je pad performansi izazvan od strane RAID kontrolera fizičkog servera koji je jedan od diskova proglasio neispravnim i trenutno je u procesu oporavka na rezervni disk što dovodi do degradacije performansi svih VM na datom serveru. Virtuelni disk se možda nalazi na SAN-u ili NAS-u u računarskom centru koji mogu biti preopterećeni ili se možda oporavljaju od degradirane RAID konfiguracije što ima negativan efekat na veći broj VM na različitim serverima.



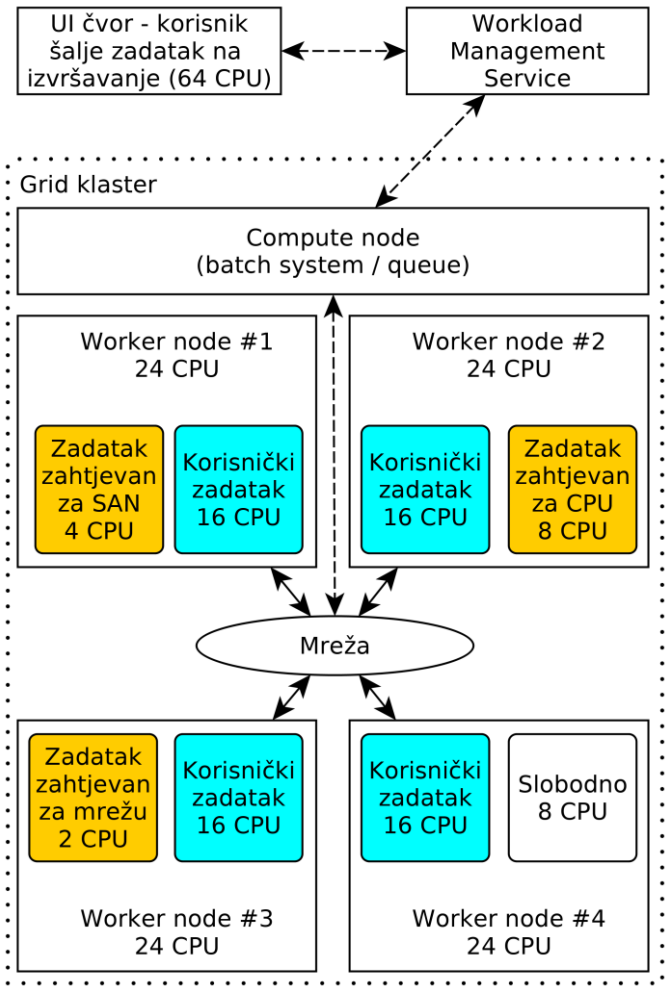
Slika 4.2. Virtuelizovana računarska infrastruktura sa dijeljenim SAN-om i mrežom

Koji god da je razlog pada performansi, podaci potrebni za njegovo lociranje nisu dostupni korisniku. Ukoliko bi korisnik imao direktan pristup serverima bio bi u prilici da locira problem, ali bi to vodilo sličnim problemima kao u slučaju dijeljenih mrežnih infrastruktura.

4.1.3. GRID I KLASTERI VISOKIH PERFORMANSI

Osnovna ideja grid računarstva je jednostavna – omogućiti korisnicima da na jednostavan i efikasan način koriste postojeće distribuirane računarske resurse bez vođenja računa o kompleksnosti infrastrukture. Imajući u vidu navedenu definiciju razumljivo je da postoje brojne interpretacije grid računarstva, ali u okviru ovog rada grid infrastrukture koriste neku formu zajedničkog midlvera koji unificira pristup heterogenim računarskim klasterima koji koriste neki vid sistema za upravljanje korisničkim zadacima.

Primjer jedne takve infrastrukture je dat na slici 4.3. Kada korisnik preda zadatak na izvršavanje u okviru infrastrukture, centralni servis (WMS) bira odgovarajući klaster i predaje zadatak lokalnom sistemu za upravljanje zadacima. Zadatak se postavlja u red za izvršavanje i čeka da se oslobode potrebni resursi za njegovo izvršavanje. Za razliku od prethodno opisanih dijeljenih infrastruktura, ovi sistemi su tipično konfigurisani na takav način da ne dozvoljavaju prebukiranje resursa što pojednostavljuje problem interferencije performansi. Jedan zadatak može biti izvršen na jednom ili više servera (eng. *Worker Node*) pri čemu svaki od servera može imati alociranu različitu količinu resursa za zadatak. U primjeru sa slike, zadatak zahtijeva ukupno 64 procesora (jezgra) na 4 servera, po 16 na svakom, što je primjer klasičnog hibridnog OpenMP+MPI zadatka. Server WN#1 već izvršava drugi zadatak koji opterećuje lokalne diskove, server WN#2 izvršava procesorski intenzivan zadatak, WN#3 izvršava zadatak koji opterećuje mrežu dok je WN#4 izvršava samo korisnički zadatak. Posmatrano sa korisničke strane, zadatak je dobio tražene resurse, ne postoji prebukiranje, te bilo kakva degradacija performansi izazvana zadacima drugih korisnika ne može biti jednostavno otkrivena.



Slika 4.3. Primjer grid okruženja

Sve navedeno može biti sumirano kao nemogućnost korisnika da pouzdano locira problem u kompleksnim dijeljenim infrastrukturama usljed netransparentnosti infrastrukture neophodne za skrivanje njene kompleksnosti. Iako operateri imaju pun pristup svim neophodnim podacima, otkrivanje kratkotrajnih poremećaja u nepravilnim intervalima je izuzetno kompleksan problem, naročito za korisnike.

Imajući sve navedeno u vidu, primjetan je nedostatak rješenja za nadzor dijeljenih infrastruktura koja bi bila usmjerena na korisnike.

4.2. PRIKAZ PREDLOŽENOG RJEŠENJA

U ovom poglavlju je dat prikaz predloženog sistema za nadzor dijeljenih infrastruktura koji zadovoljava korisničke zahtjeve ali i poštuje ograničenja definisana od strane operatera infrastrukture. Nakon prikupljanja relevantnih informacija od korisnika infrastruktura u okviru VI-SEEM projekta, definisana su tri najvažnija zahtjeva korisnika:

1. Pristupačnost – pristup podacima i integracija istih u postojeće sisteme za nadzor moraju biti razumljivi i efikasni.
2. Jednostavnost – podaci moraju apstrahovati i sakriti kompleksnost i heterogenost fizičke infrastrukture.
3. Reprezentativnost – podaci moraju na reprezentativan način predstavljati stvarno stanje nadzirane infrastrukture i omogućiti lociranje potencijalnih operativnih problema.

Nakon analize podataka dobijenih od strane operatera infrastruktura u VI-SEEM projektu, formulisan je skup od tri osnovna ograničenja:

1. Sigurnost – podaci treba da otkrivaju što je manje moguće bilo kakvih osjetljivih informacija o fizičkoj infrastrukturi, u idealnom slučaju otkrivajući minimalni potrebni skup podataka.
2. Nenametljivost – sistem treba da remeti postojeće okruženje u što je manjoj mogućoj mjeri, u idealnom slučaju bez upotrebe dodatnih agenata i bez uvođenja mjerljive degradacije performansi.
3. Upotrebljivost – sistem treba biti upotrebljiv i jednostavan za integraciju u postojeće sisteme za nadzor.

Ispunjavanje navedenih zahtjeva i ograničenja je definisalo potrebu za projektovanjem novog sistema za nadzor koji bi bio u stanju da prikupi podatke iz postojećih izvora, transformiše ih i predstavi korisnicima na standardizovan način koji bi bio široko podržan. Analiza sistema za nadzor koji su bili u upotrebi u projektu je pokaza da svih devet sistema (Cacti, Ganglia, Nagios, Icinga, OpenNMS, Opsview, Pandora FMS, Zabbix, Zenoss Core) podržava SNMP kao izvor podataka za umrežene uređaje. Pregled dostupne dokumentacije i literature snažno sugeriše da velika većina, ako ne i gotovo svi sistemi za nadzor umreženih uređaja imaju podršku za rad sa

SNMP-om. Ovo je razumljivo imajući u vidu činjenicu da je SNMP široko podržan standard koji predstavlja i zvaničan dio IP skupa protokola. Izbor SNMP-a takođe omogućava jednostavnu integraciju u postojeće sisteme i na strani operatera i korisnika jer se podaci dobijeni na takav način ne razlikuju od podataka koje sistemi za nadzor već prikupljaju i analiziraju. Na ovaj način su efikasno zadovoljeni prvi korisnički zahtjev ("pristupačnost") i treće ograničenje operatera ("upotrebljivost").

Jedno moguće trivijalno rješenje bi bilo zasnovano na jednostavnom omogućavanju korisnicima da imaju slobodan pristup potrebnim mjernim podacima i stvarnoj fizičkoj infrastrukturi, ali bi takav pristup bio u koliziji sa "jednostavnošću", jer bi korisnici morali voditi računa o topologiji i specifičnostima infrastrukture, ali i sa "sigurnošću" jer bi na taj način bila publikovana velika količina potencijalno osjetljivih podataka o infrastrukturi.

Osnovna ideja predloženog rješenja se može sažeti u jednoj rečenici: "Predloženi sistem prikuplja neophodne podatke sa fizičke infrastrukture, redukuje ih na jedan generisani reprezentativni čvor za koji računa sve potrebne atribute da bi obezbijedio efikasan nadzor za korisnike". Predloženi sistem redukuje proizvoljno kompleksnu infrastrukturu, koju čini skup nadziranih umreženih uređaja, na jedan vidljiv mrežni čvor koju pruža uvid korisnicima u sumarne nadzorne podatke, te se predstavlja kao generički SNMP uređaj. Prihvatajući ovaj pristup, na jednostavan način je riješen i drugi korisnički zahtjev ("jednostavnost") jer podaci uvijek dolaze sa tačno jednog uređaja bez obzira na kompleksnost same infrastrukture. Kako je nemoguće napraviti manje od jednog čvora za nadzor, ovime je ispunjeno i treće ograničenje operatera da se publikuje minimalna potrebna količina informacija. Treći korisnički zahtjev ("reprezentativnost") je zahtijevao pažljiv odabir načina računanja atributa generisanog čvora. Kako se potencijalno vrlo kompleksna infrastruktura svodi na jedan čvor očigledno je da će doći do određenog gubitka informacija i da će upotreba "reprezentativnog" uređaja unijeti određene kompromise u sistem.

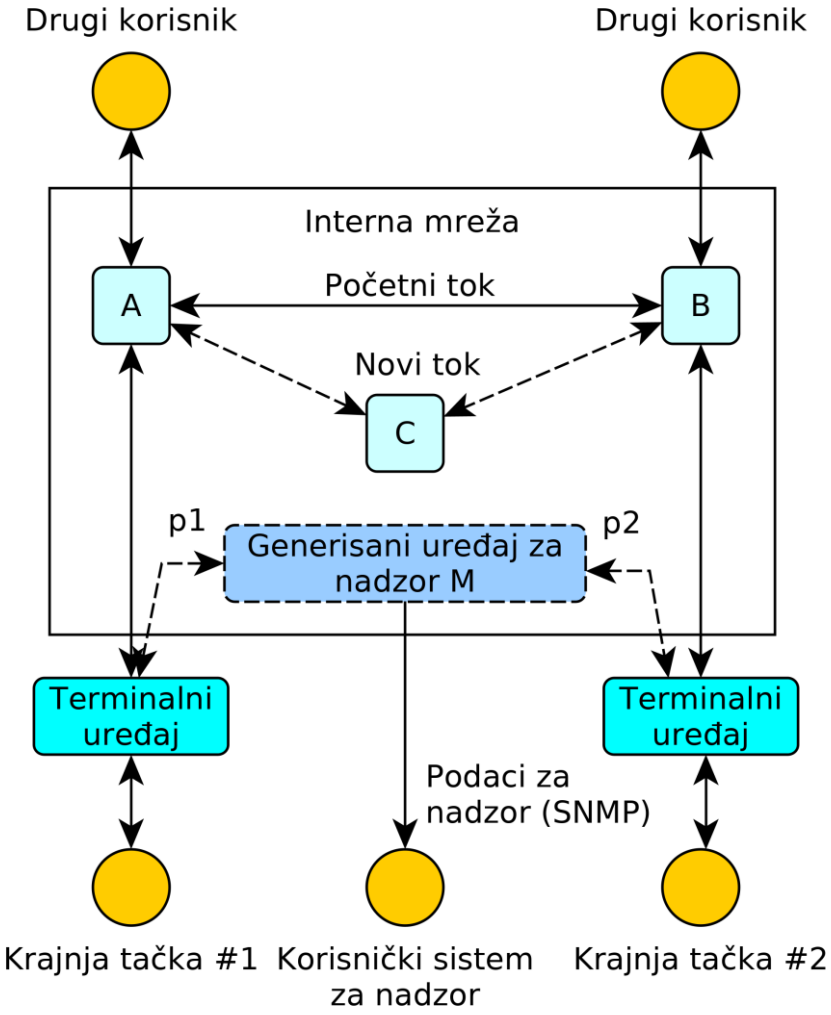
Kod odabira vrste redukcije iz skupa podataka u jednu reprezentativnu vrijednost analizirana su dva pristupa: srednja vrijednost i najgori slučaj. Upotreba srednje vrijednosti podrazumijeva izračunavanje aritmetičke sredine za sva prikupljena mjerenja i odabir najbliže moguće vrijednosti. Iako ovaj pristup može na zadovoljavajući način predstaviti stanje infrastrukture, takođe je moguće sakriti ekstremne vrijednosti koje upravo mogu biti izvor problema koje korisnik pokušava locirati. Upotreba najgoreg slučaja računa izlaznu vrijednost na

osnovu ekstremnih mjerenja nadziranih uređaja, vodeći računa o granicama parametara. Na primjer, ukoliko su posmatrana dva mrežna linka, jedan propusnog opsega od 10 Gbps i trenutnim protokom od 5 Gbps, te drugi propusnog opsega 1 Gbps i trenutnim protokom od 900 Mbps, sistem će ovo vidjeti kao opterećenja od 50% i 90% i odabraće veće relativno opterećenje kao reprezentativno iako prvi link ima veće apsolutno opterećenje. Ovaj pristup će uvijek prikazati stanje posmatrane infrastrukture kao gore nego što jeste, ali će takođe i dati prioritet podacima sa uređaja koji funkcionišu na granicama mogućnosti i možda i izazivaju probleme u funkcionisanju kompletne infrastrukture. Kod umreženih uređaja koji koriste SNMP tipično se koriste dva tipa podataka: *gauge* i *counter*. Prvi tip predstavlja mjerač koji može slobodno da mijenja svoju vrijednost u zadatim granicama, dok drugi tip predstavlja brojač čija je trenutna vrijednost zavisna i od prethodnog stanja i po definiciji je monotono neopadajuća vrijednost. Brojači omogućavaju fleksibilno računanje promjena u vremenu bez ograničavanja na unaprijed definisane vremenske intervale.

4.2.1. DIJELJENE MREŽNE INFRASTRUKTURE

U ovoj sekciji je dat prikaz predloženog rješenja i njegova primjena u oblasti nadzora dijeljenih mrežnih infrastruktura. Predloženo rješenje korisnicima daje ograničeni uvid u stvarno stanje mreže ali na način koji omogućava lociranje potencijalnih problema uzrokovanih dijeljenom prirodom infrastrukture.

Na slici 4.4. su predstavljeni osnovni principi generisanja čvora za nadzor, kao i primjer promjene topologije mreže. Neka se unutrašnja mreža sastoji od tri identična čvora A, B i C koji će biti redukovani na generisani čvor M. Korisnik je povezan na čvorove A i B. Ukoliko nije ispunjena pretpostavka o identičnosti uređaja i oni, na primjer, imaju mrežne portove različitih karakteristika, prije izračunavanje je potrebno izvršiti normalizaciju mjerenih vrijednosti. U početnom stanju podaci direktno saobraćaju između čvorova A i B, te B i A.



Slika 4.4. Generisani čvor za nadzor i promjena u topologiji mreže

Za mjeračke parametre čvor M se može posmatrati kao uređaj za koji je vrijednost mjerača Mg definisana kao maksimum, odnosno minimum u zavisnosti od posmatranog parametra, skupa koji se sastoji od elemenata Ag i Bg koji predstavljaju mjerene vrijednosti odgovarajućih mjerača za uređaje A i B. Kada dođe do promjene unutrašnje mreže usljed pada linka A-B i uspostave tokova A-C-B i B-C-A, vrijednost Mg će jednostavno postati maksimum skupa sa novim elementima Ag, Bg i Cg. Za brojački parametar čvora M, koji će biti označen kao Mc, izračunata vrijednost ne može biti jednostavan maksimum mjerenih vrijednosti Ac i Bc. Da bi korektno izračunali vrijednost, potrebno je prethodno izračunati promjenu parametara Ac i Bc u posmatranom vremenskom periodu, odabrati veću vrijednost i dodati je na prethodnu vrijednost

Mc. Kako direktne vrijednosti brojača, prema dokumentaciji [105], nemaju nikakvo specifično značenje, za početnu vrijednost Mc je moguće odabrati nulu.

Kada dođe do promjene u topologiji mreže i uspostave A-C-B i B-C-A tokova, postavlja se pitanje prethodne vrijednosti parametra Cc. U ovom slučaju postoje četiri pristupa od kojih je potrebno odabrati pogodan. Prvi pristup podrazumijeva kontinualan nadzor svih uređaja bez obzira na to da li su trenutno dio nadzirane unutrašnje mreže ili ne. U ovom slučaju sistem već ima prethodnu vrijednost Cc ali je povećano opterećenje na sistem zbog nadzora svih uređaja. Drugi način je postavljanje vrijednosti parametra Mc na nulu i tretiranje događaja kao diskontinuiteta prema specifikacijama u IF-MIB-u [106]. Treći pristup ignoriše pojavu čvora Cc u prvom koraku i u prelaznom periodu računa Mc na osnovu vrijednosti Ac i Bc. Drugi i treći način žrtvuju tačnost za bolje performanse što može biti prihvatljivo u velikim i kompleksnim infrastrukturama. Četvrti način podrazumijeva modifikaciju komponente za prikupljanje mjerenja na takav način da ne objavljuje samo trenutne vrijednosti parametara nego i promjenu u odnosu na prethodno mjerenje. Na ovaj način se dio obrade podatka prenosi na komponente za prikupljanje ali omogućava značajno jednostavniji proces izračunavanja vrijednosti za generisani čvor jer se izjednačava rad sa mjeračima i brojačima.

Sistem za nadzor mrežnih infrastruktura mora biti u stanju da izađe na kraj i sa situacijama u kojima postoje asimetrični mrežni tokovi i putanja podataka u jednom smjeru nije ista kao povratna putanja. U mreži sa slike 4.4. to bi bili tokovi A-B i B-C-A. Da bi generisani čvor M bio reprezentativan, vrijednosti za prijem za p1 i slanje za p2 se računaju tako da predstavljaju putanju A-B, dok će vrijednosti za prijem za p2 i slanje za p1 biti računati na osnovu putanje B-C-A. Kako čvor M predstavlja vezu između dvije tačke, bez obzira na kompleksnost unutrašnje mreže, uvijek će imati tačno dva porta čije će vrijednosti za ulazne i izlazne parametre biti generisane u parovima (p1.ulaz, p2.izlaz) i (p1.izlaz, p2.ulaz). Vrijednosti iz IF-MIB tabele ifTable se koriste kod 32-bitnih brojača dok su unosi iz tabele ifXTable koriste za 64-bitne brojače i njihova upotreba je poželjna jer 32-bitni brojači nisu pogodno rješenje za savremene brze veze. Pregled relevantnih parametara je dat u tabeli 4.1. Kako se vrijednosti generišu na osnovu pojedinačnih najgorih slučajeva izračunatih na osnovu mjerenja većeg broja uređaja, moguće su situacije u kojima suma svih mogućih parametara u datom vremenskom periodu prevazilazi teoretski moguću vrijednost. Na primjer, moguće je da su reprezentativne vrijednosti *unicast* i *multicast* brojača odabrane sa

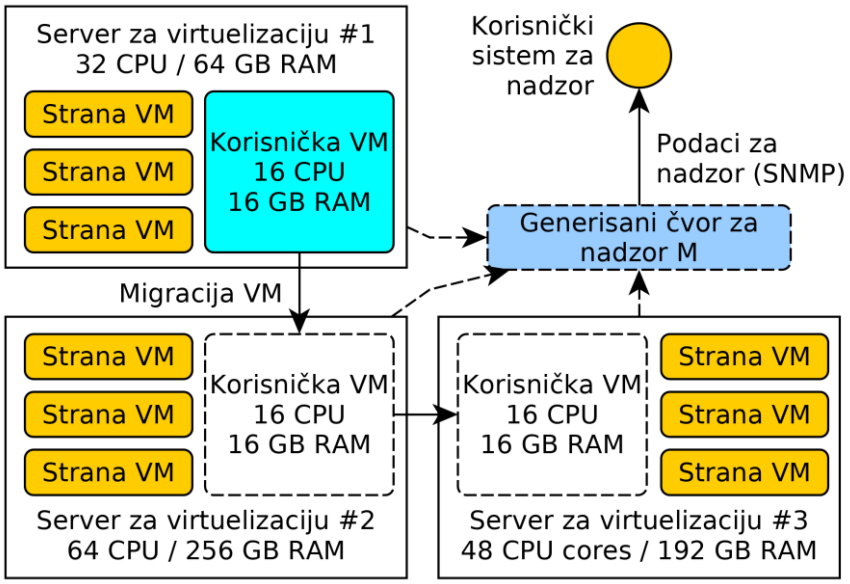
različitih uređaja i da je promjena njihove sume u posmatranom intervalu veća od maksimalne (imajući u vidu brzinu porta) ali je ovakav pristup opravdan jer je osnovna namjena sistema za detekciju problema i ne bi bilo prihvatljivo sakrivanje ekstremnih vrijednosti.

Tabela 4.1. Pregled relevantnih parametara za mrežne infrastrukture

Naziv	**Opis**
ifNumber	Jednak broju interfejsa (2)
ifIndex	Interni indeks interfejsa
ifType, ifMtu, ifSpeed	Definisano konfiguracijom
ifPhysAddress, ifDescr, ifName	Izračunate vrijednosti – definisano konfiguracijom ili slučajno generisana
ifAdminStatus, ifOperStatus, ifLastChange, ifCounterDiscontinuityTime	Izračunate vrijednosti – maksimum posmatranih mjerenja
ifInOctets, ifInUcastPkts, ifInDiscards ifInErrors, ifInUnknownProtos, ifOutOctets, ifOutUcastPkts, ifOutNUcastPkts, ifOutDiscards, ifOutErrors	Izračunate vrijednosti brojača za IF-MIB ifTable
ifHCOutOctets, ifHCOutUcastPkts, ifHCOutMulticastPkts, ifHCOutBroadcastPkts, ifInMulticastPkts, ifInBroadcastPkts ifOutMulticastPkts, ifOutBroadcastPkts, ifHCInOctets ifHCInUcastPkts, ifHCInMulticastPkts, ifHCInBroadcastPkts	Izračunate vrijednosti brojača za IF-MIB ifXTable
ifAlias, ifLinkUpDownTrapEnable, ifHighSpeed, ifPromiscuousMode, ifConnectorPresent	Izračunate vrijednosti – definisana konfiguracijom ili postavljena na podrazumijevanu vrijednost

4.2.2. INFRASTRUKTURE ZASNOVANE NA VIRTUELIZACIJI RAČUNARSKIH RESURSA

Da bi ilustrovali predloženo rješenje i upotrebu u virtuelizovanim računarskim infrastrukturama, posmatraćemo uopšteni slučaj infrastrukture koja se sastoji od tri fizička servera različite konfiguracije. Predloženo rješenje podržava rad sa heterogenim infrastrukturama i živu migraciju virtuelnih mašina redukcijom infrastrukture na jedan generisani nadzirani čvor.



Slika 4.5. Generisani čvor M i živa migracija VM

Na slici 4.5. je prikazan slučaj u kojem VMM vrši živu migraciju korisničke VM sa jednog servera na drugi. Odredišni server ima različitu konfiguraciju u odnosu na izvorni, sa dvostruko više procesorskih jezgara i četiri puta više radne memorije. Kako je jedna od osnovnih karakteristika žive migracije da je "nevidljiva" za korisnika sistema i da unosi zanemarivu pauzu u izvršavanje korisničke VM [107], i od sistema za nadzor se očekuje da bude u stanju da obradi sve promjene do kojih može doći pri ovakvoj iznenadnoj promjeni okruženja. Slično prethodno opisanom pristupu za mrežne infrastrukture, skup nadziranih čvorova (servera) može biti redukovan na jedan reprezentativan generisani čvor M. Generisani čvor se predstavlja korisniku kao obični virtuelizacijski server sa podrškom za SNMP i kao takav je vrlo jednostavan za integraciju u postojeće sisteme za nadzor. Vrijednosti parametara čvora M se računaju na osnovu skupa mjerenja koja potiču sa čvorova na kojima se izvršavala virtuelna mašina u posmatranom

periodu, u ovom slučaju su to serveri 1 i 2. Različite konfiguracije i dostupni resursi su sakriveni od korisnika upotrebom normalizacije jer se sve mjerene vrijednosti prije izračunavanja svode na unaprijed definisane granice.

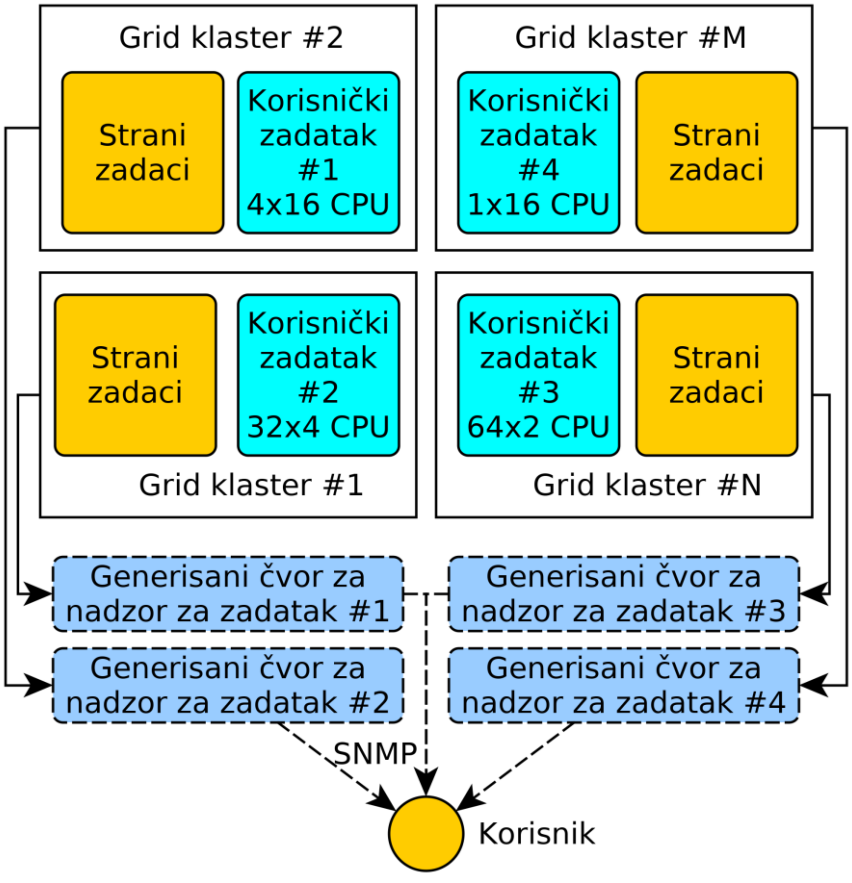
Svi parametri potrebni za nadzor se nalaze u UCD-SNMP-MIB grani *systemStats* i to OID 1.3.6.1.4.1.2021.11 za procesorske parametre, te OID 1.3.6.1.4.1.2021.4 za memorijske. Prikaz relevantnih parametara je dat u tabeli 4.2. Kao što se može vidjeti iz tabele, određeni broj parametara je zavisan od broja procesorskih jezgara u posmatranom serveru. Jedinica "tik" predstavlja vrijeme provedeno u izvršavanju određene vrste opterećenja i tipično postoji 100 "tikova" u svakoj sekundi za svako jezgro procesora. Da bi sistem obezbijedio stabilne parametre pogodne za korisničku upotrebu i u uslovima migracije između različitih servera, potrebno je izvršiti normalizaciju vrijednosti na jedan ili na definisani broj jezgara. Praćenjem vrijednosti korisnik može otkriti da li se razlozi za trenutni pad performansi nalaze unutar korisničke virtuelne mašine ili je pad izazvan procesima na koje korisnik nema direktan uticaj.

Tabela 4.2. Relevantni parametri za infrastrukture zasnovane na virtuelizaciji

Naziv	**Opis**
ssCpuRawUser, ssCpuRawNice, ssCpuRawSystem, ssCpuRawIdle ssCpuRawWait, ssCpuRawKernel, ssCpuRawInterrupt, ssCpuRawSoftIRQ	Izračunate vrijednosti brojača koje predstavljaju broj "tikova" provedenih u datom stanju.
ssIORawSent, ssIORawReceived	Izračunate vrijednosti brojača za broj poslatih, odnosno primljenih U/I blokova
ssRawInterrupts	Izračunate vrijednosti brojača za broj obrađenih prekida
ssRawContexts	Izračunate vrijednosti brojača za broj promjena konteksta
ssRawSwapIn, ssRawSwapOut	Izračunate vrijednosti brojača za broj blokova upotrijebljenih za virtuelnu memoriju

4.2.3. GRID I KLASTERI VISOKIH PERFORMANSI

Da bi analizirali upotrebu u grid i računarstvu visokih performansi posmatrajmo primjer opšte infrastrukture sa apstrahovanim centralnim servisima. U takvom okruženju korisnik može poslati na izvršavanje proizvoljan broj zadataka koji će biti dodijeljeni određenom broju klastera. Moguće je da svi zadaci budu dodijeljeni istom klasteru, možda čak i istom serveru, ali je u opštem slučaju moguće da će svaki zadatak biti izvršavan na drugom klasteru. Svaki pojedinačni zadatak se može izvršavati na jednom ili više servera u zavisnosti od specifikacije koju je definisao korisnik pri slanju na izvršavanje. Svaki korisnik u datom trenutku može nadzirati proizvoljan broj poslova koji se izvršavaju na proizvoljnom broju klastera, sa različitim karakteristikama i brojem dodijeljenih čvorova. Da bi mogao obraditi sve moguće kombinacije okruženja za izvršavanje sistem za nadzor generiše po jedan čvor za nadzor za svaki korisnički zadatak.



Slika 4.6. Primjer generisanja čvora za nadzor po korisničkom zadatku

Na slici 4.6. je predstavljen slučaj korisnika koji je poslao četiri zadatka i koji se izvršavaju na četiri različita klastera u okviru infrastrukture. Serveri koji izvršavaju korisničke zadatke takođe mogu biti opterećeni i od strane drugih zadataka ili sistemskih procesa. Sistem vrši redukciju skupa servera koji izvršavaju određeni zadatak na jedan generisani čvor na sličan način kao u slučaju virtuelizovanih infrastruktura. Jedina razlika je u činjenici da u ovakvom okruženju nema migracije između čvorova tako da je topologija konstantna za vrijeme izvršavanja zadatka. Važno je napomenuti da korisnici ovakvih infrastruktura već imaju pristup sistemima za nadzor koji rade na višem nivou tako da je jedina nedostajuća informacija upravo vezana za mjerenja na nižim nivoima, odnosno na nivou pojedinačnog servera kao što je predstavljeno u prethodnoj sekciji u tabeli 4.2. Sistemi za raspoređivanje korisničkih zadataka su konfigurisani na takav način da nije dozvoljeno prebukiranje resursa što predstavlja još jedno pojednostavljenje u procesu izračunavanja.

4.2.4. ARHITEKTURA PREDLOŽENOG RJEŠENJA

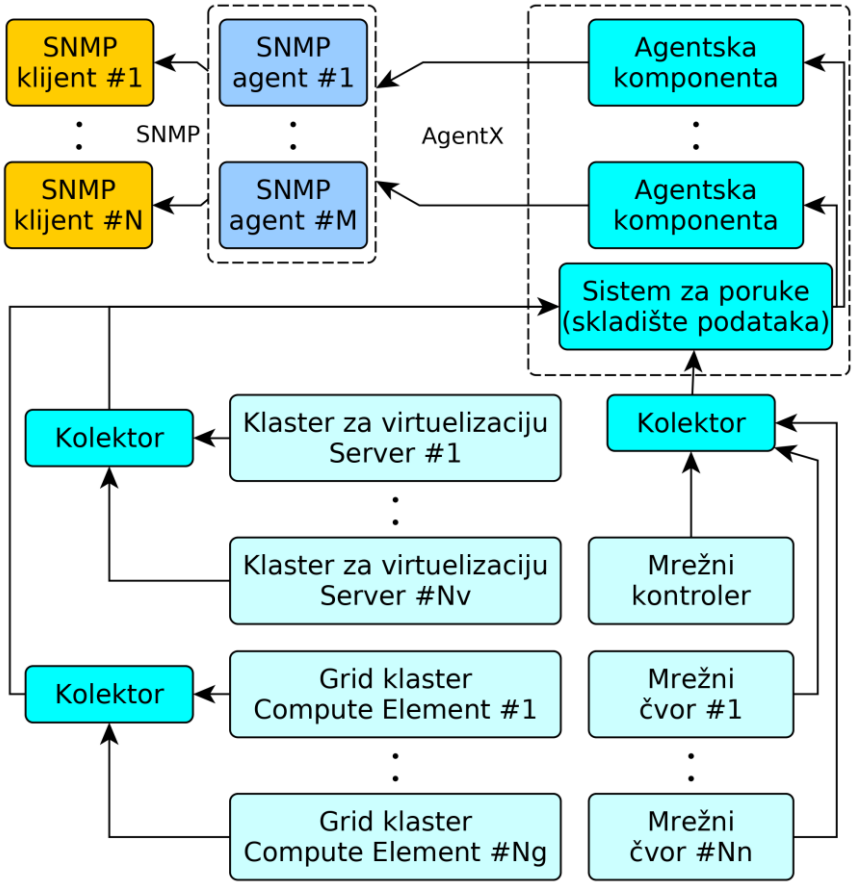
Osim što projektovani sistem mora da zadovolji prethodno navedene zahtjeve i ograničenja, sam sistem mora biti upotrebljiv u širokom spektru upotreba, od jednostavnih instalacija u kojima se sve izvršava na jednom serveru do potpuno distribuiranog okruženja koje se prostire u više administrativnih domena. Baziranje arhitekture sistema oko centralne tačke za razmjenu podataka omogućava tranziciju prema kompleksnijim topologijama, a pravilan izbor rješenja eliminiše centralnu kritičnu tačku. Sa ciljem povećanja fleksibilnosti sistema, sam sistem ne zadaje dodatne uslove za sistem za razmjenu podataka i može da ga koristi i kao jednostavno skladište podataka. Iako postoje brojne tehnologije koje se mogu koristiti u ovom slučaju, dvije najčešće korišćene su sistemi za razmjenu poruka i ključ-vrijednost sistemi.

Pregled arhitekture sistema je dat na slici 4.7. Osnovne komponente sistema su:

- kolektorske komponente koje prikupljaju podatke sa uređaja koji čine infrastrukturu,
- agentske komponente koje izračunavaju vrijednosti za generisane čvorove i
- sistem za razmjenu poruka koji povezuje sve druge komponente.

Osnovna arhitektura sistema je kompatibilna sa arhitekturom sistema za nadzor opisanom u poglavlju 3.4. Osnovne razlike su u vrsti podataka koji se prikupljaju, kao i u načinu izračunavanja i prezentacije podataka namijenjenih korisnicima sistema. Dok sistem za nadzor implementira novi MIB i prezentuje iste podatke za sve korisnike, jer se radi o podacima na nivou kompletne infrastrukture, sistem predstavljen u ovoj sekciji pruža korisnicima uvid u dijeljenu fizičku infrastrukturu na kojoj se izvršavaju korisnički servisi. Činjenica da postoji visok nivo kompatibilnosti između dva opisana sistema je iskorišćena za kreiranje jedinstvene aplikacije koja može vršiti višestruke uloge istovremeno.

U nastavku rada je dat detaljan opis načina funkcionisanja sistema.



Slika 4.7. Arhitektura predloženog rješenja

4.2.5. PRIKUPLJANJE PODATAKA

Da bi sistem mogao da ispuni postavljene zahtjeve, neophodno je da ima pristup sljedećim vrstama podataka: mjerne podatke sa uređaja i informaciju o skupu uređaja koje redukuje u generisani čvor. Kako bilo koji nadzirani uređaj može biti dio više skupova za redukciju, zasebno prikupljanje podataka sa svakog uređaja prema pripadnosti određenom skupu bi unijelo bespotrebno dodatno opterećenje na sistem i infrastrukturu što bi bilo u koliziji za ograničenjem "nenametljivosti". Prikupljanje podataka u bitno različitim vremenskim trenucima takođe komplikuje proces analize prikupljenih mjerenja. Navedeni problemi mogu biti riješeni tako što će svaki nadzirani uređaj biti kontaktiran samo jednom u datom intervalu, a podaci će biti distribuirani do svih zainteresovanih strana upotrebom sistema za razmjenu poruka. Ovaj pristup obezbjeđuje konstantno opterećenje na infrastrukturu bez obzira na broj generisanih čvorova ili korisnika koji koriste sistem.

Kao što je ranije navedeno u poglavlju 4.2.1, kolektorske komponente osim direktno posmatranih vrijednosti publikuju i brzinu njihove promjene u posmatranom vremenskom intervalu. Na taj način je pojednostavljeno računanje parametara generisanog čvora i eliminisana je potreba za dva susjedna mjerenja za brojačke vrijednosti. Ovaj pristup takođe omogućava i upotrebu jednostavnih ključ-vrijednost sistema jer su sva potrebna mjerenja dostupna u svakom trenutku i nema potrebe za praćenjem prethodnih vrijednosti. Jedino ograničenje je organizovanje ključeva na takav način da emuliraju proces rutiranja prisutan kod sistema za razmjenu poruka. Iako se za opšti slučaj preporučuje upotreba mreže brokera za razmjenu poruka, striktno govoreći, nema potrebe za kompleksnim rješenjima i sistem može da funkcioniše i kao monolitni proces koji koristi memorijske strukture za razmjenu podataka.

Alternativa ovom pristupu je slanje samo jednog mjerenja ali na takav način da svaki klijent koji prima poruke uvijek ima pristup do dva najnovija mjerenja i može sam da izračuna promjenu u vremenu. Iako je ovakav način rada izvodljiv, on unosi značajna ograničenja za sistem za razmjenu podataka i praktično eliminiše jednostavnija rješenja.

Da bi sistem redukovao podatke sa više uređaja na jedan generisani čvor, neophodno je da ima podatke o tome koji uređaji čine izvorni skup. Za slučaj mrežnih infrastruktura, taj podatak je moguće dobiti na više načina. Za centralno upravljane ili softverski definisane mreže ti podaci već

postoje u sistemu tako da problem postaje samo napisati komponentu koja će navedene podatke prikupiti i transformisati u upotrebljiv oblik. Ukoliko ovaj pristup nije moguć, potrebni podaci mogu biti izračunati na osnovu podataka koje uređaji koriste za rutiranje a kojima je moguće pristupiti preko IP FORWARD MIB-a [108]. Proces izračunavanje počinje određivanjem čvora u središtu mreže i pronalaženjem svih čvorova koji su direktno vezani na njega. Tako pronađeni čvorovi se rekurzivno ispituju dok svi čvorovi i njihove veze nisu pronađeni. Ukoliko mreža funkcioniše na drugom sloju i nije moguće koristiti tabele za rutiranje, moguće je topologiju odrediti na osnovu podataka prisutnih u tabelama za prosljeđivanje [109] prema definicijama u BRIDGE MIB-u [110]. Navedeni pristup je sličan prethodno opisanom uz drugačije računanje susjedstva jer više nema tabela rutiranja već se moraju porediti unosi iz dotldTpFdbAddress tabela uređaja.

U virtuelizovanim računarskim infrastrukturama, imajući u vidu veliki broj različitih platformi, nije moguće pronaći jedan standardizovan izvor podataka koji bi bio upotrebljiv u svim situacijama. Međutim, kako se svaka virtuelna mašina izvršava na tačno jednom serveru u datom trenutku, skup će se sastojati od jednog uređaja u normalnom radu ili od dva uređaja ukoliko je u posmatranom intervalu došlo do migracije VM na drugi server. Uz ovo ograničenje zahtjevi sistema se redukuju na informaciju o tome koji server trenutno izvršava koju virtuelnu mašinu što je podatak koji je dostupan za sve u ovom radu analizirane platforme.

Situacija je mnogo jednostavnija u grid okruženjima jer su ona od početka dizajnirana na takav način da su sve potrebne informacije o okruženju u kojem se izvršavaju korisnički zadaci dostupne i operaterima i korisnicima. Čak i u situacijama u kojima nije moguće dobiti potrebne informacije od centralnog servisa, sami sistemi za raspoređivanje zadataka u klasteru ih mogu obezbijediti.

4.2.6. INTERNI MODEL I ORGANIZACIJA PODATAKA

Interni model sistema se sastoji od skupa veza između generisanih čvorova (konteksta), fizičkih uređaja, te mjernih podataka prikupljenih sa fizičke infrastrukture. Mjerenja od interesa za ovaj sistem za nadzor su ili mjerači ili brojači. Kada govorimo o mjeračima, oni mogu biti ili nezavisni od konfiguracije i resursa, kao što je postotak realnog vremena proveden u izvršavanju

određene vrste poslova, ili zavisni od njih, kao što bi bio broj jezgro-sekundi provedenih u izvršavanju neke vrste posla. Mjerači koji su zavisni od resursa se normalizuju na jedno procesorsko jezgro za računarska okruženja, odnosno na 1 Gbps link u slučaju mrežnog okruženja. Kod brojača se prati brzina promjene u standardizovanom periodu vremena, a eventualna normalizacija se vrši prema istim pravilima kao u slučaju mjerača.

Agentska komponenta prati podatke za svaki kontekst u strukturi koja blisko prati relevantne MIB dokumente. Za procesorska okruženja model prati upotrebu procesora prema stanjima (eng. *user, nice, system, idle, wait, kernel, interrupt, soft IRQ, steal, guest*, te *guest nice*) kao i broj poslatih i primljenih blokova od U/I uređaja. Za mrežna okruženja, model se sastoji od mrežnog uređaja sa dva mrežna porta, a sistem prati dva potencijalno nezavisna skupa čvor-port parova za svaku od posmatranih putanja, kao što je navedeno u poglavlju 4.2.1. Model efektivno predstavlja dva mrežna interfejsa u odgovarajućoj ifXTable strukturi generisanog čvora.

Osim navedenog, agentske komponente rade i sa internim strukturama podataka poput naziva konteksta, vremena prikupljanja svakog mjerenja koji su neophodni za korektno funkcionisanje sistema. Kako mjerni podaci u opštem slučaju mogu biti prikupljeni u različitim trenucima i sa različitom frekvencijom, agentska komponenta izvršava internu petlju koja ima tri funkcije:

1. Pronalaženje najgoreg slučaja za svaku od posmatranih vrijednosti u skupu trenutnih mjerenja.
2. Ažuriranje stanja internog modela za svaki od praćenih konteksta.
3. Uklanjanje zastarjelih podataka.

Početne vrijednosti za sve brojače se postavljaju na nule i nakon toga se u svakom prolazu računa inkrement i nova vrijednost prema izračunatom najgorem slučaju. Kako mjerači ne zavise od prethodnih vrijednosti, sam izračun je jednostavniji i svodi se na normalizaciju vrijednosti prema ranije opisanom pristupu.

U zavisnosti od resursa koji korisnik želi nadzirati, agentska komponenta će se predstavljati na odgovarajući način i generisaće vrijednosti koji su dio željenog MIB-a. Jedan agentski proces mora biti u stanju da opsluži veći broj korisnika obezbjeđujući podatke sa više generisanih čvorova, uključujući i situaciju u kojoj jedan korisnik traži podatke sa više generisanih uređaja od

istog agenta. Jedno rješenje bi bilo dodjeljivanje više IP adresa ili portova svakom agentu ali na taj način se otvaraju brojni problemi u eksploataciji. SNMPv3 ima podršku za mehanizam konteksta koji omogućavaju da isti agent nudi klijentu različite podatke u zavisnosti od toga koji je kontekst naveden u upitu. Konteksti predstavljaju skupove upravljanih informacija (eng. *managed information items*) koje dijele isti opseg definisan parametrima contextEngineID i contextName [111]. Agentske komponente vraćaju odgovarajuće podatke na osnovu mapiranja specificiranog konteksta na jedinstveni identifikator generisanih uređaja.

U oblasti virtuelizacije je široko rasprostranjena upotreba UUID-ova za označavanje i virtuelnih mašina i srodnih resursa tako se upotreba UUID-a za generisane čvorove sama nameće. U oblasti grid računarstva svaki korisnički zadatak već ima jedinstveni identifikator tako da je izbor odgovarajućeg naziva konteksta i u ovom slučaju jednostavan. Ovakvim izborom identifikatora je riješen i problem informisanja korisnika o kontekstima i njihovom povezivanju na generisane uređaje jer korisnici već imaju pristup vrijednostima identifikatora.

Kada je riječ o mrežnim infrastrukturama ne postoji široko podržan standard ili politika koja bi omogućila jednostavno i uniformno generisanje identifikatora. Ukoliko se koristi centralno upravljana mreža, sistemi često identifikuju linkove pomoću UUID-ova što je kvalitetno i jednostavno rješenje ukoliko je izvodivo. Ukoliko se koriste tabele rutiranja ili prosljeđivanja, jedna mogućnost je definisanje konteksta na osnovu parova krajnjih adresa koje su pristupačne korisnicima. Ovako generisan kontekst je jednostavan za korisnika ali je takođe i relativno jednostavan za pogoditi za zlonamjerne strane.

4.2.7. SIGURNOSNI ASPEKTI

Kada govorimo o sigurnosti, od značaja su dva osnovna sigurnosna aspekta sistema: sigurnost SNMP-a i curenje informacija. Najveći problem vezan za upotrebu SNMP-a u praksi leži u činjenici da su zastarjele verzije protokola SNMPv1 i SNMPv2c i dalje u širokoj upotrebi. Navedene verzije podržavaju samo osnovne sigurnosne mehanizme i njihova upotreba omogućava širok spektar napada za zlonamjerne strane. Trenutna verzija protokola je SNMPv3 i, kao što je ranije navedeno, ona podržava upotrebu TLS (eng. *Transport Layer Security*) i DTLS (eng. *Datagram Transport Layer Security*) kao dio transportnog podsistema protokola. Što se tiče

heširanja pristupnih lozinki podržani su HMAC (eng. *Hash-based Message Authentication Code*) i SHA-2 (eng. *Secure Hash Algorithm 2*) autentifikacioni protokoli. Kako je riječ o *stateless* protokolu koji ne implementira *challenge-response handshake*, sam protokol je podložan *brute force* i rječničkim napadima tako da je izbor kvalitetnih pristupnih lozinki od velike važnosti.

Sam sistem je predviđen za upotrebu kroz AgentX sistem za proširenja funkcionalnosti već postojećih agenata i ne uvodi nikakva posebna ograničenja za već implementirane SNMP servere. Operater sistema ima punu slobodu odabira odgovarajućeg sigurnosnog mehanizma (ili kombinacije više mehanizama, prema zahtjevima operativnog okruženja. Ukoliko je potreban viši stepen zaštite, moguća je upotreba VACM (eng. *View-based Access Control Model*) i IPsec (eng. *Internet Protocol Security*) sigurnosnih mehanizama.

Važno je napomenuti da sam sistem za nadzor nema nikakve mogućnosti da upravlja umreženim uređajima koji čine infrastrukturu i sa stanovišta infrastrukture predstavlja samo prijemnika podataka čije slanje inicira infrastruktura što u velikoj mjeri ograničava mogućnosti koje bi imao potencijalni napadač. Ukoliko napadač ostvari korisnički pristup sistemu i u stanju je da pristupi svim podacima za nadzor i dalje su izuzetno ograničene praktične mogućnosti zloupotrebe, jer je nemoguće odrediti tačna mjerenja sa uređaja na osnovu kojih se izračunavaju parametri generisanih čvorova. Jedini identifikujući parametar je jedinstveni identifikator koji je u opštem slučaju pseudoslučajni broj i ne nosi nikakve specifične informacije. Napadač nije u stanju ni da dobije podatke o topologiji infrastrukture jer je sam sistem dizajniran na takav način da apstrahuje kompleksnost nižih nivoa što znači da ni ne publikuje informacije o topologiji infrastrukture.

Pravilno konfigurisan sistem koji poštuje važeće sigurnosne preporuke predstavlja izuzetno mali rizik po pitanju pristupa infrastrukturi dok je i problem curenja informacija sveden na minimum.

4.2.8. POREĐENJE SA DRUGIM RJEŠENJIMA ZA DETEKCIJU INTERFERENCIJE PERFORMANSI

Kada govorimo o sistemima koji implementiraju detekciju interferencije performansi važno je razlikovati sisteme koji su namijenjeni operaterima od sistema okrenutih korisnicima.

Sistemi namijenjeni operaterima imaju slobodan pristup svim mjernim podacima i obično su projektovani na takav način da računaju na mogućnost pojave interferencije i da je unaprijed otklanjaju preraspoređivanjem virtuelnih mašina na druge servere. Na primjer, Cloudscope sistem [94] automatizuje proces predikcije interferencije i migracije virtuelne mašine na podopterećene servere. Sistemi usmjereni na operatere opisani u dostupnoj literaturi ne podržavaju mrežne i grid infrastrukture.

Sistem Heifer razvijen od strane Xua, Liua i Jina koristi podatke i sa stvarnih servera i iz virtuelnih mašina da bi otkrio interferenciju performansi [112]. Autori u radu ističu probleme koji nastaju kao posljedica heterogenosti fizičke infrastrukture i uticaja koje ona ima na virtuelne mašine. Opisani sistem podržava samo rad sa virtuelizovanim računarskim okruženjima. Amannejad, Krishnamurthy i Far su kreirali sistem, zasnovan na mašinskom učenju, koji omogućava detekciju interferencije performansi za veb bazirane aplikacije [113]. U okviru rada je predstavljen CRE (eng. *Collaborative Response time Estimation*) modul koji poredi stvarno izmjereno vrijeme odziva sa referentnim vrijednostima i na osnovu odstupanja od prethodno naučenog modela može da detektuje interferenciju. Da bi sistem pravilno funkcionisao, neophodno je da su prethodna mjerenja sistema upotrebljiva kao referenca i za buduća mjerenja što podrazumijeva statičnost sistema bez značajnijih promjena. I ovaj sistem je ograničen na jednu vrstu korisničkih aplikacija i ne podržava rad sa mrežama ili grid okruženjima.

UIE (eng. *User-centric Interference Estimation*) je sistem razvijen od strane Javadija i saradnika i koristi teoriju redova čekanja [114] da bi modelovao sistem i zauzeće resursa na osnovu prethodnih mjerenja [115]. Osim što omogućava detekciju, sistem može da izvrši i procjenu uticaja interferencije na performanse korisničke aplikacije. Da bi sistem funkcionisao na korektan način, neophodno je prethodno prikupljanje referentnih podataka posmatranjem rada aplikacije u određenom vremenskom periodu [116]. Iako ne zahtijeva nikakve posebne unose od strane operatera, sistem je ograničen na aplikacije koje je moguće modelovati na odgovarajući način.

Joshi, Raj i Janakiram su razvili sistem Sherlock namijenjen detekciji interferencije performansi u kontejnerskim okruženjima [117]. I ovaj sistem funkcioniše na principu definisanja internog modela korisničke aplikacije i otkrivanju odstupanja između stvarnih mjerenih podataka i rezultata simulacije internog modela. Iako autori ističu da je vremenski period neophodan za obučavanje sistema relativno kratak, sistem i dalje zahtijeva dovoljno reprezentativnih podataka

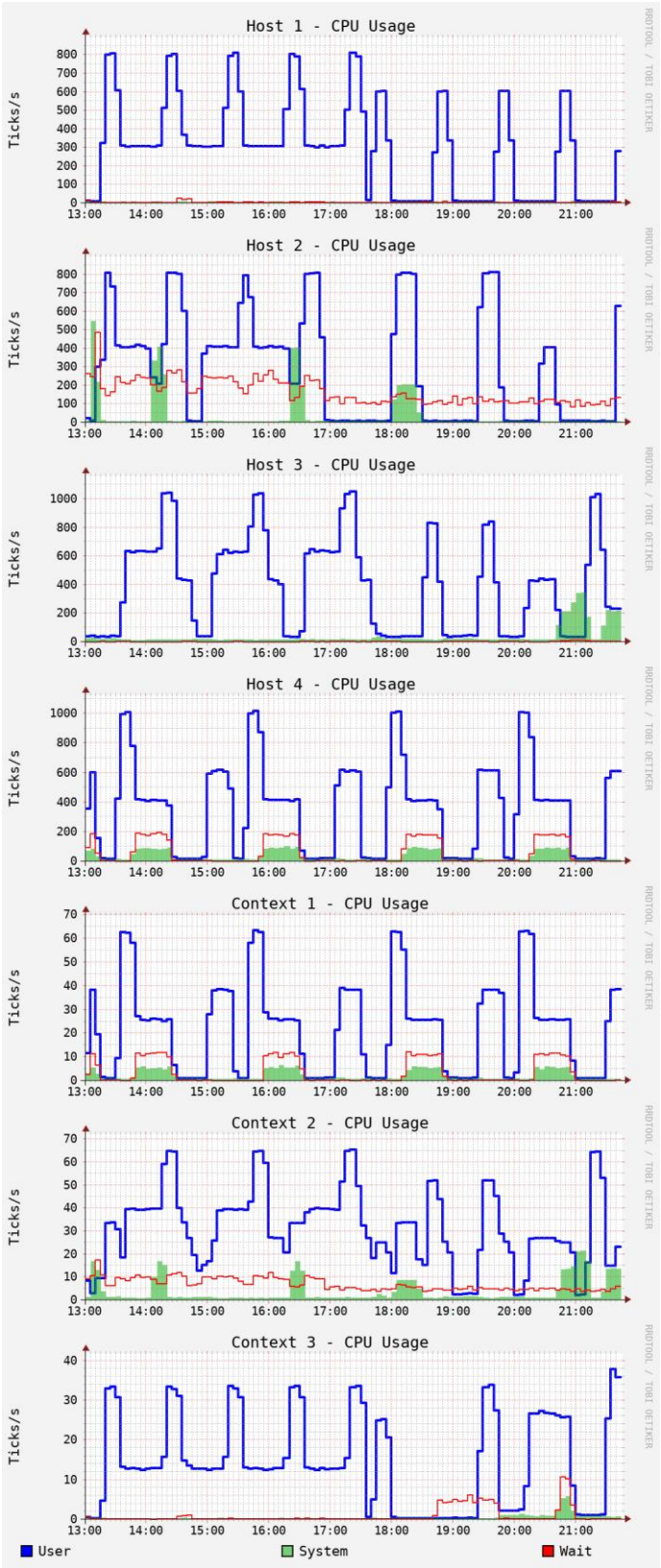
da izračuna idealnu operativnu krivu, te krivu granice interferencije da bi mogao funkcionisati na korektan način.

Za razliku od navedenih rješenja, sistem opisan u ovom radu ima pristup i podacima sa fizičkih uređaja koji čine infrastrukturu, ali je prvenstveno okrenut korisnicima infrastrukture. Nijedan sistem opisan u literaturi ne podržava na uniforman način tri različite vrste infrastruktura niti je na jednostavan način ugradiv u postojeće sisteme za nadzor upotrebom SNMP-a. Važno je napomenuti i da je bez pristupa podacima sa fizičkih uređaja proces detekcije interferencije ograničen na podatke dostupne iz virtuelizovanog okruženja i mora biti zasnovan na nekom vidu profilisanja ili obučavanja modela na osnovu istorijskih podataka. Razlog za ovakvu podjelu je djelimično i nepovjerenje operatera infrastrukture prema korisnicima, jer nije postojao jednostavan i siguran način da se korisnicima obezbijedi pristup relevantnim podacima bez uvođenja dodatnih kompleksnosti i potencijalnih sigurnosnih problema u sistem. Pristup opisan u ovom tekstu rješava upravo taj problem i omogućava da korisnici imaju uvid u podatke iz oba izvora što bitno olakšava proces detekcije interferencije performansi.

4.2.9. EKSPERIMENTALNA VALIDACIJA

Da bi sistem bio upotrebljiv u praksi, bilo je neophodno da zadovolji tri testna scenarija. Prvi test predstavlja najjednostavniji slučaj u kojem je topologija statična i prati se samo jedan kontekst, odnosno jedna virtuelna mašina koja se za cijelo trajanje eksperimenta izvršava na istom fizičkom serveru. Drugi test scenario i dalje koristi statičku topologiju, ali sada posmatrani kontekst ima kompleksnu strukturu. U ovom slučaju test predstavlja skup povezanih virtuelnih mašina ili servisa koji se izvršavaju na većem broju fizičkih uređaja. Treći slučaj ima dinamičku topologiju i predstavlja ekvivalent virtuelne mašine koja u toku trajanja eksperimenta migrira između većeg broja fizičkih servera.

Testiranje je izvršeno ETFBL-PX01 klasteru koji se nalazi na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci. U okviru eksperimenta su korišćena četiri fizička servera označena sa "Host 1" do "Host 4", dok su praćeni konteksti imenovani "Context 1" do "Context 3" u zavisnosti od toga koji test scenario predstavljaju.



Slika 4.8. Rezultati eksperimentalne validacije sistema

Rezultati mjerenja predstavljeni na slici 4.8. su prikupljeni u toku devetočasovnog posmatranja sistema, od 13:00 do 22:00. Nadzor je vršen instancom Cacti sistema za nadzor dok su grafici generisani upotrebom RRDtool alata nakon čega su spojeni u jednu sliku za potrebe jednostavnije prezentacije rezultata. Horizontalna osa predstavlja vrijeme dok vertikalna osa predstavlja broj "tikova" provedenih u izvršavanju određene vrste opterećenja procesora usrednjen na nivou od pet minuta. UCD-SNMP-MIB definiše "tik" kao mjeru vremena koje je procesor proveo izvršavajući određenu vrstu koda i tipično postoji 100 "tikova" po sekundi po jezgru procesora. Generisani čvorovi koji predstavljaju kontekste su normalizovani na jedno procesorsko jezgro bez obzira na broj jezgara koji je dodijeljen virtuelnoj mašini ili stvaran broj jezgara na fizičkom serveru. Kako je broj promjenjivih koje su definisane MIB-om i koje se prate na nivou internog modela sistema prevelik za efikasno predstavljanje u grafičkom obliku, na slici su prikazana tri ilustrativne promjenjive. One predstavljaju broj "tikova" provedenih u izvršavanju korisničkih zadataka (*User* - debela plava linija), obrađivanju sistemskih poslova (*System* - svijetlo zelena oblast), te stanju čekanja (*Wait* - tanka crvena linija).

Nadzirani fizički serveri su dio klastera koji dominantno obrađuje periodične poslove što u bitnoj mjeri olakšava vizuelnu analizu prezentovanih mjerenja. Mjerni podaci su prikupljani svakih pet minuta od strane Cacti instance, a vremena prikupljanja podataka nisu sinhronizovana između Cacti sistema i sistema za nadzor konteksta sa ciljem što realnijeg testiranja funkcionisanja sistema. Ova razlika u vremenima odmjeravanja može proizvesti manja neslaganja u predstavljanju vrijednosti sa brzim promjenama.

Prvi test (context 1) predstavlja praćenje virtuelne mašina koju izvršava fizički server broj 4 za cijelo trajanje eksperimenta. Kao što je vidljivo sa slike, mjerni podaci za posmatrani kontekst blisko prate mjerenja za fizički server sa kojim je povezan. Kako posmatrani server ima 16 fizičkih jezgara, vrijednosti za posmatrani kontekst su skalirane odgovarajućim faktorom u procesu normalizacije.

Drugi test (context 2) prati kompleksan servis sastavljen od tri instance (virtuelne mašine) koje se izvršavaju na serverima 1, 2 i 3, po jedna na svakom od servera. Kao što je vidljivo sa slike, mjerni podaci za ovaj slučaj predstavljaju najgori slučaj za svaki od posmatranih fizičkih servera. Ovo je naročito vidljivo u trenucima povećanog vremena provedenog u stanju čekanja na serveru

2, kao i u izraženim skokovima vremena provedenog u obrađivanju sistemskih poziva prisutnim na serverima 2 i 3 u približno 13:15, 14:10, 16:30, 18:20, 20:50 i 21:45.

Treći slučaj (context 3) predstavlja virtuelnu mašinu koja migrira između fizičkih servera u klasteru. Kako je virtuelni disk smješten na mrežnom serveru, sam proces migracije traje svega nekoliko sekundi i ne utiče bitno na opterećenje sistema. Na početku eksperimenta, virtuelna mašina se nalazi na serveru 1. Prva migracija se dešava u približno 18:45 na server 2, zatim na server 3 u približno 19:45, te konačno na server 4 u približno u 20:45. Vrijednosti predstavljene na slici ilustruju ove migracije jer blisko prate vrijednosti fizičkih servera u zavisnosti od toga koji je server u datom trenutku izvršavao posmatranu virtuelnu mašinu. Vrijedi istaći da serveri 1 i 2 imaju 24 procesorska jezgra dok serveri 3 i 4 imaju po 16 jezgara. Podaci generisani za posmatrani kontekst sakrivaju ovu kompleksnost od korisnika jer su sve vrijednosti korektno normalizovane. Na primjer, sistem predstavlja 600 "tikova" u sekundi za server 2 (17:55) i 400 "tikova" u sekundi za server 4 (20:30) kao istu normalizovanu vrijednost od 25 "tikova" u sekundi jer su oba servera opterećena istim relativnim opterećenjem koje zauzima četvrtinu raspoloživih resursa. Kao što je ranije navedeno, ovaj pristup sprečava sakrivanje problema izazvanih od strane čvorova sa skromnijim resursima, a koji mogu unijeti značajne probleme na nivou kompletnog sistema.

Nakon analize predstavljenih podataka može se zaključiti da predstavljeni sistem za nadzor funkcioniše na korektan način i da je u stanju da vrši nadzor i statičkih i dinamičkih topologija, te jednostavnih i kompleksnih korisničkih sistema. Sistem je takođe uspješno testiran i u okviru VI-SEEM i NI4OS-Europe projekata i regionalnih distribuiranih dijeljenih infrastruktura.

5. PRIMJENA U IOT OKRUŽENJU

Kao što je ranije navedeno, savremene infrastrukture često kombinuju više različitih okruženja, od izuzetno prostih senzorskih uređaja za jednostavna mjerenja do moćnih računarskih klastera za prikupljanje, obradu i analizu velike količine podataka. U ovom poglavlju je dat prikaz sistema koji omogućava nadzor nad IoT uređajima upotrebom SNMP protokola i integraciju u sisteme za upravljanje umreženim uređajima.

5.1. UPOTREBA SNMP PROTOKOLA U IOT OKRUŽENJU

Uređaje koje je potrebno nadzirati možemo podijeliti u tri kategorije prema njihovom nivou podrške za SNMP. Prvu kategoriju čine uređaji koji podržavaju SNMP i u stanju su da direktno obezbijede potrebne podatke. Drugu kategoriju čine uređaji koji podržavaju SNMP ali nisu u stanju da dostave podatke u željenom obliku, na primjer, nemaju dovoljne procesorske resurse da podrže napredne sigurnosne mehanizme aktuelne verzije SNMP-a, dok treću kategoriju čine uređaji koji ne podržavaju SNMP. Sa stanovišta ovog rada, druga i treća kategorija se može posmatrati na isti način jer sistem za nadzor nije u stanju da direktno pristupi potrebnim podacima.

Prvu kategoriju pretežno čine umreženi uređaji koji i inače podržavaju udaljeni nadzor i upravljanje upotrebom SNMP-a. Imajući navedeno u vidu, integracija takvih uređaja u sistem za nadzor je jednostavna, osim u slučajevima kada uređaji ne podržavaju aktuelnu verziju standarda (npr. podržavaju zastarjele verzije 1 ili 2c) ili postoje problemi sa pristupom samim uređajima (npr. upotreba NAT-a, VPN-a, itd). Čest način rješavanja ovog problema je upotreba SNMP *proxy* servisa ili drugih srodnih tehnologija. Fizička infrastruktura koja izvršava virtuelizovane servise takođe pripada ovoj kategoriji, uz uslov da imamo slobodan pristup mjernim podacima.

Postoje četiri osnovna načina da prikupimo podatke sa uređaja iz druge i treće kategorije:

1. Uređaji koji podržavaju sisteme za razmjenu poruka (eng. *messaging*) nam omogućavaju da pratimo odgovarajuća odredišta za poruke i da ih primamo čim su generisane na samom uređaju. Ovaj pristup predstavlja najbolju opciju jer su svi podaci

ažurni, dok su potrebni resursi minimalni, ali je primjena ograničena na uređaje koji podržavaju ovakav način komunikacije.

2. Periodično prikupljanje podataka prozivanjem uređaja (polling) je jednostavno, robusno i omogućava nam da unaprijed procijenimo potrebne resurse. Mane ovog pristupa su potencijalno ispitivanje uređaja koje trenutno nije potrebno ispitivati, kao i problem ostvarivanja pravilnog balansa između povećanog opterećenja izazvanog češćim prikupljanjem i povećane neažurnosti prikupljenih podataka izazvanog rjeđim prikupljanjem istih.
3. Prosljeđivanje upita uređajima samo po potrebi (eng. *proxying*) omogućava ispitivanje uređaja samo kada postoji potreba za tim što smanjuje potrebne resurse, ali uvodi nepoznato vrijeme odgovora jer je potrebno ispitati sve zahtijevane uređaje i sačekati na njihove odgovore, bez garancije da će odgovori zaista i stići u prihvatljivom roku. Na ovaj način je moguće da mali broj sporih ili preopterećenih uređaja unosi velika kašnjenja u sistem što dovodi do komplikacija pri računanju agregatnih vrijednosti na nivou okruženja.
4. Prosljeđivanje upita uređajima samo po potrebi uz keširanje (eng. *caching proxy*) proširuje prethodno opisani pristup keširanjem prethodnih rezultata što poboljšava performanse po cijenu povećanja neažurnosti podataka.

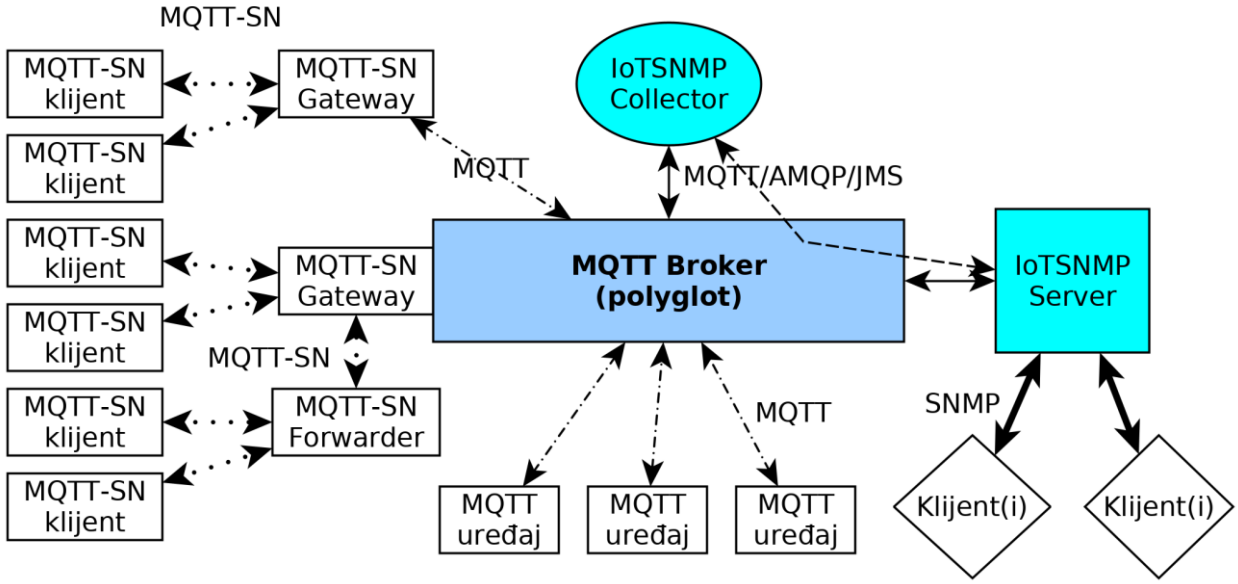
Opisani pristupi mogu biti kombinovani na veliki broj načina u zavisnosti od konkretnih ograničenja sa kojima se susrećemo u stvarnim IoT okruženjima. Lindholm-Ventola i Silverajan su pokazali da je moguć efikasan nadzor nad IoT uređajima upotrebom CoAP-SNMP proxy servisa, sa opcionom komponentom baze podataka, što u osnovi odgovara trećem opisanom pristupu [118]. U radu je posebno istaknuta potreba za buduća ispitivanja vezana za problem obavještavanja iniciranog od strane IoT uređaja u sistemima za nadzor.

Od četiri navedena pristupa samo prvi zadovoljava na smislen način sve zahtjeve koji se postavljaju pred sistem za nadzor sa stanovišta obavještavanja. Preostala tri pristupa će ili uvesti određeno kašnjenje u sistem ili će u potpunosti propustiti da detektuju obavještenje ukoliko za dati uređaj nije bilo zahtjeva u određenom vremenskom periodu. Ako sam uređaj podržava slanje poruka ili SNMP obavještenja, sistem za nadzor je u stanju da iste obradi sa minimalnim kašnjenjem.

5.2. MQTT-SNMP MOST

U cilju nadzora MQTT i MQTT-SN uređaja upotrebom SNMP-a, sistem treba da implementira komponentu koja će slušati dolazne poruke sa nadziranih uređaja i transformisati ih u oblih pogodan za SNMP klijente. Iako je moguće opsluživati individualne SNMP klijente, mnogo češći je slučaj u kom su klijenti zapravo sistemi za nadzor (NMS).

Arhitektura jednog takvog novopredloženog sistema je data na slici 5.1. Sistem se sastoji od: nadziranih uređaja koji podržavaju MQTT ili MQTT-SN, MQTT-SN Gateway i Forwarder komponenti, MQTT brokera, IoTSNMP Collector komponente za prikupljanje podatka, IoTSNMP servera i proizvoljnog broja SNMP klijenata.



Slika 5.1. Prikaz sistema sa podrškom za MQTT i SNMP

MQTT-SN Gateway i Forwarder komponente su uključene sa ciljem omogućavanja nadzora nad MQTT-SN uređajima i mogu biti integrisane u sam MQTT broker, što je i prikazano na slici za jednu instancu. Izborom poliglot tipa brokera omogućena je jednostavna integracija sa drugim komponentama sistema jer sam broker vrši translaciju poruka između različitih formata. Iako su na slici predstavljene kao odvojene komponente, sami kolektor i server mogu biti realizovani i kao monolitni servis što predstavlja jednostavnije rješenje po cijenu smanjivanja fleksibilnosti i skalabilnosti sistema. Sam broker može biti realizovan u konfiguraciji koja je

otporna na pad pojedinačne komponente upotrebom mreže brokera (eng. *network of brokers*) koja je podržana od strane Apache ActiveMQ brokera [119].

5.3. INTEGRACIJA U POSTOJEĆE SISTEME ZA NADZOR

Iako je moguće koristiti namjenske sisteme za nadzor pojedinačnih IoT infrastruktura, većina organizacija već koristi neki od standardnih sistem za nadzor i upravljanjem mrežama. Kao što je ranije navedeno, postoji veliki broj sistema za nadzor koji imaju različite arhitekture, izvršavaju se u okviru različitih platformi, te podržavaju brojne operativne procedure i načine prikupljanja podataka. Među najpopularnijima su Nagios [7], Zenoss [50], Zabbix [49] i OpenNMS [69]. Jedan primjer konkretne upotrebe Zenoss sistema za nadzor IoT okruženja je dat u [120]. Mazhelis i saradnici su analizirali mogućnost upotrebe CoAP protokola za nadzor IoT okruženja, te prilagođavanje postojećeg AMAAIS (eng. *Accounting and Monitoring of Authentication and Authorization Infrastructure Services*) projekta za takvu namjenu [121].

Sami NMS-ovi imaju međusobno velike razlike, međutim, praktično svi sistemi za nadzor imaju podršku za prikupljanje podataka upotrebom SNMP protokola što im omogućava da koriste prethodno opisani sistem za pristup IoT dijelu posmatrane infrastrukture. U zavisnosti od konkretnih zahtjeva, moguće je sistemima za nadzor dostaviti direktne podatke sa posmatranih IoT uređaja ili izvršiti određene transformacije na njima poput usrednjavanja vrijednosti za skup uređaja ili vršenje drugih agregatnih izračunavanja. Na ovaj način je moguće riješiti i potencijalne probleme vezane za osjetljive mjerne podatke jer se agregacijom podataka takođe ostvaruje i anonimizacija i više nije moguće jednoznačno povezati dobijene informacije sa određenim uređajem kao izvorom podataka.

Kako je u upotrebi veliki broj bitno različitih IoT uređaja u vrlo dinamičnim okruženjima, često je izuzetno teško ili praktično neizvodivo kreirati infrastrukturu koja bi omogućila zadovoljavajuće ispitivanje. Kao alternativa pravljenju kompleksne stvarne test infrastrukture moguća je upotreba Contiki [122] baziranog Cooja mrežnog simulatora [123] koji podržava širok spektar namjena. Za CoAP i MQTT u Java okruženju je moguće koristiti Californium CoAP okruženje [124] i Fusesource MQTT biblioteke [89].

Važno je istaći kompatibilnost sa sistemima opisanim u poglavljima 3.4. i 4.2.4. što omogućava lakšu integraciju u kompleksne infrastrukture. Kompletna IoT infrastruktura se tipično sastoji od tri osnovne komponente: IoT uređaja, servisnih komponenti koje prikupljaju, obrađuju i skladište podatke sa njih, te mrežne infrastrukture koja obezbjeđuje međusobnu povezanost individualnih komponenti. Da bi nadzor kompletnog sistema bio moguć upotrebom SNMP-a, potrebno je obezbijediti podršku za svaku od komponenti. Uređaji koji čine mrežnu infrastrukturu se i inače nadziru upotrebom SNMP-a, dok se servisne komponente tipično izvršavaju u dijeljenom okruženju, te je u oba ova slučaja moguće primijeniti rješenja opisana u poglavlju 4.2.4. Ukoliko je potrebno obezbijediti i nadzor pravilnog funkcionisanja samih servisa i njihovih instanci, moguće je iskoristiti sistem opisan u 3.4 pri čemu IoTSNMP Collector komponenta sa slike 5.1. postaje kolektor koji šalje podatke sistemu za skladištenje podataka, dok agentska komponenta sa slike 3.6. preuzima funkciju IoTSNMP Server komponente.

6. ZAKLJUČAK

U ovoj disertaciji su dati prijedlozi novih rješenja za nadzor distribuirane i heterogene računarske infrastrukture kao servisa koja na uniforman način, upotrebom SNMP protokola, omogućavaju korisniku da na efikasan način vrši nadzor i fizičkih i virtuelizovanih resursa i na operativnom i na funkcionalnom nivou.

U radu su identifikovane tri vrste infrastruktura od interesa za ovo istraživanje, dat je njihov opis, kao i pregled dostupnih sistema za nadzor identifikovanih vrsta infrastruktura. Posebna pažnja je posvećena prikazu platformi i sistema za nadzor u oblasti računarskih infrastruktura kao servisa za tri najpopularnije platforme otvorenog koda za privatne IaaS instalacije. Izvršena je i analiza dostupnih sistema za grid, računarstvo visokih performansi i računarske mrežne infrastrukture. Dat je i opis SNMP protokola i njegove upotrebe u sistemima za nadzor uz naglasak na proširenje funkcionalnosti agentske komponente sistema.

Nakon prikupljanja i analize, formulisani su korisnički zahtjevi, ali i ograničenja definisana od strane operatera infrastrukture. Tri najvažnija identifikovana korisnička zahtjeva su se odnosila na pristupačnost sistemu, te jednostavnost i reprezentativnost podataka, dok su operaterska ograničenja bila prvenstveno usmjerena na sigurnost sistema, nenametljivost u radu bez negativnih posljedica po nadzirane sisteme, te upotrebljivost realizovanog rješenja kroz integraciju u postojeće sisteme za nadzor i upravljanje.

Na osnovu dostupne infrastrukture, ali i neračunarskih resursa, u okviru VI-SEEM projekta projektovan je i realizovan sistem za nadzor distribuirane i heterogene računarske infrastrukture koja se prostirala na 16 država i tri kontinenta. Nadzirana infrastruktura je uključivala raznovrsne resurse, od mrežnih veza, preko virtuelizovanih servera, do grid klastera i superračunara do IaaS instalacija. U okviru sistema je realizovan nadzor na operativnom i funkcionalnom nivou koji je pokrivao i nadzor servisa koji koriste opisanu infrastrukturu kao osnovu. Sam sistem za nadzor je verifikovan korišćenjem za potrebe projekta u periodu od 12 mjeseci bez zabilježenih problema u radu. U cilju demonstracije fleksibilnosti realizovanog sistema, implementiran je i alternativni način pristupa mjernim podacima u vidu veb portala. Analizirane su i performanse sistema, a na osnovu dobijenih rezultata je opravdan izbor odabranih tehnologija za realizaciju.

Važan doprinos predstavlja i sistematizacija i organizacija podataka u vidu MIB dokumenta pod nazivom ETFBL-DCI-MIB. Upotrebom realizovanog sistema je moguć nadzor javnih, privatnih, ali i hibridnih infrastruktura što je bio i konkretan slučaj u VI-SEEM projektu jer je Elektrotehnički fakultet Univerziteta u Banjoj Luci bio i operater i korisnik infrastrukturnih usluga u okviru projekta. Realizovani sistem za nadzor je testiran u okviru regionalne infrastrukture i korišćen je kao operativni alat za nadzor svih resursa i servisa projekta.

Posebna pažnja u radu je posvećena i problemu degradacije performansi u dijeljenim infrastrukturama izazvanom aktivnostima van kontrole korisnika sistema što je identifikovano kao problem interferencije performansi. Ovaj problem je detaljno analiziran za tri posmatrane vrste infrastruktura, izvršen je pregled literature i analizirani su zahtjevi i ograničenja koje sistem za nadzor treba da zadovolji. Dat je prikaz predloženog rješenja uz primjere i analizu za svaku posmatranu vrstu dijeljene infrastrukture, detaljan opis i analiza arhitekture sistema, te problema prikupljanja i prezentacije podataka. Opisan je interni model sistema kao i algoritam praćenja konteksta u dijeljenim infrastrukturama. Posebna pažnja je posvećena sigurnosnim aspektima upotrebe sistema, a izvršeno je i poređenje sa drugim sistemima opisanim u dostupnoj literaturi. Nakon realizacije predloženog sistema, izvršena je eksperimentalna validacija čime je demonstrirana upotrebljivost sistema za nadzor realnih dijeljenih infrastruktura.

Imajući u vidu sve veću popularnost i velike resurse koji se ulažu u oblast interneta stvari, analiza problema je proširena i na tu oblast. Identifikovani su problemi i specifičnosti IoT uređaja i okruženja, analizirana je mogućnost upotrebe SNMP protokola za nadzor te je formulisano rješenje koje koristi SNMP da bi obezbijedilo uniforman način nadzora kompletne infrastrukture, od jednostavnih senzorskih mreža do naprednih računarskih resursa koji prihvataju, obrađuju i analiziraju njihove podatke.

Sva tri projektovana i realizovana sistema imaju međusobno kompatibilnu arhitekturu što omogućava jednostavniju integraciju, te povećanu fleksibilnost i prilagođavanje zahtjevima korisnika.

Osnovni zaključci istraživanja se mogu sumirati na sljedeći način:

1. Od velike je važnosti kvalitetan nadzor performansi dijeljenih i virtuelizovanih infrastruktura i resursa u cilju obezbjeđivanja zadovoljavajućeg nivoa kvaliteta usluga za korisnike infrastruktura.
2. Korisnici i operateri infrastruktura imaju često suprotstavljene zahtjeve čije zadovoljavanje iziskuje pažljiv odabir pristupa projektovanju i realizaciji sistema za nadzor.
3. Moguće je ostvariti kvalitetan nadzor distribuiranih računarskih infrastruktura upotrebom SNMP standarda kreiranjem namjenskog MIB-a za nadzor na nivou infrastrukture, te upotrebom postojećih MIB-ova za nadzor na nižim nivoima.
4. Pravilnim odabirom odgovarajuće arhitekture sistema za nadzor, moguće je realizovati sisteme za nadzor distribuiranih infrastruktura, dijeljenih infrastruktura, te IoT okruženja na uniforman način što olakšava upotrebu navedenih sistema i sa strane operatora i korisnika.

Buduća istraživanja se mogu bazirati na tri pravca. Prvi pravac bi obuhvatao širenje podrške na dodatne vrste infrastruktura koje nisu obuhvaćene ovim radom, drugi pravac bi predstavljao istraživanje na temu primjenjivosti na komercijalne sisteme, dok bi treći pravac razmotrio integraciju razvijenog rješenja sa postojećim sistemima za detekciju interferencije performansi i formiranje hibridnog pristupa.

LITERATURA

[1] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, 2003.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," p. 7.

[3] "OpenStack Docs: Basic architecture." [Online]. Available: https://docs.openstack.org/glance/pike/contributor/architecture.html. [Accessed: 07-Dec-2019].

[4] B. Wijnen, R. Presuhn, and D. Harrington, "An Architecture for Describing SNMP Management Frameworks." [Online]. Available: https://tools.ietf.org/html/rfc2571. [Accessed: 07-Dec-2019].

[5] "Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)." [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Accessed: 07-Dec-2019].

[6] G. Da Cunha Rodrigues *et al.*, "Monitoring of Cloud Computing Environments: Concepts, Solutions, Trends, and Future Directions," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, New York, NY, USA, 2016, pp. 378–383.

[7] "Nagios - The Industry Standard In IT Infrastructure Monitoring," *Nagios*. [Online]. Available: https://www.nagios.org/. [Accessed: 07-Dec-2019].

[8] "ARGO Availability and Reliability Monitoring," *ARGO Documentation*. [Online]. Available: http://argoeu.github.io/. [Accessed: 07-Dec-2019].

[9] "VRE for regional Interdisciplinary communities in Southeast Europe and the Eastern Mediterranean | VI-SEEM Project | H2020 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/198274/factsheet/en. [Accessed: 07-Dec-2019].

[10] M. Daniele, B. Wijnen, and M. Ellison, "Agent Extensibility (AgentX) Protocol Version 1." [Online]. Available: https://tools.ietf.org/html/rfc2741. [Accessed: 07-Dec-2019].

[11] G. Aceto, A. Botta, W. de Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, Jun. 2013.

[12] M. Díaz, C. Martín, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, May 2016.

[13] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, Jun. 2017.

[14] K. Alhamazani *et al.*, "An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art," *Computing*, vol. 97, no. 4, pp. 357–377, Apr. 2015.

[15] "Build the future of Open Infrastructure.," *OpenStack*. [Online]. Available: https://www.openstack.org/. [Accessed: 07-Dec-2019].

[16] "The OpenStack Foundation," *OpenStack*. [Online]. Available: https://www.openstack.org/foundation/. [Accessed: 07-Dec-2019].

[17] "OpenStack Docs: OpenStack API Documentation." [Online]. Available: https://docs.openstack.org/api-quick-start/. [Accessed: 07-Dec-2019].

[18] V. Koukis, C. Venetsanopoulos, and N. Koziris, "CLOUD Synnefo: A Complete Cloud Stack over Ganeti," vol. 38, no. 5, p. 5, 2013.

[19] "Ganeti." [Online]. Available: http://www.ganeti.org/. [Accessed: 07-Dec-2019].

[20] "Chef: Deploy new code faster and more frequently. Automate infrastructure and applications," *Chef*. [Online]. Available: https://www.chef.io/. [Accessed: 07-Dec-2019].

[21] "Homepage." [Online]. Available: https://puppet.com/. [Accessed: 07-Dec-2019].

[22] "JAAS - Juju as a Service | Juju." [Online]. Available: https://jaas.ai/. [Accessed: 07-Dec-2019].

[23] "OpenStack Deployment | Installer | Fuel | Mirantis," *Mirantis | Pure Play Open Cloud*. [Online]. Available: https://www.mirantis.com/software/openstack/fuel/. [Accessed: 07-Dec-2019].

[24] "VMware Cloud | Cloud Services | Cloud Infrastructure." [Online]. Available: https://cloud.vmware.com/. [Accessed: 07-Dec-2019].

[25] "IBM Cloud," 04-Dec-2019. [Online]. Available: https://www.ibm.com/cloud. [Accessed: 07-Dec-2019].

[26] "The Python Language Reference — Python 3.8.0 documentation." [Online]. Available: https://docs.python.org/3/reference/. [Accessed: 07-Dec-2019].

[27] "Messaging that just works — RabbitMQ." [Online]. Available: https://www.rabbitmq.com/. [Accessed: 07-Dec-2019].

[28] "MySQL." [Online]. Available: https://www.mysql.com/. [Accessed: 07-Dec-2019].

[29] "MariaDB Foundation," *MariaDB.org*. [Online]. Available: https://mariadb.org/. [Accessed: 07-Dec-2019].

[30] "Galera Cluster for MySQL | The world's most advanced open-source database cluster." [Online]. Available: https://galeracluster.com/. [Accessed: 07-Dec-2019].

[31] "OpenNebula – OpenNebula." [Online]. Available: https://opennebula.org/. [Accessed: 07-Dec-2019].

[32] "Open Cloud Reference Architecture," *OpenNebula Support*. [Online]. Available: http://support.opennebula.pro/hc/en-us/articles/204210319-Open-Cloud-Reference-Architecture. [Accessed: 07-Dec-2019].

[33] "KVM." [Online]. Available: https://www.linux-kvm.org/page/Main_Page. [Accessed: 07-Dec-2019].

[34] "Linux Containers." [Online]. Available: https://linuxcontainers.org/. [Accessed: 07-Dec-2019].

[35] "Welcome - Amazon Elastic Compute Cloud." [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/Welcome.html. [Accessed: 07-Dec-2019].

[36] "Open Cloud Computing Interface | Open Standard | Open Community." [Online]. Available: https://occi-wg.org/. [Accessed: 07-Dec-2019].

[37] "Apache Cloudstack," *Apache Cloudstack*. [Online]. Available: https://cloudstack.apache.org/. [Accessed: 07-Dec-2019].

[38] "Citrix Hypervisor | Open Source Server Virtualization." [Online]. Available: https://xenserver.org/. [Accessed: 07-Dec-2019].

[39] A. Velte and T. Velte, *Microsoft Virtualization with Hyper-V*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2010.

[40] "Eucalyptus." [Online]. Available: https://www.eucalyptus.cloud/. [Accessed: 07-Dec-2019].

[41] "Managed Cloud Services by Rackspace, the #1 Managed Cloud Provider," *Rackspace*. [Online]. Available: https://www.rackspace.com/cloud. [Accessed: 07-Dec-2019].

[42] "Mirantis: Cloud On Your Terms," *Mirantis | Pure Play Open Cloud*. [Online]. Available: https://www.mirantis.com. [Accessed: 07-Dec-2019].

[43] "oVirt," *oVirt*. [Online]. Available: https://www.ovirt.org/. [Accessed: 07-Dec-2019].

[44] "Home." [Online]. Available: https://openqrm-enterprise.com/. [Accessed: 07-Dec-2019].

[45] "Open-source virtualization management platform Proxmox VE." [Online]. Available: https://www.proxmox.com/en/proxmox-ve. [Accessed: 07-Dec-2019].

[46] "RightScale 2019 State of the Cloud Report from Flexera." [Online]. Available: https://info.flexera.com/SLO-CM-WP-State-of-the-Cloud-2019. [Accessed: 07-Dec-2019].

[47] "Telemetry - OpenStack." [Online]. Available: https://wiki.openstack.org/wiki/Telemetry. [Accessed: 07-Dec-2019].

[48] "Start page – collectd – The system statistics collection daemon." [Online]. Available: https://collectd.org/. [Accessed: 07-Dec-2019].

[49] "Zabbix:: The Enterprise-Class Open Source Network Monitoring Solution." [Online]. Available: https://www.zabbix.com/. [Accessed: 07-Dec-2019].

[50] "Intelligent Application and Service Monitoring + AIOps," *Zenoss*. [Online]. Available: https://www.zenoss.com/. [Accessed: 07-Dec-2019].

[51] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, Oct. 2014.

[52] "Application Monitoring Across All Environments with vRealize Hyperic," *VMware*. [Online]. Available: https://www.vmware.com/products/vrealize-hyperic.html. [Accessed: 07-Dec-2019].

[53] "DX Infrastructure Manager." [Online]. Available: https://www.broadcom.com/info/aiops/dx-infrastructure-manager. [Accessed: 07-Dec-2019].

[54] A. Vogel, D. Griebler, C. A. F. Maron, C. Schepke, and L. G. Fernandes, "Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016, pp. 672–679.

[55] "OpenStack Docs: Welcome to Monasca's Documentation!" [Online]. Available: https://docs.openstack.org/monasca-api/latest/. [Accessed: 07-Dec-2019].

[56] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, Aug. 2001.

[57] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *J Comput Sci Technol*, vol. 21, no. 4, p. 513, Jul. 2006.

[58] O. Appleton *et al.*, "The next-generation ARC middleware," *Ann. Telecommun.*, vol. 65, no. 11, pp. 771–776, Dec. 2010.

[59] D. W. Erwin, "UNICORE—a Grid computing environment," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13–15, pp. 1395–1410, 2002.

[60] C. Aiftimiei *et al.*, "Towards next generations of software for distributed infrastructures: The European Middleware Initiative," in *2012 IEEE 8th International Conference on E-Science*, 2012, pp. 1–10.

[61] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, Jul. 2004.

[62] M. Eisler <mike.ietf.xdr@eisler.com>, "XDR: External Data Representation Standard." [Online]. Available: https://tools.ietf.org/html/rfc4506. [Accessed: 07-Dec-2019].

[63] "RRDtool - The Time Series Database," *RRDtool*. [Online]. Available: http://www.rrdtool.org/. [Accessed: 07-Dec-2019].

[64] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[65] "CIDR Report." [Online]. Available: https://www.cidr-report.org/as2.0/. [Accessed: 07-Dec-2019].

[66] "Home," *Icinga*. [Online]. Available: https://icinga.com/. [Accessed: 07-Dec-2019].

[67] "Cacti® - The Complete RRDTool-based Graphing Solution." [Online]. Available: https://www.cacti.net/. [Accessed: 07-Dec-2019].

[68] "Grafana: The open observability platform," *Grafana Labs*. [Online]. Available: https://grafana.com/. [Accessed: 07-Dec-2019].

[69] "Home," *The OpenNMS Group, Inc.* [Online]. Available: https://www.opennms.com/. [Accessed: 07-Dec-2019].

[70] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[71] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)." [Online]. Available: https://tools.ietf.org/html/rfc7252. [Accessed: 07-Dec-2019].

[72] "MQTT Version 3.1.1." [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html. [Accessed: 07-Dec-2019].

[73] Z. Shelby <zach@sensinode.com>, "Constrained RESTful Environments (CoRE) Link Format." [Online]. Available: https://tools.ietf.org/html/rfc6690. [Accessed: 07-Dec-2019].

[74] "Zigbee Alliance," *Zigbee Alliance*. [Online]. Available: https://zigbee.org/. [Accessed: 07-Dec-2019].

[75] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN) Protocol Specification," p. 28, 1999.

[76] "RFC 3418 - Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)." [Online]. Available: https://tools.ietf.org/html/rfc3418. [Accessed: 07-Dec-2019].

[77] "Specification Languages - ASN.1." [Online]. Available: https://portal.etsi.org/Services/Centre-for-Testing-Interoperability/ETSI-Approach/Specification-Languages/ASN1. [Accessed: 07-Dec-2019].

[78] K. McCloghrie, K. McCloghrie, J. Schoenwaelder, and D. Perkins, "Structure of Management Information Version 2 (SMIv2)." [Online]. Available: https://tools.ietf.org/html/rfc2578. [Accessed: 07-Dec-2019].

[79] L. Heintz, M. Ellison, and S. Gudur, "Definitions of Managed Objects for Extensible SNMP Agents." [Online]. Available: https://tools.ietf.org/html/rfc2742. [Accessed: 07-Dec-2019].

[80] "South-Eastern European research and education network | SEEREN2 Project | FP6 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/80137/factsheet/en. [Accessed: 07-Dec-2019].

[81] "SEE-GRID eInfrastructure for regional eScience | SEE-GRID-SCI Project | FP7 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/86416/factsheet/en. [Accessed: 07-Dec-2019].

[82] "High-Performance Computing Infrastructure for South East Europe's Research Communities | HP-SEE Project | FP7 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/95208/factsheet/en. [Accessed: 07-Dec-2019].

[83] "European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe | EGI-InSPIRE Project | FP7 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/95923/factsheet/it. [Accessed: 07-Dec-2019].

[84] G. Mathieu, D. A. Richards, D. J. Gordon, C. D. C. Novales, P. Colclough, and M. Viljoen, "GOCDB, a topology repository for a worldwide grid infrastructure," *J. Phys.: Conf. Ser.*, vol. 219, no. 6, p. 062021, Apr. 2010.

[85] S. Andreozzi, S. Burke, L. Field, and B. Kónya, "Towards GLUE 2: evolution of the computing element information model," *J. Phys.: Conf. Ser.*, vol. 119, no. 6, p. 062009, Jul. 2008.

[86] J. Montagnat *et al.*, "A Secure Grid Medical Data Manager Interfaced to the gLite Middleware," *J Grid Computing*, vol. 6, no. 1, pp. 45–59, Mar. 2008.

[87] K. J. Modi, D. P. Chowdhury, and S. Garg, "An Ontology-Based Approach for Automatic Cloud Service Monitoring and Management," in *Advances in Big Data and Cloud Computing*, Singapore, 2018, pp. 1–16.

[88] A. Stephen, S. Benedict, and R. P. A. Kumar, "Monitoring IaaS using various cloud monitors," *Cluster Comput*, vol. 22, no. 5, pp. 12459–12471, Sep. 2019.

[89] Fusesource, "fusesource/mqtt-client," *GitHub*, 26-Jul-2019. [Online]. Available: https://github.com/fusesource/mqtt-client. [Accessed: 07-Dec-2019].

[90] "Jasmin: JAX - Java AgentX Client Toolkit." [Online]. Available: https://www.ibr.cs.tu-bs.de/projects/jasmin/jax.html. [Accessed: 07-Dec-2019].

[91] "React – A JavaScript library for building user interfaces." [Online]. Available: https://reactjs.org/. [Accessed: 07-Dec-2019].

[92] H. Nemati and M. R. Dagenais, "Virtual CPU State Detection and Execution Flow Analysis by Host Tracing," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, 2016, pp. 7–14.

[93] X. Pu *et al.*, "Who Is Your Neighbor: Net I/O Performance Interference in Virtualized Clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 314–329, Jul. 2013.

[94] X. Chen, L. Rupprecht, R. Osman, P. Pietzuch, F. Franciosi, and W. Knottenbelt, "CloudScope: Diagnosing and Managing Performance Interference in Multi-tenant Clouds," in *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2015, pp. 164–173.

[95] S. Amri, H. Hamdi, and Z. Brahmi, "Inter-VM Interference in Cloud Environments: A Survey," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017, pp. 154–159.

[96] T. Lee, M. Lee, and Y. I. Eom, "VM-aware Flush Mechanism for Mitigating Inter-VM I/O Interference," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1501–1506.

[97] R. R. Expósito, G. L. Taboada, S. Ramos, J. Touriño, and R. Doallo, "Performance analysis of HPC applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218–229, Jan. 2013.

[98] C. Ruiz, E. Jeanvoine, and L. Nussbaum, "Performance Evaluation of Containers for HPC," in *Euro-Par 2015: Parallel Processing Workshops*, Cham, 2015, pp. 813–824.

[99] M. G. Xavier, I. C. D. Oliveira, F. D. Rossi, R. D. D. Passos, K. J. Matteussi, and C. A. F. D. Rose, "A Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Clouds," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 253–260.

[100] J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair Network Bandwidth Allocation in IaaS Datacenters via a Cooperative Game Approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 873–886, Apr. 2016.

[101] Y. Gong, B. He, and D. Li, "Finding Constant from Change: Revisiting Network Performance Aware Optimizations on IaaS Clouds," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Piscataway, NJ, USA, 2014, pp. 982–993.

[102] M. Savić, N. Obradović, A. Keleč, and M. Ljubojević, "Monitoring of Distributed Computing Infrastructure in the VI-SEEM Project," in *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2019, pp. 1–6.

[103] M. Savić, "BRIDGING THE SNMP GAP: SIMPLE NETWORK MONITORING THE INTERNET OF THINGS," *Facta Universitatis, Series: Electronics and Energetics*, vol. 29, no. 3, pp. 475–487, Dec. 2015.

[104] "National Initiatives for Open Science in Europe | NI4OS-Europe Project | H2020 | CORDIS | European Commission." [Online]. Available: https://cordis.europa.eu/project/rcn/224431/factsheet/en. [Accessed: 07-Dec-2019].

[105] M. T. Rose and K. McCloghrie, "Structure and identification of management information for TCP/IP-based internets." [Online]. Available: https://tools.ietf.org/html/rfc1155. [Accessed: 07-Dec-2019].

[106] K. McCloghrie and F. Kastenholz, "The Interfaces Group MIB." [Online]. Available: https://tools.ietf.org/html/rfc2863. [Accessed: 07-Dec-2019].

[107] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," in *Cloud Computing*, Berlin, Heidelberg, 2009, pp. 254–265.

[108] Yang Xiao, "Physical path tracing for IP traffic using SNMP-a comprehensive solution for end-to-end and multicast streams monitoring," in *2010 2nd International Conference on Education Technology and Computer*, 2010, vol. 5, pp. V5-310-V5-317.

[109] L. Jun, Z. Xuefeng, S. Weihong, and Z. Qilin, "Network Topology Discovery Based on SNMP," in *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 194–199.

[110] S. Pandey, M.-J. Choi, Y. J. Won, and J. W.-K. Hong, "SNMP-based enterprise IP network topology discovery," *International Journal of Network Management*, vol. 21, no. 3, pp. 169–184, 2011.

[111] J. Schoenwaelder <j.schoenwaelder@jacobs-university.de>, "Simple Network Management Protocol (SNMP) Context EngineID Discovery." [Online]. Available: https://tools.ietf.org/html/rfc5343. [Accessed: 07-Dec-2019].

[112] F. Xu, F. Liu, and H. Jin, "Heterogeneity and Interference-Aware Virtual Machine Provisioning for Predictable Performance in the Cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2470–2483, Aug. 2016.

[113] Y. Amannejad, D. Krishnamurthy, and B. Far, "Detecting performance interference in cloud-based web services," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 423–431.

[114] S. A. Javadi, S. Mehra, B. K. R. Vangoor, and A. Gandhi, "UIE: User-Centric Interference Estimation for Cloud Applications," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, 2016, pp. 119–122.

[115] S. Javadi and A. Gandhi, "User-Centric Interference-Aware Load Balancing for Cloud-Deployed Applications," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[116] S. Votke, S. A. Javadi, and A. Gandhi, "Modeling and Analysis of Performance Under Interference in the Cloud," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2017, pp. 232–243.

[117] K. Joshi, A. Raj, and D. Janakiram, "Sherlock: Lightweight Detection of Performance Interference in Containerized Cloud Services," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2017, pp. 522–530.

[118] H. Lindholm-Ventola and B. Silverajan, "CoAP-SNMP Interworking in IoT Scenarios," *Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti-Tampere University of Technology. Department of Pervasive Computing. Report; 3*, 2014.

[119] "ActiveMQ." [Online]. Available: https://activemq.apache.org/. [Accessed: 07-Dec-2019].

[120] U. Gupta, "Monitoring in IOT enabled devices," *arXiv:1507.03780 [cs]*, Jul. 2015.

[121] O. Mazhelis, M. Waldburger, G. S. Machado, B. Stiller, and P. Tyrvainen, "Extending Monitoring and Accounting Infrastructure Towards Constrained Devices in Internet-of-Things Applications," p. 22.

[122] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 455–462.

[123] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," presented at the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), 2006.

[124] E. C. project, "Eclipse Californium," *Eclipse Californium* [Online]. Available: https://www.eclipse.org/californium/. [Accessed: 07-Dec-2019].

PRILOZI

PRILOG A - ETFBL-DCI-MIB

```
ETFBL-DCI-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, OBJECT-IDENTITY,
    Unsigned32              FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,
    TruthValue, DisplayString     FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP          FROM SNMPv2-CONF
    ;

etfbl MODULE-IDENTITY
    LAST-UPDATED "201708150817Z"
    ORGANIZATION "ETF BL"
    CONTACT-INFO
            "   Mihajlo Savic
                Faculty of Electrical Engineering Banja Luka
                Patre 5
                78000 Banja Luka
                Bosnia and Herzegovina
          "
    DESCRIPTION
            "Production revision"
    REVISION "201708150817Z"
    DESCRIPTION
            "Production revision"

    ::= { iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) 33943 }

LookupIndex ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "Lookup Index value"
    SYNTAX Unsigned32

MBytes ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "Memory size, expressed in megabytes."
    SYNTAX Unsigned32

GBytes ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "Storage size, expressed in gigabytes."
    SYNTAX Unsigned32

DecimalValueMillis ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d-3"
    STATUS current
    DESCRIPTION
        "Decimal number with three decimal places."
```

```
    SYNTAX Unsigned32

ServiceStatusDef    ::= TEXTUAL-CONVENTION
       STATUS              current
       DESCRIPTION
              "Status of a monitored DCI service.
              (0): NA
              (10): UP
              (20): DOWN
              (30): DEGRADED
              (40): PARTIAL
              (100): MAINT"
       SYNTAX INTEGER {
              na(0),
              up(10),
                 down(20),
                 degraded(30),
                 partial(40),
              maint(100)
              }

ServiceEndpointStatusDef   ::= TEXTUAL-CONVENTION
       STATUS              current
       DESCRIPTION
              "Status of a monitored DCI service endpoint.
                     (0): NA
                     (10): OK
                     (20): INFO
                     (30): NOTE
                     (40): WARN
                     (50): ERROR
                     (60): CRIT
                     (100): MAINT"
       SYNTAX INTEGER {
                            na(0),
                            ok(10),
                            info(20),
                            note(30),
                            warn(40),
                            error(50),
                            crit(60),
                            maint(100)
                     }



dci OBJECT-IDENTITY
    STATUS current
     DESCRIPTION "The subtree delegated to Computing Centre for DCI related projects"
     ::= { etfbl 28 }

dciObjects          OBJECT IDENTIFIER ::= { dci 1 }
dciEvents           OBJECT IDENTIFIER ::= { dci 2 }
dciCompliances      OBJECT IDENTIFIER ::= { dci 3 }

generalCompliance   MODULE-COMPLIANCE
       STATUS              current
       DESCRIPTION  "Compliance statement for a SNMP entity which implements this MIB"
       MODULE
       MANDATORY-GROUPS    { systemInformationGroup, siteInformationGroup,
voInformationGroup}
       GROUP ceInformationGroup
```

```
    DESCRIPTION "Mandatory only if implementing Computing Element resource monitoring"
      GROUP saInformationGroup
    DESCRIPTION "Mandatory only if implementing Storage Element resource monitoring"
      GROUP monitoringInformationGroup
    DESCRIPTION "Mandatory only if implementing functional monitoring system and
reporting detailed information"
      GROUP hypervisorInformationGroup
    DESCRIPTION "Mandatory only if implementing detailed resource monitoring of VM
based infrastructure"
      ::= { dciCompliances 1 }

systemInformationGroup OBJECT-GROUP
      OBJECTS { systemName, systemDescription, systemContact }
      STATUS current
      DESCRIPTION "Object group for hosting system information"
      ::= { dciCompliances 2 }

siteInformationGroup OBJECT-GROUP
      OBJECTS {
      siteName,siteUniqueID,siteDescription,siteEmailContact,siteUserSupportContact,

      siteSysAdminContact,siteSecurityContact,siteLocation,siteLatitude,siteLongitude
,

      siteWeb,siteOtherInfo,siteSchemaVersionMajor,siteSchemaVersionMinor}
      STATUS current
      DESCRIPTION "Object group for site related objects"
      ::= { dciCompliances 3 }

voInformationGroup OBJECT-GROUP
      OBJECTS { voName, voDescription }
      STATUS current
      DESCRIPTION "Object group for virtual organization related objects"
      ::= { dciCompliances 4 }

ceInformationGroup OBJECT-GROUP
      OBJECTS {
      ceQueueName,ceQueueUniqueID,ceQueueHostingCluster,ceQueueAccessControlBaseRule,

      ceQueueStateTotalJobs,ceQueueStateRunningJobs,ceQueueStateWaitingJobs,

      ceQueueStateFreeCPUs,ceQueueInfoTotalCPUs,ceQueuePolicyMaxRunningJobs,
                  ceQueueImplementationName,ceQueueImplementationVersion,

      ceVoQueueName,ceVoQueueStateTotalJobs,ceVoQueueStateRunningJobs,ceVoQueueStateW
aitingJobs,
                  ceVoQueueStateFreeJobSlots,ceQueueDescription,ceVoQueueDescription
}
      STATUS current
      DESCRIPTION "Object group for computing element related objects"
      ::= { dciCompliances 5 }

saInformationGroup OBJECT-GROUP
      OBJECTS {
      saPoolLocalID,saPoolTotalOnlineSize,saPoolUsedOnlineSize,saPoolFreeOnlineSize,
                  saPoolStateAvailableSpace,saPoolStateUsedSpace,saPoolDescription,

      saVoLocalID,saVoTotalOnlineSize,saVoUsedOnlineSize,saVoFreeOnlineSize,

      saVoStateAvailableSpace,saVoStateUsedSpace,saVoPath,saVoName,saPoolACBRule,saVo
Description}
      STATUS current
      DESCRIPTION "Object group for storage element related objects"
```

```
       ::= { dciCompliances 6 }

monitoringInformationGroup OBJECT-GROUP
       OBJECTS {     serviceName, serviceAbbr, serviceDescription, serviceEndpointVos,
                     serviceEndpointHostname, serviceEndpointCountryName,
serviceEndpointHostOS, serviceEndpointHostDN,
                     serviceEndpointHostArch, serviceEndpointIsMonitored,
serviceEndpointIsProduction,
                     serviceStatusID, serviceStatusTS,serviceStatusDescription,
serviceEndpointStatusDescription,
                     serviceEndpointNodeName, serviceEndpointStatusID,
serviceEndpointStatusTS, serviceEndpointDescription,
                     serviceARReliability, serviceARAvailability, serviceARUptime,
serviceARDowntime,
                     serviceARUnknown, serviceARTimestamp, serviceARDescription,
serviceEndpointARReliability,
                     serviceEndpointARAvailability, serviceEndpointARUptime,
serviceEndpointARDowntime,
                     serviceEndpointARUnknown, serviceEndpointARTimestamp,
serviceEndpointARDescription

              }
       STATUS current
       DESCRIPTION "Object group for functional monitoring environment per service and
service endpoint reporting"
       ::= { dciCompliances 7 }

hypervisorInformationGroup OBJECT-GROUP
       OBJECTS { hypervisorStatsCount, hypervisorStatsVcpus, hypervisorStatsVcpusUsed,
hypervisorStatsRam,
              hypervisorStatsRamUsed, hypervisorStatsDisk, hypervisorStatsDiskUsed,
hypervisorStatsRunningVms,
              hypervisorStatsCurrentWorkload, hypervisorVcpus, hypervisorVcpusUsed,
hypervisorRam,
              hypervisorRamUsed, hypervisorDisk, hypervisorDiskUsed,
hypervisorRunningVms,
              hypervisorCurrentWorkload, hypervisorHostname, hypervisorType,
hypervisorStatus,
              hypervisorState, hypervisorCpuInfo,
              vmInstanceName, vmInstanceHostname, vmInstanceZone, vmInstanceType,
vmInstanceImage, vmInstanceStatus, vmInstanceState, vmInstanceVcpus, vmInstanceRam,
vmInstanceSwap, vmInstanceDisk, vmInstanceDescription,
              vmImageName, vmImageSize, vmImageFormat, vmImageSource,
vmImageDescription,
              vmInstanceTypeName, vmInstanceTypeVcpus, vmInstanceTypeRam,
vmInstanceTypeSwap, vmInstanceTypeDisk, vmInstanceTypeDescription,
              vmStorageName, vmStorageDisk, vmStorageDiskUsed, vmStorageProtocol,
vmStorageSource, vmStorageDescription
       }
       STATUS current
       DESCRIPTION "Object group for detailed resource monitoring of hypervisors in VM
based infrastructure"
       ::= { dciCompliances 8 }

--
-- General information related to the system running the agent
--

systemInformation   OBJECT IDENTIFIER ::= { dciObjects 1 }

systemName OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
```

```
       STATUS       current
       DESCRIPTION  "Name of the system"
       ::= { systemInformation 1 }

systemDescription OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the system"
       ::= { systemInformation 2 }


systemContact OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Contact information for the system"
       ::= { systemInformation 3 }

--
-- Infrastructure information
--

infrastructureInformation OBJECT-IDENTITY
       STATUS current
       DESCRIPTION "Information on monitored infrastructure"
       ::= { dciObjects 2 }

--
-- Virtual organizations
--

voInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about available virtual organizations"
     ::= { infrastructureInformation 1 }

voTable OBJECT-TYPE
     SYNTAX     SEQUENCE OF VoEntry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION    "Table containing available virtual ogranizations"
     ::= { voInformation 1 }

voEntry OBJECT-TYPE
     SYNTAX     VoEntry
     MAX-ACCESS     not-accessible
        STATUS              current
        DESCRIPTION "Entry describing available virtual organization"
        INDEX       { voId }
        ::= { voTable 1 }

VoEntry      ::=
       SEQUENCE {
             voId                 LookupIndex,
             voName               DisplayString,
         voDescription         DisplayString
    }

voId   OBJECT-TYPE
       SYNTAX       LookupIndex
       MAX-ACCESS   not-accessible
       STATUS       current
```

```
       DESCRIPTION   "ID of the VO"
       ::= { voEntry 1 }

voName OBJECT-TYPE
       SYNTAX        DisplayString(SIZE(0..32))
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the VO"
       ::= { voEntry 2 }

voDescription OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the VO"
       ::= { voEntry 3 }

--
-- Monitored DCI services
--

serviceInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about monitored DCI services"
     ::= { infrastructureInformation 2 }

serviceTable OBJECT-TYPE
     SYNTAX     SEQUENCE OF ServiceEntry
     MAX-ACCESS not-accessible
     STATUS     current
     DESCRIPTION    "Table containing monitored DCI services"
     ::= { serviceInformation 1 }

serviceEntry OBJECT-TYPE
     SYNTAX     ServiceEntry
     MAX-ACCESS     not-accessible
        STATUS              current
        DESCRIPTION "Entry describing monitored DCI service"
        INDEX       { serviceId }
        ::= { serviceTable 1 }

ServiceEntry ::=
       SEQUENCE {
             serviceId                 LookupIndex,
             serviceName               DisplayString,
             serviceAbbr               DisplayString,
         serviceDescription         DisplayString
    }

serviceId    OBJECT-TYPE
       SYNTAX        LookupIndex
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "ID of the service"
       ::= { serviceEntry 1 }

serviceName  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the service"
       ::= { serviceEntry 2 }
```

```
serviceAbbr  OBJECT-TYPE
       SYNTAX       DisplayString(SIZE(0..16))
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Abbreviated name of the service"
       ::= { serviceEntry 3 }

serviceDescription  OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the service"
       ::= { serviceEntry 4 }

--
-- Site information
--

siteInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about the available sites"
     ::= { infrastructureInformation 3 }

siteTable    OBJECT-TYPE
       SYNTAX       SEQUENCE OF SiteEntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Table of available sites"
       ::= { siteInformation 1 }

siteEntry    OBJECT-TYPE
       SYNTAX       SiteEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Entry describing available site"
       INDEX        { siteId }
       ::= { siteTable 1 }

SiteEntry    ::=
       SEQUENCE {
             siteId                  LookupIndex,
             siteName                DisplayString,
        siteUniqueID            DisplayString,
             siteDescription         DisplayString,
        siteEmailContact        DisplayString,
        siteUserSupportContact  DisplayString,
        siteSysAdminContact     DisplayString,
        siteSecurityContact     DisplayString,
        siteLocation            DisplayString,
        siteLatitude            DisplayString,
        siteLongitude           DisplayString,
        siteWeb                 DisplayString,
        siteOtherInfo           DisplayString,
        siteSchemaVersionMajor      DisplayString,
        siteSchemaVersionMinor      DisplayString
       }

siteId OBJECT-TYPE
       SYNTAX       LookupIndex
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "ID of the site"
       ::= { siteEntry 1 }
```

```
siteName     OBJECT-TYPE
       SYNTAX        DisplayString(SIZE(0..32))
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the site"
       ::= { siteEntry 2 }

siteUniqueID  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Unique ID of the site"
       ::= { siteEntry 3 }

siteDescription     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the site"
       ::= { siteEntry 4 }

siteEmailContact    OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Email contact of the site"
       ::= { siteEntry 5 }

siteUserSupportContact     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "User support contact of the site"
       ::= { siteEntry 6 }

siteSysAdminContact OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "System administrator of the site"
       ::= { siteEntry 7 }

siteSecurityContact OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Security contact of the site"
       ::= { siteEntry 8 }

siteLocation  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Location of the site"
       ::= { siteEntry 9 }

siteLatitude  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Latitude of the site"
       ::= { siteEntry 10 }
```

```
siteLongitude OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Longitude of the site"
       ::= { siteEntry 11 }

siteWeb       OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Web site of the site"
       ::= { siteEntry 12 }

siteOtherInfo OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Other information for the site"
       ::= { siteEntry 13 }

siteSchemaVersionMajor     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Major version of the site middleware"
       ::= { siteEntry 14 }

siteSchemaVersionMinor     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Minor version of the site middleware"
       ::= { siteEntry 15 }

--
-- Service endpoint information
--

serviceEndpointInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about service endpoints"
     ::= { infrastructureInformation 4 }

serviceEndpointTable       OBJECT-TYPE
       SYNTAX        SEQUENCE OF ServiceEndpointEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Table of available service endpoints"
       ::= { serviceEndpointInformation 1 }

serviceEndpointEntry       OBJECT-TYPE
       SYNTAX        ServiceEndpointEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing service endpoint"
       INDEX         { siteId, serviceId, serviceEndpointId }
       ::= { serviceEndpointTable 1 }

ServiceEndpointEntry       ::=
       SEQUENCE {
              serviceEndpointId                  LookupIndex,
```

```
            serviceEndpointVos                                  DisplayString,
            serviceEndpointHostname                             DisplayString,
            serviceEndpointCountryName                          DisplayString,
            serviceEndpointHostOS                               DisplayString,
            serviceEndpointHostDN                               DisplayString,
            serviceEndpointHostArch                             DisplayString,
            serviceEndpointIsMonitored                          TruthValue,
            serviceEndpointIsProduction                         TruthValue,
            serviceEndpointDescription                          DisplayString
      }

serviceEndpointId   OBJECT-TYPE
      SYNTAX        LookupIndex
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION   "ID of the serviceEndpoint"
      ::= { serviceEndpointEntry 1 }

serviceEndpointVos OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Virtual organizations / projects the service endpoint belongs to"
      ::= { serviceEndpointEntry 2 }

serviceEndpointHostname OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Hostname of the host hosting the service endpoint"
      ::= { serviceEndpointEntry 3 }

serviceEndpointCountryName OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Name of the country hosting the service endpoint"
      ::= { serviceEndpointEntry 4 }

serviceEndpointHostOS OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Operating system of the host hosting the service endpoint"
      ::= { serviceEndpointEntry 5 }

serviceEndpointHostDN OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "DN of the host hosting the service endpoint"
      ::= { serviceEndpointEntry 6 }

serviceEndpointHostArch OBJECT-TYPE
      SYNTAX        DisplayString(SIZE(0..255))
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Architecture of the host hosting the service endpoint"
      ::= { serviceEndpointEntry 7 }

serviceEndpointIsMonitored OBJECT-TYPE
      SYNTAX        TruthValue
      MAX-ACCESS    read-only
```

```
       STATUS       current
       DESCRIPTION  "Is the service endpoint monitored"
       ::= { serviceEndpointEntry 8 }

serviceEndpointIsProduction OBJECT-TYPE
       SYNTAX       TruthValue
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Is the service endpoint in production"
       ::= { serviceEndpointEntry 9 }

serviceEndpointDescription OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of service endpoint"
       ::= { serviceEndpointEntry 10 }

--
-- Resource data
--

resourceInformation OBJECT-IDENTITY
       STATUS current
       DESCRIPTION "Information on monitored infrastructure"
       ::= { dciObjects 3 }


ceQueueTable  OBJECT-TYPE
       SYNTAX       SEQUENCE OF CeQueueEntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Supported queues on computing element"
       ::= { resourceInformation 1 }

ceQueueEntry  OBJECT-TYPE
       SYNTAX       CeQueueEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Entry describing supported queue"
       INDEX        { siteId, ceQueueId }
       ::= { ceQueueTable 1 }

CeQueueEntry ::=
       SEQUENCE {
         ceQueueId          LookupIndex,
         ceQueueName DisplayString,
         ceQueueUniqueID DisplayString,
         ceQueueHostingCluster   DisplayString,
         ceQueueAccessControlBaseRule    DisplayString,
         ceQueueStateTotalJobs Unsigned32,
         ceQueueStateRunningJobs Unsigned32,
         ceQueueStateWaitingJobs Unsigned32,
         ceQueueStateFreeCPUs    Unsigned32,
         ceQueueInfoTotalCPUs    Unsigned32,
         ceQueuePolicyMaxRunningJobs Unsigned32,
         ceQueueImplementationName   DisplayString,
         ceQueueImplementationVersion    DisplayString,
         ceQueueDescription    DisplayString
       }

ceQueueId    OBJECT-TYPE
       SYNTAX       LookupIndex
```

```
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "ID of the CE queue"
       ::= { ceQueueEntry 1 }

ceQueueName  OBJECT-TYPE
       SYNTAX        DisplayString(SIZE(0..32))
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the CE queue"
       ::= { ceQueueEntry 2 }

ceQueueUniqueID     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   " of the CE queue"
       ::= { ceQueueEntry 3 }

ceQueueHostingCluster      OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Hosting cluster of the CE queue"
       ::= { ceQueueEntry 4 }

ceQueueAccessControlBaseRule     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Access Control Base rule of the CE queue"
       ::= { ceQueueEntry 5 }

ceQueueStateTotalJobs      OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total jobs in the CE queue"
       ::= { ceQueueEntry 6 }

ceQueueStateRunningJobs    OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "RUnning jobs in the CE queue"
       ::= { ceQueueEntry 7 }

ceQueueStateWaitingJobs    OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Waiting jobs in the CE queue"
       ::= { ceQueueEntry 8 }

ceQueueStateFreeCPUs       OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Number of free CPUs in the CE queue"
       ::= { ceQueueEntry 9 }

ceQueueInfoTotalCPUs       OBJECT-TYPE
       SYNTAX        Unsigned32
```

```
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total number of CPUs in the CE queue"
       ::= { ceQueueEntry 10 }

ceQueuePolicyMaxRunningJobs      OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum number of jobs running in the CE queue"
       ::= { ceQueueEntry 11 }

ceQueueImplementationName  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Implementation name of the CE queue"
       ::= { ceQueueEntry 12 }

ceQueueImplementationVersion     OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Implementation version of the CE queue"
       ::= { ceQueueEntry 13 }

ceQueueDescription  OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the CE queue"
       ::= { ceQueueEntry 14 }


ceVoQueueTable      OBJECT-TYPE
       SYNTAX        SEQUENCE OF CeVoQueueEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Supported queues on computing element"
       ::= { resourceInformation 2 }

ceVoQueueEntry      OBJECT-TYPE
       SYNTAX        CeVoQueueEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing supported queue"
       INDEX         { siteId, ceQueueId, voId }
       ::= { ceVoQueueTable 1 }

CeVoQueueEntry      ::=
       SEQUENCE {
        ceVoQueueName   DisplayString,
        ceVoQueueStateTotalJobs Unsigned32,
        ceVoQueueStateRunningJobs Unsigned32,
        ceVoQueueStateWaitingJobs Unsigned32,
        ceVoQueueStateFreeJobSlots    Unsigned32,
        ceVoQueueDescription    DisplayString
        }

ceVoQueueName OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
```

```
       DESCRIPTION   "Name of the CE VO queue"
       ::= { ceVoQueueEntry 1 }

ceVoQueueStateTotalJobs    OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total jobs in the CE VO queue"
       ::= { ceVoQueueEntry 2 }

ceVoQueueStateRunningJobs  OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "RUnning jobs in the CE VO queue"
       ::= { ceVoQueueEntry 3 }

ceVoQueueStateWaitingJobs  OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Waiting jobs in the CE VO queue"
       ::= { ceVoQueueEntry 4 }

ceVoQueueStateFreeJobSlots OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Number of free job slots in the CE VO queue"
       ::= { ceVoQueueEntry 5 }

ceVoQueueDescription       OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the CE VO queue"
       ::= { ceVoQueueEntry 6 }

saPoolTable  OBJECT-TYPE
       SYNTAX        SEQUENCE OF SaPoolEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Supported pools on storage element"
       ::= { resourceInformation 3 }

saPoolEntry  OBJECT-TYPE
       SYNTAX        SaPoolEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing supported pool"
       INDEX         { siteId, saPoolId }
       ::= { saPoolTable 1 }

SaPoolEntry  ::=
       SEQUENCE {
              saPoolId            LookupIndex,
        saPoolLocalID   DisplayString,
        saPoolTotalOnlineSize   GBytes,
        saPoolUsedOnlineSize    GBytes,
        saPoolFreeOnlineSize    GBytes,
        saPoolStateAvailableSpace   GBytes,
        saPoolStateUsedSpace    GBytes,
        saPoolACBRule    DisplayString,
```

```
        saPoolDescription    DisplayString
    }

saPoolId    OBJECT-TYPE
       SYNTAX        LookupIndex
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "ID of the storage pool"
       ::= { saPoolEntry 1 }

saPoolLocalID    OBJECT-TYPE
       SYNTAX DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Local ID of the storage pool"
       ::= { saPoolEntry 2 }

saPoolTotalOnlineSize    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total size of the storage pool (GB)"
       ::= { saPoolEntry 3 }

saPoolUsedOnlineSize    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Used size of the storage pool (GB)"
       ::= { saPoolEntry 4 }

saPoolFreeOnlineSize    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Free size of the storage pool (GB)"
       ::= { saPoolEntry 5 }

saPoolStateAvailableSpace    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Available space state of the storage pool (GB)"
       ::= { saPoolEntry 6 }

saPoolStateUsedSpace    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Used space state of the storage pool (GB)"
       ::= { saPoolEntry 7 }

saPoolACBRule    OBJECT-TYPE
       SYNTAX DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Access Control Base Rule of the storage pool"
       ::= { saPoolEntry 8 }
```

```
saPoolDescription    OBJECT-TYPE
       SYNTAX DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the storage pool"
       ::= { saPoolEntry 9 }

saVoTable    OBJECT-TYPE
       SYNTAX        SEQUENCE OF SaVoEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Supported Vos on storage element"
       ::= { resourceInformation 4 }

saVoEntry    OBJECT-TYPE
       SYNTAX        SaVoEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing supported Vo"
       INDEX         { siteId, voId, saUniqueID }
       ::= { saVoTable 1 }

SaVoEntry    ::=
       SEQUENCE {
         saUniqueID  LookupIndex,
         saVoLocalID   DisplayString,
         saVoTotalOnlineSize   GBytes,
         saVoUsedOnlineSize    GBytes,
         saVoFreeOnlineSize    GBytes,
         saVoStateAvailableSpace   GBytes,
         saVoStateUsedSpace    GBytes,
         saVoPath    DisplayString,
         saVoName    DisplayString,
         saVoDescription    DisplayString
    }

saUniqueID    OBJECT-TYPE
       SYNTAX LookupIndex
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Unique ID of the storage node"
       ::= { saVoEntry 1 }

saVoLocalID    OBJECT-TYPE
       SYNTAX DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Local ID of the storage Vo"
       ::= { saVoEntry 2 }

saVoTotalOnlineSize    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total size of the storage Vo (GB)"
       ::= { saVoEntry 3 }

saVoUsedOnlineSize    OBJECT-TYPE
       SYNTAX        GBytes
    UNITS       "GBytes"
       MAX-ACCESS    read-only
```

```
       STATUS       current
       DESCRIPTION  "Used size of the storage Vo (GB)"
       ::= { saVoEntry 4 }

saVoFreeOnlineSize    OBJECT-TYPE
       SYNTAX       GBytes
    UNITS       "GBytes"
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Free size of the storage Vo (GB)"
       ::= { saVoEntry 5 }

saVoStateAvailableSpace    OBJECT-TYPE
       SYNTAX       GBytes
    UNITS       "GBytes"
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Available space state of the storage Vo (GB)"
       ::= { saVoEntry 6 }

saVoStateUsedSpace    OBJECT-TYPE
       SYNTAX       GBytes
    UNITS       "GBytes"
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Used space state of the storage Vo (GB)"
       ::= { saVoEntry 7 }

saVoPath    OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Vo Path"
       ::= { saVoEntry 8 }

saVoName    OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Vo name"
       ::= { saVoEntry 9 }

saVoDescription    OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the entry"
       ::= { saVoEntry 10 }

--
-- Summary hypervisor information for a site
--

hypervisorStatsInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about summary hypervisor statistics for a site"
     ::= { resourceInformation 5 }

hypervisorStatsTable       OBJECT-TYPE
       SYNTAX       SEQUENCE OF HypervisorStatsEntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Table of summary hypervisor statistics for a site"
```

```
      ::= { hypervisorStatsInformation 1 }

hypervisorStatsEntry      OBJECT-TYPE
      SYNTAX       HypervisorStatsEntry
      MAX-ACCESS   not-accessible
      STATUS       current
      DESCRIPTION  "Entry describing hypervisor statistics"
      INDEX        { siteId, hypervisorStatsId }
      ::= { hypervisorStatsTable 1 }

HypervisorStatsEntry      ::=
      SEQUENCE {
            hypervisorStatsId               LookupIndex,
            hypervisorStatsCount            Unsigned32,
            hypervisorStatsVcpus            Unsigned32,
            hypervisorStatsVcpusUsed        Unsigned32,
            hypervisorStatsRam              MBytes,
            hypervisorStatsRamUsed          MBytes,
            hypervisorStatsDisk             GBytes,
            hypervisorStatsDiskUsed         GBytes,
            hypervisorStatsRunningVms       Unsigned32,
            hypervisorStatsCurrentWorkload  Unsigned32
      }

hypervisorStatsId   OBJECT-TYPE
      SYNTAX       LookupIndex
      MAX-ACCESS   not-accessible
      STATUS       current
      DESCRIPTION  "ID of the hypervisor statistics entry"
      ::= { hypervisorStatsEntry 1 }

hypervisorStatsCount OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Number of hypervisors"
      ::= { hypervisorStatsEntry 2 }

hypervisorStatsVcpus OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total number of vcpus"
      ::= { hypervisorStatsEntry 3 }

hypervisorStatsVcpusUsed OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total number of vcpus used"
      ::= { hypervisorStatsEntry 4 }

hypervisorStatsRam OBJECT-TYPE
      SYNTAX       MBytes
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total amount of RAM"
      ::= { hypervisorStatsEntry 5 }

hypervisorStatsRamUsed OBJECT-TYPE
      SYNTAX       MBytes
      MAX-ACCESS   read-only
      STATUS       current
```

```
       DESCRIPTION   "Total amount of RAM used"
       ::= { hypervisorStatsEntry 6 }

hypervisorStatsDisk OBJECT-TYPE
       SYNTAX        GBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total amount of disk"
       ::= { hypervisorStatsEntry 7 }

hypervisorStatsDiskUsed OBJECT-TYPE
       SYNTAX        GBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total amount of disk used"
       ::= { hypervisorStatsEntry 8 }

hypervisorStatsRunningVms OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total number of running virtual machines"
       ::= { hypervisorStatsEntry 9 }

hypervisorStatsCurrentWorkload OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total load - usually equal to the number of running VMs"
       ::= { hypervisorStatsEntry 10 }


--
-- Individual hypervisors' information for a site
--

hypervisorInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about individual hypervisors for a site"
     ::= { resourceInformation 6 }

hypervisorTable     OBJECT-TYPE
       SYNTAX        SEQUENCE OF HypervisorEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Table of individual hypervisors for a site"
       ::= { hypervisorInformation 1 }

hypervisorEntry     OBJECT-TYPE
       SYNTAX        HypervisorEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing hypervisor"
       INDEX         { siteId, hypervisorId }
       ::= { hypervisorTable 1 }

HypervisorEntry     ::=
       SEQUENCE {
              hypervisorId             LookupIndex,
              hypervisorVcpus                     Unsigned32,
              hypervisorVcpusUsed                 Unsigned32,
              hypervisorRam                       MBytes,
              hypervisorRamUsed                   MBytes,
```

```
            hypervisorDisk                      GBytes,
            hypervisorDiskUsed                  GBytes,
            hypervisorRunningVms                Unsigned32,
            hypervisorCurrentWorkload           Unsigned32,
            hypervisorHostname                  DisplayString,
            hypervisorType                      DisplayString,
            hypervisorStatus                    DisplayString,
            hypervisorState                     DisplayString,
            hypervisorCpuInfo                   DisplayString
      }

hypervisorId OBJECT-TYPE
      SYNTAX       LookupIndex
      MAX-ACCESS   not-accessible
      STATUS       current
      DESCRIPTION  "ID of the hypervisor entry"
      ::= { hypervisorEntry 1 }

hypervisorVcpus OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total number of vcpus"
      ::= { hypervisorEntry 2 }

hypervisorVcpusUsed OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total number of vcpus used"
      ::= { hypervisorEntry 3 }

hypervisorRam OBJECT-TYPE
      SYNTAX       MBytes
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total amount of RAM"
      ::= { hypervisorEntry 4 }

hypervisorRamUsed OBJECT-TYPE
      SYNTAX       MBytes
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total amount of RAM used"
      ::= { hypervisorEntry 5 }

hypervisorDisk OBJECT-TYPE
      SYNTAX       GBytes
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total amount of disk"
      ::= { hypervisorEntry 6 }

hypervisorDiskUsed OBJECT-TYPE
      SYNTAX       GBytes
      MAX-ACCESS   read-only
      STATUS       current
      DESCRIPTION  "Total amount of disk used"
      ::= { hypervisorEntry 7 }

hypervisorRunningVms OBJECT-TYPE
      SYNTAX       Unsigned32
      MAX-ACCESS   read-only
```

```
       STATUS       current
       DESCRIPTION  "Total number of running virtual machines"
       ::= { hypervisorEntry 8 }

hypervisorCurrentWorkload OBJECT-TYPE
       SYNTAX       Unsigned32
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Total load - usually equal to the number of running VMs"
       ::= { hypervisorEntry 9 }

hypervisorHostname OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Hypervisor hostname"
       ::= { hypervisorEntry 10 }

hypervisorType OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Type of the hypervisor"
       ::= { hypervisorEntry 11 }

hypervisorStatus OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Status of the hypervisor"
       ::= { hypervisorEntry 12 }

hypervisorState OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "State of the hypervisor"
       ::= { hypervisorEntry 13 }

hypervisorCpuInfo OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "CPU information of the hypervisor"
       ::= { hypervisorEntry 14 }

--
-- Individual VM instance's information
--

vmInstanceInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about individual VM instance"
     ::= { resourceInformation 7 }

vmInstanceTable     OBJECT-TYPE
       SYNTAX       SEQUENCE OF VmInstanceEntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Table of individual VM instances"
       ::= { vmInstanceInformation 1 }

vmInstanceEntry     OBJECT-TYPE
```

```
        SYNTAX        VmInstanceEntry
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION   "Entry describing VM instance"
        INDEX         { siteId, vmInstanceId }
        ::= { vmInstanceTable 1 }

VmInstanceEntry     ::=
        SEQUENCE {
                vmInstanceId             LookupIndex,
                vmInstanceName           DisplayString,
                vmInstanceHostname       DisplayString,
                vmInstanceZone           DisplayString,
                vmInstanceType           DisplayString,
                vmInstanceImage          DisplayString,
                vmInstanceStatus         DisplayString,
                vmInstanceState          DisplayString,
                vmInstanceVcpus          Unsigned32,
                vmInstanceRam            MBytes,
                vmInstanceSwap           MBytes,
                vmInstanceDisk           GBytes,
                vmInstanceDescription    DisplayString
        }

vmInstanceId OBJECT-TYPE
        SYNTAX        LookupIndex
        MAX-ACCESS    not-accessible
        STATUS        current
        DESCRIPTION   "ID of the vmInstance entry"
        ::= { vmInstanceEntry 1 }

vmInstanceName OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION   "Name of the instance"
        ::= { vmInstanceEntry 2 }

vmInstanceHostname OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION   "Hostname of the host hosting the instance"
        ::= { vmInstanceEntry 3 }

vmInstanceZone OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION   "Availability zone of the instance"
        ::= { vmInstanceEntry 4 }

vmInstanceType OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-only
        STATUS        current
        DESCRIPTION   "Instance type/flavor"
        ::= { vmInstanceEntry 5 }

vmInstanceImage OBJECT-TYPE
        SYNTAX        DisplayString
        MAX-ACCESS    read-only
        STATUS        current
```

```
       DESCRIPTION   "Instance image"
       ::= { vmInstanceEntry 6 }

vmInstanceStatus OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Status of the instance"
       ::= { vmInstanceEntry 7 }

vmInstanceState OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "State of the instance"
       ::= { vmInstanceEntry 8 }

vmInstanceVcpus OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum number of VCPUs for instance"
       ::= { vmInstanceEntry 9 }

vmInstanceRam OBJECT-TYPE
       SYNTAX        MBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum RAM for instance in MB"
       ::= { vmInstanceEntry 10 }

vmInstanceSwap OBJECT-TYPE
       SYNTAX        MBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum swap for instance in MB"
       ::= { vmInstanceEntry 11 }

vmInstanceDisk OBJECT-TYPE
       SYNTAX        GBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum size of disk for instance in GB"
       ::= { vmInstanceEntry 12 }

vmInstanceDescription OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Description of the instance"
       ::= { vmInstanceEntry 13 }

--
-- Individual VM image's information
--

vmImageInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about individual VM image"
     ::= { resourceInformation 8 }

vmImageTable  OBJECT-TYPE
       SYNTAX        SEQUENCE OF VmImageEntry
```

```
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Table of individual VM images"
       ::= { vmImageInformation 1 }

vmImageEntry OBJECT-TYPE
       SYNTAX       VmImageEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Entry describing VM image"
       INDEX        { siteId, vmImageId }
       ::= { vmImageTable 1 }

VmImageEntry ::=
       SEQUENCE {
              vmImageId             LookupIndex,
              vmImageName           DisplayString,
              vmImageSize           GBytes,
              vmImageFormat         DisplayString,
              vmImageSource         DisplayString,
              vmImageDescription    DisplayString
       }

vmImageId    OBJECT-TYPE
       SYNTAX       LookupIndex
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "ID of the vmImage entry"
       ::= { vmImageEntry 1 }

vmImageName OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Name of the image"
       ::= { vmImageEntry 2 }

vmImageSize OBJECT-TYPE
       SYNTAX       GBytes
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Size of the image in GB"
       ::= { vmImageEntry 3 }

vmImageFormat OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Format of the image"
       ::= { vmImageEntry 4 }

vmImageSource OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Source of the image"
       ::= { vmImageEntry 5 }

vmImageDescription OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the image"
```

```
       ::= { vmImageEntry 6 }

--
-- Individual VM instance's information
--

vmInstanceTypeInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about VM instance types / flavors"
     ::= { resourceInformation 9 }

vmInstanceTypeTable        OBJECT-TYPE
       SYNTAX        SEQUENCE OF VmInstanceTypeEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Table of VM instance types / flavors"
       ::= { vmInstanceTypeInformation 1 }

vmInstanceTypeEntry OBJECT-TYPE
       SYNTAX        VmInstanceTypeEntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing VM instance type"
       INDEX         { siteId, vmInstanceTypeId }
       ::= { vmInstanceTypeTable 1 }

VmInstanceTypeEntry ::=
       SEQUENCE {
              vmInstanceTypeId             LookupIndex,
              vmInstanceTypeName           DisplayString,
              vmInstanceTypeVcpus          Unsigned32,
              vmInstanceTypeRam            MBytes,
              vmInstanceTypeSwap           MBytes,
              vmInstanceTypeDisk           GBytes,
              vmInstanceTypeDescription    DisplayString
       }

vmInstanceTypeId    OBJECT-TYPE
       SYNTAX        LookupIndex
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "ID of the vmInstanceType entry"
       ::= { vmInstanceTypeEntry 1 }

vmInstanceTypeName OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the instance"
       ::= { vmInstanceTypeEntry 2 }

vmInstanceTypeVcpus OBJECT-TYPE
       SYNTAX        Unsigned32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum number of VCPUs for instance"
       ::= { vmInstanceTypeEntry 3 }

vmInstanceTypeRam OBJECT-TYPE
       SYNTAX        MBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Maximum RAM for instance in MB"
```

```
       ::= { vmInstanceTypeEntry 4 }

vmInstanceTypeSwap OBJECT-TYPE
       SYNTAX       MBytes
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Maximum swap for instance in MB"
       ::= { vmInstanceTypeEntry 5 }

vmInstanceTypeDisk OBJECT-TYPE
       SYNTAX       GBytes
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Maximum size of disk for instance in GB"
       ::= { vmInstanceTypeEntry 6 }

vmInstanceTypeDescription OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the instance"
       ::= { vmInstanceTypeEntry 7 }

--
-- Individual VM instance's information
--

vmStorageInformation OBJECT-IDENTITY
     STATUS current
     DESCRIPTION "Information about VM storage"
     ::= { resourceInformation 10 }

vmStorageTable      OBJECT-TYPE
       SYNTAX       SEQUENCE OF VmStorageEntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Table of VM storage"
       ::= { vmStorageInformation 1 }

vmStorageEntry      OBJECT-TYPE
       SYNTAX       VmStorageEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Entry describing VM storage"
       INDEX        { siteId, vmStorageId }
       ::= { vmStorageTable 1 }

VmStorageEntry      ::=
       SEQUENCE {
              vmStorageId             LookupIndex,
              vmStorageName           DisplayString,
              vmStorageDisk           GBytes,
              vmStorageDiskUsed       GBytes,
              vmStorageProtocol       DisplayString,
              vmStorageSource         DisplayString,
              vmStorageDescription    DisplayString
       }

vmStorageId  OBJECT-TYPE
       SYNTAX       LookupIndex
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "ID of the vmStorage entry"
```

```
       ::= { vmStorageEntry 1 }

vmStorageName OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Name of the instance"
       ::= { vmStorageEntry 2 }

vmStorageDisk OBJECT-TYPE
       SYNTAX        GBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Total size of storage in GB"
       ::= { vmStorageEntry 3 }

vmStorageDiskUsed OBJECT-TYPE
       SYNTAX        GBytes
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Used size of storage in GB"
       ::= { vmStorageEntry 4 }

vmStorageProtocol OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Storage protocol"
       ::= { vmStorageEntry 5 }

vmStorageSource OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Storage source"
       ::= { vmStorageEntry 6 }

vmStorageDescription OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Storage description"
       ::= { vmStorageEntry 7 }

--
-- Functional monitoring data
--

monitoringInformation OBJECT-IDENTITY
       STATUS current
       DESCRIPTION "Information on monitored infrastructure"
       ::= { dciObjects 4 }


--
-- Service status
--

serviceStatusTable  OBJECT-TYPE
       SYNTAX        SEQUENCE OF ServiceStatusEntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Status of monitored DCI service"
```

```
      ::= { monitoringInformation 1 }

serviceStatusEntry  OBJECT-TYPE
      SYNTAX        ServiceStatusEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION   "Entry describing status of monitored DCI service entry"
      INDEX         { siteId, voId, serviceId }
      ::= { serviceStatusTable 1 }

ServiceStatusEntry  ::=
      SEQUENCE {
       serviceStatusID ServiceStatusDef,
       serviceStatusTS DisplayString,
       serviceStatusDescription DisplayString
    }

serviceStatusID OBJECT-TYPE
    SYNTAX      ServiceStatusDef
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Status ID of the service status"
      ::= { serviceStatusEntry 1 }

serviceStatusTS OBJECT-TYPE
    SYNTAX      DisplayString
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Timestamp of the service status"
      ::= { serviceStatusEntry 2 }

serviceStatusDescription OBJECT-TYPE
    SYNTAX      DisplayString
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION   "Description of the service"
      ::= { serviceStatusEntry 3 }

--
-- Service endpoint status - status of individual service endpoint
--

serviceEndpointStatusTable       OBJECT-TYPE
      SYNTAX        SEQUENCE OF ServiceEndpointStatusEntry
      MAX-ACCESS    not-accessible
      STATUS              current
      DESCRIPTION "Status of monitored DCI service endpoint"
      ::= { monitoringInformation 2 }

serviceEndpointStatusEntry OBJECT-TYPE
      SYNTAX        ServiceEndpointStatusEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION   "Entry describing status of monitored DCI service endpoint entry"
      INDEX         { siteId, voId, serviceId, serviceEndpointNodeId }
      ::= { serviceEndpointStatusTable 1 }

ServiceEndpointStatusEntry ::=
      SEQUENCE {
       serviceEndpointNodeId    LookupIndex,
       serviceEndpointNodeName    DisplayString,
       serviceEndpointStatusID ServiceEndpointStatusDef,
       serviceEndpointStatusTS  DisplayString,
```

```
        serviceEndpointStatusDescription  DisplayString
    }

serviceEndpointNodeId OBJECT-TYPE
    SYNTAX      LookupIndex
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Unique node index"
       ::= { serviceEndpointStatusEntry 1 }

serviceEndpointNodeName OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "FQDN of the node"
       ::= { serviceEndpointStatusEntry 2 }

serviceEndpointStatusID OBJECT-TYPE
    SYNTAX      ServiceEndpointStatusDef
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Status ID of the service endpoint status"
       ::= { serviceEndpointStatusEntry 3 }

serviceEndpointStatusTS OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Timestamp of the service endpoint status"
       ::= { serviceEndpointStatusEntry 4 }

serviceEndpointStatusDescription OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION  "Description of the service endpoint"
       ::= { serviceEndpointStatusEntry 5 }

--
-- Service availability and reliability
--

serviceARTable      OBJECT-TYPE
       SYNTAX       SEQUENCE OF ServiceAREntry
       MAX-ACCESS   not-accessible
       STATUS              current
       DESCRIPTION "Availability and Reliability of monitored DCI service"
       ::= { monitoringInformation 3 }

serviceAREntry      OBJECT-TYPE
       SYNTAX       ServiceAREntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION  "Entry describing A/R of monitored DCI service entry"
       INDEX        { siteId, voId, serviceId }
       ::= { serviceARTable 1 }

ServiceAREntry      ::=
       SEQUENCE {
        serviceARReliability     DecimalValueMillis,
        serviceARAvailability    DecimalValueMillis,
        serviceARUptime          DecimalValueMillis,
        serviceARDowntime        DecimalValueMillis,
```

```
        serviceARUnknown         DecimalValueMillis,
        serviceARTimestamp              DisplayString,
        serviceARDescription     DisplayString
    }

serviceARReliability OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service Reliability"
       ::= { serviceAREntry 1 }

serviceARAvailability OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service Availability"
       ::= { serviceAREntry 2 }

serviceARUptime OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service Uptime"
       ::= { serviceAREntry 3 }

serviceARDowntime OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service Downtime"
       ::= { serviceAREntry 4 }

serviceARUnknown OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service unknown time"
       ::= { serviceAREntry 5 }

serviceARTimestamp OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Entry TimeStamp"
       ::= { serviceAREntry 6 }

serviceARDescription OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Entry Description"
       ::= { serviceAREntry 7 }

--
-- Service Endpoint Availability and Reliability
--

serviceEndpointARTable     OBJECT-TYPE
       SYNTAX        SEQUENCE OF ServiceEndpointAREntry
       MAX-ACCESS    not-accessible
       STATUS              current
       DESCRIPTION "Availability and Reliability of monitored DCI service endpoint"
```

```
      ::= { monitoringInformation 4 }

serviceEndpointAREntry     OBJECT-TYPE
       SYNTAX        ServiceEndpointAREntry
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION   "Entry describing A/R of monitored DCI service endpoint"
       INDEX         { siteId, voId, serviceId }
       ::= { serviceEndpointARTable 1 }

ServiceEndpointAREntry     ::=
       SEQUENCE {
         serviceEndpointARReliability    DecimalValueMillis,
         serviceEndpointARAvailability   DecimalValueMillis,
         serviceEndpointARUptime         DecimalValueMillis,
         serviceEndpointARDowntime              DecimalValueMillis,
         serviceEndpointARUnknown        DecimalValueMillis,
         serviceEndpointARTimestamp             DisplayString,
         serviceEndpointARDescription    DisplayString
    }

serviceEndpointARReliability OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service endpoint Reliability"
       ::= { serviceEndpointAREntry 1 }

serviceEndpointARAvailability OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service endpoint Availability"
       ::= { serviceEndpointAREntry 2 }

serviceEndpointARUptime OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service endpoint Uptime"
       ::= { serviceEndpointAREntry 3 }

serviceEndpointARDowntime OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service endpoint Downtime"
       ::= { serviceEndpointAREntry 4 }

serviceEndpointARUnknown OBJECT-TYPE
    SYNTAX      DecimalValueMillis
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Service endpoint unknown time"
       ::= { serviceEndpointAREntry 5 }

serviceEndpointARTimestamp OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Entry TimeStamp"
       ::= { serviceEndpointAREntry 6 }
```

```
serviceEndpointARDescription OBJECT-TYPE
    SYNTAX      DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION   "Entry Description"
       ::= { serviceEndpointAREntry 7 }

END
```

PRILOG B - ETFBL-DCI-MIB PUNO STABLO REGISTRACIJA

```
# ETFBL-DCI-MIB registration tree (generated by smidump 0.4.8)

--dci(1.3.6.1.4.1.33943.28)
 +--dciObjects(1)
 |  +--systemInformation(1)
 |  |  +-- r-n DisplayString systemName(1)
 |  |  +-- r-n DisplayString systemDescription(2)
 |  |  +-- r-n DisplayString systemContact(3)
 |  +--infrastructureInformation(2)
 |  |  +--voInformation(1)
 |  |  |  +--voTable(1)
 |  |  |     +--voEntry(1) [voId]
 |  |  |        +-- --- LookupIndex   voId(1)
 |  |  |        +-- r-n DisplayString voName(2)
 |  |  |        +-- r-n DisplayString voDescription(3)
 |  |  +--serviceInformation(2)
 |  |  |  +--serviceTable(1)
 |  |  |     +--serviceEntry(1) [serviceId]
 |  |  |        +-- --- LookupIndex   serviceId(1)
 |  |  |        +-- r-n DisplayString serviceName(2)
 |  |  |        +-- r-n DisplayString serviceAbbr(3)
 |  |  |        +-- r-n DisplayString serviceDescription(4)
 |  |  +--siteInformation(3)
 |  |  |  +--siteTable(1)
 |  |  |     +--siteEntry(1) [siteId]
 |  |  |        +-- --- LookupIndex   siteId(1)
 |  |  |        +-- r-n DisplayString siteName(2)
 |  |  |        +-- r-n DisplayString siteUniqueID(3)
 |  |  |        +-- r-n DisplayString siteDescription(4)
 |  |  |        +-- r-n DisplayString siteEmailContact(5)
 |  |  |        +-- r-n DisplayString siteUserSupportContact(6)
 |  |  |        +-- r-n DisplayString siteSysAdminContact(7)
 |  |  |        +-- r-n DisplayString siteSecurityContact(8)
 |  |  |        +-- r-n DisplayString siteLocation(9)
 |  |  |        +-- r-n DisplayString siteLatitude(10)
 |  |  |        +-- r-n DisplayString siteLongitude(11)
 |  |  |        +-- r-n DisplayString siteWeb(12)
 |  |  |        +-- r-n DisplayString siteOtherInfo(13)
 |  |  |        +-- r-n DisplayString siteSchemaVersionMajor(14)
 |  |  |        +-- r-n DisplayString siteSchemaVersionMinor(15)
 |  |  +--serviceEndpointInformation(4)
 |  |     +--serviceEndpointTable(1)
 |  |        +--serviceEndpointEntry(1) [siteId,serviceId,serviceEndpointId]
 |  |           +-- --- LookupIndex   serviceEndpointId(1)
 |  |           +-- r-n DisplayString serviceEndpointVos(2)
 |  |           +-- r-n DisplayString serviceEndpointHostname(3)
 |  |           +-- r-n DisplayString serviceEndpointCountryName(4)
 |  |           +-- r-n DisplayString serviceEndpointHostOS(5)
```

```
|   |           +-- r-n DisplayString serviceEndpointHostDN(6)
|   |           +-- r-n DisplayString serviceEndpointHostArch(7)
|   |           +-- r-n TruthValue    serviceEndpointIsMonitored(8)
|   |           +-- r-n TruthValue    serviceEndpointIsProduction(9)
|   |           +-- r-n DisplayString serviceEndpointDescription(10)
|   +--resourceInformation(3)
|   |   +--ceQueueTable(1)
|   |   |   +--ceQueueEntry(1) [siteId,ceQueueId]
|   |   |      +-- ---  LookupIndex   ceQueueId(1)
|   |   |      +-- r-n DisplayString ceQueueName(2)
|   |   |      +-- r-n DisplayString ceQueueUniqueID(3)
|   |   |      +-- r-n DisplayString ceQueueHostingCluster(4)
|   |   |      +-- r-n DisplayString ceQueueAccessControlBaseRule(5)
|   |   |      +-- r-n Unsigned32    ceQueueStateTotalJobs(6)
|   |   |      +-- r-n Unsigned32    ceQueueStateRunningJobs(7)
|   |   |      +-- r-n Unsigned32    ceQueueStateWaitingJobs(8)
|   |   |      +-- r-n Unsigned32    ceQueueStateFreeCPUs(9)
|   |   |      +-- r-n Unsigned32    ceQueueInfoTotalCPUs(10)
|   |   |      +-- r-n Unsigned32    ceQueuePolicyMaxRunningJobs(11)
|   |   |      +-- r-n DisplayString ceQueueImplementationName(12)
|   |   |      +-- r-n DisplayString ceQueueImplementationVersion(13)
|   |   |      +-- r-n DisplayString ceQueueDescription(14)
|   |   +--ceVoQueueTable(2)
|   |   |   +--ceVoQueueEntry(1) [siteId,ceQueueId,voId]
|   |   |      +-- r-n DisplayString ceVoQueueName(1)
|   |   |      +-- r-n Unsigned32    ceVoQueueStateTotalJobs(2)
|   |   |      +-- r-n Unsigned32    ceVoQueueStateRunningJobs(3)
|   |   |      +-- r-n Unsigned32    ceVoQueueStateWaitingJobs(4)
|   |   |      +-- r-n Unsigned32    ceVoQueueStateFreeJobSlots(5)
|   |   |      +-- r-n DisplayString ceVoQueueDescription(6)
|   |   +--saPoolTable(3)
|   |   |   +--saPoolEntry(1) [siteId,saPoolId]
|   |   |      +-- ---  LookupIndex   saPoolId(1)
|   |   |      +-- r-n DisplayString saPoolLocalID(2)
|   |   |      +-- r-n GBytes        saPoolTotalOnlineSize(3)
|   |   |      +-- r-n GBytes        saPoolUsedOnlineSize(4)
|   |   |      +-- r-n GBytes        saPoolFreeOnlineSize(5)
|   |   |      +-- r-n GBytes        saPoolStateAvailableSpace(6)
|   |   |      +-- r-n GBytes        saPoolStateUsedSpace(7)
|   |   |      +-- r-n DisplayString saPoolACBRule(8)
|   |   |      +-- r-n DisplayString saPoolDescription(9)
|   |   +--saVoTable(4)
|   |   |   +--saVoEntry(1) [siteId,voId,saUniqueID]
|   |   |      +-- ---  LookupIndex   saUniqueID(1)
|   |   |      +-- r-n DisplayString saVoLocalID(2)
|   |   |      +-- r-n GBytes        saVoTotalOnlineSize(3)
|   |   |      +-- r-n GBytes        saVoUsedOnlineSize(4)
|   |   |      +-- r-n GBytes        saVoFreeOnlineSize(5)
|   |   |      +-- r-n GBytes        saVoStateAvailableSpace(6)
|   |   |      +-- r-n GBytes        saVoStateUsedSpace(7)
|   |   |      +-- r-n DisplayString saVoPath(8)
|   |   |      +-- r-n DisplayString saVoName(9)
|   |   |      +-- r-n DisplayString saVoDescription(10)
|   |   +--hypervisorStatsInformation(5)
|   |   |   +--hypervisorStatsTable(1)
|   |   |      +--hypervisorStatsEntry(1) [siteId,hypervisorStatsId]
|   |   |         +-- ---  LookupIndex hypervisorStatsId(1)
|   |   |         +-- r-n Unsigned32  hypervisorStatsCount(2)
|   |   |         +-- r-n Unsigned32  hypervisorStatsVcpus(3)
|   |   |         +-- r-n Unsigned32  hypervisorStatsVcpusUsed(4)
|   |   |         +-- r-n MBytes      hypervisorStatsRam(5)
|   |   |         +-- r-n MBytes      hypervisorStatsRamUsed(6)
|   |   |         +-- r-n GBytes      hypervisorStatsDisk(7)
```

```
|   |   |         +-- r-n GBytes      hypervisorStatsDiskUsed(8)
|   |   |         +-- r-n Unsigned32  hypervisorStatsRunningVms(9)
|   |   |         +-- r-n Unsigned32  hypervisorStatsCurrentWorkload(10)
|   |   +--hypervisorInformation(6)
|   |   |  +--hypervisorTable(1)
|   |   |     +--hypervisorEntry(1) [siteId,hypervisorId]
|   |   |        +-- ---  LookupIndex   hypervisorId(1)
|   |   |        +-- r-n Unsigned32     hypervisorVcpus(2)
|   |   |        +-- r-n Unsigned32     hypervisorVcpusUsed(3)
|   |   |        +-- r-n MBytes         hypervisorRam(4)
|   |   |        +-- r-n MBytes         hypervisorRamUsed(5)
|   |   |        +-- r-n GBytes         hypervisorDisk(6)
|   |   |        +-- r-n GBytes         hypervisorDiskUsed(7)
|   |   |        +-- r-n Unsigned32     hypervisorRunningVms(8)
|   |   |        +-- r-n Unsigned32     hypervisorCurrentWorkload(9)
|   |   |        +-- r-n DisplayString hypervisorHostname(10)
|   |   |        +-- r-n DisplayString hypervisorType(11)
|   |   |        +-- r-n DisplayString hypervisorStatus(12)
|   |   |        +-- r-n DisplayString hypervisorState(13)
|   |   |        +-- r-n DisplayString hypervisorCpuInfo(14)
|   |   +--vmInstanceInformation(7)
|   |   |  +--vmInstanceTable(1)
|   |   |     +--vmInstanceEntry(1) [siteId,vmInstanceId]
|   |   |        +-- ---  LookupIndex   vmInstanceId(1)
|   |   |        +-- r-n DisplayString vmInstanceName(2)
|   |   |        +-- r-n DisplayString vmInstanceHostname(3)
|   |   |        +-- r-n DisplayString vmInstanceZone(4)
|   |   |        +-- r-n DisplayString vmInstanceType(5)
|   |   |        +-- r-n DisplayString vmInstanceImage(6)
|   |   |        +-- r-n DisplayString vmInstanceStatus(7)
|   |   |        +-- r-n DisplayString vmInstanceState(8)
|   |   |        +-- r-n Unsigned32    vmInstanceVcpus(9)
|   |   |        +-- r-n MBytes        vmInstanceRam(10)
|   |   |        +-- r-n MBytes        vmInstanceSwap(11)
|   |   |        +-- r-n GBytes        vmInstanceDisk(12)
|   |   |        +-- r-n DisplayString vmInstanceDescription(13)
|   |   +--vmImageInformation(8)
|   |   |  +--vmImageTable(1)
|   |   |     +--vmImageEntry(1) [siteId,vmImageId]
|   |   |        +-- ---  LookupIndex   vmImageId(1)
|   |   |        +-- r-n DisplayString vmImageName(2)
|   |   |        +-- r-n GBytes        vmImageSize(3)
|   |   |        +-- r-n DisplayString vmImageFormat(4)
|   |   |        +-- r-n DisplayString vmImageSource(5)
|   |   |        +-- r-n DisplayString vmImageDescription(6)
|   |   +--vmInstanceTypeInformation(9)
|   |   |  +--vmInstanceTypeTable(1)
|   |   |     +--vmInstanceTypeEntry(1) [siteId,vmInstanceTypeId]
|   |   |        +-- ---  LookupIndex   vmInstanceTypeId(1)
|   |   |        +-- r-n DisplayString vmInstanceTypeName(2)
|   |   |        +-- r-n Unsigned32    vmInstanceTypeVcpus(3)
|   |   |        +-- r-n MBytes        vmInstanceTypeRam(4)
|   |   |        +-- r-n MBytes        vmInstanceTypeSwap(5)
|   |   |        +-- r-n GBytes        vmInstanceTypeDisk(6)
|   |   |        +-- r-n DisplayString vmInstanceTypeDescription(7)
|   |   +--vmStorageInformation(10)
|   |      +--vmStorageTable(1)
|   |         +--vmStorageEntry(1) [siteId,vmStorageId]
|   |            +-- ---  LookupIndex   vmStorageId(1)
|   |            +-- r-n DisplayString vmStorageName(2)
|   |            +-- r-n GBytes        vmStorageDisk(3)
|   |            +-- r-n GBytes        vmStorageDiskUsed(4)
|   |            +-- r-n DisplayString vmStorageProtocol(5)
```

```
|   |           +-- r-n DisplayString vmStorageSource(6)
|   |           +-- r-n DisplayString vmStorageDescription(7)
|   +--monitoringInformation(4)
|      +--serviceStatusTable(1)
|      |  +--serviceStatusEntry(1) [siteId,voId,serviceId]
|      |     +-- r-n ServiceStatusDef serviceStatusID(1)
|      |     +-- r-n DisplayString    serviceStatusTS(2)
|      |     +-- r-n DisplayString    serviceStatusDescription(3)
|      +--serviceEndpointStatusTable(2)
|      |  +--serviceEndpointStatusEntry(1) [siteId,voId,serviceId,
|      |                                    serviceEndpointNodeId]
|      |     +-- --- LookupIndex              serviceEndpointNodeId(1)
|      |     +-- r-n DisplayString            serviceEndpointNodeName(2)
|      |     +-- r-n ServiceEndpointStatusDef serviceEndpointStatusID(3)
|      |     +-- r-n DisplayString            serviceEndpointStatusTS(4)
|      |     +-- r-n DisplayString            serviceEndpointStatusDescription(5)
|      +--serviceARTable(3)
|      |  +--serviceAREntry(1) [siteId,voId,serviceId]
|      |     +-- r-n DecimalValueMillis serviceARReliability(1)
|      |     +-- r-n DecimalValueMillis serviceARAvailability(2)
|      |     +-- r-n DecimalValueMillis serviceARUptime(3)
|      |     +-- r-n DecimalValueMillis serviceARDowntime(4)
|      |     +-- r-n DecimalValueMillis serviceARUnknown(5)
|      |     +-- r-n DisplayString      serviceARTimestamp(6)
|      |     +-- r-n DisplayString      serviceARDescription(7)
|      +--serviceEndpointARTable(4)
|         +--serviceEndpointAREntry(1) [siteId,voId,serviceId]
|            +-- r-n DecimalValueMillis serviceEndpointARReliability(1)
|            +-- r-n DecimalValueMillis serviceEndpointARAvailability(2)
|            +-- r-n DecimalValueMillis serviceEndpointARUptime(3)
|            +-- r-n DecimalValueMillis serviceEndpointARDowntime(4)
|            +-- r-n DecimalValueMillis serviceEndpointARUnknown(5)
|            +-- r-n DisplayString      serviceEndpointARTimestamp(6)
|            +-- r-n DisplayString      serviceEndpointARDescription(7)
+--dciEvents(2)
+--dciCompliances(3)
   +--generalCompliance(1)
   +--systemInformationGroup(2)
   +--siteInformationGroup(3)
   +--voInformationGroup(4)
   +--ceInformationGroup(5)
   +--saInformationGroup(6)
   +--monitoringInformationGroup(7)
   +--hypervisorInformationGroup(8)
```

PRILOG C - ETFBL-DCI-MIB ANALIZA METRIKA

```
# ETFBL-DCI-MIB module metrics (generated by smidump 0.4.8)

MODULE        LANGUAGE SIZE REVISION DATE
ETFBL-DCI-MIB   SMIv2   159     0    2017-08-15

# The following table shows the status distribution of various
# definitions contained in the set of loaded MIB modules.

CATEGORY          TOTAL  CURRENT  DEPRECATED  OBSOLETE
Types:                6   100.0%        0.0%      0.0%
Tables:              18   100.0%        0.0%      0.0%
Columns:            150   100.0%        0.0%      0.0%
```

```
Scalars:               3   100.0%        0.0%      0.0%
Notifications:         0     0.0%        0.0%      0.0%
Groups:                7   100.0%        0.0%      0.0%
Compliances:           1   100.0%        0.0%      0.0%
Summary:             185   100.0%        0.0%      0.0%

# The following table shows the access mode distribution of all scalar
# or column definitions contained in the set of loaded MIB modules.

CATEGORY           TOTAL  READWRITE  READONLY  NOTIFY  NOACCES
Columns:             150       0.0%     90.7%    0.0%     9.3%
Scalars:               3       0.0%    100.0%    0.0%     0.0%
Summary:             153       0.0%     90.8%    0.0%     9.2%

# The following table shows the table index kind distribution of
# table definitions contained in the set of loaded MIB modules.

CATEGORY           TOTAL    INDEX  AUGMENT  REORDER   SPARSE   EXPAND
Tables:               18   100.0%     0.0%     0.0%     0.0%     0.0%

# The following table shows the table index length distribution of
# table definitions contained in the set of loaded MIB modules.

CATEGORY    TOTAL    [1]   [2]   [3]   [4]   [5]   [6]   [7]   [8]   [9]  [10]
Tables:        18  16.7% 44.4% 33.3%  5.6%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%

# The following table shows the table length distribution of
# table definitions contained in the set of loaded MIB modules.

CATEGORY    TOTAL    [1]   [2]   [3]   [4]   [5]   [6]   [7]   [8]   [9]  [10]  [11]
[12]  [13]  [14]  [15]  [16]  [17]  [18]  [19]  [20]  [21]  [22]  [23]  [24]  [25]  [2
6]  [27]  [28]  [29]
 [30]  [31]  [32]  [33]  [34]  [35]  [36]  [37]  [38]  [39]  [40]  [41]  [42]  [43]  [
44]  [45]  [46]  [47]  [48]  [49]  [50]  [51]  [52]  [53]  [54]  [55]  [56]  [57]  [58
]  [59]  [60]  [61]
 [62]  [63]  [64]  [65]  [66]  [67]  [68]  [69]  [70]  [71]  [72]  [73]  [74]  [75]  [
76]  [77]  [78]  [79]  [80]
Tables:        18   0.0%  0.0% 11.1%  5.6%  5.6% 11.1% 22.2%  0.0%  5.6%
16.7%  0.0%  0.0%  5.6%
11.1%  5.6%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0
.0%  0.0%
 0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0
.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0
%  0.0%  0.0%  0.0%
 0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0
.0%  0.0%  0.0%  0.0%  0.0%

# The following table shows the text clause usage distribution of all
# definitions contained in the set of loaded MIB modules.

CATEGORY           TOTAL  DESCRIPTION  REFERENCE     UNIT   FORMAT
Types:                 6       100.0%       0.0%     0.0%    66.7%
Tables:               18       100.0%       0.0%     0.0%     0.0%
Columns:             150       100.0%       0.0%     6.7%     0.0%
Scalars:               3       100.0%       0.0%     0.0%     0.0%
Notifications:         0         0.0%       0.0%     0.0%     0.0%
Summary:             195       100.0%       0.0%     5.1%     2.1%

# The following table shows the average text length distribution (in
# bytes) of all definitions contained in the set of loaded MIB modules.

CATEGORY           TOTAL  DESCRIPTION  REFERENCE     UNIT   FORMAT
Types:                 6         60.8        0.0      0.0      1.5
```

```
Tables:              18         37.8        0.0      0.0      0.0
Columns:            150         27.9        0.0      6.0      0.0
Scalars:              3         25.7        0.0      0.0      0.0
Notifications:        0          0.0        0.0      0.0      0.0
Summary:            195         30.7        0.0      6.0      1.5

# The following table shows the basetype usage distribution in the
# set of loaded MIB modules.

CATEGORY   Int32 Uns32 Int64 Uns64 OctSt ObjId Enums  Bits Flo32 Flo64 Flo128
Columns:    0.0% 48.0%  0.0%  0.0% 49.3%  0.0%  2.7%  0.0%  0.0%  0.0%  0.0%
Scalars:    0.0%  0.0%  0.0%  0.0% 100.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
Summary:    0.0% 47.1%  0.0%  0.0% 50.3%  0.0%  2.6%  0.0%  0.0%  0.0%  0.0%

# The following table shows the complexity metrics of the set of loaded
# MIB modules.

CATEGORY          TOTAL      RAW   WEIGHT COMPLEXITY
Scalars (ro):         3        3
Scalars (rw):         0        0
Columns (ro):       136      272
Columns (rw):         0        0
Indexes:             18      123
Summary:            139      275

# The following table shows the distribution of the number of references
# to defined types (including base types) in the set of loaded MIB
# modules.

MODULE        TYPE                        USAGE
SNMPv2-TC     DisplayString               39.7%
ETFBL-DCI-MIB LookupIndex                 28.4%
              Unsigned32                  10.8%
ETFBL-DCI-MIB GBytes                       9.8%
ETFBL-DCI-MIB DecimalValueMillis           5.2%
ETFBL-DCI-MIB MBytes                       4.1%
SNMPv2-TC     TruthValue                   1.0%
ETFBL-DCI-MIB ServiceEndpointStatusDef     0.5%
ETFBL-DCI-MIB ServiceStatusDef             0.5%

# The following table shows the distribution of the number of references
# to externally defined types (excluding base types) in the set of loaded
# MIB modules.

MODULE    TYPE          EXT-USAGE
SNMPv2-TC DisplayString   97.1%
SNMPv2-TC TruthValue       2.9%


# The following table shows the distribution of the number of references
# to externally defined items (such as types or objects) accumulated by
# the defining MIB module in the set of loaded MIB modules.

MODULE    EXT-USAGE
SNMPv2-TC  100.0%

# The following table shows the distribution of the index complexity
# in the set of loaded MIB modules.

MODULE        TABLE                       COMPLEXITY
ETFBL-DCI-MIB serviceEndpointStatusEntry     4.0%
ETFBL-DCI-MIB ceVoQueueEntry                 3.0%
ETFBL-DCI-MIB saVoEntry                      3.0%
```

```
ETFBL-DCI-MIB serviceAREntry                3.0%
ETFBL-DCI-MIB serviceEndpointAREntry        3.0%
ETFBL-DCI-MIB serviceEndpointEntry          3.0%
ETFBL-DCI-MIB serviceStatusEntry            3.0%
ETFBL-DCI-MIB ceQueueEntry                  2.0%
ETFBL-DCI-MIB hypervisorEntry               2.0%
ETFBL-DCI-MIB hypervisorStatsEntry          2.0%
ETFBL-DCI-MIB saPoolEntry                   2.0%
ETFBL-DCI-MIB vmImageEntry                  2.0%
ETFBL-DCI-MIB vmInstanceEntry               2.0%
ETFBL-DCI-MIB vmInstanceTypeEntry           2.0%
ETFBL-DCI-MIB vmStorageEntry                2.0%
ETFBL-DCI-MIB serviceEntry                  1.0%
ETFBL-DCI-MIB siteEntry                     1.0%
ETFBL-DCI-MIB voEntry                       1.0%
```

PRILOG D - PRIMJER KONFIGURACIONOG FAJLA

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <agentX agentXServer="127.0.0.1" agentXPort="705" agentXKeepAlive="1"
       agentXReconnectTime="5000" />
    <lookupFile url="http://c01.grid.etfbl.net/bbmsam/lookup.conf"
       lookupMode="nonpersistent" lookupInterval="300"/>
    <lookupFile url="lookup2017.conf" lookupMode="readwrite" lookupInterval="300" />
    <x509 keyStore="client.p12" keyStoreType="pkcs12" keyStorePassword="nekasifra"
       trustStore="server.keystore" trustStoreType="jks"
       trustStorePassword="nekasifra"/>
    <systemInfo systemContact="badaboom@etfbl.net" systemName="DCISNMPv1"
       systemDescription="Test installation of DCISNMP server" />
    <messenger type="amqp" sendBulk="n" topic="dcisnmp" routingFilter="*.*.*"
       uri="amqp://localhost"/>
    <dataEndpoints>
        <endpoint type="argo"
           url="https://web-api-devel.argo.grnet.gr/api/v2/results/Critical/PROJECT"
           queryType="ar" apiKey="cd098725facf50c17c82216abf9d372d" periodDays="7"
           projectName="VI-SEEM" pollingInterval="300"/>
        <endpoint type="argo"
           url="https://web-api-devel.argo.grnet.gr/api/v2/results/Critical/"
           queryType="ar" apiKey="abc123456acf50c17c82216abf9d372d" periodDays="7"
           projectName="VI-SEEM" pollingInterval="300"/>
        <endpoint type="openstack" url="http://c01.os.etfbl.net:5000"
           siteName="ETFBL-CC01" countryName="Bosnia and Herzegovina"
           collectHypervisorStatistics="y" collectHypervisors="y" collectImages="y"
           collectServices="y" collectFlavors="y" collectServers="y"
           pollingInterval="300"/>
        <endpoint type="proxmox" url="https://px1.etfbl.net:8006"
           username="korisnik" password="nekasifra" siteName="ETFBL-PX01"
           countryName="Bosnia and Herzegovina" pollingInterval="60"
           collectHypervisorStatistics="y" collectHypervisors="y" collectImages="y"
           collectServices="y" collectFlavors="y" collectServers="y"
           collectStorages="y"   />
        <endpoint type="gocdb" url="https://gocdb.vi-seem.eu/gocdbpi/private/"
           pollingInterval="3600" collectSites="y" collectServiceTypes="y"
           collectServiceInstances="y"/>
        <endpoint type="ldap"
           url="ldap://bdii.ipb.ac.rs:2170/Mds-vo-name=local,o=grid"
           pollingInterval="3600"
           filter="(|(GlueCEAccessControlBaseRule=VO:vo.vi-seem.eu)
```

```
            (GlueServiceAccessControlBaseRule=VO:vo.vi-seem.eu)
            (GlueServiceAccessControlRule=vo.vi-seem.eu)
            (GlueCEAccessControlBaseRule=VOMS:/vo.vi-seem.eu/*))"
            collectCEs="y" collectSEs="y" collectSites="n"/>
        <endpoint type="nagiosjson" url="https://mon-ni4os.argo.grnet.gr/nagios"
            queryType="status" periodDays="3" projectName="VI-SEEM"
            pollingInterval="3600" username="korisnik" password="nekasifra"/>
        <endpoint type="nagiosjson" url="https://mon-ni4os.argo.grnet.gr/nagios"
            queryType="ar" periodDays="1" projectName="VI-SEEM" pollingInterval="21600"
            username="korisnik" password="nekasifra"/>
    </dataEndpoints>
    <gridData maxAgeInterval="60" maxAgeVo="86400" maxAgeSite="86400"
            maxAgeService="86400" maxAgeCeQueue="3600" maxAgeCeVo="3600"
            maxAgeSaPool="3600" maxAgeSaVo="3600" maxAgeStatus="43200" />
</config>
```

BIOGRAFIJA AUTORA

Osnovni podaci:

- Datum rođenja: 26.09.1978.
- Mjesto rođenja: Banja Luka, Republika Srpska, Bosna i Hercegovina
- Kontakt: mihajlo.savic@etf.unibl.org

Obrazovanje:

- Fakultet: Elektrotehnički fakultet Banja Luka
  Automatika i računarska tehnika
  Godina završetka studija: 2004.
  Prosječna ocjena: 8,14
  Diplomski rad: Sistem za upravljanje sadržajima ETF_WWW

- Postdiplomski studij: Elektrotehnički fakultet Banja Luka
  Računarska tehnika i informatika
  Godina završetka studija: 2010.
  Prosječna ocjena: 10,00
  Magistarski rad: Sistem za nadzor računarske grid infrastrukture posredstvom SNMP protokola

Znanje stranih jezika:

- Engleski jezik: Odlično znanje jezika (simultani prevodilac1997-2003.)
- Njemački jezik: Osnovni nivo

Odabrani projekti:

- Nacionalni: SARNET, METSTARS, METSTARS2
- EU FP-5/6/7: SEE-GRID, SEE FIRE, SEEREN2, SEE-GRID-2, SEE-GRID-SCI, HP-SEE, EGI-InSPIRE
- EU Horizon 2020: VI-SEEM, NI4OS-Europe

IZJAVE AUTORA

Izjava 1

IZJAVA O AUTORSTVU

Izjavljujem
da je doktorska disertacija

Naslov rada

SISTEM ZA NADZOR RAČUNARSKE INFRASTRUKTURE KAO SERVISA

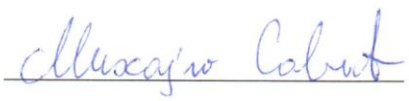
Naslov rada na engleskom jeziku

COMPUTER INFRASTRUCTURE AS A SERVICE MONITORING SYSTEM

- rezultat sopstvenog istraživačkog rada,
- da doktorska disertacija, u cjelini ili u dijelovima, nije bila predložena za dobijanje bilo koje diplome prema studijskim programima drugih visokoškolskih ustanova,
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava i koristio intelektualnu svojinu drugih lica.

U Banjoj Luci 14.05.2020.

Potpis doktoranta



Izjava 2

Izjava kojom se ovlašćuje Univerzitet u Banjoj Luci da doktorsku disertaciju učini javno dostupnom

Ovlašćujem Univerzitet u Banjoj Luci da moju doktorsku disertaciju pod naslovom

SISTEM ZA NADZOR RAČUNARSKE INFRASTRUKTURE KAO SERVISA

koja je moje autorsko djelo, učini javno dostupnom.

Doktorsku disertaciju sa svim prilozima predao sam u elektronskom formatu pogodnom za trajno arhiviranje.

Moju doktorsku disertaciju pohranjenu u digitalni repozitorijum Univerziteta u Banjoj Luci mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (Creative Commons) za koju sam se odlučio.

1. Autorstvo
2. Autorstvo – nekomercijalno
3. Autorstvo – nekomercijalno – bez prerade
4. Autorstvo – nekomercijalno – dijeliti pod istim uslovima
5. Autorstvo – bez prerade
6. **Autorstvo – dijeliti pod istim uslovima**

(Molimo da zaokružite samo jednu od šest ponuđenih licenci, kratak opis licenci dat je na poleđini lista).

U Banjoj Luci 14.05.2020.

Potpis doktoranta

Izjava 3

Izjava o identičnosti štampane i elektronske verzije doktorske disertacije

Ime i prezime autora: Mihajlo Savić

Naslov rada: SISTEM ZA NADZOR RAČUNARSKE INFRASTRUKTURE KAO SERVISA

Mentor: Prof.dr Zoran Jovanović

Izjavljujem da je štampana verzija moje doktorske disertacije identična elektronskoj verziji koju sam predao za digitalni repozitorijum Univerziteta u Banjoj Luci.

U Banjoj Luci 14.05.2020.

Potpis doktoranta