



UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U
NOVOM SADU



Dragan Stanković

RAZVOJ APLIKACIJE ZA SIMULACIJU RADA PROTOTIPA DISTRIBUTIVNOG ELEKTROENERGETSKOG SISTEMA U *SERVICE FABRIC* OKRUŽENJU

MASTER RAD

- Master akademske studije -


Novi Sad, 2020




УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:	
Идентификациони број, ИБР:	
Тип документације, ТД:	Монографска публикација
Тип записа, ТЗ:	Текстуални штампани документ/ЦД
Врста рада, ВР:	Мастер рад
Аутор, АУ:	Драган Станковић
Ментор, МН:	др Дарко Чапко, ванр. професор
Наслов рада, НР:	Развој апликације за симулацију рада прототипа дистрибутивног електроенергетског система у <i>Service Fabric</i> окружењу
Језик публикације, ЈП:	Српски (латиница)
Језик извода, ЈИ:	Српски/Енглески
Земља публикавања, ЗП:	Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2020
Издавач, ИЗ:	Ауторски репринт
Место и адреса, МА:	Факултет Техничких Наука (ФТН), Д. Обрадовића 6, 21000 Нови Сад
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	8/36/25/3/22/0/0
Научна област, НО:	Електротехничко и рачунарско инжењерство
Научна дисциплина, НД:	Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, ПО:	Дистрибутивни електроенергетски систем, <i>Service Fabric</i> , Паметна мрежа, <i>Azure</i>
УДК	
Чува се, ЧУ:	Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду представљена су два програмска решења симулације рада дистрибутивног електроенергетског система – <i>on-premise</i> и <i>Azure Service Fabric</i> . Извршена је имплементација, тестирање, анализа и поређење перформанси између два поменути решења.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: др Срђан Вукмировић, ванр. професор
	Члан: др Немања Недић, доцент
	Члан, ментор: др Дарко Чапко, ванр. професор
	Потпис ментора

		UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES 21000 NOVI SAD, Trg Dositeja Obradovića 6	
KEY WORDS DOCUMENTATION			
Accession number, ANO :			
Identification number, INO :			
Document type, DT :		Monographic publication	
Type of record, TR :		Textual material, printed/CD	
Contents code, CC :		Master thesis	
Author, AU :		Dragan Stanković	
Mentor, MN :		Darko Čapko, PhD	
Title, TI :		Development of an application for simulation of the prototype of distribution power system on Service Fabric platform	
Language of text, LT :		Serbian (latin script)	
Language of abstract, LA :		Serbian/English	
Country of publication, CP :		Serbia	
Locality of publication, LP :		Vojvodina	
Publication year, PY :		2020	
Publisher, PB :		Author reprint	
Publication place, PP :		Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad	
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)		8/36/25/3/22/0/0	
Scientific field, SF :		Electrical and computer engineering	
Scientific discipline, SD :		Applied computer science and informatics	
Subject/Key words, S/KW :		Distributed power system, Service Fabric, Smart Grid, Azure	
UC			
Holding data, HD :		Library of the Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad	
Note, N :			
Abstract, AB :		This paper presents two software solutions for simulating the distribution power system - on-premise and Azure Service Fabric. Implementation, testing, analysis and performance comparison between the two mentioned solutions was performed.	
Accepted by the Scientific Board on, ASB :			
Defended on, DE :			
Defended Board, DB :	President:	dr Srđan Vukmirović, Associate professor	
	Member:	dr Nemanja Nedić, Assistant Professor	Menthor's sign
	Member,	dr Darko Čapko, Associate professor	
	Mentor:		

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	
		Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	ПРИМЕЊЕНО СОФТВЕРСКО ИНЖЕЊЕРСТВО
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. др Драган Поповић

Студент:	Драган Станковић	Број индекса:	E5 18/2018
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	Дарко Чапко		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ – МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ ДИПЛОМСКОГ- МАСТЕР РАДА:

Развој апликације за симулацију рада прототипа дистрибутивног електроенергетског система у *Service Fabric* окружењу

ТЕКСТ ЗАДАТКА:

Анализирати на који начин функционише *on – premise* апликација за симулацију рада прототипа дистрибутивног електроенергетског система. Тестирати апликацију. Анализирати принцип рада *Service Fabric* платформе. Имплементирати софтверско решење за симулацију рада прототипа дистрибутивног електроенергетског система у *Service Fabric* окружењу. Тестирати добијено решење и упоредити добијене перформансе са перформансама *on – premise* апликације.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Досије студента; ☐ - Ментора;

Spisak korišćenih skraćenica

IT	<i>Information Technologies</i>
AMI	<i>Advanced Metering Infrastructure</i>
API	<i>Application Programming Interface</i>
NMS	<i>Network Model Service</i>
DDS	<i>Dynamic Data Service</i>
SCADA	<i>Supervisory Control And Data Acquisition</i>
PubSub	<i>Publish Subscribe</i>
URL	<i>Uniform Resource Locator</i>
XML	<i>Extensible Markup Language</i>
CIM	<i>Common Information Model</i>
RDF	<i>Resource Description Framework</i>
WCF	<i>Windows Communication Foundation</i>
GDA	<i>Generic Data Access</i>
DFS	<i>Depth First Search</i>
IP	<i>Internet Protocol</i>

Sadržaj

1. Uvod.....	1
2. Teorijske osnove	3
2.1. Elektroenergetski sistem	3
2.1.1. Proizvodnja.....	3
2.1.2. Prenos	3
2.1.3. Distribucija	4
2.1.4. Potrošnja.....	4
2.2. Tradicionalni elektroenergetski sistemi	4
2.3. Pametne mreže	5
2.4. Service Fabric.....	7
2.5. Distribuirani sistemi.....	10
3. Predmet istraživanja	14
4. Arhitektura sistema	15
4.1. Arhitektura aplikacije u lokalnom okruženju.....	15
4.2. Arhitektura aplikacije u Service Fabric okruženju	16
5. Implementacija aplikacije	18
5.1. Opis korišćenih alata i tehnologija.....	18
5.2. Tok rada aplikacije u lokalnom okruženju	19
5.3. Tok rada aplikacije u Service Fabric okruženju.....	26
6. Testiranje	31
6.1. Testiranje on-premise aplikacije.....	31
6.2. Testiranje Service Fabric aplikacije.....	31
7. Zaključak	33
8. Literatura	34
Podaci o kandidatu	36

1. Uvod

Električna energija je kroz istoriju prolazila kroz različite oblike i faze, ali je konstantno rastao njen značaj i uticaj na društvo. Sa naučnim radovima Nikole Tesle i projektovanjem sistema napajanja naizmeničnom strujom je uspostavljen temelj modernog elektroenergetskog sistema, a najveća prednost tako organizovanog sistema ogledala se u lakšem i efikasnijem prenosu električne energije na daljinu. Svoju primenu je pronašla još od momenta otkrivanja, ali je globalni efekat postigla u dvadesetom veku. Danas su svi privredni sektori, kao i transportni, komunikacioni, računarski i drugi infrastrukturni sistemi, uslovljeni kvalitetnim napajanjem električnom energijom. Ona je svojom upotrebljivošću i jednostavnom transformacijom u druge oblike energije prodrta toliko duboko u sve sfere društva, da je svakodnevni život postao nezamisliv bez nje.

Kvalitet napajanja predstavlja jedan od najvažnijih parametara u isporuci električne energije. Prekidi u napajanju imaju negativan efekat na kvalitet života, elektronske uređaje i opremu, kao i na sva preduzeća čije poslovanje na bilo koji način zavisi od napajanja električnom energijom. Kako bi se postigla odgovarajuća pouzdanost i kvalitet u isporuci električne energije, elektroenergetski sistem je podeljen na četiri osnovne celine: proizvodnja, prenos, distribucija i potrošnja. Na ovaj način došlo je do podele odgovornosti i razdvajanja procesa gde svaki podsistem brine o ispunjavanju zadataka iz svog delokruga poslova. U svakoj celini je došlo do razvoja odgovarajućih alata za vođenje tehničkih poslova kako bi se ostvarili što bolji rezultati. Tako sa ubrzanim razvojem tehnike i tehnologije, hardverskih i softverskih rešenja i telekomunikacione opreme dolazi do velikog napretka u domenu nadzora i upravljanja infrastrukturnim sistemima kao što je npr. distributivna mreža. Ukoliko se posmatra model i broj elemenata koji je karakterišu, distributivna mreža predstavlja najkompleksniji podsistem elektroenergetskog sistema [1].

Napredak i modernizacija su omogućili uvođenje koncepta pametnih mreža u elektroenergetski sistem. Pametna mreža (eng. *Smart Grid*) predstavlja automatizovanu mrežu koja nadgleda, štiti i optimizuje sve elemente sistema, od generatora pa sve do potrošačkih čvorova. Donosi brojne benefite poput korišćenja distribuiranih generatora, smanjenja zagađenja životne sredine, identifikacija grešaka u radu itd [2]. Pored navedenih prednosti omogućava i prodor računarskih programa u svet elektroenergetike.

Za razliku od tradicionalnih elektroenergetskih sistema, pametne mreže sa sobom donose softverska rešenja koja za cilj imaju kontrolu rada distributivne mreže. Aplikacije omogućuju dispečerima da na jednostavan način prate, kontrolišu i upravljaju delom mreže na koju je softver primenjen. Sve je veći obim posla koji softver obavlja umesto čoveka, pa je samim tim i sve širi spektar mogućnosti koje aplikacije nude, dok je uloga čoveka sve više usmerena ka krajnjem odlučivanju i izvršenju komandi. Praćenje potrošnje, lokalizacija kvara, različiti proračuni od značaja, energizacija i deenergizacija određenih područja su samo neki od benefita koje nude aplikacije iz oblasti elektroenergetike.

Prosperitet u svetu informacionih tehnologija (eng. *Information Technologies – IT*) je evidentan iz dana u dan i ide toliko daleko, da omogućava da se veliki infrastrukturni sistemi, kakva je i sama distributivna mreža, obuhvate i obrade u moćnim i udaljenim servisima koristeći *Cloud* tehnologije. Jedna od solucija za tu vrstu realizacije je *Service Fabric*, distribuirana sistemka platforma koja deli sistem na mikroservise

i pojednostavljuje upravljanje aplikacijama u *Cloud* okruženju [3]. Najveće prednosti *Service Fabric* – a su pouzdanost, skalabilnost i dostupnost.

U ovom radu je sprovedeno istraživanje koje je fokusirano na performanse, odnosno prednosti i mane koje nudi razvoj aplikacije „u oblaku“. Akcenat je na prednosti distribuiranih sistema koje nudi *Service Fabric* u pogledu većeg broja mašina koje izvršavaju programski kod. Teži se ka iskorišćenju dodatnog hardvera koji omogućava podelu poslova i rasterećenja i paralelizam u radu. Aplikacija koja simulira rad distributivne mreže uz onemogućavanje pojave strujnog preopterećenja je migrirana na *Service Fabric* platformu, pri čemu su izvršena testiranja gde su dobijeni rezultati upoređeni sa rezultatima dobijenim pre postavke aplikacije na *Cloud*. Sprovedena je analiza kako bi se kroz ostvarene rezultate mogla videti razlika u efikasnosti između dve platforme.

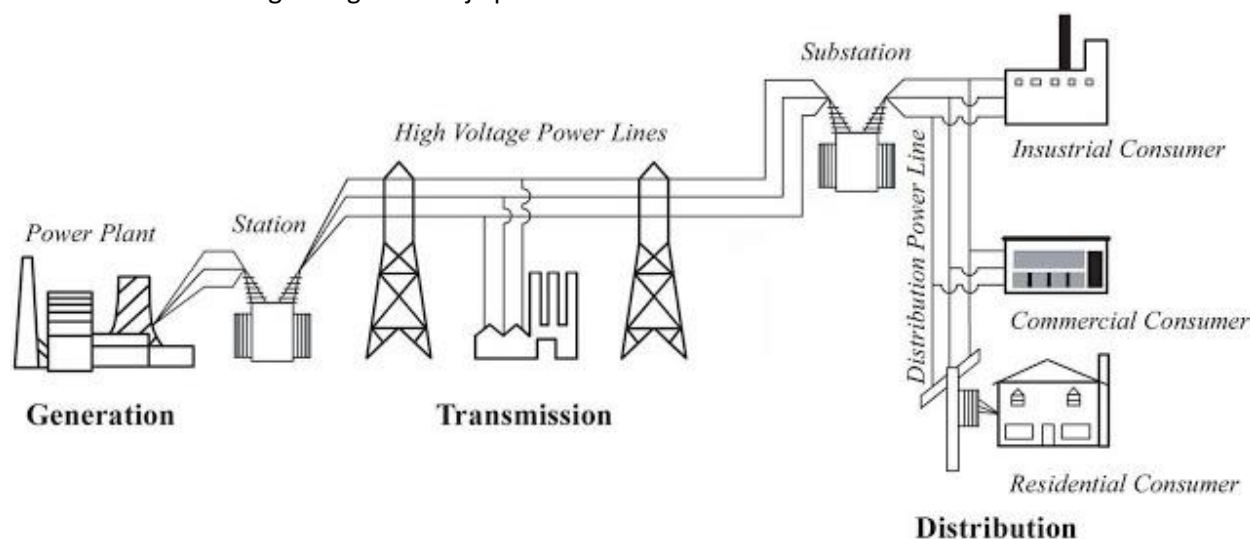
Ovaj rad je organizovan u osam poglavlja. U prvom poglavlju je dat uvod koji u kratkim crtama otkriva opšte informacije i saznanja iz oblasti elektroenergetike i opis rešavanog problema. Drugo poglavlje sadrži teorijske osnove distributivne mreže elektroenergetskog sistema i *Service Fabric* okruženja. Opisan je način na koji je organizovan i podeljen elektroenergetski sistem, karakteristike tradicionalnih i pametnih mreža, njihove prednosti i mane, organizacija i funkcionisanje *Service Fabric* platforme, kao i teorijske osnove distributivnih sistema. Treće poglavlje daje uvid u predmet istraživanja gde je objašnjen motiv rada sa osvrtom na karakteristike i očekivanja implementiranog rešenja. Arhitektura sistema se nalazi u poglavlju četiri. Prikazan je način na koje su povezane komponente koje čine sistem, kako u *on-premise* tako i u *Service Fabric* rešenju. Način implementacije, opis korišćenih alata i tehnologija i tok rada aplikacije su sadržani u poglavlju pet. Navedeni su alati i tehnologije, kao i njihova teorijska pozadina, a poslovi komponenti i tok rada programa su detaljno i postupno opisani. Testiranja pomenutih rešenja su opisana u poglavlju šest. Date su karakteristike i informacije od značaja o testnim okruženjima sa naglaskom na postignutim rezultatima. Zaključak je napisan u poglavlju sedam i u njemu su analizirani dobijeni rezultati. Korišćena literatura pri izradi rada se nalazi u poglavlju osam, a kratka biografija se nalazi na kraju rada.

2. Teorijske osnove

2.1. Elektroenergetski sistem

Elektroenergetski sistem predstavlja sistem koji obuhvata proizvodnju električne energije, prenos, distribuciju do krajnjih potrošača, kao i neposrednu potrošnju. Proizvodnja električne energije se vrši transformacijama različitih oblika energije u električnu, dok se potrošnja realizuje u obrnutom smeru, transformacijom električne energije u upotrebne oblike koje čovek može direktno da koristi. [4]

Struktura elektroenergetskog sistema je prikazana na slici 1.



Slika 1. Struktura elektroenergetskog sistema [5]

2.1.1. Proizvodnja

U podsistemu proizvodnje se vrši transformacija različitih oblika energije u električnu energiju. Pomenuti proces se najčešće odvija u hidro, termo i nuklearnim elektranama kao primarnim izvorima električne energije, uz sve veću upotrebu solarnih panela, farme vetrogeneratora i drugih alternativnih izvora električne energije. Usled nemogućnosti skladištenja električne energije, ona se mora proizvoditi u skladu sa potražnjom, tj. onda i koliko je potrebna. Zbog toga se prilikom izbora vrste elektrane za izgradnju, u zavisnosti od prirodnih resursa kao najvažnije stavke, vodi računa i o finansijskim mogućnostima, ekologiji i fleksibilnosti – mogućnost brze adaptacije na trenutnu potrošnju. Proces proizvodnje je dislociran u odnosu na potrošače radi zaštite životne sredine čoveka, kako zbog emisije štetnih gasova, tako i zbog eventualnih katastrofa.

2.1.2. Prenos

Prenosna mreža povezuje velike, geografski rasprostranjene centre za proizvodnju sa potrošačkim čvorovima, koji su smešteni blizu gradova ili industrijskih zona, na taj način održavajući elektroenergetski sistem u potpunosti povezanim i sinhronizovanim. Radi sa visokim naponima kako bi se što više smanjili gubici prilikom prenosa. Ima ključnu ulogu u održavanju ravnoteže između proizvodnje i potrošnje i predstavlja kičmu elektroenergetskog sistema. Sastoji se od dalekovoda, kablovskih vodova i

trafostanica koje vrše podizanje naponskog nivoa prilikom povezivanja proizvodnje i prenosne mreže, odnosno smanjenja prilikom povezivanja prenosne i distributivne mreže.

2.1.3. Distribucija

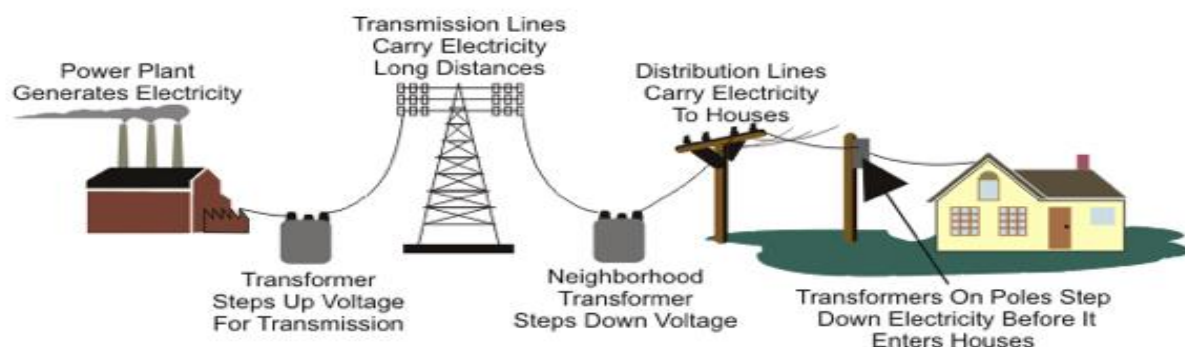
Predstavlja deo elektroenergetskog sistema koji omogućava da se električna energija preuzeta iz prenosne mreže distribuira do srednjih i malih potrošača. Naponski nivo se spušta i prilagođava krajnjim korisnicima. Distribucija električne energije zasniva se na koncepciji *izvoda* (eng. *feeder*) koji predstavljaju dovoljno jake elektroenergetske veze kojima se mogu pridružiti veći delovi jednog potrošačkog područja. Izvodi sa svojim grananjem, sve do mesta priključka individualnih potrošača čine distributivni podsistem, odnosno distributivnu mrežu [4]. Sastoji se od nadzemnih vodova, kablovskih vodova i distributivnih transformatora. Kablovski vodovi se koriste u urbanim sredinama kada se izbegava narušavanje prirodne lepote područja (npr. nacionalni parkovi) i kada je cilj povećati pouzdanost usled veće otpornosti na kvarove i ispade uzrokovane vremenskim uslovima, dok je najveća prednost nadzemnih vodova - cena (četiri do sedam puta jeftinije). Za razliku od prenosnih mreža koje imaju upetljanu strukturu, distributivne mreže su najčešće radijalne strukture, sa normalno otvorenim tačkama kako bi se povećala pouzdanost.

2.1.4. Potrošnja

U podsistemu potrošnje se vrši transformacija električne energije u energiju koju čovek može direktno da koristi. To može biti transformacija u toplotnu energiju, mehaničku, osvetljenje itd. Individualni potrošači čine podsistem potrošnje, a tu spadaju fabrike, preduzeća, domaćinstva i drugi. Skup uređaja, odnosno električnih aparata, čine individualnog potrošača i služe za transformaciju električne energije u upotrebne oblike, sa aparatom za kontrolu potrošnje koji se naziva brojilom električne energije. [4]

2.2. Tradicionalni elektroenergetski sistemi

U dvadesetom i početkom dvadeset prvog veka su primenjivani tradicionalni koncepti i tehnike da bi se obezbedilo snabdevanje potrošača električnom energijom. Proizvodnja električne energije se odvijala centralizovano, u elektranama (hidro, termo, nuklearna) koje su izgrađene neposredno u blizini primarnih izvora energije, udaljene od naseljenih mesta. Smanjenje uticaja štetnih gasova na stanovništvo i određena vrsta prevencije od mogućih katastrofa su glavni razlog njihovog distanciranja u odnosu na naselja. Nedostaci centralizovane proizvodnje su slaba pouzdanost, niska efikasnost distribucije električne energije pri niskim naponima, koncentrisano rasipanje toplote i slaba regulacija opterećenja zbog prevelike impedanse sabirnice [6]. Pored načina proizvodnje električne energije, glavna odlika tradicionalnih elektroenergetskih sistema je nizak stepen automatizacije. Detekcija kvarova je vršena isključivo na osnovu poziva potrošača o prijavi kvara i to na osnovu broja poziva koji su uzrokovani nestankom napajanja. Nisu postojali senzori, niti bilo kakvi drugi indikatori koji bi omogućili detekciju kvara. Slanje dežurnih ekipa na teren, kako bi bila izvršena lokalizacija kvara, je direktno uslovljeno brojem poziva o prijavi kvara. Ručnom manipulacijom rasklopnih uređaja je dežurna ekipa utvrđivala mesto kvara kao i izolaciju i restauraciju. Evidentiranje utrošene električne energije je takođe bio zadatak čoveka. Obavljao se ručno, na mestu potrošnje, očitavanjem vrednosti sa konvencionalnih električnih brojlara. Na slici 2 je prikazan izgled tradicionalnog elektroenergetskog sistema.



Slika 2. Tradicionalni elektroenergetski sistem [7]

2.3. Pametne mreže

Električna energija se nametnula kao najvažniji oblik energije. Nemerljiv uticaj na čovečanstvo i okolinu i potražnja za njom su doveli do toga da tradicionalni elektroenergetski sistemi postaju neodrživi u takvim uslovima. Brojni izazovi sa kojima se susreću elektroenergetski sistemi kao što su: iscrpljivanje resursa (fossilna goriva), regulatorni i javni pritisak za smanjenje zagađenja, porast potrošnje, zahtevi za pouzdanijom i kvalitetnijom energijom, zastarela infrastruktura itd. nametnuli su potražnju za boljim rešenjima [8]. Spoznaja značaja alternativnih izvora energije i benefita koji oni donose, kao i razvoj komunikacionih tehnologija, softverskih alata, nadzorno-upravljačkih sistema, mernih instrumenata i drugih naprednih sistema u oblasti elektroenergetike, doveli su do transformacije tradicionalnog elektroenergetskog sistema u pametnu mrežu.

Koncept pametnih mreža se zasniva na automatizovanom rukovođenju mrežom u realnom vremenu, uz posredstvo odgovarajuće opreme, adekvatnog nadzorno-upravljačkog sistema i softvera za upravljanje. Predstavlja mrežu koja se sastoji od računara i merne infrastrukture koja je zadužena za nadgledanje i upravljanje potrošnjom energije. Proizvođač električne energije u svojim operativnim centrima sakuplja informacije o potrošnji električne energije sa uređaja koji su raspoređeni po području koje se opslužuje. Obično se na nivou kvarta nalazi uređaj, koji periodično prikuplja podatke sa uređaja koji su instalirani u svakom domaćinstvu i potom ih šalje operativnim centrima, gde se vrši dalja obrada podataka na osnovu kojih se formira račun za naplatu potrošnje [9].

Uređaj koji izveštava o potrošnji električne energije svakog pojedinačnog domaćinstva se naziva pametno brojilo (eng. *smart meter*). Predstavlja kompjuterizovanu zamenu konvencionalnog električnog brojila koja ima procesor, memoriju i komunikacione mogućnosti. Osim što periodično izveštava o potrošnji električne energije, pruža i niz korisnih opcija kao što su praćenje potrošnje tokom dana, slanje alarma u slučaju problema, integracija sa ostalim pametnim uređajima u domaćinstvu (npr. isključivanje klima uređaja u doba najveće potrošnje i slično). Još jedan od benefita pametne mreže je mogućnost proizvodnje električne energije u sopstvenom domaćinstvu. Na taj način se potrošači direktno uključuju na tržište električne energije gde mogu da prodaju proizvedenu električnu energiju, odnosno da umanje novčanu naknadu za istu koju su utrošili. Takođe, smanjuje se proizvodni intenzitet kompanija i podstiče se proizvodnja električne energije iz alternativnih izvora. Pametna brojila predstavljaju bogat izvor informacija i stoga je pitanje poverljivosti podataka prilično osetljivo. Informacije koje se skladište u njima i distribuiraju dalje do operativnih centara daju realnu sliku o navikama i ponašanju potrošača u smislu aktivnosti i potrošnje energije, a te informacije mogu biti od velike koristi elektrodistribucionim

kompanijama, koje ih mogu upotrebiti na različite načine. Pritom, kompanije koje se bave distribucijom električne energije nisu jedine kojima bi ove informacije bile od interesa, tu spada i država, ali i kompanije iz drugih delatnosti koje mogu pronaći priliku da zarade novac pomoću pomenutih informacija. Iako u velikoj meri preovladavaju prednosti pametnih brojila u odnosu na tradicionalna električna brojila, postoje i neki nedostaci poput životnog veka brojila. Tradicionalna električna brojila imaju radni vek od 20 do 40 godina, dok pametna brojila karakteriše znatno kraće vreme rada od 5 do 7 godina. Takođe, izazivaju i troškove kompanija koje se bave proizvodnjom i isporukom električne energije, a koji se odnose na investiranje u infrastrukturu koja podržava rad pametnih brojila. [9, 10]

Problemi koji se javljaju kod konvencionalnih izvora energije kao što su smanjenja rezervi fosilnih goriva, mala energetska efikasnost i zagađenje životne sredine, doveli su do promene u proizvodnji električne energije [11]. Dolazi do sve češće proizvodnje na srednjem ili niskom naponu, pri čemu se električna energija dobija i iz tzv. alternativnih elektrana koje koriste obnovljive izvore kao što su: vetar, sunce i biomasa. Ova vrsta proizvodnje se naziva distribuirana i uglavnom je vezana za distribuirane generatore koji se najčešće priključuju na distributivnu mrežu, u blizini potrošača. Na taj način se smanjuju tokovi snaga u distributivnim vodovima, što dovodi do smanjenja snaga gubitaka i poboljšanja kvaliteta napajanja krajnjih potrošača [11]. Povećava se pouzdanost snabdevanja i nivo očuvanosti okoline i klime, a energija se proizvodi u blizini potrošnje [2]. Pametna mreža omogućava potrošačima da aktivno kontrolišu potrošnju električne energije, pružajući im informacije o njenoj ceni i potrošnji tokom dana. Tako potrošači mogu da se baziraju na isključivanje aparata u domaćinstvu koji opterećuju električnu mrežu tokom najveće dnevne potrošnje, odnosno na njihovo uključivanje tokom sati za koje je predviđena manja tarifa za naknadu [9].

Pouzdanost i pravovremene informacije su ključne u obezbeđivanju električne energije od generatora do krajnjih potrošača i funkcionisanju pametnih mreža uopšte. Posredstvom adekvatne komunikacije je obezbeđena povezanost i interoperabilnost različitih komponenti sistema. Uz pomoć različitih sistema za praćenje, dijagnostiku i zaštitu elektroenergetskog sistema se u velikoj meri može smanjiti uticaj poremećaja i nestanka struje koji izazivaju kvarovi, prirodne katastrofe itd [12]. Sigurna isporuka realnih informacija sistema i automatizacija procesa je obezbeđena uvođenjem senzora, aktuatora i pametnih brojila. Tako je obezbeđen uvid u realno stanje sistema u svakom trenutku, uz mogućnost delovanja sa udaljenog mesta, posredstvom odgovarajuće aplikacije. Tako se u slučaju kvara primenjuju algoritmi koji će brzo izvršiti lokalizaciju kvara i zona u kojoj se dogodila greška će vrlo brzo biti izolovana od ostatka mreže manipulacijom prekidača. Ostatak mreže će nastaviti sa radom sa mogućom promenom u izvoru napajanja. Povećana upotreba prekidača i uređaja za nadgledanje će povećati i troškove sistema zaštite, odnosno same infrastrukture, pa je neophodno donositi prave odluke i uklopiti povećanje pouzdanosti mreže sa ekonomičnošću [8].

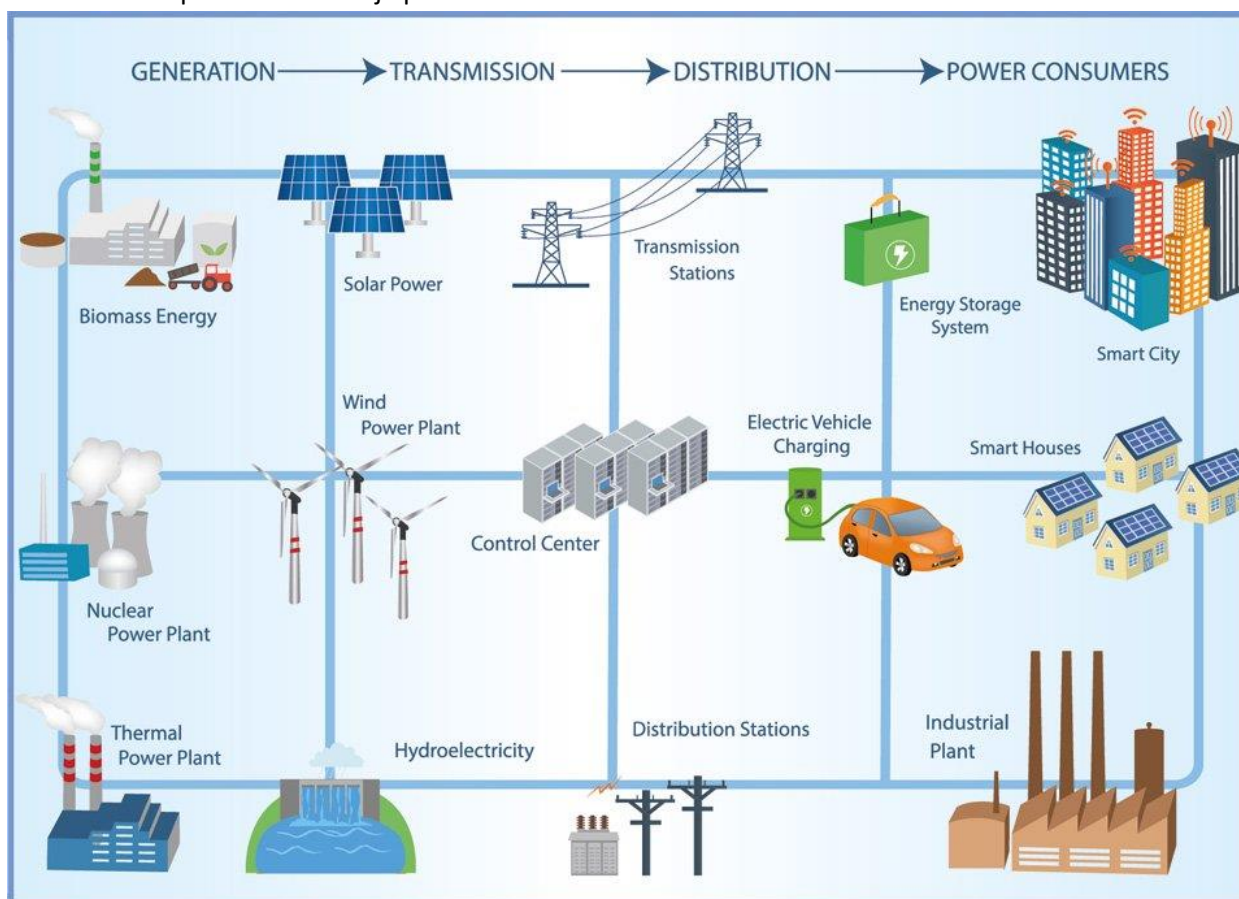
Napredna merna infrastruktura (eng. *Advanced Metering Infrastructure – AMI*) obuhvata dvosmernu komunikacionu mrežu koja omogućava prikupljanje i obradu informacija, a u koju su integrisani napredni senzori, pametna brojila, računarski hardver i softver, sistemi za nadgledanje i praćenje i sistemi za upravljanje podacima. [12]

Pametna mreža ja zamišljena tako da iskoristi sve prednosti modernih tehnologija i objedini ih tako da omogućava [13]:

- bolji uvid u stanje sistema,

- autonomne kontrolne akcije za postizanje većeg nivoa pouzdanosti povećanjem otpornosti na pojavu kvarova i prirodnih katastrofa,
- povećanje efikasnosti korišćenjem maksimalnih kapaciteta opreme,
- povećanje otpornosti na zlonamerne napade obezbeđivanjem bolje fizičke i računarske bezbednosti kako bi se očuvao integritet, autentičnost i poverljivost podataka,
- integraciju obnovljivih izvora energije u jedan elektroenergetski sistem,
- integraciju svih tipova skladištenja energije i drugih resursa kako bi se kompenzovala promenljiva proizvodnja od strane alternativnih izvora energije (količina proizvedene električne energije direktno zavisi od prisutnosti sunca, vetra...) i zahtevi potrošača,
- dvosmernu komunikaciju između potrošača i mreže kako bi krajnji korisnici mogli prilagoditi svoju potrošnju energije na osnovu individualnih mogućnosti (cena, briga o životnoj sredini...),
- bolji kvalitet usluga – manji broj prekida napajanja električnom energijom.

Infrastruktura pametne mreže je prikazana na slici 3.



Slika 3. Pametna mreža [14]

2.4. Service Fabric

Azure Service Fabric je distribuirana sistemska platforma razvijena od strane kompanije Microsoft. Omogućava stvaranje skalabilnih i pouzdanih aplikacija sastavljenih od mikroservisa koji rade na mnoštvu povezanih računara. Grupa računara, koji su tako povezani i pritom funkcionišu i ponašaju se kao jedan skladan sistem, naziva se klaster (eng. *Cluster*). Ova platforma pruža sveobuhvatne mogućnosti upravljanja aplikacijama zasnovanim na arhitekturi mikroservisa koja predstavlja grupu

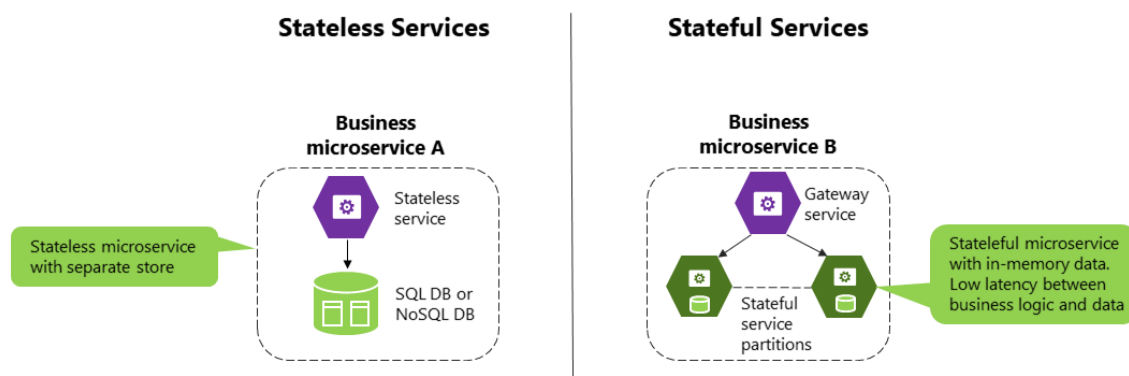
malih, nezavisno razvijenih i skalabilnih servisa koji međusobno komuniciraju preko standardnih protokola sa jasno definisanim interfejsima [3]. Karakteristike mikroservisa su [3]:

- svaki servis je nezavisan u odnosu na druge,
- može ga napraviti i održavati mali razvojni tim,
- promene u servisu ne iziskuju potrebu za *update* – om cele aplikacije,
- svaki servis je zadužen za perzistenciju svojih podataka i stanja,
- komunikacija između servisa se zasniva na interfejsima (eng. *Application Programming Interface – API*),
- servisi ne moraju da dele iste biblioteke.

Smeštaju se unutar kontejnera koji su raspoređeni širom *Service Fabric* klastera. Aplikacija može biti raspoređena i na nekoliko stotina ili hiljada mašina.

Jedna od mnogobrojnih prednosti *Service Fabric* platforme je što aplikacija može biti napisana u bilo kom programskom jeziku. Tako npr. pojedini servisi mogu biti napisani u programskom jeziku C++, dok neki drugi mogu biti napisani u C# ili Java programskom jeziku. Takođe, klasteri za *Service Fabric* se mogu kreirati u različitim okruženjima, uključujući *Azure*, *on premise*, *Windows Server* ili *Linux*. Mogu se napraviti i u drugim javnim „oblacima“. Drugim rečima, ono što se razvija na lokalnom klasteru može da funkcioniše i na klasteru u bilo kom drugom okruženju. [3]

Mikroservisi mogu biti *stateless* ili *stateful*. *Stateless* servisi ne čuvaju lokalno promenljiva stanja. Uglavnom se potrebni podaci za rad servisa čuvaju eksterno, u bazi podataka. Instance *stateless* servisa se raspoređuju po čvorovima (eng. *node* – računar ili virtuelna mašina koja predstavlja deo klastera) i u slučaju otkaza bilo koje instance ili čvora na kojoj se instanca nalazi, vrši se oporavak, tj. pravi se nova instanca u slučaju otkaza instance ili se vrši oporavak čvora koji je otkazao pri čemu se podiže nova instanca. *Stateful* servisi održavaju promenljivo stanje. Instance *stateful* servisa se nazivaju replike i njihov broj se može menjati, ali uvek je samo jedna od njih primarna. Programski kod se izvršava samo na jednom čvoru, na kojem se nalazi primarna replika pa se i on sam naziva primarni, dok su ostali čvorovi određenog servisa pasivni i na njima su pokrenute sekundarne replike, u zavisnosti koliko ih ima. U toku rada, primarni čvor konstantno replicira podatke (kopira podatke na druge lokacije kako bi podaci bili poravnati) na sekundarne čvorove i na taj način održava konzistentno stanje sistema. Odnosno, u slučaju otkaza primarnog čvora, jedan od sekundarnih preuzima posao i nastavlja njegovo izvršavanje. Takođe, vrši se oporavak čvora koji je otkazao, pri čemu se on formira kao sekundarni čvor.



Slika 4. Slikovit prikaz mikroservisa [15]

Particionisanje predstavlja jedan od osnovnih obrazaca skalabilnosti. Može se predstaviti kao koncept podele stanja (podataka) i računanja na manje dostupne jedinice kako bi se unapredile performanse. Da bi se particije određenog servisa razlikovale koristi se ključ particije (eng. *partition key*), koji se formira na osnovu odabrane jedne particione šeme [3]:

- *Ranged partition*
- *Named partition*
- *Singleton partition*

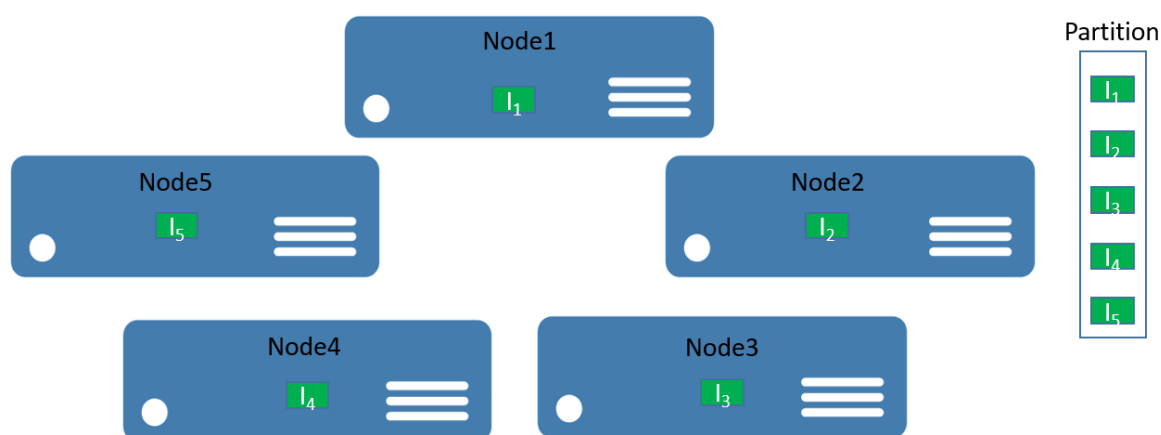
Ranged partition – particije se označavaju celobrojnim vrednostima. Jedna particija može imati ključ koji poseduje više celobrojnih vrednosti, odnosno opseg celobrojnih vrednosti. Tako npr. ukoliko su prisutne dve particije servisa, a opseg vrednosti definisan sa minimalnom vrednošću ključa (eng. *Low key*) 0, a maksimalnom (eng. *High key*) 10, onda će prva particija imati ključ opsega 0-4, a druga 5-10.

Named partition - particije se označavaju nizom karaktera. Svakoj particiji se eksplicitno kroz konfiguracioni fajl dodeljuje ime. Npr. particijama se redom mogu dodeljivati ključevi: „one“, „two“, „three“ itd.

Singleton partition - postoji samo jedna particija i nije potrebno definisati ključ.

Kod *stateless* servisa se particija može posmatrati kao logična jedinica koja sadrži jednu ili više instanci servisa. Particionisanje *stateless* servisa je veoma retka pojava jer se skalabilnost i raspoloživost mogu postići jednostavnim dodavanjem većeg broja instanci. Uglavnom se koristi jedna particija *stateless* servisa – *singleton partition*. Jedini slučaj u kome se opravdava korišćenje particija kod *stateless* servisa je kada je potrebno specijalno rutiranje, da se bi se pogodila tačno tražena particija [3].

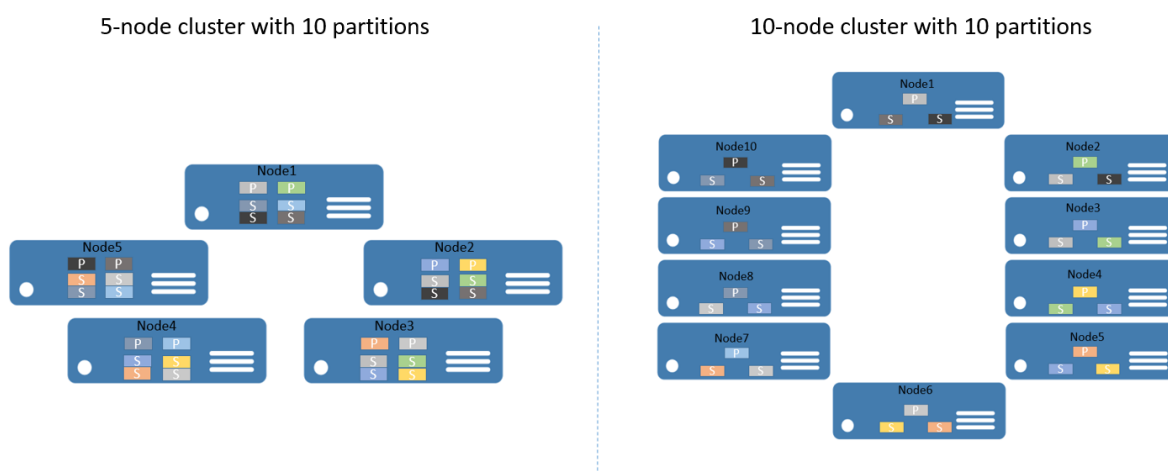
Five instances of a stateless service across a cluster



Slika 5. *Stateless servis sa 5 instanci u klasteru i jednom particijom* [3]

Kod *stateful* servisa se particija posmatra kao set replika, s tim što *Service Fabric* postavlja particije na različite čvorove [3]. Na taj način se osigurava efikasna upotreba hardverskih resursa. Kako rastu potrebe aplikacije, tako rastu i particije, a *Service Fabric* vrši rebalans particija po čvorovima [3]. Na slici 6 je prikazan primer raspoređivanja replika *stateful* servisa po čvorovima klastera. Prvobitno je prikazan raspored replika servisa koji ima 10 particija i 3 replike u klasteru od 5 čvorova. Replike su raspoređene tako da svaki čvor poseduje po 2 primarne replike i 4 sekundarne. Ukoliko se klaster skalira i poveća broj

čvorova na 10, *Service Fabric* će izvršiti rebalans replika po čvorovima. U drugom delu slike 6 se vidi raspoređivanje replika po klasteru od 10 čvorova, gde svaki čvor ima tačno jednu primarnu repliku i dve sekundarne.



Slika 6. Skaliranje stateful servisa i raspoređivanje particija [3]

Postoji i horizontalno skaliranje klastera – menjanje broja čvorova ili vertikalno – menjanje hardverskih karakteristika čvorova. Skaliranje se vrši u zavisnosti od zahteva aplikacije.

2.5. Distribuirani sistemi

U prvoj etapi razvoja računarskih sistema, u periodu od četrdesetih do osamdesetih godina prošlog veka, prva asocijacija na računare su bila visoka cena koštanja i njihova veličina. Zauzimali su po čitave prostorije, a koštali su i po nekoliko desetina hiljada dolara. Organizacije koje su posedovale veći broj računara nisu imale mogućnost njihovog međusobnog povezivanja, pa su oni funkcionisali potpuno nezavisno jedan od drugog. Ovaj trend je počeo da se menja sredinom osamdesetih godina 20. veka. Prva prekretnica je bila razvoj moćnih mikroprocesora, čime su dotadašnji 8-bitni procesori zamenjeni prvo 16-bitnim, pa 32-bitnim, a danas su najviše u upotrebi 64-bitni procesori. Od računara koji su izvršavali jednu instrukciju po sekundi i pritom koštali 10 miliona dolara, došli smo do mašina koje izvršavaju milion instrukcija po sekundi uz cenu koštanja od oko 1000 dolara. Druga prekretnica je bila razvoj računarskih mreža velike brzine. Na taj način je omogućena povezanost velikog broja računara i razmena informacija između njih, bez obzira na geografski položaj, čime su ispunjeni preduslovi za stvaranje distribuiranih sistema. [16]

Distribuirani sistem predstavlja skup nezavisnih računara koje krajnji korisnik vidi kao jedan koherentan sistem [16]. Računari su nezavisni, međusobno povezani računarskom mrežom i rade zajedno u cilju izvršavanja zajedničkog zadatka, pri čemu je sva kompleksnost skrivena od korisnika.

Razlozi za razvoj distribuiranih sistema su [16]:

1. Jednostavniji pristup resursima.
2. Transparentnost.
3. Otvorenost.
4. Skalabilnost.

Jednostavan pristup resursima - Glavni zadatak distribuiranih sistema je da omoguće jednostavniji pristup resursima, odnosno da obezbede pristup udaljenim resursima, kao i njihovu razmenu na efikasan i kontrolisan način. U resurse se ubrajaju računari, štampači, podaci, fajlovi, *web* stranice itd. Deljenje informacija donosi mnogo benefita. Tako npr. jedan štampač u maloj kancelariji, koji deli nekoliko ljudi, predstavlja ekonomski isplativo rešenje umesto da svaki radnik ima zaseban štampač. Isto tako, Internet ilustruje značaj razmene informacije gde se fajlovi, mejlovi, audio i video zapisi i drugi vidovi datoteka mogu na jednostavan način podeliti sa ostatkom sveta.

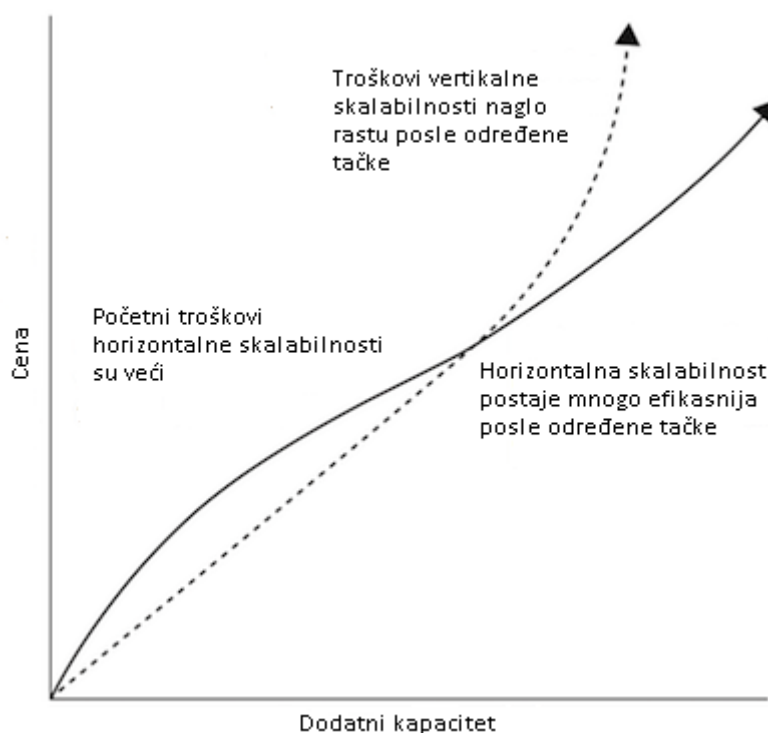
Transparentnost – Važan aspekt u distribuiranim sistemima je transparentnost, a odnosi se na skrivanje činjenice da su procesi i resursi fizički podeljeni na više distribuiranih mašina, gde se korisniku aplikacija predstavlja kao jedan skladen sistem. Tipovi transparentnosti koje se mogu javiti u distribuiranom sistemu su prikazani u tabeli 1.

Tabela 1. *Različite forme transparentnosti u distribuiranom sistemu [16]*

Transparentnost	Opis
Pristup	Sakriva razlike u reprezentaciji podataka i načinu na koji se pristupa resursu
Lokacija	Sakriva gde je resurs lociran
Migracija	Sakriva da se resurs može premestiti na drugu lokaciju
Realokacija	Sakriva da se resurs može premestiti na drugu lokaciju dok je u upotrebi
Replikacija	Sakriva da je resurs repliciran
Paralelnost	Sakriva da resurs može biti deljen između konkurentnih korisnika
Otkaz	Sakriva otkaz i oporavak resursa

Otvorenost – Distribuirani sistem izlaže svoje servise po unapred definisanim standardima i pravilima koji opisuju sintaksu i semantiku servisa. U distribuiranim sistemima, servisi su izloženi preko interfejsa. Na ovaj način se omogućuje proširivost i modifikovanje sistema na različite načine [17].

Skalabilnost – Ovaj zahtev se odnosi na proširenje i smanjenje distribuiranog sistema u vidu dodavanja i smanjivanja broja korisnika i resursa i naziva se horizontalno skaliranje. Vertikalno skaliranje je prisutno kod sistema sa jednim serverom gde se vrši unapređenje postojećeg hardvera (dodavanje RAM memorije, procesora, itd.) kako bi se povećala produktivnost. Na slici 7. je prikazano poređenje horizontalnog i vertikalnog skaliranja, odnosno ekonomska isplativost u zavisnosti od proširenja.



Slika 7. Razlika troškova horizontalne i vertikalne skalabilnosti [18]

Sa slike 7 se može primetiti da je u početnoj fazi, kada potrebe za kapacitetom nisu velike, vertikalna skalabilnost isplativija. Ali, kako raste potreba za dodatnim kapacitetom, razlika se smanjuje sve do tačke kada troškovi vertikalne skalabilnosti naglo rastu i premašuju troškove horizontalne skalabilnosti koji rastu ujednačeno. Drugim rečima, kada je potrebno malo proširenje isplativija je vertikalna skalabilnost, a u slučaju potrebe za velikim proširenjem kapaciteta isplativija je horizontalna skalabilnost. [18]

Prednosti distribuiranih sistema su višestruke [17, 18, 19, 20]:

- Svi čvorovi distribuiranih sistema su povezani tako da na jednostavan način mogu međusobno da razmenjuju informacije.
- Otkaz jednog čvora ne dovodi do otkaza celokupnog distribuiranog sistema. Ostali čvorovi i dalje mogu međusobno da komuniciraju.
- Resursi mogu biti deljeni između više čvorova npr. štampač.
- Jednostavno dodavanje novih čvorova u distribuirani sistem (horizontalno skaliranje).
- Distribuirani sistemi su daleko efikasniji zbog raspodele posla između čvorova (performanse).
- Izvršava se replikacija podataka čime se povećava pouzdanost sistema.
- Postiže se heterogenost sistema (različiti korisnici, putem različitih uređaja, operativnih sistema, mreža itd. mogu da pristupe resursima).

Izazovi sa kojim se susreću distribuirani sistemi su [17, 18, 19, 20]:

- Obezbeđenje adekvatne zaštite u distribuiranim sistemima, kako čvorova, tako i konekcija.
- Pojedine poruke i podaci se mogu izgubiti u mreži prilikom transporta na druge čvorove.
- Baza podataka distribuiranog sistema je komplikovanija i teža za rukovanje u odnosu na bazu podataka centralizovanog sistema.

- Može doći do preopterećenja u mreži ukoliko više čvorova distribuiranog sistema pokušaju da pristupe istom resursu u istom trenutku.
- Distribuirani sistem mora da odredi koji poslovi će se pokrenuti, kada i gde, prilikom čega može doći do nepredviđenih rezultata usled neadekvatnog iskorišćenja hardvera.
- Prikupljanje, procesuiranje i praćenje upotrebe hardvera za velike klastere može biti značajan izazov.
- Mogu se javiti greške u radu i neočekivani ishodi u slučaju otkaza nekog dela sistema, pa rukovanje otkazima može biti prilično težak posao zato što je greška uzrokovana delom sistema, dok drugi deo sistema radi u skladu sa očekivanjima.

3. Predmet istraživanja

Razvijena aplikacija treba da predstavlja programsko rešenje koje služi za nadzor, kontrolu i upravljanje distributivnim delom mreže. Nadzor se odnosi na uvid u realno stanje sistema, stanje prekidačke opreme, trenutne pojedinačne potrošnje svakog korisnika električne energije kao i celokupne mreže, dok se upravljanje odnosi na komandovanje tj. menjanje statusa prekidača. Mreža ili deo mreže, nad kojim aplikacija vrši nadgledanje i upravljanje, zaštićen je od pojave strujnog preopterećenja koje može nastati usled velike potražnje za električnom energijom. Vodovi imaju specificiranu maksimalnu jačinu električne struje koju mogu da provode i ukoliko jačina električne struje bude veća od definisanog limita, doći će do oštećenja opreme, tj. do topljenja provodnika. Stoga se vrši zaštita fizičke infrastrukture koja je sprovedena smanjenjem potrošnje u mreži, preventivnim isključivanjem potrošača na određeni period, pri čemu se vodi računa da potrošači istog prioriteta budu približno isto vreme bez električne energije. Aplikacija može da obrađuje više različitih distributivnih mreža.

Predmet istraživanja su efikasnost i performanse aplikacije u lokalnom i u *Service Fabric* okruženju čiji je cilj objašnjenje da li prelazak na *Cloud* bazirano rešenje nudi bolje performanse u odnosu na *on-premise* rešenje. Logika migracije aplikacije na *Cloud* se nalazi u konceptu rada distribuiranih sistema. Odnosno, *Service Fabric* klaster predstavlja distribuiran sistem i kao takav nudi niz prednosti koje su objašnjene u poglavljima 2.3. i 2.5. Prvenstveno se teži iskorišćenju prednosti koje nudi distribuiran sistem (*Service Fabric*) u vidu hardverskih resursa, raspodele posla po čvorovima i paralelizma u radu. Kao rezultat prelaska na *Cloud* rešenje očekuje se napredak u radu aplikacije koja bi mogla istovremeno da opslužuje više delova mreže nego što je to slučaj kod *on-premise* aplikacije. Kada je aplikacija pokrenuta lokalno svi njeni servisi su pokrenuti na jednom računaru i koriste njegove hardverske resurse. U *Service Fabric* okruženju dolazi do podele poslova, odnosno određeni servisi rade na virtuelnim mašinama koje pripadaju geografski udaljenom klasteru, sa različitim konfiguracijama hardvera u odnosu na lokalni računar, dok se određeni delovi programskog koda izvršavaju na lokalnoj mašini. Migracija i raspodela servisa na virtuelne mašine *Service Fabric*-a predstavljaju način efikasnog iskorišćenja hardvera i prednosti koje nudi distribuirani sistem.

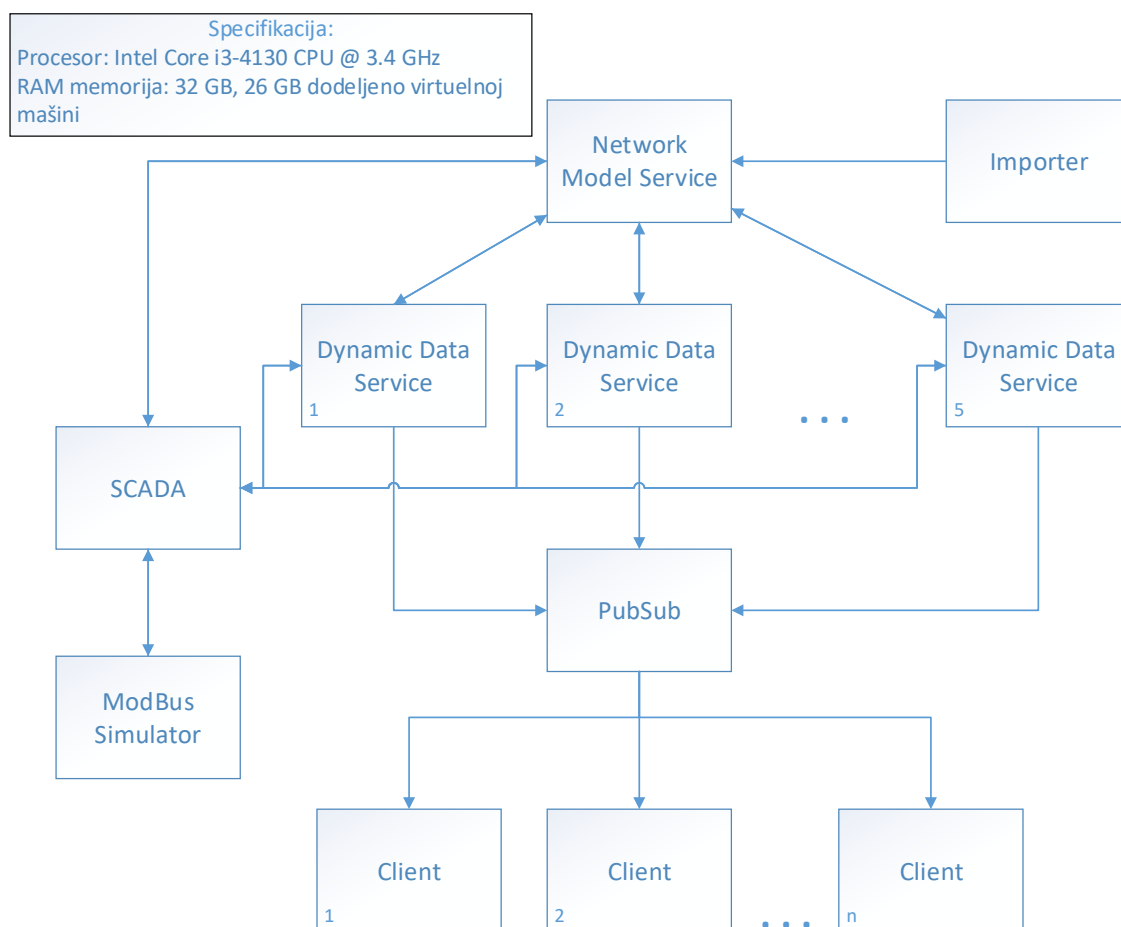
4. Arhitektura sistema

Softversko rešenje je realizovano kroz dve aplikacije različite arhitekture. Arhitektura sistema *on-premise* aplikacije se razlikuje od aplikacije iste funkcionalnosti koja je pokrenuta u *Service Fabric* okruženju zbog drugačijeg testnog okruženja i zbog pravilnog rutiranja pristiglih poruka unutar klastera.

4.1. Arhitektura aplikacije u lokalnom okruženju

Sistem je podeljen u nekoliko komponenti. Svaka od njih ima svoja zaduženja i zadatke koje ispunjava nezavisno od ostalih komponenti sistema. Sve komponente su podignute lokalno, na istom računaru. Arhitektura aplikacije i smer komunikacije između komponenti je prikazan na slici 8, odakle se vidi da je sistem izgrađen od sledećih elemenata:

- *Importer*
- *Network Model Service – NMS*
- *Dynamic Data Service – DDS*
- *Supervisory Control And Data Acquisition – SCADA*
- *PublishSubscribe - PubSub*
- *Modbus simulator*
- *Client*



Slika 8. Arhitektura *on – premise* aplikacije

Importer – učitava CIM XML dokument u kom se nalaze svi elementi elektroenergetske mreže i informacije o njima. Na osnovu pomenutog dokumenta pakuje podatke u pogodan format i šalje ih *Network Model Service*-u.

NMS – servis u kom se čuva statički model mreže. Prima podatke o elektroenergetskoj mreži od *Importer*-a na osnovu kojih kreira objekte. DDS servisu i SCADA sistemu šalje sve jedinstvene identifikatore elemenata u mreži. Ukoliko je pokrenuto više DDS instanci u sistemu, podaci se prema njima šalju po *round-robin* principu. Servisi kojima su potrebni podaci o elementima mreže obraćaju se ovoj komponenti.

DDS – čuva se dinamika sistema. Prima listu identifikatora od NMS-a. Dobijeni podaci omogućavaju servisu kreiranje topologije mreže na nivou *korena* (skup elemenata mreže napajanih sa istog izvora) uz dobavljanje potrebnih informacija sa NMS-a. Simulira trenutne vrednosti potrošnje potrošača u mreži. Vrš proračun potrošnje mreže na nivou *korena* nakon promena vrednosti elemenata pristiglih sa SCADA sistema i u toku simulacionog perioda radi preventivnog detektovanja strujnog preopterećenja. Realizuje kružna isključenja, odnosno generiše i šalje komande SCADA sistemu. Može biti pokrenuto više instanci DDS servisa gde svaka nezavisno obavlja svoje zadatke.

SCADA – komunicira sa simulatorom uz pomoć *Modbus* protokola. Prima listu identifikatora od NMS-a. Mapira digitalne i analogne ulaze/izlaze na simulator. Izvršava komande primljene od DDS komponente menjajući digitalne ili analogne vrednosti na simulatoru. Periodično proverava da li je došlo do promena vrednosti na simulatoru i promene šalje odgovarajućem DDS servisu.

PubSub – mehanizam koji obezbeđuje komunikaciju sa klijentom. Posredstvom ove komponente DDS šalje podatke klijentu o trenutnom stanju mreže.

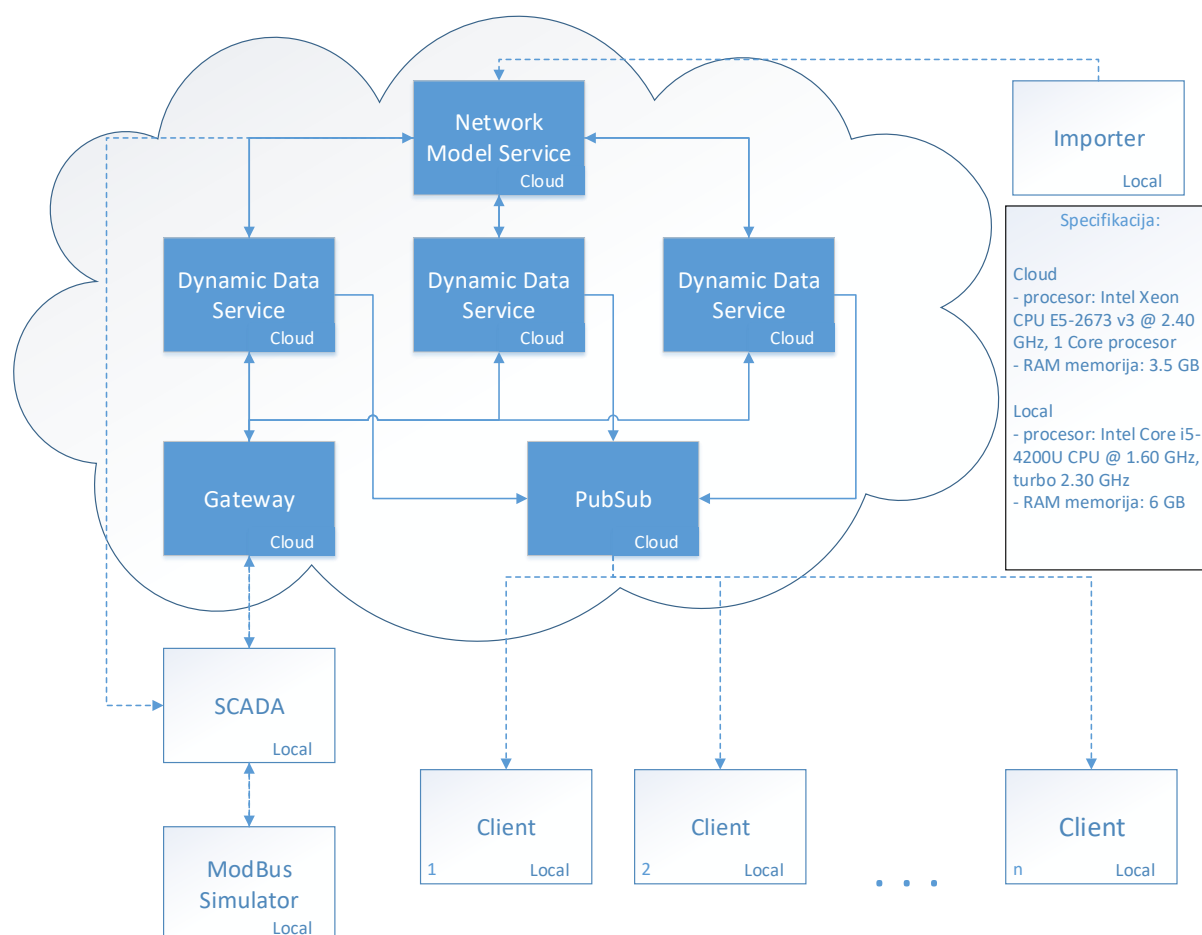
Modbus simulator – alat koji simulira stvarno stanje sistema. Sadrži analogne i digitalne ulaze/izlaze sa kojih se čita/upisuje stanje. Predstavlja stanje sistema, odnosno stanje prekidačke opreme.

Client – prati trenutno stanje mreže.

4.2. Arhitektura aplikacije u *Service Fabric* okruženju

Cloud arhitektura sistema je za nijansu drugačija u odnosu na lokalnu i prikazana je na slici 9. Pored komponenti koje su pomenute u prethodnom poglavlju, dodata je još komponenta:

- *Gateway*



Slika 9. Arhitektura Service Fabric aplikacije

Gateway – posrednik u komunikaciji između SCADA i DDS komponente. DDS je pokrenut na udaljenoj virtualnoj mašini, dok je SCADA pokrenuta na lokalnom računaru, a **Gateway** vrši rutiranje dobijenih poruka od strane SCADA sistema ka odgovarajućoj instanci DDS sistema.

Servisi koji se pokreću u *Service Fabric* okruženju su: **Gateway**, **NMS**, **DDS** i **PubSub**. **Gateway** predstavlja *stateless* servis, dok su ostali servisi pokrenuti kao *stateful*. Ostale komponente sistema se pokreću na lokalnom računaru.

5. Implementacija aplikacije

Kako bi se lakše i bolje razumeo tok rada aplikacije, kao i način na koji je implementirana, neke pojmove je potrebno dodatno objasniti.

Komponenta *Importer* koristi CIM XML fajl u kome je u XML (*Extensible Markup Language*) formatu sadržan CIM (*Common Information Model*) model koji je opisan pomoću RDF-a (*Resource Description Framework*). CIM predstavlja standard koji je razvijen od strane elektroprivrede kako bi omogućio softverskim aplikacijama razmenu informacija o konfiguraciji i statusu električne mreže [21]. Podaci se skladište u RDF formatu koji je osnova za obradu metapodataka i omogućava interoperabilnost između aplikacija koje razmenjuju mašinski čitljive informacije. Cilj RDF-a je definisanje domenski neutralnog mehanizma koji je pogodan za opisivanje informacija u bilo kom domenu. RDF opisuje resurse u obliku tripleta subjekat – predikat – objekat gde subjekat označava resurs, predikat neku osobinu resursa i označava odnos između subjekta i objekta, a objekat može biti neki drugi resurs ili literal. [22]

5.1. Opis korišćenih alata i tehnologija

Za razvoj softverskog rešenja opisanog u ovom radu korišćeno je *Microsoft Visual Studio 2015* razvojno okruženje. Programski kod je napisan u *C#* programskom jeziku, oslanjajući se na *.NET framework*, *WCF* (*Windows Communication Foundation*) i *Service Fabric*.

.NET framework je programski okvir koji služi za razvoj i izvršavanje aplikacija na *Windows* operativnim sistemima. Sačinjen je od alata, programskih jezika i biblioteka koji omogućavaju stvaranje veliki broj različitih tipova aplikacija. Deo je *.NET* platforme koja predstavlja kolekciju tehnologija za razvoj aplikacija na različitim operativnim sistemima kao što su *Linux*, *macOS*, *Windows*, *iOS*, *Android* i drugi. [23]

C# je objektno orijentisan programski jezik kompanije *Microsoft* koji kombinuje računarsku moć *C++* sa lakoćom programiranja koju poseduje *Visual Basic*. Pripada mlađoj generaciji programskih jezika, a dizajniran je za rad sa *Microsoft .NET* platformom. Najbliži je programskom jeziku *Java*. [24]

WCF predstavlja *framework* koji služi za razvoj servisno orijentisanih aplikacija. Predstavlja model za razmenu podataka koji omogućava komunikaciju lokalno ili putem mreže. Stvara komunikacioni kanal između dve tačke koje mogu biti geografske dislocirane. Poruke mogu biti sačinjene od jednog karaktera, a mogu biti i znatno kompleksnije kao što je *stream* binarnih podataka. [25]

Da bi komunikacija između dve strane bila moguća, neophodno je definisati krajnju tačku (eng. *endpoint*) koja se sastoji od 3 elementa: adresa (eng. *address*), povezivanje ili način povezivanja (eng. *binding*) i ugovor ili interfejs (eng. *contract*). Adresa predstavlja URL (*Uniform Resource Locator*) na osnovu kog se pronalazi krajnja tačka. Povezivanje definiše protokol koji se koristi u komunikaciji, a interfejs sadrži funkcije koje se nude klijentu na korišćenje. Komunikacija je moguća tek nakon što se klijent poveže sa krajnjom tačkom.

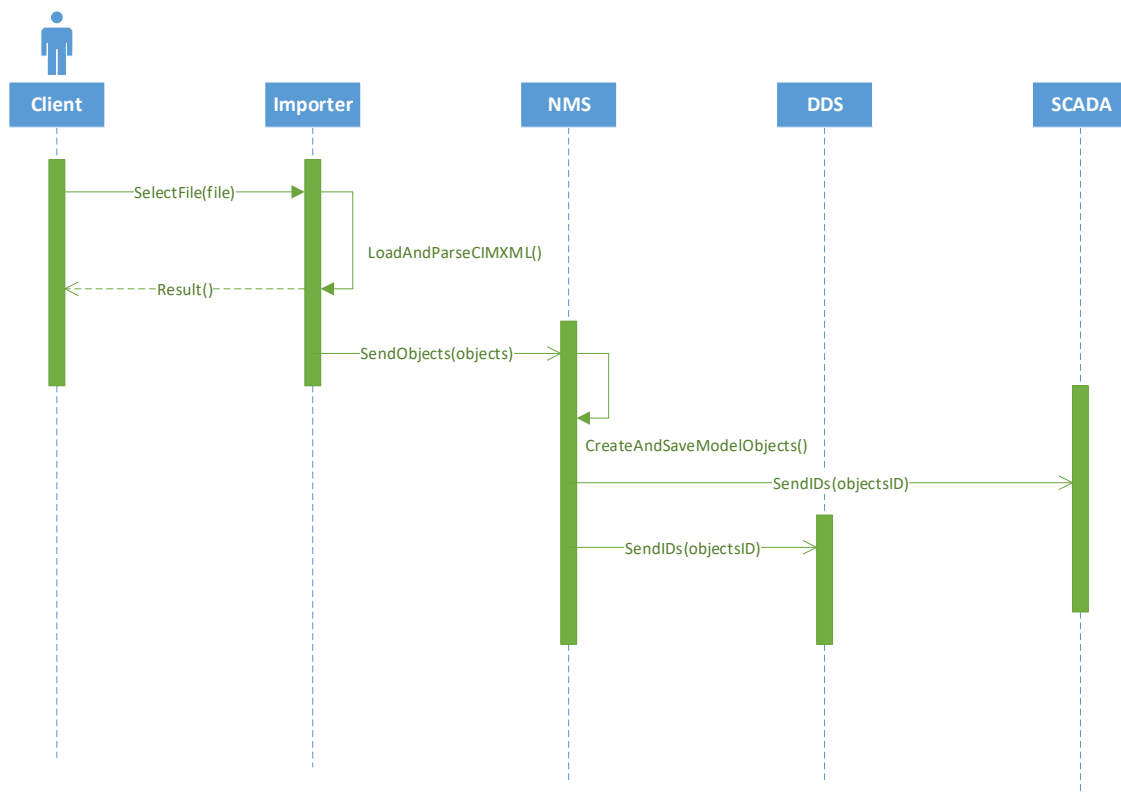
Postoji nekoliko šablona za razmenu podataka koje pruža *WCF*:

- *Request/Response* – klijent šalje zahtev ka serveru i čeka na odgovor. Najčešće se koristi.
- *One way* – jedna strana šalje podatke i ne očekuje odgovor. Tipičan šablon kod asihrone komunikacije.
- *Duplex* – dvosmerna komunikacija. Krajnje tačke mogu da šalju podatke jedna drugoj.

5.2. Tok rada aplikacije u lokalnom okruženju

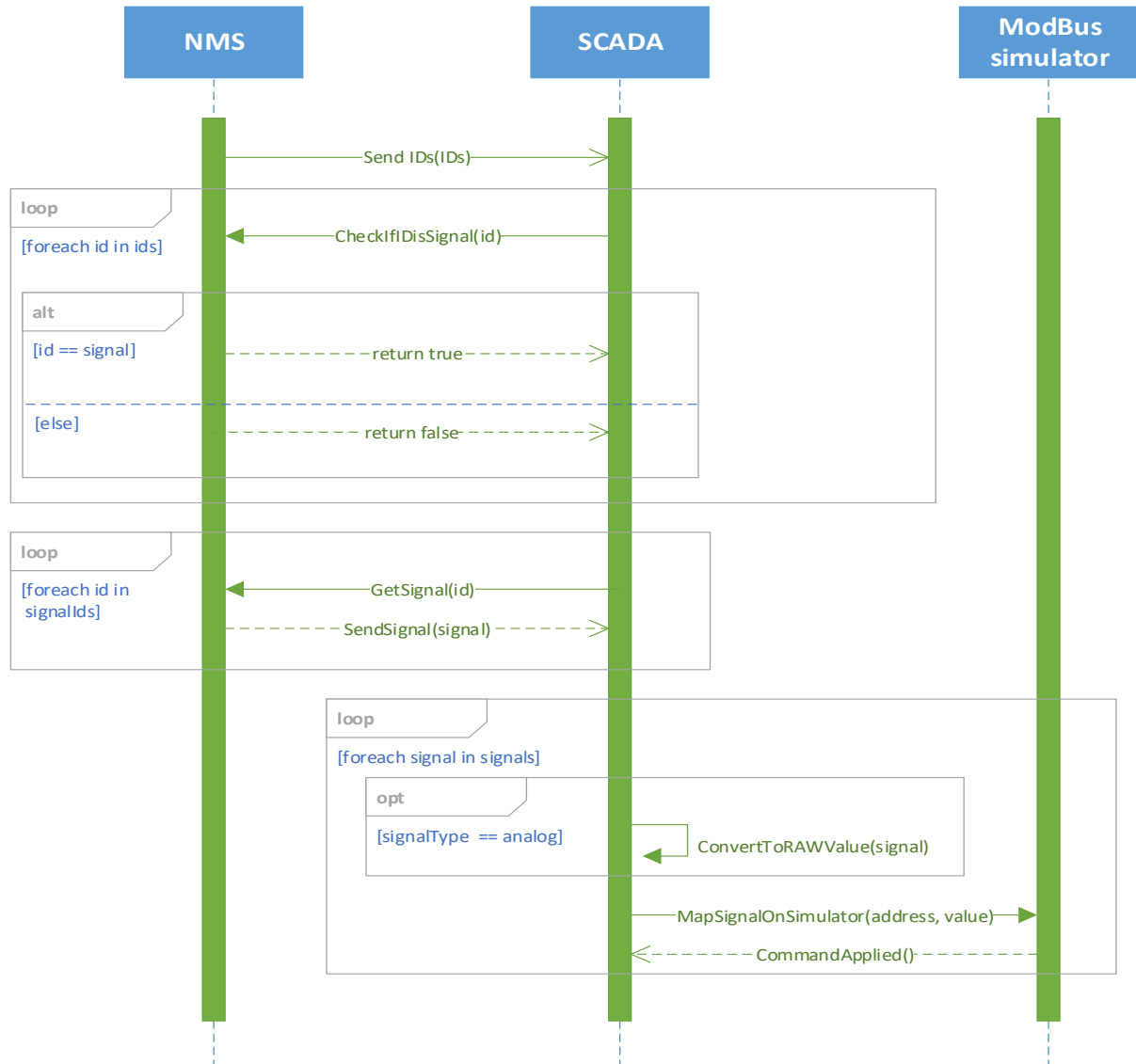
Da bi aplikacija mogla da krene sa radom, neophodno joj je dostaviti podatke koji opisuju mrežu ili deo mreže nad kojom će se programski kod izvršavati. Podaci se učitavaju pomoću CIM XML fajla za čiju obradu je zadužena komponenta *Importer* (slika 10). Kako se u učitanoj fajlu nalaze svi elementi elektroenergetske mreže u XML formatu, *Importer* prvo vrši parsiranje fajla, a potom na osnovu dobijenih podataka, za svaki element učitane mreže pravi objekat sa svim pripadajućim atributima. Ostatku sistema dostavlja podatke o mreži, tako što objekte pakuje u određene strukture i šalje ih NMS-u koji je zadužen za dalju obradu i distribuciju.

Glavni zadatak NMS-a je da čuva statički model mreže koji se dobija tako što NMS raspakuje dobijene strukture od *Importer*-a i pravi instance konkretnih klasa koje reprezentuju elemente elektroenergetske mreže. Šalje sve jedinstvene globalne identifikatore napravljenih objekata SCADA komponenti i odgovarajućoj instanci DDS-a čiji odabir se vrši po *round robin* principu. Podaci se šalju u formatu jedinstvenih globalnih identifikatora zbog rasterećenja komunikacione linije i zato što sve zainteresovane strane mogu dobiti podatke o nekom elementu slanjem jednostavnih GDA (*Generic Data Access*) upita NMS-u. Tok rada aplikacije od momenta učitavanja CIM XML dokumenta do slanja identifikatora ostalim komponentama je prikazan na slici 10.



Slika 10. Proces pravljenja i prosleđivanja objekata elektroenergetske mreže

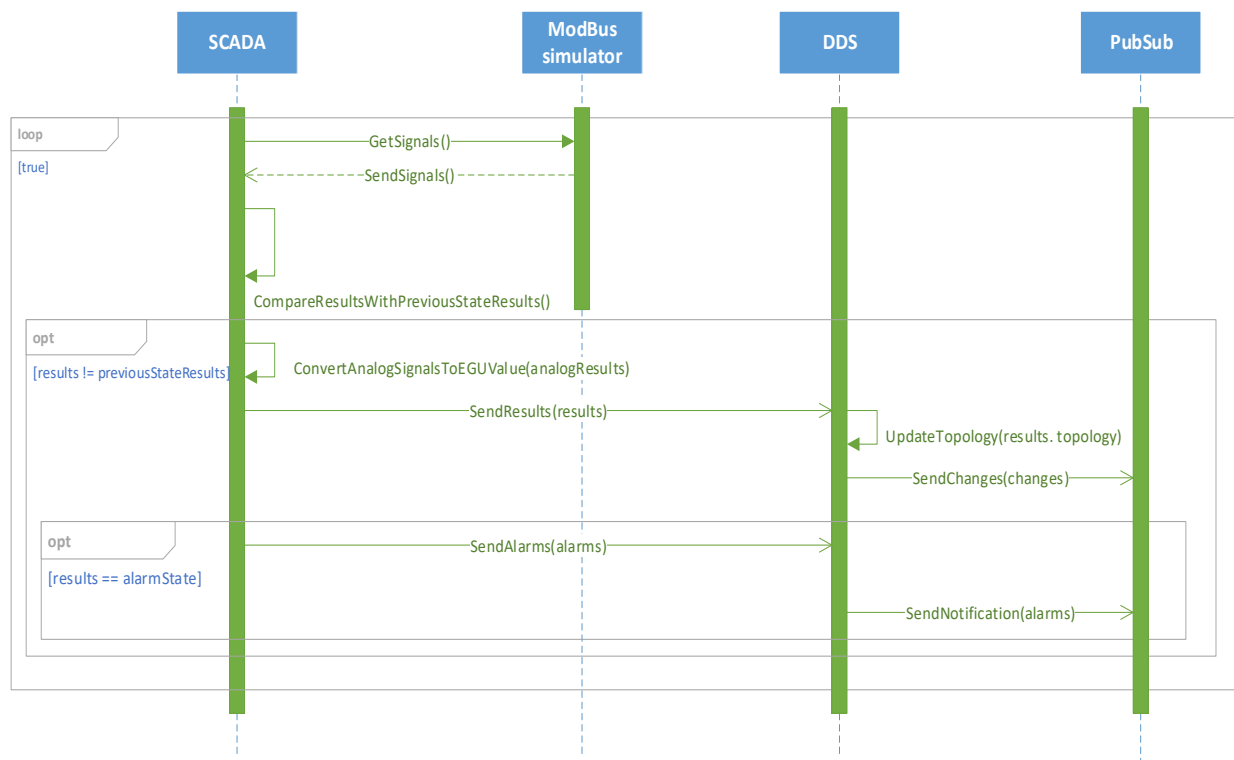
SCADA sistem uspostavlja dvosmernu komunikaciju sa NMS-om i kreira *endpoint* koji je namenjen za komunikaciju sa DDS-om. Jedan od zadataka SCADA sistema je mapiranje signala na simulator koji se izvršava po prijemu jedinstvenih globalnih identifikatora elemenata sistema. Proces mapiranja signala na simulator je prikazan na slici 11, a obuhvata filtraciju analognih i digitalnih signala, dobavljanje njihovih vrednosti sa NMS-a i slanje na *Modbus* simulator.



Slika 11. Proces mapiranja signala na simulator

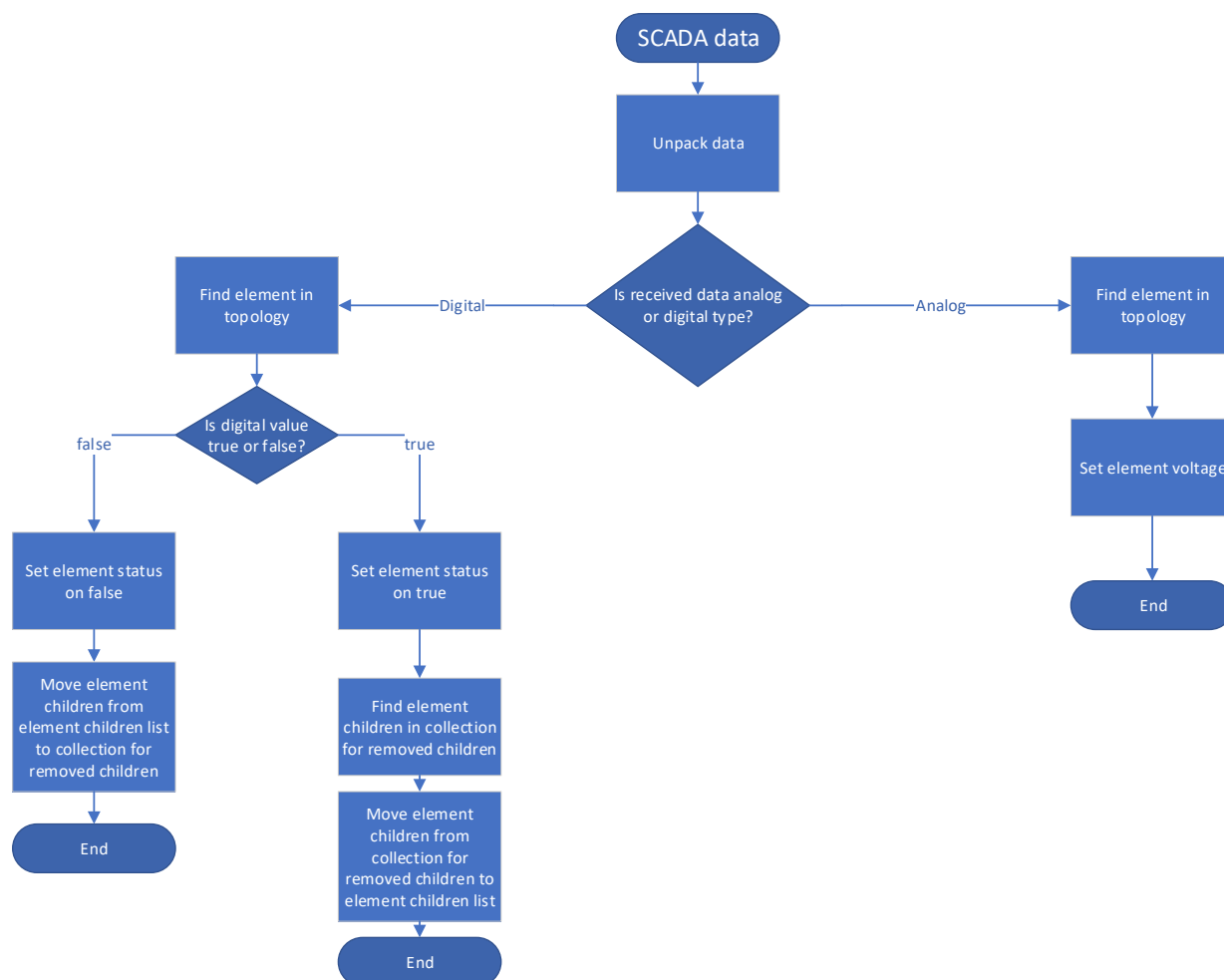
Simulator oslikava stanje sistema, odnosno stanje prekidača i vrednosti napona u mreži, a SCADA je zadužena za prikupljanje vrednosti signala. Nakon završenog procesa mapiranja signala SCADA pokreće periodičnu proveru definisanih adresa na simulatoru. U slučaju detekcije promena nad analognim i digitalnim tačkama SCADA šalje izmene odgovarajućoj DDS instanci kako bi se ažurirala topologija mreže na nivou *korena* na DDS-u, tj. kako bi DDS imao realnu sliku sistema i prekidačke opreme. Takođe, SCADA konstatuje i eventualna alarmna stanja analognih i digitalnih tačaka i prosleđuje ih DDS instanci koja ih dalje propagira na PubSub kako bi klijent dobio adekvatno upozorenje u vidu ispisa na ekranu. Na

slici 12 je dijagramom sekvenci prikazana periodična provera promene vrednosti na definisanim adresama na simulatoru i dalji tok rada u slučaju eventualne detekcije promena.



Slika 12. Periodična provera promena na simulatoru

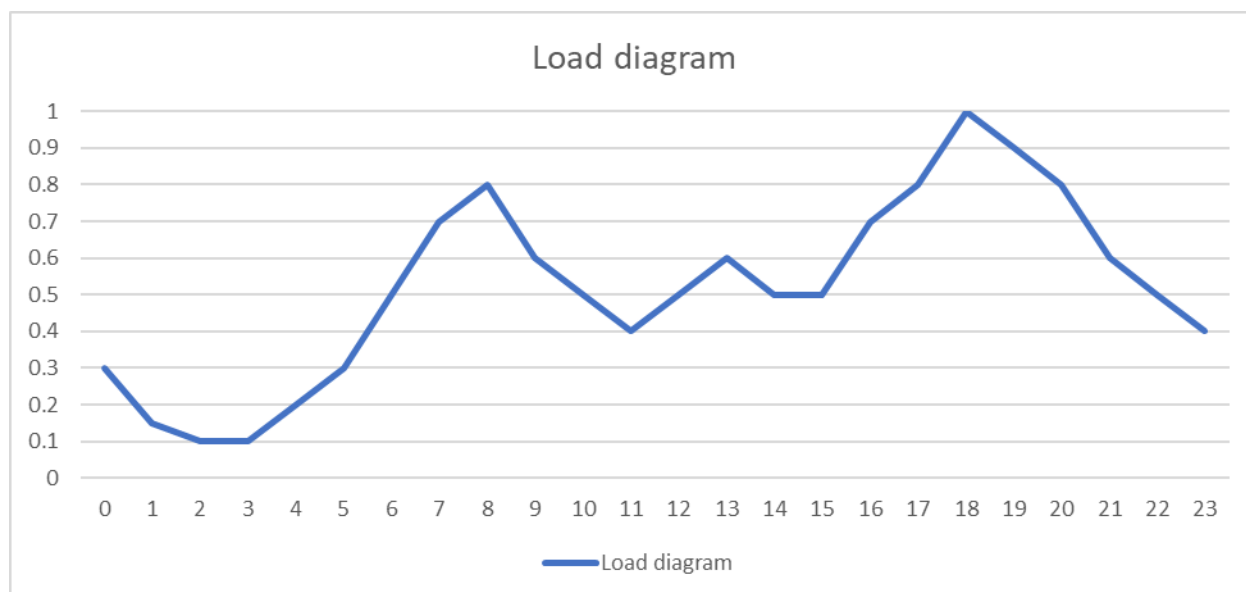
Ukoliko ih ima više, instance DDS-a se podižu po utvrđenom rasporedu i u skladu sa tim se javljaju NMS, SCADA i PubSub komponentama. Na ovaj način je obezbeđeno da i NMS i SCADA čuvaju u listama isti redosled komunikacionih kanala ka instancama DDS-a, kako ne bi došlo do greške da SCADA dostavlja promene pogrešnoj instanci DDS-a. DDS komunicira sa NMS komponentom i dobavlja potrebne informacije za izgradnju topologije mreže na nivou *korena* koja se formira u obliku N-arnog stabla. Struktura u vidu stabla omogućava jednostavno kretanje kroz elemente mreže, njihovo pronalaženje i lakšu manipulaciju nad njima. Prilikom prijema promena sa SCADA komponente se vrši ažuriranje stabla u zavisnosti od pristigle promene. Na slici 13 je prikazan proces obrade pristiglih promena sa SCADA komponente.



Slika 13. Obrada pristiglih SCADA promena na DDS-u

Obrada SCADA promena je veoma bitna zbog stvarnog uvida u stanje sistema i dobijanja tačnih proračuna prilikom računanja potrošnje mreže. Promene sa SCADA komponente dolaze na DDS kao niz bajtova, pa je neophodno prvo konvertovati pristigle podatke u odgovarajuće tipove podataka gde se na osnovu tipa signala se određuje da li promene nose analognu ili digitalnu vrednost. Kod analognih signala je potrebno pronaći element na koji se odnosi promena, a potom mu postaviti pristiglu vrednost. Ako je u pitanju digitalni signal, element se pronalazi, postavlja mu se pristigla vrednost i ažurira lista referenci ka hijerarhijski nižim čvorovima.

DDS takođe vrši i generisanje simulacionog fajla koji definiše ponašanje potrošača, odnosno njihovu potrošnju tokom rada programa. Na osnovu dnevne krive potrošnje električne energije vrši se simulacija potrošnje gde svaki potrošač koristi određenu snagu u zavisnosti od satnice, odnosno dela dana. Tako će potrošnja kada ljudi spavaju, npr. u 03:00h, biti mnogo manja nego kada se ljudi spremaju za posao, npr. u 07:00h. Slika 14 prikazuje potrošnju jednog potrošača u toku 24 časa izraženu u opsegu od 0 do 1. Na osnovu maksimalne potrošnje koju jedan potrošač može da ima i dijagrama potrošnje (Slika 14), formiraju se simulacione vrednosti svakog potrošača u toku 24 časa. Simulacija dnevne potrošnje funkcioniše tako što se svakom potrošaču postavlja određena snaga koja se menja na svakih n sekundi, a definiše se tako što se maksimalna snaga potrošača, koja je definisana kroz CIM XML fajl, pomnoži sa vrednošću sa dijagrama potrošnje u zavisnosti od sata koji se simulira. Svaka iteracija predstavlja sat vremena stvarnog života.



Slika 14. Dijagram potrošnje

Potrošnja, u bilo kom delu dana, ne sme da naruši maksimalno dozvoljenu. Ne sme da dođe do strujnog preopterećenja u distributivnoj mreži. Strujno opterećenje se računa sumiranjem snaga potrošača u *korenu*. Zatim se taj zbir deli sa naponom na izvoru kako bi se dobila jačina struje, a potom se ista poredi sa limitom sekcije koja provodi struju. Dobijena vrednost ne sme biti veća u odnosu na propisanu jačinu električne struje koju vod može da provodi. Listing 1 prikazuje pseudo kod algoritma za računanje trenutne potrošnje u mreži.

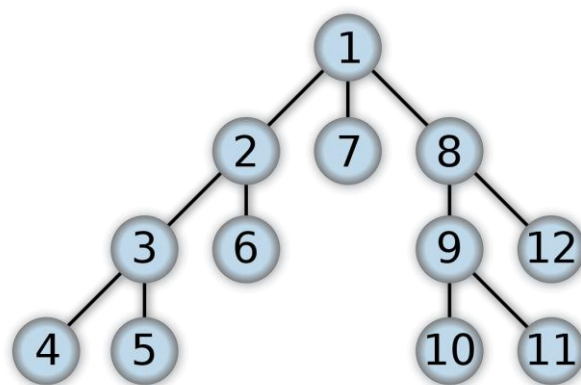
```
float CalculateConsumption(root)
{
    float load = 0
    foreach child in rootChildren
        load += CalculateConsumption(child)

    if (rootType == Node)
        rootLoad = load
    else if (rootType == Consumer)
        return rootValue

    return load
}
```

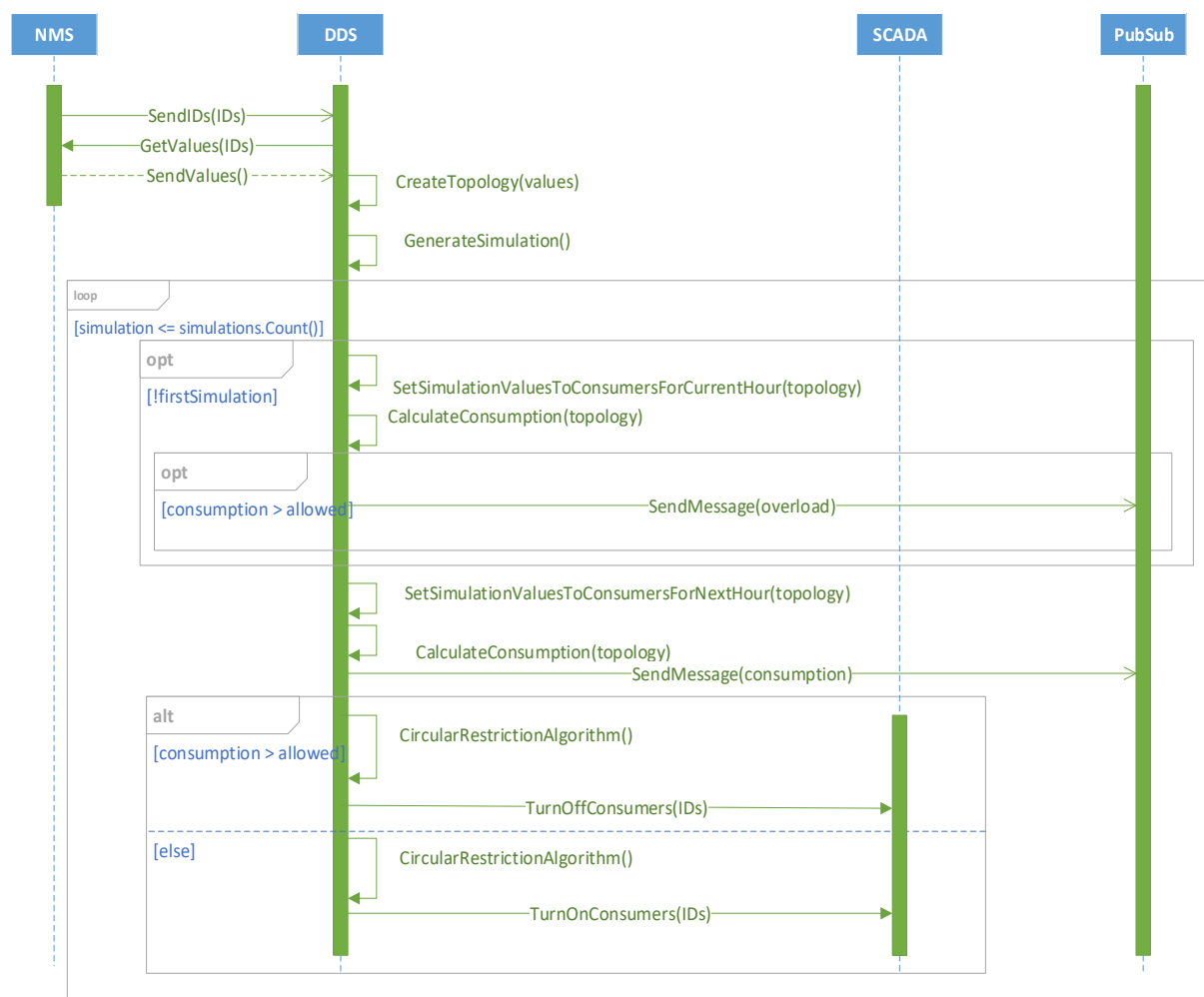
Listing 1. Pseudo kod algoritma za računanje trenutne potrošnje

Za računanje trenutne potrošnje se koristi DFS (eng. *Depth First Search*) algoritam. Početna tačka je *koren stabla* (eng. *root*) i predstavlja izvor električne energije. Uzima se prvo *dete* (eng. *child* – čvor povezan sa čvorom koji je hijerarhijski iznad njega) korenskog čvora i rekurzivno se poziva algoritam sve dok se ne dodje do čvora koji nema *dece - list* (eng. *leaf*), odnosno u ovom radu do potrošača. Rekurzija omogućava obilazak čitavog stabla i funkcioniše tako što se traži prvi *roditelj* (eng. *parent* – čvor koji je hijerarhijski iznad tekućeg čvora) koji ima neposećeno *dete* i vrši se ponovno pozivanje funkcije. Na slici 15 je prikazan primer obilaska stabla gde je sa brojem unutar čvora označen redosled posećivanja čvorova.



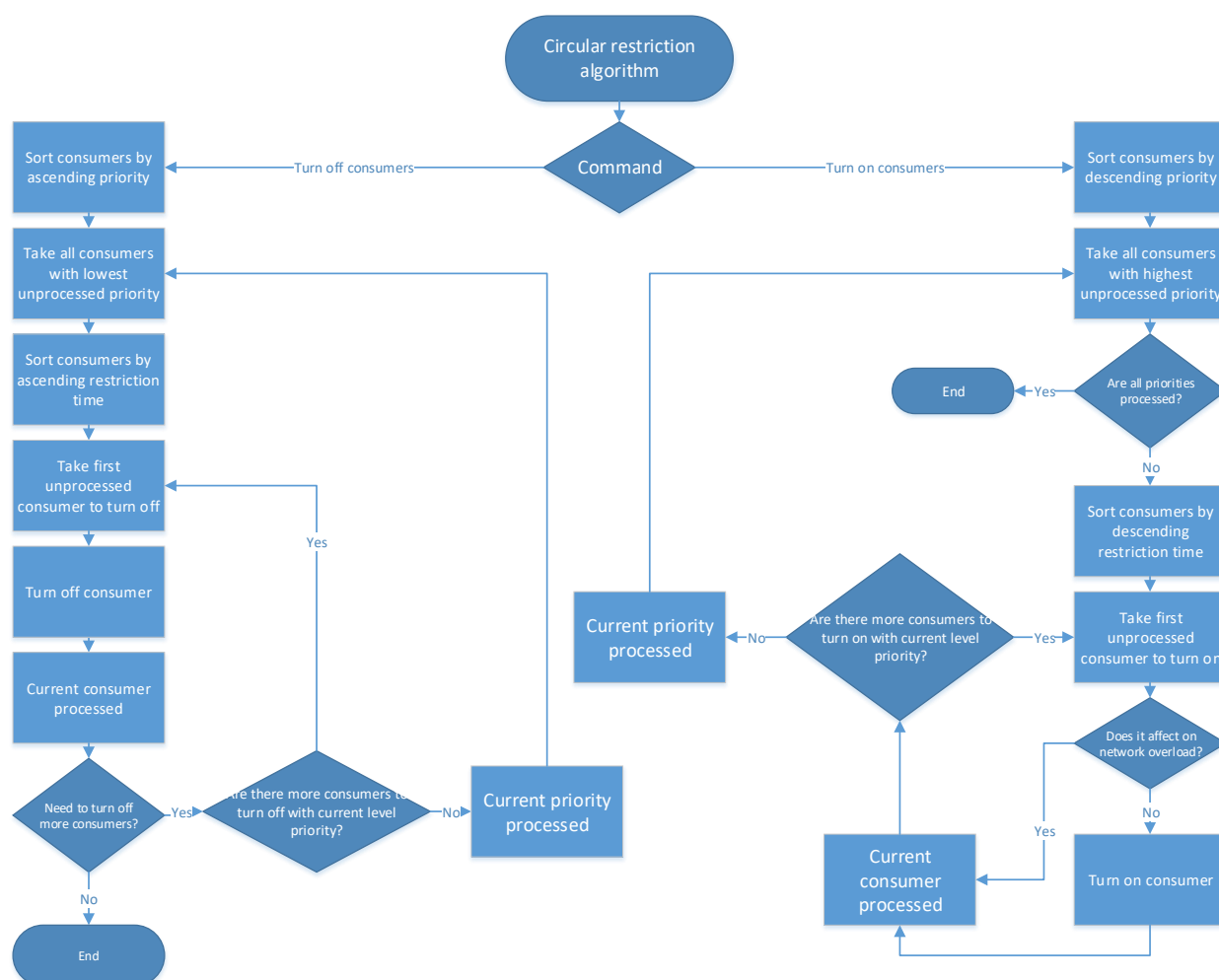
Slika 15. Stablo sa označenim redosledom posećivanja čvorova

U svakoj iteraciji simulacije se uzimaju vrednosti potrošnje za naredni sat koji su dobijeni dnevnom krivom potrošnje. Sa tim vrednostima se vrši proračun potrošnje mreže kako bi se preventivno delovalo na sistem, da ne dođe u stanje strujnog preopterećenja. Ukoliko proračun pokaže da će u sledećoj iteraciji doći do strujnog preopterećenja, pokreće se *algoritam kružnih isključenja* koji određuje potrošače koje je neophodno isključiti da ne bi došlo do pomenutog scenarija. Drugim rečima, aplikacija u tekućoj iteraciji, po potrebi, vrši isključenja potrošača kako bi mreža u narednoj iteraciji ostala u normalnom stanju. Na slici 16 je prikazan rad aplikacije tokom simulacionog perioda.



Slika 16. Proces obrade simulacija

Algoritam kružnih isključenja vodi računa o ravnomernoj restrikciji električne energije između potrošača, u slučaju strujnog preopterećenja, striktno obračavajući pažnju na njihov prioritet. Npr. bolnica je potrošač najvišeg prioriteta i ne sme ostati bez napajanja, dok su obična domaćinstva najnižeg prioriteta i oni će prvi ostati bez električne energije u slučaju prekomernog korišćenja iste. Isto tako, algoritam vodi računa da vreme provedeno bez napajanja između potrošača istih prioriteta bude ujednačeno. Dijagram stanja, na slici 17, prikazuje kako funkcioniše *algoritam kružnih isključenja*.



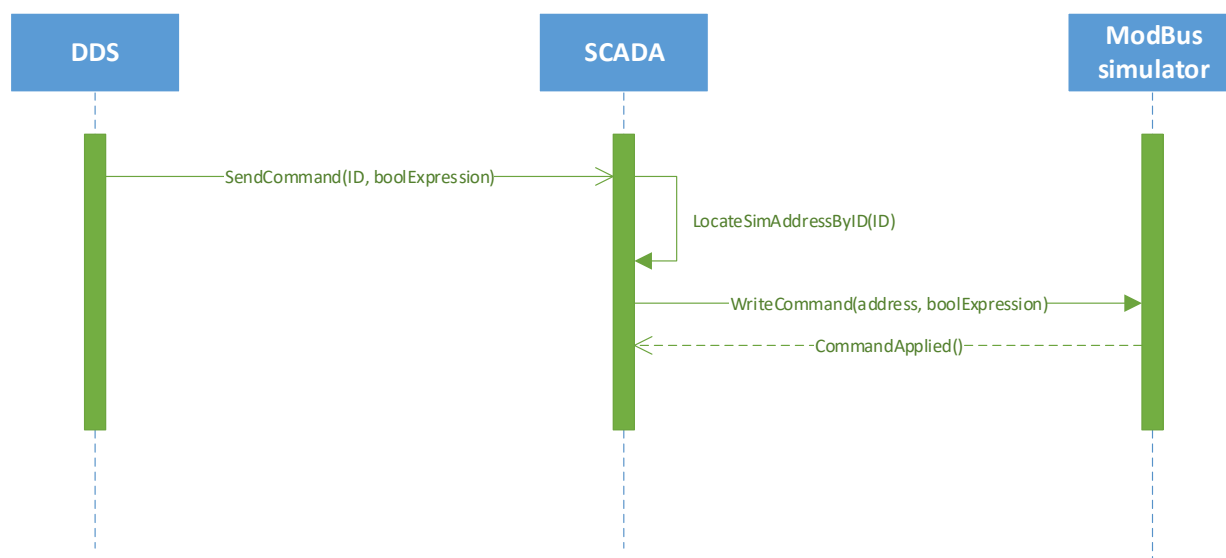
Slika 17. *Algoritam kružnih isključenja*

Sa slike 17 se može videti da se prilikom isključivanja potrošača prvo vrši sortiranje po prioritetu - od najnižeg do najvišeg prioriteta, a onda se potrošači najnižeg prioriteta sortiraju po vremenu provedenom bez električne energije, tako da prvi u nizu bude potrošač koji je najmanje vremena proveo bez električne energije. Vršiti se isključivanje potrošača koji su napajani električnom energijom sve dok se ne utvrdi da se strujno preopterećenje mreže neće desiti. U slučaju isključenja svih potrošača istog prioriteta, prelazi se na sledeći nivo prioriteta gde se postupak ponavlja.

Na dijagramu sa slike 17 je opisan i slučaj uključivanja potrošača, gde se proverava da li je moguće isključene potrošače ponovo napajati električnom energijom. Potrošači se sortiraju od najprioritetnijeg nivoa do najmanje prioritetnog, a potom se u okviru potrošača najvišeg prioriteta vrši sortiranje tako da se prvo posmatra potrošač koji je najviše vremena proveo bez električne energije. Algoritam redom

obrađuje sve potrošače, svih prioriteta, pri čemu uključuje samo one čije uključivanje neće izazvati preopterećenje mreže.

Uključivanje i isključivanje potrošača je usko vezano za komandovanje SCADA sistemom, pa je tok komunikacije prilikom slanja komandi SCADA komponenti od strane DDS-a prikazan na slici 18.



Slika 18. Slanje komandi na SCADA komponentu

Informacije o trenutnom stanju potrošnje u mreži klijent dobija od DDS-a, posredstvom PubSub-a, tako da u svakom trenutku ima uvid u realno stanje sistema. Ovaj mehanizam obezbeđuje svim zainteresovanim stranama dotok informacija u zavisnosti od tema na koju su pretplaćeni. Funkcioniše kao posrednik koji na jednostavan način može da vrši distribuciju informacija većem broju klijenata.

5.3. Tok rada aplikacije u *Service Fabric* okruženju

U poglavlju 5.2. je opisan princip rada *on-premise* aplikacije koji u velikoj meri važi i za *Service Fabric* aplikaciju. Razlika se uočava u arhitekturi sistema, početnoj komunikaciji servisa i toku podataka što je uzrokovano dodavanjem posrednika u *Service Fabric* klasteru. Aplikacija i dalje obavlja identičan posao, ali uz drugačiju efikasnost.

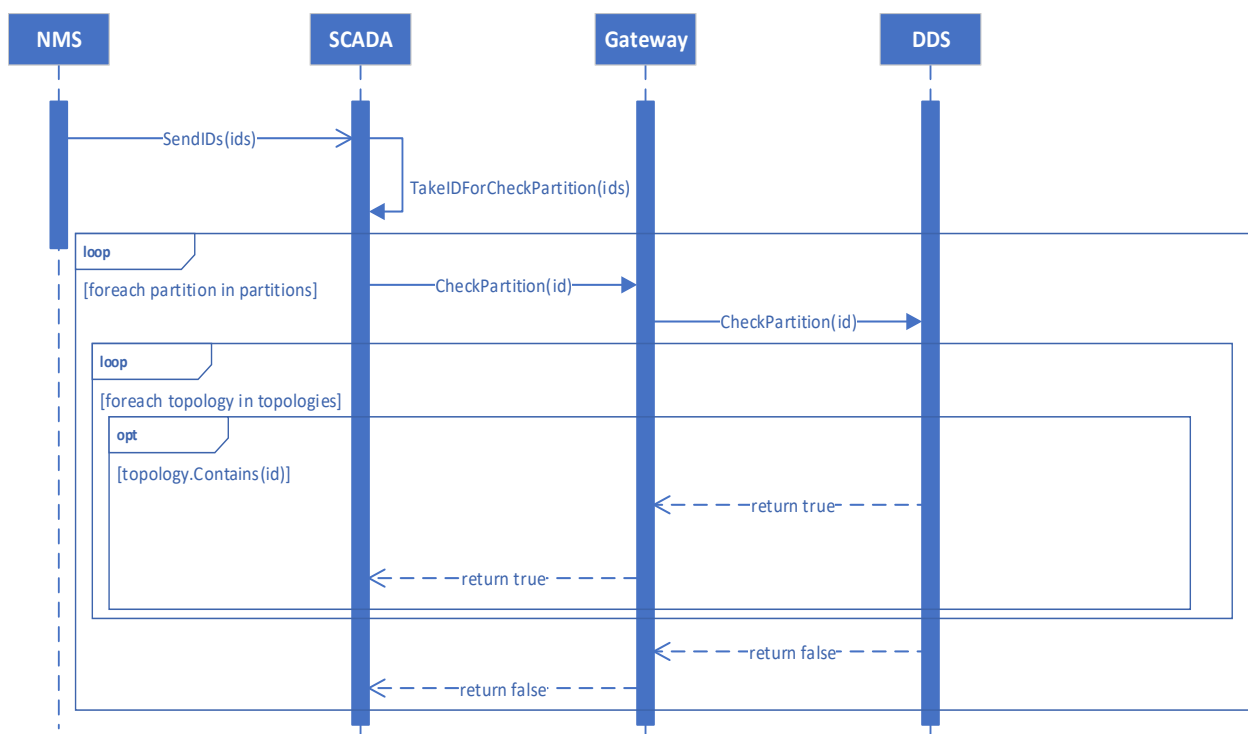
Komunikacija između komponenti se u *on-premise* aplikaciji odvijala direktno, jer je za uspostavu konekcije bilo dovoljno znanje IP adrese (eng. *Internet Protocol address*) i porta druge strane. Ali, u *Service Fabric*-u se komunikacija odvija na drugačiji način. Da bi *Service Fabric* servisi bili dostupni i da bi komunikacija sa njima bila moguća, moraju se definisati tzv. *listener*-i u okviru kojih se definišu pravila pod kojima će komunikacija biti ostvariva. Protokol i port servisa se učitavaju kroz konfiguracioni fajl.

Definisanjem *listener*-a će servis postati dostupan, ali samo u okviru klastera, dok je direktna komunikacija sa istim, izvan klastera, onemogućena. Klijentima, tj. računarima izvan mreže u kojoj se nalazi klaster, dostupna je samo adresa *Load Balancer*-a koji je zadužen za prosleđivanje saobraćaja na čvorove klastera. Sav „spoljni saobraćaj“ dolazi do njega i da bi on znao kom servisu da prosledi poruku,

neophodno je definisati pravila na *Azure* portalu, gde se određeni port vezuje za određeni servis. Na taj način se vrši *port forwarding* gde će na osnovu porta na koji se šalje zahtev, *Load Balancer* tačno znati kom servisu da ga prosledi.

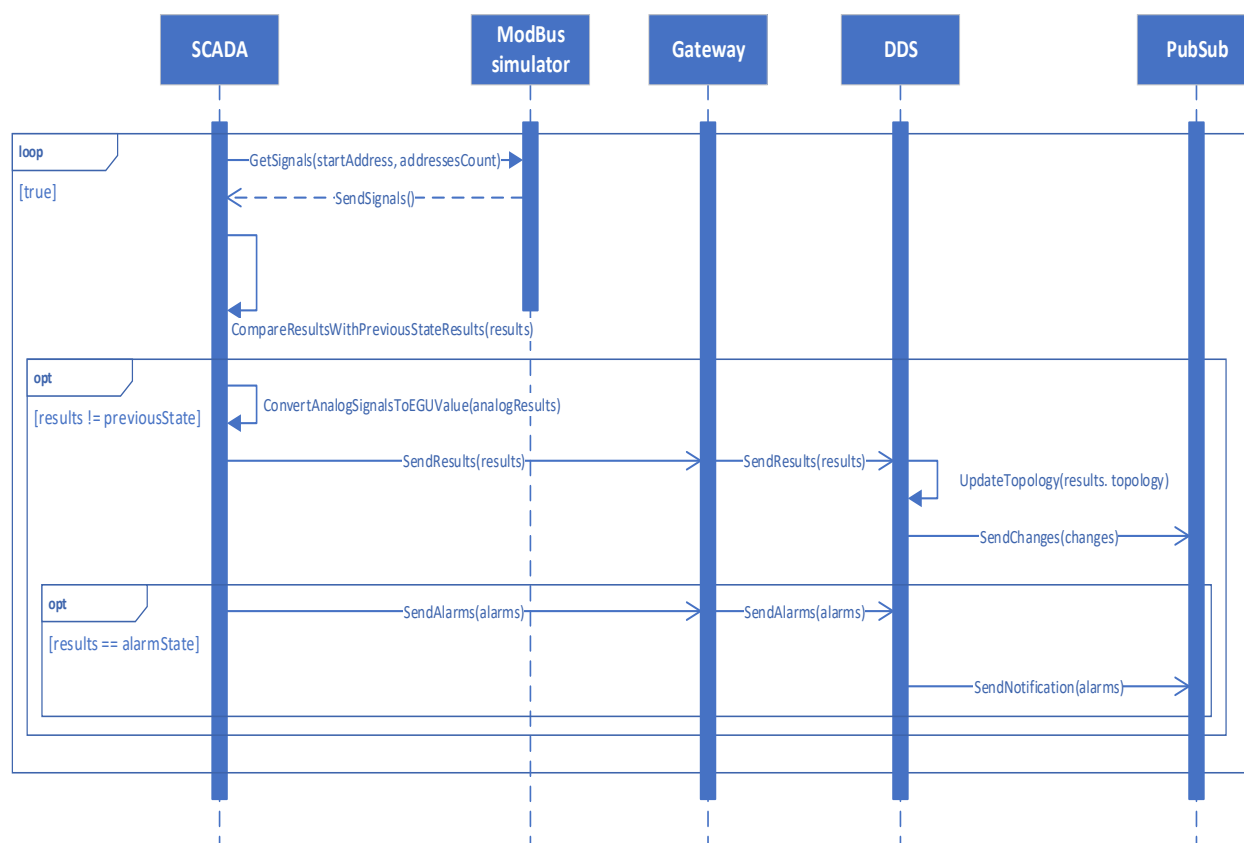
Pogađanje određene particije dodatno komplikuje situaciju. Postoji nekoliko načina za realizaciju, a jedan od opcija je uvođenja posrednika (*Gateway* komponenta). Kod ovog rešenja je potrebno obezbediti *port forwarding* za posrednika i proslediti mu ključ particije. Na osnovu ključa particije koji dobije, posrednik uspostavlja komunikaciju sa tačno utvrđenom particijom na kojoj se servis nalazi i prosleđuje zahteve. Komunikacijom unutar klastera je jednostavnije pogoditi određenu particiju nego što je to slučaj izvan.

U razvijanoj aplikaciji uloga posrednika funkcioniše tako što SCADA koja se nalazi na lokalnom računaru pogađa *endpoint Gateway*-a prosleđujući mu ključ particije. Na osnovu prosleđenog parametra *Gateway* pravi dvosmernu komunikaciju sa instancom DDS-a na određenoj particiji i obezbeđuje protok podataka između SCADA i DDS komponente. Proces određivanja particije na kojoj se nalazi tražena DDS instanca je prikazan na slici 19.



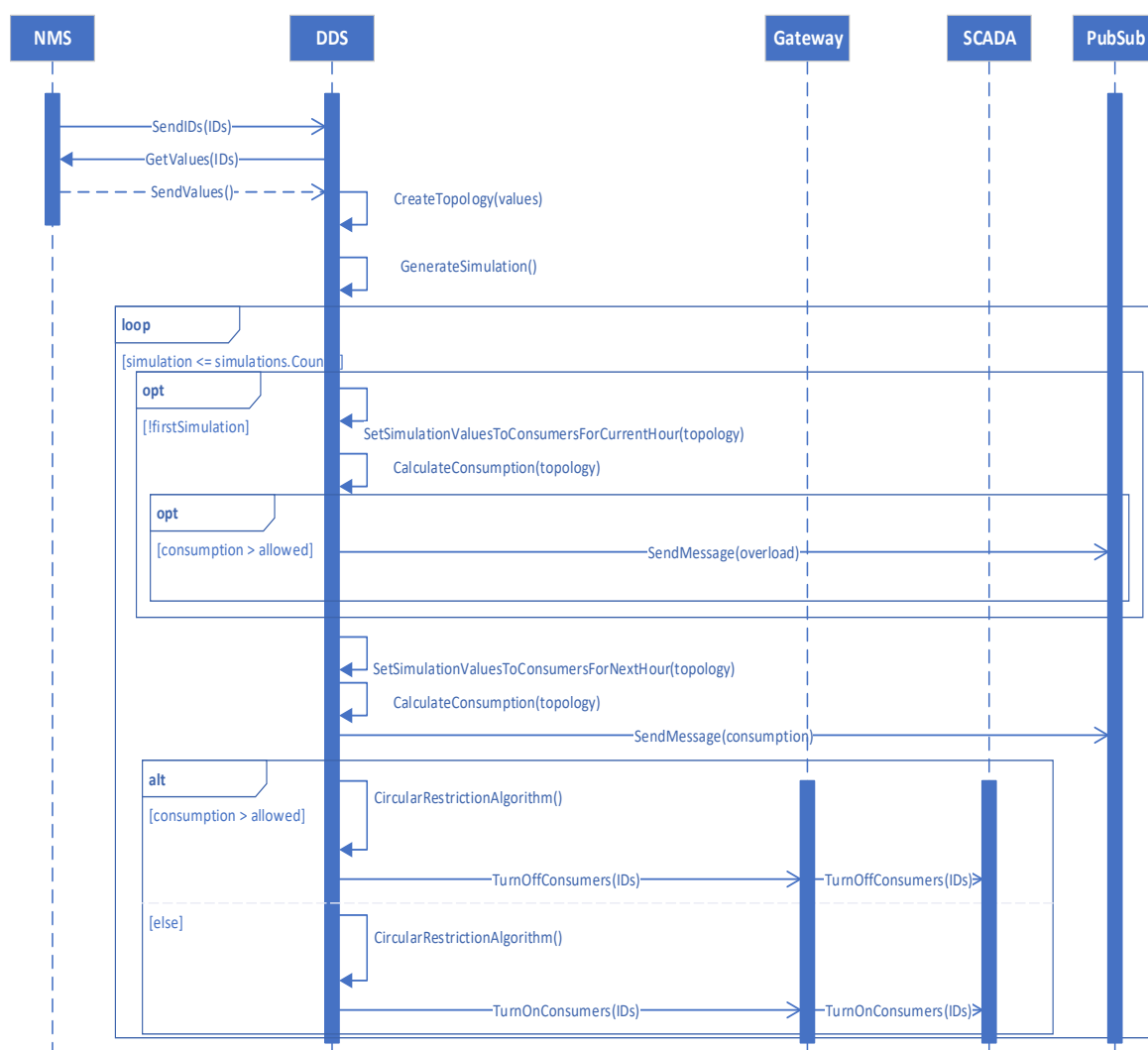
Slika 19. Postupak određivanja particije DDS instance

Prilikom periodične provere promena vrednosti na simulatoru, SCADA više ne javlja promene direktno DDS-u, nego ih šalje ih *Gateway*-u koji ih dalje prosleđuje DDS-u. Slika 20 prikazuje realizaciju periodične provere promena vrednosti na simulatoru u okviru *Service Fabric* aplikacije.



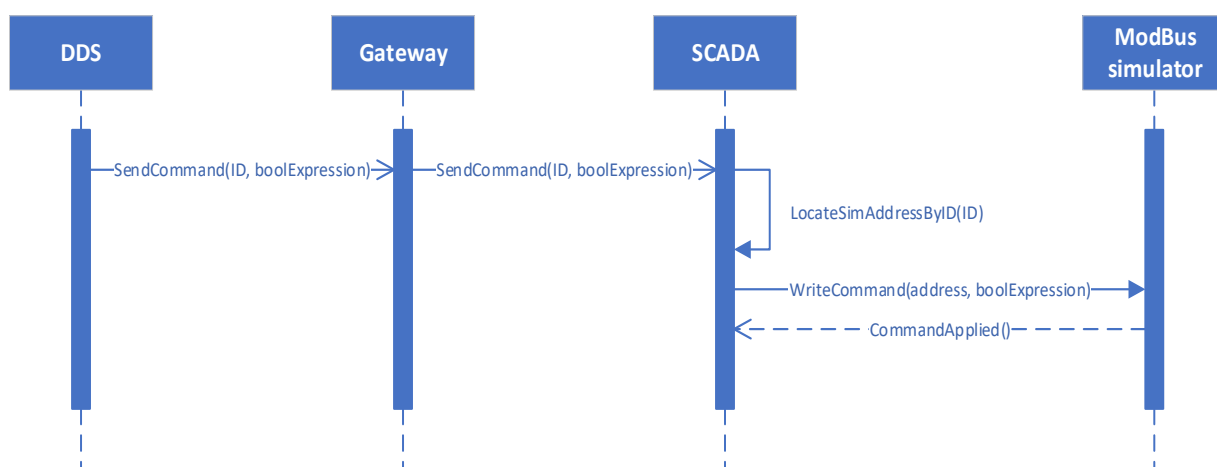
Slika 20. Periodična provera promene vrednosti u Service Fabric-u

Takođe, promena u komunikaciji je uočljiva i prilikom obrade simulacija, gde komande namenjene SCADA sistemu DDS šalje na *Gateway* koji je zadužen za njihovo dalje usmeravanje. Slika 21 prikazuje tok komunikacije prilikom simulacionog perioda u *Service Fabric* aplikaciji.



Slika 21. Obrada simulacije u Service Fabric aplikaciji

Za razliku od *on-premise* aplikacije, komandovanje SCADA sistemom od strane DDS-a će se obavljati preko *Gateway*-a u *Service Fabric* aplikaciji. Na slici 22 je prikazan slučaj slanja DDS komandi.



Slika 22. Slanje komandi ka SCADA sistemu

NMS funkcioniše i dalje na isti način, ali je pokrenut u *Service Fabric* klasteru, baš kao i PubSub. Posao *Importer*-a je identičan, prolazi kroz iste operacije kao i u *on-premise* aplikaciji, s tim što komunicira sa NMS-om koji se nalazi u *Service Fabric* klasteru. Samim tim, jedina promena je u komunikaciji koja se više ne odvija u lokalnoj mreži, već se pravi komunikacioni kanal ka servisu koji se nalazi u udaljenom klasteru. Komunikacija između SCADA sistema i simulatora ostaje nepromenjena, a klijent dobija informacije o stanju mreže pretplatom na odgovarajuću temu PubSub-a. Klijentu je omogućeno da se pretplati na temu, jer je port na kom radi PubSub otvoren, odnosno postoji pravilo na osnovu kog *Load Balancer* prosleđuje zahteve na njega.

6. Testiranje

Testiranje je sprovedeno u cilju poređenja performansi *Service Fabric* aplikacije sa performansama *on-premise* aplikacije. Kao rezultat testiranja se dobija broj *korena* koje aplikacija uspeva uspešno da obradi u iterativnom periodu. Drugim rečima, rezultat testiranja je broj *korena* koje aplikacija može da obradi bez pojave strujnog preopterećenja. Promenljivi parametri prilikom testiranja su: broj DDS instanci koje su zadužene za obradu simulacije nad mrežom, trajanje pauze između dve iteracije simulacije (iterativni period) i vremenski interval na koji se vrši periodično prikupljanje podataka na SCADA komponenti.

Rezultati prikazani u tabelama 2 i 3 predstavljaju broj uspešno obrađenih *korena*. Sa n je označen broj pokrenutih DDS instanci, sa *sim* - trajanje pauze između dve iteracije simulacije, a sa *poll* - period za prikupljanje podataka na SCADA komponenti.

6.1. Testiranje *on-premise* aplikacije

Testiranje je izvršeno na računaru, odnosno virtuelnoj mašini, sa procesorom Intel Core i3-4130 CPU @ 3.4 GHz i memorijom od 32GB, od čega je 26GB dodeljeno virtuelnoj mašini. U tabeli 2 su prikazani dobijeni rezultati.

Tabela 2. Rezultati testiranja *on-premise* aplikacije

n \ sim	Poll = 2s				Poll = 1.4s			
	3s	5s	7s	10s	3s	5s	7s	10s
1	22	33	75	75	20	35	50	60
2	22	24	26	30	18	22	22	24
3	18	21	27	30	18	21	21	27
4	20	28	24	24	16	20	20	24
5	15	25	25	25	15	20	20	20

6.2. Testiranje *Service Fabric* aplikacije

Servisi pokrenuti u *Service Fabric* okruženju su podignuti na virtuelnim mašinama koje poseduju Intel Xeon CPU E5-2673 v3 @ 2.40 GHz, 1 Core procesor i RAM memoriju od 3.5 GB.

Ostale komponente su pokrenute na računaru sa memorijom od 6 GB i procesorom Intel Core i5-4200U CPU @ 1.60 GHz, turbo 2.30 GHz.

Rezultati dobijeni testiranjem su prikazani u tabeli 3.

Tabela 3. Rezultati testiranja *Service Fabric* aplikacije

n \ sim	Poll = 2s				Poll = 1.4s			
	3s	5s	7s	10s	3s	5s	7s	10s
1	24	25	38	40	20	28	30	30
2	20	30	36	36	16	26	28	30
3	18	21	30	36	18	27	27	30

Poređenjem rezultata iz tabela 2 i 3 se može primetiti da su rezultati u većini slučajeva približni. Uočava se da u slučaju pokrenute jedne DDS instance *on-premise* aplikacija postiže bolje rezultate, odnosno može uspešno da obrađuje više *korena*. To je naročito izraženo u slučajevima kada je trajanje pauze između dve iteracije simulacije (*sim*) 7s i 10s. Aplikacija pokrenuta u *Service Fabric* okruženju daje za nijansu bolje rezultate kada je startovano više DDS instanci. Postoje slučajevi kada *on-premise* aplikacija postiže bolje rezultate, kada je pokrenuto više od jedne DDS instance (pokrenute dve DDS instance, pri pauzi između simulacija od 3 sekunde – *sim* = 3s), ali u svim ostalim slučajevima sa više DDS instanci *Service Fabric* aplikacija je uspešnija.

7. Zaključak

U ovom radu je opisana i testirana aplikacija koja simulira rad distributivnog dela elektroenergetskog sistema. Objašnjeno je na koji način radi i kojim se metodama služi u cilju zaštite mreže od strujnog preopterećenja. Ispitana je njena efikasnost varijacijom vrednosti parametara od značaja. Korišćenja su dva testna okruženja pri čemu su u većini slučajeva dobijeni slični rezultati.

On-premise aplikacija je bolje rešenje kada je pokrenuta jedna DDS instanca zato što bolje hardverske komponente lokalnog računara dolaze do izražaja u odnosu na iznajmljenu virtuelnu mašinu. Značajna prednost je i to što podaci cirkulišu brže u okviru lokalne mreže, na istom računaru, nego što je slučaj prilikom komunikacije sa *Service Fabric* klasterom. *Service Fabric* rešenje daje bolje rezultate sa više DDS komponenti, jer se svaka od njih pokreće na različitoj virtuelnoj mašini, za razliku od *on-premise* aplikacije gde se sve DDS instance pokreću na istom računaru. Resursi u vidu hardvera daju razliku u rezultatu. Ta razlika bi verovatno bila veća da je postojala mogućnost iznajmljivanja boljih računara, sa boljom hardverskom konfiguracijom. Probna verzija naloga na *Microsoft Azure* portalu ograničava korisnika da iznajmi virtuelne mašine sa novčanim ograničenjem (200\$ u periodu testiranja), za period od mesec dana. Pritom, može se napraviti klaster sa najviše 3 čvora, s ograničenjem da skup iznajmljenih virtuelnih mašina ima na raspolaganju 4 jezgra. Drugim rečima, broj jezgara procesora iznajmljenih virtuelnih mašina u zbiru ne može biti veći od 4. Kada se posmatraju pomenuta ograničenja razumljivo je što prednosti distribuiranog sistema nisu u potpunosti izražene i što rezultati nisu u skladu sa očekivanim.

Implementirano rešenje bi se moglo poboljšati uvođenjem *failover* mehanizma i replikacije. *Failover* bi povećao dostupnost i pouzdanost aplikacije, jer otkaz neke od komponenti ne bi prouzrokovao prekid rada aplikacije, dok bi replikacija obezbedila konzistentne podatke na svim *stand by* komponentama. Tako bi aplikacija u svakoj *stand by* komponenti imala najsvežije podatke i bila bi spremna da neometano nastavi sa radom, bez afektovanja korisnika.

Trenutno je komunikacija između SCADA komponente i simulatora realizovana *Modbus* protokolom. Kako je *Modbus* najstariji industrijski protokol, moguća unapređenja aplikacije bi se mogla odnositi i na implementaciju nekog novijeg i boljeg industrijskog protokola.

Sve više se pominje i migracija SCADA komponente na *cloud*, tako da bi se i ta vrsta implementacije mogla uzeti u obzir kao izazov u budućnosti.

Dalja istraživanja u vezi rada bi se mogla bazirati i na migraciju aplikacije na *cloud* platformu neke druge kompanije. U ovom radu je izvršena migracija na platformu kompanije *Microsoft*, ali bi bilo zanimljivo istražiti koje mogućnosti i performanse nude *cloud* platforme drugih kompanija (npr. *Amazon*).

Opseg istraživanja bi se mogao proširiti i na algoritme koji bi se mogli koristiti kao zamena algoritama koji se trenutno koriste u radu, pri čemu bi se vršila i komparacija postignutih performansi u odnosu na prvobitno rešenje.

8. Literatura

- [1] P. Vidović, „Proračuni tokova snaga neuravnoteženih distributivnih mreža“, doktorska disertacija, Fakultet tehničkih nauka u Novom Sadu, Novi Sad, 2015.
- [2] S. Borlase, *Smart Grids: Advanced Technologies and Solutions*, 2nd ed., CRC press, 2017.
- [3] Microsoft, *Azure Service Fabric Documentation*, <https://docs.microsoft.com/en-us/azure/service-fabric/>, (pristupljeno u novembru 2019.)
- [4] dr Vladimir C. Strezoski, „Osnovi elektroenergetike“, Fakultet tehničkih nauka u Novom Sadu, Novi Sad, 2019.
- [5] Electrical engineering course, *Power system grid*, <http://electrical-engineering-course.blogspot.com/2011/06/power-system-grid.html>, (pristupljeno u januaru 2020.)
- [6] Jovanović, M. M., Tabisz, W. A., & LEE, F. C. (1994). *Distributed power systems—benefits and challenges. International Journal of Electronics*, 77(5), 601–612.
- [7] NetGain Energy Advisors, *The US Deregulated Electricity Market*, <http://www.netgainenergyadvisors.com/market-overview.php>, (pristupljeno u januaru 2020.)
- [8] Liao, Y. (2013). *Transformation of Electric Power Grid into Smart Grid. International Journal of Advance Innovations, Thoughts & Ideas*, 02(04)
- [9] McDaniel, P., & McLaughlin, S. (2009). *Security and Privacy Challenges in the Smart Grid. IEEE Security & Privacy Magazine*, 7(3), 75–77
- [10] Samuel Bimenyimana and Godwin Norensa Osarumwense Asemota, „Traditional Vs Smart Electricity Metering Systems: A Brief Overview“, *Journal of Marketing and Consumer Research*, vol. 46, 2018.
- [11] A. Selakov, „Optimalno upravljanje mikro mrežama u karakterističnim radnim režimima“, doktorska disertacija, Fakultet tehničkih nauka u Novom Sadu, Novi Sad, 2017.
- [12] Gungor, V. C., Sahin, D., Kocak, T., Ergut, S., Buccella, C., Cecati, C., & Hancke, G. P. (2011). *Smart Grid Technologies: Communication Technologies and Standards. IEEE Transactions on Industrial Informatics*, 7(4), 529–539
- [13] Moslehi, K., & Kumar, R. (2010). *A Reliability Perspective of the Smart Grid. IEEE Transactions on Smart Grid*, 1(1), 57–64.
- [14] Meinberg, *Precise Timing for Power Industries and the Smart Grid*, <https://www.meinbergglobal.com/english/industries/smart-grid-timing.htm>, (pristupljeno u januaru 2020.)
- [15] Microsoft, *Orchestrating microservices and multi-container applications for high scalability and availability*, <https://docs.microsoft.com/en-us/dotnet/architecture/containerized-lifecycle/design-develop-containerized-apps/orchestrate-high-scalability-availability>, (pristupljeno u januaru 2020.)
- [16] Andrew S. Tanenbaum, Maarten Van Steen, „*Distributed Systems: Principles and Paradigmas*“, Vrije Universiteit, Amsterdam, The Netherlands
- [17] EJB Tutorial, *Challenges for a Distributed System*, <https://www.ejbtutorial.com/distributed-systems/challenges-for-a-distributed-system>, (pristupljeno u decembru 2019.)
- [18] Free Code Camp, *A Thorough Introduction To Distributed Systems*, <https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/>, (pristupljeno u decembru 2019.)
- [19] Tutorialspoint, *Distributed Systems*, <https://www.tutorialspoint.com/Distributed-Systems>,

- (pristupljeno u decembru 2019.)
- [20] GeeksforGeeks, *Design Issues of Distributed System*, <https://www.geeksforgeeks.org/design-issues-of-distributed-system/>, (pristupljeno u decembru 2019.)
- [21] Simmins, J. J. (2011). *The impact of PAP 8 on the Common Information Model (CIM)*. 2011 IEEE/PES Power Systems Conference and Exposition.
- [22] World Wide Web Consortium, *Resource Description Framework (RDF) Model and Syntax Specification*, <https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, (pristupljeno u januaru 2020.)
- [23] Microsoft, *What is .NET framework*, <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework>, (pristupljeno u januaru 2020.)
- [24] Tech Target, *C Sharp*, <https://searchwindevelopment.techtarget.com/definition/C>, (pristupljeno u januaru 2020.)
- [25] Microsoft, *What Is Windows Communication Foundation*, <https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>, (pristupljeno u januaru 2020.)

Podaci o kandidatu



Kandidat Dragan Stanković je rođen 03.06.1995. godine u Somboru. Završio je osnovnu školu „Nikola Tesla“ u Bačkom Brestovcu 2010. godine. Srednju ekonomsku školu u Odžacima završava 2014. godine. Fakultet Tehničkih Nauka u Novom Sadu upisao je 2014. godine, smer Elektroenergetski softverski inženjering. Osnovne studije završava 2018. godine i upisuje master studije na istom fakultetu, smer Primenjeno softversko inženjerstvo. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom.