Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Novi Sad

Kandidat: Nemanja Nedić

UPRAVLJENJE TOKOVIMA AKTIVNOSTI U DISTRIBUTIVNOM MENADŽMENT SISTEMU

doktorska disertacija

Mentor: dr Goran Švenda

Novi Sad, 2015.

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 документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		MSc Немања Недић
Ментор, **МН**:		др Горан Швенда, редовни професор
Наслов рада, **НР**:		Управљање токовима активности у дистрибутивном менаџмент систему
Језик публикације, **ЈП**:		српски
Језик извода, **ЈИ**:		српски
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		АП Војводина
Година, **ГО**:		2015
Издавач, **ИЗ**:		Факултет техничких наука
Место и адреса, **МА**:		Трг Д. Обрадовића 6, 21000 Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		9/90/144/6/25/6/2
Научна област, **НО**:		Електроенергетика
Научна дисциплина, **НД**:		Примењено софтверско инжењерство
Предметна одредница/Кључне речи, **ПО**:		управљање токовима активности, алгоритми, дистрибуирано рачунарство, паралелна рачунарска обрада, дистрибутивни менаџмент системи
УДК		
Чува се, **ЧУ**:		Библиотека Факултета техничких наука, Трг Д. Обрадовића 6, 21000 Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У раду је представљено истраживање везано за побољшање перформанси рада великих надзорно-управљачких система попут ДМС-а. Овај циљ је постигнут координацијом извршавања токова активности, што подразумева ефикасну расподелу задатака на рачунарске ресурсе. У те сврхе развијени су и тестирани различити алгоритми. Овакав приступ је обезбедио већи степен искоришћења рачунарских ресурса, што је резултирало бољим перформансама.
Датум прихватања теме, **ДП**:		12. 2. 2015.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др Владимир Стрезоски, редовни професор
	Члан:	др Миросла Хајдуковић, редовни професор
	Члан:	др Александар Ердељан, ванредни професор
	Члан:	др Драган Тасић, редовни професор
	Члан, ментор:	др Горан Швенда, редовни 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**:		Monograph documentation	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Ph.D. thesis	
Author, **AU**:		Nemanja Nedić, M.Sc.	
Mentor, **MN**:		Goran Švenda, Ph.D., Full Professor	
Title, **TI**:		Workflow management system for DMS	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Serbia	
Locality of publication, **LP**:		AP Vojvodina	
Publication year, **PY**:		2015	
Publisher, **PB**:		Faculty of Technical Sciences	
Publication place, **PP**:		Trg D. Obradovića 6, 21000 Novi Sad	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		9/90/144/6/25/6/2	
Scientific field, **SF**:		Power engineering	
Scientific discipline, **SD**:		Applied software engineering	
Subject/Key words, **S/KW**:		workflow management, algorithms, distributed computing, parallel computing, distributed management system	
UC			
Holding data, **HD**:		Library of Faculty of Technical Sciences, 21000 Novi Sad, Trg Dositeja Obradovića 6	
Note, **N**:			
Abstract, **AB**:		The paper presents an approach how to improve performance of larger scale distributed utility management system such as DMS. This goal is accomplished by using an intelligent workflow management. Workflows are divided into the atomic tasks which are scheduled to computing resources for execution. For these purposes various scheduling algorithms are developed and thoroughly tested. This approach has provided greater utilization of computing resources which further have resulted in better performance.	
Accepted by the Scientific Board on, **ASB**:		February 12th, 2015	
Defended on, **DE**:			
Defended Board, **DB**:	President:	Vladimir Strezoski, Ph.D., Full Professor	
	Member:	Miroslav Hajduković, Ph.D., Full Professor	
	Member:	Aleksandar Erdeljan, Ph.D., Associate Professor	Mentor's sign
	Member:	Dragan Tasić, Ph.D., Full Professor	
	Member, Mentor:	Goran Švenda, Ph.D., Full Professor	

SADRŽAJ

SPISAK SKRAĆENICA

Skraćenica	Engleski termin	Srpski termin
CERN	*European Organization for Nuclear Research*	Evropska organizacije za nuklearno istraživanje
CIM	*Common Information Model*	IEC-ov standard za model podataka elektroenergetskih sistema
CMS	*Customer Management System*	Sistem za menadžment podataka o potrošačima
DMS	*Distribution Management System*	Sistema za upravljanje distributivnom elektroenergetskom mrežom
EMS	*Energy Management System*	Sistema za upravljanje prenosnom elektroenergetskom mrežom
GIS	*Geographic Information System*	Sistema za upravljanje distributivnom elektroenergetskom mrežom
IEC	*International Electrotechnical Commission*	Međunarodna komisija za standardizaciju u oblasti elektrotehnike
OMS	*Outage Management System*	Sistem za upravljanje ispadima
SCADA	*Supervisory Control and Data Acquistion*	Sistema za nadzor, upravljanje i prikupljanje podataka
SDP	-	Servis dinamičkih podataka
SGP	-	Servis grafičkih podataka
SM	*Smart Metering*	Sistem za upravljanje pametnim meračima potrošnje električne energije
SSP	-	Servis statičkih podataka
SZP	-	Servis za proračune
TCP	*Transmission Control Protocol*	Transmisioni kontrolni protokol
UI	*User Interface*	Korisnički interfejs

SPISAK SLIKA

SPISAK TABELA

1. UVOD

Živimo u svetu energije. Sve što nas okružuje zasnovano je na njenoj upotrebi. Čovek je odavno shvatio da u prirodi postoji više različitih oblika energije koje može da iskoristi kako bi unapredio kvalitet života. Došao je do saznanja da vatra pomaže njegov opstanak u hladnim periodima, dok je energiju vode iskoristio za prevoz i izgradnju prvih mašina – vodenica. Na sličan način je naučio da koristi i moć vetra. Svojim napretkom čovek je sve uspešnije upotrebljavao i druge izvore energije koji su nastajali milionima godina. Ugalj, nafta i zemni gas su postali osnovni izvori energije jer ih je relativno lako eksploatisati i transformisati u osnovni oblik energije koji omogućava funkcionisanje današnje civilizacije - električnu energiju. Daljim tehnološkim postupcima i primenom određenih pretvarača, električna energija se može prevesti u druge, željene vidove energije, poput mehaničke i toplotne [1, 2].

Lako iskoristivi izvori energije, kao što su nafta i ugalj, po svojim kapacitetima predstavljaju veoma ograničene resurse, pa stoga postaju sve skuplji i teže dostupni. Ako se tome doda da njihovim korišćenjem (pretvaranjem u druge oblike energije: toplotnu energiju, električnu energiju) čovek značajno zagađuje okolinu, njihova ušteda i racionalna upotreba sa jedne strane, i traženje drugih izvora energije sa druge strane, predstavlja jedan od najvećih problema današnjice. Obzirom da je u poslednje dve decenije evidentan značajan rast potrošnje električne energije [3], neophodno je obezbediti njeno racionalno korišćenje na svim nivoima: u proizvodnji, prenosu, distribuciji i neposrednoj potrošnji [4]. Takav, veoma kompleksan zahtev nameće razvoj inteligentnih sistema, kako za nadzor i upravljanje, tako i za planiranje i analizu celokupnog elektroenergetskog sistema.

Ključnu ulogu u razvoju takvih sistema ima kvalitet i raspoloživost podataka o elektroenergetskoj mreži. Njihov nedostatak dovodi do usporavanja procesa odlučivanja, pa čak i do donošenja pogrešnih odluka u toku procesa upravljanja mrežom. Iz tog razloga, nove generacije elektroenergetskih sistema odlikuje sve veći stepen automatizacije i sve zahtevnija upotreba telekomunikacione i informacione tehnologije. To za posledicu ima da takvi, inteligentni elektroenergetski sistem, pored klasičnih elektroenergetskih komponenti, moraju da raspolažu sve zahtevnijom kontrolnom, računarskom i komunikacionom opremom, kao i opremom za dijagnostiku i automatizaciju. Dramatičan razvoj brze i pouzdane komunikacione i računarske tehnologije omogućio je da investicije u razvoj ovakvog sistema za efikasno korišćenje električne energije postanu tehnički i ekonomski opravdane. Njegova upotreba dovodi do povećanja pouzdanosti snabdevanja, smanjenja gubitaka električne energije i unapređenja procesa zaštite životne sredine. Univerzalni pojam koji se koristi za ovu vrstu sistema je *Smart Grid* [5, 6, 7].

Smart Grid čini kolekcija aplikacija koja obuhvata podsisteme kao što su:

- Sistema za nadzor, upravljanje i prikupljanje podataka (*Supervisory Control and Data Acquistion* – SCADA),
- Sistema za upravljanje prenosnom elektroenergetskom mrežom (*Energy Management System* – EMS),

- Sistema za upravljanje distributivnom elektroenergetskom mrežom [8] – Distributivni menadžment sistemi (*Distribution Management System* – DMS),
- Sistem za menadžment podataka o potrošačima (*Customer Management System* – CMS),
- Geografski informacioni sistem (*Geographic Information System* – GIS),
- Sistem za upravljanje pametnim uređajima za merenje potrošnje električne energije (*Smart Metering* – SM),
- Sistem za upravljanje ispadima (*Outage Management System* - OMS), itd.

Usled povezanosti sa gotovo svim ostalim podsistemima, Distributivni menadžment sistem (DMS) [9] predstavlja jedan od najvažnijih delova *Smart Grid* sistema. DMS je defnisan kao nadzorno - upravljački računarski sistem koji se koristi u kontrolnim centrima distributivne elektroenergetske mreže. Pored klasičnih SCADA funkcionalnosti, on podržava mogućnost analize trenutnog stanja distributivne mreže i predviđanja njenog budućeg ponašanja [10].

Upravljanje savremenim distributivnim mrežama postaje sve kompleksnije jer se proizvodnja električne energije sve više realizuje kroz nepouzdane i teško predvidive obnovljive izvore (na osnovu energije sunca, energije vetra, itd.). Broj i uticaj takvih malih i velikih distributivnih generatora, koji su u vlasništvu privatnih lica i distributivnih preduzeća, nezaustavljivo raste. Ako se tome doda da se savremena distributivna preduzeća nalaze na otvorenom tržištu, da su u obavezi da obezbede profit i da se značajno da modernizuju (da primene automatizovan sistem naplate, opslužuju električna vozila, itd.), jasna je potreba da u svoj sistem integrišu savremeni DMS [11]. Takav DMS se ne koristi samo za kontrolu distributivne mreže u realnom vremenu, već i za veliki broj "šta-ako" analiza i simulacija van realnog vremena.

Razvojem kako DMS-a, tako i infrastrukture distributivnih preduzeća, obezbeđeni su efikasan i praktično primenljiv korisnički interfejs koji podržava dinamičku vizualizaciju distributivne mreže, visoka sigurnost sistema (zaštita od neovlašćenog pristupa podacima, neovlašćenih izmena ili brisanja podataka, čime podaci postaju nedostupni neovlašćenim korisnicima), širok spektar analitičkih elektroenergetskih funkcija, istorija podataka (za analizu, kreiranje izveštaja i donošenje odluka), rukovanje multimedijom (prikaz fotografija i video zapisa sa terena), integracija sa ostalim sistemima čiji podaci mogu biti od velikog značaja za efikasno funkcionisanje [12], itd.

U toku rada DMS se oslanja na opis distributivne elektroenergetske mreže kojom upravlja. U tu svrhu se koriste modeli podataka koji opisuju celokupnu mrežu, odnosno sve njene entitete (generatore, transformatore, prekidače, vodove, potrošače, …) i njihovu konektivnost [13, 14, 15]. S obzirom na sve veće zahteve koji se postavljaju pred podsisteme za distribuciju električne energije, dolazi do proširenja tipova podataka, ali i do povećanja količine podataka koje ovi modeli sadrže. Tako na primer, za današnje, moderne distributivne sisteme broj entiteta, sa nekadašnjih par stotina hiljada, može da naraste na nekoliko miliona.

Izvršavanje analitičkih elektroenergetskih funkcija [16, 17, 18] predstavlja jednu od osnovinih i najbitnijih aktivnosti DMS-a. Uobičajeno, informacije iz distributivne mreže (izmerene vrednosti, statusi rasklopne opreme, itd.) se preko SCADA sistema šalju u DMS. Okidače za realizaciju rezidentnih DMS funkcija predstavljaju dovoljno velike izmene vrednosti (i/ili izmene statusa) ili ručni zahtevi korisnika. Rezultati tih proračuna obezbeđuju informacije o topologiji razmatrane distributivne mreže [19] i estimirane (procenjene) vrednosti njenog stanja [20]. Na osnovu topologije i estimiranog stanja omogućena je primena velikog broja DMS funkcija [9, 21]. Upotrebom DMS-a u

realnom vremenu operateri u kontrolnim centrima dobijaju informacije o celokupnoj distributivnoj mreži, što im znatno olakšava izbor upravljačkih akcija koje su potrebne da se mreža dovode u željeni režim. Korišćenje DMS-a van realnog vremena omogućava simulaciju rada distributivne mreže u željenim uslovima.

Nakon inicijalnog unosa, model podataka koji opisuje distributivnu mrežu se dosta retko menja, u odnosu na promene izmerenih vrednosti koje dolaze u DMS okruženje putem SCADA sistema. Iz tog razloga podaci koje sadrži model distributivne mreže mogu se tretirati kao *statički podaci*. Po istom osnovu, izmerene vrednosti se tretiraju kao brzo promenljivi, *dinamički podaci*.

Vrednosti električnih veličina, koje su neophodne za jednoznačno određivanje režima distributivne mreže, dobijaju se ili prenosom izmerenih veličina putem SCADA sistema u DMS ili procenom (estimacijom) njihovog stanja. Naravno, ekonomski je neisplativo obezbediti da se sve električne veličine mere i telemetrišu pa je uvek potrebno da se manji ili veći broj tih veličina proceni (estimira). Veći broj izmerenih veličina omogućava tačniju estimaciju stanja. Distributivne mreže, za razliku od prenosnih mreža, karakteriše znatno manja pokrivenost merenjima. Iskustvo stečeno na osnovu velikog broja DMS projekata pokazuje da je redundansa merenja u distributivnim mrežama (minimalan broj podataka na osnovu kojeg se u potpunosti, jednoznačno može rekonstruisati režim razmatrane mreže) svedena na nivo od par procenata [22], [23]. Shodno tome, estimacija stanja predstavlja osnovnu DMS funkciju na osnovu čijih rezultata se realizuju sve ostale DMS funkcije. Kao što je naglašeno, u distributivnim sistemima veliki broj električnih veličina ima dovoljno velike promene da u kratkom vremenu izazove veliki broj zahteva za pokretanje proračuna skupa rezidentnih funkcija (formiranje matematičkog modela mreže, provera topologije, estimacija stanja, itd.). Kako rastu dimenzije distributivne mreže, nivo njene automatizacije i broj tačaka koji je daljinski kontrolisan, raste i broj proračuna. Realizacija tih proračuna predstavlja najveće opterećenje za računarske resurse, a time ugrožava performanse savremenog DMS-a. Jedno od rešenja za ovaj problem jeste podale modela podatka mreže, u skladu sa njenom topološkom strukturom, u particije koje su nezavisne u pogledu proračuna [11]. Distributivna mreža ima radijalnu strukturu, tako da je ta podela moguća [24, 25]. Performanse sistema se mogu značajno unaprediti ako se ovako fragmentirani podaci obrađuju u paraleli.

Inicijalno kreiranje modela podataka (inicijalni unos statičkih podataka) čini niz aktivnosti koje se odvijaju pre nego što DMS omogući upravljanje distributivnom mrežom. Podaci o mreži se, velikim delom, sakupljaju iz drugih sistema kao sto su GIS, CIS, itd. [26, 27]. Pošto model podataka može da sadržati nekoliko miliona entiteta, sekvencijalni unos može da traje i do nekoliko desetina sati. U slučaju nevalidnog unosa podataka, proces mora bit ponovljen kako bi se obezbedili kvalitetni podaci i ispravan rad sistema. Iz tog razloga ažuriranje modela predstavlja jedan od kritičnih procesa i na njega je potrebno obratiti posebnu pažnju. Većina podataka koji pristižu iz drugih sistema su međusobno nezavisni, odnosno opisuju delove distributivne mreže koji nisu povezani. Shodno tome, postoji mogućnost paralelne obrade i unosa ovih podataka u DMS. Paralelizacijom ovog procesa značajno bi se uvećala iskorišćenost računarskih resursa i smanjilo vreme kreiranja inicijalnog modela.

Očigledno da su pred savremeni DMS kao glavni izazovi stavljeni obrada velike količine podataka i podrška širokom spektru funkcionalnosti (koja se ogleda u izvršenju različitih aktivnosti nad pomenutim podacima). Podrška opisanim zahtevima za posledicu može imati značajno smanjenje performansi sistema. Ovaj problem se može rešiti povećanjem kapaciteta računarskih resursa dostupnih savremenom DMS-u. Ali, umesto ovakvog jednostavnog, a skupog pristupa, rešenje se

može tražiti u primeni inteligentnog menadžmenta aktivnosti. Cilj inteligentnog menadžmenta aktivnosti je da se obezbedi optimalna upotreba računarskih resursa i na taj način postigne smanjenje njihovih troškova. Distribuirana arhitektura DMS-a se nameće kao logičan odgovor prilikom realizacije ovakvog pristupa. U tu svrhu je potrebno prilagoditi servise koji čine jezgro DMS softvera distribuiranom softverskom rešenju, kako bi svaki od njih u toku izvršenje bilo koje aktivnosti bio odgovoran za svoj deo podataka. Ovakva organizacija DMS-a je detaljno objašnjena u trećem poglavlju.

Usled složenosti posla koji obavlja, DMS treba da adekvatno odgovori na sledeće probleme po pitanju izvršavanja aktivnosti:

- aktivnosti koje se izvršavaju mogu biti međusobno zavisne (izlazni podaci jedne aktivnosti koriste se kao ulazni podaci neke druge aktivnosti),
- prioriteti određenih aktivnosti moraju da se uvaže (npr. obrada nekog alarma ima veći prioritet od ažuriranje modela distributivne mreže),
- vremenska ograničenja vezana za neku aktivnost moraju da se poštuju (npr. potrebno je završiti odgovarajući proračun do predefinisanog vremenskog trenutka, kako bi operater mogao odraditi pravilnu upravljačku akciju), i sl.

Kompleksnost problema raste ako se zna da skup svih aktivnosti nije unapred poznat (događaji koji ih definišu aktivnosti pristižu sporadično, u slučajnim vremenskim trenucima). Zato trenutni skup aktivnosti, postavljen pred ovakav sistem, evoluira tokom vremena.

Inteligentno DMS okruženje, suočeno sa ovako raznovrsnim situacijama, zahteva sledeće:

- razvoj algoritama koji će upravljati izvršavanjem različitih vrsta tokova aktivnosti uz prethodno opisana ograničenja.
- razvijeni algoritmi treba da uzmu u obzir karakteristike i specifičnosti distribuirane računarske arhitekture, ali i tokova aktivnosti koji se javljaju u DMS-u.

Doprinos ovog rada se ogleda u razvoju i implementaciji sistema za upravljanje tokovima aktivnosti u DMS-u. Rad ovakvog sistema se zasniva na upotrebi kreiranih algoritmima za koordinaciju izvršavanja tokova aktivnosti. Korisnost algoritama treba da bude potvrđena u situaciji kada je minimizacija perioda izvršenja svih DMS aktivnosti uzeta kao glavni pokazatelj performansi. U cilju verifikacije razvijenih algoritama, njihova upotreba je predstavljena kroz grupu primera. Svaki primer čini niz eksperimenata kojima je simuliran rad DMS-a pri različitim uslovima.

Nakon uvoda, u drugom poglavlju ovog rada uvedeni su i objašnjeni osnovni pojmovi koji se koriste u daljem tekstu. U istom poglavlju je dat prikaz najznačajnijih istraživanja na ovu temu koja se mogu naći u literaturi. Na kraju tog poglavlja je definisana hipoteza od koje se pošlo pri izradi ovog rada. U poglavljima tri i četiri detaljno su opisane distribuirana arhitektura i osnovni tokovi aktivnosti koji se javljaju u DMS-u. U petom poglavlju je formiran model svih zadataka koji se u jednom trenutku mogu postaviti pred DMS i objašnjena je evolucija tog modela kroz vreme. Osnovne komponente sistema za inteligentno upravljanje DMS tokovima aktivnosti prikazani su u šestom poglavlju. Sedmo i osmo poglavlje su od najvišeg interesa za ovaj rad. U njima su prikazani razvijeni algoritmi na osnovu kojih je realizovano upravljanje izvršavanjem DMS tokova aktivnosti. Razvoj ovih algoritama i njihova upotreba u okviru sistema za upravljanje tokovima aktivnosti predstavljaju svrhu istraživanja. U sedmom poglavlju je predstavljena upotreba centralizovane strategije upravljanja, dok je u osmom poglavlju opisan koncept zasnovan na kombinaciji centralizovane i

distribuirane strategije. U ta dva poglavlja predstavljeni su rezultati verifikacije razvijenih algoritama. Nakon zaključka i referentno navedene literature, u prvom prilogu je prikazan primer upotrebe distribuiranog algoritma raspodele, a u drugom reference kandidata.

2. PREDMET ISTRAŽIVANJA

Ovaj rad predstavlja rezultat višegodišnjeg istraživanja implementacije sistema za upravljanje tokovima aktivnosti koji se javljaju u DMS-u. U cilju što obuhvatnijeg sagledavanja tog problema, u radu su ilustrovana dosadašnja relevantna istraživanja i rešenja drugih autora. Polaznu osnovu za postavljanje hipoteze predstavljaju egzaktno definisani pojmovi koji su korišćeni prilikom razmatranja i rešavanja problema koordinacije izvršavanja tokova aktivnosti u DMS-u.

2.1 OSNOVNI POJMOVI I DOSADAŠNJI RAD U OBLASTI

1990-ih godina računarski *Grid* (*Grid computing*) je opisan kao distribuirana računarska infrastruktura koja se fokusira na deljenje resursa između različitih korisnika i izvršavanje inovativnih aplikacija sa orijentacijom prema uvek željenim visokim performansama [28]. Od tada je računarski *Grid* postao moćna platforma koja omogućava naučnicima i inženjerima širom sveta da pomeraju granice u svojim područjima istraživanja [29, 30]. Kao takav, on integriše distribuirane računarske resurse bez obzira na hardver i softver i tako postiže izuzetnu moć obrade. Najočigledniji primer je *Large Hadron Collider*, gigantski akcelerator u okviru Evropske organizacije za nuklearno istraživanje u Ženevi (CERN). To je jedan od najsloženijih i najambicioznijih projekata u svetu koji se oslanja na *Grid* tehnologiju, unutar kojeg se obrađuju izuzetno velike količine podataka. Konačno, rezultati takve obrade su dostupni brojnim naučnicima širom sveta.

Izvršavanje distribuiranih aplikacija na računarskom *Grid*-u je odavno prešlo iz akademskog u komercijalni domen. Različite vrste obrade podataka, koje omogućava računarski *Grid*, pružaju podršku za širok spektar zahtevnih aplikacija [31, 32]. Na primer, na ovom principu su kreirane distribuirane aplikacije vezane za biologiju [33], fiziku [34], astronomiju [35], medicinu [36], finansije [37], modelovanje klimatskih pojava [38], itd.

Istraživanje u ovom radu se prvenstveno bavi razvojem algoritama za koordinaciju aktivnosti koje izvršava distribuirani DMS. Iz tog razloga usvojena je definicija računarskog *Grid*-a data u literaturi [39]: Računarski *Grid* predstavlja tip distribuiranog sistema koji omogućava deljenje i izbor geografski distribuiranih, autonomnih i heterogenih računarskih resursa u cilju (eventualnog) paralelnog izvršavanja zadataka. Manipulaciju računarskim resursima je moguće izvršavati dinamički, u toku rada sistema, u zavisnosti od njihove dostupnosti i kapaciteta. Pri tome se u obzir uzimaju ograničenja nametnuta od strane korisnika koji zahteva izvršenje zadataka (prioritet zadataka, vremenska ograničenja i sl.).

U cilju lakšeg razumevanja teksta koji sledi, u nastavku su definisani najčešće korišćeni termini:

- *Zadatak* predstavlja atomsku jedinicu (nedeljiv deo posla) koja može biti dodeljena nekom resursu računarskog *Grid*-a na izvršenje. Dakle, zadatak se može definisati kao proces koji se dalje ne može deliti na jednostavnije korake. S obzirom da je zadatak nedeljiv deo posla, uvek se sprovodi u celini. Ukoliko iz nekog razloga dođe do greške u toku izvršavanja zadatka, sistem se vraća u stanje pre početka njegovog izvršenja.

- *Svojstva zadatka* predstavljaju parametri poput prioriteta zadatka, vremena izvršenja zadatka na pojedinim resursima računarskog *Grid*-a i sl. Koriste se pri primeni logike kojom se određuje dodela zadatka nekom resursu računarskog *Grid*-a.
- *Tok aktivnosti* (*workflow*) je skup (najčešće zavisnih) zadataka koji predstavljaju jednu celinu. Predviđeno je da se ovaj skup zadataka izvršava na odgovarajućim resursima računarskog *Grid*-a.
- *Kolekcija tokova aktivnosti* (*workflow collection*) opisuje sve tokove aktivnosti koji se u datom trenutku nalaze u sistemu.
- *Resurs računarskog Grid-a* omogućava da se izvrši određeni zadatak. Podrazumeva se da je svaki resurs računarskog *Grid*-a jedinstveno identifikovan, kao i da poseduje određeni kapacitet. U ovom radu se polazi od pretpostavke da resurs računarskog *Grid*-a u jednom trenutku ne može izvršavati više od jednog zadatka. Pojmovi *računarski resurs* ili *čvor računarskog Grid*-a predstavljaju sinonime terminu resurs računarskog *Grid*-a.

Očigledno je da se DMS može osloniti na računarski *Grid* radi skladištenja, obrade i analize velike količine distribuiranih podataka. Na osnovu zahteva koji se postavljaju pred DMS (uvod rada), jasno je da on mora postati izuzetno moćna distribuirana aplikacija visokih performansi. To znači da se mora osigurati adekvatna koordinacija izvršavanja tokova njegovih aktivnosti uz maksimalno iskorišćenje resursa računarskog *Grid*-a. S obzirom da se svaki tok aktivnosti može razložiti na jednostavnije korake, zadatke koji ga čine, neophodno je obezbediti efikasnu raspodelu zadataka na resurse računarskog *Grid*-a. Pored toga, pri raspodeli zadataka mora se uzeti u obzir veliki broj ograničenja: međusobna zavisnost zadataka, uspostavljanje redoslede izvršavanja zadataka po prioritetima, itd.

Proces raspodele zadataka preuzima niz zadataka i dodeljuje ih raspoloživim resursima računarskog *Grid*-a u cilju optimizacije različitih pokazatelja performansi DMS-a. U zavisnosti od strategije koja je primenjena, proces raspodele je usmeren ka različitim ciljevima: ujednačenost opterećenja računarskih resursa, minimizacija vremena izvršenja kolekcije tokova aktivnosti, minimizacija vremena izvršenja pojedinačnog toka aktivnosti i drugo. Pravilna raspodela zadataka ima značajan uticaj na performanse celokupnog sistema. Rešenja zasnovana na iscrpnim pretragama su nepraktična, jer vreme potrebno za generisanje različitih mogućih raspodela zadataka, može prevazići vreme koje je potrebno za njihovo izvršenje. Zato, u distribuiranom okruženju kao što je računarski *Grid*, odluke vezane za raspodelu zadataka treba doneti u najkraćem mogućem roku. Tako se povećava iskorišćenost resursa računarskog *Grid*-a i smanjuje ukupno vreme izvršenje zadataka, što rezultira sveukupnim uvećanjem performansi sistema.

Proces raspodele zadataka može se realizovati primenom različitih algoritama. U svetskoj literaturi može se naći veliki broj radova objavljenih na ovu temu, samo neki od njih su [40, 41, 42, 43]. Oni se mogu podeliti na različite načine, a za potrebe ovog rada su podeljeni na dva temeljno istražena tipa algoritama raspodele koji se zasnivaju na statičkoj [40, 41] i dinamičkoj [40, 43] raspodeli zadataka.

Statička raspodela se primenjuje kada je unapred poznat kompletan skup zadataka koje je potrebno izvršiti (kolekcija tokova aktivnosti je predodređena). Tada se procena iskorišćenja resursa računarskog *Grid*-a može odrediti pre stvarnog izvršenja zadataka. Računarski resursi, koji su zaduženi za pojedine zadatke, su određeni pre izvršenja bilo kog zadatka. Jedna od glavnih prednosti statičke raspodele je jednostavnija implementacija.

U ovom slučaju mapiranje zadataka na računarske resurse je određeno unapred, tako da je omogućen sveobuhvatni pogled na izvršavanje zadataka i opterećenje resursa računarskog *Grid*-a. Na osnovu ove analize može se dobiti odgovor na pitanja da li strategija raspodele zadataka treba da se zasniva na: 1) izvršenju zadataka na jednom ili više usko spregnutih računarskih resursu (kako bi se eliminisalo vreme potrebno za prenos podataka) ili je 2) povoljnija strategija u kojoj bi se tražili računarski resursi koji će pre izvršiti zadatke (ali će ukupno vreme izvršenja biti uvećano za vreme potrošeno na komunikaciju) [44]. Treba znati da procena ukupnog vremena izvršenja zadataka prilikom statičke raspodele nije imuna na situacije u kojima dolazi do greške u toku izvršavanja nekog zadatka. Moguće je da neki resurs računarskog *Grid*-a postane nedostupan u toku izvršavanja zadataka usled grešaka na računarskoj mreži ili je preopterećen internim aktivnostima. U takvim situacijama izvršenje njemu dodeljenog zadatka se ne završava u predviđenom roku i potrebno je doneti novi skup odluka vezanih za raspodelu zadataka [45].

Dnamička raspodela zadataka na resurse računarskog *Grid*-a se primenjuje kada skup zadataka koji su predviđeni za izvršenje nije unapred poznat [46]. Ona se obično realizuje i u slučajevima kada je teško proceniti vreme izvršenja zadataka, ili kada tokovi aktivnosti dinamički pristižu radi obrade [44, 47]. Postupak dinamičke raspodele zadataka se najčešće realizuje u dva koraka [48]. Prvi koraku predstavlja prikupljanje informacija vezanih za stanja resursa računarskog *Grid*-a. Na osnovu prikupljenih informacija, u drugom koraku se donose odluke o dodeli zadataka odgovarajućim računarskim resursima.

Prednost dinamičke raspodelu, u odnosu na statičku, predstavlja činjenica da proces zadužen za raspodelu zadataka uzima u obzir stanja resursa računskog *Grid*-a u toku rada i tako može da reaguje na različite okolnosti [49]. Ovakva raspodela je korisna u sistemima gde je primarni cilj maksimalno iskorišćenje resursa, umesto minimizacije vremena izvršenja pojedinačnih zadataka [50]. Prilikom realizacije dinamičke raspodele zadataka postoje dva pristupa: *online* and *batch-mode*. Ukoliko se upotrebljava *online* pristup zadaci se dodeljuju odgovarajućim računarskim resursima odmah pošto pristignu u sistem [43]. U toku *batch-mode* postupaka zadaci se ne šalju na izvršenje odmah nakon pristizanja. Umesto toga, više zadataka se prikuplja na gomilu (eng. *batch*). Prikupljeni zadaci se analiziraju i šalju na izvršenje u planirano vreme [43].

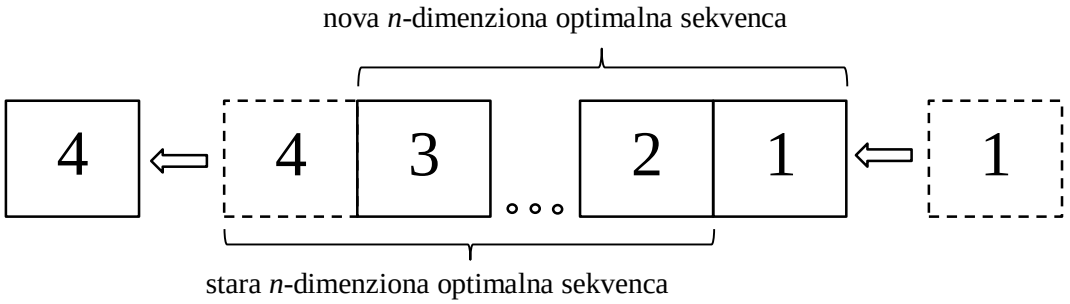
Očigledno da je implementacija dinamičke raspodele zadataka značajno složenija od statičke. Sa druge strane, u toku dinamičke raspodele prati se ponašanje računarskih resursa i moguće je adekvatno odgovoriti prilikom pojave nepredviđenih situacija (npr. nedostupnosti resursa računarskog *Grid*-a i sl.). Međutim, predložene su implementacije pomoćnih mehanizama, poput preraspodele zadataka [45] ili promene algoritma raspodele [51] u okviru statičke raspodele, da bi se ona unapredila i pružila zadovoljavajuće odgovore pri pojavi nepredviđenih situacija. Kada primarna statička raspodela ne daje odgovarajuće rezultate, pokreće se neki od pomoćnih mehanizama kako bi se preduzele odgovarajuće mere i sistem vratio u adekvatno stanje. Posledica uvođenja ovih pomoćnih mehanizama je da se razlika u implementaciji statičke i dinamičke raspodele zadataka dodatno smanjuje [52].

S obzirom da kolekcija tokova aktivnosti u DMS-u evoluira u vremenu (tokovi aktivnosti pristižu u slučajnim vremenskim trenucima i stoga se kolekcija neprekidno menja) evidentno je da mora biti realizovana dinamička raspodela zadataka.

Za rešavanje različitih problema u inženjerskoj praksi primena veštačke inteligencije predstavlja moderan pristup [53]. Tako je našla primenu u radu elektroenergetskih sistema [54], ali i kao alat za rešavanju problema inteligentne raspodele zadataka [55].

Jedan od osnovnih algoritama veštačke inteligencije, genetski algoritam, predstavlja robusnu, evolutivnu tehniku za pretragu velikog prostora. Klasična upotreba genetskog algoritma prilikom raspodele zadataka podrazumeva kreiranje populacije u kojoj svaka jedinka predstavlja jedno moguće rešenje. Svaka jedinka opisuje jednu sekvencu dodele zadataka resursima računarskog *Grid*-a. Populacija evoluira i bira se najbolje ocenjena jedinka kao optimalno rešenje. Jedan od glavnih nedostataka genetskog algoritma jeste da mu je potreban znatno duži vremenski period za generisanje rešenja u odnosu na neke druge tipove veštačke inteligencije (na primer veštačke neuronske mreže). Duži vremenski period je posledica iterativne obrade skupa jedinki kako bi se generisale nove populacije i pronašlo najbolje rešenje. Ovakva upotreba genetskog algoritma definiše statičku raspodelu zadataka pošto je neophodno da svi zadaci budu unapred poznati [56]. Stoga je neophodno izvršiti odgovarajuće korekcije upotrebe genetskog algoritma i prilagoditi ga problemima sličnim raspodeli DMS zadataka. Primena genetskog algoritma u rešavanju problema manipulacije zadacima prikazana je u [57, 58].

U radu [59] predstavljen je način prilagođenja genetskog algoritma potrebama dinamičke raspodele skupa zadataka koji se javljaju u *Smart Grid* sistemima. Prilikom pokretanja sistema, proces raspodele zadataka koristi genetski algoritam i već pristigle zadatke da bi odredio optimalnu sekvencu izvršavanja od *n* zadataka. Nakon što je prvi zadatak iz sekvence poslat na izvršenje, optimalna sekvenca sadrži *n*–1 zadatak. U narednom koraku raspodele zadataka dodaje se novi zadatak na kraj sekvence izvršavanja kako bi sekvenca ostala optimalna u datom trenutku. Na slici 2.1 je prikazano prethodno opisano kreiranje *n*-dimenzione optimalne sekvence zadataka.



Slika 2.1 – Upotreba genetskog algoritma pri raspodeli zadataka: n-dimenziona optimalna sekvenca

Prikazano prilagođenje genetskog algoritma za raspodelu zadataka omogućava dinamičku raspodelu zadataka i smanjuje vreme potrebno da se pronađe rešenje. Međutim, ovaj pristup uvodi niz pretpostavki koje se odnose na zadatke: zadaci moraju biti međusobno nezavisni, svi su istog prioriteta, a pretpostavljeni vremenski period izvršenja svakog od zadataka je isti.

Za raspodelu *Smart Grid* zadataka takođe su korišćene veštačke neuronske mreže [60]. U literaturi [61] je prikazana klasična upotreba višeslojnih neuronskih mreža kako bi se postigla adekvatna dodela zadataka resursima računarskog *Grid*-a. Unapređenje postupka se zasniva na uvođenju doobuke (adaptacije) višeslojne neuronske mreže. Adaptacija neuronske mreže se zasniva na evoluciji težinskih faktora da bi se adekvatno odgovorilo na promenljivo okruženje [62]. Hijerarhijska neuronska mreža takođe je našla svoju primenu u procesu raspodele zadataka [63]. Razvijena hijerarhijska neuronskih mreža se sastoji od dve veštačke neuronske mreže. Prva veštačka neuronska mreža predviđa buduće stanje resursa računarskog *Grid*-a na osnovu tekućeg stanja i tipova zadataka koji se trenutno izvršavaju. Druga veštačka neuronska mreža koristi izlaze iz prve i tekuća stanja resursa računarskog *Grid*-a kako bi se predvidelo vreme izvršavanja pojedinih tipova zadatka.

Upotreba veštačkih neuronskih mreže omogućava da se isključe pretpostavke da su izvršenja zadataka vremenski jednaka, dok se ograničenja vezana za ravnopravnosti (isti prioritet) i međusobnu nezavisnost zadataka ne menjaju. Očigledna su očekivanja da se razvije napredni mehanizam koji će moći da odgovori na zahteve DMS-a opisane u uvodnom izlaganju.

2.2 HIPOTEZA I CILJ ISTRAŽIVANJA

Inteligentno DMS okruženje je suočeno sa raznovrsnim situacijama i zahteva implementaciju sistema za upravljanje tokovima aktivnosti. Primena kompleksnih algoritama za koordinaciju izvršenja različitih vrsta DMS tokova aktivnosti omogućila bi da se paralelno izvršavaju zadaci i optimalno iskorišćavaju resursi računarskog *Grid*-a. Na taj način bi se značajno podigle performanse celokupnog DMS-a.

U skladu sa tim, definisana je hipoteza ove disertacije. Potrebno je razviti algoritme koji uzimaju u obzir karakteristike i specifičnosti računarskog *Grid*-a, ali i tokova aktivnosti u DMS-u. Ovi algoritmi treba da potvrde svoju korisnost u situaciji kada je kao glavni pokazatelj performansi uzeta minimizacija perioda izvršenja kolekcije tokova aktivnosti (*makespan* [64]). Period izvršenja kolekcije tokova aktivnosti (*makespan*) predstavlja vremenski interval između vremena početka obrade prvog zadatka i vremena završetka poslednjeg zadatka.

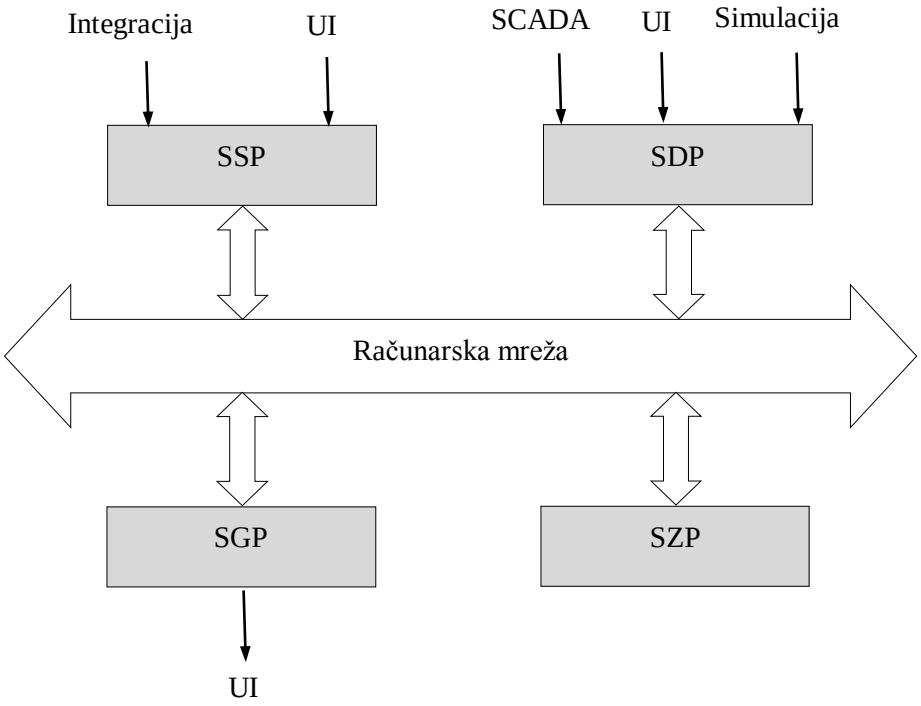
Cilj rada je da se kroz dva eksperimenta verifikuje efikasnost razvijenih algoritama za upravljanje tokovima aktivnosti. Prvi eksperiment se zasniva na inicijalnom unosu modela distributivne mreže u DMS. Drug eksperiment predstavlja simulaciju rada DMS-a u toku svakodnevnog upravljanja distributivnom mrežom.

3. VEZA DMS ARHITEKTURE I INFRASTRUKTURE RAČUNARSKOG GRID-a

Težište ovog poglavlja usmereno je ka distribuiranoj arhitekturi DMS-a. Predstavljeni su uloga i funkcionalnosti najbitnijih DMS komponenti, kao i njihov odnos prema infrastrukturi računarskog *Grid*-a.

S obzirom na značaj pouzdanog i efikasnog funkcionisanja DMS-a (budući da upravlja snabdevanjem električnom energijom miliona korisnika među kojima su neki od izuzetnog značaja, poput bolnica, škola, univerziteta, fabričkih postrojenja kod kojih bi nestanak električne energije ugrozio živote ljudi i/ili izazvao velike finansijske gubitke), instalacija DMS-a se vrši na namenskom računarskom *Grid*-u. To znači da se računarski resursi ne dele ni sa jednom drugom (distribuiranom) aplikacijom i u potpunosti su na raspolaganju DMS-u. Ukoliko bi resursi računarskog *Grid*-a bili deljeni, DMS bi, u toku rada, morao da identifikuje i obrađuje dinamičke informacije o njihovoj dostupnosti. Pored toga, on bi se dodatno "borio" za zauzeće resursa računarskog *Grid*-a koje koriste i druge distribuirane aplikacije [65, 66, 67]. Međutim, DMS je oslobođen takve "borbe", pa može u potpunosti da se usresredi na realizaciju svojih aktivnosti. Konstantna dostupnost računarskih resursa i mogućnost da se operativno vreme u celosti potroši na izvršavanje aktivnosti koje se odnose na upravljanje distributivnom mrežom, značajno unapređuju performanse sistema, pa je sama investicija u namenski računarski *Grid* potpuno opravdana.

DMS je izgrađen kao kolekcija servisa prilagođenih za rad u distribuiranom računarskom okruženju. Da bi DMS korektno funkcionisao, neophodno je da servisi međusobno sarađuju i razmenjuju informacije. Najznačajniji servisi DMS softvera i njihova interakcija sa spoljašnjim okruženjem prikazani su na slici 3.1.



Slika 3.1 – DMS servisi i infrastruktura računarskog Grid-a

Svaki od servisa implementira specifičnu funkcionalnost, a pri tom čuva podatke koji će mu osigurati delotvoran, siguran i neprekidan rad:.

- *Servis statičkih podataka* (SSP) se nalazi na dnu hijerarhije servisa koji se javljaju u okviru DMS-a. U osnovi ovog servisa je model podataka distributivne mreže usaglašen sa CIM standardom [13, 14]. Model je kreiran na osnovu predefinisanog CIM profila, uz mogućnost proširenja dodatnim tipovima podataka. Na taj način je u omogućeno da se distributivna mreža u potpunosti modeluje (parametri uređaja i njihova međusobno povezanost, konektivnost u normalnom uklopnom stanju). Osnovna namena SSP-a je da ostalim komponentama u sistemu obezbedi pristup ovom modelu kroz odgovarajuće interfejse. Zbog toga je neophodno da svaka izmena podataka modela distributivne mreže ide preko ovog servisa. Unos podataka se obezbeđuje iz dva izvora. Prvi izvor informacija predstavlja integracija sa drugim sistemima, tipa GIS [26, 27], dok je u drugom slučaju izmena modela omogućena direktno, upotrebom klijentske aplikacije, od strane korisnika koji ima odgovarajuće privilegije. Nakon inicijalnog unosa podataka, izmene modela su veoma retke, pa se iz tog razloga njegovi podaci tretiraju kao statički podaci. Naknadno ažuriranje modela se vrši kada dođe do izmena u izvornom sistemu ili na zahtev korisnika. Da bi mogao da čuva modelske podatke, SSP se najčešće oslanja na relacionu bazu. Zbog toga se SSP nalazi na čvoru računarskog *Grid*-a na kome je instaliran sistem za upravljanje relacionom bazom podataka.
- *Servis dinamičkih podataka* (SDP) sakuplja podatke o izmerenim veličinama i statusima uređaja distributivne mreže (pozicije prekidača, izmerene vrednosti napona, jačine struje, itd.). Ovi dinamički podaci se u DMS okruženje prosleđuju automatski iz SCADA sistema, ali i ručno od strane operatera upotrebom klijentske aplikacije. U toku upravljanja distributivnom mrežom, dovoljno velike promene izmerenih vrednosti i promene statusa rasklopne opreme i regulacionih resursa predstavljaju okidače za pokretanje analitičkih DMS funkcija. S obzirom da je frekvencija izmena velika, neophodno je obezbediti efikasno izvršavanje analitičkih DMS funkcija. Pošto je u okviru DMS-a omogućena simulacija rada distributivne mreže za različite režime, postoji i mogućnost učitavanja unapred pripremljenog skupa kojim su ti režimi definisani.
- *Servis grafičkih podataka* (SGP) je odgovoran za održavanje grafičkih objekata koji se koriste prilikom prikaza distributivne mreže na korisničkom interfejsu (*user interface* – UI). Servis obezbeđuje interfejse koji nude metode za čitanje ovih podataka od strane drugih komponenti. U skladu sa tim, svakoj korisničkoj UI aplikaciji omogućeno je da u svojoj memoriji skladišti samo podatke neophodne za prikaz trenutnog pogleda na distributivnu mrežu. Kada korisnik želi da izmeni svoj pogled na mrežu, može da se osloni na SGP radi pribavljanja dodatno potrebnih grafičkih podataka. Na ovaj način, obezbeđen je minimum zahteva za računarskim resursima koji je potreban klijentskim aplikacijama.
- *Servis za proračune* (SZP) je najopterećeniji u toku rada sistema. Servis obezbeđuje sredstva za vršenje proračuna neophodnih analitičkim elektroenergetskim funkcijama. Prihvata statičke i dinamičke podatke od interesa za deo distributivne mreže i kreira novi prikaz mreže, zasnovan na realnim, trenutno aktuelnim statusima uređaja. U suštini, servis povezuje i kombinuje informacije sa SSP-a i SDP-a i na taj način analizira povezanost uređaja na osnovu izmerenih veličina iz polja, umesto na osnovu normalnog uklopnog stanja. Ovaj proces se naziva analiza topologije. Nakon određivanja tačne topologije distributivne mreže moguće je realizovati zahtevane proračune. Podaci koji se prihvataju sa SSP-a i SDP-a mogu se razlikovati u zavisnosti od vrste proračuna koji je neophodan za izvršenje analitičke DMS funkcije. Kako se od servisa zahtevaju komplikovani

proračuni nad velikom količinom podataka, SZP je postavljen na čvor sa moćnom centralnom procesorskom jedinicom, tzv. čvor za proračune.

Svaki od navedenih DMS servisa je raspoređen na poseban čvor (resurs) računarskog *Grid*-a koji poseduje potrebne karakteristike. Da bi započela realizacija bilo kog zadatka neophodno je da svi potrebni ulazni podaci budu skladišteni na čvoru na kome će taj zadatak biti izvršen. U prethodnom tekstu naglašena je potreba DMS-a da obrađuje i manipuliše ogromnom količinom podataka. U skladu sa tim poruke koje se razmenjuju između DMS servisa (a time i čvorova računarskog *Grid*-a) mogu biti velikih obima. U distribuiranim aplikacijama koje karakteriše intenzivan prenos velike količine podataka, neophodno je da infrastruktura računarskog *Grid*-a obezbedi ne samo sigurnost podataka, već i pouzdan transfer visokih performansi [68]. Pored hardverskog dela koga čine moćni, namenski računarski resursi (snažni procesori, računarska mreža visokog propusnog opsega i slično), ono što posebno odlikuje infrastrukturu računarskog *Grid*-a je mehanizam za upravljanje podacima [69].

U distribuiranim okruženjima, posebno u aplikacijama kao što je DMS, koje direktno utiču na život stanovništva, pristup podacima treba da bude strogo kontrolisan, odnosno, potrebno je obezbediti njihovu sigurnost [70, 71]. Samim tim, jedno od glavnih zaduženja mehanizma za upravljanje podacima jeste da se onemogući neautorizovan pristup podacima. U suprotnom, neovlašćeni pristup podacima može izazvati različite posledice, npr.: njihovo neovlašćeno objavljivanje ili upotrebu, njihovo zadržavanje čime postaju nedostupni ovlašćenim korisnicima ili aplikacijama, izmena podataka ili čak uništenje čime se gubi njihova verodostojnosti i celovitost. S obzirom da kvalitetni, raspoloživi podaci predstavljaju osnovu za rad svakog sistema, njihova nevalidnost i/ili nekorektnost mogu imati različite posledice kao što su narušavanje kritičnih funkcionalnosti celokupnog sistema, sprečavanje delotvorne kontrole sistema, pa čak i prekid njegovog rada. Zato, mehanizam za upravljanje podacima mora da obezbedi njihovu sigurnost.

Sigurnost podataka je osnovni uslov prilikom njihove razmene između čvorova računarskog *Grid*-a. Nepostojanje enkripcije podataka koji se razmenjuju daje mogućnost njihovog čitanja u toku transfera kroz računarsku mrežu. Stoga je prenos kriptovanih podataka neminovnost. Pri tome, neophodno je da i rukovanje skladištenim podacima bude ostvareno tako da se osigura njihova zaštita od pristupa neautorizovane aplikacije ili korisnika [72, 73]. Da bi se sačuvala privatnost skladištenih podataka neophodno je da mehanizam za njihovo upravljanje, koji predstavlja deo infrastrukture računarskog *Grid*-a, obezbedi enkripciju tih podataka putem upotrebe predefinisanog, privatnog ključa. Osim toga, u situacijama kada se javljaju zahtevi za skladištenjem novih podataka, nophodan je dokaz autentičnosti i autorizacija korisnika koji je poslao zahtev kako maliciozni podaci ne bi postali deo skladištenih.

Drugi veliki zadatak postavljen pred mehanizam za upravljanje podacima je velika količina podataka koje je potrebno preneti putem računarske mreže. To znači da postoji suštinski zahtev za efikasnim prenosom podataka između različitih lokacija u toku intenzivnog rada DMS-a. U tu svrhu mehanizam za upravljanje podacima koristi razvijene protokole koji obezbeđuju visoke performanse prilikom prenosa [74, 75] .

Protokoli za transfer podataka u računarskom *Grid*-u su kreirani tako da obezbede prethodno opisanu, neophodnu funkcionalnost [74]. Podrška efikasnom i sigurnom prenosu podataka postiže se nadogradnjom postojećih protokola dodatnim funkcijama kao što su: dimenzionisanje TCP (*Transmission Control Protocol*) bafera; paralelan prenos podataka upotrebom višestrukih TCP tokova kako bi se povećao protok podataka u odnosu na upotrebu jednog TCP toka; funkcionisanje fleksibilnog i robusnog mehanizma za dokaz autentičnosti; očuvanje integriteta podataka i zaštitu od neovlašćenog pristupa podacima koji se prenose [75].

Upotreba odgovarajućeg modela komunikacije između čvora računarskog *Grid*-a je ključna kako

prenos podataka ne bi predstavljao usko grlo sistema. Najčešći modeli komunikacije su: centralizovani, posrednički i *peer-to-peer* [76].

Primenom *centralizovanog modela* za prenos podatka između dva čvora računarskog *Grid*-a koristi se jedna centralna tačka. Implementacija može biti zasnovana na ideji da centralna jedinica, koja manipuliše raspodelom zadataka na čvorove, prihvata podatke koji su izlaz upravo završenog zadataka i prosleđuje ih kao ulazne podatke na drugi čvor, na kome treba da se izvrši naredni zadatak. Ovakav model je jednostavan za implementaciju, ali nije dovoljno efikasan za probleme gde se zahteva prenos velike količine podataka, što je upravo slučaj sa DMS-om.

Posrednički pristup, za razliku od centralizovanog, nema centralnu jedinicu, već je manipulacija međupodacima (podaci koji predstavljaju izlaz jednog zadataka, a ulaz drugog zadataka) implementirana u okviru mehanizma za upravljanje podacima [77]. Ovaj model komunikacije pogodan je u distribuiranim aplikacija u kojim korisnici žele da nadgledaju međurezultate.

Primenom *peer-to-peer* modela prenos podataka se realizuje direktno od izvornog čvora do odredišnog čvora, bez uključivanja posrednika [78, 79, 80]. Time je značajno smanjeno vreme koje je potrebno za prenos velike količine podataka unutar distribuirane aplikacije. Dakle, da bi se osiguralo korektno funkcionisanje DMS-a neophodna je upotreba ovakvog model komunikacije. Da bi se podržala *peer-to-peer* komunikacija, čvorovi računarskog *Grid*-a moraju, pored obrade podataka, da obezbede i potrebnu funkcionalnost za njihovu razmenu. Svaki čvor može da dobije ulogu i klijenta i servisa, u zavisnosti od potrebe. Kao klijent može da zahteva neku uslugu od preostalih čvorova, dok kao servis treba da pruži usluge drugima.

Prethodno opisani mehanizmi predstavljaju osnovni deo infrastrukture računarskog *Grid*-a i kao takvi treba da su implementirani na takav način da imaju minimalni uticaj na logiku i funkcionalnost DMS servisa koji su raspoređeni po čvorovima računarskog *Grid*-a. Treba naglasiti da, najčešće, čvorovi imaju svoje replike. Tada, ukoliko dođe do greške na nekom čvoru, njegova replika preuzima posao kako celokupno funkcionisanje DMS-a ne bi trpelo posledice. Usled važnosti pouzdanog funkcionisanje DMS-a, investicije u replike su u potpunosti opravdane.

4. DMS TOKOVI AKTIVNOSTI

U ovom delu analizirani su tokovi aktivnosti koje DMS izvršava, u cilju da se njihove karakteristike i specifičnosti iskoriste u procesu raspodele zadataka.

U prethodnom poglavlju naglašeno je da zadatak predstavlja atomsku, nedeljivu celinu koja se izvršava na određenom resursu računarskog *Grid*-a, a da je tok aktivnosti skup zavisnih zadataka. Struktura tokova aktivnosti ukazuje na zavisnost zadataka koji su njime obuhvaćeni [81, 82]. Ukoliko je neki zadatak u potpunosti nezavisan od drugih zadatka, modeluje se kao tok aktivnosti od jednog zadatka. Fundamentalni cilj uvođenja ovakvog predstavljanja tokova aktivnosti je da se na istaknut i razumljiv način prikažu osnovni zahtevi koji se javljaju tokom njihovog modelovanja. U tu svrhu potrebno je opisati zadatke i redosled njihovog izvršavanja, koliko god da je redosled složen.

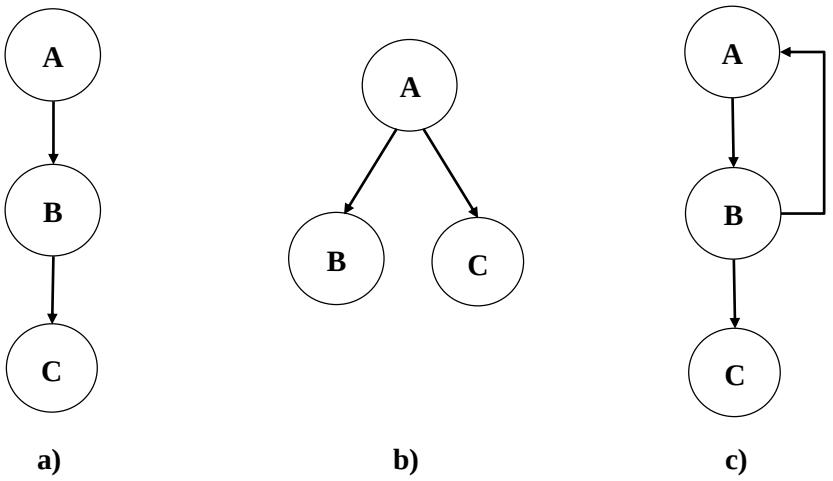
4.1 OSNOVNE STRUKTURE TOKOVA AKTIVNOSTI

Generalno mogu se izdvojiti dva osnovna tipa strukture tokova aktivnosti:

- struktura koja opisuje tokove aktivnosti za koje je predviđeno izvršenje svakog zadatka unutar toka aktivnosti,
- struktura koja podržava selektivan izbor izvršavanja zadataka koji čine tok aktivnosti.

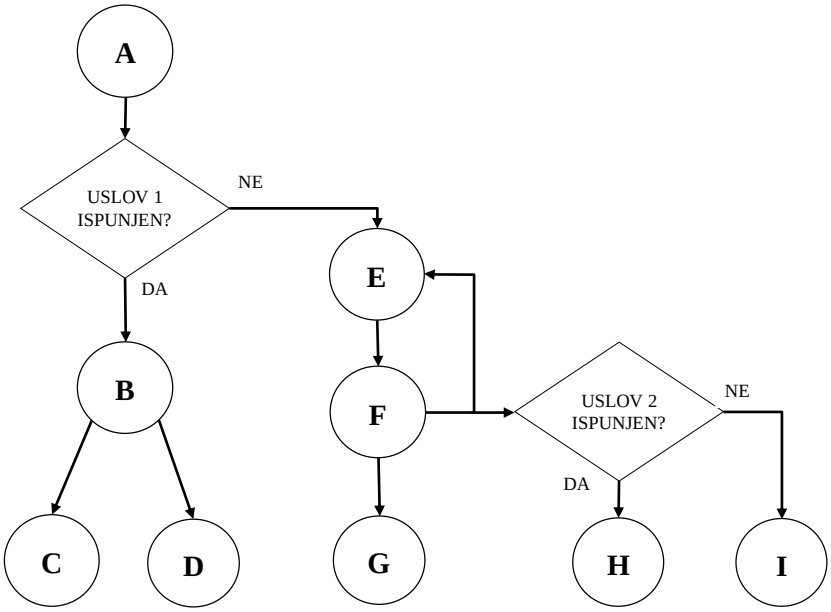
Prvi tip strukture je jednostavniji. On nameće pristup da je neophodno uspešno izvršiti svaki zadatak kako bi se neki tok aktivnosti mogao smatrati završenim. U zavisnosti od složenosti, on se dalje deli na odgovarajuće podtipove [83]:

- Linearna struktura opisuje najjednostavniju, bazičnu klasu toka aktivnosti. Predstavlja niz zadataka koji se moraju izvršavati u određenom sekvencijalnom redosledu. Prvi zadatak preuzima inicijalne podatke i u toku izvršenja transformiše ih u podatke koji predstavljaju ulaz narednog zadatka. Ovaj proces se nastavlja sve dok se ne izvrši i poslednji zadatak u nizu. Primer linearnog toka aktivnosti koji se sastoji od tri zadatka: A, B i C prikazan je na slici 4.1a.
- Usmerena aciklična struktura opisuje sledeći nivo složenosti tokova aktivnosti. U ovom slučaju tok aktivnosti može biti predstavljen kao usmereni aciklični graf [84, 85]. Čvorovi usmerenog acikličnog grafa predstavljaju zadatke, dok grane opisuju zavisnosti zadataka uslovljene razmenom podataka, slika 4.1b. Za ovaj podtip, zadaci se izvršavaju delimično sekvencijalno, a delimično konkurentno (moguće je njihovo izvršavanje u paraleli). Na primer, neki zadaci zavise od izvršenja nekolicine drugih zadataka koji se mogu izvršavati istovremeno.
- Umerena ciklična struktura služi kako bi se modelovala klasa tokova aktivnosti koja je kompleksnija od prethodno opisane usmerene aciklične strukture. Ovakva struktura podržava mogućnost da se jedan segment toka aktivnosti (zadatak ili grupa zadataka) iterativno ponavlja kao deo petlje. Umerena ciklična struktura se često upotrebljava u naučnim aplikacijama gde se jedan ili više zadataka neprekidno ponavljaju [86, 87]. Primer toka aktivnosti koji pripada ovoj klasi je prikazan na slici 4.1c.



Slika 4.1 – Strukture tokova aktivnosti bez selektivnog izbora zadataka: a) linearna, b) usmerena aciklična, c) usmerena ciklična

Drugi tip strukture definiše tokove aktivnosti za koje se ne zahteva izvršenje svih zadataka da bi oni bili označeni kao završeni. U zavisnosti od rezultata obrade nekog zadatka moguće je izabrati koji zadatak ili grupa zadataka treba da se izvrše sledeći, odnosno dozvoljeno je modelovati uslovno grananje. Ovakva složenija struktura može da sadrži bilo koju grupu zadataka kao direktnu ili uslovnu granu toka aktivnosti koji se modeluje. Pored toga, grupa zadataka koja čini odgovarajuću granu se predstavlja jednim od dva osnovna tipa strukture, slika 4.2.



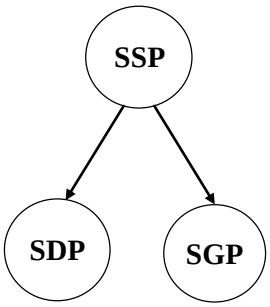
Slika 4.2 – Struktura tokova aktivnosti sa uslovnim grananjem

U literaturi se mogu naći različite tehnike za modelovanje ovako kompleksnih tokova aktivnosti [88, 89, 90, 91, 92]. Petrijeve mreže su jedan od najpopularnijih pristupa za opis sekvencijalnog, paralelnog, iterativnog i uslovnog izvršavanja zadataka [88, 89, 90]. Pored Petrijevih mreža, procesna algebra je omiljena tehnika koja se upotrebljava za opis ovakve, kompleksnije strukture tokova aktivnosti [91, 92]. Potrebno je naglasiti sistem za upravljanje tokovima aktivnosti mora da podrži manipulacije opisanim složenim tokovima aktivnosti [88, 93, 94].

4.2 OSNOVNI TOKOVI AKTIVNOSTI U DMS-u

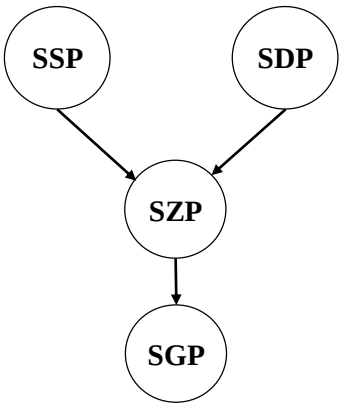
U uvodnom delu su opisani osnovni zahtevi koji su postavljeni pred DMS: izvršenje analitičkih elektroenergetskih funkcija, ažuriranje modela distributivne mreže, dinamička vizualizacija mreže, itd., uz sve prisutnu manipulaciju velikom količinom podataka. U trećem poglavlju je analizirana distribuirana arhitektura DMS-a i interakcija DMS servisa sa infrastrukturom računarskog *Grid*-a. Na osnovu razmotrenih činjenica utvrđeno je da je u DMS-u moguće definisati tokove aktivnosti čije izvršenje ima presudan uticaj na funkcionisanje čitavog sistema. Osnovne funkcionalnosti DMS-a su opisane sledećim bazičnim tipovima tokova aktivnosti:

- *Ažuriranje modela* distributivne mreže predstavlja tok aktivnosti koji DMS treba da izvrši kada stigne zahtev za izmenom statičkih podataka. SSP čuva trenutno stanje modela distributivne mreže. Ažuriranje modela je zasnovano na inkrementalnoj promeni podataka koji definišu navedeno trenutno (a za ovaj tok aktivnosti početno) stanje modela. Inkrementalna promena može da sadrži veliki broj operacija koje definišu izmene na nivou entiteta modela. Operacije se kao jedinstvena transakcija primenjuju nad modelom po unapred definisanom redosledu. Ako bilo koja operacija ne uspe, kompletan zahtev za izmenom modela se odbacuje. Dodatno, ukoliko su uspešno primenjene sve operacije, novonastalo stanje modela se validira predefinisanim skupom pravila vezanih za konektivnost, opseg odgovarajućih parametara i sl. Ako su sva validaciona pravila zadovoljena, izmene modela su prihvaćene. U suprotnom, ako je neko validaciono pravilo narušeno, sve izmene nastale primenom operacija se odbacuju. Kada je inkrementalna promena u potpunosti primenjena i potvrđena skupom validacionih pravila, model mreže prelazi u novo stanje. Od tog momenta, SSP stavlja na raspolaganje novi model distributivne mreže svim zainteresovanim DMS komponentama. Na osnovu izvršenih modifikacija statičkih podataka, neophodno je ažurirati grafičke i dinamičke podatke. Operacije su definisane na nivou entiteta, te je potrebno ukloniti grafičke objekte koji su vezani za entitete izbrisane iz modela, dodati novu grafičku predstavu za entitete koji su upravo uključeni u model i ažurirati grafičku predstavu izmenjenih entiteta. Na sličan način je potrebno ažurirati i dinamičke podatke, tako što se u zavisnosti od izmena statičkih podataka dodaju ili uklone definicije merenih veličina koje su vezane za pojedine uređaje. Tako na primer, ukoliko je iz modela uklonjen jedan prekidač, iz dinamičkih podatka na SDP-u uklanjaju se definicije njegovih signala (najčešće vezane za njegov status). Ovaj tip toka aktivnosti je uobičajan tokom inicijalnog unosa podataka, odnosno pre nego što upravljanje distributivnom mrežom počne. Znatno se ređe javlja nakon toga, tj. u toku svakodnevnog funkcionisanja sistema. Analizom ovog tipa toka aktivnosti može se zaključiti da je prvo potrebno izvršiti izmene podataka na SSP-u. Nakon što je ovaj zadatak završen, prosleđuju se potrebne informacije do SGP-a i SDP-a, koji svoje zadatke mogu izvršavati istovremeno. Očigledno je da se ovakav tip toka aktivnosti modeluje usmerenim acikličnim grafom, slika 4.3. Radi jasnijeg prikaza, u ovom delu teksta, pre uvođenja formalnog prikaza, svaki zadatak koji predstavlja deo nekog toka aktivnosti je označen DMS servisom na kome se izvršava.



Slika 4.3 – Ažuriranje modela

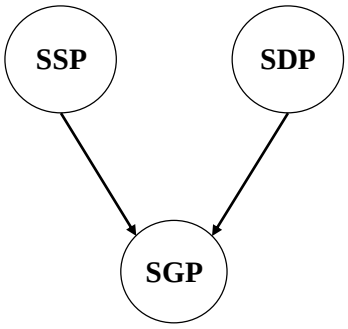
- *Izvršenje DMS funkcije* predstavlja izvršenje analitičke elektroenergetske funkcije. Ovaj tip toka aktivnosti je uobičajeno izazvan izmenom vrednosti neke izmerene veličine. Kada se trenutno stanje distributivne mreže izmeni (promeni se status prekidačke opreme, vrednost električne veličine, itd.), ta informacija stiže u DMS. Ona u DMS može da pristigne automatski sa SCADA sistema ili da bude ručno uneta od strane operatera upotrebom klijentske aplikacije. Nakon što su izmene uvažene na SDP-u, deo podataka dinamičkog modela i korespodentni deo podataka statičkog modela prenosi se na SZP. Zahvaljujući radijalnoj strukturi distributivne mreže, model statičkih podatka na SSP-u je izdeljen na particije koje su nezavisne u pogledu proračuna [24, 25]. Podaci se prikupljaju u svim particijama koje obuhvataju opremu na koju je uticala izmenjena merena veličina. Prikupljeni podaci se šalju na SZP. Istovremeno, na istu adresu i SDP šalje odgovarajuće, korespodentne dinamičke podatke. Nakon što je prihvatio podatke sa oba DMS servisa (SSP i SDP) SZP može da obavi proračune. Rezultat se prosleđuje u SGP kako bi se osvežili grafički podaci, a samim tim i prikaz stanja distributivne mreže u kontrolnom centru. Nakon toga, operater je upoznat sa novim stanjem mreže i u mogućnosti je da, po potrebi, reaguje (generiše upravljačke akcije). Ovaj tip toka aktivnosti je najčešći i javlja se u toku svakodnevnog nadzora i upravljanja distributivnom mrežom. Slično kao i za prethodni tip toka aktivnosti, Ažuriranje modela, i ovaj tip toka aktivnosti, Izvršenje DMS funkcije, modeluje se primenom usmerenog acikličnog grafa, slika 4.4.



Slika 4.4 – Izvršenje DMS funkcije

- *Osvežavanje grafičkog prikaza* je tip toka aktivnosti koji se najređe izvršava. Obično se izvršava periodično, kao potencijalni oporavak SGP-a od greške ili na zahtev korisnika. Ukoliko dođe do prekida komunikacije između SGP i ostalih DMS servisa, nakon ponovne uspostave komunikacije, pokreće se ovaj tok aktivnosti kako bi se grafički podaci ažurirali na osnovu najnovijih informacija. Takođe, moguće je da klijentska aplikacija izgubi konekciju prema SGP-u. Tada klijentska aplikacija inicira osvežavanje grafičkog prikaza pogleda koji je predstavljen korisniku (najčešće operateru).

Obzirom da je neprihvatljivo da operater ostane bez uvida u radni režim distributivne mreže na duži vremenski period, potrebno je osvežiti tekući prikaz što pre. Iz tog razloga, SGP prihvata samo podatke koji su već dostupni na ostalim DMS servisima, bez dodatnih, vremenski zahtevnih proračuna. Odnosno, da bi se ažurirali grafički podaci, na SGP se prenosi samo predodređeni deo statičkih podataka i njemu korespodentni deo dinamičkih podataka (izmerene vrednosti). I ovaj tip toka aktivnosti, Osvežavanje grafičkog prikaza, može se modelovati usmerenim acikličnim grafom, slika 4.5.



Slika 4.5 – Osvežavanje grafičkog prikaza

Analizom najbitnijih tokova aktivnosti koji se javljaju u DMS-u može se uočiti sledeće:

- Ne postoji uslovno grananje na osnovu rezultata obrade nekog zadatka (ne postoji potreba primene modela Petrijeve mreže), niti tok aktivnosti koji treba opisati primenom ciklične strukture. Sa druge strane, postoji mogućnost paralelnog izvršavanja zadataka. Stoga se zaključuje da je svaki tok aktivnosti moguće prikazati usmerenim acikličnim grafom. Ulazni zadaci nekog toka aktivnosti nemaju prethodnike, dok izlazni nemaju ni jednog sledbenika. Podaci koji predstavljaju rezultat izvršenja izlaznih zadataka su ujedno i rezultati samog toka aktivnosti.
- Ukoliko se razmatra kompleksnost predstavljenih DMS tokova aktivnosti sa stanovišta zavisnosti koja postoji između zadataka, ni jedan tip toka aktivnosti nema više od tri nivoa zadataka, pri čemu su zadaci u okviru istog nivoa međusobno nezavisni. Dodatno, čvorovi usmerenih acikličnih grafova koji predstavljaju zadatke imaju najviše dve grane usmerene ka ostalim čvorovima.

Procena o tome koliko opterećenje računarskih resursa neki zadatak da unosi u sistem dobija se na osnovu istorijskih podataka [95, 96, 97]. Kapaciteti računarskih resursa se određuju na osnovu njihovih karakteristika, pa se stoga, procena vremena izvršenja zadatka jednostavno računa kao količnik opterećenja koje donosi zadatak i kapaciteta računarskog resursa.

U distribuiranim sistemima velikih razmera neki tokovi aktivnosti mogu biti od većeg značaja u odnosu na ostale. U tim slučajevima njihovo izvršavanje ima prednosti, nezavisno od optimizacije iskorišćenja resursa računarskog *Grid*-a. Kako bi sistem za upravljanje tokovima aktivnosti bio sposoban da obradi ovakve situacije, neophodno je uvesti rangiranje tokova aktivnosti na osnovu prioriteta [98, 99]. Odnosno, potrebno je da se tokovima aktivnosti od većeg značaja dodele viši prioriteti.

Na identičan način funkcioniše i inteligentno DMS okruženje. Izvršavanje prethodno opisanih DMS tokova aktivnosti inicira korisnik ili neki događaj u sistemu. Rangiranje po prioritetu tokova aktivnosti, koji su pristigli radi izvršenja, najčešće je implementirano u samom DMS-u. Na osnovu formirane liste prioriteta proces raspodele zadatala (ne)favorizuje tokove aktivnosti. Na primer, moguće je da su u DMS istovremeno pristigla dva obaveštenja: 1) iz GIS podsistema, informacija da je došlo do izmena vezanih za model statičkih podataka i 2) sa SCADA sistema, informacija o značajnoj promeni vrednosti napona na čvoru

distributivne mreže. Ovim događajima inicirana su dva toka aktivnosti: Ažuriranje modela podataka i Izvršenje DMS funkcije. Imajući u vidu da velika promena vrednosti napona može kritično da utiče na stabilnost sistema za distribuciju električne energije, neophodna je što hitnija estimacija stanja mreže da bi operater bio obavešten u što kraćem roku i da bi se, eventualno, generisali odgovarajući alarmi. Shodno tome, DMS okruženje obeležava Izvršenje DMS funkcije kao prioritetniji tok aktivnosti.

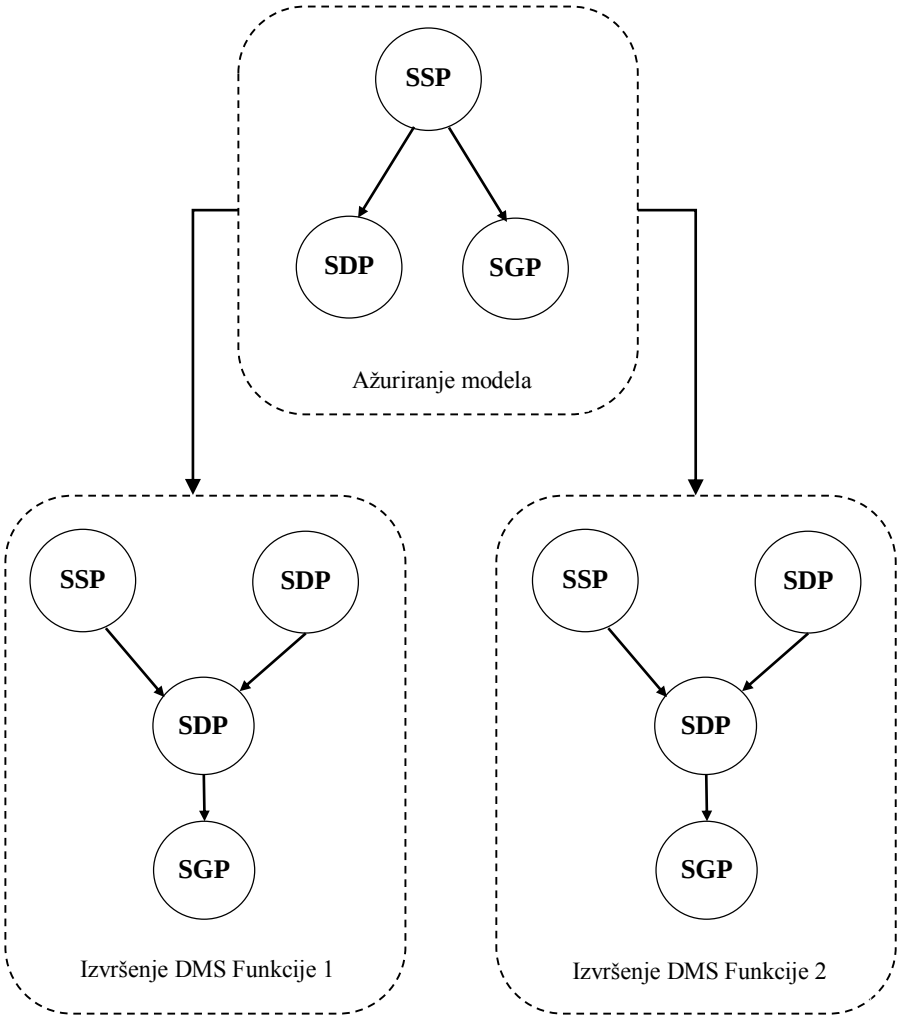
Postoje situacije kada korisnik sam dodeljuje prioritet nekom toku aktivnosti. Na primer, kada korisnik radi na simulacijama rada distributivne mreže pri različitim okolnostima (obavlja "šta-ako" analize za različite grupe vrednosti procesnih veličina) i želi prikaz više simuliranih režima (npr. za svaku grupu vrednosti procesnih veličina). Tada se iniciraju tokovi aktivnosti tipa Izvršenje DMS funkcije za svaku simulaciju. Korisnik može da označi koje simulacije su mu bitnije i tako posredno dodeli prioritet određenim tokovima aktivnosti.

S obzirom da je definisan prioritet tokova aktivnosti, taj prioritet se dodeljuje i zadacima koji ih čine. Dakle, zadacima koje neki tok aktivnosti grupiše u celinu postavlja se identičan prioritet koji je dodeljen i samom toku aktivnosti. Na taj način prvo se obrađuju zadaci većeg prioriteta, pa samim tim i tok aktivnosti kome pripadaju.

4.3 ZAVISNI TOKOVI AKTIVNOSTI U DMS-u

Daljom analizom rada DMS-a, uočava se da postoji međusobna zavisnost između tokova aktivnosti koji pristižu u sistem. To znači da je neophodno da se pojedini tokovi aktivnosti izvrše u celini kako bi neki drugi krenuli sa izvršenjem. U skladu sa tim, svaki tok aktivnosti može da ima druge tokove aktivnosti kao svoje prethodnike i sledbenike. Sistem za upravljanje tokovima aktivnosti može da podrži različite algoritme za raspodelu zadataka, pa je neophodno da svaki od njih uzme u obzir ograničenja koja su nametnuta međusobnom zavisnošću pojedinih tokova aktivnosti.

Klasičan primer zavisnih tokova aktivnosti predstavlja grupa tokova aktivnosti koja se generiše kada je potrebno uneti novi deo distributivne mreže i prikazati ga operatoru. Npr. ukoliko je potrebno uneti novi izvod i sve entitete koji sadrži, unose se statički podaci. Da bi operater dobio korektnu sliku o stanju mreže, nakon unosa statičkih podataka neophodno je, za aktuelne dinamičke podatke, izvršiti analitičke funkcije. Na slici 4.6 je prikazana zavisnost dva toka aktivnosti tipa Izvršenje DMS funkcije od toka aktivnosti tipa Ažuriranje modela.



Slika 4.6 – Zavisnosti između različitih tokova aktivnosti

Na osnovu dosadašnjih razmatranja može se reći da su osnovne karakteristike svakog zadatka u distribuiranom DMS-u:

- dodeljeni prioritet (dodeljeni od strane sistema ili korisnika);
- procenjeno vreme izvršenja zadatka (vreme izvršenja je korespodentno opterećenju računarskih resursa koje zadatak unosi u sistem);
- čvor računarskog *Grid*-a na kome je predviđeno da se zadatak izvrši (jedan od DMS servisa: SSP, SDP, SGP ili SZP).

U cilju optimalne alokacije računarskih resursa, algoritmi za upravljanje tokovima aktivnosti koriste osnovne karakteristike svakog zadatka. Pri tome, cilj optimizacije je da se preuredi redosled izvršavanja zadataka u skladu sa prioritetima tokova aktivnosti i maksimalnim mogućnostima upotrebe resursa računarskog *Grid*-a.

5. MODEL KOLEKCIJE TOKOVA AKTIVNOSTI

DMS se može okarakterisati kao računarski sistem koji reaguje na uticaje iz svoje okoline, u literaturi poznat kao *reactive systems* [100]. Ovu kategoriju računarskih sistema odlikuje interakcija sa sopstvenim okruženjem u okviru koje dolazi do razmene informacija. Njihova reakcija na uticaje koji dolaze iz okruženja predstavlja pokretanje određenih aktivnosti što za posledicu može imati izmenu stanja ovih sistema. Nakon izmene stanja, krajnji efekat pokrenutih aktivnosti je povratni uticaj na svoje okruženje. Može se uočiti da ključnu ulogu u njihovom ponašanju igraju komunikacija i interakcija sa sopstvenim računarskim okruženjem.

Klasični računarski sistemi prihvataju skup ulaznih podataka i nakon odgovarajuće obrade kreiraju određeni izlaz. Njihovo funkcionisanje može se predstaviti algoritmom koji opisuje transformaciju inicijalnog stanja programa u neko finalno. Krajnje je nepoželjno da se rad ne završi, odnosno da se ne dostigne finalno stanje. U takvoj situaciji ne bi se raspolagalo izlaznim skupom podataka koji predstavljaju rezultate obrade. Sa druge strane, ukoliko se posmatraju računarski sistemi koji treba da imaju konstantu interakciju sa svojim okruženjem, dolazi se do zaključka da je neprihvatljivo da imaju finalno stanje [101]. Za njih je neophodno da podržavaju neprekidan rad bez obzira u kom se stanju nađu, kako bi u svakom trenutku mogli da odgovore na uticaje iz spoljašnje sredine.

Sve navedene osobine računarskih sistema koji imaju interakciju sa okruženjem odnose se na DMS. S obzirom na važnost posla koji obavlja, neprekidan rad je prvi uslov funkcionisanja DMS-a. On prihvata stimulacije poslate od strane svog okruženja: akcije generisane putem korisničkog interfejsa, izmene veličina izmerenih vrednosti poslate od strane SCADA-e, najnovije informacije vezane za model distributivne mreže (statičke podatke) koje proizvodi GIS, itd. Za svaku od njih generišu se odgovarajući tokovi aktivnosti koji se prosleđuju na obradu. Nakon što se ovi tokovi aktivnosti izvrše, može doći do promene stanja DMS-a, koje se najbolje ogleda u izmenama statičkih i dinamičkih podataka. Dodatno, moguće je da se pojavi potreba za kreiranjem novog pogleda na distributivnu mrežu ili za generisanje odgovarajućih upravljačkih signala koji svoj put do opreme u polju nalaze preko SCADA sistema. S obzirom na slučajnost vremenskih trenutaka u kojima pristižu zahtevi iz okruženja, skup tokova aktivnosti, a time i skup zadataka koje nose sa sobom, ima dinamičku prirodu.

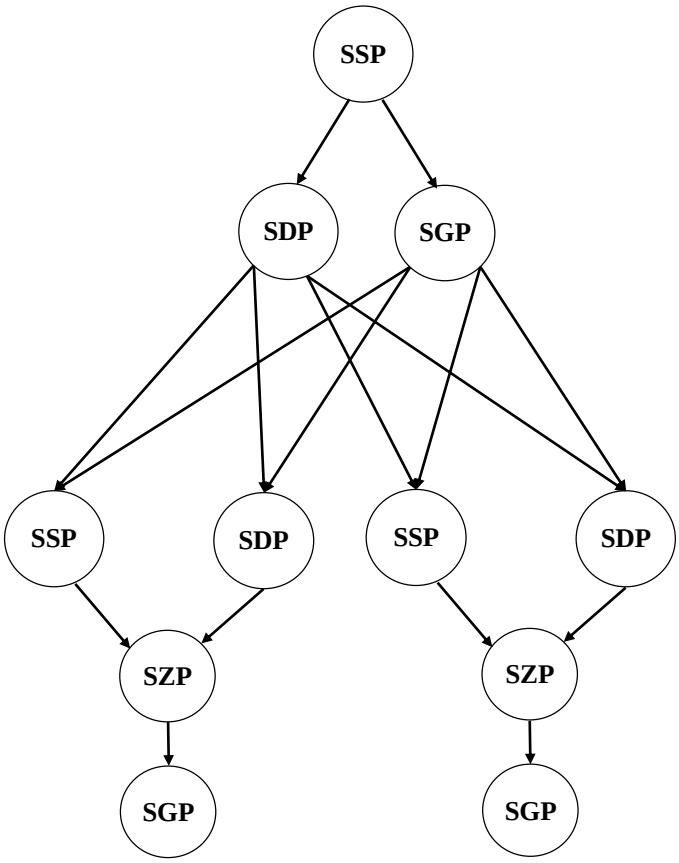
Da bi se proces raspodela zadataka mogao izvršavati na zadovoljavajući način, neophodno je analizirati postojeće tokove aktivnosti. Stoga je kolekcija tokova aktivnosti definisana kao skup svih tokova aktivnosti koji su prisutni u sistemu, u aktuelnom trenutku. Već je opisano da u toku rada DMS-a postoji stalna interakcija sa okruženjem uz dodatni faktor njene slučajne pojave, pa se dodatni tokovi aktivnosti uključuju u kolekciju tokova aktivnosti u nasumičnim vremenskim trenucima. Visok stepen dinamike koji je prisutan u ponašanju računarskog *Grid*-a onemogućava tačno predviđanje kada će određeni tok aktivnosti biti izvršen i uklonjen iz kolekcije. S obzirom na ovakvu dinamičku prirodu DMS-a, kolekcija tokova aktivnosti se konstantno menja, odnosno evoluira u toku rada sistema.

Već je objašnjeno da kao posledica poslovne logike tokovi aktivnosti mogu biti međusobno zavisni. Stoga, model njihove kolekcije mora uzeti u obzir ovu osobinu. Kako su u pitanju međusobno zavisne celine

logično bi bilo uvesti novi usmereni aciklični graf, čiji bi svaki čvor predstavljao jednu celinu, odnosno određeni tok aktivnosti, slika 4.5. Međutim, opterećenje resursa računarskog *Grid*-a je definisano na nivou zadatka koji predstavlja nedeljivu jedinicu. Iz tog razloga je potrebno raditi optimizaciju iskorišćenja računarskih resursa na nivou zadatka, a ne toka aktivnosti, i na taj način se prethodno pomenuti usmereni aciklični graf (čiji čvorovi reprezentuju tokove aktivnosti) dodatno uprošćava i unapređuje.

Da bi kolekcija tokova aktivnosti bila definisana na nivou zadatka, svaki tok aktivnosti se razlaže na zadatke koje obuhvata, dok se zadaci između međusobno zavisnih tokova aktivnosti povezuju granama grafa [102]. Ukoliko je tok aktivnosti B zavisan od toka aktivnosti A, onda izlazni zadaci toka aktivnosti A postaju neposredni prethodnici ulaznim zadacima toka aktivnosti B. Odnosno, izvršenje svih ulaznih zadataka toka aktivnosti B je uslovljeno izvršenjem svih izlaznih zadataka toka aktivnosti A.

Na slici 5.1 je prikazana kolekcija tokova aktivnosti kreirana na osnovu zavisnih tokova aktivnosti predstavljenih na slici 4.5. Izlazni zadaci toka aktivnosti tipa Ažuriranje modela su postali neposredni prethodnici ulaznim zadacima oba toka aktivnosti tipa Izvršenje DMS funkcije.



Slka 5.1 – Kreiranje kolekcije tokova aktivnosti

Kolekcija tokova aktivnosti predstavljena je preko usmerenog acikličnog grafa koji evoluira sa proticanjem vremena. Čvorovi predstavljaju zadatke (kompletan skup zadataka u sistemu), dok grane manifestuju zavisnosti između njih. Svaki čvor može da ima proizvoljan broj zadataka od kojih zavisi – ti zadaci predstavljaju neposredne prethodnike razmatranog čvora. Takođe, svaki čvor može da ima i proizvoljan broj zadataka koji su zavisni od njega – ti zadaci su neposredni sledbenici razmatranog čvora. Zadaci bez neposrednih prethodnika su ulazni zadaci, dok su zadaci bez neposrednih sledbenika izlazni zadaci kolekcije tokova aktivnosti.

Sistem za upravljanje tokovima aktivnosti analizira opisanu kolekciju tokova aktivnosti, prikuplja

zadatke bez neposrednih prethodnika i pokreće implementirane algoritme koji donose odluku o raspodeli zadataka na čvorove računarskog *Grid*-a. Da bi ovakva manipulacija uopšte bila moguća, neophodno je da sistem za upravljanje tokovima aktivnosti ima jasnu predstavu o modelu kolekcije tokova aktivnosti. Kreiranje formalnog modela kolekcije tokova aktivnosti koji se koristi je opisno u nastavku.

Sa V je označen skup svih zadataka koji trenutno čine kolekciju tokova aktivnosti. Zadatak T_i koji trenutno pripada kolekciji tokova aktivnosti, definisan je kao uređena trojka:

$$T_i = (p_i, t_i, n_i), \quad 1 \le i \le N_V, \tag{5.1}$$

gde je:

p_i – prioritet zadataka;

t_i – procenjeno vreme izvršenja zadatka (proporcionalno je opterećenju računarskih resursa koje unosi zadatak – w_i);

n_i – čvor izvršenja zadatka (DMS servis);

N_V – kardinalitet skupa svih zadataka koji trenutno čine kolekciju tokova aktivnosti.

Neka je sa E definisan skup uređenih parova elemenata skupa V **(**zadataka) između kojih postoji direktna zavisnost. Uređeni par koji opisuje zavisnost zadatka T_i od neposrednog prethodnika T_j predstavljen je na sledeći način:

$$(T_i, T_j), 1 \le i \le N_V \wedge 1 \le j \le N_V \wedge i \ne j\ . \tag{5.2}$$

Kolekcija tokova aktivnosti predstavljena je usmerenim acikličnim grafom $D(V, E)$, gde su sa V i E naznačeni skup čvorova grafa D i skup usmerenih grana grafa D, respektivno. Očigledno je da čvorovi grafa D predstavljaju zadatke unutar kolekcije tokova aktivnosti, dok usmerene grane opisuju zavisnost između njih. Zadaci bez neposrednih prethodnika, ulazni zadaci, predstavljaju elemente skupa *Inputs*, dok su izlazni zadaci elementi skupa *Outputs*. Napomena: u kontekstu usmerenog acikličnog grafa termin "čvor" odnosi se na pojedinačni zadatak unutar kolekcije tokova aktivnosti, a ne na resurs računarskog *Grid*-a.

Dodatno, skupovi zadataka neposrednih prethodnika i neposrednih sledbenika zadatka T_i su definisani kao P_{T_i} i S_{T_i}, respektivno. Očigledno je da važi:

$$T_i \in Inputs \ \Rightarrow \ P_{T_i} = \varnothing \ \text{ i} \tag{5.3}$$

$$T_i \in Outputs \ \Rightarrow \ S_{T_i} = \varnothing\ . \tag{5.4}$$

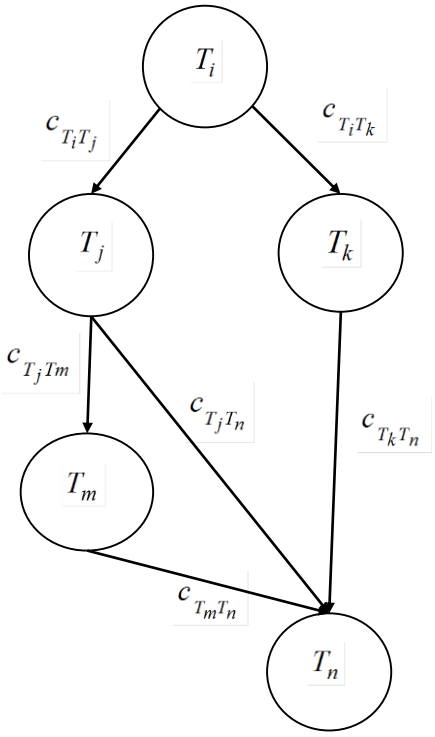
Prilikom rešavanja problema sličnih problemu koji se obrađuje u ovom rada, praksa je da se sistem za upravljanje tokovima aktivnosti dodatno snabde informacijama vezanim za prenos podataka između čvorova računarskog *Grid*-a. Na taj način se dobija efikasniji proces raspodele zadataka, a time optimizacija iskorišćenja računarskih resursa uspešnija [103]. Pored činjenice da je svakom čvoru usmerenog aciličnnog grafa dodeljena definicija nekog zadatka, svakoj grani se dodeljuje težinski faktor koji odgovara procenjenom vremenu za prenos podataka između dva čvora računarskog *Grid*-a [84].

Infrastruktura računarskog *Grid*-a predstavlja u potpunosti povezanu topologiju u kojoj se komunikacija obavlja bez bilo kakvih ograničenja. Podatke je moguće razmenjivati između bilo koja dva čvora (tj. DMS servisa). Ako je zadatak T_i neposredni prethodnik zadatka T_j tada je nakon izvršenja zadatka T_i neophodno preneti rezultate obrade sa čvora n_i na čvor n_j (da bi počelo izvršenje zadatka T_j). Količina

podataka koju je potrebno preneti naznačena je sa $data_{T_iT_j}$, a brzina prenosa podataka između čvorova n_i i n_j sa $rate_{n_in_j}$. Konačno, vremenski interval koji je potrebno da protekne od kraja izvršenja zadatka T_i do početka izvršenja zadatka T_j računa se kao:

$$c_{T_iT_j} = \frac{data_{T_iT_j}}{rate_{n_in_j}} \ . \tag{5.5}$$

Nakon što su svakoj grani usmerenog acikličnog grafa dodeljeni težinski faktori koji odgovaraju procenjenom vremenu prenosa podataka, model kolekcije tokova aktivnosti sadrži dovoljno informacija za implementaciju efikasnog procesa raspodele, slika 5.2.



Slika 5.2 – Model kolekcije tokova aktivnosti

Na osnovu slike 5.1 može se uočiti da postoji neposredna zavisnost između zadataka koji se izvršavaju na istom čvoru. Takva forma usmerenog acikličnog grafa je posledica zavisnosti tokova aktivnosti koji su modelovani na način da se izlazni zadatak toka aktivnosti prethodnika i ulazni zadatak toka aktivnosti sledbenika izvršavaju na istom čvoru računarskog *Grid*-a.

Na slici 5.1 su prikazani zadaci koji se izvršavaju na SDP-u. U prethodnom poglavlju rada je naglašeno da u toku ažuriranja modela distributivne mreže dolazi i do izmene dinamičkih podataka. Nakon izmene, odgovarajuće dinamičke podatke je potrebno prikupiti sa SDP-a i proslediti u SZP, da bi se izvršile analitičke elektroenergetske funkcije. Očigledno je da se oba navedena zadatka izvršavaju na SDP-u. Važno je naglasiti da u ovakvim situacijama, gde su izlazni podaci neposrednog prethodnika već stacionirani na čvoru na kojem treba da se izvrši neposredni sledbenik, težinski faktor grane usmerenog acikličnog grafa između ova dva zadatka uzima nultu vrednost, jer nema prenosa podataka.

Na osnovu prethodno opisanih atributa koji su dodeljeni čvorovima i granama usmerenog acikličnog grafa, kreiraju se dodatni, izvedeni atributi koji imaju značajnu ulogu prilikom implementacije algoritama razvijenih za potrebe ovog rada. U toku procesa raspodele zadataka, vrše se procene tih atributa i na osnovu

njihovih vrednosti se donose odluke koji od zadataka će dobiti prednost prilikom dodele odgovarajućeg računarskog resursa.

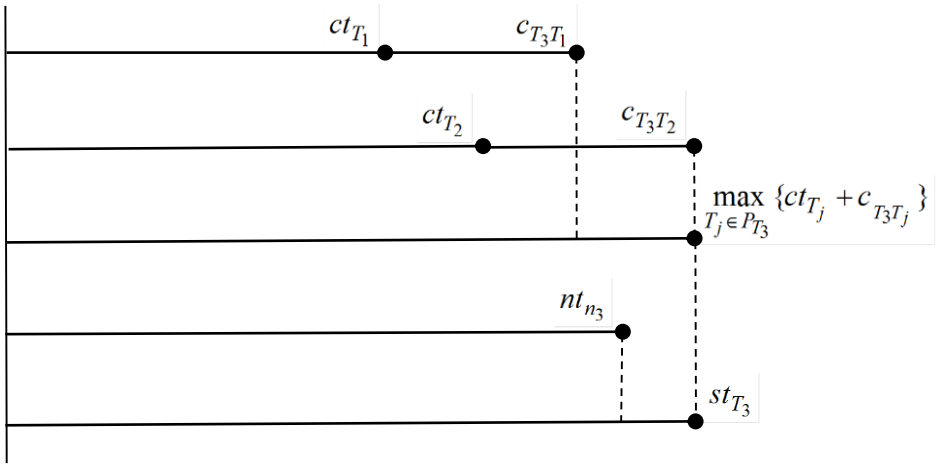
U nastavku rada su sa st_{T_i} i ct_{T_i} naznačena procenjena vremena početka i kraja izvršenja zadatka T_i, respektivno. Potrebno je definisati pravila na osnovu kojih se ove vrednosti mogu izračunati. Čvor računarskog *Grid*-a se smatra raspoloživim kada su izvršeni svi zadaci koji su mu prethodno dodeljeni. Predviđeno je da zadatak T_i bude izvršen na čvoru n_i. Tada nt_{n_i} predstavlja najraniji vremenski trenutak kada je čvor n_i raspoloživ. Da bi se utvrdilo kada može da otpočne izvršenje zadatka T_i, potrebno je proceniti kada će svi neposredni prethodnici zadatka T_i biti završeni, a njihovi rezultujući podaci dostupni na raspoloživom čvoru n_i. Dakle, najranije vreme početka izvršenje zadatka T_i je definisano kao vremenski trenutak kada su oba, prethodno navedena uslova ispunjena:

$$st_{T_i} = \max\left\{ \max_{T_j \in P_{T_i}} \{ct_{T_j} + c_{T_iT_j}\},\ nt_{n_i} \right\}, \tag{5.6}$$

gde je sa P_{T_i} naznačen skup neposrednih prethodnika zadatka T_i. Iz tog proizilazi da se vreme kraja izvršenja, ct_{T_i} računa kao vreme početka izvršenja zadatka st_{T_i} uvećano za procenjeno vremena njegovog izvršavanja t_i:

$$ct_{T_i} = st_{Ti} + t_i\,. \tag{5.7}$$

Jednostavan primer prikazan je na slici 5.3. Potrebno je odrediti najranije vreme početka izvršenja zadatka T_3. Neposredni prethodnici zadatka T_3 su zadaci T_1 i T_2, pa je $P_{T_3}=\{T_1,T_2\}$. Na slici su prikazana vremena završetka zadataka T_1 i T_2 (ct_{T_1} i ct_{T_2}, respektivno), kao i vreme potrebno da se prenesu rezultati njihovog izvršavanja na čvor n_3 ($c_{T_3T_1}$ i $c_{T_3T_2}$). Na osnovu ovih informacija određeno je vreme kada su svi ulazni podaci zadatka T_3 dostupni ($\max_{T_j \in P_{T_3}} \{ct_{T_j} + c_{T_3T_j}\}$). Kada se u analizu uključi i vreme kada n_3 postaje raspoloživ (nt_{n_3}) određuje se najranije vreme početka izvršenje zadatka T_3 (st_{T_3}).



Slika 5.3 – Primer određivanja najranijeg vremena početka izvršenja zadatka

Svrha primene bilo kog algoritma raspodele zadataka je da se u što većoj meri omogući paralelizacija izvršavanja zadataka i time postigne maksimalno iskorišćenje resursa računarskog *Grid*-a. Posledica opisane

optimalne upotrebe računarskih resursa je smanjenje ukupnog vremena izvršenja svih zadataka koji su predstavljeni u kolekciji tokova aktivnosti, čime se podižu performanse čitavog sistema.

Makespan kolekcije tokova aktivnosti predstavlja vremenski period od početka izvršenja prvog zadataka, do trenutka kada su svi izlazi na raspolaganju [104]. Njegova vrednost zavisi kako od vremena izvršenja zadataka, tako i od vremena potrebnog za razmenu podataka:

$$makespan = \max_{T_i \in Outputs} \{ct_{T_i}\}. \tag{5.8}$$

Cilj optimizacije je da se zadaci dodele resursima računarskog *Grid*-a na takav način da se *makespan* minimizuje.

6. SISTEM ZA UPRAVLJANJE TOKOVIMA AKTIVNOSTI

Sistem za upravljanje tokovima aktivnosti predstavlja kičmu organizacije poslova unutar računarskog *Grid*-a. On ne izvršava zadatke samostalno, već koordiniše njihovo izvršavanje na čvorovima računarskog *Grid*-a i obezbeđuje da se ne preskače ni jedan korak prilikom izvršavanja nekog toka aktivnosti, da se zadaci sprovode odgovarajućim redosledom, a pri tom da se oni paralelizuju gde god je to moguće. Iz navedenog proizilazi da sistem za upravljanje tokovima aktivnosti treba da raspolaže informacijama o stanju računarskih resursa i kolekciji tokova aktivnosti kako bi ispravno rukovodio tokovima aktivnosti i poslao zadatke na prikladne čvorove.

Razdvajanje upravljanja zadacima od njihovog izvršavanja nosi veliki broj prednosti [105]:

- Moguća je izolacija ove funkcionalnosti od ostatka sistema što značajno uprošćava eventualni, dodatni razvoj na njoj. Na primer, ako je neophodno izmeniti centralizovani algoritam raspodele zadataka, takav poduhvat se realizuje isključivo na ovom mestu.
- Omogućeno je da se upravljanje svim tipovima tokova aktivnosti realizuje na identičan način. Svaki tok aktivnosti se može razložiti na jednostavnije korake, zadatke, koji se izvršavaju kao nedeljive celine.
- Servisi instalirani na računarskim resursima su zaduženi samo za izvršavanje zadataka i pri tome ne zahtevaju funkcionalnost upravljanja zadacima. Servisi nemaju informacije o strukturi tokova aktivnosti, odnosno potpuno su nezavisni od nje, čime je njihova implementacija znatno jednostavnija. Ova prednost omogućava izmene strukture tokova aktivnosti u kasnijoj fazi.
- Posmatrano sa strane upravljanja, jednostavno je identifikovati pojedinačne tokove aktivnosti i njihovo stanje u toku izvršenja. Jasno je koji zadaci treba da se izvrše kako bi se korektno rukovodilo tokovima aktivnosti i koje računarske resurse je potrebno upotrebiti za to. S obzirom da je nadzor izvršavanja zadataka jednostavan, uska grla u sistemu se mogu jednostavno detektovati.

Sistem za upravljanje tokovima aktivnosti može da se realizuje na različite načine. Na primer, zavisno od okolnosti i funkcionalnosti koju celokupan sistem treba da osigura: kojim računarskim resursima raspolaže, na koji način je predviđeno da se upravlja greškama, kakva je interakcija poželjna sa korisnikom i slično. Iz tih razloga postoje različite implementacije [106, 107, 108, 109, 110, 111]. U nastavku su opisani struktura i interni procesi koje izvršava sistem za upravljanje tokovima aktivnosti predstavljen u ovom radu.

Prvenstveno je potrebno definisati kada se neki zadatak može uzeti u razmatranje da bi se poslao na izvršenje. Spreman zadatak je definisan kao zadatak čiji su svi neposredni prethodnici uspešno izvršeni. Izvršenje nekog zadatka može početi kada su ispunjeni svi relevantni preduslovi: svi neposredni prethodnici su izvršeni (postao je spreman), a ulazni podaci koje zadatak zahteva radi izvršenja su dostupni na predodređenom čvoru računarskog *Grid*-a. Proces raspodele zadataka je sveobuhvatan postupak od ulaska toka aktivnosti u sistem za upravljanje tokovima aktivnosti do dodele zadataka pojedinim čvorovima računarskog *Grid*-a. Algoritam raspodele zadataka predstavlja deo procesa raspodele koji koristi usmereni aciklični graf da bi se odredilo kada ili kojim redom će zadaci bit poslati na izvršenje. Ukoliko sistem za

upravljanje tokovima aktivnosti raspolaže većom količinom informacija, utoliko će proces raspodele zadataka biti efikasniji.

Sa stanovišta procesa raspodele, *fluktuacija performanse računarskih resursa* je jedna od najvažnijih karakteristika računarskog *Grid*-a [112]. Drugi važan uticaj ima *frekvencija pristizanja novih tokova aktivnosti*. Efikasna raspodela zadataka treba da se prilagodi i odgovori na dinamičko ponašanje okruženja. U tu svrhu potrebna je implementacija dinamičke raspodela zadataka, gde će se odluke donositi u toku rada celokupnog sistema.

Da bi se u potpunosti odgovorilo na visok stepen stohastike, dinamička raspodela se dodatno unapređuje potencijalnom preraspodelom odgovarajuće grupe spremnih zadataka [113]. To se postiže ponovnim pokretanjem algoritma raspodele nad zadacima koji su spremni i koji su taj deo procesa raspodele već jednom prošli. Posledica ovog postupka je da se menjaju prethodno donete odluke vezane za raspodelu, na osnovu novih, aktuelnih informacija iz okruženja.

U dosadašnjim istraživanjima se uglavnom uzimala u obzir preraspodela zadataka kao opcija u slučaju da su performanse izvršavanja nezadovoljavajuće [45]. Postupak je sledeći [84]: u toku rada sistema neprestano se prate odgovarajući parametri (na primer razlika između stvarnog vremena završetka zadatka i procenjenog vremena njegovog završetka) i ukoliko neki od njih naruši predefinisanu granicu vrši se preraspodela zadataka (da bi se performanse sistema poboljšale). Smatra se da će se utroškom dodatnog vremena na preraspodelu zadataka smanjiti ukupno vreme koje je neophodno da se svi zadaci izvrše, u odnosu na vreme koje bi bilo potrebno da se isti zadaci izvrše na osnovu prethodne, neefikasne raspodele. Međutim, sistem za upravljanje tokovima aktivnosti koji je razvijen u ovom radu mora da uzme u obzir i prioritete tokova aktivnosti, a ne samo kvalitet raspodele. Novi tokovi aktivnosti konstantno pristižu i moguće je da neki od njih imaju viši prioritet od tokova aktivnosti koji su već prošli kroz raspodelu. Obzirom da je moguće da su neki zadaci novih tokova aktivnosti višeg prioriteta od prethodno razmotrenih zadataka, neophodno je uključiti nove, prioritetnije zadatke u raspodelu da bi dobili prednost prilikom dodele računarskog resursa. Ovo ima za posledicu da je preraspodelu neophodno konstanto vršiti ukoliko se ne stvori negativan uticaj na iskorišćenje računarskih resursa, odnosno ukoliko preraspodela ne donosi više štete nego koristi.

Sistem za upravljanje tokovima aktivnosti koji je razvijen u ovom radu, uzima u obzir dinamičku prirodu kako DMS-a tako i računarskog *Grid*-a, na kome je celokupan sistem postavljen da bi adekvatno reagovao [102]. Prilikom primene algoritma raspodele razmatraju se karakteristike zadataka koji su uključeni u usmereni aciklični graf i struktura grafa, u želji da se *makespan* minimizuje i tako poboljšaju performanse čitavog DMS-a.

Kako bi sistema za upravljanje tokovima aktivnosti mogao kvalitetno da optimizuje upotrebu resursa računarskog *Grid*-a i omogući što efikasnije funkcionisanje DMS-a definisana su sledeća pravila:

- S obzirom da nije moguće tačno odrediti kada će neki čvor računarskog *Grid*-a biti raspoloživ, algoritam raspodele zadataka treba da odredi redosled po kojem će se do tada spremni zadaci izvršavati.
- Algoritam za raspodelu zadataka treba da se primenjuje, u što većoj meri, u toku izvršavanja zadataka, odnosno dok su čvorovi računarskog *Grid*-a zauzeti. Na taj način se unapred pripremaju naredni zadaci koji treba da budu poslati na izvršenje.
- U preraspodelu zadataka je potrebno uključiti što veći broj zadataka kako bi se na izvršenje poslali oni koji će najviše doprineti minimzaciji vrednosti *makespane*-a. Iz tog razloga se spreman zadatak šalje

na izvršenje tek kada je čvor računarskog *Grid*-a raspoloživ.

- Da bi čvorovi računarskog *Grid*-a bili maksimalno iskorišćeni potrebno je da budu okupirani poslom u što dužem vremenskom periodu u toku rada DMS-a. To se postiže tako što dodela zadatka pojedinim čvorovima ima prednost u odnosu na njihovu preraspodelu.

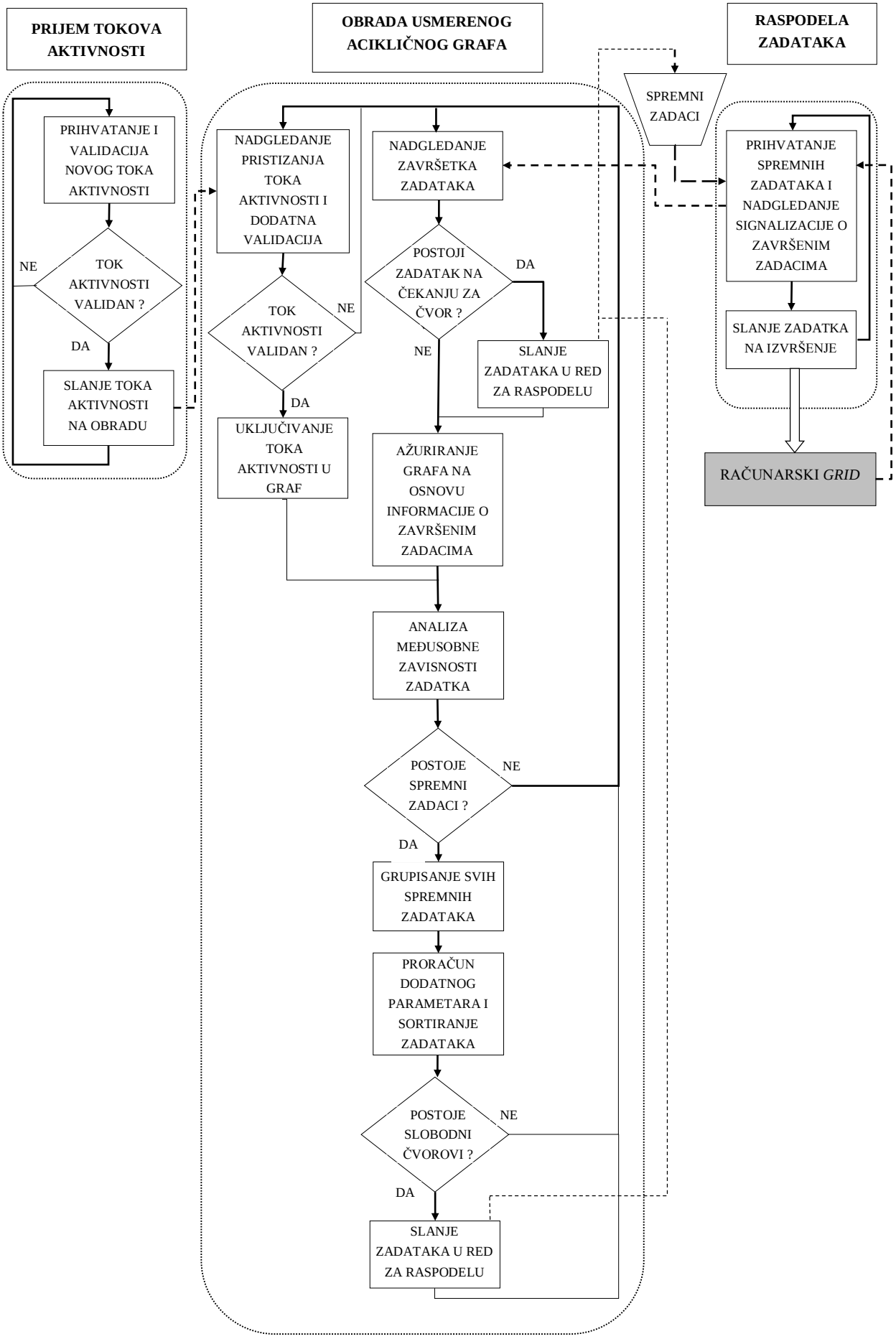
Resursi računarskog *Grid*-a nisu savršeni. Moguća je pojava različitih disfunkcija poput neispravnih memorijskih blokova, mehaničkih kvarova na čvrstom disku, izmena vrednosti bita u radnoj memoriji usled zračenja i slično. Sistemi za upravljanje tokovima aktivnosti koje odlikuje konstantan rad uz željene performanse moraju implementirati odgovarajuće tehnike za oporavak nakon pojave greške. Pod greškom se podrazumeva neuspešno završen zadatak iz bilo kog razloga. Postoje dva osnovna pristupa za poboljšanje otpornosti na greške [114]:

- Preventivni pristup koji predlaže analizu zadataka pre dodele računarskim resursima, što znači da razmatra mogućnost pojave greške pre izvršenja zadataka. Ukoliko se zadaci smatraju korektnim, šalju se u nadi da će se uspešno izvršiti.
- Pristup sanacije grešaka podrazumeva pokretanje odgovarajuće procedure tek nakon što se greška pojavi. Definisana su odgovarajuća pravila ponašanja sistema za upravljanje tokovima aktivnosti kojih treba da se pridržava da bi se oporavio i nastavio normalno da funkcioniše.

Prikazani sistem za upravljanje tokovima aktivnosti uzima u obzir mogućnost pojave grešaka i koristi odgovarajuće tehnike da bi DMS pouzdano radio i odgovorio na zahteve korisnika.

Razvijena arhitektura sistema za upravljanje tokovima aktivnosti koji se javljaju u DMS-u pruža potrebnu podršku za proces raspodele zadataka visoke propusne moći, slika 6.1. Arhitektura definiše tri osnovne komponente koje imaju sledeće funkcije:

- prijem i validacija tokova aktivnosti,
- obrada usmerenog acikličkog grafa,
- neposredna raspodela zadataka na čvorove računarskog *Grid*-a.



Slika 6.1 – Sistem za upravljanje tokovima aktivnosti

6.1 INICIJALNA VALIDACIJA TOKOVA AKTIVNOSTI

Da bi se osigurala ispravnost funkcionisanja DMS-a, mora se sprovesti validacija tokova aktivnosti koji pristižu na izvršenje. Inicijalna validacija predstavlja prvu liniju odbrane od potencijalnih grešaka ili blokade rada čitavog sistema za upravljanje tokovima aktivnosti [115, 116]. U okviru ovog istraživanja predložena inicijalna validacija se sastoji od sledeća tri nivoa:

- Validacija strukture – U toku rada sistema, komponenta za prijem tokova aktivnosti prihvata tokove aktivnosti i prvenstveno izvršava validaciju strukture. Proverava da li strukture njihovih usmerenih acikličnih grafova odgovaraju nekom od predefinisanih obrazaca [105]. Ukoliko je struktura validna, prelazi se na naredni nivo validacije toka aktivnosti.
- Validacija zahtevanih resursa – Proverava se da li postoje nelogičnosti između pristiglih tokova aktivnosti i čvorova računarskog *Grid*-a koji su potrebni da se tokovi aktivnosti izvrše [117, 118]. Uloga ovog nivoa validacije je da definitivno potvrdi da postoji odgovarajući računarski resurs na kome može da se izvrši svaki od zadataka (moguće je da neki od zadataka zahteva nepostojeći računarski resurs).
- Validacija ovlašćenja – U trećem poglavlju je opisana podrška koju pruža infrastruktura računarskog *Grid*-a. Naglašeno je da, s obzirom na važnost posla koji DMS obavlja, sigurnost podatka igra značajnu ulogu. Shodno distribuiranom okruženju, neophodna je implementacija decentralizovanih sigurnosnih mehanizama. Autorizacija podrazumeva obezbeđivanje zaštite tako da neautorizovan korisnik ne može da pristupi ni jednom resursu [119, 120]. Iz ugla računarskog *Grid*-a koji koristi DMS, ovakav koncept podrazumeva da svaki čvor implementira mehanizme potrebne za očuvanje sigurnosti podataka koje koristi DMS servis postavljen na taj čvor. Dodatno, ovi mehanizmi obezbeđuju zaštitu od neovlašćenog pokretanja odgovarajućih procedura koju DMS servis na tom čvoru podržava. Različita prava mogu biti dodeljene različitim korisnicima. Na primer, korisnik koji je zadužen za ažuriranje modela distributivne mreže (na osnovu podataka koji stižu iz nekog podsistema, npr. iz GIS), ima prava i da čita i da menja statičke podatke na SSP-u, dok su mu, najčešće, uskraćena prava pokretanja analitičkih elektroenergetskih funkcija. Sa druge strane, operater u kontrolnom centru nema dozvolu da menja statičke podatke, ali ima mogućnost pokretanja analitičkih elektroenergetskih funkcija. Dakle, prvi korisnik (koji generiše tok aktivnosti tipa Ažuriranje modela) nema prava pokretanja nikakve funkcionalnosti koju obezbeđuje SZP. Iz tog razloga, čvor za proračune implementira sigurnosne mehanizme koji ne dozvoljavaju izvršavanje bilo kog zadatka poslatog od strane prvog korisnika. Operater generiše tok aktivnosti tipa Izvršenje DMS funkcije. Neophodno je pročitati odgovarajuće statičke i korespodentne dinamičke podatke kako bi se pokrenuo predviđeni proračun. Stoga, sigurnosni mehanizmi implementirani na čvorovima na kojima se nalaze SSP i SDP, dozvoljavaju izvršavanje zadataka koji nose kredencijale operatera samo ako se odnose na čitanje podataka. Ukoliko operater zahteva izmenu nekog od ova dva modela biće odbijen. Očigledno, prilikom pristizanja nekog toka aktivnosti neophodno je proveriti da li korisnik koji je odgovoran za njegovo generisanje uopšte ima prava da prosledi bilo kakav tok aktivnosti. Ukoliko su prava odgovarajuća, dodatno se proverava da li korisnik ima prava da izvrši baš taj tip toka aktivnosti (da li ima prava za manipulaciju računarskim resursima koje prosleđeni tok aktivnosti zahteva). Ako se zaključi da bi došlo do neautorizovanog pristupa resursima, sistem za upravljanje tokovima aktivnosti može da preduzme zaštitne mere već u ovoj fazi manipulacije nekim tokom aktivnosti. Ključni izazov na ovom nivou validacije je obezbediti da sistem za upravljanje tokovima aktivnosti poznaje i razume

sigurnosne mehanizme podržane od strane infrastrukture računarskog *Grid*-a [116, 121].

Inicijalna validacija ima veliki značaj za kompletan proces upravljanja tokovima aktivnosti. Ukoliko bi u dalju obradu ušao neki tok aktivnosti koji nema odgovarajuću strukturu, postojala bi mogućnost kreiranja cikličnih veza između zadataka u okviru usmerenog acikličnog grafa. U predstavljenom sistemu za upravljanje tokovima aktivnosti, ovakva struktura bi prilikom analize grafa imala za posledicu ulazak u beskonačnu petlju i sistem bi bio blokiran. Nedovoljna prava korisnika koji je odgovoran za generisanje toka aktivnosti mogu prouzrokovati različite nepovoljne efekte. Prvenstveno, nepotrebno se uvećava kolekcija tokova aktivnosti, čime se povećava vreme analize usmerenog acikličnog grafa i troši se više vremena na izvršenje algoritama za raspodelu zadataka (ti zadaci kasnije mogu biti odbijeni od strane infrastrukture računarskog *Grid*-a). Dodatno, nakon odbacivanja pomenutog zadatka, svi zadaci koji su sledbenici (neposredni ili posredni) zadatka generisanog od strane korisnika koji ima neodgovarajuća prava će biti uklonjeni iz usmerenog acikličnog grafa. Treba istaći da se inicijalnom validacijom proverava svaki tok aktivnosti pojedinačno, bez njegovog uticaja na kolekciju tokova aktivnosti.

6.2 OBRADA USMERENOG ACIKLIČNOG GRAFA

Komponenta zadužena za obradu usmerenog acikličnog grafa je kičma sistem za upravljanje tokovima aktivnosti. Ukoliko je inicijalna validacija uspešno završena, tok aktivnosti se šalje na dalju obradu. Komponenta za obradu usmerenog acikličkog grafa praktično manipuliše i upravlja kolekcijom tokova aktivnosti. Ova komponenta se obaveštava o postojanju novog, inicijalno validnog toka aktivnosti. Ona taj tok aktivnosti prihvata i proverava da li unosi nelogičnosti vezane za kolekciju tokova aktivnosti. Ako novi tok aktivnosti nosi informaciju o svojim prethodnicima, proverava se da li su prethodnici pristigli ranije, odnosno da li su već u potpunosti izvršeni ili uključeni u kolekciju tokova aktivnosti. Ako komponenta za obradu usmerenog acikličkog grafa ne poseduje takve informacije o svim prethodnicima, novi tok aktivnosti se proglašava nevalidnim i odbacuje se. Treba napomenuti da mogu biti implementirani i napredniji mehanizmi. Na primer, moguće je da usled opterećenosti računarske mreže prethodnici tokova aktivnosti kasne. Tada se novi tok aktivnosti može čuvati predefinisani vremenski period. Ukoliko se za to vreme pojave svi validni prethodnici, čitava grupa tokova aktivnosti se šalje na dalju obradu kako bi se uključila u usmereni aciklični graf kojim je kolekcija tokova aktivnosti modelovana.

6.2.1 MODIFIKACIJA I ANALIZA USMERENOG ACIKLIČNOG GRAFA

Uključivanje u kolekciju tokova aktivnosti odvija se na način opisan u petom poglavlju. Ako je novi tok aktivnosti u potpunosti nezavisan, onda ne postoji potreba da se njegovih ulazni zadaci povežu, već se tok aktivnosti tretira kao ostrvo – nepovezani deo grafa. U suprotnom, ulazni zadaci novog toka aktivnosti postaju neposredni sledbenici izlaznih zadataka svih tokova aktivnosti od kojih razmatrani tok aktivnosti zavisi (njegovi prethodnici). Nakon ove akcije inicira se analiza usmerenog acikličnog grafa koja će detaljno biti objašnjena u okviru ovog dela rada.

Dolazak novih tokova aktivnosti u slučajnim vremenskim trenucima predstavlja jedan od dva glavna razloga koji utiču da se kolekcija tokova aktivnosti (odnosno njen usmereni aciklični graf) neprekidno menja i evoluira u toku rada DMS-a. Istovremeno, dok se prate validni tokovi aktivnosti koji pristižu, motri se na signalizaciji koja potvrđuje da je neki zadatak izvršen. Sistem za upravljanje tokovima aktivnosti poseduje kompletan skup zadataka kao i procenjene atribute dodeljene svakom od tih zadataka (da bi se procenilo kada će neki od zadataka biti izvršen). Međutim, ponašanja računarskog *Grid*-a nije predvidivo u dovoljno

dobroj meri [112], jer npr. neki čvor može biti zauzet izvršavanjem internih zadataka koji su nametnuti od strane infrastrukture računarskog *Grid*-a ili računarska mreža može biti preopterećena. Iz tog razloga, procenjeno vreme kada će se neki zadatak završiti se veoma često ne podudara sa vremenom kada je zadatak stvarno završen. Sistem za upravljanje tokovima aktivnosti mora uzeti u obzir i ovakvu stohastičku prirodu računarskog *Grid*-a. Stoga, pojava da se zadaci završavaju u slučajnim vremenskim trenucima predstavlja drugi razlog zbog koga se usmereni aciklični graf neprekidno menja kroz vreme.

Kada dođe do pojave bilo kojeg od dva prethodno navedena događaja, neophodno je ažurirati usmereni aciklični graf. Prilikom pristizanja novog validnog toka aktivnosti, njegovi zadaci bivaju uključeni u graf, dok u slučaju signalizacije da je određeni zadatak završen isti biva uklonjen iz grafa. Informacije o zavisnosti zadataka se ažuriraju. Potpuno je jasno da postoji mogućnost da se nakon modifikacije usmerenog acikličnog grafa pojave novi spremni zadaci: ukoliko je uklonjen upravo završen zadatak, svi njegovi neposredni sledbenici mogu postati spremni dok, sa druge strane uključivanje novog toka aktivnosti koji je potpuno nezavisan od onih koji trenutno postoje u kolekciji tokova aktivnosti unosi spremne zadatke (svoje ulazne zadatke). Na slici 6.2 su prikazane opisane pojave. Njome su predstavljeni zadaci koji čine usmereni aciklični graf kao i prioritet za svakog od njih.

Da bi primer bio jasniji, pojednostavljene su okolnosti uz pretpostavku da se svi zadaci izvršavaju na istom čvoru računarskog *Grid*-a. Prikazano, inicijalno stanje usmerenog acikličnog grafa, slika 6.2a, ima tri spremna zadatka koji su na slici uokvireni pravougaonikom: T_1, T_2 i T_3. S obzirom da je zadatak T_1 najvišeg prioriteta, ima prednost u toku procesu raspodele u odnosu na preostala dva zadatka, odnosno prvom mu se dodeljuje računarski resurs. Ako se pretpostavi da je uključen novi tok aktivnosti koji sadrži zadatke T_8 i T_9 i koji je nezavisan od preostalih tokova aktivnosti, njegov ulazni zadataka T_8 je spreman za izvršenje. Ukoliko se pri tome zadatak T_1 izvršio, skup spremnih zadataka se dodatno menja u odnosu na početni. Neposredni sledbenici T_4 i T_5 takođe postaju spremni, slika 6.2b.

Primer sa slike 6.2 predstavlja evoluciju usmerenog acikličnog grafa u toku rada DMS-a. Neprestana interakcija DMS-a sa okruženjem i nepredvidivo ponašanje računarskih resursa su inicijatori ovakvog ponašanja sistema. Posledice stalne promene sistema su dvojake:

- prethodno procenjene vrednosti završetka spremnih zadataka ne moraju biti tačne;
- skup spremnih zadataka se menja, uz mogućnost da se novi zadaci, višeg prioriteta, uključe u njega.

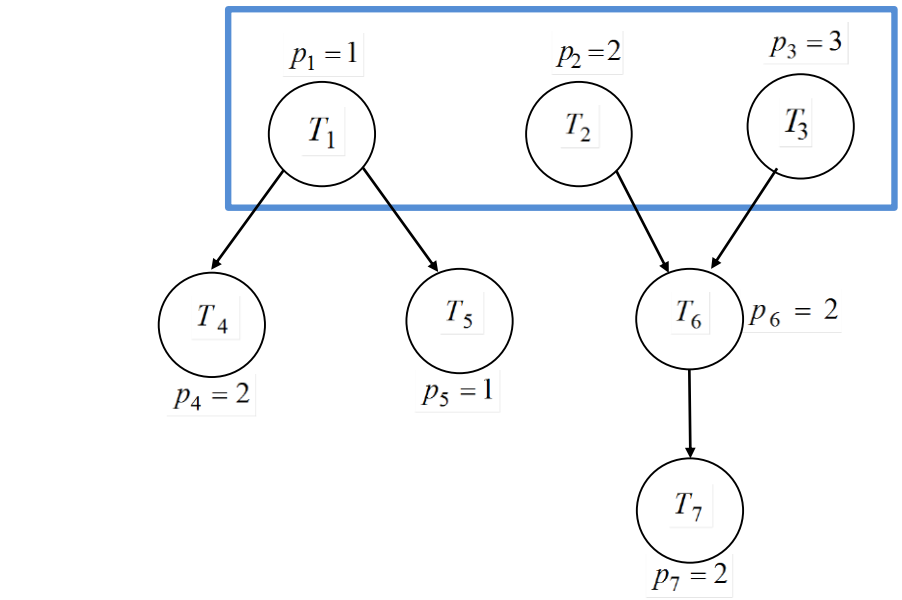
Tako na primer, za razmatrani primer nakon prvobitne primene algoritma raspodele nad spremnim zadacima, zadatak T_1 je usled dodeljenog prioriteta dobio prednost u odnosu na zadatke T_2 i T_3. Nakon signalizacije o uspešno izvršenom zadatku T_1 skup svih trenutno spremnih zadataka u usmerenom acikličnom grafu se izmenio. Pred sistemom za upravljanje tokovima aktivnosti su dve mogućnosti:

- da se izvrše svi zadaci čiji je redosled izvršavanja određen prvobitnom raspodelom i da se nakon toga krene u primenu algoritma raspodele nad svim spremnim zadacima u tom trenutku;
- da se novi, spremni zadaci grupišu sa postojećim, spremnim zadacima, i da se nad svima njima u celini primeni algoritam raspodele.

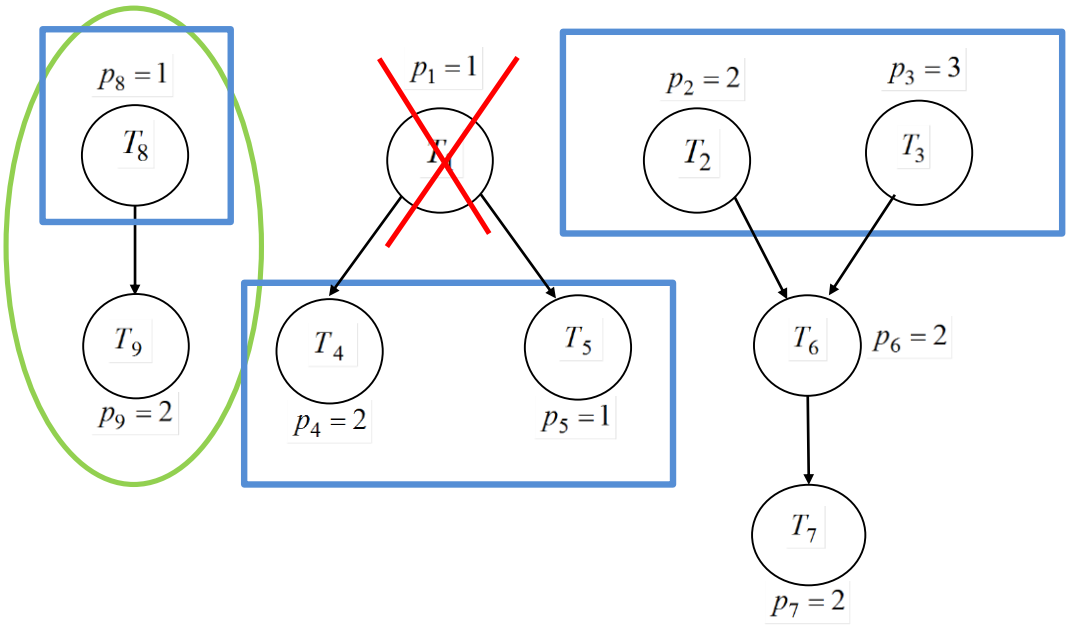
Ukoliko se izabere prva opcija, moguće je da će zadaci manjeg prioriteta biti izvršeni pre zadataka višeg prioriteta. Na primer, za razmatrani primer, zadatak T_2 će bit poslat na izvršenje pre zadatka T_5, iako je zadatak T_5 višeg prioriteta. Takođe, postoji verovatnoća da na izvršenje neće biti poslat spreman zadatak koji će umanjiti *makespan* kolekcije tokova aktivnosti u najvećoj meri. Na primer, možda je izvršenje

zadatka T_5 ili T_8 optimalno, radi smanjenja *makespan*-a, a sistem za upravljanje tokovima aktivnosti to ne može da proceni pošto algoritam raspodele nije uzeo u obzir ta dva zadatka.

Sa druge strane, pokretanje algoritma raspodele usled svake interakcije sa spoljnim okruženjem, može da bude vremenski veoma skupo, jer će se vreme trošiti na primenu algoritma raspodele dok neki od čvorova računarskog *Grid*-a nije uposlen. Potrebno je naći svojevrstan kompromis da se algoritam raspodele zadataka izvršava kada je to potrebno, ali da to ne utiče na rad čvorova računarskog *Grid*-a.



a)



b)

Slika 6.2 – Spremni zadaci u kolekciji tokova aktivnosti: a) inicijalni skup, b) skup nakon izvršenja jednog zadatka i uključivanje novog toka aktivnosti

Efikasan proces raspodele mora reagovati na dinamičke promene u sistemu, uz ograničenje da se usled odgovora na te promene iskorišćenje računarskih resursa ne umanji. Cilj optimizacije je da se u što većoj meri uposle čvorovi računarskog *Grid*-a, odnosno da se vremenski interval u kome čvor ne obrađuje neki

zadatak svede na minimum.

Razmatra se početni momenat kada je kolekcija tokova aktivnosti prazna i kada su svi čvorovi raspoloživi. Nakon što pristigne prvi validni tok aktivnosti, prikupljaju se svi spremni zadaci i primenjuje se algoritam raspodele (čime je otpočeo prvi ciklus raspodele zadataka). Spremni zadaci koje je algoritam raspodele označio kao primarne, šalju se na odgovarajuće čvorove računarskog *Grid*-a. U nastavku rada sistema za upravljanje tokovima aktivnosti prednost dobija akcija slanja zadataka (radi izvršenja na raspoloživim čvorovima), u odnosu na primenu algoritma raspodele. Iz tog razloga, primena algoritma raspodele je ograničena na podskup trenutno spremnih zadataka koji je određen statusom čvorova. Posmatraju se raspoloživi čvorovi računarskog *Grid*-a i na svaki od njih se šalju spremni zadaci koji su dobili prednost u prethodnom ciklusu raspodele (u odnosu na sve spremne zadatke iz prethodnog podskupa). Nakon toga, algoritam raspodele se primenjuje nad preostalim spremnim zadacima, poštujući njihov prioritet i korigujući procene vremena završetka svakog od njih. Ova akcija, ponovljene primene algoritma raspodele nad nekim zadacima, naziva se preraspodela zadataka. U daljem tekstu je detaljno objašnjeno celokupno funkcionisanje komponente zadužene za obradu usmerenog acikličnog grafa.

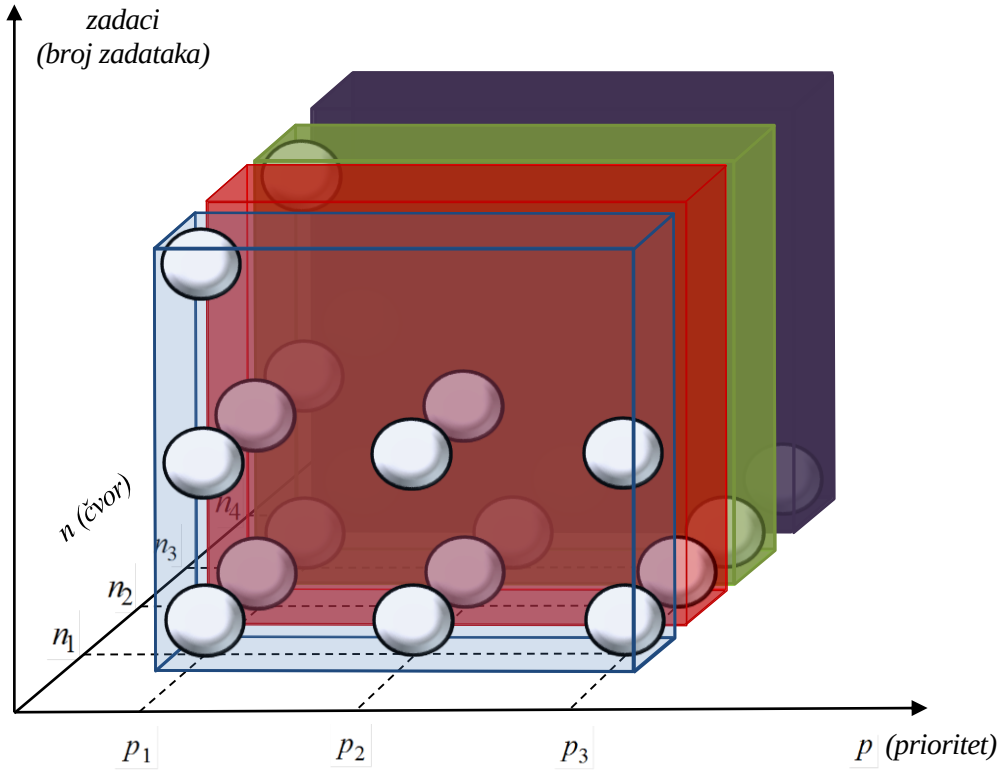
Kada pristigne signalizacija da je neki od zadataka završen, jasno je da je čvor računarskog *Grid*-a, kome je zadatak bio dodeljen, postao raspoloživ i moguće mu je poslati novi zadatak na izvršenje. Sa druge strane, neki od neposrednih sledbenika pomenutog završenog zadatka postaju spremni, pa ih je moguće uključiti u algoritam raspodele. Kako je već objašnjeno, komponenta zadužena za obradu usmerenog acikličnog grafa prednost daje dodeli zadatka raspoloživom čvoru, ukoliko takav zadatak postoji. Na taj način se upošljava raspoloživ čvor računarskog *Grid*-a u što kraćem roku. Ovakav pristup povećava iskorišćenje računarskih resursa, ali ima i svojih nedostataka. Upravo poslat zadatak dobija prednost pri izvršenju u odnosu na preostale spremne zadatke u prethodnom ciklusu primene algoritma raspodele. U novom ciklusu raspodele, pored do sada spremnih zadataka, uključuju se i neposredni sledbenici upravo završenog zadatka. Moguće je da je za neke od njih predviđeno da se izvršavaju na pomenutom, raspoloživom čvoru računarskog *Grid*-a, a da pri tom imaju viši prioritet od upravo poslatog zadatka ili imaju veći uticaj na minimizaciju *makespan*-a kolekcije tokova aktivnosti. U razmatranom primeru, slika 6.2, opisana je takva situacija. U prvom ciklusu raspodele, zadatak T_1 je dobio prednost u odnosu na zadatak T_2 koji je sa druge strane dobio prednost u odnosu na zadatak T_3. Ovakav dodeljen redosled izvršavanja je posledica prioriteta zadataka. Nakon što se izvršio zadatak T_2, zadaci T_5 i T_8 postaju spremni. Međutim, po pravilima opisane procedure, na izvršenje će biti poslat zadatak T_2, koji je nižeg prioriteta od zadataka T_5 i T_8. Nakon toga sledi novi ciklus raspodele u koji će biti uključeni i zadaci T_5 ili T_8. Očigledno, situacije u kojima je moguće da se zadatak nižeg prioriteta pošalje na izvršenje pre zadatka višeg prioriteta posledica su želje za što boljim iskorišćenjem računarskih resursa. Međutim, prostor za ovakve greške je sveden na minimum, dok su računarski resursi maksimalno iskorišćeni. Dodatno, izvršenje algoritma raspodele zadataka u velikoj meri se odvija istovremeno dok traje izvršavanje zadataka, što ima minimalni uticaj na uvećanje *makespan*-a.

Nakon uspešne modifikacija usmerenog acikličnog grafa, sledi analiza međusobne zavisnosti zadataka, odnosno određivanja zadataka koji su postali pogodni za izvršenje. Ukoliko postoje takvi, spremni zadaci pristupa se primeni algoritma raspodele.

Proces raspodele zadataka je odgovoran za izbor računarskih resursa i mapiranje zadataka na njih. Ovaj proces je od najvišeg interesa, jer je zadužen za najbitniji deo optimizacije. Prvenstveno treba istaći da se svaki od zadatka mapira na tačno jedan čvor računarskog *Grid*-a. Ovakva situacija znatno olakšava proces

raspodele, jer se od sistema za upravljanje tokovima aktivnosti ne zahteva da u datom trenutku vodi računa na kom čvoru je optimalno izvršenje zadatka. S obzirom da su zadaci već mapirani na čvorove računarskog *Grid*-a, optimizacija se vrši po čvoru, uzimajući u obzir dodeljeni prioritet svakog od zadataka. Shodno tome, primena algoritma raspodele je odgovorna za definisanje redosleda izvršavanja zadataka na svakom čvoru posebno, ali tako da se minimizira ukupno vreme izvršavanja zadataka na svim čvorovima zajedno.

Svi spremni zadaci se analiziraju i grupišu u zavisnosti od čvora računarskog *Grid*-a na kome treba da se izvrše. Rezultat analize je skup spremnih zadataka dodeljen svakom od čvorova. Svaki od skupova se dodatno deli u nivoe. Zadaci istog prioriteta se raspoređuju na isti nivo, a nivoi su sortirani u opadajućem redosledu po prioritetu. Opisana klasifikacija zadataka je prikazana na slici 6.3. Zadaci organizovani na taj način su spremni za primenu algoritma raspodele koji će odrediti njihov redosled izvršavanja.

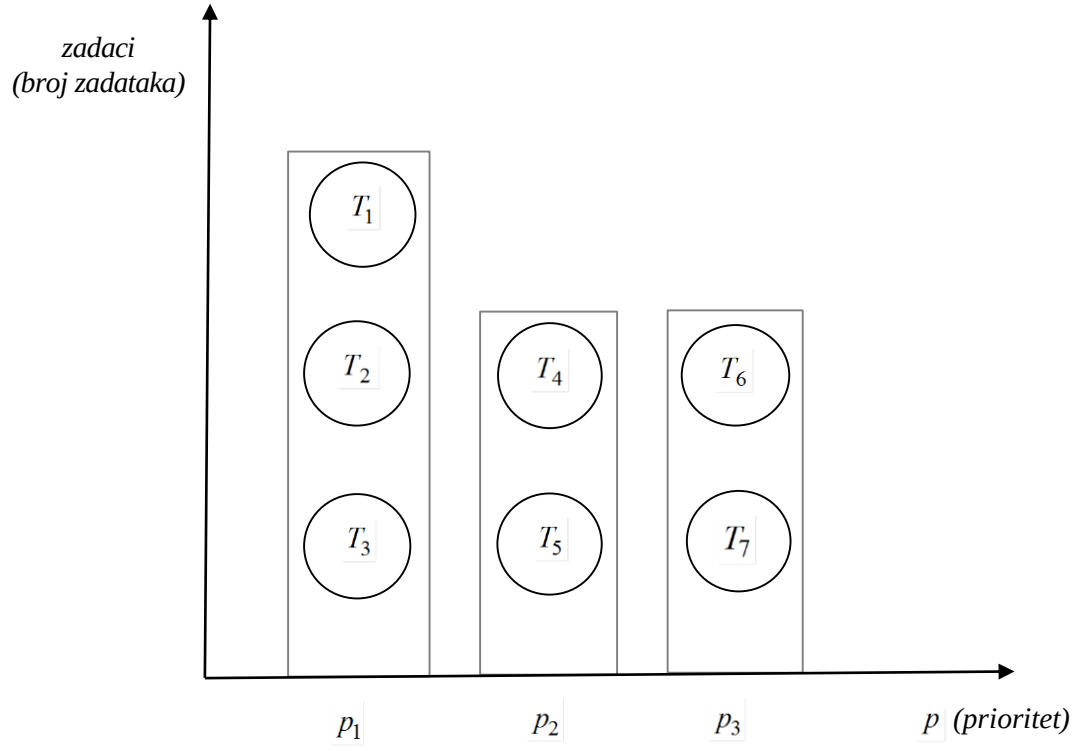


Slika 6.3 – Grupisanje spremnih zadataka po čvorovima računarskog Grid-a i prioritetima

6.2.2 PRIMENA ALGORITMA RASPODELE

Algoritam raspodele uzima u obzir skup spremnih zadataka za svaki čvor računarskog *Grid*-a posebno. Spremni nezavisni zadaci su klasifikovani po nivoima, prema prioritetu. Na slici 6.4 su izdvojeni klasifikovani zadaci koji odgovaraju zadacima dodeljenim čvoru n_1 sa slike 6.3. Tri zadatka su najvišeg prioriteta p_1, a nakon toga slede po dva zadatka nižih prioriteta p_2 i p_3. Neophodno je da zadaci višeg prioriteta dobiju prednost prilikom dodele računarskih resursa. Iz tog razloga, nivo po nivo razmatra se trenutno stanje čvorova računarskog *Grid*-a i karakteristike spremnih zadatka na tekućem nivou obrade. Zadaci najvišeg prioriteta se prikupljaju i za svaki od njih se izračunava dodatni, *algoritamski parametar*. Na njegovu vrednost utiču ne samo zadaci za koje je predviđeno izvršavanje na obrađivanom čvoru, već i podskup zadataka iz čitavog usmerenog acikličnog grafa. Konačno, algoritamski parametar se koristi kako bi se zadaci u okviru istog nivoa prioriteta sortirali i tako utvrdio redosled kojim se šalju na izvršenje.

Njegovom primenom omogućeno je da izvršenje nekog zadatka na jednom čvoru ima uticaj na izvršenje zadataka na ostalim čvorovima računarskog *Grid*-a, odnosno da algoritam raspodele optimizuje *makespan* celokupne kolekcije tokova aktivnosti. Nakon obrade nivoa višeg prioriteta, prelazi se na nivo nižeg prioriteta, itd.



Slika 6.4 – Zadaci izdeljeni po nivoima prioriteta za jedan čvor računarskog Grid-a

Na primeru sa slike 6.4 algoritam raspodele se prvo primenjuje na zadatke T_1, T_2 i T_3 jer im je dodeljen najviši nivo prioriteta. Cilj je da se odredi optimalan redosled njihovog izvršavanja tako da se ukupno vreme izvršavanja svih zadataka koji čine usmereni aciklični graf u datom trenutku, umanji u maksimalnoj meri. Nakon obrade zadataka kojima je dodeljen prioritet p_1, prelazi se na obradu zadataka koji su prioriteta p_2, (zadaci T_4 i T_5), a nakon toga na obradu zadataka koji su prioriteta p_3 (zadaci T_6 i T_7).

U zavisnosti od specifičnih osobina računarskog *Grid*-a i karakteristika zadataka u toku procesa raspodele zadataka mogu se primeniti različiti algoritmi. U prikazanoj arhitekturi, zamena algoritma raspodele zadataka odvija se na krajnje jednostavan način. Za zamenu je potrebno izmeniti definiciju algoritamskog parametra, odnosno proceduru na koji način se on računa za svaki od spremnih zadataka. Zadaci se i dalje sortiraju po izračunatom algoritamskom parametru u okviru istog nivoa prioriteta. Zbog toga preostali deo sistema za upravljanje tokovima aktivnosti ostaje identičan. Prilikom izbora algoritma neophodno je uzeti u obzir i vreme koje je potrebno da se primenom algoritma donese odluka. Na taj način se postiže ravnoteža između vremena primene algoritma i ukupnog vremena izvršenja zadataka.

Imajući u vidu da se u toku primene algoritma raspodele zadaci sortiraju u okviru istog nivoa prioriteta, moguća je njegova implementacija uz naknadnu optimizaciju, čime se dodatno skraćuje vreme potrebno za preraspodelu spremnih zadataka. Na primer, ukoliko je došlo do pojave novih spremnih zadataka koji treba da se izvršavaju na nekom čvoru računarskog *Grid*-a, a koji utiču samo na pojedine nivoe prioriteta prethodno dodeljene tom čvoru, nije neophodno izvršiti preraspodelu zadataka na svim nivoima,

već samo na onima kojih se ova izmena tiče. Na primer, neka se pretpostavi da su u početnom trenutku sva tri nivoa prioriteta koji su prikazani na slici 6.4 sortirana po algoritamskom parametru i da je potrebno uključiti novi zadatak T_8 čiji je prioritet p_1. Očigledno je da se zadatak T_8 uključuje u nivo najvišeg prioriteta, pa je potrebno uraditi preraspodelu zadataka jedino tog nivoa. Preostala dva nivoa prioriteta se izostavljaju iz preraspodele.

Nakon primene algoritma raspodele nad spremnim zadacima, sistem za upravljanje tokovima aktivnosti proverava koji su sve čvorovi računarskog *Grid*-a od interesa za izvršenja pomenutih zadataka. Može se dogoditi da na nekom čvoru ne postoji ni jedan zadatak za izvršenje. Na primer, takva situacija se javlja prilikom inicijalnog unosa podataka kada se ne vrše nikakvi proračuni, pa je SZP neopterećen. Za sve čvorove od interesa koji su raspoloživi biraju se spremni zadaci najvišeg prioriteta koji su dobili prednost u odnosu na sve zadatke u okviru istog nivoa prioriteta. Ti zadaci se šalju u red za raspodelu. Dalje ih preuzima komponenta za neposrednu raspodelu zadataka i šalje ih na izvršenje.

6.3 OTPORNOST NA GREŠKE

Otpornost sistema na greške (eng. *fault tolerance*) predstavlja sposobnosti sistema da obavlja svoju funkciju pravilno čak i u prisustvu neispravnosti [122]. Pod pojmom neispravnost smatra se neuspešno izvršenje nekog zadataka (iz bilo kog razloga) [123]. Stepen otpornost sistema na greške definisan je merom njegove pouzdanosti – osobinom sistema da pruži usluge za koje je namenjen. Pouzdanost sistema se meri pomoću dve metrike: verodostojnost i raspoloživost [114]. Verodostojnost je osobina sistema da obavlja svoja zaduženja korektno, dok je raspoloživost opisana kao mogućnost sistema da pruži uslugu u datom trenutku, kada se to od njega zahteva.

Verodostojnost sistema je definisana kao verovatnoća da će sistem korektno funkcionisati do datog vremenskog trenutka. Ona je vezana za prosečno vreme pojave dve uzastopne greške *mtbf* :

$$mtbf = mtf + mtr\ , \tag{6.1}$$

gde je sa *mtf* naznačeno prosečno vreme kada sistem funkcioniše korektno između greški, a sa *mtr* vremenski period potreban da se sistem oporavi od greške.

Raspoloživost sistema je definisana kao verovatnoća da je sistem operativan u željeno vreme. Ova osobina direktno zavisi od vremena opravka sistema nakon pojave greške:

$$raspolozivost = \frac{mtf}{mtbf}\ . \tag{6.2}$$

Da bi se uvećala otpornost na greške i time postigao porast pouzdanost distribuiranih sistema koji se postavljaju na računarski *Grid*, razvijene su različite tehnike [124].

Treba istaći da dopunski pristup za povećanje pouzdanosti nekog sistema predstavlja prevencija od grešaka. Dopunski pristup koristi različite tehnike čija je namera da eliminišu okolnosti koje dovode do grešaka. Prethodno je opisano da sistem za upravljanje DMS tokovima aktivnosti obezbeđuje predefinisani skup validacija prilikom prihvatanja nekog toka aktivnosti, pa su preventivne mere zaštite prisutne.

Shodno važnosti posla koji obavlja, neophodno je preuzeti sve mere da bi se obezbedio pouzdan rad DMS-a. Sistem treba da je dizajniran tako da njegovo korektno funkcionisanje bude garantovano uprkos greškama koje se u toku rada mogu javiti na resursima računarskog *Grid*-a [124]. Rad DMS-a ne sme da se blokira usled potencijalnih grešaka, odnosno neuspešnog izvršenja nekog zadatka. Zato se, u okviru DMS-a, otpornost na greše realizuje sa ciljem da do neuspešnog izvršenja zadataka nikada ne dođe. Međutim,

ukoliko u nekim izvanrednim okolnostima do ovakve pojave ipak dođe, potrebno je pravilno obraditi i takve situacije.

6.3.1 REPLIKACIONA TEHNIKA

Replikacija je jedna od tehnika za postizanje adekvatne otpornosti na greške u distribuiranim sistemima i povećanje njihove pouzdanosti. Ona je zasnovana na pretpostavci da je bilo koji pojedinačni računarski resurs znatno podložniji neuspehu prilikom izvršavanja nekog zadatka u poređenju sa izvršavanjem istog zadatka na više različitih računarskih resursa [125, 126].

Prilikom upotrebe replikacione tehnike dolazi do transfera podataka i definicija zadataka između distribuiranog računarskog resursa i njegovih replika. Cilj implementacije replikacione tehnike je da osigura identično stanje svake replike nekog distribuiranog računarskog resursa. Iz tog razloga razvijene su sledeće strategije implementacije:

- *Pasivna replikacija* razlikuje dva vrste distribuiranih računarskih resursa: primarni računarskih resurs i njegove replike. Primarni računarskih resurs izvršava sve zadatke. U slučaju greške na primarnom računarskom resursu, jedna od replika preuzima izvršavanje zadatka, odnosno postaje novi primarni računarski resurs [127].
- *Delimično aktivna replikacija* se implementira tako što i primarni računarski resurs i replike primaju zadatak. Zadatak se izvršava nezavisno na svakom računarskom resursu, ali se samo sa primarnog računarskog resursa šalju rezultati.
- *Aktivna replikacija* se odnosi prema svim računarskim resursima na isti način. Svaki računarski resurs prima zadatak i svaki od njih izvršava zadatak nezavisno od ostalih [128]. Ovakav pristup je pogodan za unapređenje performansi čitavog sistema. Ukoliko su računarski resursi organizovani tako da se prilikom istovremenog izvršavanja zadatka pošalje odgovor računarskog resursa koji je prvi izvršio zadatak jasno je da će se dodatno umanjiti *makespan* kolekcije tokova aktivnosti.

Pored implementacije replikacione tehnike koja definiše kako će se ponašati replike nekog distribuiranog računarskog resursa, postoje i različiti pristupi koji određuju način dodele računarskih resursa nekom zadatku u toku replikacije. *Statička replikacija* podrazumeva da se pre izvršenja zadatka odredi na kom skupu računarskih resursa se zadatak može izvršavati. Ukoliko ne uspe da se izvrši ni na jednom od njih, ne postoji dodatni računarski resurs koji bi ih zamenio. Prilikom *dinamičke replikacije*, za zadatak koji nije korektno izvršen na prethodno dodeljenim računarskim resursima mogu se obezbediti dodatni računarski resursi, kojima će se taj zadatak dodeliti. Ovakav pristup je moguć ako se zadaci mogu izvršavati na više različitih računarski resursa. Tada se, po potrebi, zadatak dodeli neuposlenom računarskom resursu iz predefinisanog skupa. Očigledno da je implementacija dinamičke replikacije znatno komplikovanija. Naravno, moguća je i kombinacija dve prethodno opisane implementacije [129].

Da bi se u što većoj meri minimizovao *makespan* kolekcije tokova aktivnosti, računarski *Grid* koristi aktivnu implementaciju replikacije. Međutim, integralni zadaci tokova aktivnosti koji se javljaju u DMS-u imaju predefinisan DMS servis, a time i čvor računarskog *Grid*-a na kome se izvršavaju. Posmatrajući povezanost DMS arhitekture i računarskog *Grid*-a jasno je da je neophodna upotreba statičke replikacije. Uobičajeno je da svaki čvor ima jednu redundantnu repliku koja radi istovremeno nad istim skupom podatka. Ako se želi postići još veći stepen pouzdanosti i ukoliko raspoloživa finansijska sredstava to dozvoljavaju, za neki čvor računarskog *Grid*-a se može definisati više replika.

Infrastruktura računarskog *Grid*-a ima zadatak da tehničke detalje opisanog mehanizma replikacije

sakrije od ostatka sistema i tako pojednostavi njegovu implementaciju. Ona je zadužena da se podaci u svakom trenutku nađu na pravoj lokaciji (računarskom resursu), odnosno da podaci stignu kako do primarnog čvora tako i do njegovih replika. Sistem za upravljanje tokovima aktivnosti nije svestan postojanja replika nekog čvora, tako da smatra da se obratio odgovarajućem DMS servisu. On očekuje odgovor i nije zainteresovan za informaciju da li je odgovor stigao sa primarnog čvora računarskog *Grid*-a ili njegove replike. Komponenta za neposrednu raspodelu zadataka je deo sistema za upravljanje tokovima aktivnosti. Ta komponenta je zadužen za interakciju sa računarskim Grid-om i detaljno je predstavljena u nastavku rada.

6.3.2 NEPOSREDNA RASPODELA ZADATAKA

Komponenta za neposrednu raspodelu zadataka na čvorove računarskog *Grid*-a zadužena je za više funkcionalnosti:

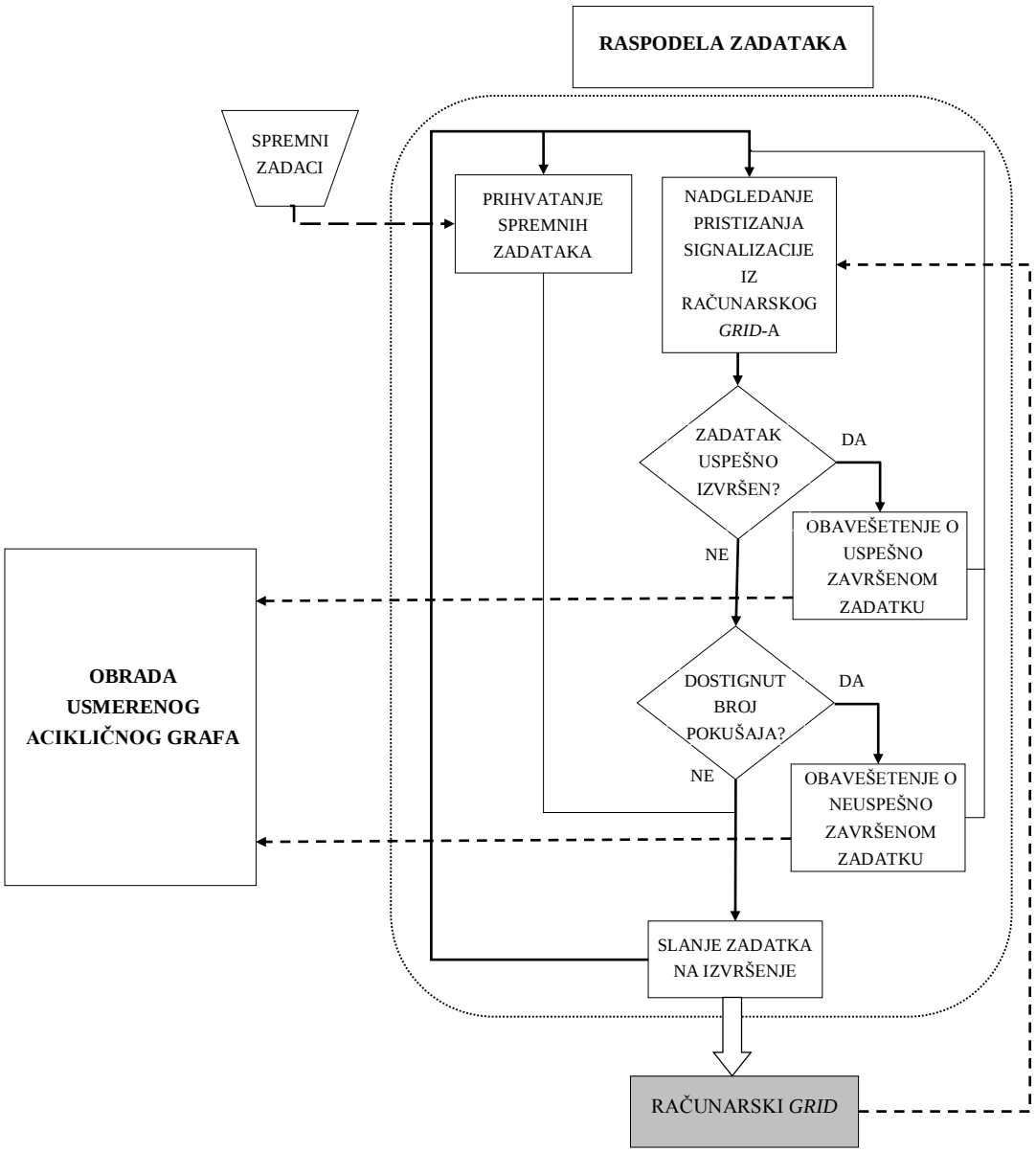
- slanje zadataka na čvorove računarskog *Grid*-a;
- dodatna manipulacija zadacima usled grešaka prilikom izvršenja;
- obaveštavanje komponente za obradu usmerenog acikličnog grafa o (uspešno ili neuspešno) završenom zadatku.

Detaljniji prikaz komponente za neposrednu raspodelu zadataka dat je na slici 6.5.

Da bi se podržalo slanje zadataka implementirana je sledeća jednostavna logika. Preuzima se prvi naredni zadataka iz reda spremnih zadataka i identifikuje se čvor računarskog *Grid*-a na kome zadatak treba da se izvrši. Zatim se kontaktira identifikovani čvor i šalje se opis zadatka. Istovremeno se startuje proces kojim se nadzire izvršenje zadatka. U nastavku, čvor računarskog *Grid*-a vrši autorizaciju korisnika koji je zahtevao izvršenje toka aktivnosti, čiji je razmatrani zadatak deo, i proverava da li je zahtevana akcija dozvoljena. Ukoliko korisnik ima odgovarajuća prava, zadatak se izvršava kada svi podaci potrebni za njegovo izvršenje, ukoliko već nisu, postanu dostupni.

Kako je prethodno opisano, robusnost DMS-a inicijalno je skrivena iza infrastrukture računarskog *Grid*-a. Međutim, preporučuje se da odgovarajuća podrška za manipulaciju greškama bude implementirana i u okviru sistema za upravljanje tokovima aktivnosti [124]. Ulaganja u infrastrukturu računarskog *Grid*-a su takva da se od njega očekuje da efikasno reši probleme koji se javljaju kada dođe do neuspešnog završetka zadataka. Uloga komponente za neposrednu raspodelu zadataka je da obrađuje nepredviđene situacije kada infrastruktura računarskog *Grid*-a iz nekog razloga nije u stanju da reši problem. Takve krajnje neuobičajene situacije, manifestuju se kao pojave da je došlo do gubitka komunikacije sa nekim čvorom računarskog *Grid*--a ili istovremeno određeni čvor i njegova replika nisu u stanju da izvršavaju zadatke. Očigledno je da ovakvo ponašanje infrastrukture računarskog *Grid*-a predstavlja pravu retkost.

Podršku za rukovanje greškama prilikom izvršavanja zadataka predstavljaju procedure koje definišu koliko dugo može da se čeka na odgovor o uspešnosti izvršenja nekog zadatka, kao i koliko puta neuspešno završen zadatak može biti poslat na ponovno izvršavanje. Ukoliko odgovor o uspešnosti izvršenja zadatka nije stigao u predefinisanom roku, smatra se da je zadatak neuspešno završen [130]. Sa druge strane, ukoliko signalizacija o završetku zadatka stigne na vreme proverava se kakvi su rezultati njegovog izvršenja.



Slika 6.5 – Komponenta za neposrednu raspodelu zadataka na čvorove računarskog Grid-a

Ukoliko je zadatak uspešno izvršen, obaveštava se komponenta za obradu usmerenog acikličnog grafa kako bi ona nastavila svoj rad i uklonila zadatak iz grafa. Ako je zadatak neuspešno izvršen, ili je istekao predefinisani dozvoljeni vremenski interval, komponenta za neposrednu raspodelu zadataka poverava da li zadatak može biti poslat na ponovno izvršenje, odnosno da li je dostignut dozvoljen broj pokušaja da se zadatak ponovo pošalje na izvršenje. U slučaju da je ova granica dostignuta, obaveštava se komponenta za obradu usmerenog acikličnog grafa o neuspešno završenom zadatku. Ako granica nije dostignuta, ponavlja se slanje zadatka na odgovarajući čvor računarskog *Grid*-a. Ovakav koncept oporavka zadatka ne opterećuje komponentu za obradu usmerenog acikličnog grafa (ta komponenta nema informacije da li se i koliko puta pokušalo sa ponovnim izvršavanjem zadatka). Na taj način je omogućeno da se komponenta za obradu usmerenog acikličnog grafa neometano usresredi na analizu grafa i primenu algoritma raspodele.

Ukoliko se zadatak neuspešno završi, komponenta za obradu usmerenog acikličnog grafa prima obaveštenje o tome, pa mora da reaguje na adekvatan način. U tom slučaju, jedina korektna akcija je da se pomenuti zadatak i kompletan skup njegovih sledbenika (svi zadaci koji od njega direktno ili indirektno zavise) uklone iz grafa. Svi tokovi aktivnosti kojima su ti zadaci pripadali se proglašavaju neuspešno

izvršenim. Ukoliko u narednom periodu pristigne neki tok aktivnosti koji zavisi od ovih, prethodno proglašenih neuspešnim, označava se kao nevalidan, i odbacuje se.

7. KONCEPT CENTRALIZOVANE RASPODELE ZADATAKA

Odgovornost za donošenje globalnih odluka vezano za raspodelu zadataka u dinamičkom računarskom okruženju može biti dodeljena samo jednoj centralnoj komponenti. Ovakav koncept naziva se *centralizovana raspodela zadataka*.

Prilikom upotrebe centralizovane raspodele zadataka, jedinstvena centralna komponenta donosi odluke o raspodeli svih zadataka koji čine usmereni aciklični graf na čvorove računarskog *Grid*-a [131]. Ova komponenta poseduje informacije o kompletnoj kolekciji tokova aktivnosti, a pri tome prikuplja informacije iz računarskog okruženja koje su vezane za izvršavanje zadataka. Kako se sve potrebne informacije nalaze na jednom mestu, njihovom analizom je omogućena efikasna raspodela zadataka čija je realizacija relativno jednostavna. Ipak, sa druge strane, smatra se da ovakav koncept nije dovoljno skalabilan u pogledu broja zadataka ili čvorova računarskog *Grid*-a. Tako da ovakav koncept ponekad može predstavljati usko grlo sistema, npr. ako je usled greške na računarskoj mreži centralna komponenta ostala odsečena od čvorova računarskog *Grid*-a [132].

Prilikom implementacije algoritama koji se oslanjaju na usmereni aciklični graf važno pitanje je kako da se rangiraju čvorovi grafa, odnosno zadaci, i kako da se odredi redosled prilikom njihove dodele nekom resursu računarskog *Grid*-a. U toku ovog procesa, u cilju minimizacije *makespan*-a, potrebno je voditi računa o sledeća dva problema [133, 134]:

- kako parelelizovati izvršavanje zadataka za koje strukturom usmerenog acikličnog grafa nije definisan međusobni redosled;
- kako omogućiti da kompletno izvršenje kritične putanje usmerenog acikličnog grafa bude najkraće moguće; kritična putanja usmerenog acikličnog grafa definisana je kao vremenski "najduža" putanja izvršenja, od nekog ulaznog do nekog izlaznog zadatka [103, 135].

Da bi se resursi računarskog *Grid*-a optimalno iskoristili, algoritmi raspodele mogu da koristite različite informacije koje usmereni aciklični graf obezbeđuje. Odluka o dodeli nekog zadatka čvoru računarskog *Grid*-a može da se donese na osnovu informacija vezanih samo za jedan zadatak (eventualno uključujući i informacije vezane za njegovu blisku okolinu u grafu) ili na osnovu informacija iz celokupnog usmerenog acikličnog grafa. Ova dva načina donošenja odluka se nazivaju lokalno i globalno donošenje odluka u vezi sa raspodelom određenog zadatka [76]. *Lokalno donošenje odluka* treba pažljivo koristiti jer je moguće obezbediti efikasno izvršenje pojedinačnih zadataka, dok se uvećava vreme izvršavanja celokupnog usmerenog acikličnog grafa. Sa druge strane, s obzirom da *globalno donošenje odluka* koristi znatno veću količinu informacija, često je za realizaciju analize potreban znatno duži vremenski period. Zbog toga je prilikom izbora načina donošenja odluka potrebno uzeti u obzir ravnotežu između vremena koje je potrebno da se odgovarajući algoritam izvrši i celokupnog vremena izvršavanja svih zadataka usmerenog acikličnog grafa (tj. *makespan*-a kolekcije tokova aktivnosti).

Kako skup tokova aktivnosti koji unutar DMS-a treba da se izvrše nije poznat unapred, jer tokovi aktivnosti neprestano pristižu, pre početka procesa raspodele informacije o svim zadacima nisu u potpunosti poznate. Zbog toga odluke vezane za dodelu resursa računarskog *Grid*-a određenom zadatku

moraju da se donose u hodu, odnosno neophodni su algoritmi koji podržavaju dinamičku raspodelu zadataka. Nakon obrade usmerenog acikličnog grafa i izdvajanja spremnih zadataka na raspolaganju su međusobno nezavisni zadaci, kategorisani po čvorovima računarskog *Grid*-a i nivoima prioriteta. Algoritmi raspodele razmatraju zadatke po svakom čvoru, kao i u okviru čvora nivo po nivo, počevši od nivoa najvišeg prioriteta. Svakom od spremnih zadataka dodeljuje se algoritamski parametar. Algoritamski parametar služi kako bi se odredio redosled izvršavanja zadataka u okviru istog nivoa prioriteta.

7.1 NAJRANIJI START SVIH ZADATAKA NEPOSREDNIH SLEDBENIKA

Prvi algoritam raspodele zadataka, koji je razvijen u ovom istraživanju zasniva se na lokalnom donošenju odluka. Da bi se njegovom primenom izvršila optimizacija, algoritam uzima u obzir prirodu DMS-a, prvenstveno količinu podataka kojom sistem raspolaže.

Imajući u vidu da je između čvorova računarskog *Grid*-a potrebno razmeniti velike količine podataka (podaci koji se razmenjuju između zadataka i njihovih neposrednih sledbenika), baš ta razmena nosi značajan udeo u *makespan*-u kolekcije tokova aktivnosti. Iz tog razloga prilikom odabira redosleda izvršenja zadataka potrebno je i razmenu podataka uključiti u analizu. To znači da nije dovoljno proceniti kada će se neki zadatak završiti, već i kada će rezultati njegovog rada biti dostupni ostalim zadacima.

Algoritam se zasniva na ideji da se primarno ostvari raspodela zadataka koji će dovesti do mogućnosti početka izvršenja svih svojih neposrednih sledbenika u najkraćem roku. Pitanje na koje algoritam daje odgovor je: *koji od spremnih zadataka treba da bude završen kako bi se što pre omogućilo izvršenje svih njegovih neposrednih sledbenika?* Na ovaj način je obezbeđen raniji početak izvršenja zadataka, odnosno sužava se prostor da neki zadatak pristigne na čvor računarskog *Grid*-a radi izvršenja, a da pri tome mora da čeka na ulazne podatke koji predstavljaju rezultat izvršenja njegovog neposrednog prethodnika. Pri tome se kompletan skup spremnih zadataka uvećava, više zadataka konkuriše za izvršenje, pa postoji veći izbor zadataka kada se donosi odluka koji naredni zadatak treba da bude izvršen i na taj način se postigne značajno smanjenje *makespan*-a.

Za svaki spreman zadatak T_i računa se algoritamski parametar $succst_{T_i}$. On predstavlja procenjeno vreme kada će svi neposredni sledbenici zadatka T_i biti spremni za izvršenje i svi njihovi ulazni podaci dostupni. Računa se kao suma procenjenog vremena završetka zadatka T_i i maksimalnog procenjenog vremena potrebnog da rezultat izvršenja T_i bude dostupan na svim čvorovima računarskog *Grid*-a na kojima treba da se izvrše njegovi neposredni sledbenici:

$$succst_{T_i} = ct_{T_i} + \max_{T_j \in S_{T_i}} \{c_{T_i T_j}\}. \tag{7.1}$$

Iz tog proizilazi da za izlazne zadatke kolekcije tokova aktivnosti kao posledica izraza (5.4) važi:

$$succst_{T_i} = ct_{T_i}. \tag{7.2}$$

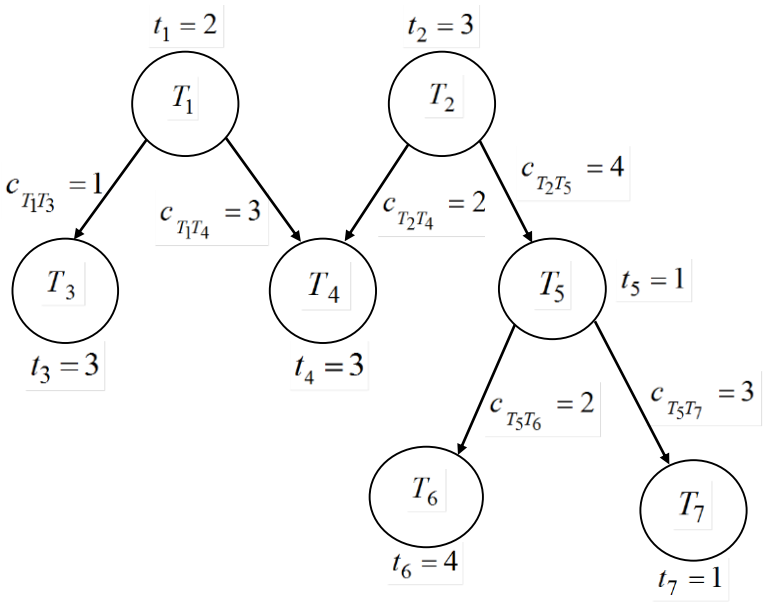
Tokom odluke o redosledu izvršavanja zadataka, za svaki čvor računarskog *Grid*-a algoritam razmatra spremne zadatke pojedinačno za svaki nivo prioriteta koji je vezan za dati čvor. Na taj način se izračunava *succst* za svaki od njih. Prilikom dodele odgovarajućeg resursa računarskog *Grid*-a, zadatak koji se nalazi na nivou najvišeg prioriteta, a ima minimalnu vrednost za *succst*, dobija prednost u odnosu na ostale zadatke. Očekuje se da će se manji *makespan* ostvariti ako se prvobitno izvrše zadaci koji

dozvoljavaju svojim neposrednom sledbenicima da što pre počnu izvršenje i tako ne gube dragoceno vreme čekajući da im pristignu potrebni ulazni podaci. Nakon što se odrede zadaci koji imaju prednost na pojedinim čvorovima računarskog *Grid*-a, isti se šalju na izvršenje čime se menja statusa raspoloživosti čvora.

Na slici 7.1 je dat primer kolekcije tokova aktivnosti. Prikazana su procenjena vremena izvršenja svakog od zadataka kao i procenjena vremena koja su potrebna za prenos međupodataka kako bi se mogli izvršavati neposredni sledbenici. Da bi primer bio jasniji, pojednostavljene su okolnosti uz sledeće pretpostavke:

- Svi zadaci su istog prioriteta. Na ovaj način prioritet neće imati presudnu ulogu prilikom odabira nekog zadataka radi izvršenja.
- Zadaci T_1 i T_2 se izvršavaju na istom čvoru računarskog *Grid*-a.
- Ni jedan ciklus primene algoritma raspodele se do sada nije dogodio, što znači da su svi čvorovi računarskog *Grid*-a raspoloživi.

Nad prikazanom kolekcijom tokova aktivnosti potrebno je primeniti algoritam najranijeg starta svih zadataka neposrednih sledbenika.



Slika 7.1 – Primer kolekcije toka aktivnosti

Prvi korak primene algoritma je analiza usmerenog acikličnog grafa u cilju provere da li postoje spremni zadaci. Zadaci T_1 i T_2 nemaju prethodnike i spremni su za izvršenje, što znači da procenjeno vreme kada će se svaki od njih završiti, ukoliko su im računarski resursi dostupni, odgovara procenjenom vremenu njihovog izvršenja:

$$ct_{T_1} = 2, \qquad ct_{T_2} = 3. \tag{7.3}$$

S obzirom da su oba zadatka istog prioriteta i da treba da se izvrše na istom čvoru računarskog *Grid*-a koji je trenutno raspoloživ, ostaje pitanje koji od zadataka će dobiti prednost. Na osnovu usmerenog acikličnog grafa vrednosti algoritamskih parametara za zadatke T_1 i T_2 su:

$$succst_{T_1} = ct_{T_1} + \max\{ c_{T_1T_3}, c_{T_1T_4} \} = 2 + \max\{1,3\} = 5, \tag{7.4}$$

$$succst_{T_2} = ct_{T_2} + \max\{ c_{T_2T_4}, c_{T_2T_5} \} = 3 + \max\{ 2, 4 \} = 7 . \quad (7.5)$$

S obzirom da je $succst_{T_1} < succst_{T_2}$, zadatak T_1 će dobiti prednost prilikom dodeljivanja raspoloživog resursa računarskog *Grid*-a.

7.2 KRITIČNA PUTANJA ZADATAKA

Novi algoritam raspodele zadataka koji je implementiran u ovom radu koristi centralizovan pristup i globalni način donošenja odluka, odnosno oslanja se na informacije iz celokupnog usmerenog acikličnog grafa kako bi se donela odluka o redosledu slanja zadataka na izvršenje. Ovaj algoritam se zasniva na ideji da se izvršenje zadataka usmeri ka delu kolekcije tokova aktivnosti koji zahteva najveće opterećenje resursa računarskog *Grid*-a.

Algoritam se oslanja na koncept *kritične putanje kolekcije tokova aktivnosti*. Za svaki od zadataka u usmerenom acikličnom grafu se vezuje jedna ili više putanja izvršenja (zadataka), u zavisnosti od strukture grafa. Putanja izvršenja nekog zadatka se definiše kao niz zadataka (čvorova grafa) i predefinisanih akcija prenosa podataka (grana grafa) između tih zadataka koji su usmereni od tekućeg zadatka do nekog izlaznog zadataka. Na primer, za zadatak T_2 iz usmerenog acikličnog grafa sa slike 7.1, jedna od putanja izvršenja kreće od čvora T_2, nakon čega se granom $c_{T_2T_5}$ dolazi do čvora T_5, a posle toga sa granom $c_{T_5T_7}$ do čvora T_7. Ova putanja izvršenja je kraće označena sa $T_2 \rightarrow T_5 \rightarrow T_7$.

Vrednost putanje izvršenja je određena zbirom procenjenog vremena izvršenja svih zadataka na putanji i procenjenog vremena prenosa podataka za svaku akciju. Kritična putanja izvršenja zadatka je "najduža" od svih putanja njegovog izvršenja, odnosno putanja sa najvećim procenjenim vremenom.

Ovaj pristup ima za cilj da utvrdi najdužu od svih putanja izvršenja zadataka od ma kog spremnog do nekog izlaznog zadatka u usmerenom acikličnom grafu. Odnosno, da odgovori na pitanje: *koji zadaci imaju najveći uticaj na ukupno vreme izvršenja kolekcije tokova aktivnosti?*

Nakon što su spremni zadaci kategorisani po čvorovima računarskog *Grid*-a i nivoima prioriteta, algoritam identifikuje spremne zadatke (vezane za isti čvor i u okviru istog nivoa prioriteta) koji u okviru kolekcije tokova aktivnosti blokiraju najdugotrajnije aktivnosti. Identifikovanim zadacima se dodeljuje prednost u odnosu na ostale spremne zadatke, kako bi se prvi poslali na izvršenje. Na taj način minimizuje se vreme izvršenja celokupne kolekcije tokova aktivnosti. Ideja je da se u toku primene algoritma "napadne" kritična putanja usmerenog acikličnog grafa u svakom koraku.

Dakle, algoritamski parametar koji se izračunava za svaki spreman zadatak je vrednost njegove kritične putanje izvršenja (*rang*). Na osnovu vrednosti ranga, sortirani su spremni zadaci vezani za isti čvor računarskog *Grid*-a i isti nivo prioriteta.

S obzirom da se rang računa kao vrednost kritične putanje izvršenja jasno je da rang izlaznih zadataka odgovara njihovom vremenu izvršenja:

$$\underset{T_i \in Outputs}{rank_{T_i}} = t_i . \quad (7.6)$$

Rang preostalih zadataka najjednostavnije se računa rekurzivno. Na taj način obezbeđeno je da se, u okviru kolekcije tokova aktivnosti, na osnovu jednog prolaska kroz usmereni aciklični graf odrede vrednosti ranga za sve spremne zadatke:

$$rank_{T_i} = t_i + \max_{T_j \in S_{T_i}} \{c_{T_i T_j} + rank_{T_j}\}, \tag{7.7}$$

gde je sa S_{T_i} naznačen skup svih neposrednih naslednika zadataka T_i. Zadatak višeg ranga ima prednost prilikom dodele resursa računarskog *Grid*-a.

Primena algoritma Kritične putanje zadataka prikazana je na primeru iste kolekcije tokova aktivnosti kao u odeljku 7.1, slika 7.1. Na taj način je istaknuta razlika rezultata koji su dobijeni primenom algoritma Kritične putanje zadatka u odnosu na rezultat primene algoritma Najranijeg starta svih zadataka neposrednih sledbenika. Napomena: važe iste pretpostavke koje su navedene u odeljku 7.1.

U toku primene algoritma Kritične putanje zadataka neophodno je analizirati ceo usmereni aciklični graf. Da bi se odredio rang spremnih zadataka potrebno je uzeti u obzir sve njihove putanje izvršenja do izlaznih zadataka grafa. Kako bi se rang odredio u što kraćem vremenskom periodu neophodno je u jednom prolazu uraditi analizu svih čvorova i grana usmerenog acikličnog grafa. Zato se računanje ranga vrši rekurzivno. U prvom koraku je potrebno odrediti rangove izlaznih zadataka T_3, T_4, T_6 i T_7 (njihovi rangovi su jednaki procenjenim vremenima izvršenja):

$$rank_{T_3} = 3, \qquad rank_{T_4} = 3, \qquad rank_{T_6} = 4, \qquad rank_{T_7} = 1. \tag{7.8}$$

Daljom analizom usmerenog acikličnog grafa uočava se da zadatak T_5 ima dve putanje izvršenja. Jedna vodi do zadatka T_6, a druga do zadatka T_7 ($T_5 \to T_6$ i $T_5 \to T_7$). Kritična je ona putanja izvršenja koja ima maksimalnu "dužinu". Iz tog razloga njegov rang se računar rekurzivno na sledeći način:

$$rank_{T_5} = t_5 + \max\{c_{T_5T_6} + rank_{T_6},\ c_{T_5T_7} + rank_{T_7}\} = 1 + \max\{2+4,\ 3+1\} = 7. \tag{7.9}$$

U ovom primeru su od najvišeg interesa rangovi zadataka T_1 i T_2. Slično kao zadatak T_5, T_1 poseduje dve putanje izvršenja: $T_1 \to T_3$ i $T_1 \to T_4$. Zadatak T_2 ima tri putanje: $T_2 \to T_4$, $T_2 \to T_5 \to T_6$ i $T_2 \to T_5 \to T_7$. Uzevši u obzir da se rangovi računaju rekurzivno počevši od izlaznih zadataka, a putanje $T_2 \to T_5 \to T_6$ i $T_2 \to T_5 \to T_7$ imaju zajednički zadatak T_5 nije potrebno računati dužinu obe putanje, nego se jednostavno može iskoristiti prethodno izračunati rang zadatka T_5. Na taj način je za računanje ranga bilo kog spremnog zadatka dovoljno obezbediti samo jedan prolazak kroz usmereni aciklični graf. Konačno, rangovi zadataka T_1 i T_2 se računaju na osnovu jednačina:

$$rank_{T_1} = t_1 + \max\{c_{T_1T_3} + rank_{T_3},\ c_{T_1T_4} + rank_{T_4}\} = 2 + \max\{1+3,\ 3+3\} = 8, \tag{7.10}$$

$$rank_{T_2} = t_2 + \max\{c_{T_2T_4} + rank_{T_4},\ c_{T_2T5} + rank_{T_5}\} = 3 + \max\{2+3,\ 4+7\} = 14. \tag{7.11}$$

Zadatak T_2 ima najviši rang, odnosno izvršenje tog zadatka blokira aktivnosti koje najduže traju u usmerenom acikličnom grafu. Iz tog razloga, zadatak T_2 dobija prednosti u odnosu na zadatak T_1 prilikom dodele nekom resursu računarskog *Grid*-a.

Kada se uporede primeri upotrebe algoritama Kritične putanje zadataka i algoritma Najranijeg starta svih zadataka neposrednih sledbenika može se uočiti da će, u zavisnosti od primenjenog algoritma, različiti spremni zadaci dobiti prednost, odnosno sam tok izvršavanja pristiglih tokova aktivnosti će biti drugačiji. Na osnovu ove činjenice jasno je koliko algoritma raspodele zadataka utiče na performanse celokupnog DMS-a.

7.3 EKSPERIMENTALNI REZULTATI

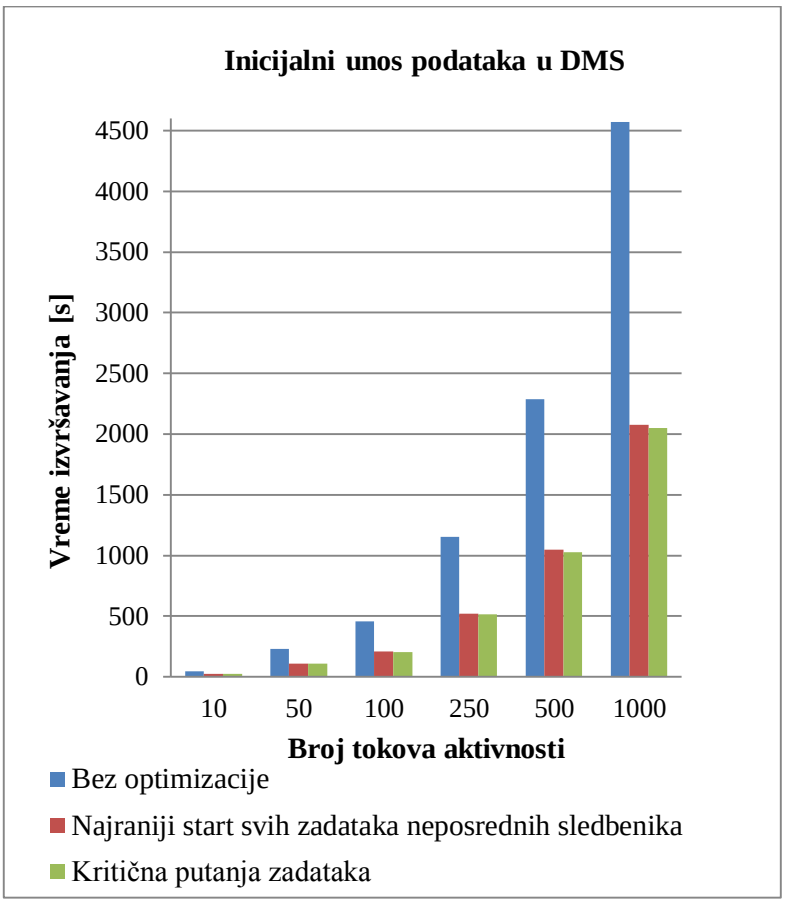
Razvijeno je distribuirano računarsko okruženje za testiranje čija je osnovna namena simulacija rada realnog DMS-a. Eksperimenti, koji su realizovani u cilju da se verifikuje efikasnost i praktična upotrebljivost razvijenih algoritama, podeljeni su u dve grupe: inicijalni unos podataka u DMS i upravljanje distributivnom mrežom. Za obe grupe eksperimenata mereno je vreme izvršenja kolekcije tokova aktivnosti (*makespan*) i ukupno vreme utrošeno na rad algoritma raspodele zadataka.

7.3.1 INICIJALNI UNOS PODATAKA U DMS

U okviru prve grupe eksperimenata, simuliran je inicijalni unos podataka u DMS koji se obično realizuje *offline*, pre nego što upravljanje distributivnom mrežom može da počne. Kolekcija tokova aktivnosti je sačinjena isključivo od tokova aktivnosti koji definišu modifikaciju modela distributivne mreže (svi pripadaju tipu Ažuriranje modela). Broj tokova aktivnosti, koji dinamički stižu radi obrade, varira u zavisnosti od veličine distributivne mreže koja se opisuje modelom – broja entiteta koje treba uneti u sistem (od nekoliko hiljada do nekoliko miliona). S toga se u eksperimentima koristi između deset i hiljadu tokova aktivnosti. U eksperimentima su korišćena tri nivoa prioriteta. Uporedni prikaz prethodno opisanih algoritama i sekvencijalnog načina obrade tokova aktivnosti dati su u tabeli 7.1 i na slici 7.2. Pod sekvencijalnom obradom tokova aktivnosti podrazumeva se izvršavanje tokova aktivnosti jedan za drugim, istim redosledom kojim su pristigli.

Tabla 7.1 – Vreme izvršavanja tokova aktivnosti pri inicijalnom unosu podataka u DMS

Broj tokova aktivnosti	Bez optimizacije – sekvencijalna obrada – [s]	Najraniji start svih zadataka neposrednih sledbenika [s]	Kritična putanja zadataka [s]
10	45.70	25.50	25.44
50	228.72	108.53	107.23
100	456.52	208.54	206.50
250	1154.37	522.77	513.87
500	2288.11	1048.65	1026.14
1000	4572.27	2075.71	2050.68

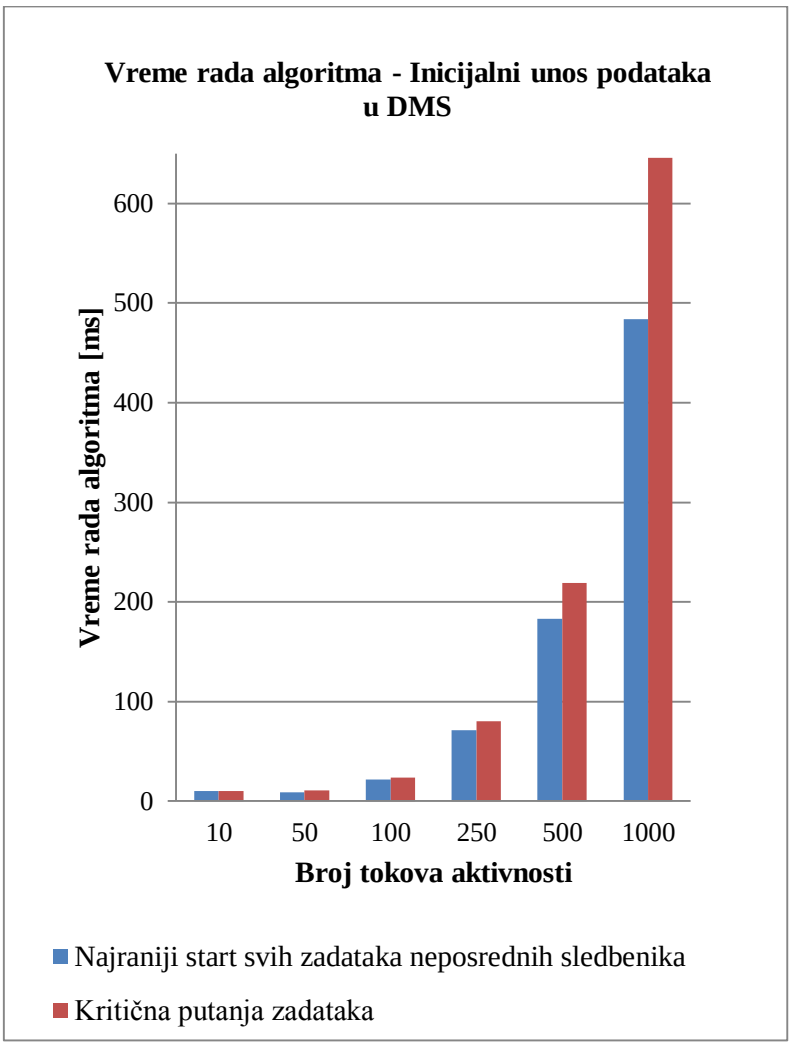


Slika 7.2 – Vreme izvršavanja tokova aktivnosti pri inicijalnom unosu modela distributivne mreže

Osim vrednosti *makespan*-a za kolekciju tokova aktivnosti, potrebno je proceniti koliko na tu vrednost utiče trajanja primene algoritma raspodele zadataka. Iz tog razloga u toku eksperimenata mereno je ukupno vreme koje sistem za upravljanje tokovima aktivnosti potroši na primenu algoritma raspodele zadataka. Dobijeni rezultati su prikazani u tabeli 7.2 i na slici 7.3 (izmerene vrednosti trajanja primene algoritma raspodele zadataka date se u milisekundama).

Tabla 7.2 – Vreme rada algoritma pri inicijalnom unosu podataka u DMS sistem

Broj tokova aktivnosti	Najraniji start svih zadataka neposrednih sledbenika [ms]	Kritična putanja zadataka [ms]
10	10	10
50	9	11
100	22	24
250	71	80
500	183	219
1000	484	646



Slika 7.3 – Vreme rada algoritma pri inicijalnom unosu modela distributivne mreže

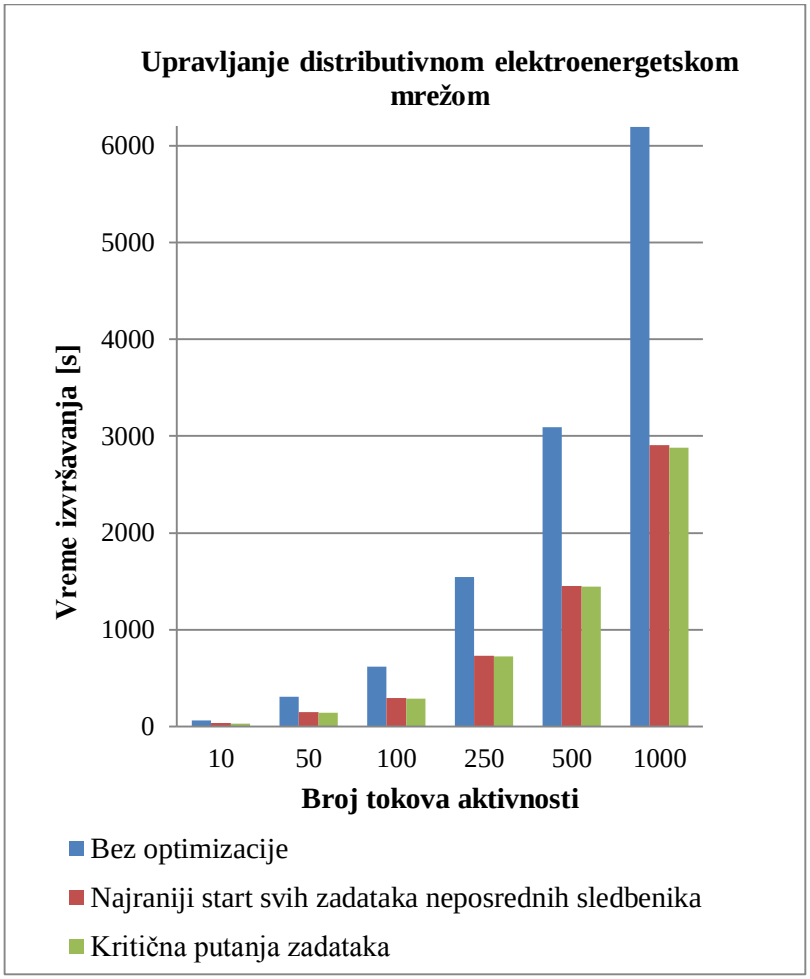
7.3.2 SIMULACIJA UPRAVLJANJA DISTRIBUTIVNOM MREŽOM

U okviru druge grupe eksperimenata simulirano je upravljanja distributivnom mrežom. Ovakav rad DMS-a je moguć nakon inicijalnog unosa opisa distributivne mreže. Kao što je prethodno objašnjeno, izmene vrednosti merenih veličina su veoma česte, tako da je izvršenje analitičkih funkcija najčešća aktivnost u DMS-u.

Kolekcija tokova aktivnosti se sastoji od sva tri tipa toka aktivnosti: Ažuriranje modela, Izvršenje DMS funkcije i Osvežavanje grafičkog prikaza. Izvršenje DMS funkcije je najčešći tip toka aktivnosti i predstavlja više od 70% svih tokova aktivnosti, sledi Ažuriranje modela distributivne mreže sa približno 20 %, dok je Osvežavanje grafičkog prikaza najređi tip toka aktivnosti sa manje od 10 % ukupnih tokova aktivnosti. Slično prvom skupu eksperimenata, i ovde se koristi između deset i hiljadu tokova aktivnosti. Dobijeni rezultati su prikazani u tabeli 7.3 i na slici 7.4.

Tabla 7.3 – Vreme izvršavanja tokova aktivnosti prilikom upravljanja distributivnom mrežom

Broj tokova aktivnosti	Bez optimizacije – sekvencijalna obrada – [s]	Najraniji start svih zadataka neposrednih sledbenika [s]	Kritična putanja zadataka [s]
10	62.03	36.89	34.05
50	310.35	148.01	146.33
100	618.62	293.37	288.58
250	1546.19	729.05	723.41
500	3093.00	1452.24	1442.90
1000	6192.01	2906.62	2881.85

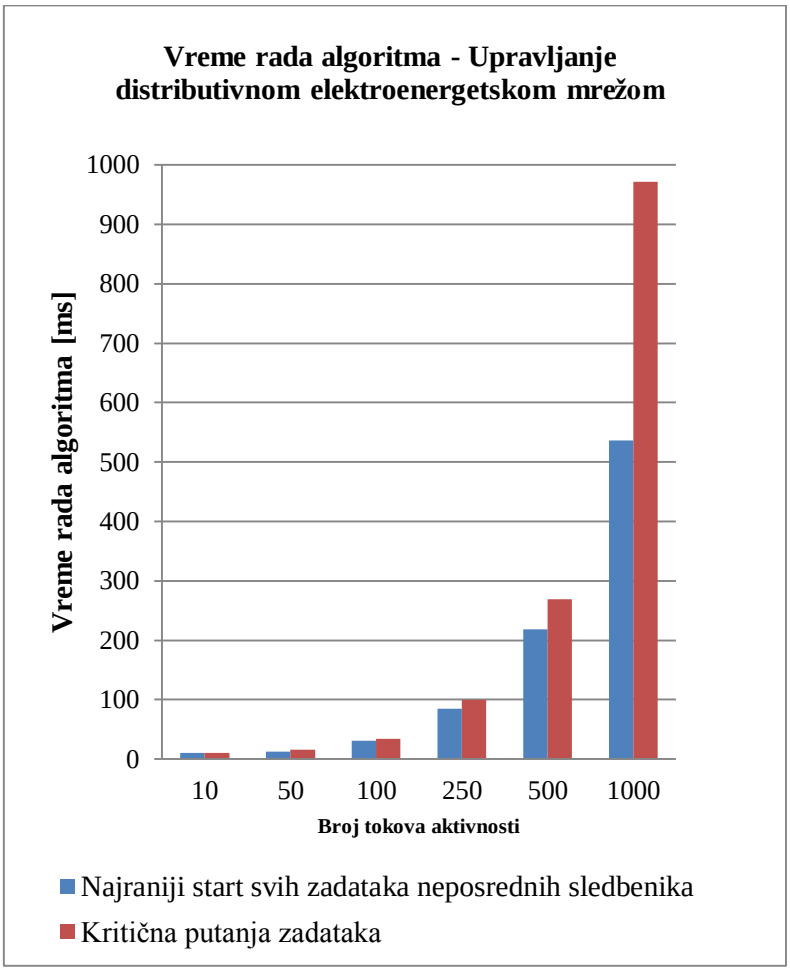


Slika 7.4 – Vreme izvršavanja tokova aktivnosti prilikom upravljanja distributivnom mrežom

Kao i u prethodnoj grupi eksperimenata koja se tiče modifikacije modela distributivne mreže, prilikom izvršavanja eksperimenata koji se odnose na upravljanje distributivnom mrežom vršena su merenja vezana za vreme utrošeno na primenu algoritma raspodele zadataka. Rezultati su prikazani u tabeli 7.4 i na slici 7.5.

Tabla 7.4 – Vreme rada algoritma pri upravljanju distributivnom mrežom

Broj tokova aktivnosti	Najraniji start svih zadataka neposrednih sledbenika [ms]	Kritična putanja zadataka [ms]
10	11	10
50	13	16
100	31	34
250	85	100
500	218	269
1000	536	971



Slika 7.5 –Vreme rada algoritma pri upravljanju distributivnom mrežom

Kao što se moglo očekivati, na osnovu prikazanih rezultata očigledno je da paralelizacija izvršavanja međusobno nezavisnih zadataka predstavlja prvi korak ka unapređenju iskorišćenja resursa računarskog *Grid*-a. Ukoliko se takav pristup unapredi primenom odgovarajućeg algoritma, dodatno će se popraviti performanse čitavog sistema.

Na osnovu prikazanih rezultata verifikacije prikazanih algoritama može se zaključiti sledeće:

- Proces raspodele zadataka koji koristi algoritam zasnovan na Kritičnoj putanji zadataka nadmašuje rešenje koje koristi Najraniji start svih zadataka neposrednih sledbenika. Objašnjenje za to se nalazi u činjenici da se prilikom određivanja Kritične putanje zadataka razmatraju svi putevi izvršenja

spremnih zadataka (ispituje se čitav usmereni aciklični graf), dok algoritam Najraniji start svih zadataka neposrednih sledbenika analizira samo blisko okruženje spremnog zadatka u usmerenom acikličkom grafu.

- Vreme potrebno za rad algoritma raspodele zadataka je izuzetno kratko u odnosu na ukupno vreme izvršenja svih tokova aktivnosti. Zbog toga vreme potrebno za rad algoritma nema značajniji uticaj na *makespan* kolekcije tokova aktivnosti
- Raspodela zadataka donosi više koristi, odnosno veću vremensku uštedu ukoliko postoji više različitih tokova aktivnosti čije izvršenje je potrebno koordinisati.
- Pozitivni efekti primene algoritama raspodele zadataka rastu sa porastom broja tokova aktivnosti. Njihovom primenom se dobija znatno bolje iskorišćenja resursa računarskog *Grid*-a u odnosu na iskorišćenje računarskih resursa kada je raspodela zadataka urađena bez optimizacije. Pri tome, primena Kritične putanje zadataka obezbeđuje bolje iskorišćenja resursa računarskog *Grid*-a u odnosu u na primenu Najranijeg start svih zadataka neposrednih sledbenika.

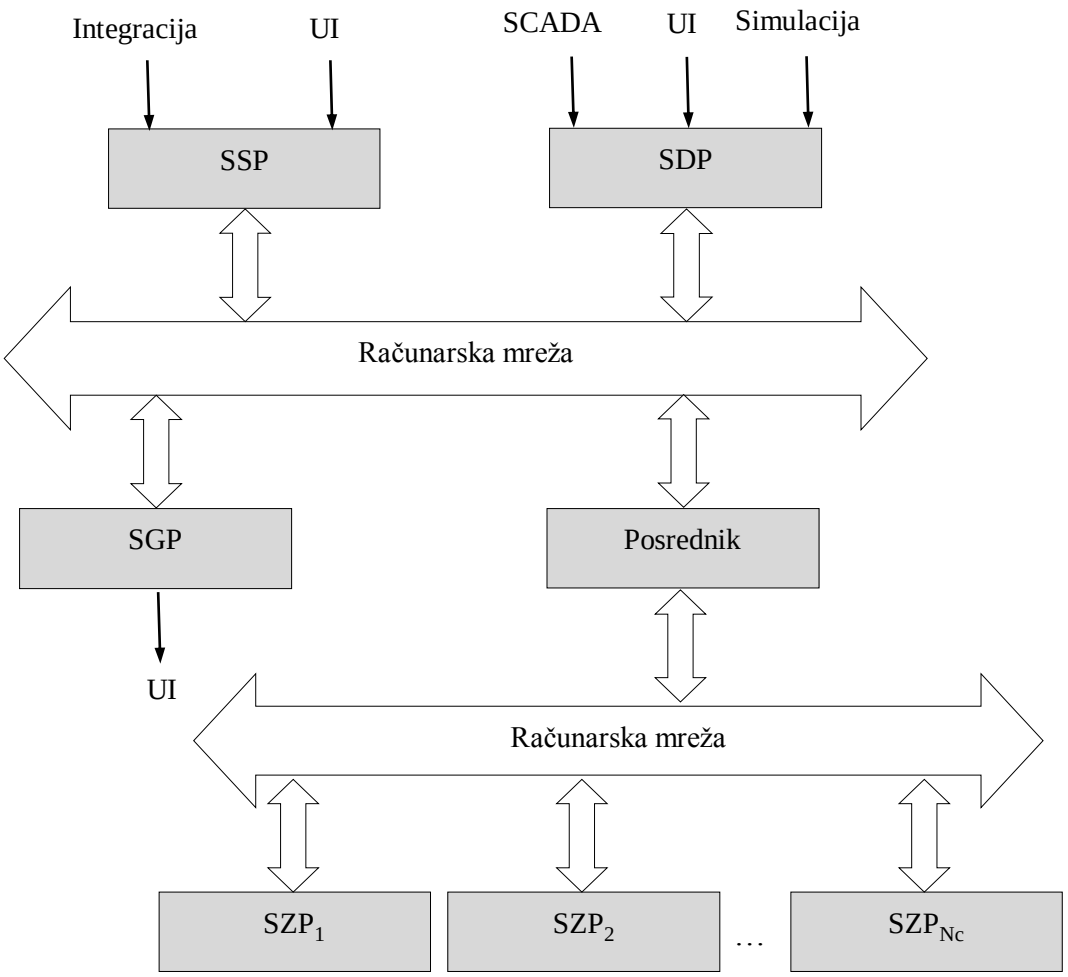
8. KONCEPT KOMBINOVANE RASPODELE ZADATAKA

U toku svakodnevnog rada DMS-a daleko najčešći tip tokova aktivnosti koji se javlja je Izvršenje DMS funkcije. Prilikom izvršavanja analitičkih funkcija uglavnom se obrađuje velika količina podataka, tako da je čvor za proračune (na kome je instaliran SZP, slika 3.1) veoma opterećen, odnosno može da predstavlja usko grlo sistema. Zato se u infrastrukturu računarskog *Grid*-a uvode dodatni čvorovi za proračune.

Do ovog momenta razmatrana je centralizovana raspodela zadataka. Prilikom upotrebe takvog koncepta, jedinstvena centralna komponenta koordiniše izvršavanje zadataka i ima uvid u sve čvorove računarskog *Grid*-a. Kada je potrebno promeniti broj čvorova i tako uvećati računarske kapacitete *Grid*-a, neophodno je izmeniti i centralnu komponentu kako bi ona postala svesna dostupnosti novih čvorova [105]. Da bi se izbegle izmene na centralnoj komponenti prilikom izmena skupa čvorova za proračune uvodi se kombinovana raspodela zadataka. Uvođenjem dodatnih čvorova za proračune i njihovom međusobnom kooperacijom, otvara se mogućnost dopunske distribucije opterećenja.

U zavisnosti od prirode distributivne mreže, različita je učestalost pristizanja tokova aktivnosti koji definišu Izvršenje DMS funkcije. To ima za posledicu različit broj proračuna (proračunskih zadataka) koje je potrebno izvršiti. Ukoliko bi infrastruktura računarskog *Grid*-a podržavala višestruke čvorove za proračun, bilo bi veoma pogodno da se njihov broj može jednostavno izmeniti bez većeg uticaja na ostatak sistema. Ovakva mogućnost bi na jednostavan način obezbedila željenu skalabilnost DMS-a u odnosu na veličinu distributivne mreže i frekvenciju izmena izmerenih vrednosti, koje predstavljaju glavne okidače za izvršenje proračuna. U skladu sa tim svaka elektrodistribucija koja poseduje DMS, mogla bi koristiti namenski računarski *Grid* koji obezbeđuje odgovarajući broj čvorova za proračune. Odnosno, ako je prvobitno procenjen broj čvorova za proračune nedovoljan, jednostavno se mogu uključiti novi čvorovi za proračune i time rasteretiti postojeći. Prethodno opisana unapređena infrastruktura računarskog *Grid*-a je prikazana na slici 8.1.

Napredni mehanizam za proračune obuhvata više čvorova računarskog *Grid*-a i uspostavlja čvor koji ima ulogu posrednika između njih i ostatka sistema – *posrednički čvor*. To znači da posrednički čvor ima zadatak da sakrije (promenljiv) skup čvorova za proračune od preostalih čvorova računarskog *Grid*-a i sistema za upravljanje tokovima aktivnosti. Takođe je algoritam raspodele proračunskih zadataka (proračuna) implementiran isključivo u okviru mehanizma za proračune, pa je nepoznat ostatku sistema. Svi ostali čvorovi računarskog *Grid*-a, kao i sistem za upravljanje tokovima aktivnosti (tačnije njegova komponenta za neposrednu raspodelu zadataka) komuniciraju isključivo sa posredničkim čvorom.



Slika 8.1 – Infrastruktura unapređenog računarskog Grid-a – mehanizam za proračune ima više čvorova

Infrastruktura računarskog *Grid*-a na slici 8.1 daje mogućnosti za dodatno unapređenje sveukupne koordinacije izvršavanja tokova aktivnosti. Potrebno je iskoristiti prednosti koje ona obezbeđuje. Zbog toga se sistem za upravljanje tokovima aktivnosti koji je predstavljen u šestom poglavlju unapređuje i tako podržava kombinovani koncept raspodele:

- Menja se raspodela proračunskih zadatka. Zadaci tog tipa se rasporede u prioritetne nivoe i za svaki od njih se izračuna algoritamski parametar (upotrebom jednog od centralizovanih algoritama opisanih u sedmom poglavlju) koji kasnije koristi raspodela zadataka implementirana na nivou mehanizma za proračune. Nakon određivanja algoritamskog parametra, zadaci se odmah šalju na posrednički čvor (za razliku od implementacije sistema za upravljanje tokovima aktivnosti prikazane u šestom poglavlju, na osnovu koje je predviđeno da se čeka da SZP postane raspoloživ). S obzirom da postoji više čvorova za proračune, a da je sistem za upravljanje tokovima aktivnosti svestan samo posredničkog čvora, smatra se da je od trenutka predaje zadatka, izvršenje tog zadatka briga mehanizma za proračune. Potrebno je istaći da preostali čvorovi računarskog *Grid*-a takođe nisu svesni broja čvorova za proračune, nego komuniciraju isključivo sa posredničkim čvorom i sa njim, po potrebi, razmenjuju međurezultate. Napominje se da raspodela zadataka koji se izvršavaju na preostala tri čvora računarskog *Grid*-a (SSP, SDP, SGP) ostaje nepromenjena.
- Bira se adekvatan algoritam za centralizovani deo raspodele, odnosno procenu algoritamskog parametara za svaki spreman zadatak. Centralizovani algoritmi opisani u sedmom poglavlju se zasnivaju na procenjenom vremenu izvršavanja zadataka, odnosno koriste te informacije kako bi

odredili vrednost algoritamskog parametra. U infrastrukturi računarskog *Grid*-a prikazanoj na slici 8.1 postoji više različitih čvorova za proračune čiji računarski kapaciteti mogu biti različiti, pa se procena vremena izvršavanja nekog proračunskog zadatka uzima kao srednje vreme izvršavanja na svim čvorovima za proračune. Suština algoritma Najranijeg starta svih zadataka neposrednih sledbenika se zasniva na proceni kada će se neki zadatak stvarno završiti i omogućiti izvršavanje svojih neposrednih sledbenika. S obzirom da se ne zna tačno koji će čvor za proračune izvršavati neki proračunski zadatak (određuje se na nivou mehanizma za proračune), a uzimajući u obzir i da se algoritam Kritična putanja zadataka pokazao nešto uspešnijim, izbor je pao na upotrebu Kritične putanja zadataka kada je u pitanju centralizovan deo raspodele.

U skladu sa izmenama prilikom raspodele proračunskih zadataka, ovi zadaci i potrebni ulazni podaci za njihovo izvršenje pristižu na posrednički čvor. Kada neki od slobodnih čvora za proračune usvaja zadatak sa posredničkog čvora, on preuzima i njegove podatke. Nakon završetka zadatka, rezultati se zapisuju na posrednički čvor, koji ih dalje prosleđuje na odgovarajući čvor i obaveštava sistem za upravljanje tokovima aktivnosti da je zadatak završen.

Očigledno je da ne postoji direktna komunikacija između čvorova za proračune i preostalih komponenti u sistemu (ostalih čvorova računarskog *Grid*-a i sistema za upravljanje tokovima aktivnosti), nego se ona odvija isključivo preko posredničkog čvora. Ovakva situacija je posledica želje da funkcionisanje ostatka sistema ne bude uslovljeno skupom čvorova za proračune. Ukoliko je potrebno dodatno pojačati računarsku moć unutar mehanizma za proračune i na taj način obezbediti bolje performanse sistema, nije potrebna izmena u radu preostalih čvorova računarskog *Grid*-a, već se jednostavno doda jedan ili više novih čvorova za proračune. Međutim, ovakav pristup, koji omogućava veću skalabilnost sistema, ima i svojih nedostataka. Glavni nedostatak je upravo to što se kompletna komunikacija vezana za proračunske zadatke odvija preko posredničkog čvora. Stoga, protok podataka kroz ovu računarsku komponentu može predstavljati usko grlo sistema ukoliko dođe do značajnog uvećanja čvorova za proračune.

Alternativa ovom pristupu je da se menja način rada sistema za upravljanje tokovima aktivnosti i svih čvorova u okviru infrastrukture računarskog *Grid*-a. Tada bi sistem za upravljanje tokovima aktivnosti, u toku procesa raspodele, morao da ima uvid u svaki čvor za proračune. Odnosno, ne bi bilo dovoljno da se utvrdi samo redosled izvršavanja proračunskih zadataka već bi bilo potrebno da se definiše na kom čvoru za proračune će se koji zadatak izvršavati. Očigledno je da ukoliko, bi došlo do izmene unutar skupa čvorova za proračune, to bi se odrazilo i na izmenu algoritma raspodele. Pored toga, kao problem ostaje otvoreno pitanje razmene međupodataka. odnosno kada se izvrše neposredni prethodnici proračunskih zadataka pitanje je na koji čvor za proračune treba da se prenesu međurezultati. Jedna od opcija je da sistem za upravljanje tokovima aktivnosti obezbedi njihov prenos na tako što će ih poslati na odgovarajući čvor za proračune nakon što odredi tačno na kom čvoru će se zadatak izvršiti. U tom slučaju, sistem za upravljanje tokova aktivnosti može da predstavlja novo usko grlo celog sistema što se tiče prenosa podataka.

Druga opcija je da svaki od preostalih čvorova računarskog *Grid*-a bude svestan postojanja svakog čvora za proračune i da mu sistem za upravljanje tokovima aktivnosti naknadno javi na koji od tih čvorova treba da se prenesu međurezultati. Nedostatak ovakvog pristupa je gubitak vremena, zato što međurezultati nisu pravovremeno preneti. Iz navedenih razloga uveden je posrednički čvor. Jasno je da se na taj način odustalo od potpunog *peer-to-peer* modela komunikacije, ali se izbeglo usložnjavanje funkcionisanja komponenti i omogućila se jednostavna podrška skalabilnosti. Međutim, s obzirom da

peer-to-peer model komunikacije ostaje na snazi u preostalom delu računarskog *Grid*-a za očekivati je da će ovakav pristup dati dobre rezultate. Konačno, ako se posmatra računarski *Grid* čija infrastruktura poseduje samo jedan čvor za proračune, očigledno je da opisano rešenje ne može biti lošije.

8.1 DISTRIBUIRANI ALGORITAM RASPODELE

Glavni cilj distribuiranog računarstva je postizanje paralelizma kroz preraspodelu opterećenja kako bi se dobila bolja iskorišćenost računarskih resursa i kraće vreme izvršenja. Ukoliko je odgovornost za donošenje odluka vezanih za raspodelu zadataka poverena većem broju distribuiranih komponenti u pitanju je distribuirani koncept raspodele zadataka [136].

Ukoliko je usvojen distribuirani koncept raspodele zadataka veoma je bitno da li distribuirane komponente koje su zadužene za raspodelu zadataka rade nezavisno jedna od druge (samostalno) ili su usaglašene u toku rada (kooperativan rad) [44]. Ukoliko ove komponente deluju samostalno, svaka odluka se donosi isključivo u korist njihovih individualnih ciljeva, nezavisno od efekta koje će proizvesti na ostatak sistema. U slučaju kooperativnog rada [137], svaka komponenta je zadužena da sprovede sopstveni deo posla vezan za raspodelu zadataka, ali tako da on bude u pravcu zajedničkog cilja. Kako bi se postigao postavljeni globalni cilj, lokalna strategija svake komponente za raspodelu zadataka zasnovana je na donošenju odluka u skladu sa ostalim komponentama, umesto da se zasniva na donošenju odluka koje će uticati samo na performanse izvršenja pojedinih zadataka. To znači da sve komponente za raspodelu zadataka izvršavaju svoj posao kako bi se zadovoljio odgovarajući kriterijum na nivou čitavog sistema.

Implementacija distribuirane raspodele zadataka je očigledno znatno komplikovanija od centralizovane raspodele. Kod nje kompletan skup informacija se ne nalazi na jednom mestu (što je slučaj kada je u pitanju centralizovana raspodela zadataka), tako da je potrebno voditi računar da svaka distribuirana komponenta zadužena za raspodelu zadataka obavlja svoj posao u smeru zadovoljenja globalnog cilja, a da se pri tom izbegnu konflikti kako neki od zadataka ne bi preuzelo više pomenutih komponenti. Sa druge strane, ovakav pristup omogućava željeni stepen skalabilnosti.

Računarski kapaciteti čvorova računarskog *Grid*-a koji učestvuju u distribuiranoj raspodeli opterećenja mogu biti heterogeni. Ako se ovakva heterogenost ne bi uzela u obzir, neki čvorovi bi završili posao pre ostalih i određeni deo vremena bi provodili neuposleni. Da bi se izbegao prazan hod ovih računarskih resursa i obezbedila njihova efikasna upotreba, potrebno je opterećenje (količinu posla koju je neophodno odraditi) rasporediti srazmerno njihovim računarskim mogućnostima. Problem koji se obrađuje u ovom radu jeste situaciju kada ukupna količina posla koju je potrebno odraditi nije poznata unapred, pa distribuirani algoritam mora da podrži dinamičku raspodelu.

Posebno treba istaći povezanost čvorova računarskog *Grid*-a koji učestvuju u distribuiranoj raspodeli opterećenja kao bitnu informaciju za razvoj distribuiranog algoritma. Naime, pošto u toku primene algoritma zadaci (opterećenja) treba da migriraju sa čvora na čvor, njihov transfer je moguć samo ukoliko postoji komunikaciona veza između tih čvorova. Ovakva organizacija resursa računarskog *Grid*-a se predstavlja neusmerenim cikličnim grafom čiji čvorovi reprezentuju računarske resurse na koje se opterećenje raspoređuje. Grane grafa postoje između onih čvorova koji predstavljaju računarske resurse između kojih je moguća dvosmerna razmena poruka [138].

Skup svih čvorova računarskog *Grid*-a koji učestvuju u distribuiranoj raspodeli je označen sa NC (u ovom radu to su svi čvorovi za proračun u okviru infrastrukture računarskog *Grid*-a). Njegova

kardinalnost je označena sa N_c. Trenutno opterećenje čvora za proračune n_i je označeno sa nw_{n_i}, a njegov računarski kapacitet sa np_{n_i}. U skladu sa tim, trenutna raspodela opterećenja predstavljena je vektorom:

$$NW = [nw_{n_1}, nw_{n_2}, \ldots, nw_{n_{N_c}}]. \tag{8.1}$$

Distribuirani proces raspodele ima zadatak da rasporedi opterećenje tako da ono na svakom čvoru konvergira ka željenoj, uravnoteženoj vrednosti. Ukoliko su u pitanju čvorovi koji imaju heterogene računarske kapacitete (npr. računari različitih procesorskih snaga) željeno opterećenje čvora n_i računa se na sledeći način:

$$\overline{nw_{n_i}} = \frac{\sum_{n_i \in NC} nw_{n_i}}{\sum_{n_i \in NC} np_{n_i}} np_{n_i}. \tag{8.2}$$

Ako je sistem sačinjen od čvorova homogenih računarskih kapaciteta, primenom distribuiranog algoritma raspodele teži se uniformnoj raspodeli opterećenja, odnosno teži se da opterećenje svakog čvora ima istu vrednost:

$$nw_{n_i} \to \overline{nw}, \quad \forall n_i \in NC. \tag{8.3}$$

S obzirom da su čvorovi računarskih kapaciteta homogenih, očekivana vrednost opterećenja čvorova $\overline{nw}$ je definisan kao:

$$\overline{nw} = \frac{\sum_{n_i \in NC} nw_{n_i}}{N_c}. \tag{8.4}$$

Željena, uniformna raspodela opterećenja svih čvorova tada se prikazuje vektorom $\overline{NW}$ (čija je dužina N_c):

$$\overline{NW} = [\overline{nw}, \overline{nw}, \ldots, \overline{nw}]. \tag{8.5}$$

U cilju pojednostavljenja objašnjenja, u nastavku je predstavljen distribuirani algoritam raspodele za čvorove homogenih računarskih kapaciteta. Nadogradnja algoritma za situacije u kojima čvorovi za proračune imaju različite računarske kapacitete je veoma jednostavna.

Odluke vezane za razmenu opterećenja mogu podržati različite inteligentne logike [139]. U ovom radu je predstavljen dinamički distribuirani algoritam raspodele koji se zasniva na fizičkom fenomenu difuzije [139, 140]. Difuzija predstavlja spontano kretanja materije ili energije iz regiona više koncentracije u region niže koncentracije. Tokom ovog procesa materija ili energija koja difunduje se ravnomerno raspoređuje u raspoloživom prostoru. Ideja je da se na sličan način dođe do kretanja opterećenja kako bi se isto ravnomerno rasporedilo po čvorovima računarskog *Grid*-a koji učestvuju u distribuiranom algoritmu raspodele.

Distribuirani algoritam raspodele opterećenja zasnovan na difuziji funkcioniše iterativno. U toku iteracije h, na osnovu informacije o opterećenju iz prethodne iteracije $h-1$, donosi se odluka o distribuciji opterećenja [141]. U toku iteracije h opterećenje čvora n_i naznačeno je sa $nw_{n_i}^h$.

Primenom iterativnog algoritma difuzije, u svakoj iteraciji čvor za proračune n_i razmenjuje deo

opterećenja $\alpha \cdot \left| nw_{n_i}^h - nw_{n_j}^h \right|$ sa čvorom n_j. Očigledno da je ovakva razmena opterećenja moguća samo ukoliko postoji direktna komunikaciona veza između ta dva čvora. Parametar α određuje koji udeo razlike opterećenja će biti razmenjen između čvorova. On je određen u zavisnosti od razmatranog problema [141, 142, 143]. Izmena opterećenje čvora za proračune n_i je opisana sledećim jednačinama:

$$nw_{n_i}^h = nw_{n_i}^{h-1} + \sum_{n_j \in NC} \alpha \cdot [nw_{n_j}^{h-1} - nw_{n_i}^{h-1}], \tag{8.5}$$

$$nw_{n_i}^h = [1 - \alpha \cdot N_c]\, nw_{n_i}^{h-1} + \alpha \cdot \sum_{n_j \in NC} nw_{n_j}^{h-1}. \tag{8.6}$$

S obzirom da je jednačina (8.6) linearna, izmena opterećenja čvorova računarskog *Grid*-a između iteracija $h-1$ i h može da se predstavi u matričnom obliku:

$$NW^h = DM \cdot NW^{h-1}, \tag{8.7}$$

gde je sa *DM* naznačena difuziona kvadratna matrica reda N_c [139]. Njeni elementi su označeni sa dm_{ij} i imaju sledeće vrednosti:

$$dm_{ij} = \begin{cases} \alpha, & i \neq j \\ 1 - \alpha \cdot N_c, & i = j \end{cases}. \tag{8.8}$$

Prilikom procene uspešnosti distribuiranog algoritma raspodele razmatraju se sledeće tri karakteristike algoritma[139, 144]:

- Završetak algoritma je povezan sa mogućnošću da se bilo kakva početna raspodela opterećenja po čvorovima računarskog *Grid*-a dovede u uravnoteženo stanje. U literaturi [141] je predstavljena konvergencija predstavljenog iterativnog algoritma difuzije.
- Efikasnost algoritma je povezana sa vremenom njegovog završetka. Ovo svojstvo se odnosi na ukupno vreme koje je potrebno da bi se postigla uravnotežena raspodela opterećenja na čvorovima računarskog *Grid*-a. Ukoliko je u pitanju iterativni algoritam, za ocenu njegove efikasnosti je pogodniji koristiti potreban broj iteracija da se dostigne uravnoteženo stanje. Količina podataka koja se prenosi kroz mrežu između čvorova (da bi se postigla preraspodela opterećenja) ima značajan uticaj na efikasnost distribuiranog algoritma [145]. Ovi troškovi prenosa podataka naročito dolaze do izražaja ukoliko čvorovi ne podržavaju mogućnost istovremenog izvršavanja zadataka i razmene opterećenja. Prethodno je naglašeno da problem kojim se ovaj rad bavi podrazumeva razmenu značajne količine podataka kako bi se izvršio neki zadatak. S obzirom da razmena opterećenja praktično znači razmenu zadataka, to za posledicu ima i razmenu ulaznih podataka tih zadataka. Zbog toga, prenos podataka neophodnih za izvršenje zadatka može da ima značajan uticaj na performanse sistema.
- Kvalitet dobijene raspodele opterećenja predstavlja veoma bitnu osobinu nekog algoritma. Često, nakon završetka algoritma opterećenje nije u potpunosti ravnomerno distribuirano. Odnosno, da postoje razlike u opterećenju između dva čvora računarskog *Grid*-a koji su direktno povezani komunikacionim vezama. Kvalitet dobijene raspodele opterećenja se kvalitativno može odrediti normom vektora $\overline{NW} - NW$. Na primer, može se koristi Euklidova maksimalna norma:

$$\max_{n_i \in NC} (\overline{nw} - nw_{n_i}), \tag{8.9}$$

ili kvadratnu normu (Euklidska razdaljina):

$$\sqrt{\sum_{n_i \in NC} [\overline{nw} - nw_{n_i}]^2} \ . \tag{8.10}$$

8.2 PRILAGOĐENJE DISTRIBUIRANOG ALGORITMA RASPODELE

U ovom delu rada objašnjena je primena distribuiranog algoritma raspodele u okviru mehanizma za proračune prikazanog na slici 8.1. Mehanizam za proračune se sastoji od skupa čvorova za proračune i posredničkog čvora. Da bi se predstavljen distribuirani algoritam mogao implementirati, uz odgovarajuća ograničenja koja su uslovljena osobinama zadataka (prioritet zadataka i uticaj njihovog izvršenja na ostale zadatke iz usmerenog acikličnog grafa) i da bi se njegovom primenom postigla što veća iskorišćenost računarskih resursa, a time i veća brzina izvršenja proračunskih zadataka, očigledno je da su potrebne odgovarajuće modifikacije.

Prvenstveno treba naglasiti da sami čvorovi za proračune implementiraju distribuirani algoritam raspodele zadataka. Oni su međusobno u potpunosti povezani komunikacionom infrastrukturom, pri čemu je moguća razmena poruka između bilo koja dva čvora za proračune. Pošto je razmena informacija neophodna za implementaciju kooperativnog rada prilikom distribucije zadataka, ukupan posao koji čvorovi za proračune obavljaju se dodatno komplikuje. To znači da pored izvršavanja zadataka svaki čvor za proračune treba da obezbedi dopunske funkcionalnost koje obuhvataju razmenu poruka sa preostalim čvorovima za proračune i posredničkim čvorom, kao i primenu određenih internih akcija koje predstavljaju deo sveukupnog distribuiranog algoritma raspodele proračunskih zadataka. Osnovna karakteristika ovog algoritma je da koristi pretragu da bi pronašao partnerske čvorove za proračune koji su manje opterećeni i na koje proračunski zadaci mogu da migriraju. Pored toga, donošenja odluka se realizuje istovremeno dok se zadaci izvršavaju i na taj način se štedi dragoceno vreme.

Druga bitna karakteristika na koju treba obratiti pažnju je količina opterećenja koja može da se razmeni između čvorova za proračune. Pod opterećenjem nekog čvora za proračune smatra se ukupna količina aktivnosti koju definišu proračunski zadaci koje je taj čvor preuzeo. Kao što je prethodno objašnjeno, u distribuiranom algoritmu raspodele se pretpostavlja da je moguća razmena bilo koje količine opterećenja između čvorova, odnosno, smatra se da je deo opterećenja predviđen za razmenu kontinualna vrednost (realan broj). U predstavljenom DMS okruženju situacija je nešto drugačija, s obzirom da je zadatak atomska, nedeljiva celina. Skup zadataka je dodeljen čvoru za proračune, a između čvorova je moguće distribuirati isključivo celokupne zadatke. Stoga deo opterećenja za razmenu može dobiti samo vrednosti koje predstavljaju zbir opterećenja nekih postojećih proračunskih zadataka. Kako parametar α određuje udeo razlike opterećenja za razmenu, on uslovljava brzinu konvergencije i kvalitet dobijenog rešenja. Da bi se postigla najveća moguća razmena opterećenja između čvorova za proračune, u ovom istraživanju parametar α je podešen na $1/2$ ($\alpha = 1/2$). U nastavku je prikazana matematička formulacija distribuiranog algoritma raspodele zadataka koji implementiraju čvorovi za proračune.

Za svaki čvor za proračune definiše se ukupno opterećenje kao skup njemu dodeljenih proračunskih zadataka, bez obzira da li je izvršenje nekih zadataka počelo ili ne. Sa TC_{n_i} je naznačen skup proračunskih zadatka koji su dodeljeni čvoru za proračune n_i, a da njihovo izvršenje još uvek nije počelo (podskup ukupnog opterećenja čvora za proračune n_i). Sa $\Delta nw_{n_i n_j}$ je naznačen maksimalni iznos opterećenja koje čvor za proračune n_i može razmeniti sa čvorom za proračune n_j. Kako je prethodno objašnjeno, maksimalni iznos opterećenja se računa kao proizvod razlike ukupnih opterećenja čvorova za

proračune i predefinisanog parametara α:

$$\Delta nw_{n_i n_j} = \alpha \cdot [nw_{n_i} - nw_{n_j}]. \tag{8.11}$$

Očigledno je da ukoliko $\Delta nw_{n_i n_j}$ ima negativnu vrednost, čvor za proračune n_j je opterećeniji od čvora za proračune n_i i ne treba da dođe do prenosa opterećenja sa n_i na n_j. U suprotnom treba odrediti koja količina opterećenja se može razmeniti. S obzirom da je moguća razmena samo celokupnih zadataka kao nedeljivih jedinica, potrebno je odrediti koji proračunski zadaci treba da migriraju sa čvora za proračune n_i na čvor za proračune n_j. Naravno, mogu migrirati samo oni proračunski zadaci čije izvršenje još uvek nije počelo na n_i.

Sa $\Delta TC_{n_i n_j}$ je naznačen podskup zadataka usvojenih od strane čvora za proračune n_i (podskup skupa TC_{n_i}) za koje ukupna suma opterećenja ima maksimalnu vrednost:

$$\max_{\Delta TC_{n_i n_j} \in Pow(TC_{n_i})} \left\{ \sum_{T_k \in \Delta TC_{n_i n_j}} w_k \right\}, \tag{8.12}$$

uz ograničenje da ta vrednost ne prelazi $\Delta nw_{n_i n_j}$:

$$\sum_{T_k \in \Delta TC_{n_i n_j}} w_k \le \Delta nw_{n_i n_j}. \tag{8.13}$$

$Pow(TC_{n_i})$ predstavlja partitivni skup skupa TC_{n_i} (skup svih podskupova TC_{n_i}, uključujući i prazan skup). Na ovaj način je određen skup zadataka koji može da preuzme čvor za proračune n_j.

Suma opterećenja proračunskih zadataka koji treba da migriraju ne mora biti jednaka dozvoljenoj količini opterećenja za prenos, pa je nakon transfera zadataka neophodno iskoristi upravo vrednost te sume da bi se odredilo novonastalo stanje čvorova za proračune. Ako $\Delta^h_{n_i n_j}$ predstavlja deo opterećenja koji će stvarno biti razmenjen između čvorova za proračune n_i i n_j u toku iterativnog koraka h (suma proračunskih zadataka koji migriraju), onda se evolucija opterećenja opisuje sledećim pravilima:

1) Odrediti količinu opterećenja koju je dozvoljeno preneti sa čvora za proračune n_i na čvor za proračune n_j i skup zadataka čije izvršenje nije počelo, a da pri tome nose ukupnu maksimalnu količinu opterećenja, ne veću od dozvoljene:

$$\Delta nw^{h-1}_{n_i n_j} = \alpha \cdot [nw^{h-1}_{n_i} - nw^{h-1}_{n_j}], \tag{8.14}$$

$$\Delta TC^{h-1}_{n_i n_j} \subset TC^{h-1}_{n_i}, \sum_{T_k \in \Delta TC^{h-1}_{n_i n_j}} w_k \le \Delta nw^{h-1}_{n_i n_j}. \tag{8.15}$$

2) Izračunati stvarnu količinu opterećenja koja će biti razmenjena između čvorava za proračune:

$$\Delta^{h-1}_{n_i n_j} = \sum_{T_k \in \Delta TC^{h-1}_{n_i n_j}} w_k. \tag{8.16}$$

3) Odrediti novo stanje (ukupno opterećenje) čvorova za proračune koji su učestvovali u razmeni zadataka:

$$nw^h_{n_l} = nw^{h-1}_{n_i} - \Delta^{h-1}_{n_i n_j}, \tag{8.18}$$

$$nw_{n_j}^{h} = nw_{n_j}^{h-1} + \Delta_{n_i n_j}^{h-1} . \quad (8.19)$$

4) Utvrditi novi skup zadataka čiji izvršenje još uvek nije počelo na oba čvora za proračune:

$$TC_{n_i}^{h} = TC_{n_i}^{h-1} \setminus \Delta TC_{n_i n_j}^{h-1} , \quad (8.20)$$

$$TC_{n_j}^{h} = TC_{n_j}^{h-1} \bigcup \Delta TC_{n_i n_j}^{h-1} . \quad (8.22)$$

Algoritam je završen kada je opterećenje raspoređeno tako da dalja razmena nije neophodna. To je ostvareno kada $\Delta TC_{n_i n_j}^{h}$ postane prazan skup za svaku moguću kombinaciju (i, j), što je ekvivalentno sa $\Delta_{n_i n_j}^{h} = 0, \forall (i, j)$. Tada je svaki čvor za proračune dobio svoju dozu opterećenja koju treba da iznese kako bi se ubrzalo izvršavanje zadataka i na taj način podigle performanse čitavog sistema. U listingu 8.1 prikazan je pseudo kod kojim se opisuje upravo predstavljeni distribuirani algoritam raspodele.

```
while( finished ≠ true )
    // kreće se od pretpostavke da raspodela nije neophodna
     finished ← true
    for all ( n_i ∈ CN )
      for all ( n_j ∈ CN )
            compute Δnw_{n_i n_j}
            determine ΔTC_{n_i n_j}
            if ( ΔTC_{n_i n_j} not empty)
                compute Δ_{n_i n_j}
                nw_{n_i} ← nw_{n_i} − Δ_{n_i n_j}
                nw_{n_j} ← nw_{n_j} + Δ_{n_i n_j}
                TC_{n_i} ← TC_{n_i} / ΔTC_{n_i n_j}
                TC_{n_j} ← TC_{n_j} ⋃ ΔTC_{n_i n_j}
                // algoritam nije završen
                 finished ← false
            end if
      end for all
    end for all
end while
```

Listing 8.1 – Distribuirani algoritam raspodele

Implementacija ovakvog kooperativnog distribuiranog algoritma raspodele se najčešće zasniva na činjenici da komponente za raspodelu zadataka periodično propituju jedna drugu, kako bi bile prikupljene informacije o trenutnim opterećenjima. Međutim, da se ne bi gubilo vreme na nepotrebnu razmenu poruka, mnogo je efikasnije realizovati algoritam koji bi podržao iniciranje komunikacija između čvorova za proračune samo kada je to stvarno neophodno, npr. nakon odgovarajućih događaja.

U nastavku su definisana pravila implementacije opisanog distribuiranog algoritma raspodele. Svi učesnici u distribuiranoj raspodeli zadataka (čvorovi za proračune i posrednički čvor) treba da se pridržavaju pravila, na osnovu kojih bi se uzeo u obzir prioritet proračunskih zadataka i njihov uticaj na ostale zadatke iz usmerenog acikličnog grafa, tako da se smanji nepotrebna komunikacija i eliminiše nepotrebna razmena podataka:

- Posrednički čvor prima zahtev da se izvrši neki proračunski zadatak. Nakon što pristignu ulazni podaci, on na osnovu metodologije redom u krug (onaj koji je sledeći na redu) obaveštava odgovarajući čvor za proračune da mu je dodeljen zadatak.
- Čvor za proračune izvršava samo jedan zadatak u određenom vremenskom periodu. S obzirom da se razmenjuju samo zadaci čije izvršenje nije počelo, na ovaj način mu ostaju više mogućnosti za razmenu zadataka, odnosno veći broj zadataka može da se iskombinuje kako bi se prilikom razmene postigla odgovarajuća količina opterećenja (ne veća od maksimalne dozvoljene).
- Za svaki od proračunskih zadataka uzimaju se u obzir dva atributa u toku distriburaine respodele: prioritet i vrednost kritične putanje izvršenja. Prilikom izbora koji od dodeljenih zadataka će izvršiti čvor za proračune, izvršava se onaj koji je najvišeg prioriteta. Ako više zadataka ima isti prioritet, bira se onaj zadatak koji ima najveću vrednost kritične putanje izvršenja. Na taj način distribuirani algoritam raspodele uzima u obzir prioritet proračunskih zadataka, kao i njihov uticaj na preostale zadatke iz usmerenog acikličnog grafa.
- Ulazni podaci nekog zadatka se povlače na čvor za proračun tek kada počine izvršavanje zadatka. Suprotno ponašanje bi imalo za posledicu da se nakon migracije zadataka migriraju i njihovi ulazni podaci (čija veličina nije zanemarljiva). Ovakvi transferi podatka značajno bi uticali na pogoršanje performansi mehanizma za proračune.
- Ukoliko bi došlo do prelaza zadataka sa jednog čvora za proračune na drugi, s obzirom da je drugi čvor manje opterećen, smatra se da je veća verovatnoća da će pre početi sa izvršenjem nekog zadatka. Iz tog razloga, prilikom izbora zadataka koji treba da se prenesu, pored osnovnih zahteva definisanih jednačinama (8.14) i (8.15), na isti način kao i prilikom izbora zadataka za izvršenje, u obzir se uzimaju prioritet i vrednost kritične putanje izvršenja.
- Da bi se izbegla periodična razmena informacija o količini opterećenja, čvor za proračune pokreće procedure za prenos zadataka sa svim ostalim čvorovima prilikom pojave sledećih događaja:
    - kada mu je dodeljen zadatak od strane posredničkog čvora;
    - kada primi zadatak od strane nekog drugog čvora za proračune;
    - kada je u toku rada uključen novi čvor za proračune.

Na početku rada sistema svi čvorovi za proračun su neopterećeni, odnosno postoji početno ustaljeno stanje. Moguće je da nakon pojave bilo kog od prethodno navedenih događaja dođe do narušavanja ustaljenog stanja, pa ga je potrebno ponovo dostići. Nakon što je čvor za proračune razmenio zadatke (odgovarajuću količinu opterećenja) sa svim ostalim učesnicima, sa njegovog aspekta postignuto je ustaljeno stanje. Pomenuti čvor za proračune ne nastavlja dalju razmenu opterećenja ukoliko ona nije inicirana od strane nekog drugog čvora za proračune.

8.3 OTPORNOST NA GREŠKE

U šestom poglavlju objašnjene su osnovne tehnike koje je potrebno implementirati radi uvećanja pouzdanosti rada DMS-a. Pored inicijalne validacije tokova aktivnosti, uvedena je statička replikacija kao

adekvatna tehnika za rukovanje greškama ukoliko dođe do otkaza nekog resursa računarskog *Grid*-a. Predloženo je da svaki čvor računarskog *Grid*-a poseduje svoju aktivnu repliku koja bi mogla da ga zameni u slučaju da dođe do greške. Podsećanja radi, ukoliko neki zadatak, nakon što je poslat od strane komponente za neposrednu raspodelu zadataka, nije završen u predviđenom vremenskom roku, smatra se neuspešnim i potencijalno šalje na ponovno izvršenje.

Pomenuta statička organizacija redundantnih računarskih resursa (replika) bila je uslovljena ograničenjem da se neki zadatak izvršava isključivo na jednom, predodređenom čvoru računarskog *Grid*-a. U situaciji kada postoji više čvorova za proračun kao logično unapređenje prethodnog rešenja nameće se ukidanje statičkih redundantnih računarskih resursa u okviru mehanizma za proračune i implementacija podrške za aktivnu replikaciju. Štaviše, ukoliko se razmotri prethodno opisano rukovanje greškama kada se neuspešno izvršen zadatak šalje na ponovno izvršenje, ukoliko nije prekoračen ukupan broj pokušaja da se neki zadatak izvrši, potpuno je jasno da je podrška aktivnoj replikaciji već realizovana. Dakle, ukoliko neki od čvorova za proračune postane nedostupan i zadatak koji je preuzeo se ne izvrši u predviđenom roku (ili se ne izvrši uopšte), obaveštenje o uspešno izvršenom zadatku neće stići do komponente za neposrednu raspodelu zadataka. Skreće se pažnja da se ulazni podaci potrebni za izvršenje prethodno neuspešnog proračunskog zadatka i dalje nalaze na posredničkom čvoru. U toku daljeg rada, komponenta za neposrednu raspodelu zadataka ponovo šalje opis zadatka na posrednički čvor. S obzirom da su ulazni podaci tamo već stacionirani bilo koji od dostupnih čvorova za proračune može preuzeti pomenuti zadatak i pokušati da ga izvrši.

Opisano rešenje se može dodatno unaprediti tako što bi posrednički čvor takođe obrađivao informaciju o neuspešno izvršenom proračunskom zadatku. On bi pored ulaznih podatka čuvao i opis zadatka. Ukoliko čvor za proračune, koji je preuzeo zadatak, ne uspe da ga izvrši, posrednički čvor bi proglasio da zadatak nije dodeljen i neki drugi čvor za proračune bi mogao da ga preuzme kako bi pokušao da ga izvrši.

8.4 EKSPERIMENTALNI REZULTATI

Realizacija eksperimenata na osnovu kojih se verifikovala efikasnost kombinovane raspodele zadataka odvijala se u dva pravca. U prvoj seriji eksperimenata izvršena su merenja u identičnim uslovima, kao prilikom upotrebe centralizovane raspodele zadataka. Dobijeni rezultati su upoređeni sa rezultatima koje su dobijeni primenom centralizovanog algoritma. U okviru druge serije eksperimenata uzeti su u obzir rezultati koji se dobijaju kada mehanizam za proračune sadrži različit broj čvorova za proračune.

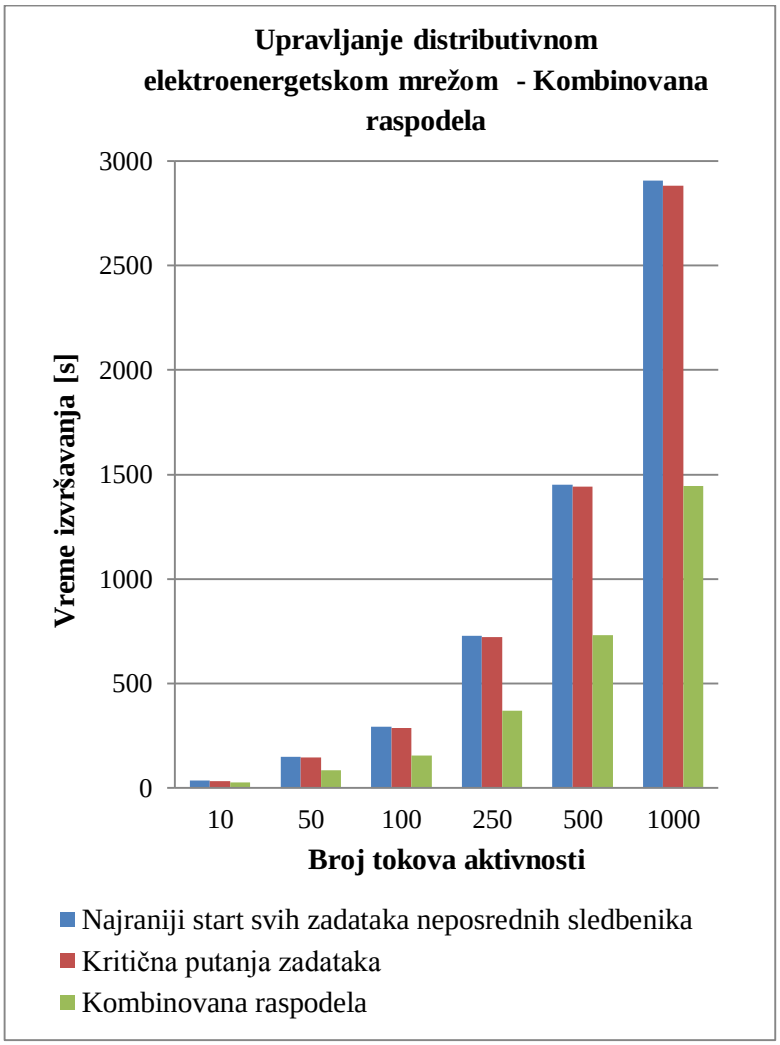
Čvorovi za proračune ne učestvuju u tokovima aktivnosti koji definišu inicijalni unos podataka u DMS, pa nema smisla ponavljati ovaj skup eksperimenata uz upotrebu modifikovanog mehanizma za proračune. Iz tog razloga, svi eksperimenti su posvećeni isključivo simulaciji rada DMS-a u toku upravljanja distributivnom mrežom. Još jednom se napominje da sistem za upravljanje tokovima aktivnosti, nezavisno od mehanizma za proračune, koristi algoritam zasnovan na kritičnoj putanji zadataka. Sa druge strane, čvorovi za proračun upotrebljavaju distribuirani algoritam za preraspodelu opterećenja. Upotrebljen je skup testova identičan testovima za ispitivanja korisnosti centralizovanih algoritama raspodele.

U prvoj seriji eksperimenata, računarski *Grid*, pored čvorova za obradu statičkih, dinamičkih i grafičkih podatka (SSP, SDP, SGP), obuhvata i mehanizam za proračune koga čine tri čvora za proračune i posrednički čvor. U toku testiranja upotrebljen je različit broj tokova aktivnosti koji dinamički pristižu

na obradu. U tabeli 8.1 i na slici 8.2 prikazane su vrednosti *makespan*-ova koje su dobijene primenom kombinovanog i centralizovanog algoritma raspodele za iste kolekcije tokova aktivnosti.

Tabla 8.1 – Vreme izvršavanja tokova aktivnosti – kombinovani algoritam raspodele

Broj tokova aktivnosti	Najraniji start svih zadataka neposrednih sledbenika [s]	Kritična putanja zadataka [s]	Kombinovani algoritam raspodele zadataka [s]
10	36.89	34.05	27.87
50	148.01	146.33	84.75
100	293.37	288.58	155.26
250	729.05	723.41	371.45
500	1452.24	1442.90	732.15
1000	2906.62	2881.85	1445.92



Slika 8.2 – Upravljanje distributivnom mrežom – kombinovani algoritam: vreme izvršavanja tokova aktivnosti

Na osnovu prikazanih rezultata prve serije eksperimenata može se zaključiti sledeće:

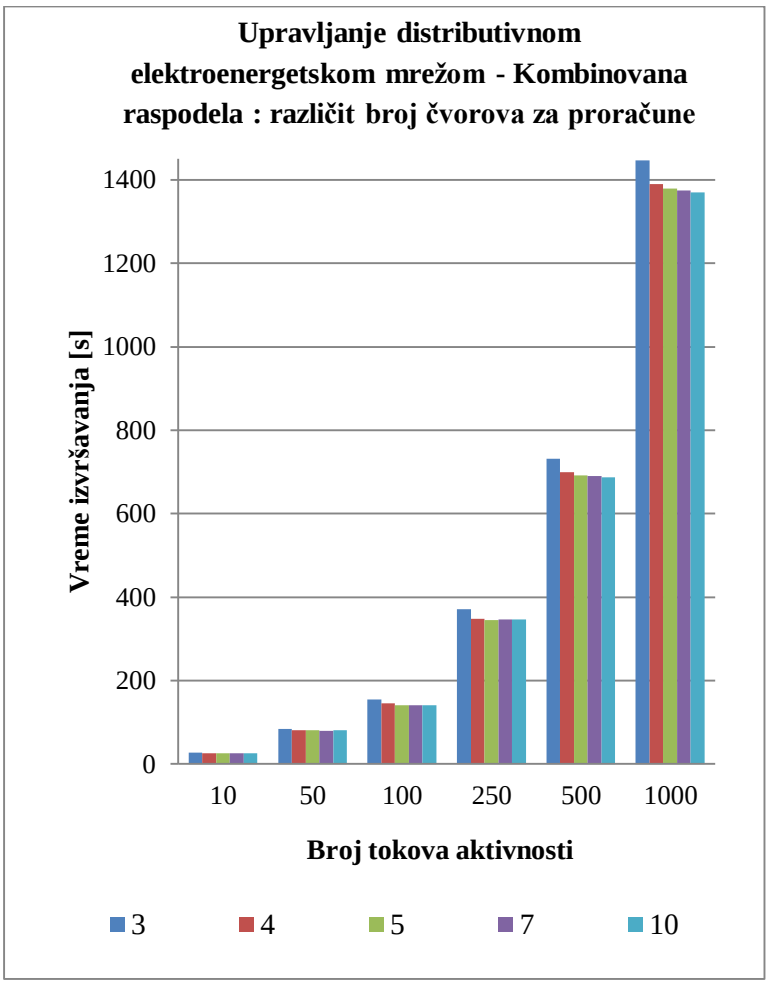
- Kombinovani algoritam raspodele značajno prevazilazi dosadašnja (centralizovana) rešenja.
- Razlika *makespan*-a kolekcije tokova aktivnosti povećava se sa povećanjem broja tokova aktivnosti.
- Relativno mala ulaganja u resurse računskog *Grid*-a i upotreba kombinovanog algoritma raspodele rešava problem uskog grla sistema koje predstavlja čvor za proračune u DMS-u velikih dimenzija.
- Uvođenjem dinamičke raspodele zadataka, uz upotrebu kombinovanog algoritma, ostvarena je značajno bolja eksploatacija resursa računarskog *Grid*-a, što rezultuje boljim performansama celokupnog sistema.

U okviru druge serija eksperimenata analizirana je zavisnost između performansi sistema i broja čvorova u okviru mehanizma za proračune. Razmatran je isti primer, identične kolekcije tokova aktivnosti, kao u prvoj seriji eksperimenata. Poređenja rezultata, *makespan*-ova upotrebljenih kolekcija tokova aktivnosti, prikazano je u tabeli 8.2.

Tabla 8.2 – Vreme izvršavanja tokova aktivnosti – Kombinovani algoritam: različit broj čvorova za proračun

	Vreme izvršavanja tokova aktivnosti [s]				
Broj tokova aktivnosti	$N_C = 3$	$N_C = 4$	$N_C = 5$	$N_C = 7$	$N_C = 10$
10	27.87	26.56	26.17	26.13	25.84
50	84.75	81.38	81.07	80.25	80.85
100	155.26	145.18	141.65	141.18	141.10
250	371.45	348.65	345.05	346.14	346.19
500	732.15	699.94	692.40	689.89	687.37
1000	1445.92	1389.95	1379.16	1374.87	1369.84

Kvalitet rezultata (brzina izvršavanja tokova aktivnosti) nastalih upotrebom mehanizma za proračune koji poseduje različit broj čvorova i kombinovanog algoritma raspodele zadataka prikazan je na i slici 8.2.



Slika 8.3 – Upravljanje distributivnom mrežom – kombinovani algoritam: različit broj čvorova za proračune

Na osnovu prikazanih rezultata iz druge serije eksperimenata može se zaključiti sledeće:

- Povećanjem broja čvorova u okviru mehanizma za proračune poboljšavaju se performanse sistema, ali, s obzirom da su ovi čvorovi zaduženi samo za proračunske zadatke, progresija performansi sistema nije srazmerna broju čvorova za proračune.
- Nakon uključivanja određenog broja čvorova u okviru mehanizma za proračune dolazi do zasićenja. Primećuje se da uključivanjem četvrtog čvora za proračune dolazi do primetnijeg smanjenja vremena izvršenja kolekcije tokova aktivnosti (kompletan skup tokova aktivnosti se izvrši brže) u odnosu na mehanizam za proračune sa tri čvora. Nakon uključivanja petog čvora, performanse sistema su za nijansu bolje, dok upotrebom sedam ili deset čvorova za proračune ne postiže se značajniji napredak.
- Investicije u dodatne resurse računarskog *Grid*-a, tačnije mehanizam za proračune, su opravdane ukoliko postoji značajno poboljšanje performansi sistema. S obzirom da nakon određenog broja čvorova za proračune nastupa zasićenje, evidentno je da preveliko ulaganje u dodatne čvorove koje koristi mehanizam za proračune nema smisla, jer tako unapređen mehanizam za proračune ne mora dati bolje rezultate. Na osnovu odnosa cene računarskih resursa i postignutih performansi sistema određuje se optimalan broj čvorova za proračune.

9. ZAKLJUČAK

Savremeni život je nezamisliv bez električne energije. Zahtevi za njenom upotrebom rastu iz dana u dan. Da bi se optimizovalo njeno iskorišćenje i postigla odgovarajuća ušteda prilikom proizvodnje, prenosa i potrošnje, neophodan je konstantan, sistematičan nadzor i kontrola kompletne elektroenergetske mreže. Kontrola je bolja ukoliko je dostupna veća količina informacija. Pošto se cene današnje komunikacione tehnologije mogu smatrati pristupačnim, razvijeni su *SmartGrid* sistemi koji prikupljaju velike količine informacija, obrađuju ih i na taj način omogućavaju efikasno upravljanje elektroenergetskom mrežom.

SmartGrid obuhvata kolekciju podsistema: SCADA, GIS, EMS, CMS, itd. DMS je takođe jedan od njih. Osnovna svrha ovog podsistema je da odredi radni režim distributivne mreže. Karakteriše ga preuzimanje značajnih količina podataka od preostalih podsistema da bi mogao korektno da funkcioniše. Savremeni DMS, pod uticajem svog računarskog okruženja, izvršava niz aktivnosti uz koje se nameće niz ograničenja kao što su njihov prioritet i međusobna zavisnosti. Izvršenje svake od pomenutih aktivnosti zahteva intenzivnu i efikasnu obradu velike količine distribuiranih podataka, pa je neophodno da DMS postane moćna distribuirana aplikacija koja će odgovoriti na ovakve zahteve.

Za efikasan rad DMS-a neophodna je odgovarajuća koordinacija izvršavanja tokova aktivnosti, uz poštovanje nametnutih ograničenja. Namera je da se izvršavanje tokova aktivnosti paralelizuje u što većoj meri. Na taj način se optimizuje iskorišćenje resursa računarskog *Grid*-a na kome je DMS postavljen, a ukupna kolekcija tokova aktivnosti se izvršava u najkraćem roku. U tu svrhu uvodi se sistem za upravljanje tokovima aktivnosti.

Predloženi sistem za upravljanje tokovima aktivnosti uzima u obzir kako dinamičku prirodu DMS-a, tako i variranje performansi resursa računarskom *Grid*-a. On se zasniva na ideji da se tokovi aktivnosti izdele na jednostavnije korake, zadatke, koji se dalje tretiraju kao nedeljive celine. S obzirom na nepredvidivost računarskog okruženja neophodno je da sistem za upravljanje tokovima aktivnosti podrži dinamičku raspodelu zadataka. Zbog toga on osluškuje pristizanje novih tokova aktivnosti, prati statuse čvorova računarskog *Grid*-a i primenjuje odgovarajući algoritam raspodele zadataka da bi podržao kompletan proces raspodele visoke propusne moći. Sistem za upravljanje tokovima aktivnosti kreiran u ovom istraživanju omogućava jednostavnu izmenu algoritma koji realizuje dinamičku raspodelu zadataka. Razvijeni su i ispitani različiti algoritmi raspodele zadataka koji uzimaju u obzir međusobnu zavisnost tokova aktivnosti prilikom izvršavanja, a pri tome obezbeđuju poštovanje njihovih prioriteta.

Kreirana su dva centralizovana algoritma raspodele zadataka koje koristi sistem za upravljanje tokova aktivnosti u cilju boljeg iskorišćenja računarskih resursa: Najraniji start svih zadataka neposrednih sledbenika i Kritična putanja zadataka. Algoritam Najranijeg starta svih zadataka neposrednih sledbenika koristi lokalni način donošenja odluka, dok se Kritična putanja zadataka oslanja na globalni način donošenja odluka. Eksperimentalne studije koje su predstavljene u radu pokazuju korisnost i upotrebljivosti ovih algoritama. Blagu prednost u performansama sistema pokazala je upotreba algoritma koji koristi kritičnu putanju zadataka. U okviru iste eksperimentalne studije, pored određivanja

makespan-a kolekcije tokova aktivnosti, vršena su i merenja ukupnog vremena potrošenog na rad algoritama. Vreme rada algoritama je zanemarljivo u odnosu na *makespan*, pa je jasno da ono nema uticaj na proces raspodele. Prednost centralizovane raspodele zadataka predstavlja jednostavna implementacija. Međutim, ovakav način rada pokazuje odsustvo skalabilnosti.

Dalja analiza rada DMS-a tokom upravljanja distributivnom mrežom pokazuje da je neophodno obezbediti dodatnu podršku posebno opterećenom čvoru za proračune. Predloženo rešenje opisuje proširenje infrastrukture računarskog *Grid*-a i uvođenje naprednog mehanizam za proračune koji obuhvata višestruke čvorove. Svaki od čvorova za proračune implementira dizajnirani distribuirani algoritam raspodele opterećenja. Na ovaj način je postignut koncept kombinovane raspodele: centralizovani algoritam raspodele koristi algoritam Kritične putanje zadataka, dok se mehanizam za proračune oslanja na distribuirani algoritam. Cilj predstavljene eksperimentalne studije je da prikaže efikasnost raspodele zasnovane na kombinovanom algoritmu. Ova studija se sastoji iz dve serije eksperimenata. Prva serija eksperimenata ima za cilj da uporedi rezultate kombinovanog algoritma raspodele sa centralizovanim algoritmima. Analiza performansi pokazuje da kombinovani algoritam raspodele značajno smanjuje ukupno vreme izvršenja i povećava performanse celokupnog sistema. Pokazano je da se uz relativno mala finansijska ulaganja može dobiti značajno unapređenje performansi. Druga serija eksperimenata se odnosi na ispitivanje uticaja broja čvorova za proračune na *makespan* kolekcije tokova aktivnosti. Smanjenje *makespan*-a nije srazmerno uvećanju broja čvorova za proračune. Nakon određenog broja čvorova nastaje zasićenje, pa je potrebno naći određenu ravnotežu između željenih performansi sistema i finansijskih ulaganja u računarske resurse. Treba naglasiti da distribuirani algoritam raspodele omogućava dodavanje novih čvorova za proračune bez uticaja na sistem za upravljanje tokovima aktivnosti, pa se time obezbeđuje odgovarajući stepen skalabilnosti.

S obzirom na važnost posla koji obavlja, neophodan je pouzdan rad DMS-a. Sistem treba da je realizovan na način da neometano nastavi funkcionisanje ukoliko dođe do greške. Predložena su dva pristupa kojim bi se osigurala otpornost na greške. Prvi, preventivni pristup predviđa validaciju tokova aktivnosti pre nego što se oni uključe u kolekciju tokova aktivnosti. Na taj način se izbegava obrada nevalidnih tokova aktivnosti od strane algoritma za raspodelu. Pošto je moguća pojava grešaka i usled neispravnosti resursa računarskog *Grid*-a uvodi se tehnika aktivne replikacije kao drugi pristup za rukovanje greškama. Dodatni napredak u ovom segmentu postiže se uvođenjem mehanizma za proračune koji sadrži višestruke čvorove. Naime, s obzirom da bilo koji čvor za proračune može preuzeti posao nekog drugog čvora nije neophodno da svaki od njih poseduje svoju repliku. Iz tog razloga je pogodna primena dinamičke replikacije unutar mehanizma za proračune. U okviru dinamičke replikacije zadatke nedostupnog čvora za proračune će preuzeti ostali čvorovi.

Dalja istraživanja bi se mogla zasnivati na razvoju algoritama raspodele zadataka koji bi u obzir uzimali i eventualna ograničenja vezana za vremenske rokove do kada pojedini tokovi aktivnosti treba da budu izvršeni. Takvi algoritmi bi predstavljali unapređenje opisanih algoritama na način da stvore mogućnost da se odgovarajući resursi računarskog *Grid*-a rezervišu unapred, a u sistem za upravljanje tokovima aktivnosti bi morale biti uvedene dodatne inicijalne validacije kojima bi se proveravalo da li uopšte postoji mogućnost da se neki tok aktivnosti završi u određenom roku, ukoliko takvo ograničenje postoji.

Pored toga, određena pažnja bi mogla biti posvećena unapređenju distribuiranog algoritma raspodele. Naime, postoji mogućnost da dođe do prekida komunikacionih veza između pojedinih čvorova računarskog *Grid*-a koji učestvuju u distribuciji opterećenja. U tom smislu, distribuirani algoritam

raspodele bi trebalo unaprediti kako bi adekvatno mogao da funkcioniše i u ovakvom okruženju, koje uvodi novu dimenziju dinamičnosti.

10. LITERTURA

[1] V.Strezoski: Osnovi elektroenergetike, Fakulteta tehničkih nauka, Novi Sad, 2014.

[2] [T.Gutowski, 2006] T.Gutowski, J.Dahmus, A.Thiriez: Electrical Energy Requirements for Manufacturing Processes, In Proceedings of International Conference on Life Cycle Engineering (CIRP), page(s) 5, Leuven, Belgium, 2006.

[3] F.P.Sionsansi: Smart Grid: Integrating Renewable, Distributed & Efficient Energy, Academic Press, 2011.

[4] H.Tram: Technical and operation considerations in using Smart Metering for outage management, Transmission and Distribution Conference and Exposition, T&D. IEEE/PES, Chicago, USA, pp. 1-3, 2008.

[5] E.Santacana, G.Rackliffe, L.Tang, X.Feng: Getting Smart; IEEE Power and Energy Magazine, Vol. 8, No. 2, pp. 41-48, 2010.

[6] A.Bose: Smart Transmission Grid Applications and Their Supporting Infrastructure, IEEE Transactions on Smart Grid, Vol. 1, No. 1, pp. 11-19, 2010.

[7] R.DeBlasio, T.Cherry: Standards for the Smart Grid, In Proceedings of International Conference IEEE Energy 2030, Atlanta, USA, pp. 1-7, 2008.

[8] E.Lakervi, E.Holmes: Electricity Distribution Network Design, Peter Peregrinus Ltd., London, United Kingdom, 1998.

[9] D.Popović: Power applications: A cherry on the top of the DMS cake, DA/DSM DistribuTECH Europe, Vienna, Austria, 2000.

[10] W.R.Cassel: Distribution Management Systems: Functions and Payback, IEEE Transactions on Power Systems, Vol. 8, No. 3, pp. 796-801, 1993.

[11] D.Čapko: Optimalna podela velikih modela podataka u okviru nadzorno upravljačkih elektroenergetskih sistema, doktorska disertacija, Fakultet tehničkih nauka, Novi Sad, 2012.

[12] M.Gavric, M.Martinov, S.Bojic, Dj.Djatkov, M.Pavlovic: Short- and long-term dynamic accuracies determination of satellite-based positioning devices using a specially designed testing facility, Computers and Electronics in Agriculture, Vol. 76, No. 2, pp. 297–305, 2011.

[13] IEC 61970 Energy management system application program interface (EMS-API) – Part 301: Common Information Model (CIM) Base, IEC, Edition 2.0, 2007.

[14] IEC/TR 61968-11 Application integration at electric utilities - System interfaces for distribution management - Part 11: Common information model (CIM) extensions for distribution, IEC, Edition 1.0, 2010.

[15] N.Nedić, A.Erdeljan, I.Lendak, D.Čapko: Pristup modelu podataka elektroenergetskog sistema putem Web servisa, 53. Konferencija ETRAN 2009, Vrnjačka Banja, Srbija, 2009.

[16] V.Strezoski, D.Popović, D.Bekut, N.Katić, G.Švenda, Z.Gorečan, J.Dujić: Osnovne energetske funkcije za analizu, upravljanje i planiranje pogona srednjenaponskih distributivnih mreža, IV skup Trendovi Razvoja: Nove tehnologije u elektrodistribuciji, Kopaonik, Srbija, 1998.

[17] S.Cheng, D.Shirmohammadi: A three phase power flow method for real time distribution system analysis, IEEE Transactions on Power Systems, Vol. 10, No. 2, pp. 671-679, 1995.

[18] D.Shirmohammadi, H.W.Hong, A.Semlyen, G.X.Luo: A Compensation-Based Power Flow Method For Weakly Meshed Distribution And Transmission Networks, IEEE Transactions on PS,

Vol. 3, No. 2, pp. 753-762, 1988.

[19] S.Pandit, S.A.Soman, S.A.Khaparde: Object-oriented network topology processor [power system automation], IEEE Computer applications in Power, Vol. 14, No. 2, pp. 42-46, 2001.

[20] M.E.Baran, A.W.Kelley: A Branch-Current-Based State Estimation method for Distribution Systems, IEEE Transactions on PS, Vol. 10, No. 1, pp. 483-491, 1995.

[21] M.He, J.Zhang: Fault Detection and Localization in Smart Grid: A Probabilistic Dependence Graph Approach, 2010 First IEEE International Conference on Smart Grid Communications, USA, pp. 43-48, 2010.

[22] G.Švenda, V.Strezoski: Distribution State Estimation: Wishes and Practical Possibilities, 2013 IEEE Power & Energy Society General Meeting – Shaping the Future Energy Industry – IEEE PESGM 2013, Vancouver, BC Canada, 21-25 July 2013, Panel Session: State Estimation for Distribution Operations: Sharing the Experiences of Implementation, Usage and Complexities, No. GM2515 – predavanje po pozivu IEEE PES Committees

[23] G.Švenda, V.Strezoski: Distribution State Estimation in Real-Life: Challenges and Experiences, 2014 IEEE Power & Energy Society General Meeting – IEEE PESGM 2014, Washington, National Harbor, MD, USA, 27-31 July 2014, Panel Session: State Estimation for Distribution System Monitoring and Control-Implementation Challenges, No. 14PESGM2761 – predavanje po pozivu IEEE PES Committees

[24] D.Čapko, A.Erdeljan, M.Popović, G.Švenda: An Optimal Initial Partitioning of Large Data Model in Utility Management Systems, Advances in Electrical and Computer Engineering, Vol. 11, No. 4, pp. 41-46, 2011.

[25] D.Čapko, A.Erdeljan, S.Vukmirović, I.Lendak: A Hybrid Genetic Algorithm for Partitioning of Data Model in Distribution Management Systems, Information Technology and Control, Vol. 40, No. 4, pp. 316-322, 2011.

[26] I.Lendak, E.Varga, A.Erdeljan, M.Gavric: RESTful web services and the Common Information Model (CIM), Energy Conference and Exhibition (EnergyCon), pp. 716-721, Manama, Bahrain, 2010.

[27] E.Varga, I.Lendak, M.Gavric, A.Erdeljan: Applicability of RESTful web services in control center software integrations, International Conference on Innovations in Information Technology – IIT, Abu Dhabi, United Arab Emirates, pp. 282–286, 2011.

[28] I.Foster, C.Kesselman, S.Tuecke: The anatomy of the grid: enabling scalable virtual organization, International Journal of Supercomputer Applications, Vol. 15, No. 3, pp. 200-222, 2001.

[29] A.Kaceniauskas, R.Pacevic, A.Bugajev, T.Katkevicius: Effective visualization by using Paraview software on Baltic grid, Information Technology and Control, Vol. 39, No. 2, pp. 108-15, 2010.

[30] A.Kaceniauskas: Solution and Analysis of CFD Applications by Using Grid Infrastructure, Information Technology and Control, Vol. 39, No. 4, pp. 284-90, 2010.

[31] D.Abramson, et al.: Deploying Scientific Applications to the PRAGMA Grid Testbed: Strategies and Lessons, Sixth IEEE International Symposium on Cluster Computing and the Grid, pp. 241-248, Vashington DC, USA, 2006.

[32] G.Allen, et al.: GridLab: Enabling applications on the Grid, Third International Workshop on Grid Computing (GRID '02), pp. 39-45, London, UK, 2002.

[33] E.Deelmana, et al.: Pegasus: Mapping scientific workflows onto the grid, 2-nd European across Grids Conference, pp. 11-20, Nicosia, Cyprus, 2004.

[34] E.Deelman, et al.: GriPhyN and LIGO, Building a virtual data grid for gravitational wave scientists, 11-th Intl. Symposium on High Performance Distributed Computing, pp. 225-234, Edinburgh, Scotland, 2002.

[35] G.B.Berriman, et al.: Montage: A Grid Enabled Image Mosaic Service for the National Virtual Observatory, Astronomical Data Analysis Software and Systems XIII, pp. 593-596, Strasbourg,

France, 2003.

[36] D.R.Simpson, et al.: GeneGrid: A practical workflow implementation for a grid based Virtual Bioinformatics Laboratory. In Proceedings of UK e-Science All Hands Meeting 2004 (AHM04), pp. 547–554, Nottingham, UK, 2004.

[37] R.Buyya, D. Abramson, S.Venugopal: The grid economy, In Proceedings of IEEE, Vol.93, No.3, pp. 698–714, 2005.

[38] D.Abramson, J.Kommineni, J.L.McGregor, J.Katzfey: An atmospheric sciences workflow and its implementation with web services, In Proceedings of International Conference on Computational Science, Krakow, Poland, pp. 164–173, 2004.

[39] M.Baker, R.Buyya, D.Laforenza: Grids and Grid Technologies for Wide-area Distributed Computing, Software: Practice and Experience, Vol. 32, No. 15, pp. 1437-1466, 2002.

[40] H.J.Siegel, S.Ali: Techniques for mapping tasks to machines in heterogeneous computing systems, Journal of Systems Architecture, Vol. 46, No. 8, pp. 627-639, 2000.

[41] T.D.Braun, et al.: A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems, In Proceedings of Heterogeneous Computing Workshop, pp. 15-29, San Juan, Puerto Rico, 1999.

[42] R.Armstrong, D.Hensgen, T.Kidd: The relative performance of various mapping algorithms is independent of sizable variances in runtime predictions, In Proceedings of Heterogeneous Computing Workshop, pp. 79-87, Orlando, USA, 1998.

[43] M.Maheswaran, S.Ali, H.J.Siegel, D.Hensgen, R.F.Freund: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, Journal of Parallel and Distributed Computing, Vol. 59. No. 2, pp. 107-131, 1999.

[44] F.Dong, S.G.Akl: Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, Technical Report No. 2006-504, Kingston, Canada, 2006.

[45] K.Cooper, et al.: New Grid Scheduling and Rescheduling Methods in the GrADS Project, In Proceedings of International Parallel and Distributed Processing Symposium (IPDPS'04), pp. 199--206, Santa Fe, USA, 2004.

[46] H.Chen, M.Maheswaran: Distributed Dynamic Scheduling of Composite Tasks on Grid Computing Systems, In Proceedings of Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 88-97, Fort Lauderdale, USA, 2002.

[47] S.J.Chapin, D.Katramatos, J.Karpovich, A.S.Grimshaw: The Legion Resource Management System, In Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP '99), pp. 162-178, San Juan, Puerto Rico, 1999.

[48] H.G.Rotithor: Taxonomy of Dynamic Task Scheduling Schemes in Distributed Computing Systems, Computer and Digital Techniques, Vol. 141, No. 1, pp. 1-10, 1994.

[49] A.K.Aggarwal, R.D.Kent: An Adaptive Generalized Scheduler for Grid Applications, In Proceedings of High Performance Computing Systems and Applications (HPCS'05), pp.15-18, Ontario, Canada, 2005.

[50] H.A.James: Scheduling in Metacomputing Systems, Ph.D. Thesis, The Department Of Computer Science, University of Adelaide, Australia, 1999.

[51] N.Spring, R.Wolski: Application Level Scheduling of Gene Sequence Comparison on Metacomputers, In Proceedings of International Conference on Supercomputing (ICS'98), pp. 141-148, Melbourne, Australia, 1998.

[52] J.Gehring, T.Preiss: Scheduling a Metacomputer with Uncooperative Sub-schedulers, In Proceedings of Job Scheduling Strategies for Parallel Processing, pp. 179–201, San Juan, Puerto Rico, 1999.

[53] A.Gokhan Ak, G.Cansever, A.Delibasi: Robot Trajectory Tracking with Adaptive RBFNN-Based Fuzzy Sliding Mode Control, Information Technology and Control, Vol. 40, No. 2, pp. 151-156, 2011.

[54] K.Siwek, S.Osowski, R.Szupiluk: Ensemble neural network approach for accurate load forecasting in a power system, Journal of Applied Mathematics and Computer Science, Vol. 19, No. 2, pp. 303-315, 2009.

[55] A.Kuczapski, M.V.Micea, L.A.Maniu, V.I.Cretu: Efficient generation of near optimal initial populations to enhance genetic algorithms for job-shop scheduling, Information Technology and Control, Vol. 39, No. 1, pp. 32-37, 2010.

[56] R.Braun, et al.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, Parallel and Distributed Computing, Vol. 61, No. 6, pp. 810-837, 2001.

[57] V.Di Martino, M.Mililotti: Sub optimal scheduling in a grid using genetic algorithms, Parallel Computing, Vol. 30, pp. 553-565, 2004.

[58] U.Fissgus: Scheduling Using Genetic Algorithms, 20-th IEEE International Conference on Distributed Computing Systems, pp. 662-699, Taipei, Taiwan, 2000.

[59] N.Nedić, S.Vukmirović, A.Erdeljan, L.Imre, D.Čapko: A Genetic Algorithm Approach for Utility Management System Workflow Scheduling, Information Technology and Control, Vol. 39, No. 4, pp. 310-316, 2010.

[60] S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko, N.Nedić: Optimization of workflow scheduling in Utility Management System with hierarchical neural network, International Journal of Computational Intelligence Systems, Vol. 4, No. 4, pp. 672-679, 2011.

[61] S.Vukmirović, A.Erdeljan, I.Lendak, N.Nedić: Neural network workflow scheduling for large scale Utility Management Systems, Systems Man and Cybernetics (SMC), pp. 2307- 2311, Istanbul, Turkey, 2010.

[62] [S.Vukmirović – Adaptive, 2010] S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko, N.Nedic: Adaptive Neural Network Workflow Management for Utility Management Systems, Neural Network Applications in Electrical Engineering (NEUREL 2010) pp. 93-96, Belgrade, Serbia, 2010.

[63] [S.Vukmirović - Hierarchical, 2010] S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko, N.Nedić: Hierarchical Neural Model for Workflow Scheduling in Utility Management Systems, International Workshop on Soft Computing Applications (SOFA 2010), pp. 51-56, Arad, Romania, 2010.

[64] M.Iverson, F.Ozguner: Dynamic, competitive scheduling of multiple DAGs in a distributed heterogeneous environment, Heterogeneous Computing Workshop, pp. 70-78, Orlando, Florida, USA, 1998.

[65] A.Andrzejak, D.Kondo, D.P.Anderson: Exploiting non-dedicated resources for cloud computing, Network Operations and Management Symposium (NOMS), pp. 341-348, Osaka, Japan, 2010.

[66] M.Wu, X.Sun: A General Self-Adaptive Task Scheduling System for Non-Dedicated Heterogeneous Computing, In Proceedings of IEEE International Conference on Cluster Computing (CLUSTER'03), pp.354-361, Hong Kong, China, 2003.

[67] V.Toporkova, A.Tselishchevb, D.Yemelyanova, A.Bobchenkova: Composite Scheduling Strategies in Distributed Computing with Non-dedicated Resources, Procedia Computer Science, Vol. 9, pp. 176–185, 2012.

[68] A.Chervenak, I.Foster, C.Kesselman, C.Salisbury, S.Tuecke: The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets, Journal of Network and Computer Applications, Vol. 23, No. 3, pp. 187–200, 2000.

[69] S.Venugopal, R.Buyya, K.Ramamohanarao: A taxonomy of data grids for distributed data sharing, management, and processing, ACM Computing Surveys, Vol. 38, No. 1, Article 3, 2006.

[70] I.Foster, C.Kesselman: The grid: blueprint for a new computing infrastructure, Morgan Kaufmann Publishers Inc, 2nd edition, USA, 2004.

[71] LPearlman, et al: The community authorization service: status and future, In Proceedings of

Computing in High Energy Physics (CHEP ’03), San Diego, USA, page(s) 9, 2003.

[72] R.Alfieri, et al.: From gridmap-file to VOMS: managing authorization in a grid environment, Future Generation Computer Systems, Vol. 21, No. 4, pp. 549–558, 2005.

[73] L.Seitz, J.-M.Pierson, L.Brunie: Sygn: A certificate based access control in grid environments, Technical Report, France, 2005.

[74] G.Kola, M.Livny: Diskrouter: A flexible infrastructure for high performance large scale data transfers, Technical report cs-tr-2003-1484, USA, 2003.

[75] B.Allcock, et al.: Secure, efficient data transport and replica management for high-performance data-intensive computing, In Proceedings of IEEE Symposium on Mass Storage Systems (MSS 2001), Washington, DC, USA, page(s) 13, 2001.

[76] J.Yu, R.Buyya: A Taxonomy of Workflow Systems for Grid Computing, Journal of Grid Computing, Vol. 3, No. 4, pp. 171-200, 2005.

[77] A.Chervenak, et al.: Giggle: A Framework for Constructing Scalable Replica Location Services, Supercomputing (SC2002), page(s) 58, Baltimore, USA, 2002.

[78] D.S.Milojicic, et al.: Peer-to-peer computing, Technical Report HPL-2002-57, USA, 2002.

[79] R.Schollmeier: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications, In Proceedings of International Conference on Peer-to-Peer Computing (P2P’01), pp. 101–102, Washington, DC, USA, 2001.

[80] M.P.Singh: Peering at peer-to-peer computing, IEEE Internet Computing, Vol. 5, No. 6, pp. 4–5, 2001.

[81] W.M.P. van der Aalst, A.H.M.ter Hofstede, B.Kiepuszewski, A.P.Barros: Workflow Patterns; Technical Report, Eindhoven University of Technology, The Netherlands, 2000.

[82] N.Russell, et al.: Workflow Control-Flow Patterns, Technical Report, BPM Center Report BPM-06-22, 2006.

[83] G.C.Fox, D.Gannon: Workflow in grid systems, Concurrency and Computation: Practice & Experience, Vol. 18, No. 10, pp. 1009–1019, 2006.

[84] R.Sakellariou, H.Zhao: A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems, Scientific Programming, Vol. 12, No. 4, pp. 253-262, 2004.

[85] T.Tannenbaum, D.Wright, K.Miller, M.Livny: Condor: A Distributed Job Scheduler, Beowulf Cluster Computing with Linux, The MIT Press, MA, USA, 2002.

[86] A.Mayer, et al.: ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time, UK e-Science All Hands Meeting, Nottingham, UK, pp. 627-634, 2003.

[87] B. Ludäscher, et al.: Scientific Workflow Management and the KEPLER System, Concurrency and Computation: Practice & Experience - Workflow in Grid Systems, Vol. 18, No. 10, pp. 1039-1065, 2006.

[88] Z.Guan, et al.: Grid-Flow: A Grid-Enabled Scientific Workflow System with a Petri Net-based Interface, Concurrency and Computation Practice and Experience, Vol. 18, No. 10, pp. 1115-1140, 2006.

[89] W.M.P. van der Aalst: The application of Petri Nets to workflow management, Journal of Circuits, Systems and Computers, Vol. 8, No. 1, pp. 21-66, 1998.

[90] N.Adam, V.Aturi, W.Huang: Modeling and analysis of workflows using Petri Nets, Journal of Intelligent Information Systems, Vol. 10, No. 2, pp. 131–158, 1998.

[91] M.Hennessy: Algebraic Theory of Processes (Series in the Foundations of Computing), MIT Press, USA, 1998.

[92] D.Sangiorgi, D.Walker: The Pi-Calculus: A Theory of Mobile Processes, Cambridge University Press, UK, 2004.

[93] A.Lerina, C.Aniello, G.Pierpaolo, V.M.Luisa: FlowManager: A Workflow Management System

Based on Petri Nets, In Proceedings of Computer Software and Applications Conference (COMPSAC 2002), Oxford, UK, pp. 1054-1059, 2002.

[94] H.M.W.Verbeek, A.Hirnschall, W.M.P. van der Aalst: XRL/Flower: Supporting Inter-Organizational Workflows Using XML/Petri-nets Technology, In Proceedings of Workshop on Web Services, e-Business, and the Semantic Web (CAiSE 2002), pp. 93-108, Toronto, Canada, 2002.

[95] S.Hotovy: Workload evolution on the Cornell theory center IBM SP2, Workshop on Job Scheduling Strategies for Parallel Processing, pp. 27-40, Honolulu, Hawaii, USA, 1996.

[96] S.Jang, X.Wu, V.Taylor, G.Mehta, K.Vahi, E Deelman: Using Performance Prediction to Allocate Grid Resources. Technical Report 2004-25, USA, 2004.

[97] W.Smith, I.Foster, V.Taylor: Predicting Application Run Times Using Historical Information, Journal of Parallel and Distributed Computing, Vol. 64, No. 9, pp. 1007-1016, 2004.

[98] L.Ke, et al.: A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on Cloud Computing Platform, International Journal of High Performance Computing Applications, Vol. 24, No. 4, pp. 1-16, 2010.

[99] P.Kumar, V.Anandarangan, A.Reshma: An Approach to Workflow Scheduling using Priority in Cloud Computing Environment, International Journal of Computer Applications, Vol. 109, No. 11, pp. 32-38, 2015.

[100] D.Harel, A.Pnueli: On the development of reactive systems, Logics and models of concurrent systems, Springer-Verlag New York, pp. 477–498, New York, USA, 1985.

[101] D.Park: Concurrency and Automata on Infinite Sequences, In Proceedings of GI-Conference on Theoretical Computer Science, pp. 167-183, London, UK, 1981.

[102] N.Nedić, G.Švenda: Workflow Management System for DMS, Information Technology and Control, Vol. 42, No. 4, pp. 373-385, 2013.

[103] H.Topcuoglu, S.Hariri, M.Y.Wu:. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.3, pp. 260-274, 2002.

[104] [H. Casanova, 2000] H.Casanova et al.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, The 9th Heterogeneous Computing Workshop (HCW'00), pp. 349-363, Cancun, Mexico, 2000.

[105] W.M.P. van der Aalst: Fundamental methods, and systems, MIT Press: Cambridge, USA, 2002.

[106] E.Deelman, et al.: Mapping Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, Vol. 1, No. 1, pp. 25-39, 2003.

[107] B.Ludäscher, et al.:Scientific Workflow Management and the Kepler System, Concurrency and Computation: Practice and Experience, Vol. 18, No. 10, pp. 1039–1065, 2006.

[108] S.McGough, L.Young, A.Afzal, S.Newhouse, J.Darlington: Workflow Enactment in ICENI. In Proceedings of UK e-Science All Hands Meeting, pp. 894-900, Nottingham, UK, 2004.

[109] D.Churches, et al.: Programming scientific and distributed workflow with triana services, Concurrency and Computation: Practice & Experience, Vol. 18 No. 10, pp.1021–1037, 2006.

[110] I.Altintas, et al.: A Framework for the Design and Reuse of Grid Workflows, Scientific Applications on Grid Computing (SAG 2004,), pp. 120-133, Beijing, China, 2004.

[111] K.Amin, et. al.: GridAnt: A Client-Controllable Grid Workflow System, In Proceedings of Hawaii International Conference on System Sciences (HICSS'04), page(s) 10, Big Island, USA, 2004.

[112] Y.Zhang, C.Koelbel, K.Cooper: Hybrid Re-scheduling Mechanisms for Workflow Applications on Multi-cluster Grid, Cluster Computing and the Grid (CCGRID '09), pp. 116 - 123, Shanghai, China, 2009.

[113] K.Kurowski, et. al: Improving Grid Level Throughput Using Job Migration And Rescheduling,

Scientific Programming, Vol. 12, No. 4, pp. 263-273, 2004.

[114] R.Garg, A.K.Singh: Fault Tolerance In Grid Computing: State of the Art and Open Issues, International Journal of Computer Science & Engineering Survey, Vol. 2 , No. 1, pp. 88-97, 2011.

[115] P.Kurdel, J.Sebestyenova: Grid workflow specification and verification, WSEAS Transactions on Computers, Vol. 8, No. 7, pp. 1199-1208, 2008.

[116] J.Chen, Y.Yang: A taxonomy of grid workflow verification and validation, Concurrency and Computation: Practice and Experience, Vol. 20, No. 4, pp. 347-360, 2008.

[117] H.Li, Y.Yang, T.Y.Chen: Resource constraints analysis of workflow specifications, The Journal of Systems and Software, Vol. 73, No. 2, pp. 271–285, 2004.

[118] H.Li, Y.Yang: Dynamic checking of temporal constraints for concurrent workflows, Electronic Commerce Research and Applications, Vol. 4, No. 2, pp. 124–142, 2005.

[119] E.Bertino, E.Ferrari, V.Atluri: An authorisation model for supporting the specification and enforcement of authorisation constraints in workflow management systems, ACM Transactions on Information System Security Vol. 2, No. 1, pp. 65–104, 1999.

[120] S.Wu, A.Sheth, Z.Luo: Authorisation and access control of application data in workflow systems, Journal of Intelligent Information Systems, Vol. 18, No. 1, pp. 71-94, 2002.

[121] R.Buyya, S.Venugopal: The Gridbus toolkit for service oriented grid and utility computing: An overview and status report. Technical Report GRIDS-TR-2004-2, University of Melbourne, Australia, 2004.

[122] R.Medeiros, W.Cirne, F.Brasileiro, J.Sauve: Faults in grids: why are they so bad and what can be done about it?, In Proceedings of International workshop on Grid Computing, pp. 18–24, Phoenix, USA, 2003.

[123] S.Brunett, et al.: Application Experiences with the Globus Toolkit, In Proceedings of High Performance Distributed Computing, pp. 81-88, Chicago, USA, 1998.

[124] S.Hwang, C. Kesselman: A Flexible Framework for Fault Tolerance in the Grid, Journal of Grid Computing, Vol. 1, No. 3, pp. 251-272, 2003.

[125] J.H.Abawajy: Fault-Tolerant Scheduling Policy for Grid Computing Systems. In Proceedings of Parallel and Distributed Processing Symposium (IPDPS'04), pp. 238-244, Santa Fe, USA, 2004.

[126] M.Chtepen, et al.: Evaluation of Replication and Rescheduling Heuristics for Grid Systems with Varying Availability, In Proceedings of Parallel and Distributed Computing and Systems, page(s) 6, Dallas, USA, 2006.

[127] N.Budhiraja, K.Marzullo, F.B.Schenider, S.Toueg: The Primary-Backup Approach, Distributed Systems (Second Edition), ACM Press/Addison-Wesley, pp. 199-216, 1993.

[128] F.B.Schneider: Replication Management using the State-Machine Approach, Distributed Systems (Second Edition), ACM Press/Addison-Wesley, pp. 169-197, 1993.

[129] J.B.Weissman: Gallop: The Benefits of Wide-Area Computing for Parallel Processing, Journal of Parallel and Distributed Computing, Vol. 54, No. 2, pp. 183-205, 1998.

[130] Y.Li, Z.Lan: Exploit failure prediction for adaptive fault-tolerance in cluster, In Proceedings of Cluster Computing and the Grid (CCGRID 06), pp. 531-538, Singapore, 2006.

[131] G.Sabin, R.Kettimuthu, A.Rajan, P.Sadayappan: Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment, In Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP 2003), pp. 87-104, Seattle, USA, 2003.

[132] V.Hamscher, U.Schwiegelshohn, A.Streit, R.Yahyapour: Evaluation of Job-Scheduling Strategies for Grid Computing. In Proceedings of IEEE/ACM International Workshop on Grid Computing (Grid 2000), pp.191-202, London, UK, 2000.

[133] R.Sakellariou, H.Zhao: A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems, In Proceedings of Parallel and Distributed Processing Symposium (IPDPS'04), pp. 111-123, Santa Fe, USA, 2004.

[134] A.Radulescu, A.J.C. van Gemund: On the Complexity of List Scheduling Algorithms for Distributed Memory Systems, In Proceedings of the 13th International Conference on Supercomputing, pp. 68-75, Portland, USA, 1999.

[135] J.H.Son, M.H.Kim: Analyzing the critical path for the well-formed workflow schema, Database Systems for Advanced Applications, pp. 146-147, Hong Kong, China, 2001.

[136] M.Arora, S.K.Das, R.Biswas: A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments, International Conference on Parallel Processing Workshops, pp. 499-505, Vancouver, Canada, 2002.

[137] H.Shan, L.Oliker, R.Biswas, W.Smith: Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration, In Proceedings of Advanced Computing and Communication (ADCOM2004), page(s) 8, Ahmedabad Gujarat, India, 2004.

[138] W.Aiello, B.Awerbuch, B.Zkfaggs, S.Rao: Approximate load balancing on dynamic and asynchronous networks, In Proceedings of Symposium on Theory of Computing (STOC'93), pp. 632–641, San Diego, USA 1993.

[139] F.Magoules: Fundamentals of Grid Computing: Theory, Algorithms and Technologies, CRC Press, 2010.

[140] R.Diekmann, A.Frommer, B.Monien: Efficient schemes for nearest neighbor load balancing, Parallel Computing, Vol. 25, No. 7, pp. 789–812, 1999.

[141] G.Cybenko: Dynamic load balancing for distributed memory multiprocessors, Journal of Parallel and Distributed Computing, Vol. 7, No. 2, pp. 279-301, 1989.

[142] C.Xu, F.Lau: Optimal parameters for load balancing with the diffusion method in mesh networks, Parallel Processing Letters, Vol. 4, No. 2, pp. 139-147, 1994.

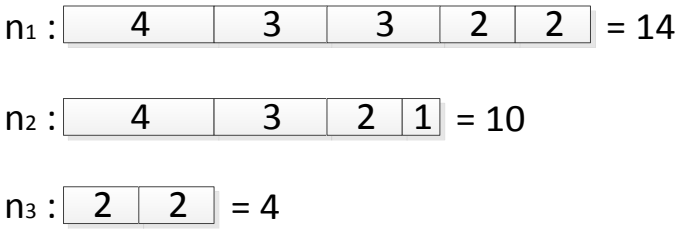
[143] J.Boillat: Load balancing and Poisson equation in a graph, Concurrency: Practice and Experience, Vol. 2, No. 4, pp. 289–313, 1990.

[144] C.Xu, F.Lau: Analysis of the generalized dimension exchange method for dynamic load balancing, Journal of Parallel and Distributed Computing, Vol. 16, No. 4, pp. 385–393, 1992.

[145] A.Sider, R.Couturier: Fast load balancing with the most to least loaded policy in dynamic networks, The Journal of Supercomputing Vol. 49, No. 3, pp. 291-317, 2009.

11. DODATAK A – PRIMER UPOTREBE DISTRIBUIRANOG ALGORITMA RASPODELE

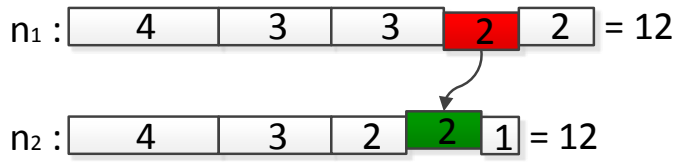
U ovom delu je prikazan uprošćen primer preraspodele opterećenja po čvorovima za proračune. Opterećenje koje neki proračun (zadatak na čvoru za proračune) unosi u sistem predstavljeno je kao celobrojna vrednost. U primeru, mehanizam za proračune, pored posredničkog čvora, čine tri čvora za proračune koji imaju identične računarske kapacitete. Pretpostavka je da još uvek nije počelo izvršenje ni jednog zadatka. Početna stanje svih čvorova su prikazana na slici 11.1: ukupno opterećenje prvog čvora je 14, drugog 10, a trećeg 4. Vrednost parametara α je postavljena na $1/2$ ($\alpha = 1/2$).



Slika 11.1 – Početno opterećenje tri čvora za proračune

Prva iteracija počinje transferom odgovarajućih zadataka sa čvora n_1 na ostale čvorove za proračune. Izračunava se maksimalna, dozvoljena količina opterećenja koja može biti preneta između n_1 i nekog od preostalih čvorova. Međutim, s obzirom da je svaki zadatak nedeljiva celina, moguće je preneti samo grupu zadataka čije ukupno opterećenje ne prelazi prethodno dobijene vrednosti. Iz tog proizilazi da stvarna količina opterećenja koja će biti razmenjena zavisi od skupa zadataka čije izvršenje još nije počelo, a koji su dodeljeni čvoru za proračune n_1.

Transfer opterećenja sa čvora za proračune n_1 na čvor za proračune n_2 prikazan je na slici 11.2.



Slika 11.2 – Transfer zadatka koji ima opterećenje 2 sa n_1 na n_2

Maksimalna količina opterećenja koja može da se prenese u prvoj iteraciji označena je sa $\Delta nw^1_{n_1 n_2}$ i računa se kao proizvod koeficijenta difuzije (parametar α) i razlike ukupnog opterećenja čvora n_1 i čvora n_2:

$$\Delta nw^1_{n_1 n_2} = 0.5 \cdot (14 - 10) = 2. \quad (11.1)$$

Ukoliko je dobijena vrednosti veća od nule, potrebno je pronaći skup zadataka na čvoru n_1 čije ukupno opterećenje ne prelazi izračunatu vrednost. S obzirom da je maksimalna količina opterećenja koja može biti preneta jednaka dvojci, analizom zadataka dodeljenih čvoru za proračune n_1 uočava se da postoji zadatak koji nosi upravo ovu vrednosti opterećenja. Pronađeni zadatak prelazi sa n_1 na n_2, pa $\Delta^1_{n_1 n_2}$ iznosi:

$$\Delta^1_{n_1 n_2} = 2 . \tag{11.2}$$

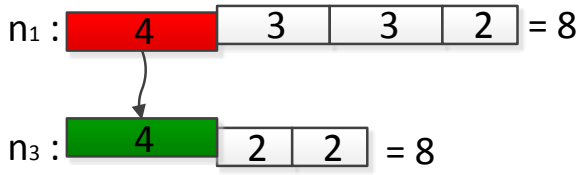
Vrednosti opterećenja čvorova za proračune n_1 i n_2 prikazane na slici 11.2 predstavljaju ukupna opterećenja nakon transfera zadataka. Na identičan način prikazana su ukupna opterećenja čvorova za proračune na svim slikama koje opisuju prelaz zadataka.

Dalja preraspodela opterećenja odvija se između čvorova za proračune n_1 i n_3. Nakon izračunate vrednosti $\Delta nw^1_{n_1 n_3}$, a na osnovu preostalih zadataka dodeljenih čvoru n_1 i opterećenja koje oni unose, određuje se $\Delta^1_{n_1 n_3}$. U tekućoj razmeni, maksimalna, dozvoljena količina opterećenja iznosi:

$$\Delta nw^1_{n_1 n_3} = 0.5 \cdot (12 - 4) = 4 . \tag{11.3}$$

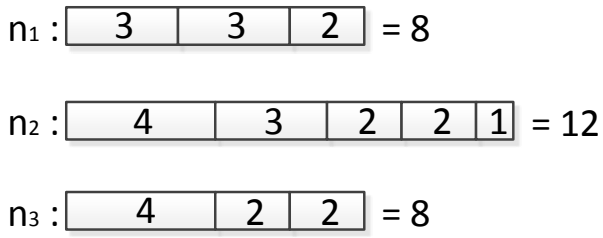
Na slici 11. 3, slično kao i u prethodnom slučaju je dovoljno preneti jedan zadatak, pa je:

$$\Delta^1_{n_1 n_3} = 4 . \tag{11.4}$$



Slika 11.3 – Transfer zadatka koji ima opterećenje 4 sa n_1 na n_3

Na slici 11.4 prikazano je ukupno opterećenje na čvorovima za proračune nakon transfera opterećenja sa čvora n_1 na ostale čvorove.



Slika 11.4 – Opterećenje čvorova za proračune u prvoj iteraciji nakon transfera zadataka sa n_1

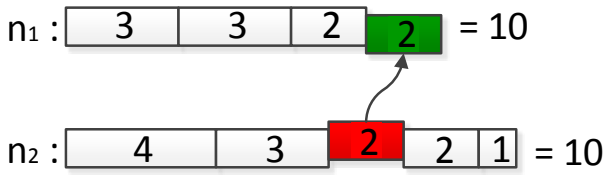
Sledeći korak u prvoj iteraciji je razmena opterećenja između čvora za proračune n_2 i ostalih čvorova. Aktivnost počinje razmenom sa čvorom za proračune n_1. U ovoj situaciji je potrebno odrediti koja količina opterećenja, odnosno koji skup zadataka, treba preneti sa čvora n_2 na čvor n_1. Kao i u prethodnim slučajevima, na osnovu ukupnog opterećenja na oba čvora, prvo se određuje maksimalna dozvoljena količina opterećenja za razmenu:

$$\Delta nw^1_{n_2 n_1} = 0.5 \cdot (12 - 8) = 2 . \quad (11.5)$$

Daljom analizom zadataka koji su na čvoru za proračune n_2 određuje se $\Delta^1_{n_2 n_1}$:

$$\Delta^1_{n_2 n_1} = 2 . \quad (11.6)$$

Na slici 11.5 prikazan je prelaz zadatka čije je opterećenje dva, sa čvora n_2 na čvor n_1.



Slika 11.5 – Transfer zadatka koji ima opterećenje 2 sa n_2 na n_1

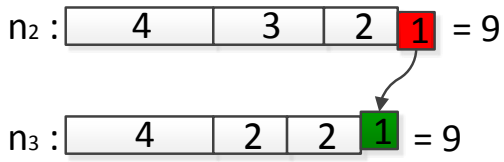
U daljem toku se vrši razmena opterećenja sa čvorom za proračune n_3. Kao i do sada, prvi potez predstavlja izračunavanje vrednosti $\Delta nw^1_{n_2 n_3}$:

$$\Delta nw^1_{n_2 n_3} = 0.5 \cdot (10 - 8) = 1 . \quad (11.7)$$

Na osnovu ove vrednosti određuje se $\Delta^1_{n_2 n_3}$ koje iznosi:

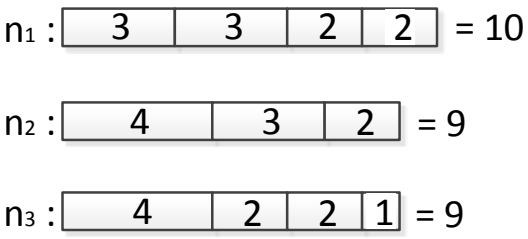
$$\Delta^1_{n_2 n_3} = 1 . \quad (11.8)$$

Razmena opterećenja je svedena na jedan zadatak čije je opterećenje jednako jedinici (slika 11.6):



Slika 11.6 – Transfer zadatka koji ima opterećenje 1 sa n_2 na n_3

Nova raspodela opterećenja u prvoj iteraciji, nakon transfera zadataka sa čvora za proračune n_2 na preostale čvorove za proračune, prikazana je na slici 11.7.



Slika 11.7 – Opterećenje čvorova za proračune u prvoj iteraciji nakon transfera zadataka sa n_2

Postupak se nastavlja poređenjem količine opterećenja na čvoru za proračune n_3 sa količinom opterećenja na ostalim čvorovima za proračune. Ukupno opterećenje koje su uneli zadaci dodeljeni čvoru n_3 nije veće od opterećenja na čvoru, onda je :

$$\Delta nw^1_{n_3 n_1} = 0.5 \cdot (9 - 10) < 0 . \quad (11.9)$$

Stoga neće biti razmene zadataka, pa je:

$$\Delta^1_{n_3 n_1} = 0 . \quad (11.10)$$

Ista situacija važi i za čvor n_2 :

$$\Delta nw^1_{n_3 n_2} = 0.5 \cdot (9 - 9) = 0 , \quad (11.11)$$

$$\Delta^1_{n_3 n_2} = 0 . \quad (11.12)$$

Druga iteracija počinje na identičan način kao prva, analizom i poređenjem količine opterećenja na čvoru za proračune n_1 sa ostalim čvorovima za proračune. Nakon toga eventualno sledi razmena zadataka.

Uočava se da razlika opterećenja između čvorova n_1 i n_2 nije dovoljno velika, odnosno manja je od opterećenja koje unosi bilo koji od zadataka dodeljen čvoru n_1. Iz tog razloga ne dolazi do razmene zadataka:

$$\Delta nw^2_{n_1 n_2} = 0.5 \cdot (10 - 9) = \frac{1}{2} , \quad (11.13)$$

$$\Delta^2_{n_1 n_2} = 0 . \quad (11.14)$$

Isto važi i za odnos opterećenja čvorova n_1 i n_3:

$$\Delta nw^2_{n_1 n_3} = 0.5 \cdot (10 - 9) = \frac{1}{2} , \quad (11.15)$$

$$\Delta^2_{n_1 n_3} = 0 . \quad (11.16)$$

Količina opterećenja na čvoru za proračune n_2 takođe nije dovoljno velika da bi moglo doći do razmene zadataka sa ostalim čvorovima:

$$\Delta nw^2_{n_2 n_1} = 0.5 \cdot (9 - 10) < 0 , \quad (11.17)$$

$$\Delta^2_{n_2 n_1} = 0 , \quad (11.18)$$

$$\Delta nw^2_{n_2 n_3} = 0.5 \cdot (9 - 9) = 0 , \quad (11.19)$$

$$\Delta^2_{n_2 n_3} = 0 . \quad (11.20)$$

Situacija je identična i za trenutnu količinu opterećenja na čvoru n_3:

$$\Delta nw^2_{n_3 n_1} = 0.5 \cdot (9 - 10) < 0 , \quad (11.21)$$

$$\Delta^2_{n_3 n_1} = 0 , \quad (11.22)$$

$$\Delta nw^2_{n_3 n_2} = 0.5 \cdot (9 - 9) = 0 , \quad (11.23)$$

$$\Delta^2_{n_3 n_2} = 0 . \quad (11.24)$$

Iz navedene situacije jasno se uočava da u toku druge iteracije distribuiranog algoritma za

preraspodelu opterećenja čvorova za proračune nije došlo do razmene zadataka između bilo koja dva čvora. Stoga se može zaključiti da je postignuta optimalna preraspodela opterećenja, odnosno da je ispunjen uslov za završetak algoritma:

$$\Delta_{n_i n_j} = 0,\ \forall(i,\ j),\ i \in \{1,2,3\} \wedge\ j \in \{1,2,3\}, \tag{11.25}$$

a samim tim je završena preraspodela opterećenja završena.

Dakle, na slici 11.7 je prikazano finalno stanje čvorova za proračune.

12. DODATAK B – BIBLIOGRAFIJA AUTORA

Rad u istaknutom međunarodnom časopisu (M22):

1. S.Vukmirović, A.Erdeljan, D.Čapko, I.Lendak, N.Nedić: Optimization of workflow scheduling in Utility Management System with hierarchical neural network; *International Journal of Computational Intelligence Systems*, Vol. 4, No. 4, pp. 672-679, 2011.

Radovi u međunarodnim časopisima (M23):

1. N.Nedić, G.Švenda: Workflow Management System for DMS; *Information Technology and Control*, Vol. 42, No. 4, pp. 373-385, 2013.
2. N.Nedić, S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko: A genetic algorithm approach for utility management system workflow scheduling; *Information Technology and Control*, Vol. 39, No. 4, pp. 310-316, 2010.

Saopštenja sa međunarodnih skupova štampana u celini (M33):

1. S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko, N.Nedić: Hierarchical Neural Model for Workflow Scheduling in Utility Management Systems; *International Workshop on Soft Computing Applications (SOFA 2010)*, Arad, Romania, pp. 51-56, 2010.
2. S.Vukmirović, A.Erdeljan, I.Lendak, N.Nedić: Neural network workflow scheduling for large scale Utility Management Systems; *Systems Man and Cybernetics (SMC)*, Istanbul, Turkey, pp. 2307-2311, 2010.
3. S.Vukmirović, A.Erdeljan, I.Lendak, D.Čapko, N.Nedić: Adaptive Neural Network Workflow Management for Utility Management Systems; *Symposium on Neural Network Applications in Electrical Engineering (NEUREL 2010)*, Belgrade, Serbia, 2010.

Radovi u časopisima od nacionalnog značaja (M52):

1. D.Dimitrijević, V.Dimitrieski, N.Nedić: Prototype Implementation of a Scalable Real-Time Dynamic Carpooling and Ride-Sharing Application; *Informatica*, Vol. 38, No. 3, pp. 213-221, 2014.

Saopštenja sa skupa od nacionalnog značaja štampana u celini (M63):

1. N.Nedić, A.Erdeljan, I.Lendak, D.Čapko: Pristup modelu podataka elektroenergetskog sistema putem web servisa; *Etran*, Vrnjačka Banja, Srbija, 2009.