Postupci raspoređivanja zadataka u sustavima s jednim i više poslužitelja

Cikač, Matko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

POSTUPCI RASPOREĐIVANJA ZADATAKA U SUSTAVIMA S JEDNIM I VIŠE POSLUŽITELJA

Diplomski rad

Matko Cikač

Osijek, 2017.

SADRŽAJ

1. UVOD

Većina ugrađenih suvremenih računalnih sustava ima ograničenja ″stvarnog vremena″. Riječ je o računalnim sustavima korištenim u proizvodnji, prijevoznim sustavima i mnogim drugim tehničkim procesima gdje je pravovremenost informacije najbitnija. Stara informacija nije samo beskorisna nego i pogrešna, što uvelike može utjecati na funkcionalnost sustava. Stvarno vrijeme znači kako informacijski sustav više ne kontrolira vremensku domenu, nego je vremenska domena proizvod okruženja u kojem se nalazi. Okruženje određuje vremenske uvjete procesa unutar samog sustava.

Programska logika aplikacije u sustavu stvarnog vremena svodi se na vremenske zahtjeve zadataka unutar istog sustava. Bitne informacije u takvim sustavima postaju najraniji početak izvršavanja nekog zadatka kao i njegovo najkasnije izvršavanje. Prethodno navedeni uvjeti se skupa s programskom logikom sustava mogu definirati kao specifikacija računalnog sustava koji kontrolira i koordinira što napraviti i u koje vrijeme. Na temelju tih uvjeta u radu su opisane osnovne tehnike korištene u računalnim sustavima stvarnog vremena za raspoređivanje i izvođenje zadataka. Osnovne tehnike u računalnim sustavima se temelje na algoritmima za raspoređivanje zadataka i analizi raspoređivanja u računalnim sustavima s jednim, dva ili više poslužitelja.

U drugom poglavlju opisana je podjela algoritama raspoređivanja zadataka u računalnim sustavima ovisno o stanju zadataka i ovisno o vremenskim uvjetima za izvođenje zadatka. Opisani su algoritmi koji se koriste u računalnim sustavima s jednim ili s više poslužitelja, te njihova usporedba. Princip funkcioniranja pojedinih algoritama raspoređivanja zadataka prikazan je eksperimentalnim dijelom rada, odnosno programskim rješenjem. U trećem poglavlju opisano je programsko rješenje, te su prikazani algoritmi i alati korišteni prilikom izrade. Četvrto poglavlje prikazuje rad programskog rješenja s konkretnim primjerima zadataka i analizu dobivenih rezultata ovisno o vrsti zadataka, obrazloženje o korištenom algoritmu za raspoređivanje zadataka, te o broju poslužitelja. Usporedba prirode i ponašanja algoritama temelji se na vrijednostima ključnih parametara za optimalan rad računalnog sustava.

2. RASPOREĐIVANJE ZADATAKA U RAČUNALNIM SUSTAVIMA

Raspoređivanje zadataka (engl. *task scheduling*) u računalnim sustavima definira uređenje i vremensko raspoređivanje zadataka u slučajevima dodjeljivanja zadataka skupu poslužitelja u računalnom sustavu. Nastavno na raspoređivanje zadataka, postoje slučajevi kada ima mnogo više zadataka spremnih za izvođenje nego što postoji fizičkih procesora. Kako bi se omogućilo istodobno izvođenje više zadataka na računalu, potrebno je na što kvalitetniji i učinkovitiji način raspodijeliti zadatke u operacijskom sustavu. Postoje razne metode raspoređivanja zadataka, iste su proučene i istražene u narednim potpoglavljima ovog rada. Istraživanje se grana na raspoređivanje zadataka u sustavima s jednim poslužiteljem i raspoređivanje zadataka u sustavima s dva ili više poslužitelja.

Svi zadaci u operacijskom sustavu spremni za izvođenje ne mogu se izvoditi istovremeno. Naime, njihovu preraspodjelu izvođenja kontrolira i koordinira raspoređivač zadataka na temelju odabranog algoritma za raspoređivanje zadataka u sustavima s jednim, dva ili više poslužitelja. Raspoređivač zadataka na temelju odabranog algoritma za raspoređivanje zadataka izvodi, kontrolira i koordinira redoslijed izvođenja zadataka, prioritet izvođenja zadataka, te vrijednost vremenskog odsječka, odnosno vremenskog intervala koji će biti dodijeljen pojedinom zadatku. Krajnjem korisniku koji sve nadgleda kroz prizmu operacijskog sustava najbitniji je dojam istovremenog izvođenja zadataka. Prema [16], zadacima se dodjeljuju kratki vremenski odsječci, odnosno vremenski intervali koji traju između minimalno 1 milisekunde i maksimalno 100 milisekundi zbog dojma istovremenog izvođenja.

Pri pravilnom i optimalnom raspoređivanju zadataka u obzir se uzima i prioritet zadatka, te se na temelju prioriteta zadatka odlučuje kada će zadatak dobiti procesorsko vrijeme i kolika će biti vrijednost vremenskog odsječka, odnosno vremenskog intervala. Svrha istraživanja i proučavanja problematike raspoređivanja zadataka jest upoznavanje s općim metodama i načinima raspoređivanja zadataka u različitim slučajevima računalnih sustava s jednim, dva ili više poslužitelja.

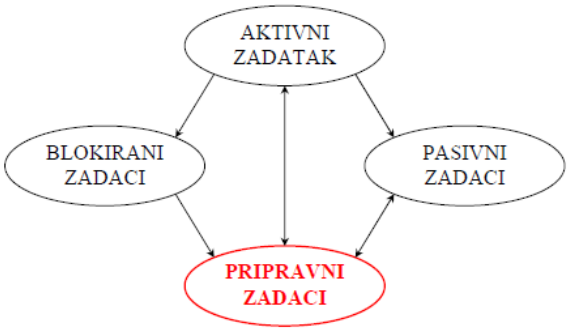
2.1 Raspoređivanje zadataka u računalnim sustavima s jednim poslužiteljem

Raspoređivanje zadataka u računalnim sustavima se razlikuje ovisno o broju poslužitelja. Računalni sustav s jednim poslužiteljem predstavlja sustav u kojem će sve zadatke izvoditi jedan

procesor po određenom rasporedu. Prilikom raspoređivanja zadataka mogu nastati poteškoće u slučaju da u sustavu postoji više zadataka nego što ima procesorskih jedinica. Ukoliko nastanu moguće poteškoće, sustav radi raspored dodjele procesorskog vremena među zadacima koji su u stanju pripravnosti. Svaki zadatak ima različita svojstva i zahtjeve prema raspoređivaču zadataka, stoga vremenski kritični zadaci moraju zadovoljiti svoje vremenske zahtjeve.

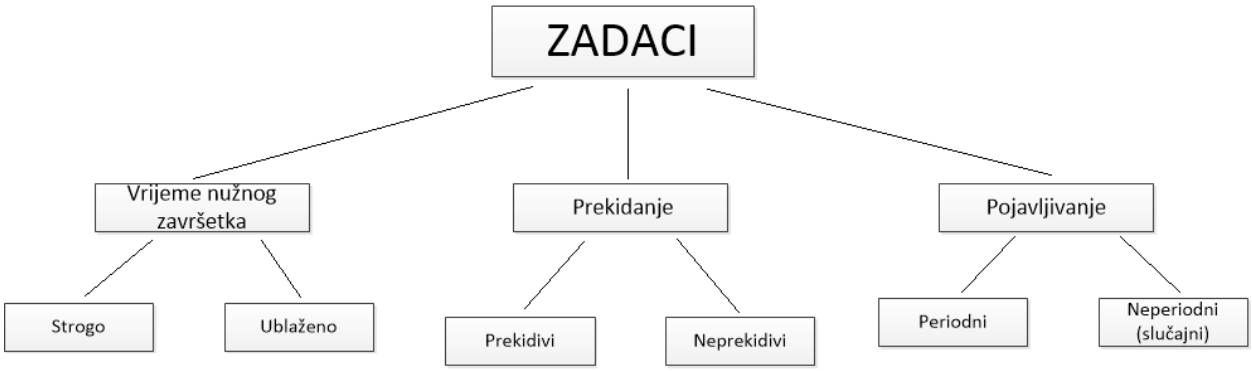
2.1.1 Klasifikacija zadataka u računalnom sustavu

Kada se govori o raspoređivanju zadataka, prije svega, bitno je naglasiti da se raspoređuju samo pripravni zadaci. Prema [1], stanja u kojem zadatak može biti su: aktivno stanje koje znači da se zadatak trenutno izvodi na procesoru. Pripravno stanje što znači da je zadatak spreman za izvođenje. Blokirano ili pasivno stanje, zadatak ili čeka na nekom sredstvu ili je završen, a možda još nije niti pokrenut. Prema slici 2.1 predstavljen je međusobni odnos stanja zadataka u računalnom sustavu, te način prijelaza stanja.



Slika 2.1 *Dijagram odnosa stanja zadataka*

Kako bi raspoređivanje zadataka bilo optimalno za cijeli računalni sustav, te kako bi se zadaci izveli, odnosno raspodijelili u najboljim mogućim vremenskim odsječcima, odnosno vremenskim intervalima, pripravne zadatke se nadalje može podijeliti ovisno o vremenu nužnog završetka (engl. *deadline, drop dead time*), o prekidanju zadataka, te o pojavljivanju zadatka. Vrijeme nužnog završetka (T_d) je strogo vremensko ograničenje do kojega zadatak mora završiti, a ovo se vremensko ograničenje zadataka najčešće pojavljuje u računalnim sustavima stvarnog vremena. Također, uz strogo vremensko ograničenje, postoji i blago, odnosno, ublaženo vremensko ograničenje, no kako je već prethodno navedeno razmatrat će se samo strogo vremensko ograničenje do kojeg zadatak mora završiti. Zadaci s obzirom na prekidanje mogu biti prekidivi i neprekidivi, te zadaci s obzirom na pojavljivanje mogu biti periodni (ponavljaju se u jednakim vremenskim odsječcima, odnosno vremenskim intervalima) i aperiodni zadaci, slučajni zadaci su posebna vrsta zadataka. Podjela zadataka prikazana je u nastavku slika 2.2.



Slika 2.2 *Prikaz podjele zadataka*

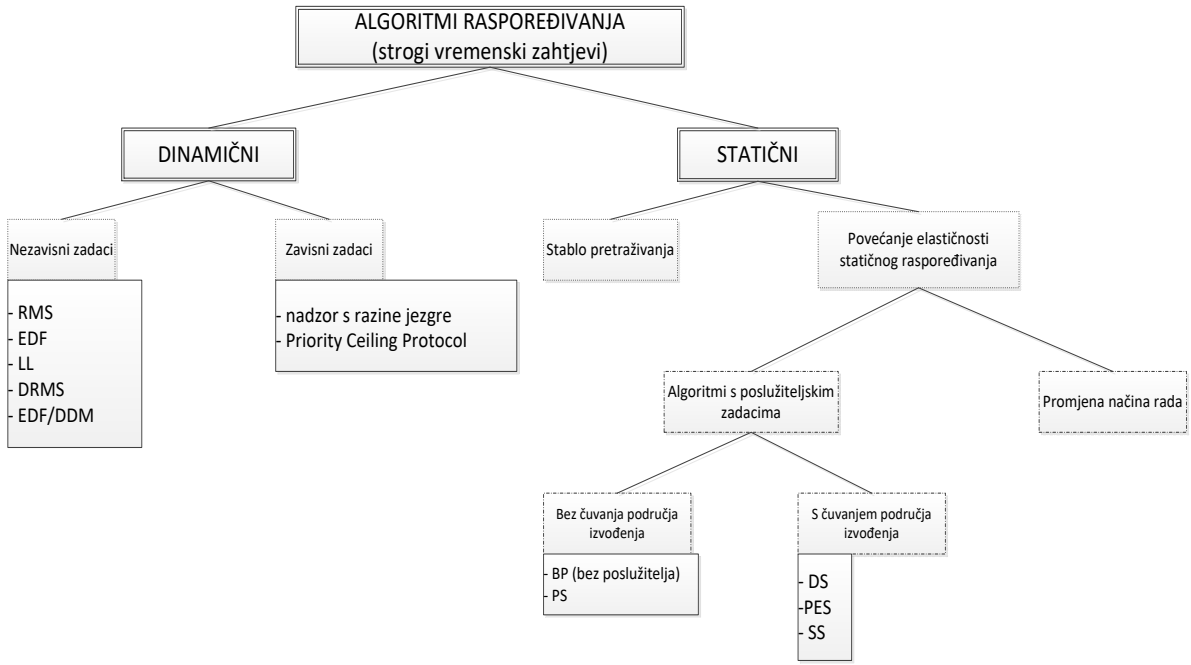
Raspoređivač zadataka ima funkciju raspoređivanja i kao takav bitan je dio višezadaćnog sustava. Namjena raspoređivača zadataka je da u suradnji s cijelim računalnim sustavom, omogući izvođenje više različitih zadataka istovremeno ili da u slučaju kad to nije moguće, omogući da izvođenje zadataka prividno izgleda istovremeno, odnosno da se dobije dojam izvođenja zadatka u stvarnom vremenu. Prema [2], cilj raspoređivanja zadataka u sustavima stvarnog vremena je dodjeljivanje procesora za obradu i resursa zadatku koji se nalazi u redu za obradu i to na takav način da su svi vremenski uvjeti ispunjeni.

Raspoređivači zadataka, prema [17], mogu biti dugoročni, srednjoročni i kratkoročni raspoređivač zadataka. Dugoročni raspoređivač zadataka odlučuje koji će zadaci biti propušteni u red čekanja spremnih procesa, što znači koji zadaci će imati odobrenje za izvršavanje, a koji zadaci trebaju napustiti sustav. Dugoročni raspoređivač zadataka koristi više metoda odlučivanja, kao što su metoda u kojoj se uzima u obzir red prispijeća ili metoda raspoređivanja zadataka po prioritetu. Dugoročni raspoređivač zadataka odlučuje o broju istovremeno izvršavanih zadataka i o upravljanju procesima koji rade ulazno – izlazne operacije. U operacijskim sustavima novije generacije, prema [17], dugoročni raspoređivač zadataka se koristi kako bi se osiguralo dovoljno procesorskog vremena i ispunjenje rokova završetka za zadatke stvarnog vremena.

Srednjoročni raspoređivač zadataka postoji u svim sustavima u kojima postoji virtualna memorija. Srednjoročni raspoređivač zadataka privremeno premješta procese iz glavne memorije u sekundarnu, kao što je čvrsti disk, ili obrnuto. Prema [17], suradnja srednjoročnog i kratkoročnog raspoređivača zadataka je jedna od ključnih stavki za kvalitetan i brz rad sustava. Budući da je sekundarna memorija vrlo spora, pojavljuje se usporavanje sustava ukoliko se pogrešni zadaci premještaju iz glavne memorije.

Kako je opisano u [17], kratkoročni raspoređivač zadataka radi odluku za procese u redu čekanja, odlučuje koji će biti izvršeni, a kojima će se nakon prekida sata, ulazno - izlaznog uređaja ili nekog drugog signala dodijeliti procesorsko vrijeme. Kratkoročni raspoređivač zadataka donosi odluke na osnovi ugrađenog algoritma raspoređivanja. Algoritam određuje tip operacijskog sustava (multifunkcionalni sustav, sustav za rad u stvarnom vremenu). Prema [17], postoje tri kriterija koje kratkoročni raspoređivač zadataka pokušava zadovoljiti, a to su: brze promjene zadataka, učinkovito izvršavanje dijeljenih podatkovnih struktura (podatkovne strukture bi se trebale nalaziti u memoriji), te davanje pravednih vremenskih udjela svakom zadatku u sustavu.

Prema [8], algoritam raspoređivanja definira pravednost raspoređivanja na temelju karakteristika jednostavnosti, kompatibilnosti s platformom računalnog i operacijskog sustava. Algoritam raspoređivanja koristi standardne funkcije, kao što su funkcije operacijskog sustava. Algoritmi se dijele ovisno o potrebama raspoređivanja, kao što je vidljivo na slici 2.3.



Slika 2.3 *Dijagram podjele algoritama raspoređivanja*

Prema [17], algoritmi raspoređivanja dijele se na više vrsta. Sama podjela algoritama raspoređivanja ovisi o vrsti problema u strogo definiranim i ograničenim vremenskim uvjetima. Iz tog razloga postoji više vrsta raspoređivanja kao što su: statičko, dinamičko i mješovito raspoređivanje. Statičko raspoređivanje se koristi ukoliko su poznate sve bitne značajke i svi zadaci koji se trebaju uzeti u obzir prilikom raspoređivanja. Statičko raspoređivanje pruža mogućnost neovisnog (engl. *off-line*) raspoređivanja za izvedbu određenih rasporeda zadataka, za

analizu ostvarivosti rasporeda zadataka i analizu raspoređivanja (engl. *schedulability analysis*). Korištenjem statičkog raspoređivanja kod raspoređivačkih zahtjeva potrebno je uzeti u obzir periodnost zahtjeva radi obrade računalnih vremena (engl. *computation times*). Raspoređivački zahtjevi, kao što je učestalost, primjenjuju se u upravljanju industrijskih procesa i rasporedi proizašli iz statičkog raspoređivanja su stalni (nepromjenjivi) zbog stalnih (engl. *fixed*) prioriteta dodanih zadacima za vrijeme izvršavanja aplikacije.

Svaka nova promjena, odnosno novi zahtjev koji se dovodi raspoređivaču, iziskuje potpunu preraspodjelu zadataka, shodno tome, postupci statičkog raspoređivanja ne mogu se nositi s nepredviđenim okolnostima (pojavama ili odstupanjima). Unatoč nefleksibilnosti, statičko raspoređivanje je vrlo povoljno zbog malih gubitaka i niskih troškova računalnog sustava, te mogućnosti neovisnog raspoređivanja. Predvidljivost (engl. *predictability*) je bitna karakteristika raspoređivača ove vrste, izrazito u slučaju njihove primjene u sustavima gdje je sigurnost kritična.

Dinamičko raspoređivanje obuhvaća algoritme koji su predviđeni za rad sa zadacima s nepredvidivim vremenom pojavljivanja ili nepredvidivim vremenom trajanja, odnosno izvođenja. Dinamičko raspoređivanje omogućuje promjenjive zahtjeve iz okoline, što je rezultat trenutne reakcije na novonastalu situaciju pri izvođenju. Prema [3], zbog prethodno navedenih karakteristika, dinamičko raspoređivanje ima drugačije karakteristike od statičkog raspoređivanja. Ukoliko se uzme u obzir i funkcionalnost, dinamičko raspoređivanje je fleksibilno, ali i skupo, zbog povećanog utroška vremena pri izvođenju raspoređivanja.

Mješovito raspoređivanje ima zadatak uzeti najbolje iz oba prethodno navedena raspoređivanja, tako da zadatke s poznatim vremenskim značajkama raspoređuje statički raspoređivač, koji ujedno prilagođava druge zadatke prilikom njihovog dolaska, odnosno nastanka.

2.1.2 Discipline raspoređivanja aperiodnih zadataka

Korištenjem statičkog raspoređivanja aperiodnih zadataka, koji su ujedno prisutni i u eksperimentalnom dijelu rada, koriste se i discipline (načini) raspoređivanja. Osnovno raspoređivanje (engl. *Basic Scheduling*, BS) može se nazvati neprekidivo, budući da zadatak koji je jednom započeo s izvođenjem, izvodi se bez prekidanja do kraja (engl. *Non – preemptive Scheduling*).

Prekidivo raspoređivanje (engl. *Preemptive Scheduling*, PSch) označava da se zadaci mogu prekinuti tokom izvođenja radi izvođenja drugih zadataka, što ovisi o prioritetu zadatka.

Kod općeg raspoređivanja (engl. *General Scheduling*, GS) procesor se tretira kao sredstvo koje u trenutku može biti dodijeljeno zadacima, odnosno zadaci crpe dio snage procesora. Ukoliko svi zadaci koriste dio procesorske snage za izvođenje, produljuju svoje vrijeme izvođenja razmjerno korištenoj procesorskoj snazi.

Prema [5], određeni udio snage računala koju dodijeljeni procesor može dati zadatku na korištenje se označava s α, što znači da $0 \leq \alpha \leq 1$. Nastavno na prethodnu tvrdnju, ukoliko zadatak treba *c* vremenskih jedinica za izvršavanje (izvođenje) na dodijeljenom procesoru, tada će procesoru s dodijeljenom snagom biti potrebno za potpuno izvođenje $\frac{c}{\alpha}$ vremenskih jedinica.

Prema [5], ukoliko postoji *k* identičnih procesora na kojima se istovremeno izvodi *n* zadataka, mora biti ispunjeno:

$$\sum_{i=1}^{n} a_i \leq k \tag{2-1}$$

gdje je:

α – udio snage računala koju procesor može koristiti za određeni zadatak,

k – broj identičnih procesora na kojima se u istom vremenu izvode zadaci i

n – broj zadataka koji se u isto vrijeme izvode na procesoru.

Shodno tome, niti jedan zadatak se ne može izvoditi istovremeno (paralelno) na više procesora, stoga samo različiti zadaci mogu koristiti određeni procesor na osnovi dijeljenja procesorske snage. Prema [7], ovaj način raspoređivanja zadataka je teorija, te predstavlja podlogu optimalnog raspoređivanja zadataka radi korištenja danih procesora pri korištenju prekidivog raspoređivanja.

Algoritam PS (engl. *Polling Server*) koji za svaku vremensku jedinicu provjerava spremnost poslužitelja. Ukoliko je poslužitelj spreman, provjerava se spremnost zadatka i izvodi li se zadatak na nekom drugom poslužitelju. Zadatak je spreman ako je uvjet zadovoljen. Ukoliko se ustanovi da zadatak nije spreman ili je u fazi izvođenja, prelazi se na sljedeći spremni zadatak. U slučaju da poslužitelj nije spreman, prelazi se na sljedeći poslužitelj. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, algoritam se završava. Najčešća je primjena algoritma PS u računalnim mrežama, telekomunikacijama i upravljanju cestovnim prometom.

Algoritam DS (engl. *Deferrable Server*) za razliku od prethodno objašnjenog algoritma PS, za svaku vremensku jedinicu uz provjeru spremnosti poslužitelja, provjerava ima li raspoloživog kapaciteta. Potrebno je čuvati preostali kapacitet poslužitelja. Ukoliko je poslužitelj spreman i ima kapaciteta, provjerava se spremnost zadatka i izvodi li se zadatak na nekom drugom poslužitelju. Ako je uvjet zadovoljen, zadatak je spreman. Ako zadatak nije spreman ili je u fazi izvođenja, prelazi se na sljedeći spremni zadatak. U slučaju da poslužitelj nije spreman ili nema kapaciteta, prelazi se na sljedeći poslužitelj. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, završava se algoritam.

Algoritam SS (engl. *Sporadic Server*) čijim se korištenjem pri raspoređivanju zadataka za svaku vremensku jedinicu provjerava spremnost poslužitelja i treba li se obnoviti kapacitet. Uz kapacitet poslužitelja, prati se i vrijeme trošenja kapaciteta poslužitelja, te količina za koju se potrošio kapacitet, kako bi se znalo kada će se i za koliko obnoviti kapacitet poslužitelja. Ukoliko su poslužitelj i zadatak spremni, zadatak se izvršava, te se prati vrijeme kada je počeo prazniti kapacitet kao i količina za koju se kapacitet ispraznio. Ukoliko zadatak nije spreman, prelazi se na novi zadatak. Ukoliko poslužitelj nije spreman ili nema kapaciteta, prelazi se na sljedeći poslužitelj. Ako više nema poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, završava se algoritam.

2.1.3 Rate Monotonic Scheduling algoritam raspoređivanja

Algoritam RMS (engl. *Rate Monotonic Scheduling*), prema [11], pripada kategoriji algoritama sa zadacima poslužiteljske naravi. Prema slici 2.3, koja prikazuje dijagram podjele algoritama, može se vidjeti da je riječ o raspoređivanju nezavisnih zadataka. Algoritam RMS je metoda dodjeljivanja prednosti skupu zadataka, što istovremeno ovisi i o periodu zadatka. Kraći period zadatka znači višu razinu prednosti skupa zadataka. Prema [11], algoritam raspoređivanja RMS je optimalni predvidivi algoritam za raspoređivanje zadataka sa strogo definiranim vremenskim uvjetima (zahtjevima).
Prema [11], korištenje algoritma RMS postavlja pretpostavke:

1. Ukoliko postoje strogo definirani vremenski uvjeti za zadatke iz skupa $\{T_i\}$ znači da će zahtjevi biti periodni. Skup $\{T_i\}$ je skup periodnih zadataka za izvršavanje.
2. Međusobna neovisnost zadataka, nepostojanost ograničenja u prednosti i nepostojanost međusobnog isključivanja među bilo kojim zadacima ili parovima zadataka.
3. Vrijeme nužnog završetka odgovara periodu, odnosno $T_d = p$.

4. Uvjetovano (zahtijevano) maksimalno vrijeme obrade podataka c_i poznato je prije izvođenja, te je konstantno tijekom izvođenja (engl. *Worst-Case Execution Time*, WCET).
5. Zahtijevano vrijeme za promjenom okoline je zanemarivo.
6. Zbroj korisnosti μ za n zadataka izražava se pomoću formule:

$$\mu = \sum \frac{c_i}{p_i} \leq n(2^{\frac{1}{u-1}}) \tag{2-2}$$

gdje je:

$\{T_i\}$ - skup periodnih zadataka za izvršavanje,

T_d - vrijeme nužnog završetka,

p - periodi za nužni završetak,

c_i - zahtjevano maksimalno vrijeme obrade podataka,

μ – zbroj korisnosti i

n - broj periodnih zahtjeva.

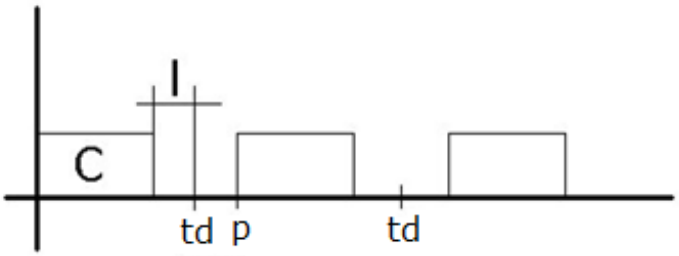
Nakon izračuna korisnosti, može se procijeniti osigurava li algoritam RMS zadovoljavanje strogih vremenskih zahtjeva za n periodnih zadataka. Zahtjevi će se ispuniti ukoliko je korisnost manja od 69%.

Prema [5], svojstva algoritma RMS su:

- Visoka korisnost kod raspoređivanja.
- Prag rasporedivosti za RMS dobar je i za stupanj korisnosti od 88%, a često zadovoljava i za stupanj korisnosti od 100%.
- Stabilnost na prijelaznu preopterećenost.
- Algoritam RMS omogućava dovoljno visoku korisnost tokom obrade aperiodnih zahtjeva. Algoritmi za obradu aperiodnih zahtjeva ne utječu uvelike na rasporedivost RMS-a.
- Dijeljenje sredstava (resursa). Algoritam RMS se primjenjuje i u situacijama međusobnog isključivanja.
- Ukupni trošak raspoređivanja poprilično nizak.

2.1.4 Algoritam raspoređivanja EDF

Prema [2], za raspoređivanje periodnih zadataka se koriste algoritmi raspoređivanja temeljeni na dva različita principa. Jedan od tih algoritama raspoređivanja je EDF (engl. *Earliest Dedline First*). Algoritam EDF optimalan je pri izvođenju, ukoliko za grupu ili skup zadataka postoji raspored izvođenja. Tada će algoritam EDF uspjeti rasporediti zadatke.



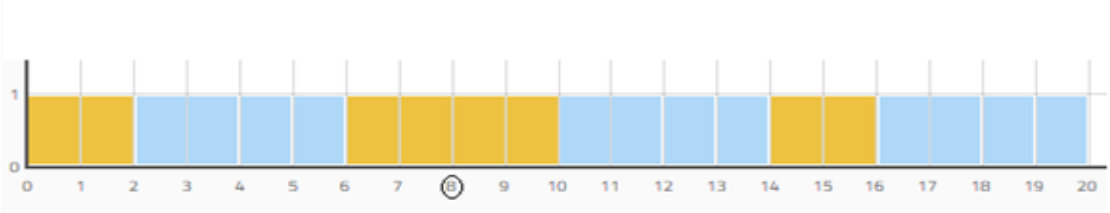
Slika 2.4 *Algoritam EDF*

Algoritam EDF radi na principu izvođenja zadataka s najranijim vremenom završetka, čineći nedeterministički izbor u slučaju jednakih uvjeta. Za isti postupak raspoređivanja na principu izvođenja zadataka s najranijim vremenom završetka nekada je korišten naziv termina DDS (engl. *Deadline Driven Scheduling*). Promjena naziva termina uvjetovana je samo promjenom konteksta.

Svojstva algoritma EDF, prema [1], su:

- Opisan je parametrima T_i, C_i, p_i, t_{di}.
- Algoritam EDF je optimalni, dinamički i predvidivi algoritam za sustave s jednim procesorom koji su temeljeni na dinamičkim razinama prvenstva.
- Korisnost algoritma je vrlo blizu 1, čak i ukoliko periode zadatka nisu cjelobrojni umnošci najmanje periode.
- Nakon značajnog događaja zadatak s najranijim vremenom nužnog završetka dobiva najveću dinamičku razinu prvenstva, odnosno najveću primarnu razinu.

Na slici 2.5. prikazan je primjer raspoređivanja algoritma EDF, gdje je c_i vrijeme trajanja pojedinog zadatka, p_i period pojedinog zadatka, t_{di} vrijeme nužnog završetka zadatka, t je vremenska jedinica u kojoj se zadatak izvodi. Zadaci su označeni s T_1 i T_2.



T_1

T_2

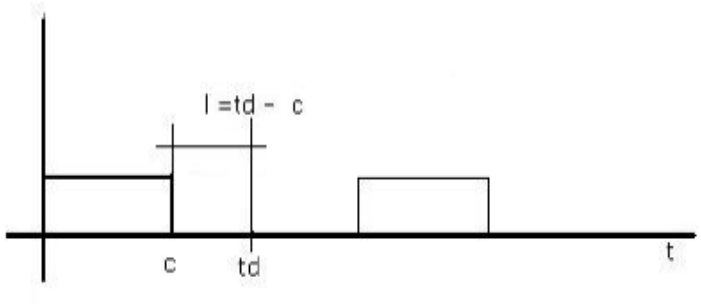
	C_i	p_i	td_i
T_1	2	4	5
T_2	4	6	8

Slika 2.5 *Prikaz raspoređivanja po algoritmu EDF*

Izvođenje T_1 i T_2 zadatka pokrenuto je istovremeno, no kako T_1 ima manje vrijeme nužnog završetka, on u izvođenje kreće prvi. U trenutku t = 4 pojavljuje se ponovno T_1, ali s većim vremenom nužnog završetka od T_2 koji se trenutno izvodi, stoga T_2 nastavlja s izvođenjem. T_1 se izvodi naknadno. U trenutku t = 6 pojavljuje se T_2, te T_1 koji se pojavio u trenutku t = 4 također čeka na izvođenje i ima manje vrijeme nužnog završetka, stoga se izvodi T_1. U trenutku t = 8 pojavljuje se novi T_1 koji također ima manje vrijeme nužnog završetka od T_2 koji čeka na izvođenje, te se izvodi T_1. Algoritam nastavlja s radom po istom principu do kraja.

2.1.5 Algoritam najmanjeg vremena čekanja

Prema [6], algoritam najmanje latencije (engl. *Least Laxity*, LL) je algoritam raspoređivanja zadataka posebno namijenjen za jednoprocesorske sustave za raspodjelu dinamičnih nezavisnih zadataka. Algoritam LL označava algoritam raspoređivanja zadataka, a primarni su oni zadaci s najmanjom latencijom. Zaključno tome je pravilo algoritma, koje kaže, zadatak koji u nekoj točki zaključivanja ima najmanji l (vrijeme čekanja) dobiva najveću razinu dinamičke dinamičkog prvenstva, stoga se dobiva isti nedeterministički izbor u slučaju jednakih uvjeta. Prema [6], dijagram stanja algoritma LL može se vidjeti na slici 2.6, gdje je l oznaka za vrijeme čekanja (*laxity*) što znači slabost zadatka, a t_d je vrijeme nužnog završetka zadatka.



Slika 2.6 *Algoritam LL*

2.1.6 Optimalnost algoritama raspoređivanja zadataka u računalnom sustavu s jednim poslužiteljem

Optimalnost algoritma za raspoređivanje zadataka odnosi se na komparativne prednosti jednih u odnosu na druge unutar iste klase algoritama. Razmatrane klase algoritama sastoje se od klase algoritama za raspoređivanje zadataka sa stalnim prioritetima i od klase svih algoritama za višeprocesorsko raspoređivanje.

Prema [3], algoritam raspoređivanja je optimalan ukoliko se njime može rasporediti bilo koji skup zadataka. Isti skup zadataka se može ujedno rasporediti drugim algoritmom iste klase. Algoritamska optimalnost omogućuje povećanje vjerojatnosti, što daje dodatnu sigurnost za izvodljivost skupa zadataka. Povećanje vjerojatnosti prikazuje proširenje opsega mogućih slučajeva do kojih algoritam može garantirati izvođenje novih zadataka unutar njihovih vremenskih ograničenja, odnosno vremenskih razdoblja, uz istovremeno ispunjavanje obveza prema postojećim zadacima.

Vjerojatnost za sigurno izvođenje zadataka postoji ukoliko se uzmu u obzir pretpostavke koje se odnose na najlošiji slučaj trajanja vremena odziva i mogućih pojava nepredvidivih pogrešaka. Prema [5], ukoliko postoji slučaj jednoprocesorskog raspoređivanja, oba algoritma EDF i LL su optimalni. Ova se tvrdnja može prihvatiti kao teorem i njen dokaz proizlazi iz činjenice da se raspored zadataka ostvaren nekim trećim postupkom uvijek može prevesti ili u EDF ili u LL kroz odgovarajuću zamjenu pojedinih parova zadataka, što može zahtijevati vremenski prekid tokom izvođenja zadataka, ukoliko je dekompozicija zadataka dozvoljena.

2.2 Raspoređivanje zadataka u sustavima s više poslužitelja

Arhitektura s više slojeva sadrži i višestruke (specijalizirane i/ili redundantne) aplikacijske poslužitelje i/ili baze podataka. Kroz ovaj rad su opisani računalni sustavi s dva ili više procesora,

jednako kao i u eksperimentalnom dijelu rada. Glavne karakteristike ovakvih sustava su postojanje vremenske baze (služi za potpuno raspoređivanje događaja i zadataka) i memorije (služi za implementaciju komunikacijskih vektora između zadataka). Sustavi s više poslužitelja, prema [5], imaju mogućnost uvida u stanje sustava, te mogućnost pristupanja bilo kojem poslužitelju unutar sustava u svakom trenutku.

Prethodno spomenuta memorija sadrži cijeli kod i podatke koji se dijele između različitih zadataka, dok procesor može imati lokalnu memoriju (stog, predmemoriju, itd). Prema [5], isti sustavi predstavljaju potpunu analogiju sustavima s jednim procesorom, prvenstveno kapacitetom za mogućnosti paralelnih izvršavanja zadataka. U višeprocesorskom okruženju, algoritmi raspoređivanja zadataka se koriste ukoliko je moguće zadovoljiti rokove nužnog završetka za sve zadatke.

Definicija računalnih sustava s više poslužitelja, prema [5], ima sličnost definiciji s jednim poslužiteljem, ali je ujedno i proširena sa dva popratna uvjeta:

1. Procesor može izvršavati samo jedan zadatak u određenom vremenu.
2. Zadatak može biti obrađen isključivo od strane samo jednog procesora.

Prema [5], okruženje ovih sustava je ograničeno arhitekturom koju tvore dva identična procesora (jednaka brzina obrade) s mrežnim prekidivim raspoređivanjem.

2.2.1 Višeprocesorsko statičko raspoređivanje zadataka

Postupak višeprocesorskog statičkog raspoređivanja zadataka izvodi se prema vremenskom redoslijedu izvođenja zadataka. Redoslijed izvođenja zadataka prilagođen je odnosu međusobnih prioriteta zadataka, odnosno međusobnim relacijama prvenstva. Međusobni prioriteti zadataka ujedno definiraju i krajnja vremena završetka izvođenja zadataka. Postupak je primjenjiv za zadatke s pravima prvenstva koji se mogu prikazati s acikličkim grafom, te za zadatke koji tvore utemeljenu strukturu stabla.

2.2.2 Višeprocesorsko dinamičko raspoređivanje zadataka

Postupci dinamičkog raspoređivanja zadataka ostvaruju svoju učinkovitost oslanjanjem na relativno jednostavne kriterije, koji su vezani i na neke od parametara zadataka, kao što su vrijeme nužnog završetka ili hitnost izvođenja zadatka. Prema [5], cilj postupaka dinamičkog raspoređivanja zadataka je izbjegavanje računalno skupe optimizacije rasporeda, tako da sustav može brzo odgovoriti na promjene u okolini. Brzom odgovaranju na promjene u okolini uvelike

može pripomoći i razumijevanje ponašanja okoline, te uvid u strategiju planiranja i potpuno prilagođavanje svim ograničenjima.

2.2.3 Optimalnost pri višeprocesorskom dinamičkom raspoređivanju zadataka

Prema [3], postoji način za utvrđivanje optimalnosti algoritama EDF i LL u višeprocesorskom raspoređivanju. Može se postaviti pitanje postoji li uopće optimalni algoritam koji može rasporediti skup zadataka samo s djelomičnim poznavanjem njihovih parametara? Odgovor na ovo pitanje, prema [5], je da ne postoji takav optimalni algoritam. Isti algoritam može se razraditi kroz sustavno ispitivanje mogućih slučajeva s djelomičnim poznavanjem značajki zadataka.

Prema [11], najveća poteškoća u višeprocesorskom raspoređivanju (ukoliko se koristi algoritam raspoređivanja LL i EDF) je potreba za radom iznad njihovih mogućnosti zbog konstantnih novih potreba raspoređivača zadataka. Potrebe mogu biti i ukupno vrijeme potrebno za izradu rasporeda izvođenja koje uključuje i vrijeme čekanja zadatka u redu, stalna promjena konteksta odrade zadatka od čekanja u redu do trenutka njegovog izvođenja i kašnjenje odrade zadatka zbog gubitaka koji su nastali u predmemoriji. Prema [11], kako bi se značajno umanjio utjecaj poteškoća na izvođenje zadataka, uzevši u obzir promjene konteksta i gubitaka u predmemoriji, iznimno je važno da algoritam odobri izvođenje zadatka na istom procesoru od početka do kraja.

3. RASPOREĐIVANJE ZADATAKA NA POSLUŽITELJU PUTEM APLIKACIJE

Programsko rješenje u ovom radu je aplikacija napravljena u svrhu simulacije raspoređivanja zadataka na jednom ili više poslužitelja. Aplikacija nudi mogućnosti odabira algoritama raspoređivanja, trajanja simulacije te broj i karakteristike poslužitelja (vrijeme obnavljanja i kapacitet).

3.1 Način rada aplikacije

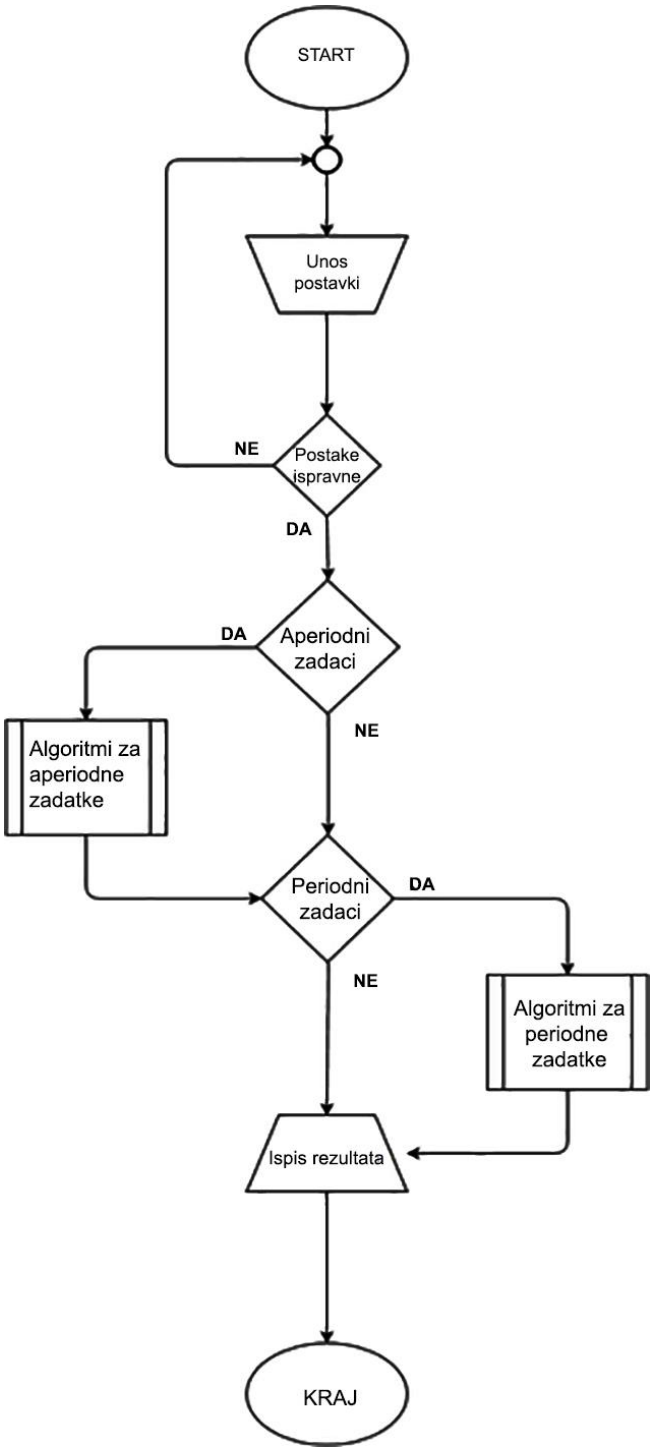
Aplikacija je izrađena u PHP (engl. *Hypertext Preprocessor*) i JavaScript programskim jezicima koristeći *jQuery* uz *Flot* dodatak za prikaz grafova u web pregledniku. Bazirana je na objektno orijentiranom PHP radnom okviru MVC (engl. *Model–View–Controller*) arhitekture. MVC je oblik arhitekture softverske aplikacije zasnovan na odvajanju pojedinih dijelova aplikacije u komponente ovisno o njihovoj namjeni.

Radi boljeg shvaćanja oblika arhitekture aplikacije potrebno je znati i sastavne dijelove MVC-a:

- Model (M) (engl. *model*) - podaci i programska logika (engl. *business logic*)
- Pogled (V) (engl. *view*) - prikaz podataka (obrazac, tablica, graf, itd.)
- Upravitelj (C) (engl. *controller*) - prihvaća ulazne naputke i pretvara ih u naloge modelu i pogledu

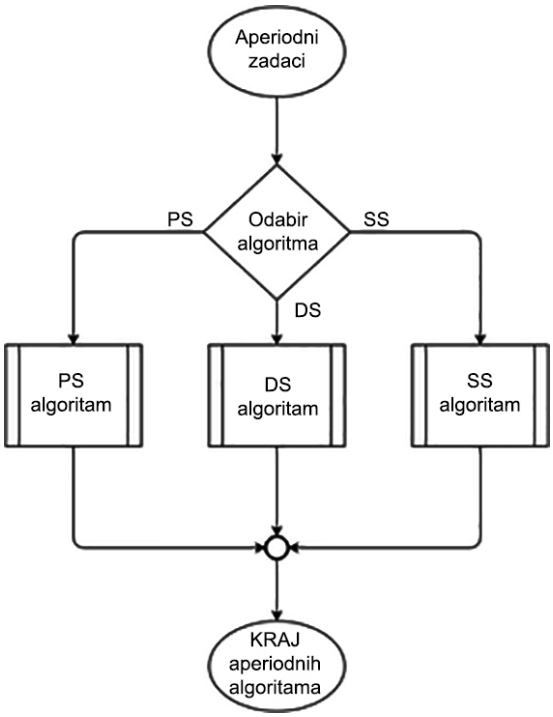
U aplikaciju su ugrađeni sljedeći algoritmi raspoređivanja: za raspoređivanje periodnih zadataka ugrađeni su algoritmi RMS (*Rate-Monotonic Scheduling*), EDF (*Earliest Deadline First*) i LL (*Least Laxity*) sa i bez značajnog događaja, a za raspoređivanje aperiodnih zadataka algoritmi PS (*Polling Server*), DS (*Deferrable Server*) i SS (*Sporadic Server*).

Aplikacija se sastoji od modela zadataka i modela algoritama raspoređivanja (Sl. 3.1). Modeli zadataka sadrže metode za učitavanje zadataka u aplikaciju i pohranjivanje u XML (engl. *Extensible Markup Language*) formatu, a modeli algoritama njihovu logiku raspoređivanja.



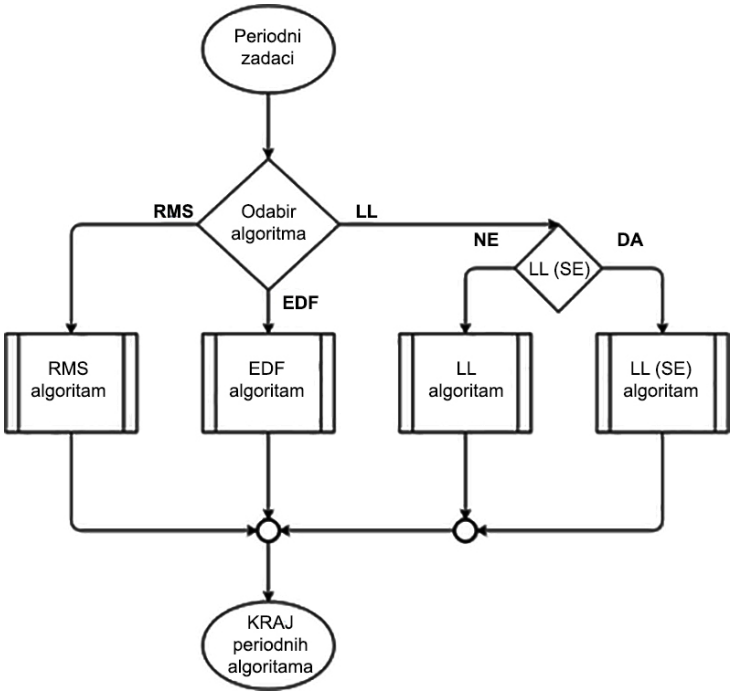
Slika 3.1 *Općeniti algoritam programskog rješenja*

Nakon pokretanja aplikacije, unose se svi parametri simulacije, a zatim se provjerava ima li danih zadataka. Ukoliko ima, model raspoređivanja izvršava odabrani algoritam, kao što je prikazano na slici 3.2. Ako nema aperiodnih zadataka ili su raspoređeni, prelazi se na izvršavanje periodnog algoritma, što se može vidjeti na slici 3.3. Nakon izvršavanja svih algoritama, ispisuju se rezultati.



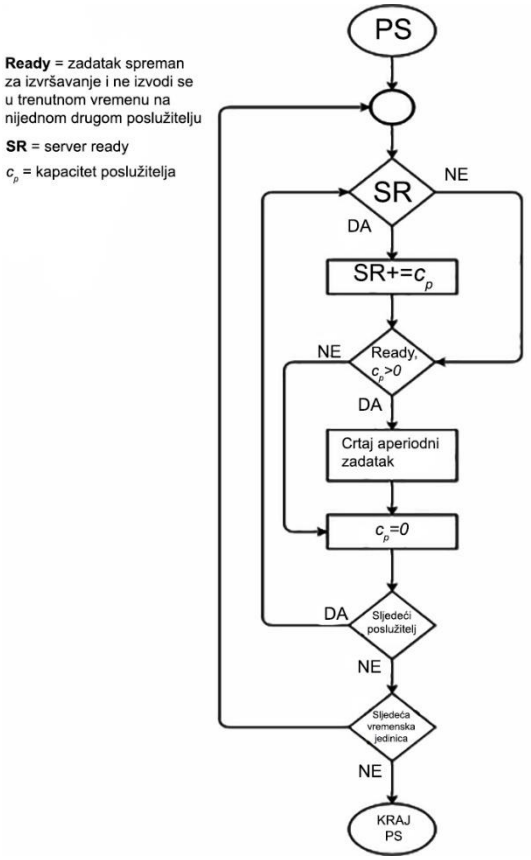
Slika 3.2 *Algoritam u slučaju aperiodnih zadataka*

Prilikom raspoređivanja aperiodih zadataka učitavaju se postavke poslužitelja. Nakon što su postavke poslužitelja učitane, poziva se metoda za raspoređivanje zadataka. Metoda raspoređivanja zadataka pojavljuje se u jednom od modela algoritama za raspoređivanje zadataka.



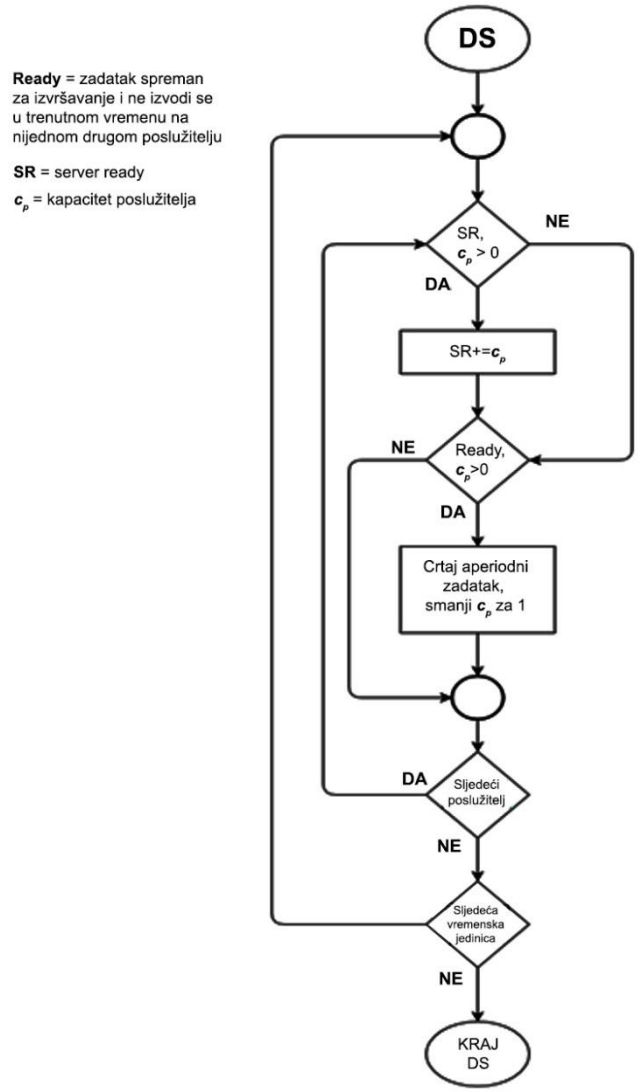
Slika 3.3 *Algoritam u slučaju periodnih zadataka*

U slučaju algoritma PS (engl. *Polling Server*, PS) (Sl. 3.4) za svaku vremensku jedinicu provjerava se spremnost poslužitelja. Ukoliko je poslužitelj spreman, provjerava se spremnost zadatka i izvodi li se zadatak na nekom drugom poslužitelju. Zadatak se upisuje ako je uvjet zadovoljen. Ukoliko se ustanovi da zadatak nije spreman ili je u fazi izvođenja prelazi se na sljedeći spremni zadatak. U slučaju da poslužitelj nije spreman, prelazi se na sljedeći poslužitelj. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, simulacija se završava.



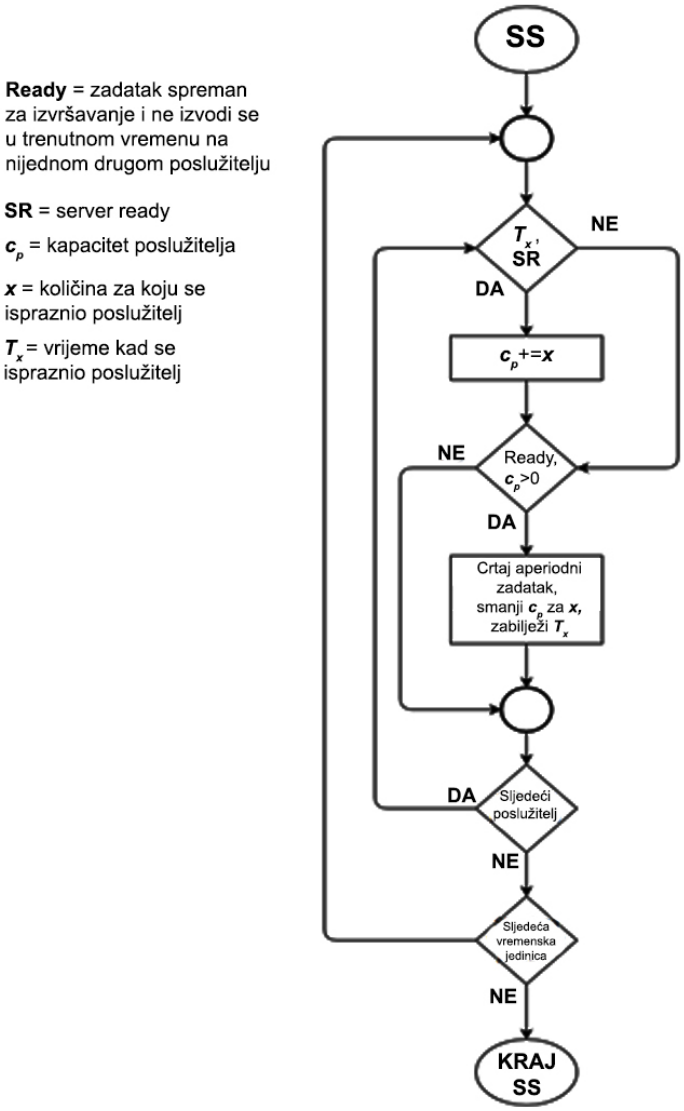
Slika 3.4 *Algoritam PS*

Algoritam DS (Sl. 3.5) za razliku od algoritma PS, za svaku vremensku jedinicu uz provjeru spremnosti poslužitelja provjerava ima li raspoloživog kapaciteta, to znači da je potrebno čuvati preostali kapacitet poslužitelja. Ukoliko je poslužitelj spreman i ima kapaciteta, provjerava se spremnost zadatka i izvodi li se zadatak na nekom drugom poslužitelju. Ako je uvjet zadovoljen, zadatak se upisuje. Ako zadatak nije spreman ili je u fazi izvođenja, prelazi se na sljedeći spremni zadatak. U slučaju da poslužitelj nije spreman ili nema kapaciteta, prelazi se na sljedeći poslužitelj. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, završava se simulacija.



Slika 3.5 *Algoritam DS*

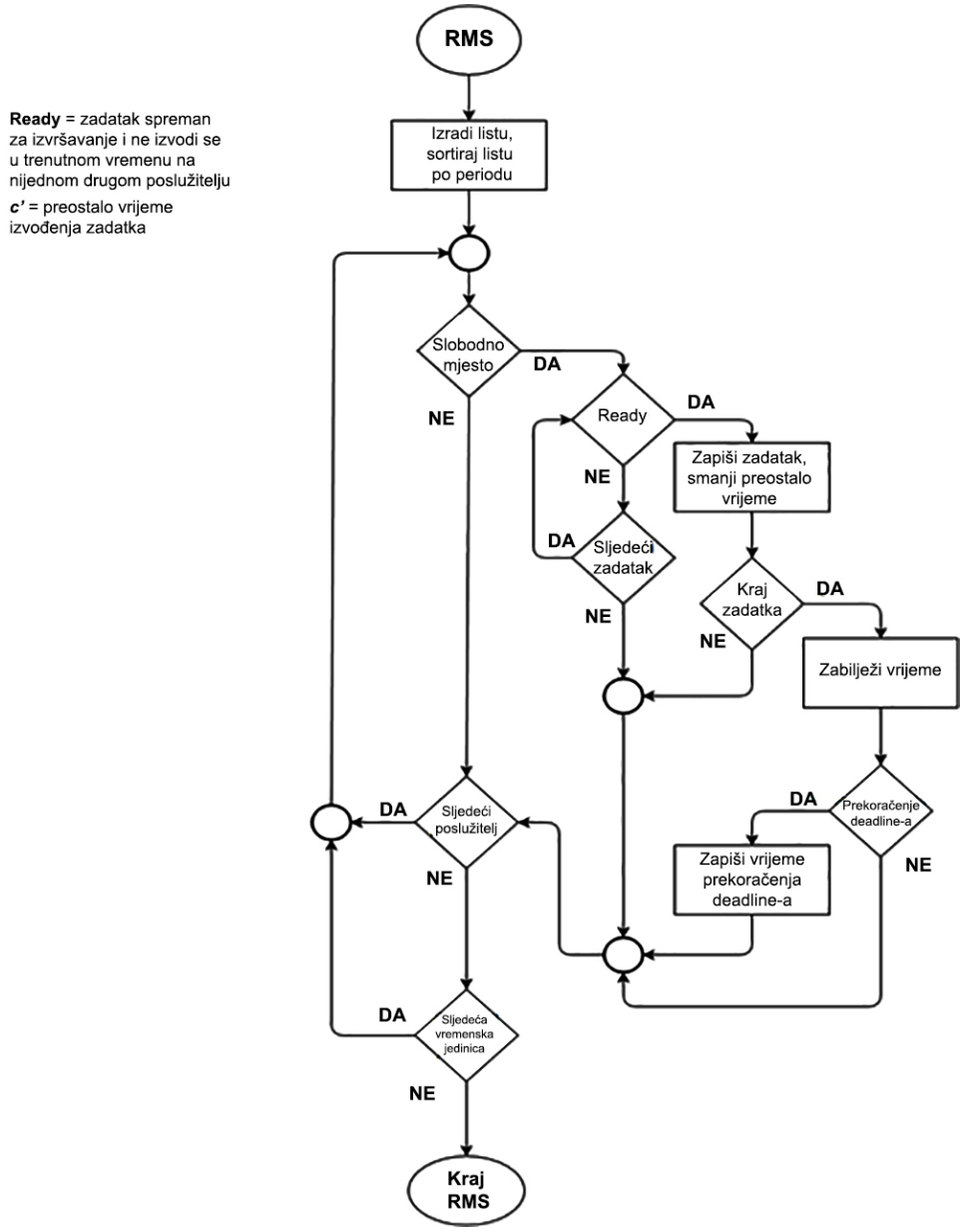
Na slici 3.6 prikazan je način rada algoritma SS kod kojeg se za svaku vremensku jedinicu provjerava spremnost poslužitelja i treba li se obnoviti kapacitet. Uz kapacitet poslužitelja prati se i vrijeme trošenja kapaciteta poslužitelja, te količina za koju se potrošio kapacitet, kako bi se znalo kada će se i za koliko obnoviti kapacitet poslužitelja. Ukoliko su poslužitelj i zadatak spremni, zadatak se izvršava, te se prati vrijeme kada je počeo prazniti kapacitet kao i količina za koju se kapacitet ispraznio. Ukoliko zadatak nije spreman, prelazi se na novi zadatak. Ukoliko poslužitelj nije spreman ili nema kapaciteta, prelazi se na sljedeći poslužitelj. Ako više nema poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, završava se simulacija.



Slika 3.6 *Algoritam SS*

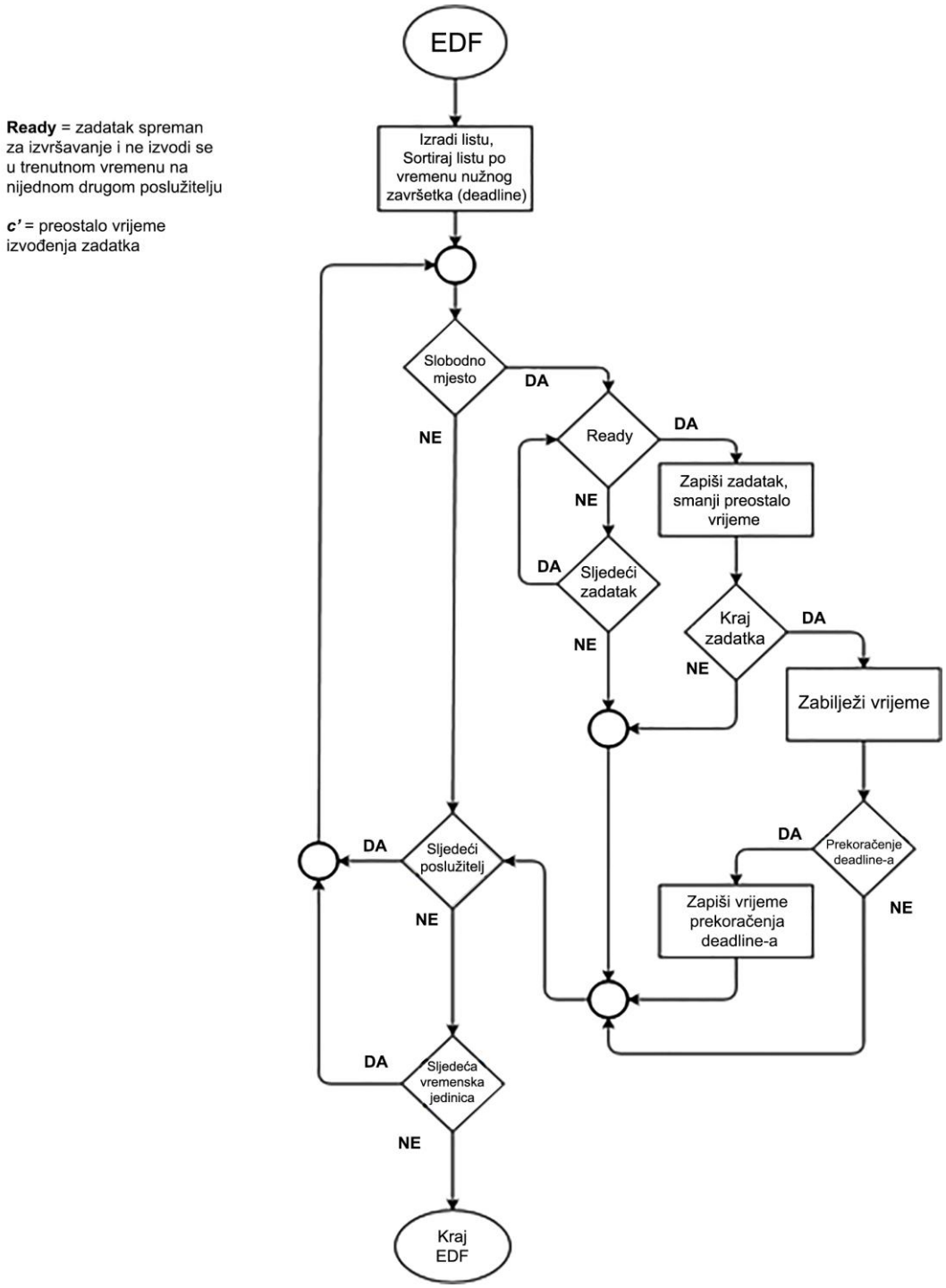
Slika 3.7 prikazuje kako algoritam RMS izrađuje listu periodnih zadataka i sortira ih uzlazno prema periodu. Provjerava se postoji li slobodno mjesto u trenutnoj vremenskoj jedinici. Ako mjesto nije slobodno, prelazi se na sljedeći poslužitelj. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko postoji slobodno mjesto, provjerava se da li je zadatak spreman za izvršavanje i da se ne izvodi se na nekom drugom poslužitelju u trenutnoj vremenskoj jedinici. Ako uvjet nije zadovoljen, prelazi se na sljedeći zadatak, a ako postoje uvjeti za izvođenje zadatka, zadatak se upisuje na vremensku crtu i smanjuje se preostalo vrijeme izvođenja zadatka za jednu vremensku jedinicu. Nakon toga se provjerava je li to kraj zadatka, te ukoliko nije, prelazi se na sljedeći poslužitelj, odnosno na sljedeću vremensku jedinicu ovisno ima li još spremnih zadataka i slobodnih poslužitelja. Ukoliko je zadatak završio, bilježi se vrijeme završetka te se ispisuje prekoračenje roka izvođenja. Bilježi se ako je zadatak prekoračio vrijeme nužnog

završetka. Nakon toga se prelazi na sljedeći poslužitelj, odnosno vremensku jedinicu ukoliko nije kraj simulacije.



Slika 3.7 *Algoritam RMS*

Kod algoritma EDF (Sl. 3.8) izrađena lista se sortira prema vremenu nužnog završetka izvođenja od najmanjeg roka prema najvećem. Ostatak izvođenja algoritma raspoređivanja EDF je jednak kao kod algoritma RMS.

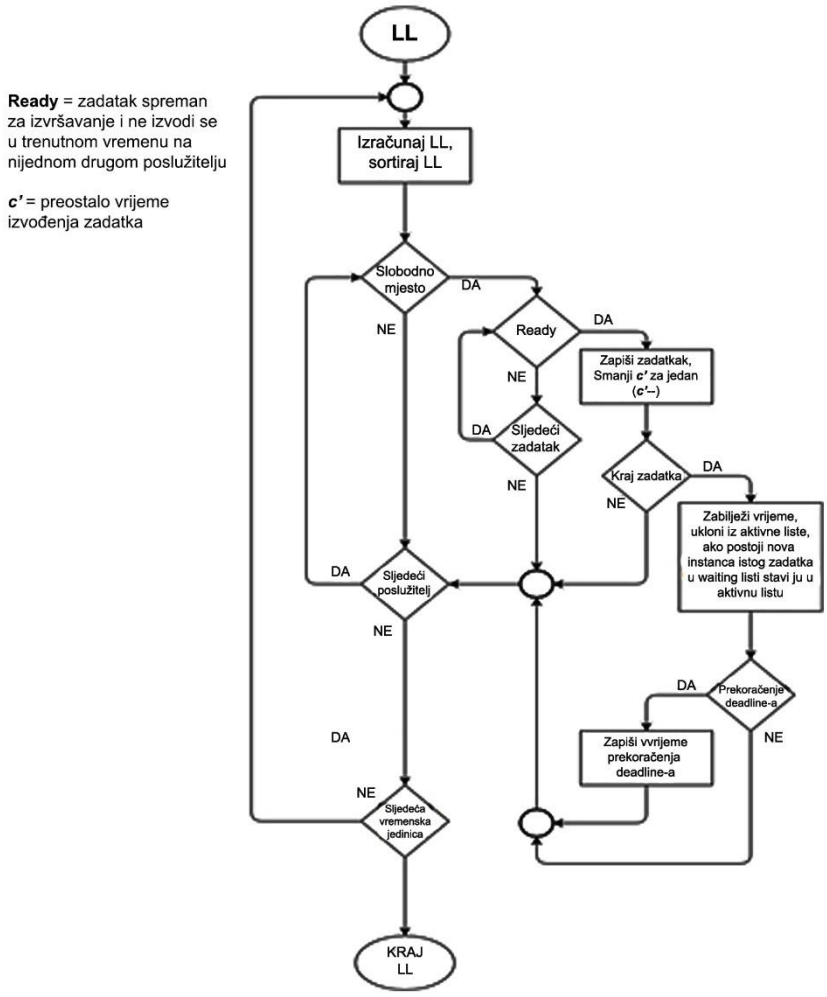


Slika 3.8 *Algoritam EDF*

Na slici 3.9 prikazan je način rada algoritma LL gdje se za svaki pojedini zadatak računa latencija (*l*) i pravi aktivna lista zadataka uzlazno sortirana po *l* u svakoj vremenskoj jedinici. Zatim se provjerava slobodno mjesto na poslužitelju, te ukoliko ga nema, prelazi se na sljedeći poslužitelj. Ako više nema poslužitelja, prelazi se na sljedeću vremensku jedinicu te se ponovno računa latencija i sortira aktivna lista. Ukoliko slobodno mjesto postoji, provjerava se spremnost zadatka. Ako zadatak nije spreman, prelazi se na sljedeći zadatak. Ukoliko ima mjesta, zadatak je spreman

i ne izvršava se na nekom drugom poslužitelju, zadatak se zapisuje, smanjuje se preostalo vrijeme izvođenja zadatka za 1 i provjerava se da li je zadatak završen. Ako nije, prelazi se na sljedeći poslužitelj, odnosno sljedeću vremensku jedinicu ukoliko nema više poslužitelja. Ako je zadatak završio, bilježi se vrijeme završetka, zadatak se uklanja iz aktivne liste i ako postoji instanca tog zadataka u listi čekanja, ta se instanca premješta u aktivnu listu.

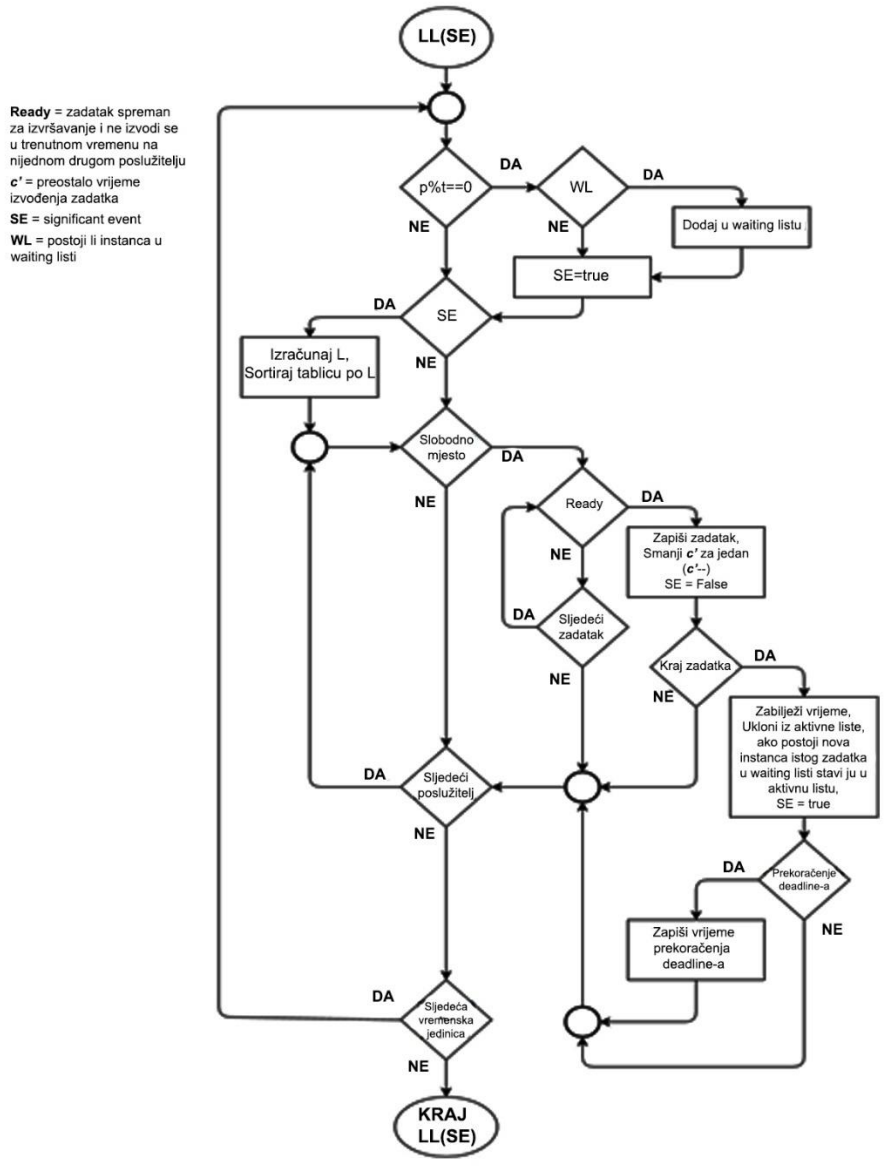
Nakon toga se provjerava prekoračenje roka završetka. Ako je bilo prekoračenja, ono se bilježi. Ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. Ukoliko nema više vremenskih jedinica, završava se simulacija.



Slika 3.9 *Algoritam LL*

Za razliku od običnog algoritma LL, algoritam LL(SE) (Sl. 3.10) baziran na značajnim događajima ne računa latenciju u svakoj vremenskoj jedinici. Algoritam LL(SE) računa latenciju samo ako se dogodio značajni događaj, odnosno završetak nekog od zadataka ili pojava novog zadatka. Proces započinje provjerom dogodi li se značajni događaj u trenutnoj vremenskoj jedinici. Ukoliko se

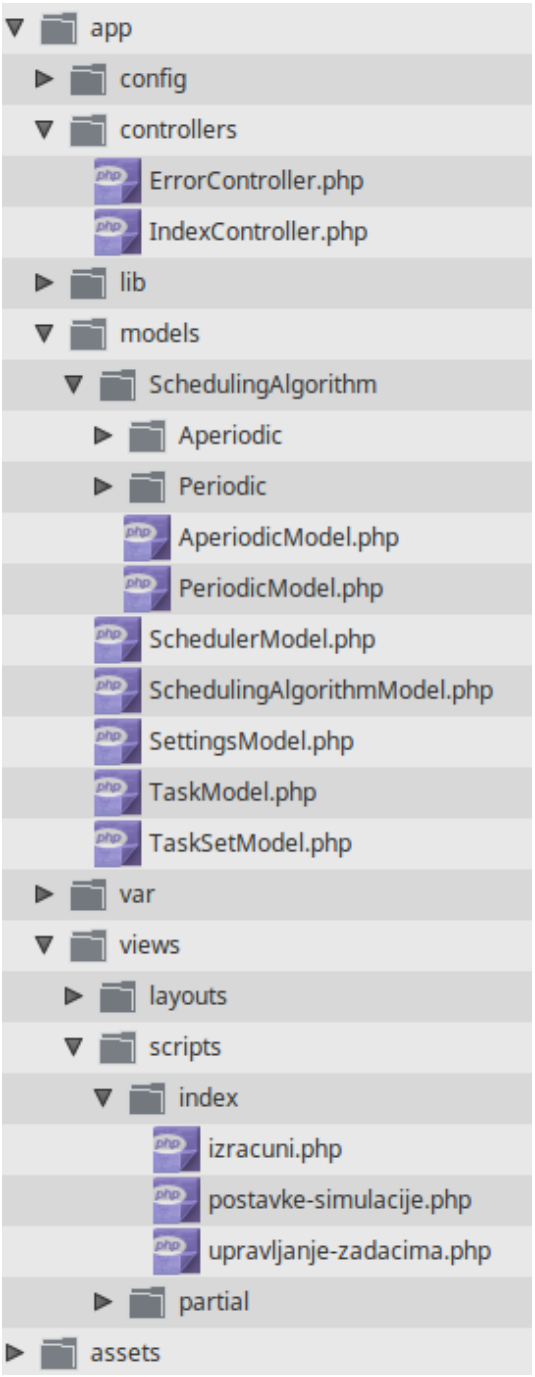
značajni događaj ostvario računa se latencija i sortira lista, ako se značajni događaj nije dogodio lista ostaje sortirana onako kako je prvotno bilo. Zatim se kao i kod običnog algoritma LL provjerava ima li slobodno mjesto, ako nema prelazi se na sljedeći poslužitelj, odnosno sljedeću vremensku jedinicu. Postoji li slobodno mjesto i spreman zadatak on se upisuje i smanjuje se preostalo vrijeme izvođenja zadatka za 1. Nakon toga, provjerava se da li je to završetak zadatka i ukoliko jeste, zapisuje se i obilježava kao značajni događaj. Zadatak se uklanja iz aktivne liste. Postoji li instanca tog zadataka u listi čekanja, ta se instanca premješta u aktivnu listu. Idući korak jest provjera prekoračenja roka završetka. Ukoliko je bilo prekoračenja, bilježi se, ukoliko nema više poslužitelja, prelazi se na sljedeću vremensku jedinicu. U slučaju da nema više vremenskih jedinica simulacija se završava.



Slika 3.10 *Algoritam LL(SE)*

3.2 Prikaz i način uporabe aplikacije

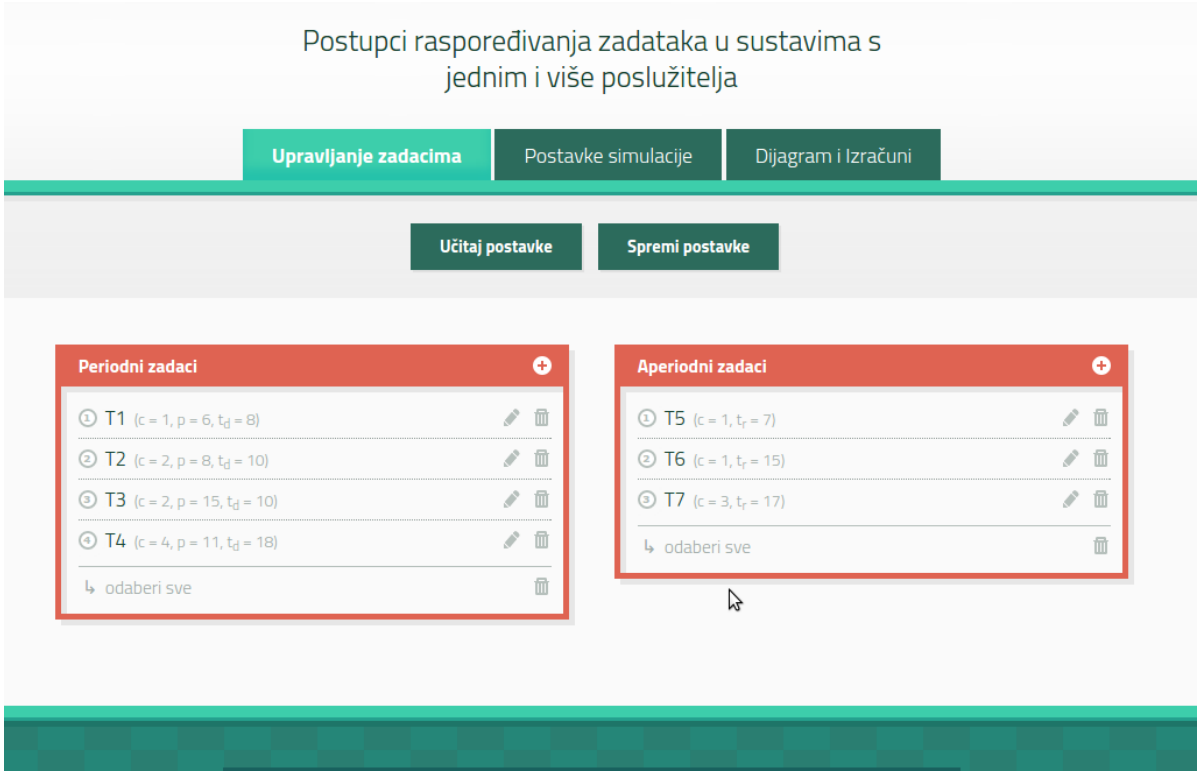
Na slici 3.11 prikazana je struktura aplikacije. *Pogled* (engl. *View*) dio aplikacije su skripte ispisa obrazaca za unos i odabir podataka, te ispis rezultata raspoređivanja i grafova. Upraviteljski dio aplikacije sadrži akcije za povezivanje logike i prikaza.



Slika 3.11 *Struktura aplikacije*

Aplikacija se sastoji od tri komponente: *Upravljanje zadacima*, *Postavke simulacije*, te *Dijagrami i Izračuni*.

U dijelu *Upravljanje zadacima* zadaci se mogu dodavati, uklanjati, uređivati i pohranjivati, te učitavati skupovi zadataka (Sl. 3.12).



Slika 3.12 *Upravljanje zadacima*

Pogled se sastoji od liste periodnih i aperiodnih zadataka, te obrasca za dodavanje zadataka. Zadaci s liste se mogu uređivati i uklanjati preko modela zadataka pozivom odgovarajućih metoda u akciji *upravljanjeZadacimaAction* glavnog upravitelja.

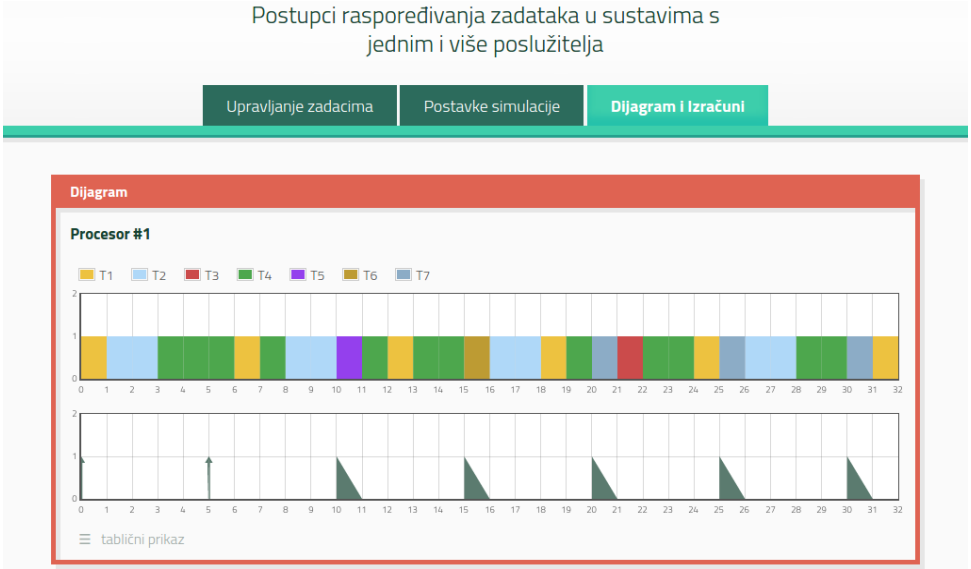
Dodavanje zadataka prikazano je slikom 3.13. Zadaci se dodaju preko obrasca akcijom *saveTaskAction* glavnog upravitelja. Akcija zatim poziva metodu za provjeru i metodu za spremanje zadataka iz modela zadataka koja preko *save* metode *TaskSet* modela zadatke sprema u kolačiće (engl. *cookies*) (spremanje unešenih podataka u preglednik, koji se šalju prema poslužitelju prilikom svakog ponovnog spajanja na internet stranicu).

Slika 3.13 *Dodavanje zadataka*

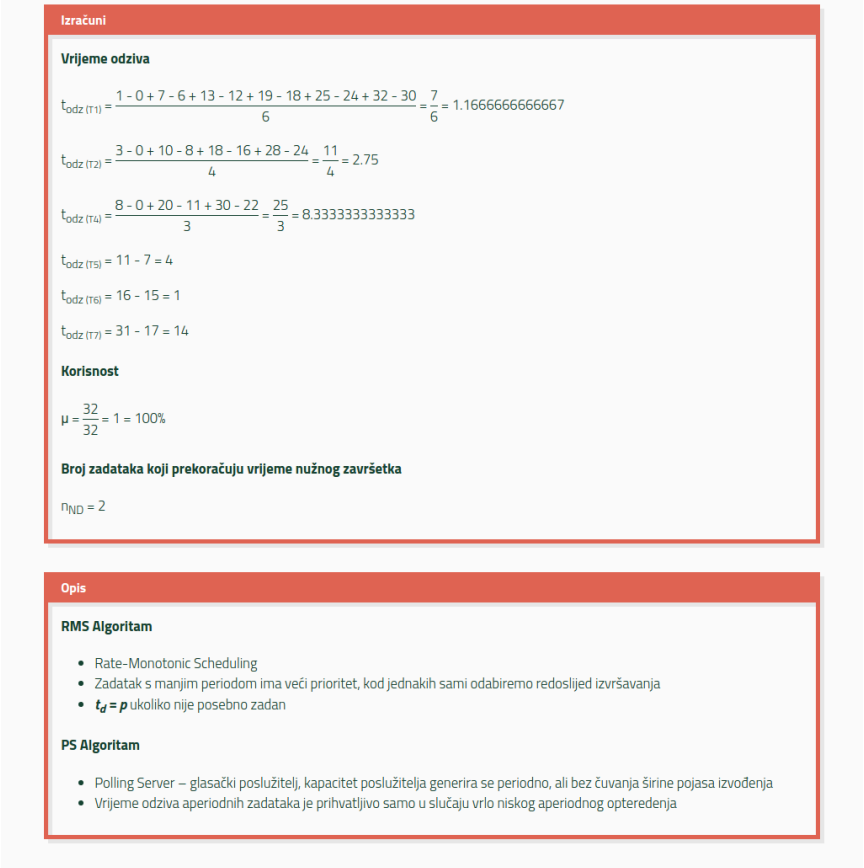
Preko izbornika za dodavanje zadataka (Slika 3.13) pohranjuju se parametri zadataka na datotečni sustav (engl. *file system*) u XML formatu. Isti parametri iz datoteke se po potrebi mogu naknadno učitati u aplikaciju. U postavkama simulacije (Sl. 3.14) odabiru se zadaci, algoritmi i određuju parametri poslužitelja, te trajanje simulacije. U lijevom dijelu stranice nalaze opisi svih algoritama. Pokretanjem simulacije preko upravitelja spremaju se postavke koristeći *model postavke* (engl. *Settings*) i pokreće izvršavanje raspoređivanja uz zadane parametre koristeći model raspoređivanja (engl. *Scheduler*). *Model raspoređivanja* izvršava raspoređivanje pozivom metoda za izvršavanje modela odabranih algoritama. Nakon izvršavanja algoritama, upravitelj preusmjerava na izbornik *Dijagrami* i *Izračuni*.

Slika 3.14 *Postavke simulacije*

Akcija za izračune u glavnom upravitelju preuzima i prikazuje rezultate raspoređivanja koji se mogu vidjeti na slici 3.15. Pomoću parametara izračunatih simulacijom, iscrtava se dijagram uz pomoć *Flot jQuery* dodatka. Ispod dijagrama nalaze se tablični prikaz rezultata, izračuni i opis korištenih algoritama raspoređivanja kao što je prikazano na slici 3.16, u nastavku potpoglavlja.



Slika 3.15 *Dijagram raspoređivanja zadataka na procesoru*



Slika 3.16 *Izračun vrijednosti parametara algoritma raspoređivanja*

Rezultati raspoređivanja prikazuju se grafom i tablično, te se vrši analiza na osnovu izračunatih parametara. Parametri koji se računaju su:

- vrijeme odziva - vrijeme proteklo od pojavljivanja zadatka do njegovog završetka
- korisnost (μ) - postotak iskorištenosti procesora
- broj zadataka koji prekoračuju vrijeme nužnog završetka

4. ANALIZA ALGORITAMA RASPOREĐIVANJA

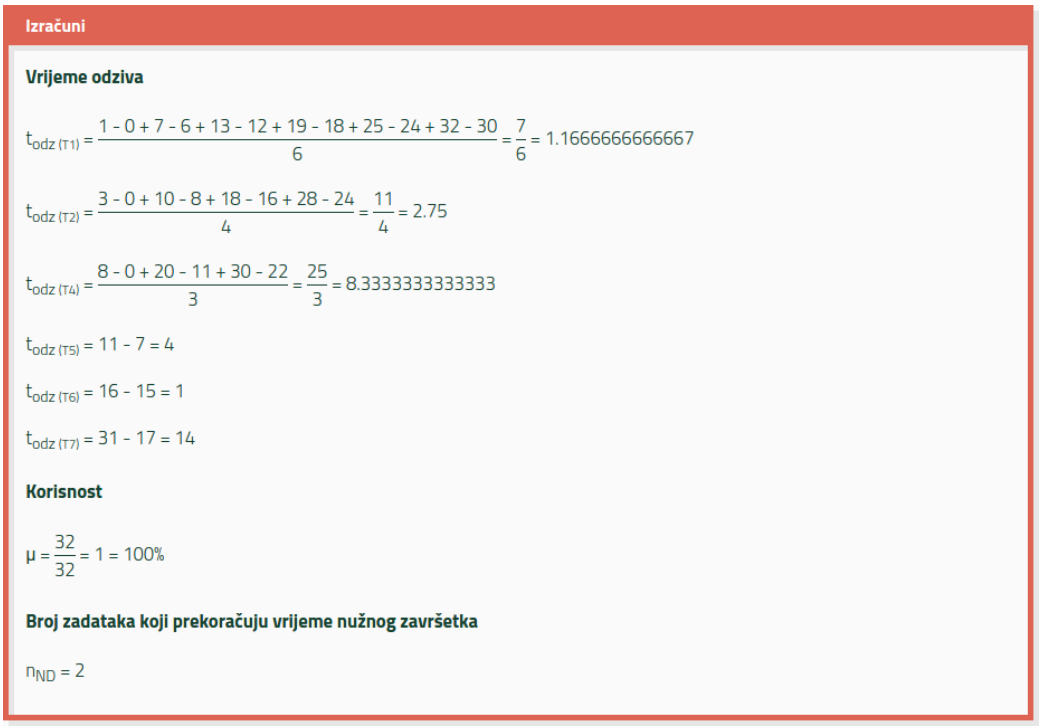
Eksperimentalnom analizom želi se otkriti ponašanje i priroda algoritama raspoređivanja, te ustanoviti koji algoritam je pogodan kojim zadacima tj. koji će algoritam dati bolje rezultate za određeni skup zadataka. Nije isto postoji li puno kraćih ili dužih zadataka. Odabir algoritma raspoređivanja ovisi i o vremenima nužnih završetaka, te o broju aperiodnih zadataka. Testiranja su izvršena na različitim skupovima zadataka. Prikazan je utjecaj na rezultat raspoređivanja obzirom na promjenu načina raspoređivanja i broj poslužitelja.

U svrhu određivanja najboljeg algoritma, praćeni su sljedeći pokazatelji. Prvi pokazatelj je grafički prikaz raspoređivanja pojedinih zadataka na vremenskoj crti. Pokazuje ponašanje algoritama, tj. kada će koji zadatak i na kojem poslužitelju biti izvršen. Najvažniji pokazatelj je broj zadataka koji prekoračuju vrijeme nužnog završetka.

Vrijeme prekoračenja nužnog završetka je vrijeme do kojeg bi se zadatak trebao izvršiti. Ukoliko se zadatak ne izvrši, to može imati velike posljedice za rad sustava, ovisno o vrsti sustava. Algoritam je bolji što je manji broj prekoračenja vremena nužnog završetka.

Sljedeći pokazatelj je vrijeme odziva zadatka. Vrijeme odziva je prosječno vrijeme proteklo od pojave do završetka zadatka. Poželjno je da vrijeme odziva bude što manje, jer je sustav s manjim odzivom brži pa će biti i manje prekoračenja vremena nužnog završetka. Zadatak s dužim vremenom izvođenja imati će veće vrijeme odziva. Zato su u radu korišteni isti zadaci pri testiranju različitih algoritama.

Određuje se još i stupanj korisnosti (μ). To je broj vremenskih jedinica kada se zadaci izvode u odnosu na broj vremenskih jedinica trajanja simulacije. Manji stupanj korisnosti znači da je poslužitelj manje iskorišten te se vjerojatno može i jače opteretiti. Primjer izračuna navedenih pokazatelja prikazan je na slici 4.1.



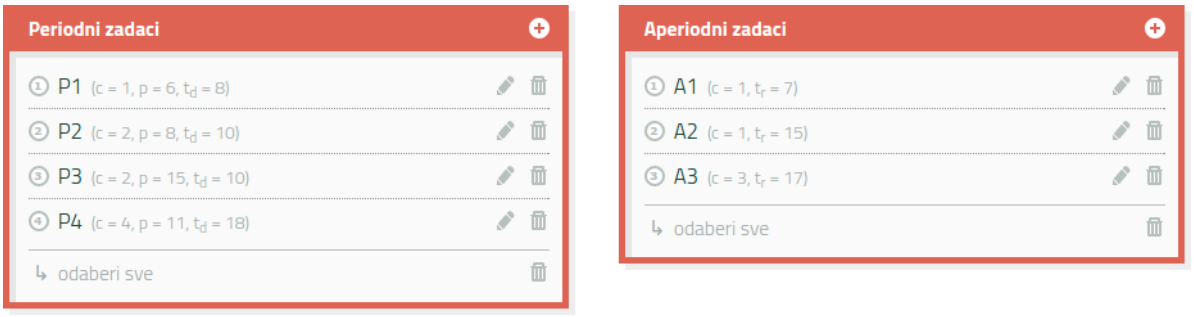
Slika 4.1 *Izračuni pokazatelja uspješnosti raspoređivanja*

Prvi skup zadataka je skup koji malo opterećuje sustav. Očekivano je da nema prekoračenja vremena nužnog završetka i da je iskorištenost poslužitelja mala. Skup sadrži manji broj zadataka manjeg kapaciteta i perioda pojavljivanja. Korišten je još skup zadataka koji maksimalno opterećuje sustav. Karakteristike tog skupa su velik broj zadataka što rezultira većim stupnjem korisnosti, ali postoji puno veća vjerojatnost da zadatak prekorači vrijeme nužnog završetka. Zatim je korišten skup koji se po opterećenju sustava nalazi između prethodna dva navedena skupa. Odabrani su ovi skupovi zadataka kako bi se utvrdilo ponašanje sustava kad je pod malim opterećenjem, te kad pod velikim opterećenjem što dovodi do opasnosti prekoračenja vremena nužnog završetka. Skup zadataka sa srednjim opterećenjem također je odabran za testiranje rada sustava. Uz navedena tri skupa napravljeno je i testiranje na zadacima iz kolegija Računalni sustavi stvarnog vremena sa ispitnog roka održanog 28.06.2016. godine na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.

4.1. Rezultati i analiza raspoređivanja aperiodnih zadataka u sustavu pri srednjem opterećenju

Prvim testom je dan uvid u utjecaj odabira algoritama za raspoređivanje aperiodnih zadataka na ukupan rezultat raspoređivanja. Testiranje je provođeno na zadacima sa slike 4.2, a isti zadaci

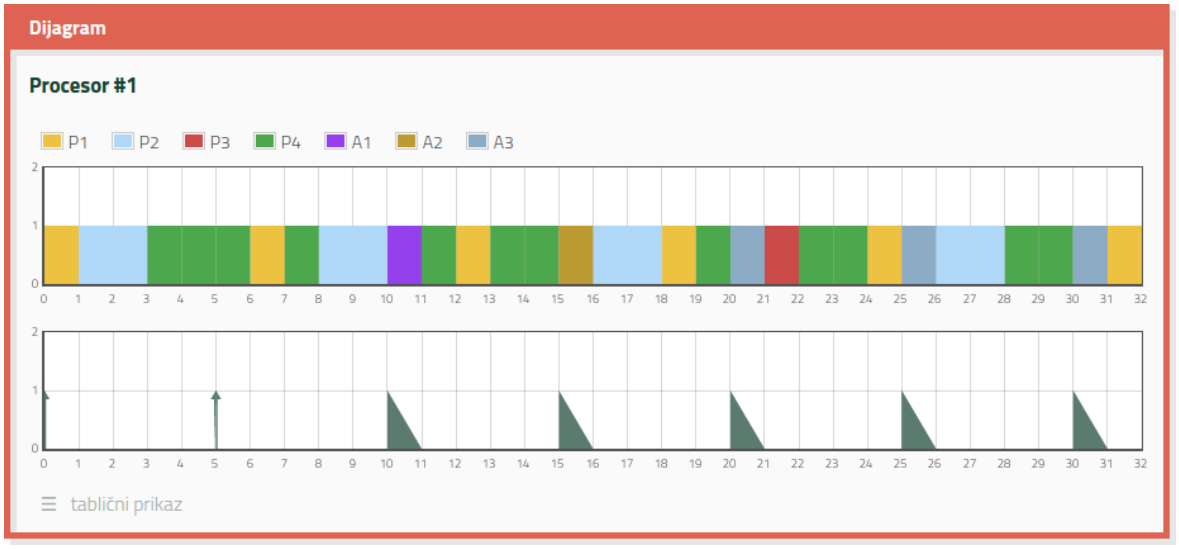
predstavljaju sustav u normalnom radu kada opterećenje nije niti preveliko niti premalo. Provedene su simulacije na 1, 2 i 4 poslužitelja karakteristika kapaciteta c_S = 1 i perioda obnavljanja p_S = 5 u 32 vremenske jedinice.



Slika 4.2 *Zadaci na poslužiteljima*

Budući da su dobiveni slični rezultati bez obzira na broj poslužitelja, prikazani su rezultati za jednoprocesorski sustav za sve algoritme raspoređivanja, dok su rezultati ostalih testiranja na digitalnom zapisu dani kao prilog.

Na slikama 4.3, 4.4 i 4.5 prikazani su dijagrami rezultata raspoređivanja korištenjem algoritma RMS u kombinaciji s algoritmima za raspoređivanje aperiodnih zadataka.



Slika 4.3 *Rezultati raspoređivanja zadataka prikazanih slikom 4.2 na jednoprocesorskom sustavu c_S = 1 i p_S = 5, koristeći algoritme raspoređivanja RMS i PS*

Slika 4.4 *Rezultati raspoređivanja zadataka prikazanih slikom 4.2 na jednoprocesorskom sustavu $c_S = 1$ i $p_S = 5$, koristeći algoritme raspoređivanja RMS i DS*

Slika 4.5 *Rezultati raspoređivanja zadataka prikazanih slikom 4.2 na jednoprocesorskom sustavu $c_S = 1$ i $p_S = 5$, koristeći algoritme raspoređivanja RMS i SS*

Usporedbom dijagrama vidljivo je kako su zadaci vrlo slično raspoređeni što se također vidi i iz tablice 4.1.

Tablica 4.1. *Izračunate vrijednosti raspoređivanja RMS-a s obzirom na algoritam za raspoređivanje aperiodnih zadataka*

RMS +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	1.17	2.75	-	8.33	4	1	14	100	2
DS	1.17	2.75	-	9.33	1	1	14	100	2
SS	1.17	2.75	-	9.33	1	1	14	100	2

Vidljivo je da su svi odzivi gotovo jednaki osim odziva zadatka A1 i P4. Odziv aperiodnog zadatka A1 veći je zbog same prirode rada algoritma PS koji ne pamti kapacitet, te je zbog toga nešto veći, što je ujedno utjecalo na smanjenje odziva periodnog zadatka P4 na način da se P4 izvršio jednu vremensku jedinicu ranije što je vidljivo iz rezultata odziva $t_{odz(P4)}$. Kako aperiodni zadaci imaju prednost pri izvršavanju, kod algoritama koji čuvaju kapacitet poslužitelja, zadatak A1 se izvršio odmah te nije bilo mjesta za izvršenje periodnog zadatka P4 stoga se P4 zadatak izvršio jednu vremensku jedinicu kasnije. Zadatka P3 se nije izvršio, stoga nema podatka o njegovom odzivu što je rezultiralo prekoračenjem vremena nužnog završetka.

Jednaki postupak proveden je na ostalim algoritmima za raspoređivanje periodnih zadataka i dobiveni su sljedeći rezultati prikazani tablicama 4.2, 4.3 i 4.4.

Tablica 4.2. *Izračunate vrijednosti raspoređivanja EDF-a s obzirom na aperiodni algoritam*

EDF +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	1.17	4.25	4	12	4	1	14	100	0
DS	1.17	4.25	4	12.5	1	1	14	100	0
SS	1.17	4.25	4	12.5	1	1	14	100	0

Tablica 4.3. *Izračunate vrijednosti raspoređivanja LL-a s obzirom na aperiodni algoritam*

LL +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	1	3.5	3	9.5	4	1	14	100	0
DS	1	3.25	3	9.5	1	1	14	100	0
SS	1	3.25	3	9.5	1	1	14	100	0

Tablica 4.4. *Izračunate vrijednosti raspoređivanja LL-a sa značajnim događajem s obzirom na aperiodni algoritam*

$LL_{(SE)}$ +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	1	2	2	9.5	4	1	14	100	0
DS	1	2	2	8.5	1	1	14	100	0
SS	1	2	2	8.5	1	1	14	100	0

S obzirom na rezultate, moguće je zaključiti da odabir algoritma za raspoređivanje aperiodnih zadataka puno manje utječe na ukupni rezultat raspoređivanja nego odabir algoritma za raspoređivanje periodnih zadataka. To je posljedica toga što se aperiodni zadaci pojavljuju samo jednom, a periodni više puta.

Uspoređujući algoritme za raspoređivanje aperiodnih zadataka može se zaključiti da algoritmi DS i SS daju jednake rezultate, dok PS nešto lošije rezultate. Ishod toga jest što PS ne čuva kapacitet poslužitelja.

Usporedivši algoritme za raspoređivanje periodnih zadataka može se zaključiti da najbolje karakteristike ima LL(SE). Najlošije rezultate je dao algoritam RMS. Pri korištenju algoritma RMS, dva puta je prekoračeno vrijeme nužnog završetka što se vidi odabirom tabličnog prikaza u aplikaciji, a prikazano je na slici 4.6.

☰ tablični prikaz

Vrijeme	Izvođenje	Završetak	Prekoračenje
0	P1	–	–
1	P2	P1	–
2	P2	–	–
3	P4	P2	–
4	P4	–	–
5	P4	–	–
6	P1	–	–
7	P4	P1	–
8	P2	P4	–
9	P2	–	–
10	A1	P2	P3
11	P4	A1	–
12	P1	–	–
13	P4	P1	–
14	P4	–	–
15	A2	–	–
16	P2	A2	–
17	P2	–	–
18	P1	P2	–
19	P4	P1	–
20	A3	P4	–
21	P3	–	–
22	P4	–	–
23	P4	–	–
24	P1	–	–
25	A3	P1	P3
26	P2	–	–
27	P2	–	–
28	P4	P2	–
29	P4	–	–
30	A3	P4	–
31	P1	A3	–
32	–	P1	–

Slika 4.6 *Tablični prikaz raspoređivanja algoritma RMS i PS*

Na istom skupu zadataka provedeno je testiranje na različitom broju poslužitelja kako bi se vidjelo na koji način broj poslužitelja utječe na rezultate. Dio rezultata je prikazan tablično (tablica 4.5, 4.6, 4.7 i 4.8), a ostatak rezultata je u digitalnom obliku u prilogu.

Tablica 4.5. *Rezultati raspoređivanja algoritmima RMS i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	1.17	2.75	-	8.33	4	1	14	100	2
2	1	2	3	5	4	1	14	56.25	0
4	1	2	2	4	4	1	14	28.91	0

Tablica 4.6. *Rezultati raspoređivanja algoritmima EDF i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	2.17	4.25	4	12	4	1	14	100	0
2	1	2	2.5	5.33	4	1	14	56.25	0
4	1	2	2	4	4	1	14	28.91	0

Tablica 4.7. *Rezultati raspoređivanja algoritmima LL i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	1	3.5	3	9.5	4	1	14	100	0
2	1	2	2	4.67	4	1	14	56.25	0
4	1	2	2	4	4	1	14	28.91	0

Tablica 4.8. *Rezultati raspoređivanja algoritmima LL(SE) i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	1	2	2	9.5	4	1	14	100	0
2	1	2	2	4.67	4	1	14	56.25	0
4	1	2	2	4	4	1	14	28.91	0

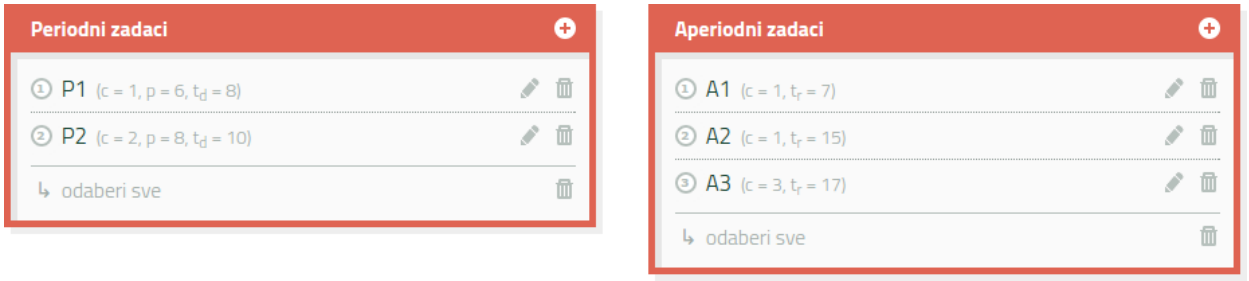
Iz priloženih rezultata vidljivo je kako povećanjem broja poslužitelja opadaju vremena odziva kao i broj prekoračenja vremena nužnog završetka, no opada iskoristivost poslužitelja što je očekivano zbog toga što se jednak broj zadataka izvršava na većem broju poslužitelja.

Primijeti se i puno veća razlika između korištenja jednog i dva poslužitelja nego li je razlika između korištenja dva i četiri poslužitelja što se vidi iz vremena odziva. Također se može primijetiti da je RMS dao slabije rezultate za jedan poslužitelj u odnosu na ostale, jer ima prekoračenje vremena nužnog završetka koje se može izbjeći povećanjem broja poslužitelja.

RMS daje brže odzive za zadatke manjeg trajanja, ali zbog toga zadaci većeg trajanja imaju veće odzive što može rezultirati prekoračenjem vremena nužnog završetka. Udvostručenjem broja poslužitelja, ukupna korisnost poslužitelja se gotovo dvostruko smanjila.

4.2. Rezultati i analiza raspoređivanja aperiodnih zadataka u sustavu pri malom opterećenju

Drugi test je proveden sa zadacima koji malo opterećuju sustav (slika 4.7). Usporedbom rezultata analize zaključak analize je da algoritmi za raspoređivanje aperiodnih zadataka imaju mali utjecaj na ukupan rezultat. Razlog tome je manja opterećenost poslužitelja prilikom raspoređivanja aperiodnih zadataka u odnosu na raspoređivanje periodnih zadataka. Prikazani su samo rezultati provedenog testa sa algoritmom SS za raspoređivanje aperiodnih zadataka dok se ostali testovi nalaze u digitalnom obliku u prilogu.



Slika 4.7 *Zadaci na poslužiteljima*

Rezultati testiranja prikazani su tablicom 4.9.

Tablica 4.9. *Izračunate vrijednosti raspoređivanja SS-a s obzirom na periodni algoritam*

SS +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
RMS	1.17	2.75	1	1	14	59.38	0
EDF	1.17	2.75	1	1	14	59.38	0
LL	1	2	1	1	14	59.38	0
LL(SE)	1	2	1	1	14	59.38	0

Rezultati su ponovno pokazali kako smanjenjem opterećenja pada i korisnost, ali vremena odziva su bolja i opada broj prekoračenja vremena nužnog završetka. Vidljivo je da su algoritmi LL dali bolje rezultate od algoritama RMS i EDF te da je pri malom opterećenju manja i razlika među algoritmima.

4.3. Rezultati i analiza raspoređivanja aperiodnih zadataka u sustavu pri velikom opterećenju

Treći test je proveden na istom skupu zadataka koji više opterećuju sustav u odnosu na prethodne (slika 4.8).

Slika 4.8 *Zadaci na poslužiteljima*

Provodeći test s gore prikazanim zadacima dobiveni su sljedeći rezultati koje prikazujemo tablicama 4.10, 4.11, 4.12 i 4.13.

Tablica 4.10. *Izračunate vrijednosti raspoređivanja algoritmima RMS i DS s obzirom na broj poslužitelja*

n_p	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(P5)}$	$t_{odz(P6)}$	$t_{odz(P7)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	$t_{odz(A4)}$	$t_{odz(A5)}$	μ (%)	n_{ND}
1	1.17	2.75	-	-	13	-	-	1	1	14	-	-	100	7
2	1	2	-	11	5.33	5.5	4.7	1	1	5	4	4	100	2
4	1	2	3	4.33	5	3.33	3	1	1	3	1	2	60.16	0

Tablica 4.11. *Izračunate vrijednosti raspoređivanja algoritmima EDF i DS s obzirom na broj poslužitelja*

n_p	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(P5)}$	$t_{odz(P6)}$	$t_{odz(P7)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	$t_{odz(A4)}$	$t_{odz(A5)}$	μ (%)	n_{ND}
1	4.67	9.33	9.5	11	23	27	14	1	1	14	-	-	100	5
2	1	2	2.5	9	11.67	7	6.67	1	1	3	1	2	100	2
4	1	2	2	4	5.5	3.33	3	1	1	3	1	2	60.16	0

Tablica 4.12. *Izračunate vrijednosti raspoređivanja algoritmima LL i DS s obzirom na broj poslužitelja*

n_p	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(P5)}$	$t_{odz(P6)}$	$t_{odz(P7)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	$t_{odz(A4)}$	$t_{odz(A5)}$	μ (%)	n_{ND}
1	1	4.5	4	15	16	27	18	1	1	14	-	-	100	7
2	1	2	2	8.5	8.67	7	9.33	1	1	5	4	4	100	2
4	1	2	2	3.33	5	3.33	3.33	1	1	3	1	2	60.16	0

Tablica 4.13. *Izračunate vrijednosti raspoređivanja algoritmima LL(SE) i DS s obzirom na broj poslužitelja*

n_p	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(P5)}$	$t_{odz(P6)}$	$t_{odz(P7)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	$t_{odz(A4)}$	$t_{odz(A5)}$	μ (%)	n_{ND}
1	1	6.5	2	15	16	27	18	1	1	14	-	-	100	7
2	1	2	2	8.5	8.67	7	8.67	1	1	5	4	4	100	2
4	1	2	2	3.33	5	3.33	3.33	1	1	3	1	2	60.16	0

Na osnovu rezultata primijeti se da ukoliko je sustav preopterećen što je ovdje bio slučaj s jednim poslužiteljem, najbolje rezultate dao je EDF jer ima 5 prekoračenja vremena završetaka dok ostali imaju po 7, ali EDF ima veća vremena odziva.

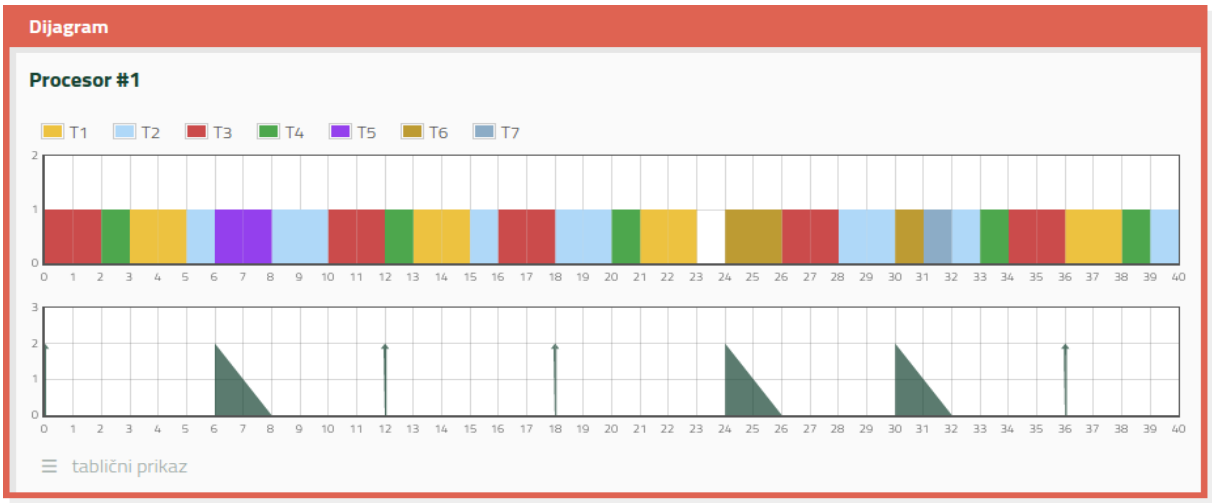
Niti jedan algoritam nije uspio izvršiti sve zadatke barem jednom. Ukoliko je sustav imalo preopterećen, u ovom slučaju sustav s dva poslužitelja, svi algoritmi dali su slične rezultate.

Jedino RMS nije uspio izvršiti sve zadatke barem jednom, ali pokazuje bolja vremena odziva u odnosu na ostale. Ako je sustav preopterećen, dodavanjem novih poslužitelja može se riješiti problem pa je tako vidljivo da su u sustavu s četiri poslužitelja svi algoritmi dali zadovoljavajuće rezultate s vrlo sličnim odzivima, ali to je bilo i za očekivati jer je sustav s četiri poslužitelja opterećen samo 60%.

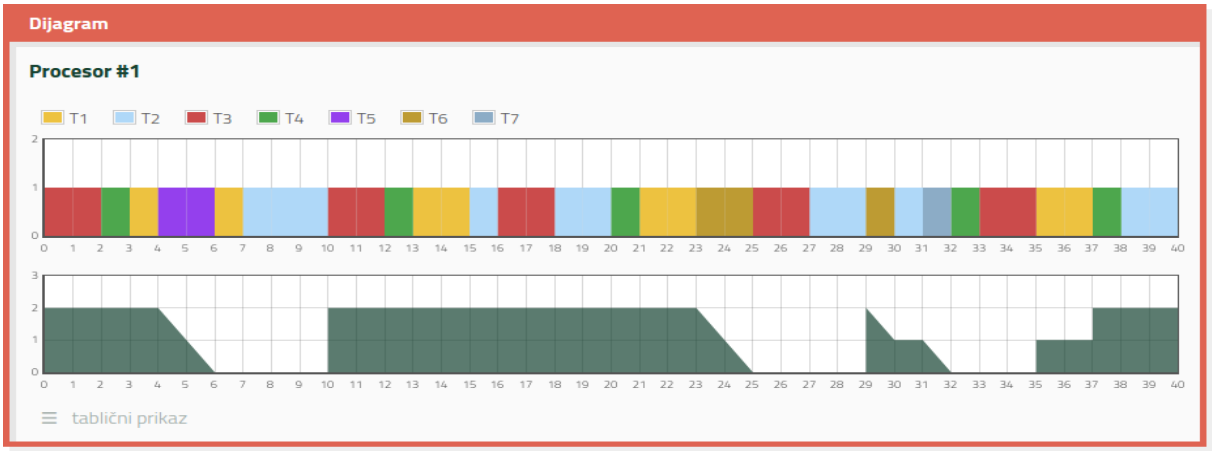
Testiranjem provedenim na skupu zadataka s ispita te kapacitetom poslužitelja $c_S = 2$ i periodom obnavljanja $p_S = 6$ u trajanju simulacije od 40 vremenskih jedinica dobili smo sljedeće rezultate. Karakteristike ispitnih zadataka korištenih u testiranju opisane su slikom 4.9.

Slika 4.9 *Ispitni zadaci na poslužiteljima*

Na slikama 4.10, 4.11 i 4.12 prikazani su rezultati za algoritam EDF s odgovarajućim algoritmom za raspoređivanje aperiodnih zadataka. Slike ostalih algoritama nalaze se u dodatku.

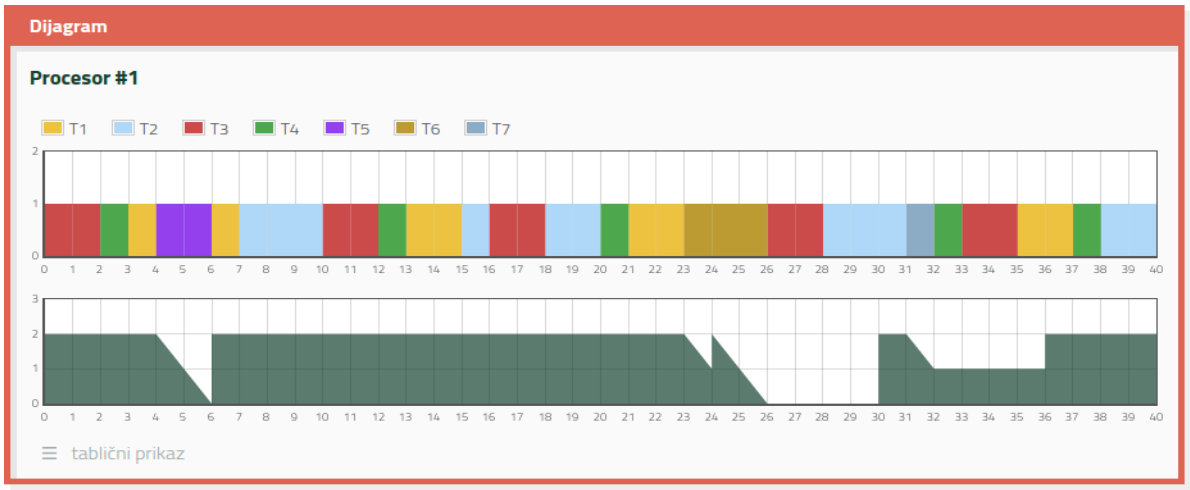


Slika 4.10 *Rezultati raspoređivanja zadataka prikazanih slikom 4.9 na jednoprocesorskom sustavu c_S = 2 i p_S = 6, koristeći algoritme raspoređivanja EDF i PS*



Slika 4.10 *Rezultati raspoređivanja zadataka prikazanih slikom 4.9 na jednoprocesorskom sustavu c_S = 2 i p_S = 6, koristeći algoritme raspoređivanja EDF i PS*

Slika 4.11 *Rezultati raspoređivanja zadataka prikazanih slikom 4.9 na jednoprocesorskom sustavu c_S = 2 i p_S = 6, koristeći algoritme raspoređivanja EDF i DS*



Slika 4.12 *Rezultati raspoređivanja zadataka prikazanih slikom 4.9 na jednoprocesorskom sustavu c_S = 2 i p_S = 6, koristeći algoritme raspoređivanja EDF i SS*

Pregled rezultata po algoritmima raspoređivanja prikazani su tablicama 4.14, 4.15, 4.16 i 4.17.

Tablica 4.14. *Izračunate vrijednosti raspoređivanja RMS-a s obziorm na aperiodni algoritam*

RMS +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	4	13.66	2.4	1.8	4	8	1	97.5	2
DS	4.25	13.33	2.4	1.8	2	3	1	100	2
SS	4.25	13.33	2.2	1.6	2	7	1	100	2

Tablica 4.15. *Izračunate vrijednosti raspoređivanja EDF-a s obziorm na aperiodni algoritam*

EDF +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	5.25	9	3.2	4	4	8	1	97.5	0
DS	5.5	8.33	3	3.6	2	3	1	100	0
SS	5.5	8.33	2.8	3.6	2	7	1	100	0

Tablica 4.16. *Izračunate vrijednosti raspoređivanja LL-a s obziorm na aperiodni algoritam*

LL +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	2.25	5.33	2	1	4	8	1	97.5	0
DS	2.75	4	2	1	2	3	1	100	0
SS	2.75	4.33	2	1	2	7	1	100	0

Tablica 4.17. *Izračunate vrijednosti raspoređivanja LL(SE)-a s obziorm na aperiodni algoritam*

$LL_{(SE)}$ +	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
PS	2	5	2	1	4	8	1	97.5	0
DS	3	4	2	1	2	3	1	100	0
SS	3	4.33	2	1	2	7	1	100	0

Iz dobivenih rezultata vidi se da algoritam LL (neovisno o inačici) daje najkraća vremena odziva, dok RMS ima najslabije rezultate. Tablicama 4.18, 4.19, 4.20 i 4.21 prikazani su rezultati za algoritme raspoređivanja na skupu zadataka s ispita s obzirom na broj poslužitelja.

Tablica 4.18. *Rezultati algoritama RMS-a i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	5	13.66	2.4	1.8	4	8	1	97.5	2
2	2.25	4	2	1	4	8	1	51.25	0
4	2	3	2	1	4	8	1	25.625	0

Tablica 4.19. *Rezultati algoritama EDF-a i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	5.25	9	3.2	4	4	8	1	97.5	0
2	2.25	4	2	1	4	8	1	51.25	0
4	2	3	2	1	4	8	1	25.625	0

Tablica 4.20. *Rezultati algoritama LL-a i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	2.25	5.33	2	1	4	8	1	97.5	0
2	2	3	2	1	4	8	1	51.25	0
4	2	3	2	1	4	8	1	25.625	0

Tablica 4.21. *Rezultati algoritama LL(SE)-a i PS s obzirom na broj poslužitelja*

$n_{processor}$	$t_{odz(P1)}$	$t_{odz(P2)}$	$t_{odz(P3)}$	$t_{odz(P4)}$	$t_{odz(A1)}$	$t_{odz(A2)}$	$t_{odz(A3)}$	μ (%)	n_{ND}
1	2	5	2	1	4	8	1	97.5	0
2	2	3	2	1	4	8	1	51.25	0
4	2	3	2	1	4	8	1	25.625	0

Rezultati u tablicama prikazuju kako broj poslužitelja raste tako opada korisnost, ali i vremena odziva kao i broj zadataka koji su prekoračili vremena nužnog završetka. Usporede li se rezultati ovog skupa zadataka s onim iz prvog testiranja čiji su rezultati dani tablicama 4.1 do 4.8, može se primjetiti slično ponašanje, jer u oba slučaja sustav nije pri ekstremnom opterećenju (opterećenje nije ni malo, ni veliko).

5. ZAKLJUČAK

Metode obrade podataka u računalnim sustavima uvelike olakšavaju i ubrzavaju rješavanje zadataka u bilo kojem području života. Brzina, preciznost i pouzdanost ovise o izabranoj metodi obrade, te o raspoloživim resursima. Raspoređivanje zadataka u računalnim sustavima se temelji na nekoliko ključnih algoritama čiji je cilj optimalno rasporediti opterećenje uz pomoć poslužitelja. Kako većina sustava treba svoj rad obavljati u stvarnom vremenu, vrlo je važno da što manje zadataka prekorači vrijeme nužnog završetka, kako ne bi rezultiralo pogreškom u radu sustava. Razvijeni sustav za simulaciju raspoređivanja zadataka na jednom ili više poslužitelja bazira se na web aplikaciji koja je izrađena u PHP i JavaScript programskim jezicima koristeći *jQuery* uz Flot dodatak za prikaz grafova u web pregledniku koristeći metode objektno orjentiranog PHP-a u radnom okviru MVC arhitekture. Kroz aplikaciju se mogu odabrati razni algoritmi raspoređivanja. Za raspoređivanje periodnih zadataka ugrađeni su algoritmi RMS, EDF i LL sa i bez značajnog događaja, a za raspoređivanje aperiodnih zadataka algoritmi PS, DS i SS. Aplikacija se sastoji od modela zadataka i modela algoritama raspoređivanja. Modeli zadataka sadrže metode za učitavanje zadataka u aplikaciju i pohranjivanje u XML datoteku. Modeli algoritama njihovu logiku raspoređivanja. Sustav bi mogao biti unaprijeđen ukoliko mu se doda upravitelj koji bi kombinirao razne algoritme kako bi dobio bolje rezultate. Sustav radi sa statičkim zadacima tj. svi parametri zadataka su poznati prije samog raspoređivanja dok u stvarnom vremenu to nije uvijek tako. Ako se sustav želi primijeniti na skup zadataka kod kojih je moguće da se neki zadaci pojave bez najave treba ga napraviti dinamički. Moguće je ugraditi upravitelj koji će po potrebi uključivati dodatni poslužitelj ili promijeniti algoritam raspoređivanja kako bi se dobili bolji rezultati.

LITERATURA

[1] J. W. S. Liu, Real-Time Systems, Pearson Education, Upper Saddle River, NJ, USA, 2000.

[2] W. Ecker, W. Muller, R. Domer, Hardware-Dependent Software, Springer, science and Business Media, 2009.

[3] G. Martinović, Ž. Hocenski, L. Budin, A Tool for Evaluation of Scheduling Algorithms in Real-Time Systems, Advances in Signal Processing and Computer Technologies, str. 173-179, MA, USA, 2001.

[4] J. A. Stankovic, K. Ramamritham, S. Cheng, Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems, IEEE Transactions on Computers, br. 12, sv. c-34, str. 1130-1143, 1985.

[5] EECE 276, Embedded Systems, RTOS Basics Process Scheduling EECE 276 Course - Embedded Systems, Vanderbilt University, Nashville, TN, USA, 2006.

[6] I. Lee, CSE 480/CIS 700: OS Overiew-Real-time Scheduling, -Department of Computer and Information Science, University od Pennsylvania, PA, USA, 2006.

[7] G. Martinović, Ž. Hocenski, L. Budin, Validation of Scheduling Algorithms for Aperiodic Tasks in the Real-TimeSystems, Information Technology Interfaces, Pula, Croatia, 2000.

[8] G. Martinović, Ž. Hocenski, L. Budin, Undergraduate Teaching of Real-Time Scheduling Algorithms by Developed Software Tool, IEEE Transactions on Education, br. 1, sv. 46, str. 185 – 196, 2003.

[9] P. Kuacharoen, M. A. Shalan, V. J. Mooney III, A Configurable Hardware Scheduler for Real-Time Systems, Center for Research on Embedded Systems and Technology, School of Electrical And Computer Engineering, Georgira Institute Od Technology, Atlanta, GA, USA, 2003.

[10] H. Cho, B. Ravindran, E. D. Jensen, An Optimal Real-Time Scheduling Algorithm for Multiprocessors, Proceedings of 27th IEEE International Symposium based on Real-Time Systems, Rio de Janeiro, Brazil, 05. – 08. prosinac 2006.

[11] A. Mohammadi, S. G. Akl, Scheduling Algorithms for Real-Time Systems, Technical Report No. 2006-499, School of computing, Queen's University, Kingston, Ontario, srpanj 2005.

[12] J. Carpenter, S. Funk, A Categorization of Real-Time Multiprocessor Scheduling Problems and Algorithms, Handbook on Scheduling Algorithms, Methods and Models, Chapman Hall/CRC, Boca Raton, FL, USA, 2004.

[13] J. Anderson, V. Bud, U. C. Devi, An EDF-based scheduling algorithm for multiprocessor soft real-time systems, Proceedings of the 17th Euromicro Conference on Real-Time Systems (ECRTS), str. 199–208, Balearic Islands, Španjolska, 06. – 08. srpanj 2005.

[14] J. A. Stankovic, Deadline Scheduling for Real Time Systems – EDF and Related Algorithms, Kulwer Academic Publications, New York, NY, USA, 1998.

[15] J. Stankovic, K. Ramamritham, The Spring Kernel: A New Paradigm for Real-Time Systems, IEEE Software, br. 3, sv. 8, str. 62-72, 1991.

[16] L. Zhou, Real-Time Performance Guarantees in Manufacturing Systems, Michigan Academic Publications, str. 4 – 16, 1999.

[17] A. Silberschatz, P. Baer Galvin, G. Gagne, Operating System Concepts, 9. izdanje, John Wiley & Sons, Inc., 2013.

KRATICE

BS - Basic Scheduling

DDS - Deadline Driven Scheduling

DL - Deadline

DS - Deferrable Server

EDF - Earliest Dedline First, EDF

GS - General Scheduling

LL - Least Laxity

LLF - Least Laxity First

MVC - Model–View–Controller

PHP - Hypertext Preprocessor

PS - Polling Server

PSch - Preemptive Scheduling

RMS - Rate Monotonic Scheduling

SS - Sporadic Server

WCET - the Worst-Case Execution Time

XML - Extensible Markup Language

SAŽETAK

S ciljem optimiranja rada računalnog sustava koriste se algoritmi raspoređivanja zadataka s poslužiteljskim zadacima. Načelo rada algoritma mijenja se ovisno o stanju zadatka ili vremenskim uvjetima odrade zadatka. U izrađenom progamskom rješenju (web aplikaciji) testirani su često korišteni algoritmi raspoređivanja u sustavima s jednim i više polužitelja. Napravljena je usporedba rezultata izračunom i dijagramskim prikazom. Primjećeno je na korištenim primjerima problema da je bitno opterećenje sustava, gdje je u slučaju preopterećenosti rješenje dodavanje dodatnih poslužitelja. Aplikacija omogućuje raspoređivanje zadataka zasnovano na nekoliko ključnih algoritama kao što su algoritmi RMS, EDF i LL, te PS, DS i SS. Razvijeni sustav za simulaciju raspoređivanja zadataka na jednom ili više poslužitelja zasniva se na web aplikaciji koja nudi mogućnost odabiranja algoritma raspoređivanja i broja poslužitelja. Sustav za simulaciju raspoređivanja zadataka radi sa statičkim zadacima.

Ključne riječi: algoritmi raspoređivanja periodnih zadataka, algoritmi raspoređivanja aperiodnih zadataka, arhitektura MVC, sustav simulacije raspoređivanja zadataka.

ABSTRACT

For an optimised system work, computer systems use scheduling algorithms for server tasks. Principle of the algorithm is changed depending on task status or task time conditions. In created program solution (web application) most frequent placing algorithms were tested in systems with one or more servers. Comparison of the results has been made by calculations and diagrams. It has been noticed on tested example problems that there is a significant load on the system where, in case of overload, solution is to add more servers. The application allows tasks scheduling based on a few key algorithms like RMS, EDF, LL, PS, DS and SS. The developed system for the simulation of tasks scheduling to one or more servers is based on Internet application that offers the possibility of selecting a scheduling algorithm and the servers number. System for scheduling tasks simulation works with static tasks.

Keywords: periodic tasks scheduling algorithm, aperiodic tasks scheduling algorithm, MVC architecture, scheduling tasks simulation system.

ŽIVOTOPIS

Rođen u Hrvatskoj, 09. travnja 1986. Pohađao osnovnu školu „Vladimira Nazora“ u gradu Vinkovci. Nakon osnovne škole upisao prirodoslovno-matematički smjer u gimnaziji „Matija Antun Reljković“, Vinkovci, zatim Elektrotehnički fakultet Osijek, smjer Računarstvo. U prosincu 2006. godine zapošljava se na pučkom otvorenom učilištu Algebra na kojem predaje web dizajn, grafički dizajn, održavanja informatičkih sustava i ukupnog IT potrebnog za poslovanje istočne regije. Početkom siječnja 2009. godine osniva IT Sektor d.o.o. koji se bavi uslužnim informatičkim djelatnostima. Krajem studenog 2013. godine preuzima Terra Argenta d.o.o. u čijem je vlasništvu nekretnina trgovački centar „Golubica Mall“. Radi centar management za Golubica Mall (Vukovar, Hrvatska) i Duga Mall (Crikvenica, Hrvatska). U siječnju 2016. godine zapošljava se u Konzum d.d. kao direktor upravljanja K Centara za sve trgovačke centre u vlasništvu Konzum d.d. i Mercator-H d.o.o. u Hrvatskoj. Upravlja sa 655.915,15 m^2 prostora za zakup i odgovoran je za prihod od 39 milliona eura godišnje. U studenom 2016. godine osniva 2.advice j.d.o.o. koji se bavi centar managementom i savjetovanjem trgovačkih centara, trenutna vrijednost projekata u razvoju iznosi cca 250 miliona eura.