ALGORITMI PLANERA HRTS-a SA TOLERISANJEM GREŠKE

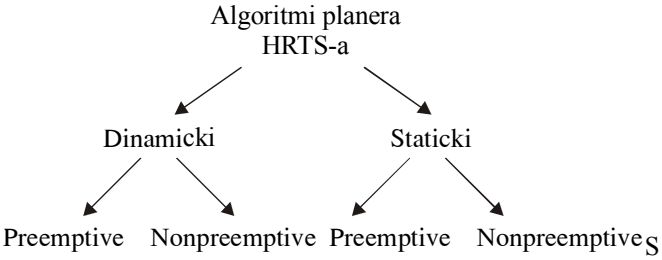
Sandra Brankov, Milun Jevtić, *Elektronski fakultet u Nišu*

Sadržaj – *Osnovni kriterijumi po kojima algoritmi izvršavaju zadatke kod real-time sistema su rokovi izvršenja zadataka ili njihova vrednost tj. činjenica koliko je zadatak bitan sistemu. Kombinacijom ova dva kriterijuma nastao je fault tolerant ADM (adaptive deadline-monotonic) algoritam koji je i predstavljen u ovom radu. Posebna pažnja posvećena je modifikaciji ADM algoritma za slučaj pojavljivanja zadatka sa najvećim prioritetom koji ima mogućnost da prekine izvršenje bilo kog trenutno aktivnog zadatka.*

1. UVOD

Osnovna karakteristika po kojoj se razlikuju *real-time* sistemi (sistemi koji rade u realnom vremenu) od ostalih sistema jeste ponašanje u vremenu. Da bi *real-time* sistem (RTS) ispravno funkcionisao potrebno je ne samo da daje korektan rezultat na izlazu već i da ga da u tačno definisanom vremenskom intervalu, [1]. Ovo je posebno kritično kod *hard real-time* sistema (HRTS) jer neblagovremeno izvršavanje zadataka može dovesti do katastrofe. Zbog toga se pri realizaciji HRTS-a koriste tehnike koje omogućavaju rad sistema i kada se pojavi neki otkaz tj. realizuju se kao *fault tolerant* sistemi (sistemi sa tolerancijom greške). U ovom radu će se razmatrati jedna tehnika rada HRTS-a pri pojavi otkaza, koja je bazirana na korišćenju redudanse u vremenu.

Da bi se svi zadaci jednog hard real-time sistema izvršili striktno u predviđenim vremenskim rokovima treba napraviti algoritam rada planera koji će definisati koji zadatak i u koje vreme će se izvršiti u skladu sa vremenskim rokovima, prioritetima i dostupnim resursima.



l.1. *Podela algoritama vremenskog planiranja izvršenja zadataka*

Slično kao i RTS-i i algoritmi se mogu podeliti na *soft* i *hard,* [2]. *Hard* algoritami se dalje dele na dinamičke i statičke. Dinamički algoritmi su zasnovani na dinamičkim parametrima koji se mogu menjati tokom rada sistema dok su statički zasnovani na fiksnim parametrima koji su prethodno dodeljeni zadacima i koji se u toku rada sistema ne mogu menjati. I statički i dinamički algoritmi se dele na *preemptive* i *nonpreemptive*. Kod *preemptive* zadatak koji se izvršava može biti prekinut dolaskom zadatka višeg prioriteta dok kod *nonpreemptive* zadatak koji se izvršava ne može biti prekinut.

2. ALGORITMI VREMENSKOG PLANIRANJA

Razmatraćemo tri algoritma: DMS, VBS i ADM, [3], uporedićemo njihove karakteristike i dati predlog modifikacije ADM algoritma za slučaj pojavljivanja kritičnog zadatka koji ima mogućnost da prekine izvršenje trenutno aktivnog zadatka.

DMS (*deadline-monotonic scheduling*) algoritam je vremenski baziran algoritmi kod koga je osnovni kriterijum prilikom određivanja rasporeda izvršenja zadataka rok zadatka što znači da će se prvo izvršavati zadatak koji ima najkraći vremenski rok.

VBS (*value-based scheduling*) algoritam je vrednosno baziran algoritam kod koga se raspored izvršenja zadataka pravi na osnovu same vrednosti zadatka tj. činjenice koliko je zadatak važan sistemu.

Kombinacijom prethodna dva algoritma nastao je ADM (*adaptive deadline-monotonic*) algoritam koji daje bolje rezultate (u odnosu na prethodna dva algoritma) u slučajevima kada se u toku rada sistema jave greške koje dovode do preopterećenja procesora tj. sistema.

DMS algoritmi su popularni jer su jednostavni i mogu da zadovolje sva vremenska ograničenja i daju garanciju da će svi rokovi zadataka biti ispoštovani. Međutim, ako se jave greške koje dovode do preopterećenja sistema vremenski bazirani algoritmi ne daju dobre rezultate jer ne uzimaju u obzir vrednost zadatka prilikom određivanja prioriteta zadataka u sistemu, što može dovesti do neizvršenja nekih kritičnih zadataka. Problem se može rešiti korišćenjem vremensko-vrednosno baziranih algoritama kod kojih su i vreme i vrednost zadataka kriterijumi prilikom određivanja priorita izvršenja zadataka.

U grupu vremensko-vrednosno baziranih algoritama spada i ADM algoritam. Ovaj algoritam pomoću mehanizma nazvanog promenljivi okviri (*sliding-windows*), [3], omogućava prelazak sa vremenski baziranih na vrednosno bazirane algoritme u trenucima kada trenutno aktivni algoritmi ne mogu da daju zadovoljavajuće rezultate. Ovim mehanizmom se teži da se izbegne degradacija performansi sistema prilikom promene moda suviše rano ili suviše kasno.

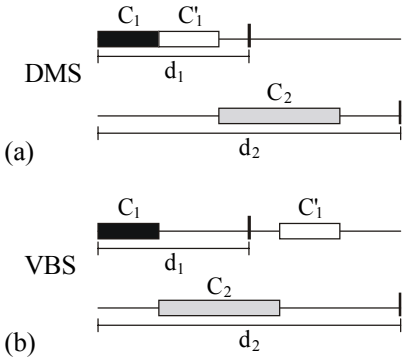
2.1 POREĐENJE ALGORITAMA

Posmatrajmo dva real-time zadatka τ_1 i τ_2 koji stižu istovremeno i koji imaju parametre C_i, d_i i v_i, i=1,2 koji su dati u tabeli 1., gde je C_i vreme potrebno da se izvrši zadatak τ_i, d_i rok za izvršenje zadatka τ_i i v_i vrednost zadatka τ_i. Neka je i u toku izvršavanja zadatka τ_1 došlo do detektovanja greške tako da je taj zadatak neophodno ponovo izvršiti.

Tabela 1. *Parametri zadatka τ_1 i τ_2*

Zadatak	C_i	d_i	v_i
τ_1	4	10	5
τ_2	8	20	20

Na slici 2. dat je primer pojave greške koja ne dovodi do preopterećenja sistema kao i realizacija zadataka u slučaju da je kada je aktivan (a) DMS algoritam i (b) VBS algoritam.



Sl.2. *Primer pojave greške koja ne dovodi do preopterećenja sistema kada je aktivan (a) ADM i (b) VBS algoritam*

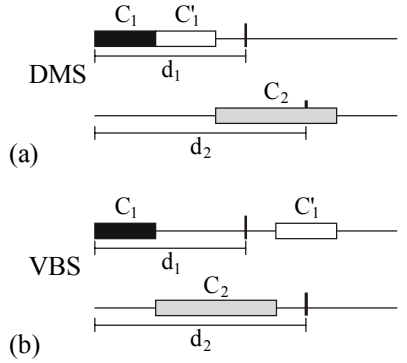
Ako je aktivan DMS algoritam nakon neuspešnog izvršenja zadatka τ_1 ovaj zadatak se ponovo izvršava a zatim se izvršava zadatak τ_2. U ovom slučaju se oba zadatka izvršavaju pre roka. Karakteristika za procenu kvaliteta algoritma planera izvršenja zadataka se dobija kao suma vrednosti zadataka pri čemu vrednost zadatka ima predznak + ako se zadatak izvršio na vreme a - ako nije.

Ako nakon neuspešnog izvršenja zadatka τ_1 dođe do promene moda tj. postaje aktivan VBS algoritam, prelazi se na izvršenje zadatka τ_2 jer je njegova vrednost, v_2, veća a zatim se izvršava τ_1. U ovom slučaju zadatak τ_1 će zakasniti sa izvršenjem što dovodi do degradacije karakteristika VBS algoritma datih u tabeli 2.

Tabela 2. *Karakteristike DMS i VBS algoritma*

Algoritam	Karakteristike
DMS	5+20=25
VBS	-5+20=15

Na osnovu tabele 2. izvodi se zaključak da je u ovom slučaju bolje koristiti DMS algoritam. Kako se svi zadaci završavaju pre roka d_2 ne dolazi do preopterećenja sistema jer se za izvršenje zadataka koji su zakasnili koristi procesorsko vreme predviđeno za realizaciju u našem slučaju zadatka τ_3.



Sl.3. *Primer pojave greške koja dovodi do preopterećenja sistema kada je aktivan (a) ADM i (b) VBS algoritam*

Na slici 3. dat je primer pojave greške koja dovodi do preopterećenja sistema kao i realizacija zadataka u slučaju da je kada je aktivan (a) DMS algoritam i (b) VBS algoritam.

Tabela 3. *Izmenjeni parametri zadatka τ_1 i τ_2*

Zadatak	C_i	d_i	v_i
τ_1	4	10	5
τ_2	8	14	20

U ovom slučaju smanjen je rok za izvršenje zadatka τ_2, tako da je sada d_2=14. U tabeli 3. dati su parametri zadatka.

Ako je aktivan DMS algoritam nakon neuspešnog izvršenja zadatka τ_1 ovaj zadatak se ponovo izvršava a zatim se prelazi na izvršenje zadatka τ_2. U ovom slučaju zadatak koji ima veću vrednost τ_2 kasni što će značajno degradirati karakteristike DMS algoritma.

Ako bi nakon neuspešnog izvršenja zadatka τ_1 postao aktivan VBS algoritam, onda bi se prvo izvršio zadatak τ_2, pa bi tek onda došlo do ponovnog izvršenja zadatka τ_1. Sada se zadatak većeg prioriteta τ_2 izvršava u roku dok zadatak nižeg prioriteta τ_1 kasni.

Tabela 4. *Karakteristike DMS i VBS algoritma*

Algoritam	Karakteristike
DMS	5-20=-15
VBS	-5+20=15

Na osnovu tabele 4., gde su date karakteristike DMS i VBS algoritma za ovaj slučaj, dolazi se do zaključka da je bolje koristiti VBS algoritam.

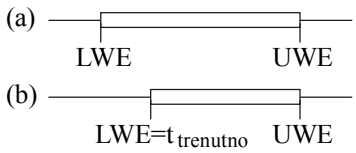
Na osnovu prethodnog može se zaključiti da kada greška dovodi do preopterećenja sistema bolje je izvršitit promenu moda i preći na VBS algoritam a kada greška ne dovodi do preopterećenja ne menjati mod tako da je stalno aktivan DMS algoritam.

Ovu činjenenicu koristi i ADM algoritam koji prvo treba da pretpostavi da li će greška dovesti do preopterećenja sistema ili ne a zatim da izvrši promenu moda sa DMS na VBS algoritam ako je to potrebno. Sve ovo ADM algoritam rešava korišćenjem mehanizma nazvanog promenljivi okviri. Promenljivi okvir svakog zadatka čuva procenu o tome koliko dozvoljenog vremena za njegovo izvršenje ostaje pre nego što zadatak zakasni. Ako je veličina okvira veća ili jednaka od 0, ADM pretpostavlja da neće doći do preopterećenja sistema i koristi DMS algoritam. Ako je veličina okvira manja od 0, ADM pretpostavlja da će doći do preopterećenja sistema i inicira promenu moda tako da VBS postaje aktivan algoritam.

Kada se javi greška ADM algoritam izračunava gornju i donju granicu promenljivih okvira za svaki zadatak. Niža granica okvira *LWE* (*lower window edge*) predstavlja najraniji trenutak kada može doći do promene moda. Da bi ADM odredio *LWE* okvira potrebno je da se za svaki zadatak izračuna njegov pseudorok kao sumu vremena izvršenja svih zadataka sa višim prioritetom, a zatim se *LWE* okvira postavlja na vrednost pseudoroka. Veličina okvira svakog zadatka se računa kao vreme koje se može dodati vremenu izvršenja zadatka a da pri tome zadaci nižeg prioriteta ne zakasne. Gornja granica promenljivog okvira *UWE* (*upper*

window edge) se dobija na osnovu podatka o veličini promenljivog okvira tj. važi da je

veličina okvira = UWE-LWE.



Sl.4. *Mehanizam promenljivih okvira (a) određivanje granica (b) okvir nakon detekcije greške*

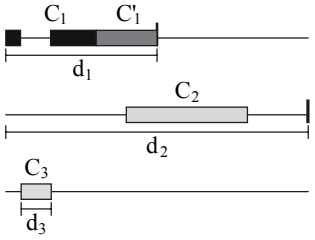
Na slici 4. prikazan je mehanizam promenljivih okvira. Za trenutno aktivan zadatak, ADM algoritam prvo određuje gornju i donju granicu klizećeg okvira (a). Ako se detektuje greška *LWE* se povećava na trenutno vreme, smanjujući veličinu prozora kao i preostalo vreme za izvršenje zadatka (b). Na slici je prikazan primer kada je nakon detekcije greške veličina okvira veća od nule sto znači da ne dolazi do promene moda.

Primenjujući ADM algoritam u našem prvom slučaju nakon pojave greške, koja ne dovodi do preopterećenja sistema, neće doći do promene moda i ostaće aktivan DMS algoritam. Ovde je veličina okvira prvog zadatka veća od nule, čime uslovi za promenu moda nisu ispunjeni. U drugom slučaju se javlja greška koja dovodi do preopterećenja sistema i nakon njene detekcije veličina okvira prvog zadatka postaje negativna što je uslov da ADM algoritam izvrši promenu moda.

2.2 MODIFIKACIJA ADM ALGORITMA

Posmatrajmo sada slučaj kada postoji mogućnost da se pored zadatka τ_1 i τ_2 pojavi i treći zadatak τ_3 (neka je na primer C_3=2 a v_3=50) koji je aperiodičan i čije je minimalno vreme između dva pojavljivanja t_{3min} veće od dužeg roku za izvršenja zadatka, u našem slučaju d_2. To znači da se zadatak τ_3 može pojaviti najviše jednom u intervalu koji nas zanima. Pored toga zadatak τ_3 ima najkraći rok za izvršenje kao i najveću vrednost tako da će imati i najveći prioritet bilo da je aktivan ADM bilo VBS algoritam i kao takav može prekinuti sa izvršenjem bilo koji zadatak koji je trenutno aktivan. Ovaj zadatak može predstavljati *interrupt* (prekid) kod *real-time* sistema i njega je realno očekivati.

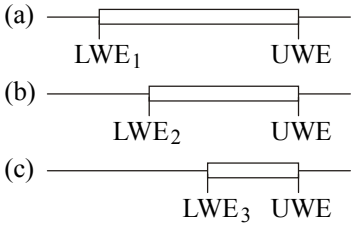
Sada, ADM algoritam prilikom izračunavanja granica klizećih okvira i odlučivanja da li menjati mod ili ne, mora uzeti u obzir i mogućnost pojave ovog zadatka. Najgori slučaj je da se zadatak τ_3 javi jednom u toku izvršavanja zadatka τ_1 i τ_2.



Sl.5. *Zadatak τ_3 prekida ivršenje zadatka τ_1*

Na slici 5. prikazan je slučaj kada se zadatak τ_3 javlja za vreme izvršenja zadatka τ_1 i kao zadatak sa najvećim prioritetom prekida njegovo izvršenje. Sada se vreme izvršenja zadatka τ_1 povećava za vreme izvršenja zadatka τ_3.

ADM algoritam mora prilikom određivanja gornje i donje granice okvira uzeti u obzir i parametre zadatka τ_3.



Sl.6. *Mehanizam promenljivih okvira (a) određivanje granica (b) okvir kada se zadatak τ_3 nije izvršio (c) okvir nakon uspešnog izvršenja zadatak τ_3*

Na slici 6. prikazan je:

(a) okvir pre izvršenja zadatka τ_1;

(b) stanje promenljivog okvira nakon izvršenja zadatka τ_1 u slučaju da se nije pojavio zadat τ_3;

(c) stanje promenljivog okvira nakon izvršenja zadatka τ_1 u slučaju da se pojavio zadat τ_3.

U slučaju (c) vidi se smanjenje promenljivog okvira zadatka τ_1 u odnosu na slučaj (b) i to korekcijom donje granice za vreme izvršenja zadatka τ_3 tj.

$LWE_3=LWE_2+C_3$

Konkretno za ovaj slučaj je veličina okvira smanjena, ali je i dalje veća od nule tako da nakon detekcije greške neće doći do promene moda i ostaje aktivan ADM algoritam.

U opštem slučaju, ako sa τ_i označimo zadatak za koga se vrši proračun veličine promenljivog okvira, a sa τ_j zadatak koji ima najveći prioritet i mogućnost da prekine bilo koji aktivan zadatak onda se određivanje granica okvira zadatka τ_i može izvršiti primenom sledećeg algoritma:

Određivanje_granica
if *(τ_i* **or** *greška$_i$)* **and not** *τ_j* **then**
  if *(t_k - $t_{j,n-1}$) ≤ t_{jmin}* **then** *LWE = $t_{trenutno}$*
  else *LWE = $t_{trenutno}$ + C_j*
  endif
else *LWE = $t_{trenutno}$*
endif
if *(UWE – LWE) < 0* **then** *Promena_moda*
endif

Znači, ADM algoritam prvo određuje gornju i donju granicu promenljivog okvira zadatka τ_i. Zatim se ispituje da li se izvršio zadatak τ_i ili je došlo do greške u toku njegovog izvršenja (greška$_i$) kao i da li se zadatak τ_j izvršio ili ne.

Ako jeste, *LWE* dobija vrednost trenutnog vremena $t_{trenutno}$ a zatim se ispituje veličina promenljivog okvira na osnovu koje se odlučuje da li će doći do promene moda ili ne.

Ako se zadatak τ_j nije izvršio prvo se proverava vremenski interval od njegovog zadnjeg pojavljivanja ($t_{j,n-1}$)

do trenutka kada se završava vremenski interval koji nas interesuje (t_k) (u našem slučaju, dva zadatka, to je vremenski interval od trenutka kada oba zadatka stižu na izvršenje do kraja roka za izvršenje drugog zadatka d_2 koji je u našem slučaju veći). Ako je taj vremenski interval manji od minimalnog intervala između dva pojavljivanja zadatka τ_j (t_{jmin}) onda se zadatka τ_j sigurno neće pojaviti i *LWE* dobija vrednost trenutnog vremena $t_{trenutno}$ i ne vrši se nikakva dodatna korekcija. A ako je taj interval veći od t_{jmin} onda je potrebno izvršiti korekciju donje granice okvira za vreme C_j tj. za vreme izvršenja zadatka τ_j.

Nakon određivanja granica okvira ispituje se njegova veličina na osnovu koje se odlučuje da li će doći do promene moda ili ne.

Ovom modifikacijom je ADM algoritam prilagođen za slučaj pojavljivanja jednog aperiodičnog zadatka za koga se zna vreme izvršenja, činjenica koliko je zadatak bitan sistemu kao i minimalno vreme između dva pojavljivanja ovog zadatka.

3. ZAKLJUČAK

U ovom radu razmatrali smo tri algoritma:

- DMS (*deadline-monotonic scheduling*) kod koga je osnovni kriterijum prilikom određivanja rasporeda izvršenja zadataka rok zadatka

- VBS (*value-based scheduling*) kod koga se raspored izvršenja zadataka pravi na osnovu same vrednosti zadatka tj. činjenice koliko je zadatak važan sistemu i

- ADM (*adaptive deadline-monotonic*) koji je nastao kombinacijom prethodna dva algoritma.

Ako u toku rada RTS-a dođe do greške koja ne opterećuje sistem uspešno se može primeniti DMS algoritam koji u ovom slučaju ima bolju karakteristiku od VBS algoritma. Međutim, ako se u toku rada RTS-a javi greška koja će dovesti do preopterećenja sistema DMS algoritam ne daje dobre rezultate. Sada VBS algoritam ima mnogo bolju karakteristiku i treba ga u ovom slučaju primeniti.

Kombinacijom DMS i VBS algoritma nastao je ADM algoritam. Kada se u toku rada RTS-a pojavi greške ADM algoritam pomoću mehanizma nazvanog promenljivi okviri određuje da li će greška koja se javila dovesti do preopterećenja sistema ili ne. Ako ne dolazi do preopterećenja sistema ostaje aktivan DMS algoritam dok u suprotnom dolazi do promene moda i postaje aktivan VBS algoritam. ADM algoritam u zavisnosti od situacije primenjuje jedan ili drugi algoritam tako da uvek bude aktivan onaj koji ima bolju karakteristiku.

Posebno je razmatran slučaj kada se u toku rada RTS-a, kod koga je aktivan ADM algoritam, javi zadatak koji ima najveći prioritet i kao takav mogućnost da prekine izvršenje bilo kog tadatka koji je trenutno aktivan. Realno, ovo se često javlja kod RTS-a gde taj zadatak može biti neki *interrupt* koga je neophodno obraditi u trenutku njegovog pojavljivanja.

Pretpostavili smo da je za ovaj zadatak (*interrupt*) poznato njegovo vreme izvršenja, činjenica koliko je on bitan sistemu kao i minimalno vreme između njegova dva pojavljivanja. U skladu sa ovim pretpostavkama dat je primer modifikacije ADM algoritma na osnovu koje ovaj algoritam može da pretpostavi da li će se *interrupt* pojaviti ili ne i u zavisnosti od toga da odredi veličinu promenljivog okvira na osnovu koje će odlučiti da li će ostati aktivan DMS algoritam ili je potrebno da dođe do promene moda tj. da postane aktivan VBS algoritam. I u ovom slučaju cilj je izabrati algoritam koji će imati bolju karakteristiku tj. dati bolje rezultate prilikom planiranja izvršenja zadataka kod RTS-a.

LITERATURA

[1] N. Nissanke, *Raltime Systems,* Prentice Hall 1997

[2] K. Juvva, *Real-Time Systems,* Carnegie Mellon University 18-849b Dependable Embedded Systems ili http://www-2.cs.cmu.edu/~koopman/des_s99/real_time/

[3] P. Richardson, L. Sieh and A.M. Elkateeb, "Fault-Tolerant Adaptive Scheduling for Embedded Real-Time Systems," *IEEE MICRO* September-October 2001 pp.41-51.

Abstract – The basic criteria how the algorithms execute tasks in real-time systems are the deadlines of the task execution or the tasks value, i.e. how much is the task important for the system. The product of combining these two criterias is the fault tolerant ADM (adaptive deadline-monotonic) algorithm, which is presented in this work. Special importance is put on the modification of the ADM algorithm for the case when a task with the topmost priority, which has the priority to interrupt any other task, appears.

FAULT TOLERANT SCHEDULING FOR HARD REAL TIME SYSTEMS

Sandra Brankov, Milun Jevtić