Multiprocesorsko raspoređivanje i raspoređivanje u realnom vremenu

Operativni sistemi 2

Institut za matematiku i informatiku

Institut za matematiku i informatiku
Prirodno-matematički fakultet, Kragujevac

Novembar 2010. god.

O čemu će biti reči?

Kategorije multiprocesorskih sistema

1. Labavo spregnut, distribuirani multiprocesor ili klaster. Svaka mašina ima sopstvenu memoriju i UI kanale
2. Funkcionalno specijalizovani procesori. Primer je UI procesor
3. Čvrsto spregnuti multiprocesor. Sastoji se od skupa procesora koji dele zajedničku memoriju i nalaze se pod integrisanim upravljanjem OS-a

Bavimo se poslednjom kategorijom, a posebno pitanjima koja utiču na kratkoročno raspoređivanje procesa (*scheduling*).

Granularnost

1. Nezavisni paralelizam - Nema eksplicitne sinhronizacije među procesima. Svaki proces predstavlja nezavisnu aplikaciju ili posao. Isto se može postići stavljanjem PC-a na raspolaganje svakom korisniku. Deljeni sistem može biti isplativiji od distribuiranog sistema.
2. Paralelizam vrlo grube i grube granulacije - Postoji sinhronizacija, ali na vrlo opštem planu. Primer je `make -j` opcija `GNU Make` alata. Ako je sinhronizacija retka, distribuirani sistem je u redu, a ako su sinhronizacije češće, gubi se vreme na režiju na mreži.
3. Paralelizam srednje granilacije - Paralelizam eksplicitno uređuje programer. Odluke o raspoređivanju jedne niti utiču na performanse cele aplikacije!
4. Paralelizam fine granilacije - Još uvek u fazi istraživanja

Granularnost
Nastavak

Table 10.1 Synchronization Granularity and Processes

Grain Size	Description	Synchronization Interval (Instructions)
Fine	Parallelism inherent in a single instruction stream.	<20
Medium	Parallel processing or multitasking within a single application	20–200
Coarse	Multiprocessing of concurrent processes in a multiprogramming environment	200–2000
Very Coarse	Distributed processing across network nodes to form a single computing environment	2000–1M
Independent	Multiple unrelated processes	not applicable

Pitanja projektovanja

1. Dodeljivanje procesa procesorima
2. Upotreba multiprogramiranja na pojedinačnim procesorima
3. Stvarno raspoređivanje procesa

Dodeljivanje procesa procesorima

- Statičko dodeljivanje - trajna dodela procesa procesoru od početka do završetka. Namenski red čekanja za svaki procesor. **Prednost** je što ima manje režije u zameni procesa, a **mana** je što procesor može da bude besposlen iako u redu pred drugim procesorom ima procesa koji čekaju. Može se donekle reešiti zajedničkim redom čekanja.
- Dinamičko dodeljivanje - Niti se mogu pomerati između redova čekanja radi uravnotežavanja opterećenja. `Linux` koristi ovakav pristup.

Modeli *scheduling*-a

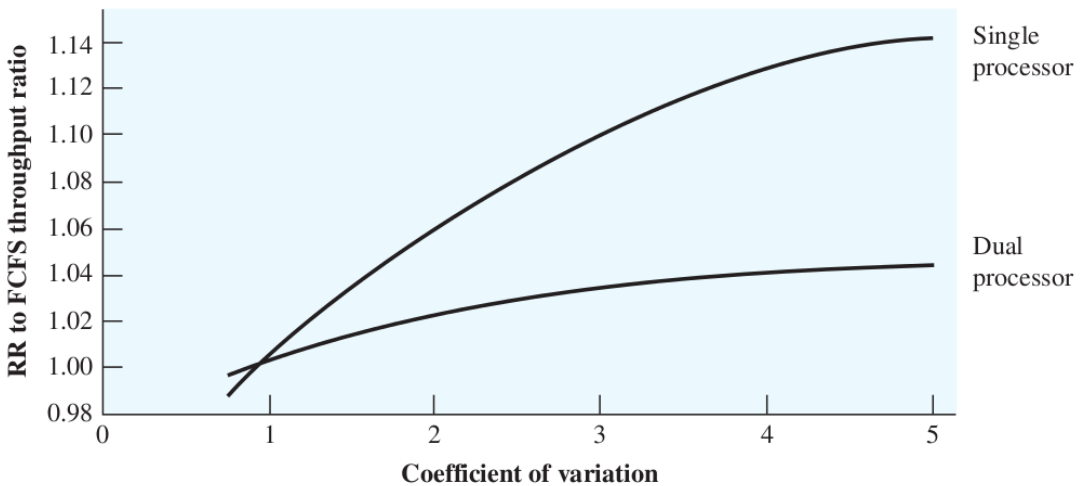
- Master-slave - na jednom procesoru se izvršavaju kernel i scheduler, a na ostalima korisnički procesi
- Ravnopravnost - svi procesi mogu da izvršavaju i korisnički i kernel/scheduler kôd. Mora da se obezbedi *mutex* kontrola.

Upotreba multiprogramiranja na pojedinačnim procesorima

- Kada se proces statički dodeli procesoru u toku celog životnog veka, postavlja se pitanje da li bi uopšte trebalo da se multiprogramira?
- Za aplikacije grube granularnosti logično je da procesor može da komutira između izvesnog broja procesa radi boljeg iskorišćenja
- Za aplikacije srednje granularnosti kada je na raspolaganju mnogo procesora, više nije vrhunski cilj da svaki pojedinačni procesor bude što zaposleniji, već da se obezbedi isključivo bolja performansa za aplikacije!
- Višenitna aplikacija može slabo da radi ako joj sve niti nisu raspoložive za istovremeno izvršavanje!

Raspoređivanje procesa

- U jednoprocesorskom sistemu, upotreba algoritama prioritetia i onih zasnovani na istoriji upotrebe može značajno da poveća performanse
- U višeprocesorkom sistemu, takve složenosti mogu biti nepotrebne ili čak kontraproduktivne
- U sledećem primeru poredi se FCFS sa *Round Robin* strategijom. Na apscisi je varijacija vremena opsluživanja (*service time*)



(a) Comparison of RR and FCFS

Raspoređivanje niti

- Izvršavaju se konkurentno u istom adresnom prostoru
- Puna snaga niti je očigledna u multiprocesorskom sistemu
- Kod paralelizma srednje granulacije, vrlo je bitan način upravljanja nitima!

Opšti pristupi raspoređivanju niti

1. Deljenje opterećenja - svaki procesor kada je besposlen bira nit iz zajedničkog reda
2. Grupno raspoređivanje - Skup povezanih niti se u isto vreme raspoređuje na više procesora
3. Namenska dodela procesora - procesu se dodeljuje skup procesora jednak njegovom broju niti
4. Dinamičko raspoređivanje - broj niti u procesu menja se u toku izvršenja procesa

Deljenje opterećenja

Prednosti

1. Ravnomerna raspodela opterećenja,
2. *Raspoređivač* se izvršava na trenutno slobodnom procesoru,
3. Može da se koristi bilo koja šema iz *jednoprocesorskog raspoređivanja*-a, recimo FCFS, proces sa najmanje niti prvi, prekidni proces sa najmanje niti prvi
4. Zanimljivo, simulacija pokazuje da FCFS daje najbolje performanse

Mane

1. Pristup centralnom redu čekanja mora da se sprovede preko *mutex*-a, što u sistemima sa mnogo procesora može da postane usko grlo
2. Nije verovanto da će se obrada prekinutih niti nastaviti na istom procesoru
3. Malo je verovatno da će grupa niti iz aplikacije dobiti procesore istovremeno

Grupno raspoređivanje

Prednosti

1. Blisko povezane niti se izvršavaju istovremeno
2. Režija se smanjuje jer jedna odluka utiče na više procesa/niti
3. Komutacije procesa su minimizovane

Mane

1. Moguće je da neki procesori ostanu besposleni iako ima procesa koji čekaju u redu. Smanjuje se dodlelom procesora po težini određenoj brojem niti

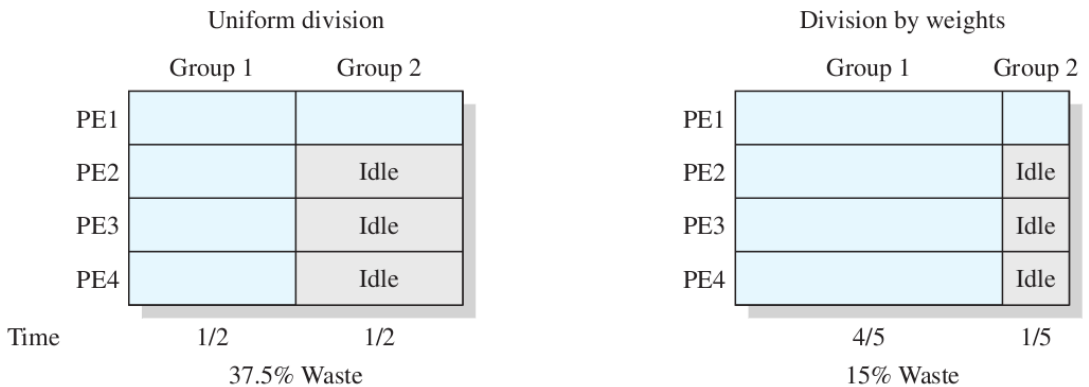


Figure 10.3 **Example of Scheduling Groups with Four and One Threads [FEIT90b]**

Namensko dodeljivanje procesora

1. Ekstremni oblik grupnog raspoređivanja, procesori ostaju dodeljeni aplikaciji dok njeno izvršenje traje
2. Krajnje rasipnički pristup ako ima malo procesora, ali ako ih ima dosta povećava performanse
3. Recimo, ako proces čeka na UI, procesor ostaje besposlen
4. Na slici je primer konkurentnog izvođenja niti za FFT i množenje matrica na sistemu sa 16 procesora
5. Broj niti ne treba da prelazi broj procesora u sistemu!

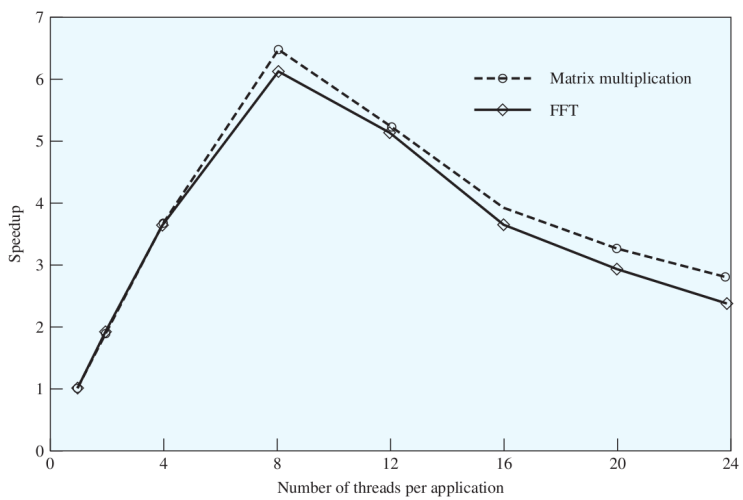


Figure 10.4 **Application Speedup as a Function of Number of Threads [TUCK89]**

Dinamičko raspoređivanje

- I OS i aplikacija su uključeni u donošenje odluke o raspoređivanju procesa
- OS se brine o raspoređivanju procesa
- Aplikacija brine (npr. pomoću bibliotečkih rutina) koji skup niti će se trenutno izvršavati
- Prednosti ovakvog pristupa može da poništi cena režije i kompleksnost

Raspoređivanje u realnom vremenu

Definicija (Računarstvo u realnom vremenu)

Tačnost rezultata ne zavisi samo od logičkog rezultata računanja, već i od vremena u kome je rezultat proizveden.

- Sve značajnija disciplina, sistemi za upravljanje, robotika
- Neki od zadataka imaju izvestan stepen hitnosti

Vrste zadataka u realnom vremenu

1. Čvrst zadatak - rok mora da se ispoštuje
2. Labav zadatak - ispunjenje roka je poželjno
3. Aperiodičan zadatak
4. Periodičan zadatak

Osobine *real-time* sistema

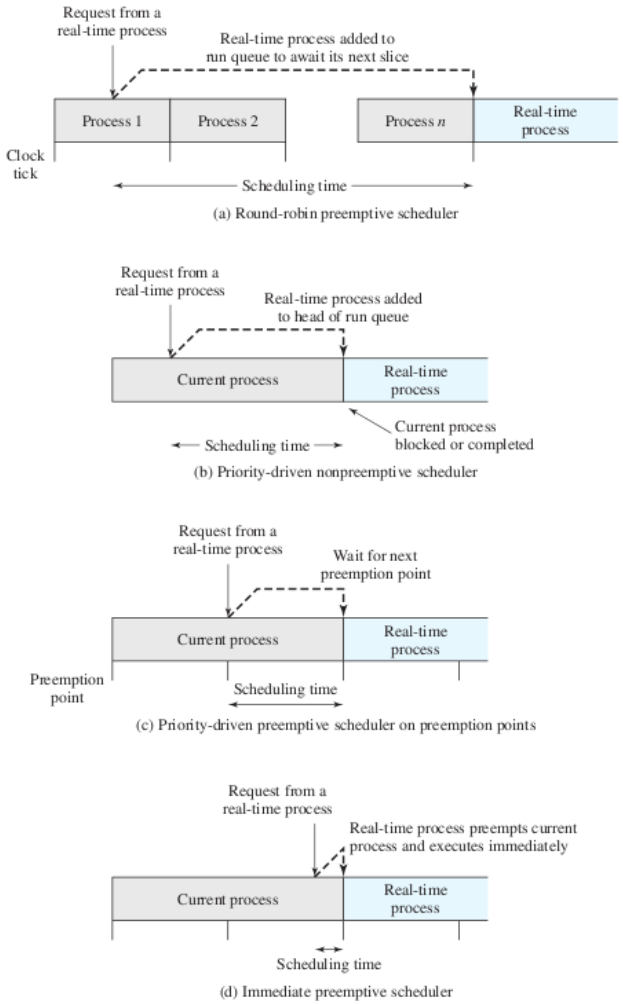
1. Determinizam - Mera ove sposobnosti je vremenski interval od pojave prekida sa uređaja visokog prioriteta do početka opsluživanja. Mora da bude od nekoliko μs do najviše $1ms$
2. Odzivnost - Koliko je dugo potrebno OS-u da posle potvrđivanja prekida potrebno da izvrši ISR.
   - inicijalizacija
   - Izvršenje ISR
   - Efekat ugnežđenja prekida
3. Korisničko upravljanje - Pravljenje razlike između čvrstih i labavih zadataka i određivanje prioriteta
4. Pouzdanost - Smanjenje nivoa usluge, ali ne potpun otkaz
5. Operacija postepenog otkaza - Održati konzistentnost po svaku cenu

Standardna svojstva *real-time* sistema

1. Brza komutacija procesa i niti
2. Minimalna veličina
3. Brz odziv na spoljne prekide
4. Istovremena obrada više zadataka pomoću semafora, signala i događaja
5. Upotreba specijalnih sekvencijalnih fajlova za akumulaciju podataka velikom brzinom
6. Raspoređivanje sa prekidanjem zasnovano na prioritetnoj šemi
7. Minimizacija intervala u kome su prekidi onemogućeni
8. Specijalni alarmi i pauze

Kratkoročni *scheduler* sistema u realnom vremenu

Cilj je ispoštovati rokove čvrstih zadataka i da prođe što više labavih zadataka.



Kratkoročni *scheduler* sistema u realnom vremenu

Pregled metoda statičkog pristupa

1. Statički pristupi pomoću tabela - Primenljivo na periodične zadatke. Razvija se raspored koji omogućava izvršenje svih zadataka. Jedna od metoda je *posao sa najranijim rokom prvi*. Predvidiva, ali nefleksibilna metoda
2. Statičko prekidno raspoređivanje na osnovu prioriteta - Kod OS-eva koji ne rade u realnom vremenu, prioriteti se određuju na osnovu količine UI itd. U Real-Time OS-evima prioriteti se povezuju s vremenskim ograničenjima. Jedan je *RMS*

Kratkoročni *scheduler* sistema u realnom vremenu

Pregled metoda dinamičkog pristupa

1. Dinamičko raspoređivanje zasnovano na planiranju - Kada posao stigne, pokušava se da se napravi raspored koji poštuje sve rokove. Za aperiodične zadatke. Fleksibilan.
2. Dinamičko raspoređivanje "Najbolje što može" - Mnogi današnji OS-evi ga koriste. OS mu dâ prioritet na osnovu karakteristika. Postoji mogućnost promašaja, ali vrlo fleksibilno za aperiodične zadatke.

Raspoređivanje na osnovu roka
Informacije o zadacima

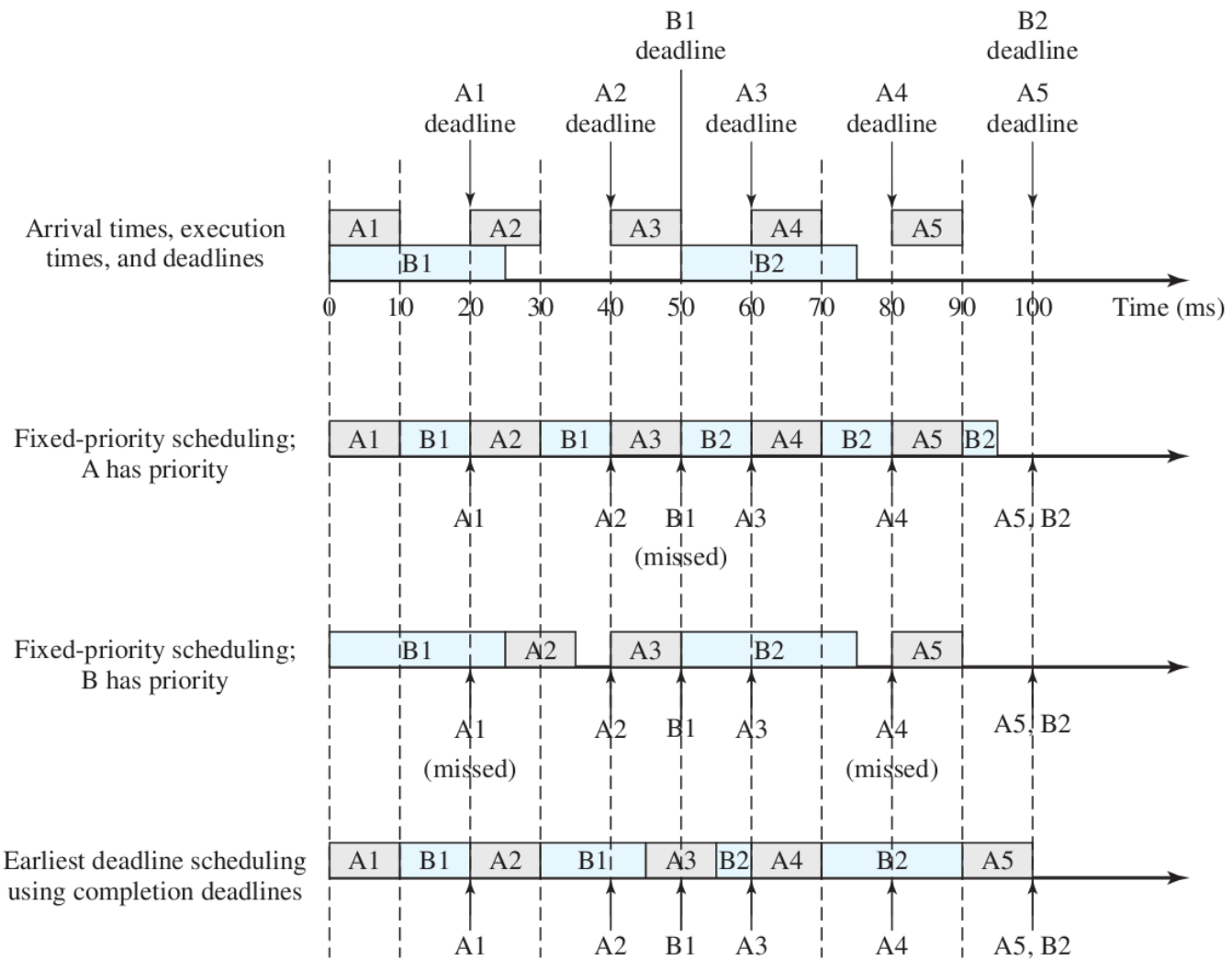
Aplikacije u realnom vremenu

Aplikacije u realnom vremenu se ne bave čistom brzinom, već pre započinjanjem i/ili završavanjem poslova u određenim rokovima.

1. Vreme spremnosti
2. Rok započinjanja
3. Rok završetka
4. Vreme obrade (dato ili eksponencijalna sredina)
5. Zahtevi za resursima
6. Prioritet
7. Struktura podzadatka

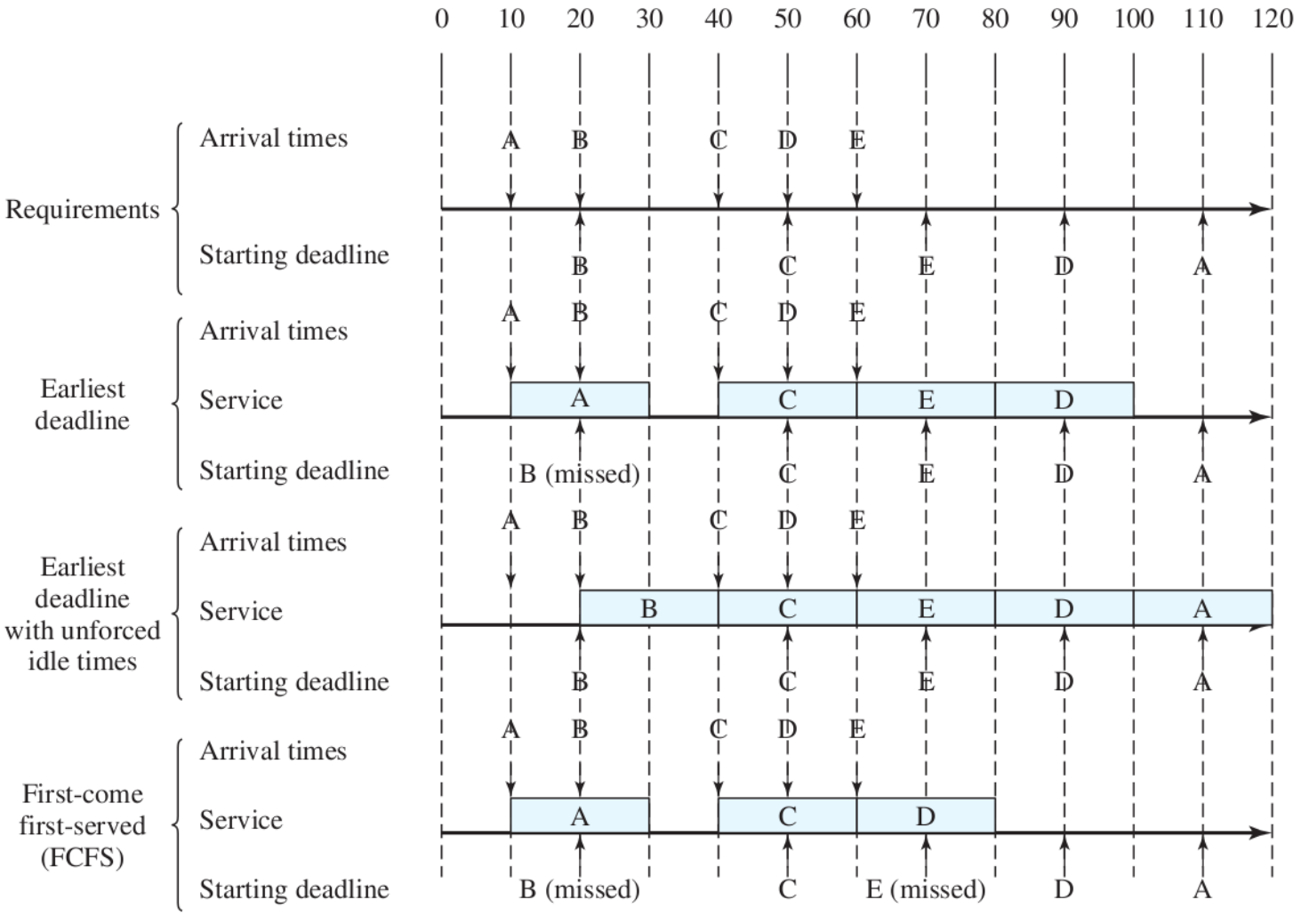
Raspoređivanje sa rokovima završetaka

Upotreba prekidnog *scheduler*-a, periodični zadaci



Raspoređivanje sa rokovima započinjanja

Upotreba neprekidnog *scheduler*-a, aperiodični zadaci



Raspoređivanje monotonom učestanošću

Rate monotonic scheduling-RMS

Veličine

- T - period zadatka, vreme između sdva dolaska
- C - vreme izvršenja, vreme koje se zahteve za svako izvršenje zadatka. Mora biti $C \leq T$
- $U = \frac{C}{T}$ - iskorišćenje procesora od strane zadatka

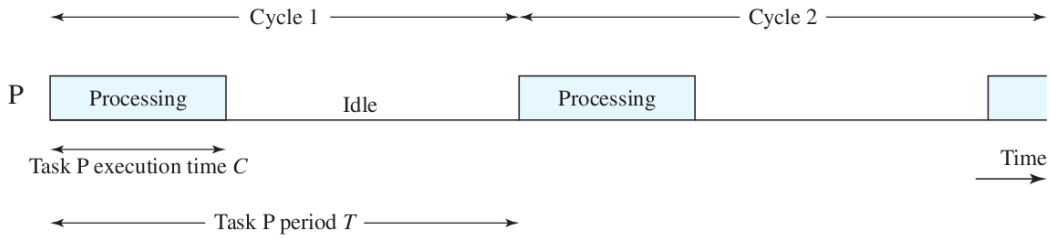
$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \leq 1$$

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \leq n(2^{\frac{1}{n}} - 1), \quad \text{za RMS}$$

Za RMS, zadatak sa najvećim prioritetom je onaj sa najkraćim periodom, zatim onaj sa sledećim najkraćim periodom itd.

Raspoređivanje monotonom učestanošću

Rate monotonic scheduling-RMS



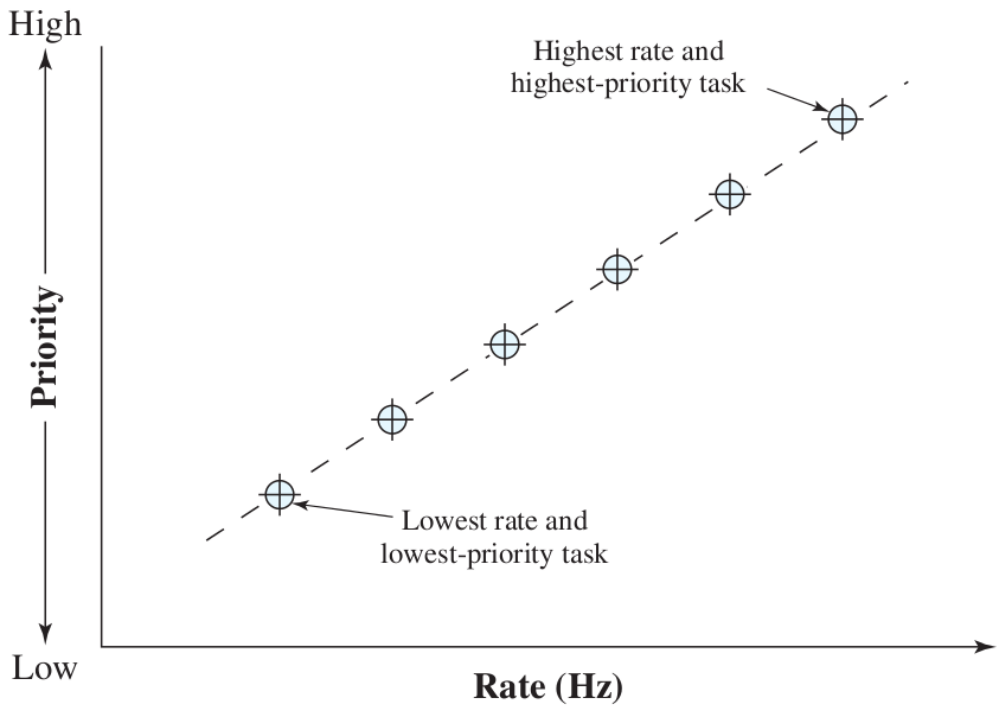


Table 10.4 Value of the RMS Upper Bound

n	$n(2^{1/n} - 1)$
1	1.0
2	0.828
3	0.779
4	0.756
5	0.743
6	0.734
•	•
•	•
•	•
∞	$\ln 2 \approx 0.693$

Raspoređivanje monotonom frekvencijom
Primer

Primer

- Zadatak P_1 : $C_1 = 20;\ T_1 = 100;\ U_1 = 0.2$
- Zadatak P_2 : $C_2 = 40;\ T_2 = 150;\ U_2 = 0.267$
- Zadatak P_3 : $C_3 = 100;\ T_3 = 350;\ U_3 = 0.286$

Ukupno vreme iskorišćenja: $0.2 + 0.267 + 0.286 = 0.753$.
Gornja granica za RMS: $\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} \leq 3(2^{\frac{1}{3}} - 1) = 0.779$

Može se pokazati da gornja granica jednačine $\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \leq 1$ važi za politiku raspoređivanja po najranijem roku. Ipak, prednosti RMS su:

1. Mala razlika u performansama
2. Apsorbovanje CPU vremena od strane ne *real-time* zadataka
3. Skalabilnost je bolja sa RMS

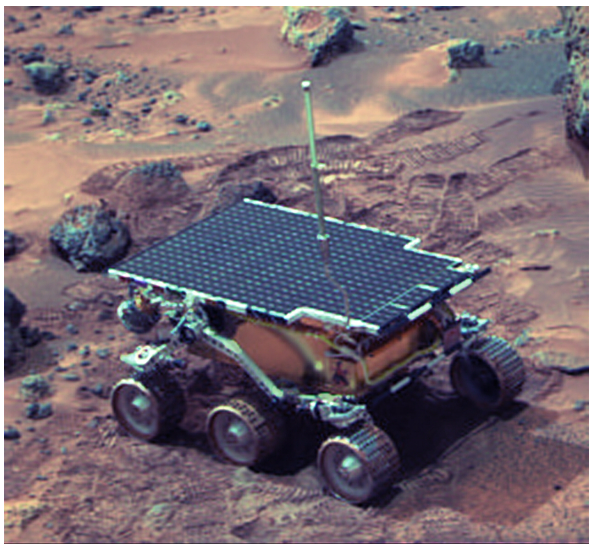
Inverzija prioriteta

Definicija (Inverzija prioriteta)

Pojava kada okolnosti unutar sistema prisiljavaju zadatak višeg prioriteta da čeka zadatak nižeg prioriteta.

Definicija (Neograničena inverzija prioriteta)

Pojava u kojoj trajanje inverzije prioriteta zavisi ne samo od vremena zahtevanog za rukovanje reljenim resursom, već i od nekih drugih spoljnih okolnosti.



Inverzija prioriteta

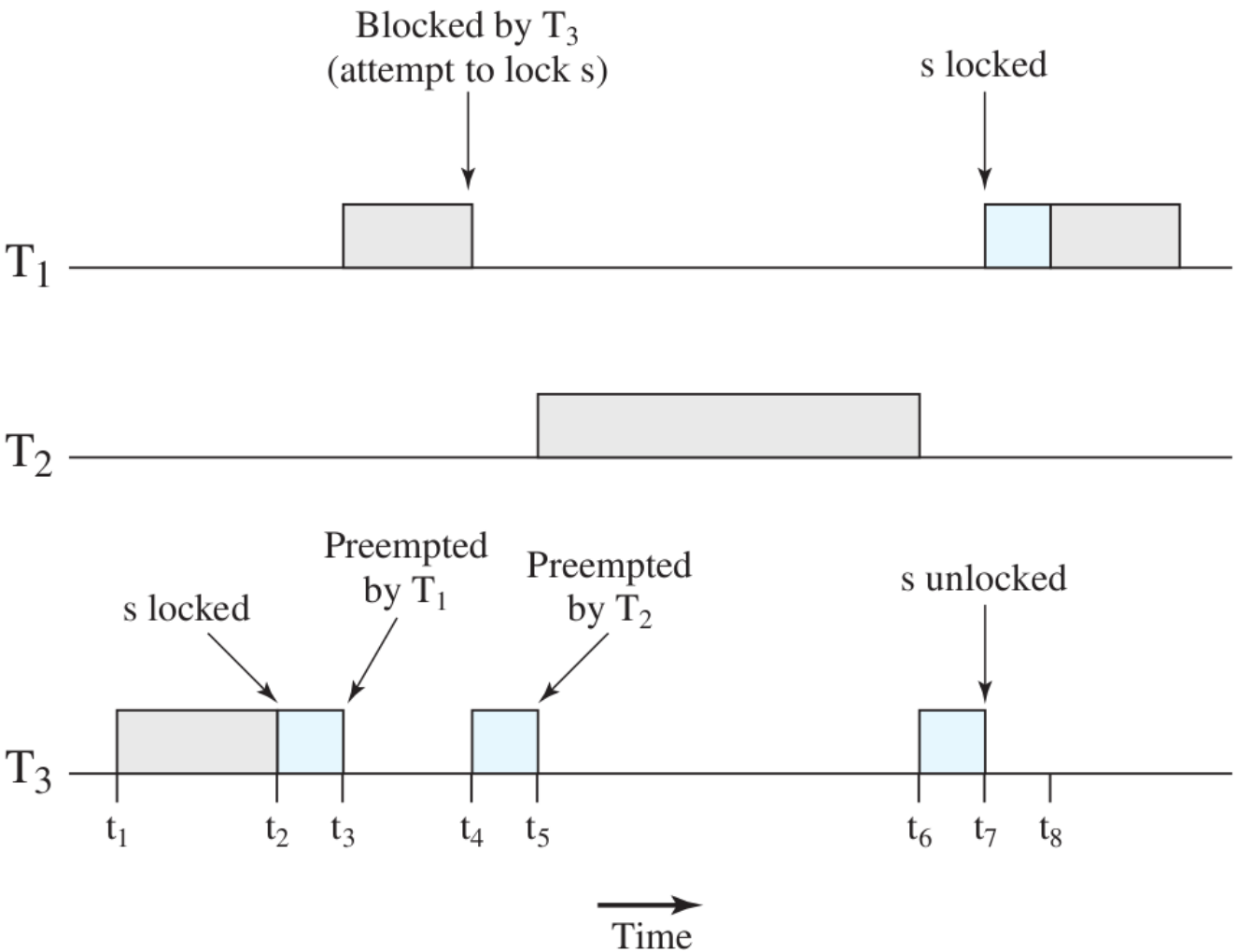
Real-time zadaci Pathfinder-a sa prioritetima

- T_1 - periodično proverava stanje sistema i softvera *Pathfinder*-a
- T_2 - obrađuje podatke slike
- T_3 - izvodi povremena ispitivanja stanja opreme

U slučaju da zadatak T_1 predugo potraje, *Pathfinder*-ov računar se resetuje i reinicijalizuje sve sisteme. Upravo to se i desilo zbog situacije da je T_1 pokušava da zaključa semafor koji je već zaključao T_3.

Inverzija prioriteta

Šta se desilo na Marsu?



Inverzija prioriteta

Rešenja: nasleđivanje prioriteta i vrhunski prioritet

Definicija (Nasleđivanje prioriteta)

Zadatak nižeg prioriteta nasleđuje prioritet zadatka višeg prioritetom koji traži resurs koji oni dele.