Planiranje u stvarnom vremenu

Klasifikacije višeprocesorskih sustava

Slabo spojeni ili distribuirani višeprocesorski sustavi

- Sastoje se od zbirke relativno autonomnih sustava, pri čemu svaki procesor ima svoju glavnu memoriju i U/I kanale

Funkcionalno specijalizirani procesori

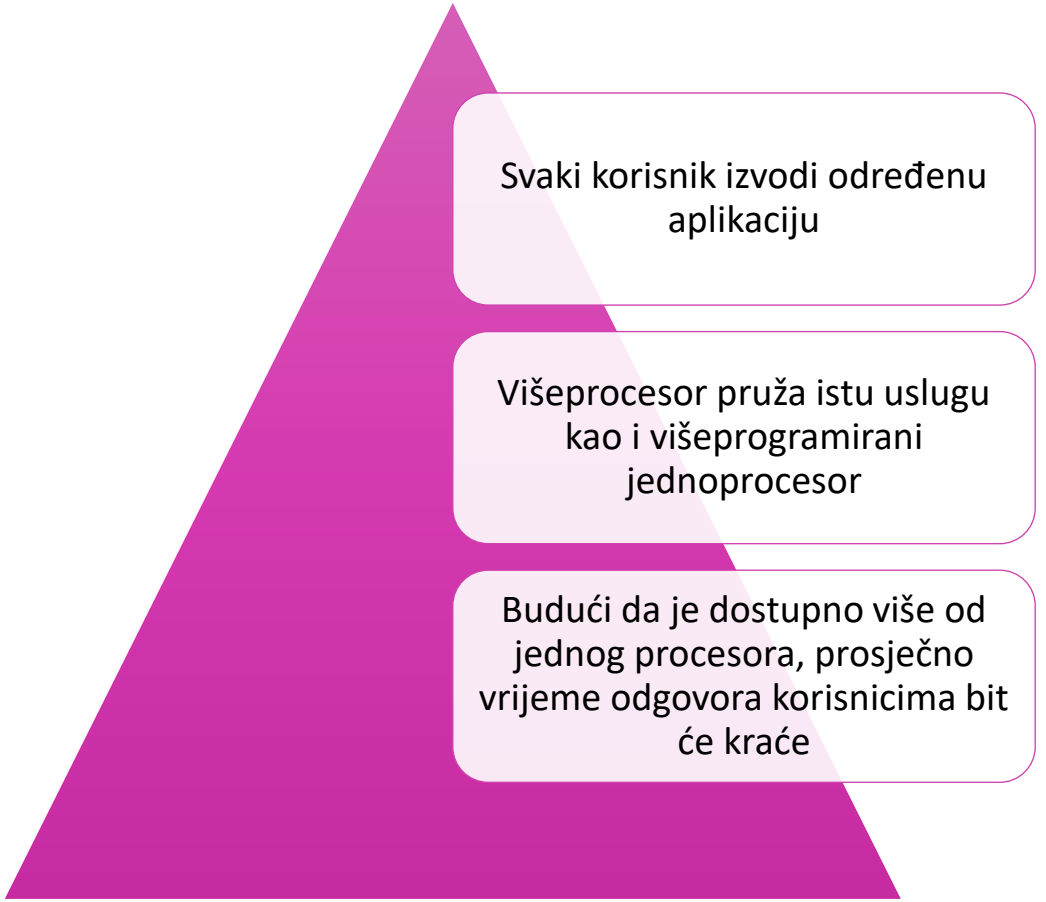
- Postoji glavni procesor opće namjene
- Specijalizirani procesori su pod kontrolom glavnog procesora

Čvrsto spojeni višeprocesorski sustavi

- Sastoji se od skupa procesora koji dijele zajedničku glavnu memoriju i koji su pod integriranom kontrolom operacijskog sustava

Nezavisni paralelizam

- Nema eksplicitne sinkronizacije među procesima
  - Svaki predstavlja zasebni, neovisni posao
  - Tipično se koristi u sustavima dijeljenja vremena



Grubo i vrlo grubo granularni paralelizam

- Postoji sinkronizacija među procesima, ali na vrlo gruboj razini
- Lako se obrađuje kao skup istodobnih procesa koji se izvode na višeprogramiranom jednoprocesoru
- Mogu se izvršavati na višeprocesoru s malo ili bez promjene korisničkog softvera

Srednje granularni paralelizam

- Jedna aplikacija može se učinkovito implementirati kao zbirka dretvi unutar jednog procesa
  - Programer mora eksplicitno specificirati potencijalni paralelizam aplikacije
  - Mora postojati visok stupanj koordinacije i interakcije među dretvama aplikacije, što dovodi do srednje razine sinkronizacije
- Budući da različite dretve aplikacije međusobno često komuniciraju, odluke o rasporedu koje se odnose na jednu dretvu mogu utjecati na izvedbu cijele aplikacije

Fino granularni paralelizam

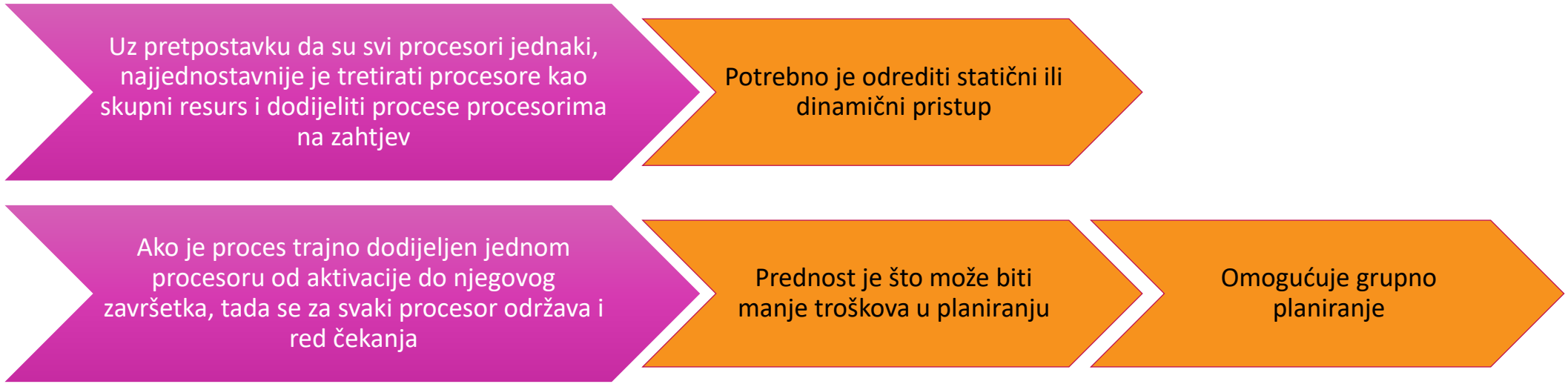
- Predstavlja mnogo složeniju upotrebu paralelizma nego što se nalazi u upotrebi dretvi
- To je specijalizirano područje s mnogo različitih pristupa

Problemi s dizajnom

- Poduzeti pristup ovisit će o stupnju granularnosti aplikacija i broju dostupnih procesora

Dodjela procesa procesorima

- Nedostatak statičke dodjele je što jedan procesor može biti u stanju mirovanja, s praznim redom čekanja, dok drugi procesor ima zaostatak
- Kako bi se spriječila ova situacija, može se koristiti uobičajeni red čekanja
- Druga opcija je dinamičko balansiranje opterećenja



Dodjela procesa procesorima

- I dinamičke i statičke metode zahtijevaju neki način dodjeljivanja procesa procesoru
- Pristupi:
  - Master/Slave
  - Peer

Master/Slave arhitektura

- Ključne funkcije kernela uvijek rade na određenom procesoru
- Master je odgovoran za raspored
- Slave šalje zahtjev za uslugu masteru
- Jednostavan je i zahtijeva malo poboljšanja jednoprocesorskog operacijskog sustava za više programa
- Rješavanje sukoba je pojednostavljeno jer jedan procesor ima kontrolu nad svim memorijskim i U/I resursima

Nedostaci:

- Prestanak rada Mastera ruši cijeli sustav
- Mastermože postati usko grlo

Peer arhitektura

- Kernel se može izvršiti na bilo kojem procesoru
- Svaki procesor vrši dodjeljivanje iz skupa dostupnih procesa

Komplicira operacijski sustav
• Operacijski sustav mora osigurati da dva procesora ne odaberu isti proces i da se procesi na neki način ne izgube iz reda čekanja

Planiranje procesa

- U većini tradicionalnih višeprocesorskih sustava procesi nisu dodjeljeni procesorima
- Za sve procesore koristi se jedan red čekanja
  - Ako se koristi neka vrsta prioritetne sheme, postoji više redova koji se temelje na prioritetu, a svi se unose u zajednički skup procesora
- Sustav se promatra kao arhitektura čekanja na više poslužitelja

Planiranje dretvi

- Izvršavanje dretvi je odvojeno od ostatka definicije procesa
- Aplikacija može biti skup dretvi koje surađuju i izvršavaju se istodobno u istom adresnom prostoru
- Na jednoprocesoru, dretve se mogu koristiti kao pomoć pri strukturiranju programa i za preklapanje u obradi U/I zahtjeva
- U višeprocesorskom sustavu dretve se mogu koristiti za iskorištavanje istinskog paralelizma u aplikaciji
- Dobici u performansama mogući su u višeprocesorskim sustavima
- Razlike u načinu upravljanja dretvama i rasporedu mogu imati utjecaj na aplikacije koje zahtijevaju učestalu interakciju

Pristupi planiranju dretvama

Dijeljenje opterećenja

- Najjednostavniji pristup i onaj koji se najviše prenosi iz jednoprocesorskog okruženja

Prednosti:
• Opterećenje se ravnomjerno raspoređuje na procesore, osiguravajući da nijedan procesor nije u stanju mirovanja dok ima posla • Nije potreban centralizirani planer

- Verzije dijeljenja opterećenja:
  - First-come-first-served (FCFS)
  - Prvo najmanji broj dretvi
  - Prekidno prvo najmanji broj dretvi

Nedostaci dijeljenja opterećenja

- Središnji red čekanja zauzima područje memorije kojem se mora pristupiti na način koji provodi međusobno isključivanje
  - Može dovesti do uskog grla
- Malo je vjerojatno da će prekinute dretve nastaviti s izvršavanjem na istom procesoru
  - Keširanje može postati manje učinkovito
- Ako se sve dretve tretiraju kao zajednički skup, malo je vjerojatno da će sve dretve programa dobiti pristup procesorima u isto vrijeme
  - Uključeni procesni prekidači mogu ozbiljno ugroziti performanse

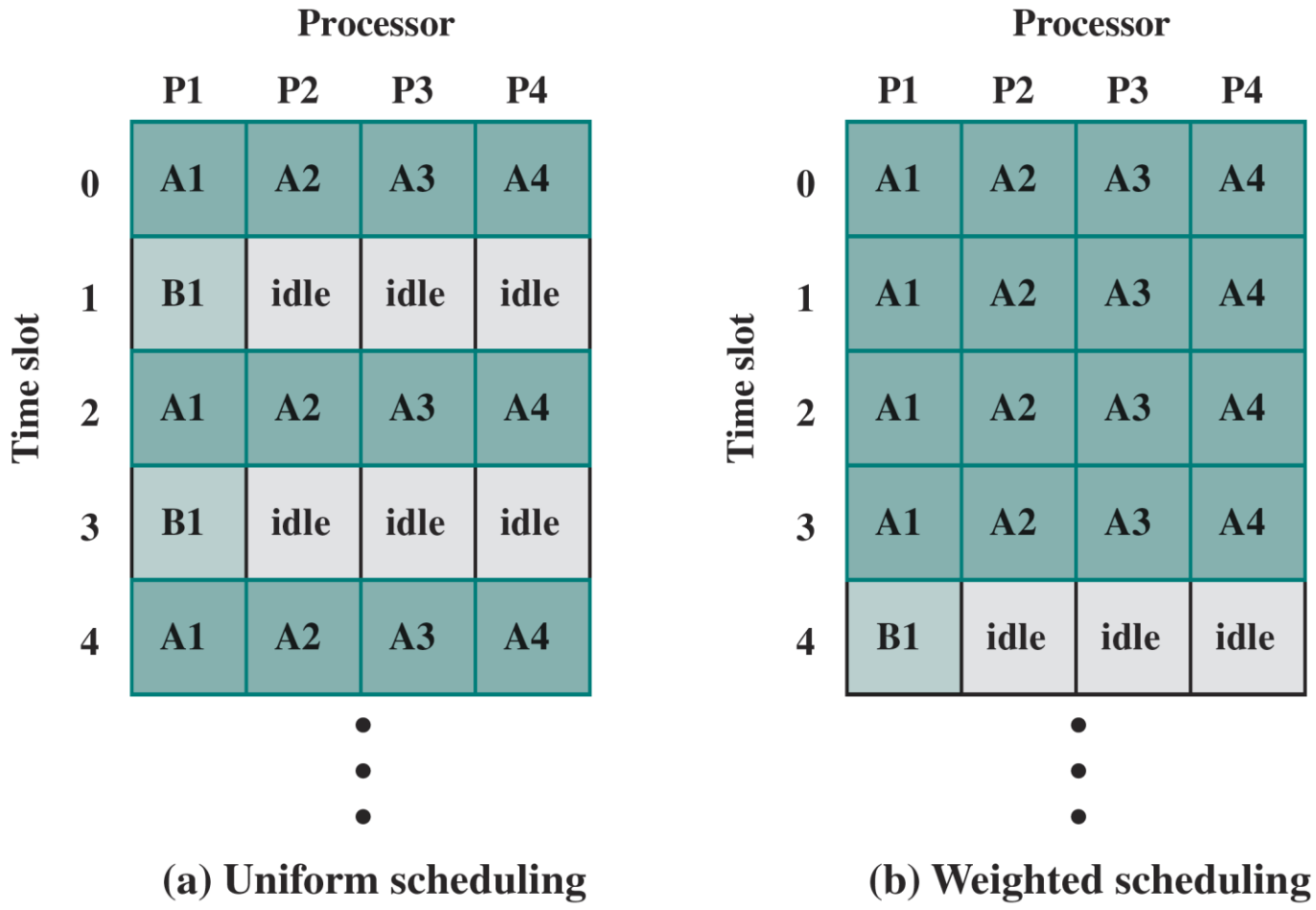
Grupno planiranje

- Istovremeno raspoređivanje dretvi koje čine jedan proces

Prednosti:
• Blokiranje radi sinkronizacije može se smanjiti, možda će biti potrebno manje prebacivanja procesa, a performanse će se povećati • Troškovi planiranja mogu se smanjiti

- Korisno za srednje granulirano i fino granulirano paralelne aplikacije čija se izvedba ozbiljno pogoršava kada bilo koji dio aplikacije nije pokrenut dok su drugi dijelovi spremni za rad
- Također je korisno za svaku paralelnu primjenu

Grupno planiranje



(a) Uniform scheduling

(b) Weighted scheduling

Dodjela namjenskog procesora

- Kada je aplikacija pokrenuta, svaku njenu dretvu dodjeljuje se procesoru koji joj ostaje dodjeljen dok se aplikacija ne završi
- Ako je dretva aplikacije blokirana čekajući U/I ili sinkronizaciju s drugom dretvom, tada procesor te dretve ostaje neaktivan
  - Nema multiprogramiranja procesora
- Prednosti strategije:
- U vrlo paralelnom sustavu, s desecima ili stotinama procesora, iskorištenost procesora više nije toliko važna kao metrika za učinkovitost ili performanse
- Potpuno izbjegavanje promjene procesa tijekom životnog vijeka programa trebalo bi rezultirati značajnim ubrzanjem tog programa

Dinamičko planiranje

- Za neke aplikacije moguće je osigurati jezične i sistemske alate koji dopuštaju da se broj dretvi u procesu dinamički mijenja
  - To bi omogućilo operacijskom sustavu da prilagodi opterećenje kako bi se poboljšala iskorištenost
- I operacijski sustav i aplikacija sudjeluju u donošenju odluka o rasporedu
  - Odgovornost za raspoređivanje operacijskog sustava prvenstveno je ograničena na dodjelu procesora
  - Ovaj pristup je superiorniji od grupnog planiranja ili dodjele namjenskog procesora za aplikacije koje ga znaju iskoristiti

Dijeljenje predmemorije

Kooperativno dijeljenje resursa

- Više dretvi pristupa istom skupu lokacija glavne memorije
- Primjeri:
- Aplikacije koje su višedretvene
- Interakcija dretvi kod proizvođača/potrošač sustava

Nadmetanje za resurse

- Dretve, ako rade na susjednim jezgrama, natječu se za pričuvnu memoriju
- Ako je više predmemorije dinamički dodijeljeno jednoj dretvi, druga nužno ima manje dostupnog prostora i stoga trpi degradaciju performansi
- Cilj planiranja svjesnog sukoba je dodijeliti dretvu jezgrama kako bi se povećala učinkovitost dijeljene pred-memorije i minimizirala potreba za pristupima glavnoj memoriji

Sustavi u stvarnom vremenu

- Operacijski sustav, a posebno planer, možda je najvažnija komponenta

Primjeri:

- Kontrola laboratorijskih pokusa
- Kontrola procesa u industrijskim postrojenjima
- Robotika
- Kontrola zračnog prometa
- Telekomunikacija
- Sustavi vojnog zapovijedanja

- Ispravnost sustava ovisi ne samo o logičnom rezultatu izračuna, već i o vremenu u kojem se rezultati proizvode
- Zadaci ili procesi pokušavaju kontrolirati ili reagirati na događaje koji se događaju u vanjskom svijetu
- Događaje u "stvarnom vremenu" moraja pratiti njihovo izvršavanja

“Tvrdi” i “meki” zadaci u stvarnom vremenu

Tvrdi zadaci

- Oni koji moraju ispoštovati svoj rok
- Inače će uzrokovati greškui u sustavu

Meki zadaci

- Ima pridruženi rok koji je poželjan, ali nije obavezan
- I dalje ima smisla planirati i dovršiti zadatak čak i ako je rok prošao

Periodični i neperiodični zadaci

- Periodični zadaci
  - Zahtjev se može navesti kao:
    - Jednom po priodu T
    - Točno T jedinica do ponovnog izvršenja
- Neperiodični zadaci
  - Ima rok do kojeg mora završiti ili započeti
  - Može imati ograničenje na vrijeme početka i završetka

Karakteristike sustava u stvarnom vremenu

Operacijski sustavi u stvarnom vremenu imaju zahtjeve u pet područja:

- Determinizam
- Odaziv
- Kontrola korisnika
- Pouzdanost
- Otpornost na greške

Determinizam

- Koliko dugo operacijski sustav odgađa prije nego što potvrdi prekid
- Operacije se izvode u fiksnim, unaprijed određenim vremenima ili u unaprijed određenim vremenskim intervalima
- Kada se više procesa natječe za resurse i vrijeme procesora, nijedan sustav neće biti potpuno deterministički

Odaziv

- Zajedno s determinizmom čine vrijeme odgovora na vanjske događaje
  - Kritično za sustave u stvarnom vremenu koji moraju zadovoljiti zahtjeve koje nameću uređaji i tokovi podataka izvan sustava
- Zabrinutost o tome koliko dugo, nakon potvrde, treba operativnom sustavu da obradi prekid

Odaziv uključuje:

- Količina vremena potrebna za početnu obradu prekida i početak izvršavanja rutine usluge prekida (ISR)
- Količina vremena potrebna za izvođenje ISR-a
- Učinak gniježđenja prekida

Kontrola korisnika

- Mnogo šire u operaciskim sustavima u stvarnom vremenu nego u običnim
- Bitno je omogućiti korisniku finu kontrolu nad prioritetom zadatka
- Korisnik bi trebao biti sposoban razlikovati teške i meke zadatke i odrediti relativne prioritete unutar svake klase
- Može dopustiti korisniku da navede takve karakteristike kao što su:

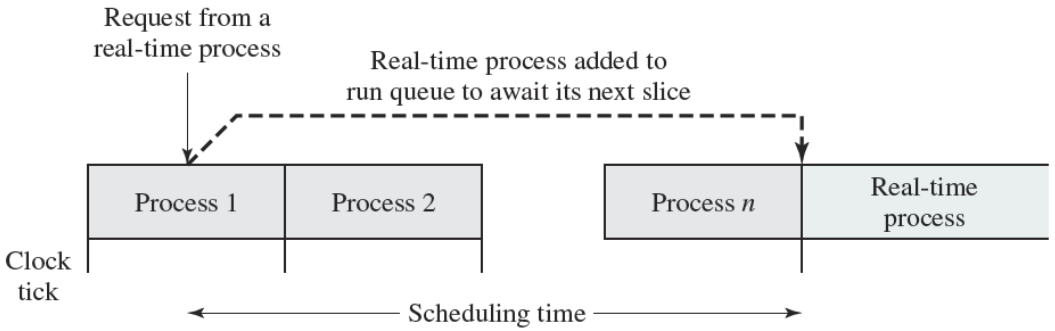
Pouzdanost

- Važnije za sustave u stvarnom vremenu od sustava koji nisu u stvarnom vremenu
- Sustavi u stvarnom vremenu reagiraju i kontroliraju događaje u stvarnom vremenu tako da gubitak ili degradacija performansi može imati katastrofalne posljedice kao što su:
  - Financijski gubitak
  - Velika oštećenja opreme
  - Gubitak života

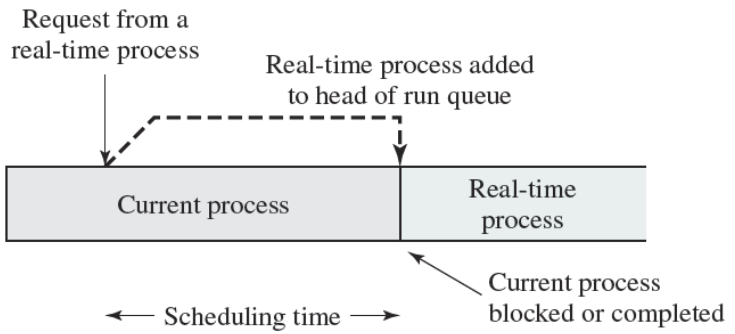
Otpornost na greške

- Karakteristika koja se odnosi na sposobnost sustava da kod pojave greške sačuva što je moguće više sposobnosti i podataka
- Važan aspekt je stabilnost
  - Sustav u stvarnom vremenu je stabilan ako sustav ispunjava rokove svojih najkritičnijih zadataka najvišeg prioriteta, čak i ako neki manje kritični rokovi nisu uvijek ispunjeni
- Sljedeće značajke zajedničke su većini OS-ova u stvarnom vremenu
  - Stroža uporaba prioriteta nego u običnom OS-u, s prekidnim planiranjem koje je dizajnirano da zadovolji zahtjeve u stvarnom vremenu
  - Latencija prekida je ograničena i relativno kratka
  - Preciznije i predvidljive vremenske karakteristike od OS opće namjene

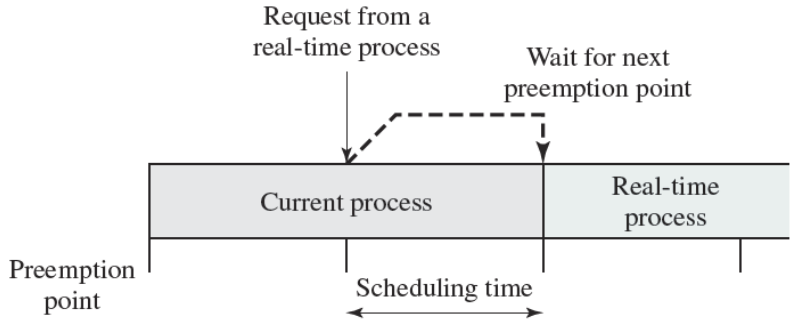
Planiranje procesa u stvarnom vremenu



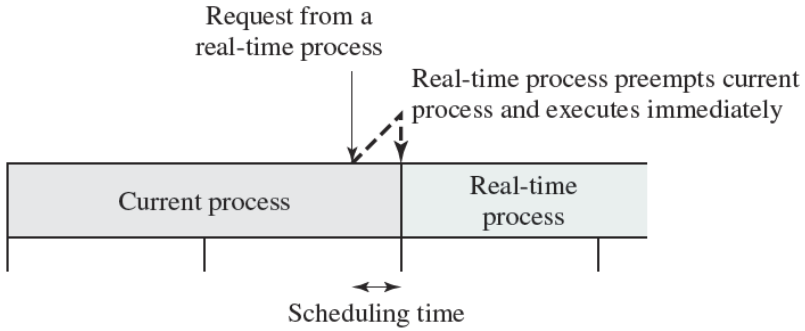
(a) Round-robin preemptive scheduler



(b) Priority-driven nonpreemptive scheduler

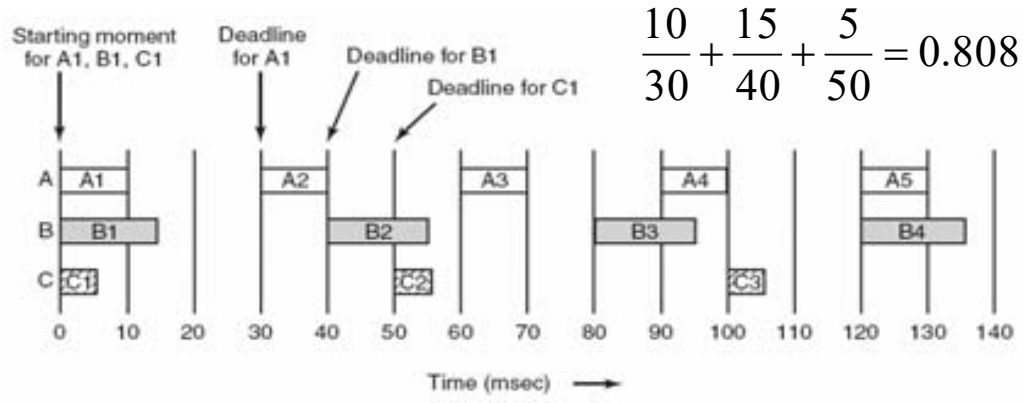


(c) Priority-driven preemptive scheduler on preemption points



(d) Immediate preemptive scheduler

Planiranje u stvarnom vremenu

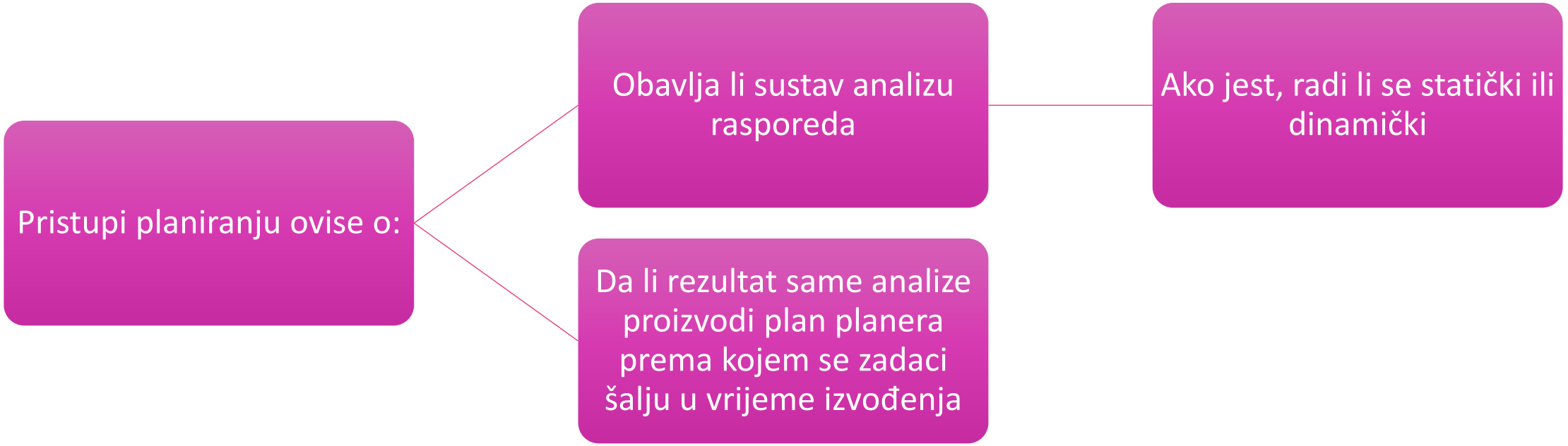


$$\frac{10}{30}+\frac{15}{40}+\frac{5}{50}=0.808\leq 1$$

- Tri procesa A, B i C
- Svaki radi svojim tempom
- Može li se sustav riješiti?
  - P*i* period procesa *i* (msec)
  - C*i* potrebno CPU vrijeme procesa *i* (msec)
  - m – brojevi procesa u sustavu
- JEDNADŽBA MORA VRIJEDITI DA BI SUSTAV RADIO!

$$\sum_{i=1}^{m}\frac{Ci}{Pi}\leq 1$$

Planiranje u stvarnom vremenu



Klase algoritama za planiranje u stvarnom vremenu

Pristupi vođeni statičkim tablicama

- Obavlja statičku analizu izvedivih rasporeda dodjele
- Rezultat je raspored koji određuje, u vrijeme izvođenja, kada se zadatak mora početi izvršavati

Prekidni pristupi vođeni statičnim prioritetom

- Provodi se statička analiza, ali se ne izrađuje raspored
- Analiza se koristi za dodjelu prioriteta zadacima tako da se može koristiti tradicionalni prekidni planer

Pristupi temeljeni na dinamičkom planiranju

- Izvedivost se utvrđuje tijekom izvođenja, a ne prije početka izvođenja
- Rezultat analize je plan koji se koristi za odlučivanje kada će se zadatak poslati na izvršenje

Dinamički pristupi *best-effort*

- Ne radi se analiza izvodljivosti
- Sustav pokušava ispuniti sve rokove i prekida svaki započeti proces čiji je rok propušten

Algoritmi planiranja

Pristupi vođeni statičkim tablicama

- Primjenjivo na zadatke koji su periodični
- Ulaz u analizu sastoji se od periodičnog vremena dolaska, vremena izvršenja, periodičnog roka završetka i relativnog prioriteta svakog zadatka
- Ovo je predvidljiv pristup, ali je nefleksibilan jer svaka promjena bilo kojeg zahtjeva zadatka zahtijeva da se raspored ponovi

Prekidni pristupi vođeni statičnim prioritetom

- Koristi mehanizam prioritetnog planiranja koji je uobičajen za većinu višeprogramskih sustava koji nisu u stvarnom vremenu
- U sustavu u stvarnom vremenu, dodjela prioriteta povezana je s vremenskim ograničenjima pridruženim svakom zadatku
- Jedan primjer ovog pristupa je monotoni algoritam brzine (RMS) koji dodjeljuje statičke prioritete zadacima na temelju duljine njihovih perioda

Algoritmi planiranja

Pristupi temeljeni na dinamičkom planiranju

- Nakon što zadatak stigne, ali prije početka njegovog izvršavanja, pokušava se stvoriti raspored koji sadrži prethodno zakazane zadatke kao i novi
- Ako se novi može zakazati na način da su njegovi rokovi zadovoljeni i da nijedan trenutno zakazani zadatak ne propusti rok, tada se raspored revidira kako bi se prilagodio novom zadatku

Dinamički pristupi *best-effort*

- Pristup koji koriste mnogi sustavi u stvarnom vremenu koji su trenutno komercijalno dostupni
- Kada stigne zadatak, sustav dodjeljuje prioritet na temelju karakteristika zadatka
- Obično se koristi neki oblik rasporeda rokova
- Zadaci obično nisu periodični tako da nije moguća statička analiza rasporeda
- Glavni nedostatak ovog oblika rasporeda je taj što dok ne stigne rok ili dok se zadatak ne završi, ne znamo hoće li se ispuniti vremensko ograničenje
- Njegova prednost je što ga je lako implementirati

Planiranje prema rokovima

- Operativni sustavi u stvarnom vremenu dizajnirani su s ciljem pokretanja zadataka što je brže moguće i naglašavaju brzo rukovanje prekidima i slanje zadataka
- Aplikacije u stvarnom vremenu općenito se ne bave samo brzinom, već dovršavanjem (ili pokretanjem) zadataka u određeno vrijeme
- Prioriteti su grubi alat i ne obuhvaćaju zahtjev za dovršetak (ili pokretanje) u određenom trenutku

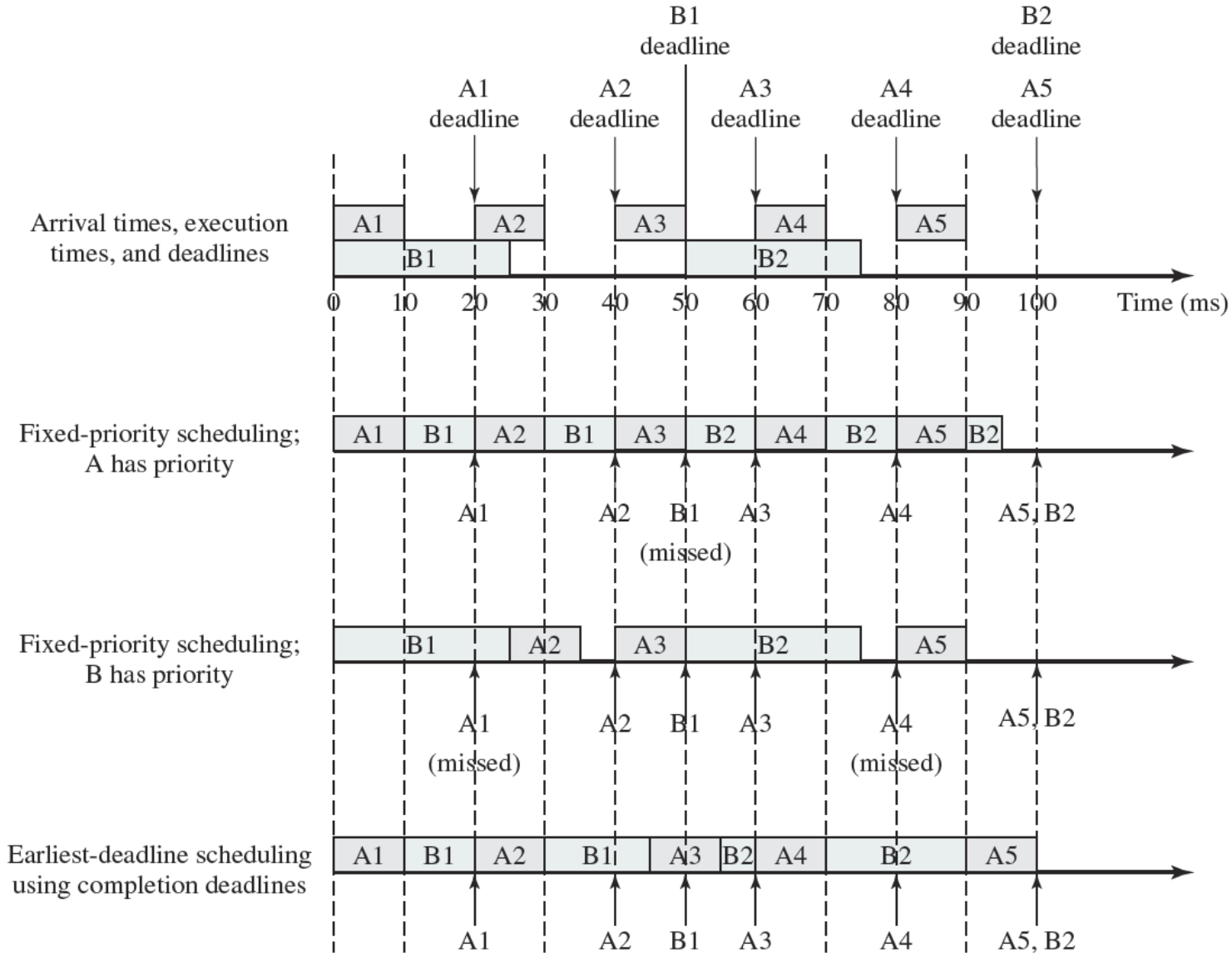
Informacije koje se koriste za rokove

Vrijeme spremnosti	• Vreme kada zadatak postaje spreman za izvršenje
Početni rok	• Vrijeme kada zadatak mora početi
Rok završetka	• Vrijeme kada se zadatak mora završiti
Vrijeme obrade	• Vrijeme izvršavanja zadatka
Zahtjevi za resurse	• Resursi potrebni tjekom izvršavanja
Prioriteti	• Relativna važnost zadatka
Planer podzadataka	• Zadatak se može rastaviti na obvezni i neobavezni podzadatak

Profil izvršenja dvaju periodičnih zadatka

Process	Arrival Time	Execution Time	Ending Deadline
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
•	•	•	•
•	•	•	•
•	•	•	•
B(1)	0	25	50
B(2)	50	25	100
•	•	•	•
•	•	•	•
•	•	•	•

Planiranje periodičnih zadataka u stvarnom vremenu s rokovima dovršetka



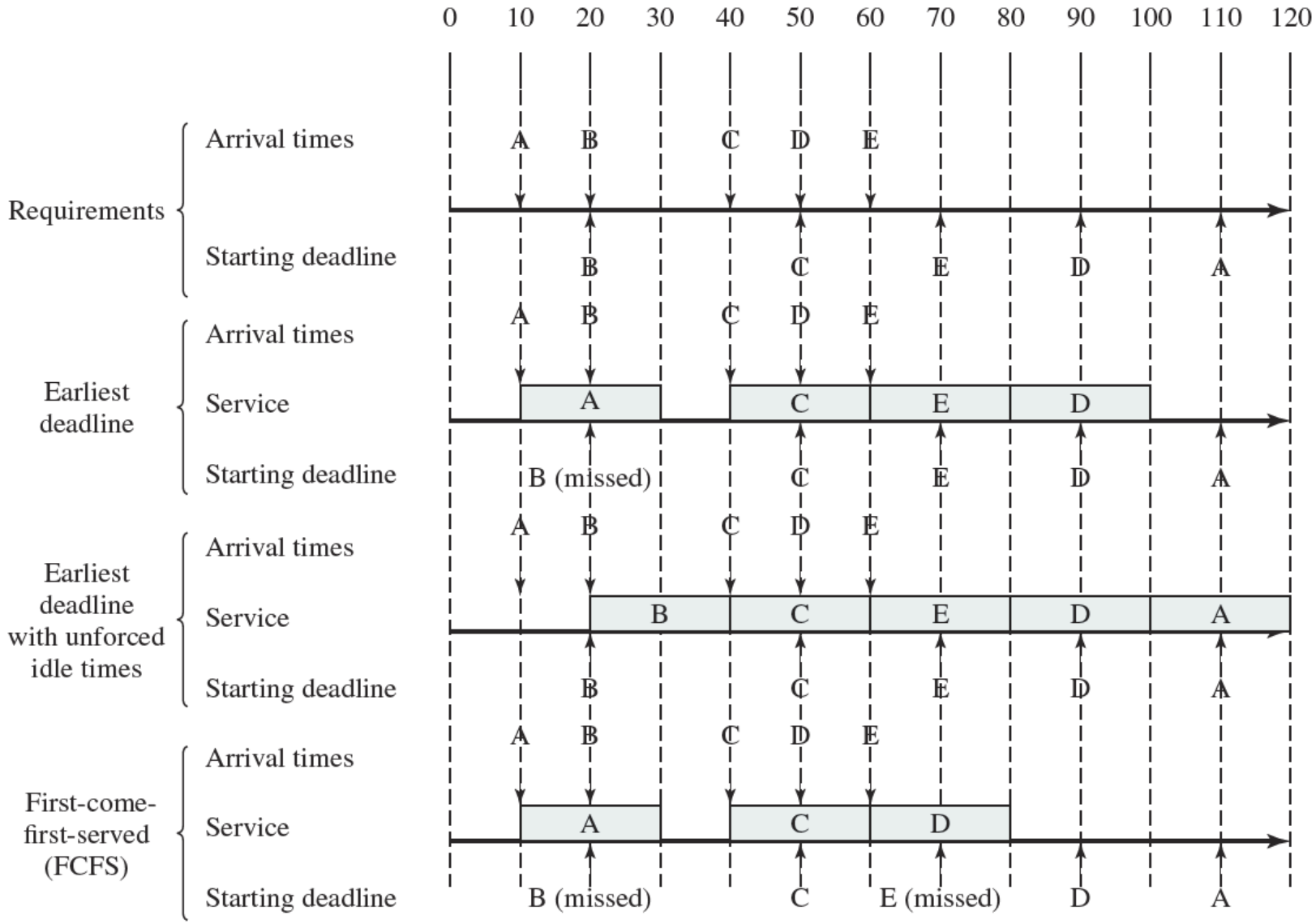
Najbliži rok (EDF)

- Dinamički algoritam koji ne zahtijeva da procesi budu periodični ili da jednako troše CPU cijelo vrijeme
- Algoritam
  - Kreira se popis pokrenutih procesa razvrstanih po vremenskom roku
  - Počinje onaj koji ima najbliži rok
  - Kada novi proces uđe u sustav, njegov rok se odmah provjerava, a ako je manji, trenutni proces se prekida

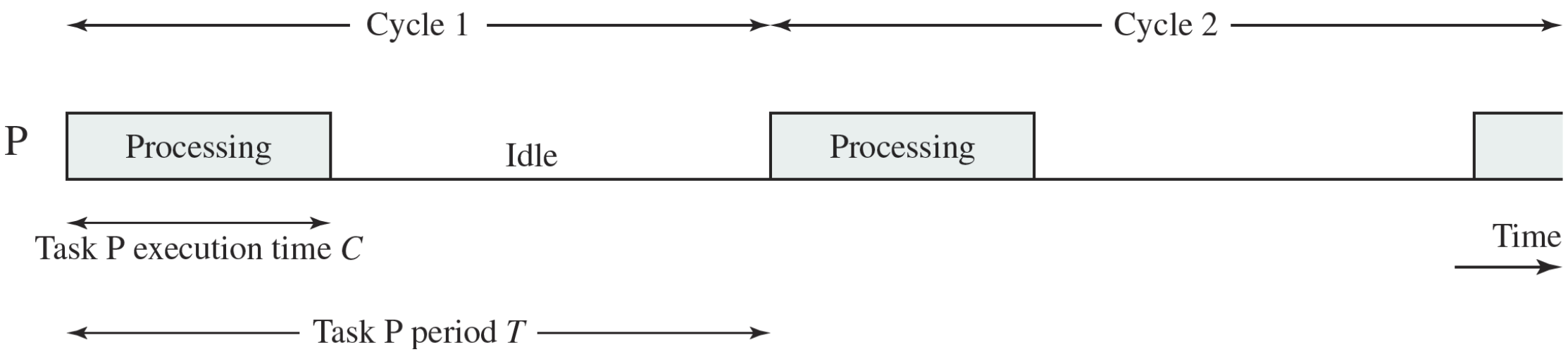
Profil izvršenja pet neperiodičnih zadatka

Process	Arrival Time	Execution Time	Starting Deadline
A	10	20	110
B	20	20	20
C	40	20	50
D	50	20	90
E	60	20	70

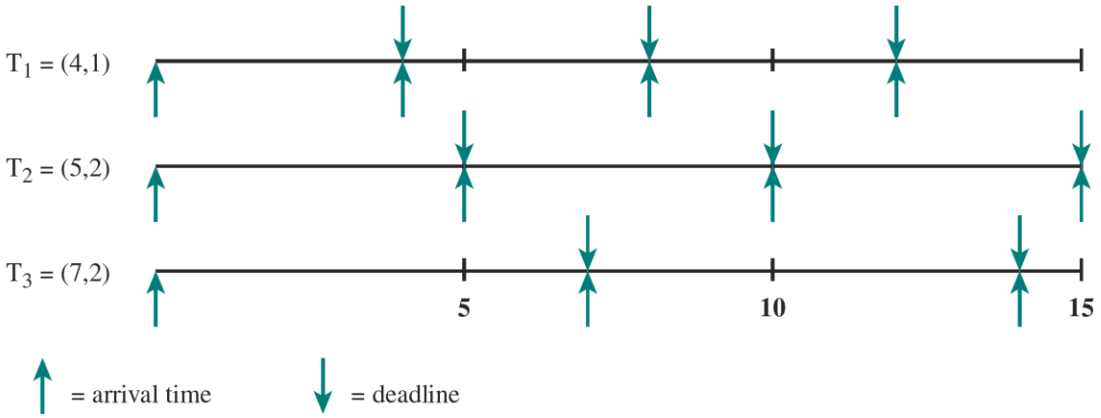
Planiranje neperiodičnih zadataka u stvarnom vremenu s početnim rokovima



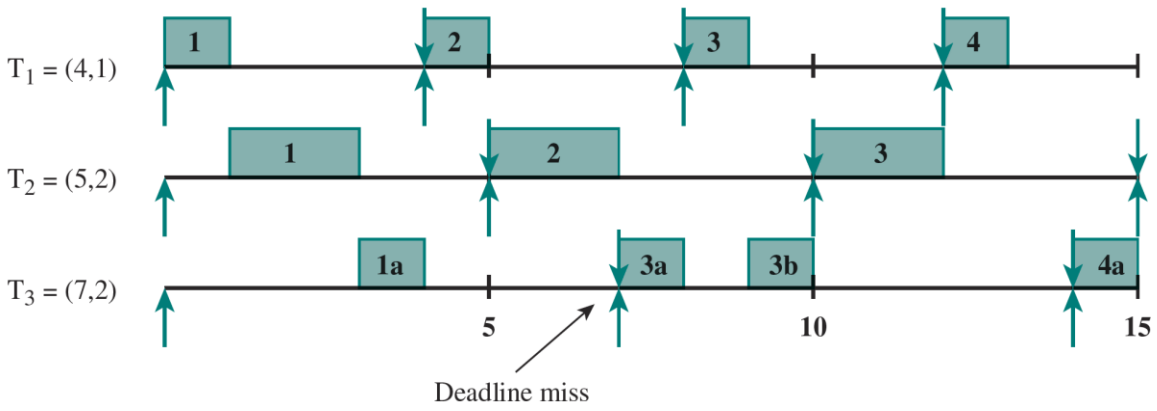
Rate Monotonic Scheduling



Rate Monotonic Scheduling



(a) Arrival times and deadines for task $T_i = (P_i, C_i)$; P_i = period, C_i = processing time



(b) Scheduling results

Rate Monotic Scheduling

- Uvjeti:
  - Svaki proces mora završiti svoj rad unutar svog perioda
  - Nijedan proces ne ovisi ni o kome drugome
  - Svaki proces zahtijeva jednako CPU vrijeme za svaki poziv
  - Svaki neperiodični proces nema rok
- Algoritam
- Svaki proces dobiva prioritet koji se izračunava kao broj izvršenja u sekundi (svakih 30 sekundi = 33 puta u sekundi, svakih 50 ms = 20 puta u sekundi...)
- Uvijek se pokreće proces s najvišim prioritetom. Viši prioritet uvijek prekida niži prioritet.

RMS - Example

- A priority 33
- B priority 25
- C priority 20

C ima najniži prioritet i čekajući da svi završe

A ima veći prioritet i prekida **B**

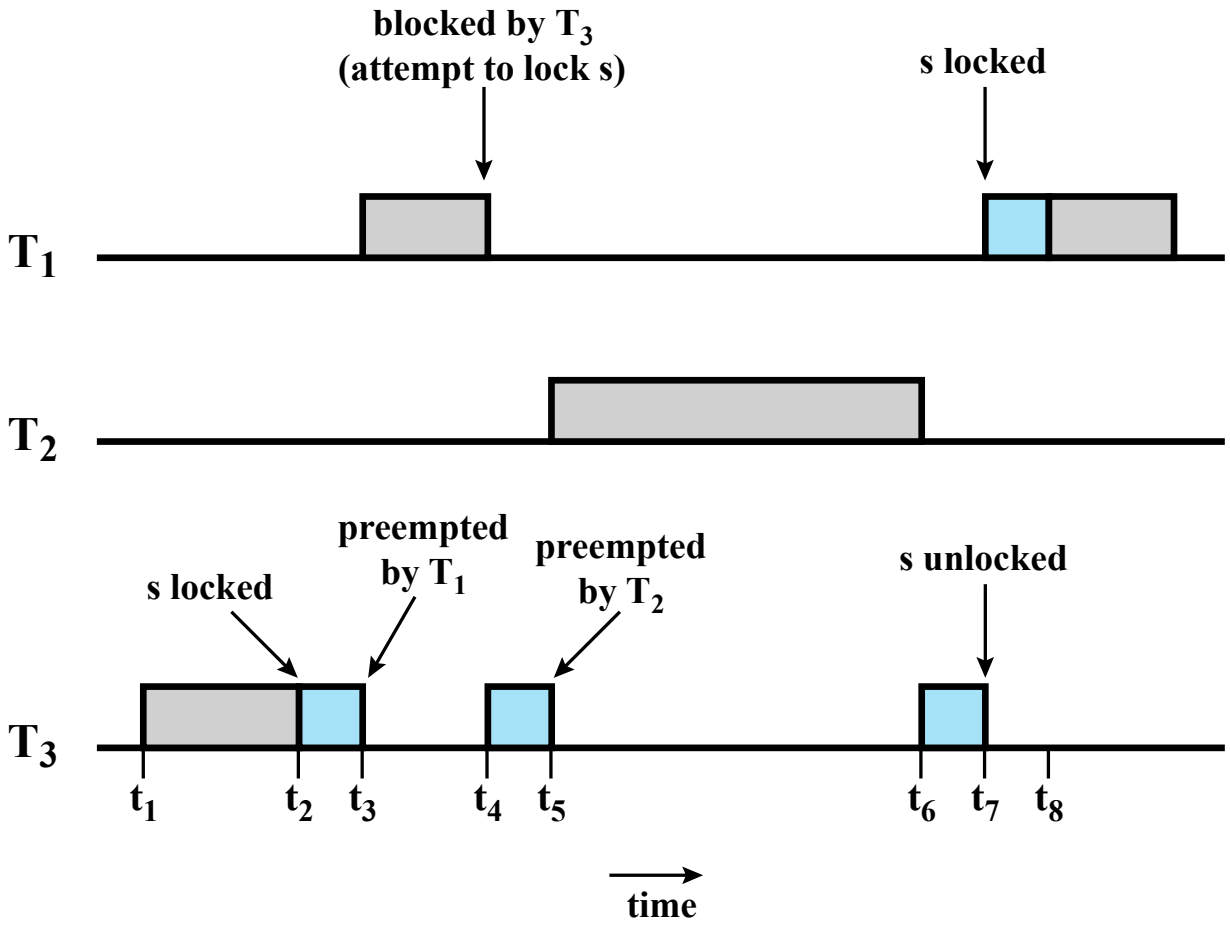
Inverzija prioriteta

- Može se pojaviti u bilo kojoj shemi prekidnog planiranja temeljenoj na prioritetu
- Posebno relevantno u kontekstu planiranja u stvarnom vremenu
- Najpoznatiji primjer uključivao je misiju Pathfinder na Mars
- Pojavljuje se kada okolnosti unutar sustava prisile zadatka višeg prioriteta da čeka zadatak nižeg prioriteta

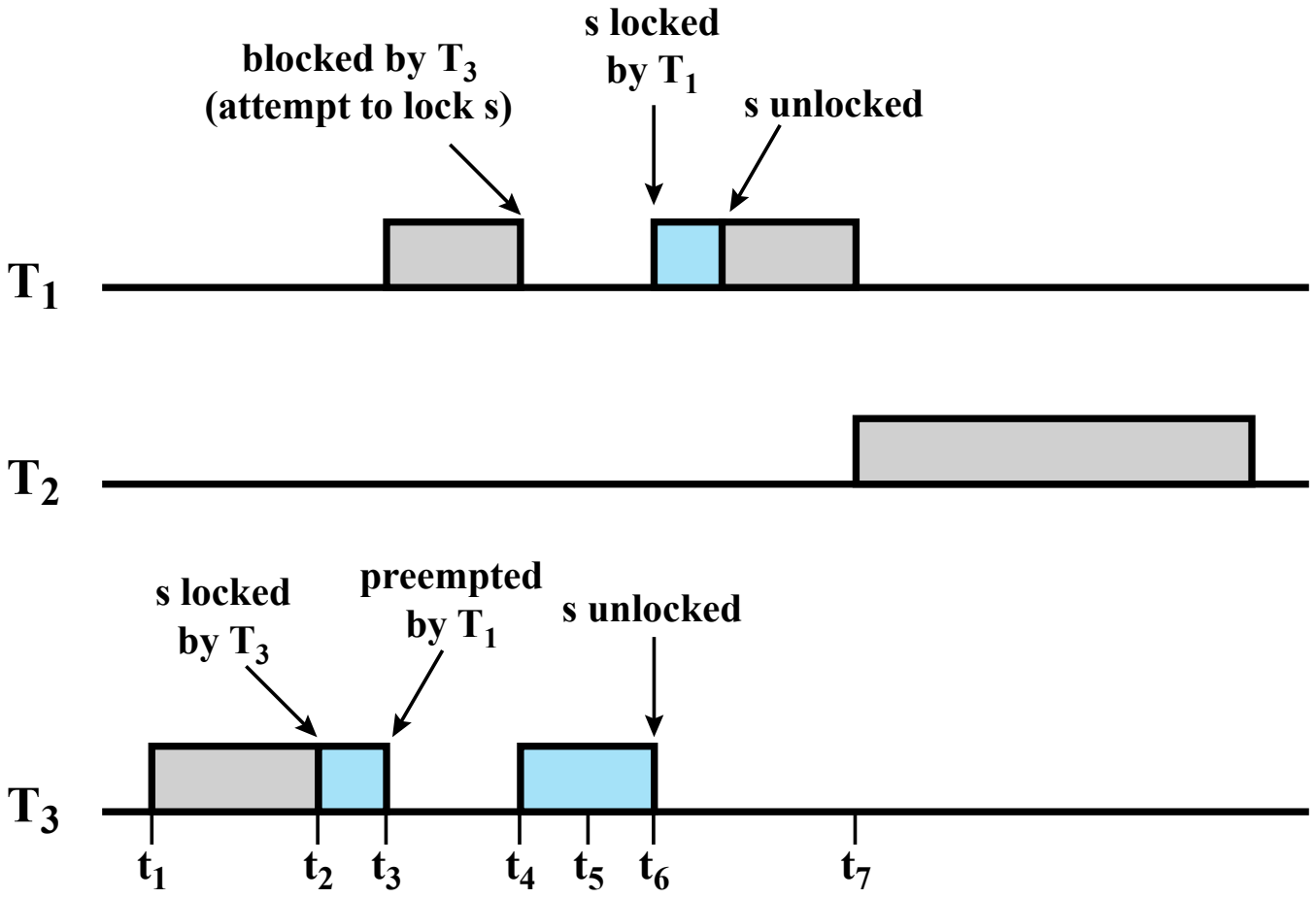
Neograničena inverzija prioriteta

- Trajanje inverzije prioriteta ne ovisi samo o vremenu potrebnom za rukovanje zajedničkim resursom, već i o nepredvidivim radnjama drugih nepovezanih zadataka

Neograničena inverzija prioriteta



Nasljeđivanje prioriteta



Sažetak

- Višeprocesorsko i višejezgreno raspoređivanje
  - Granularnost
  - Problemi dizajna
  - Planiranje procesa
  - Planiranje dretvi
  - Planiranje višejezgrenih dretvi
- Raspored u stvarnom vremenu
  - Pozadina
  - Karakteristike operacijskih sustava u stvarnom vremenu
  - Raspored u stvarnom vremenu
  - Zakazivanje roka
  - Prioritetno monotono zakazivanje
  - Inverzija prioriteta

Thank you for your attention!