Praktikum iz operativnih sistema

Lekcija 3: Raspoređivanje poslova

zima 2019/2020

Prof. dr Branimir Trenkić

Raspoređivanje instance posla

- U ovoj lekciji ćemo razmatrati probleme koji mogu nastaju prilikom postupka ***raspoređivanja individualne instance nekog posla*** (job scheduling)
- Da se podsetimo...

Neka je dat ***skup poslova*** $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$,
raspored je funkcija tipa $\sigma : R^+ \rightarrow N$ tako da
je $\forall t \in R^+, \exists t_1, t_2$:

$$t \in [t_1, t_2) \rightarrow \forall t' \in [t_1, t_2) : \sigma(t) = \sigma(t')$$

$$\sigma(t) = \begin{cases} k > 0 & \tau_k \text{ se izvrsava} \\ 0 & \text{procesor slobodan} \end{cases}$$

Definicije

- Za raspored σ kažemo da je ***izvodljiv*** (moguć) ako je za svaki posao omogućeno da bude kompletiran unutar pridruženog skupa uslova

- Za skup poslova Γ kažemo da je ***rasporedljiv*** ako za njega postoji bar jedan izvodljiv raspored

Opšti problem raspoređivanja

- Da bi ***definisali problem*** – odrediti ***tri skupa***:

  - skup Γ od **n** **poslova**, $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$;

  - skup **P** od **m** **procesora** $P = \{P_1, P_2, \ldots, P_m\}$ i

  - skup **R** od **r** **resursa** $R = \{R_1, R_2, \ldots, R_r\}$,

  pored toga ***može biti dato još*** i:

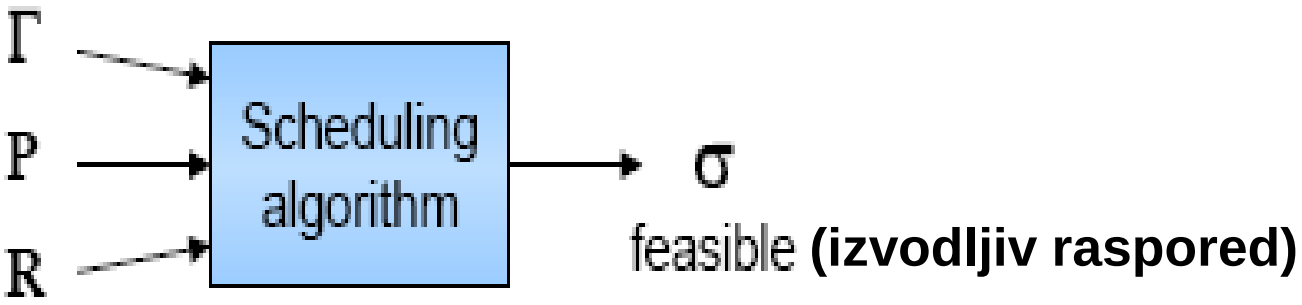
  - ***relacije prvenstva*** izvršavanja specificirane *direktnim acikličnim grafom*

  - ***vremenski zahtevi*** dodeljeni svakom poslu

Opšti problem raspoređivanja

Raspoređivanje označava:

dodeljivanje elemenata skupova **P** i **R** elementima skupa Γ, **tako da se dobije izvodljiv raspored**, odnosno, da se svi poslovi kompletiraju tako da svi postavljeni zahtevi budu ispunjeni



Složenost

- Sredinom 70-tih godina je pokazano da se ***opšti* problem raspoređivanja** odlikuje ***NP složenošću***
- Takođe, mogu se naći i algoritmi sa ***polinomnim vremenom***, ***O(p(n))***, ako se oni izvršavanja pod određenim uslovima
- ***Složenost*** algoritama raspoređivanja je ***relevantna*** za ***dinamičke sisteme*** – odluke o raspoređivanju se donose ***on-line***

Složenost

- Veoma je važno ***naći algoritme sa polinomnim vremenom***

Npr.:

broj poslova (n) = 30

osnovni korak = 1μs

- Algoritam 1: O(n) 30 μs
- Algoritam 2: $O(n^6)$ 12 min
- Algoritam 3: $O(6^n)$ 7 miliona godina

Predpostavke uprošćavanja

- U cilju ***redukovanja složenost*** određivanja izvodljivog rasporeda:
  - **Jedan** procesor
  - Svi poslovi se izvršavaju u **ne-prekidajućem modu**
  - **Istovremeno** aktiviranje
  - Ne postoje ***uslovi prvenstva***
  - Ne postoji ***uslovljenost resursima***

Kategorizacija algoritama

- Po pitanju prekidanja
  - ***Prekidajući*** (sa istiskivanjem) – ***Ne-prekidajući***
- Po pitanju prirode odlučivanja
  - ***Statički*** – ***Dinamički***
- Po pitanju korišćenja
  - ***On line*** – ***Off line***
- Po pitanju optimalnosti
  - ***Optimalni*** - ***Heuristički***

- "Vidovit" algoritam – unapred zna vremena aktiviranja svih poslova

Prekidajući – Ne- prekidajući

- ***Prekidajući***

  ovim algoritmom, posao koji se izvršava ***može biti prekinut*** u bilo kom trenutku da bi se procesor dodelio drugom aktivnom poslu, saglasno predhodno definisanoj politici raspoređivanja

- ***Ne- prekidajući***

  ovim algoritmom, posao čije je izvršenje započeto ***izvršava se do svog kompletiranja***. Sve odluke o raspoređivanju poslova donose se nakon terminacije izvršavanja posla

Statički – Dinamički

- ***Statički***

  odluka u procesu raspoređivanja se donosi na osnovu ***fiksnih parametara***, koji se dodeljuju poslovima pre njihovog aktiviranja

- ***Dinamički***

  odluka u procesu raspoređivanja se donosi na osnovu ***dinamičkih parametara***, čija se vrednost može promeniti tokom evolucije sistema

On line – Off line

- ***Off-line***

  algoritam se koristi off-line ako se on izvršava na celom skupu poslova ***pre aktuelnog izvršavanja poslova***, tako dobijeni raspored se smešta u tabelu (***tabelarno raspoređivanje***) koju dalje koristi dispečer

- ***On-line***

  algoritam se koristi on-line ako se odluke u procesu raspoređivanja donose ***u vreme izvršavanja*** kad god ***(I)*** novi posao ulazi u sistem ili ***(II)*** izlazi iz sistema.

Optimalni - Heuristički

- ***Optimalni***

  oni generišu raspored koji ***minimizira ciljanu cost – funkciju*** definisanu na skupu poslova

  Ako je jedini ***uslov - izvodljivi raspored***, ovaj algoritam ga neće ostvariti jedino ako ne postoji ni jedan algoritam te klase koji to može ostvariti

- ***Heuristički***

  oni generišu raspored saglasno heurističkoj funkciji koja ***pokušava*** da zadovolji optimalnost kriterijuma, ***ali ne garantuje*** se uspeh

Još jedna kategorizacija

- Algoritmi ***sa garancijama***
- Algoritmi ***sa dobrim namerama***
- Algoritmi bazirani na ***nepreciznim (aproksimativnim) izračunavanjima***

Algoritmi sa garancijama

- ***Sistemi*** sa vremenskim ograničenjem odziva => obezbediti ***visoki stepen predvidljivog*** ponašanja,

  tj. izvodljivost rasporeda se garantuje ***u napred***

⇒ Ako se ***kritičan posao ne može izvršiti*** do njegovog deadline-a, još uvek ima vremena da se izvrši ***alternativna akcija*** i spreče katastrofalne posledice

⇒ Sistem planira akcije, gledajući napred u budućnost, predpostavljajući najgori mogući scenario

Algoritmi sa garancijama

- ***Statički sistemi*** (skup poslova fiksni & poznat unapred):
  - ***Aktiviranje*** svih poslova ***istovremeno*** – raspored se može izračunati ***off-line***
  - Ceo ***raspored*** se može smestiti ***u tabelu***
  - U vreme izvršavanja, ***dispečer*** jednostavno ***prati sadržaj tabele***

\+ moguć je vrlo sofisticirani off-line algoritam

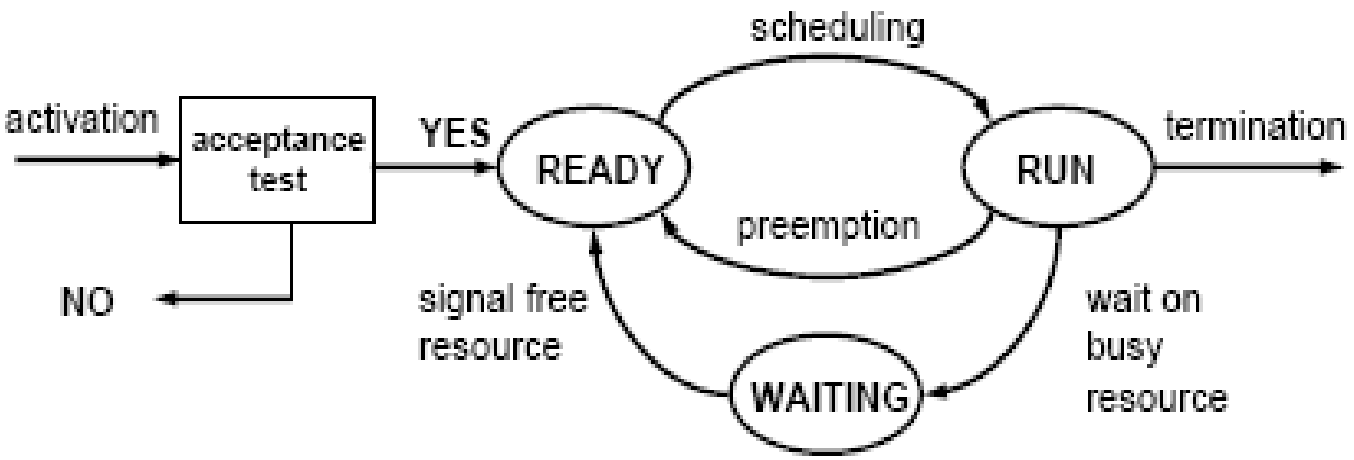
\- sistem je nefleksibilan na promene u okruženju

Algoritmi sa garancijama

- ***Dinamički sistemi*** :

Garancije moraju biti ***date on-line*** u svakom trenutku kada novi posao ulazi u sistem

Tipična ***šema mehanizma garancija*** za dinamičke sisteme u realnom vremenu:



Algoritmi sa garancijama

- ***Dinamički sistemi*** :

Test prihvatanja:

- Neka je **Γ** tekući skup poslova, kome su već date garancije,

  τ_{new} - novo-prispeli posao u sistem

  τ_{new} se prihvata ***ako i samo ako*** je skup poslova $\Gamma' = \Gamma \cup \{\tau_{new}\}$ **rasporedljiv**, inače se posao τ_{new} odbacuje

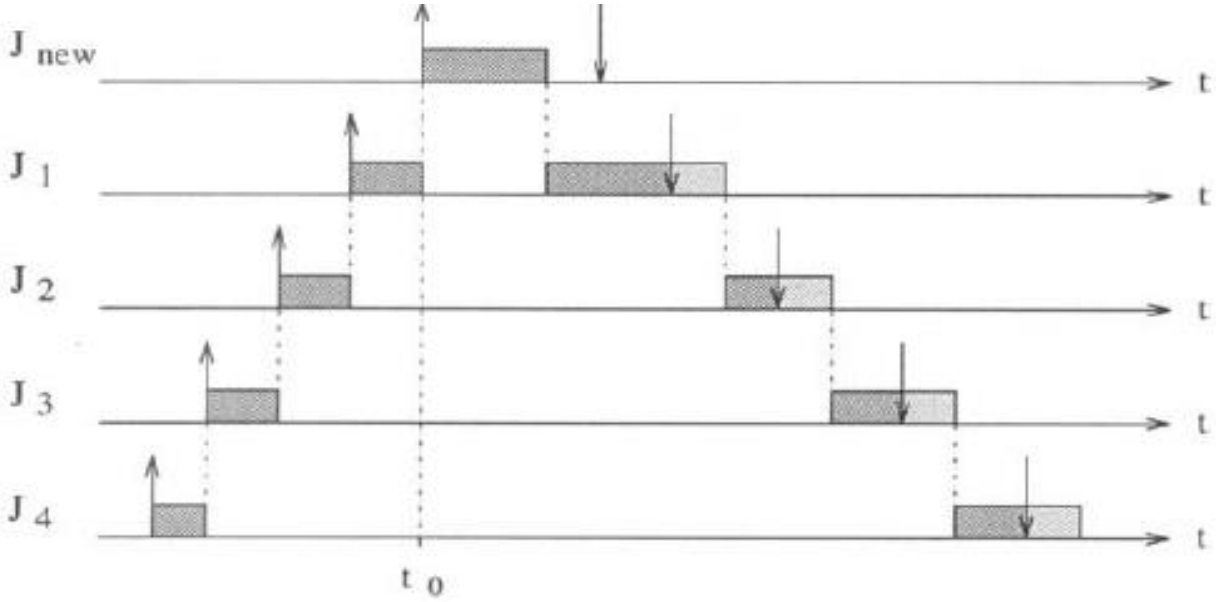
- Mehanizam garancije se bazira na predpostavci najgoreg slučaja

  => pesimistički (posao će nepotrebno biti odbačen)

Algoritmi sa garancijama

- Obezbeđivanje garancija za (hard) poslove sa ograničenim odzivom po cenu ***smanjenja opštih performansi*** sistema

+ Mehanizam garancija može u napred detektovati potencijalno preopterećenje – ***domino efekat***



Algoritmi sa dobrim namerama

- ***Soft- sistemi*** sa vremenskim ograničenjem odziva
- Jedina posledica neostvarivanja zadatog tajminga: ***degradacija performansi sistema***
- Primer: multi-medijalni prenos (može se desiti neki jitter)
- Algoritmi sa dobrim namerama (*Best-Effort algorithms*) - adekvatni za soft- sisteme sa vremenskim ograničenjem odziva

Algoritmi sa dobrim namerama

- ***Soft- sistemi*** :
- „pokušati najbolje”, tj. poslovi se stavljaju u red saglasno algoritmu koji uzima u obzir vremenske zahteve tog posla, ali se ***ne izvodi provera izvodljivosti*** => posao može biti prekinut i odbačen

\+ u proseku bolje funkcionišu od algoritama sa garancijama (manje opterećuju sistem)

\- nepredvidljiv, nepogodan za hard- aplikacije u realnom vremenu

Algoritmi sa nepreciznim izračunavanjima

- U dinamičkim sistemima, kada ne postoji dovoljno vremena i resursa da se izračunavanja kompletiraju do njihovog deadline-a – možda postoji ipak dovoljno vremena i resursa da se proizvede aproksimativni ***(približni) rezultat*** koji možda može sprečiti katastrofalne posledice.

Algoritmi sa nepreciznim izračunavanjima

- Sistemi koji podržavaju neprecizna (ili aproksimativna) izračunavanja:
- ***Svaki posao*** $\mathbf{J_i}$ (C_i) je podeljen u dva dela:
- ***Obavezni*** pod-posao $\mathbf{M_i}$ (m_i)
- ***Opcioni*** pod-posao $\mathbf{O_i}$ (o_i)

$$m_i + o_i = C_i$$

- Oba pod-posla imaju isto vreme aktiviranja (a_i) i isti deadline (d_i) kao i originalni posao J_i
- Međutim, O_i se aktivira tek nakon kompletiranja M_i pod-posla

Algoritmi sa nepreciznim izračunavanjima

- Greška ε_i u rezultatu proizvedena sa J_i je definisana kao dužina O_i pod-posla koja je ostala ne kompletirana (odbačena) sa algoritmom raspoređivanja
- Ako je σ_i ukupno procesorsko vreme dodeljeno pod-poslu O_i, tada je

$$\epsilon_i = O_i - \sigma_i$$

Algoritmi sa nepreciznim izračunavanjima

- Prosečna greška na skupu poslova Γ :

$$\bar{\varepsilon} = \sum_{i=1}^{n} w_i \varepsilon_i$$

gde je w_i relativan značaj posla J_i u skupu Γ

- **Raspored je izvodljiv**, ako je svaki obavezni pod-posao kompletiran u intervalu $[a_i, d_i]$
- **Raspored je precizan**, ako je prosečna greška na skupu poslova jednaka 0, ili - i obavezni i opcioni pod-poslovi kompletirani u $[a_i, d_i]$

Kriterujumi optimalnosti

- ***Izvodljivost*** - Naći izvodljiv raspored ako postoji
- Minimizirati maksimalno kašnjenje
- Minimizirati broj vremenskih prekoračenja deadline
- Dodeliti vrednost svakom poslu, a zatim maksimizirati vrednosti poslova koji su izvodljivi

Algoritmi raspoređivanja
Graham-ova notacija

$$\alpha \mid \beta \mid \gamma$$

α - označava ***broj procesora***

β - označava ***uslov(e)*** pridružen(e) poslovima

γ - označava ***kriterijum optimalnosti***

Primeri:

$1 \mid \mathrm{preem.} \mid \overline{R}$:

jednoprocesorski algoritam za ***prekidajuće*** poslove koji ***minimizira*** **prosečno vreme odziva** (***SJF***)

Graham-ova notacija

Primeri:

1 | sync. | L_{max} :

jednoprocesorski algoritam za ***sinhrone*** poslove koji ***minimizira maksimalno prekoračenje***

1 | preem. | L_{max} :

jednoprocesorski algoritam za ***prekidajuće*** poslove koji ***minimizira maksimalno prekoračenje***, ***EDF***

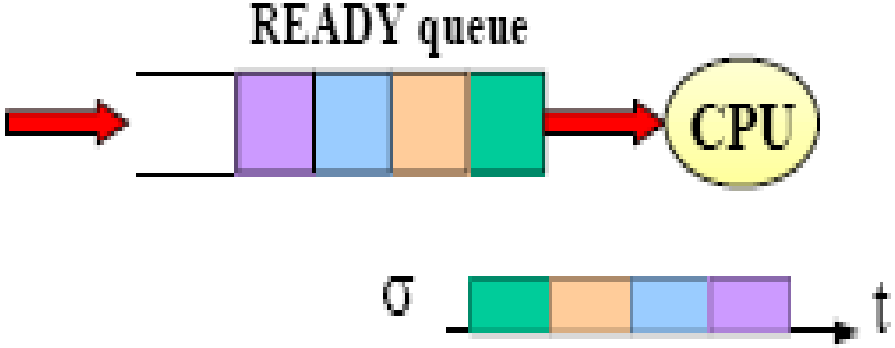
Klasični algoritmi raspoređivanja

- Prvi aktivirao (došao), prvi uslužen (***FCFS***)
- Prvo najkraći posao (***SJF***)
- Raspoređivanje ***na osnovu prioriteta***
- ***Round Robin***

Napomena: Nisu pogodni za sisteme sa vremenskim ograničenjem odziva

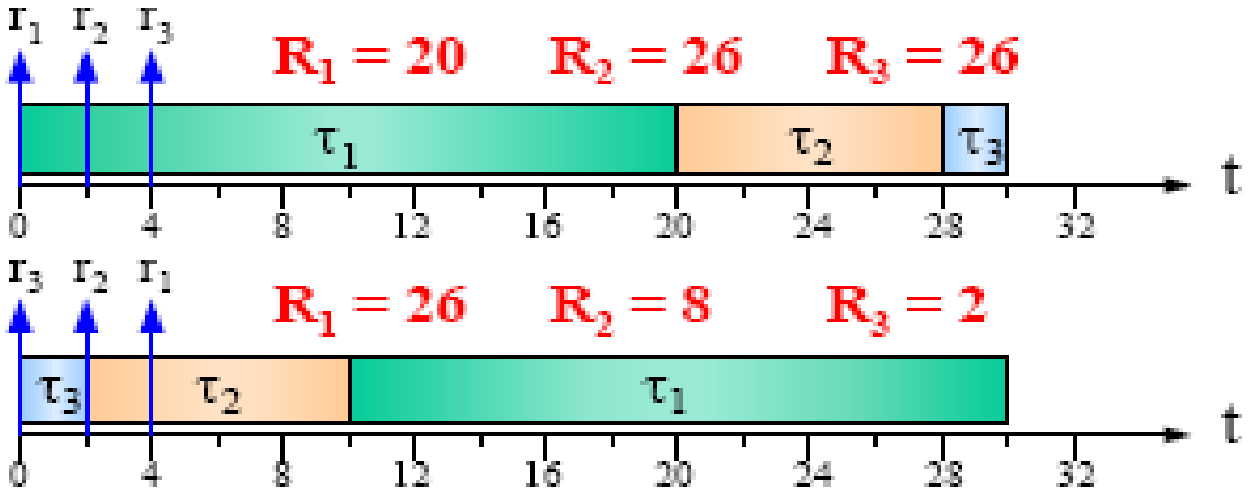
Prvi aktivirao, prvi izvršio (FCFS)

- ***Dodeljuje*** procesor poslovima ***na osnovu*** njihovog ***vremena aktiviranja*** (*arrival time*)
  - ***ne-prekidajući*** (ne poštuje prioritete već samo vreme aktiviranja)
  - ***dinamički***
  - ***on-line***
  - ***nije optimalan*** (***krajnje nepodesan*** za interaktivne sisteme)



Prvi aktivirao, prvi izvršio (FCFS)

- ***Vrlo nepredvidljiv***
  - ***vremena odziva*** strogo ***zavise*** od ***vremena aktiviranja***
  - Generalno, ***vreme izvršavanja poslova*** znatno utiče na performanse
    - Više ***kratkih poslova*** mogu da ***čekaju*** izvršavanje jednog posla



Prvo najkraći posao (SJF)

- ***Izbor*** posla na osnovu dužine ***vremena izvršavanja***
- Procesor se dodeljuje onom procesu kome treba ***najkraće vreme*** za izvršavanje

- SJF varijante:
  A. ***Ne-prekidajući***
  B. ***Prekidajući***

Prvo najkraći posao (SJF)

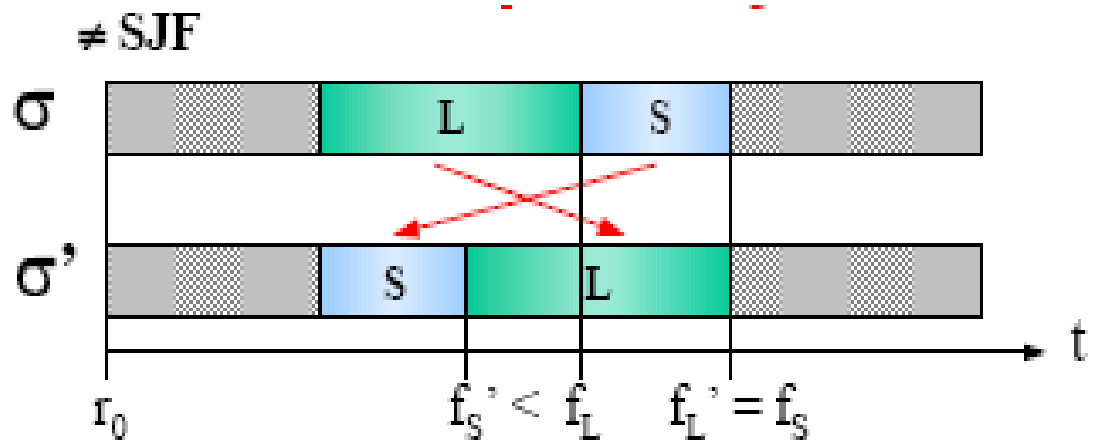
- ***Ne-prekidajući SJF***
  - Uvek završiti tekući posao bez obzira kakav se novi posao pojavio u redu čekanja
- ***Prekidajući SJF***
  - Ukoliko je vreme potrebno za izvršenje novog posla ***kraće*** od rezidualnog vremena izvršavanja (vremena potrebnog za završetak aktivnosti) tekućeg posla, procesor će biti dodeljen novom poslu (***$c_i(r_j) > C_j$***, ***J_j*** - novi; ***J_i*** - tekući)
  - Naziva se i ***raspređivanje sa najmanjim preostalim vremenom*** (*shortest-remaining time first, SRTF*)

Prvo najkraći posao (SJF)

- SJF varijante:
  - ***Statički*** (C_i je konstantni parametar) ili
  - ***dinamički*** (procenjuje se vreme izvršavanja – ne zna se unapred)
- SJF varijante:
  - Može se koristiti kao ***on-line*** i
  - Kao ***off-line***

- ***Minimizira srednje vreme odziva*** (*optimalan po pitanju srednjeg vremena odziva*)

SJF Optimalnost



$$f_s' + f_L' \le f_s + f_L \Rightarrow \left(f_S' - r_S\right) + \left(f_L' - r_L\right) \le \left(f_S - r_S\right) + \left(f_L - r_L\right)$$

$$\bar{R}(\sigma') = \frac{1}{n}\sum_{i=1}^{n}\left(f_i' - r_i\right) \le \frac{1}{n}\sum_{i=1}^{n}\left(f_i - r_i\right) = \bar{R}(\sigma)$$

SJF Optimalnost

$$\sigma \to \sigma' \to \sigma'' \to \cdots \to \sigma^*$$

$$\bar{R}(\sigma) \geq \bar{R}(\sigma') \geq \cdots \geq \bar{R}(\sigma^*)$$

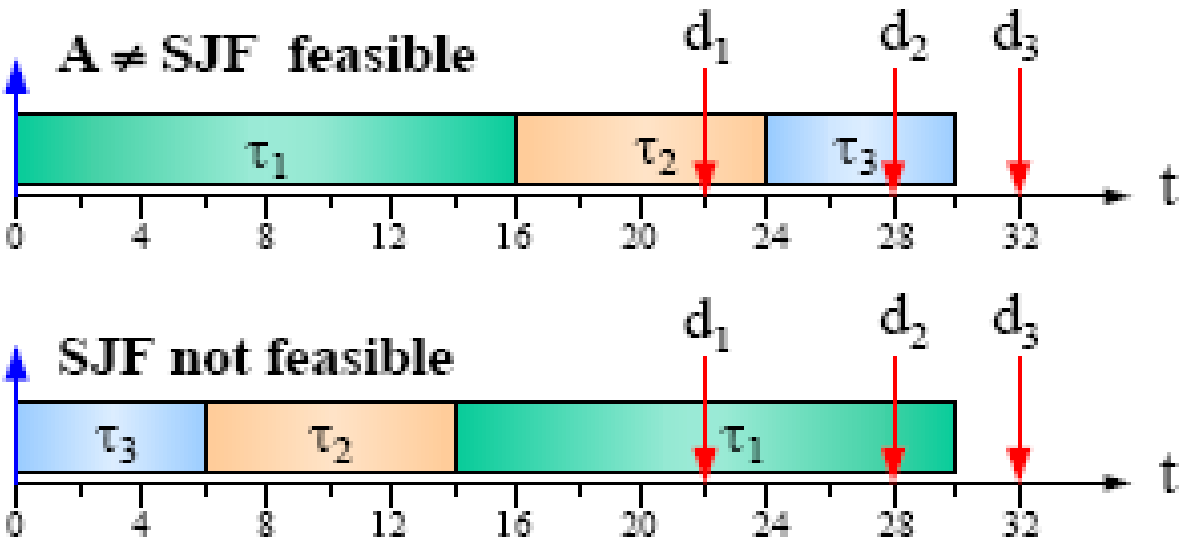
$$\sigma^* = \sigma_{SJF}$$

$$\bar{R}(\sigma_{SJF}) -$$

minimalno srednje vreme odziva koje može biti dostignuto sa bilo kojim algoritmom

SJF pogodan za RT sisteme?

- ***Nije optimalan*** u smislu ***izvodljivosti !!!***



Raspored na osnovu prioriteta

- Svakom poslu je ***dodeljen prioritet***: $p_i \in [0,255]$
- Posao ***sa najvišim prioritetom*** se bira za ***izvršavanje***
- Raspoređivanje poslova ***istog prioriteta po principu FCFS***
- ***Varijante:***
  - ***prekidajući***
  - ***statički*** ili ***dinamički***
  - ***on line***

Raspored na osnovu prioriteta

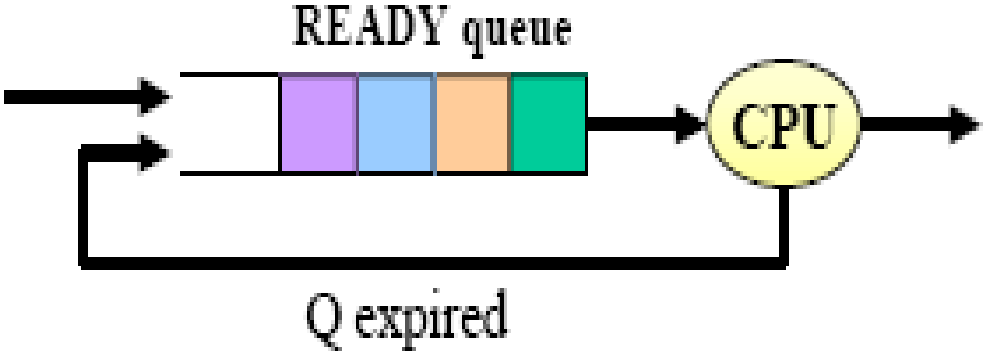
- **Problem**: neki poslovi predugo bez kompletiranja
  - ***Poslovi niskog prioriteta*** mogu pretrpeti ***veliko kašnjenje*** u slučaju prekidanja izvršenja od strane poslova sa višim prioritetom
  - ***starenje***
- ***Rešenje***: prioritet se ***povećava*** saglasno vremenu čekanja

Napomena:

$$p_i \propto 1/C_i \Rightarrow \text{SJF}$$
$$p_i \propto 1/r_i \Rightarrow \text{FCFS}$$

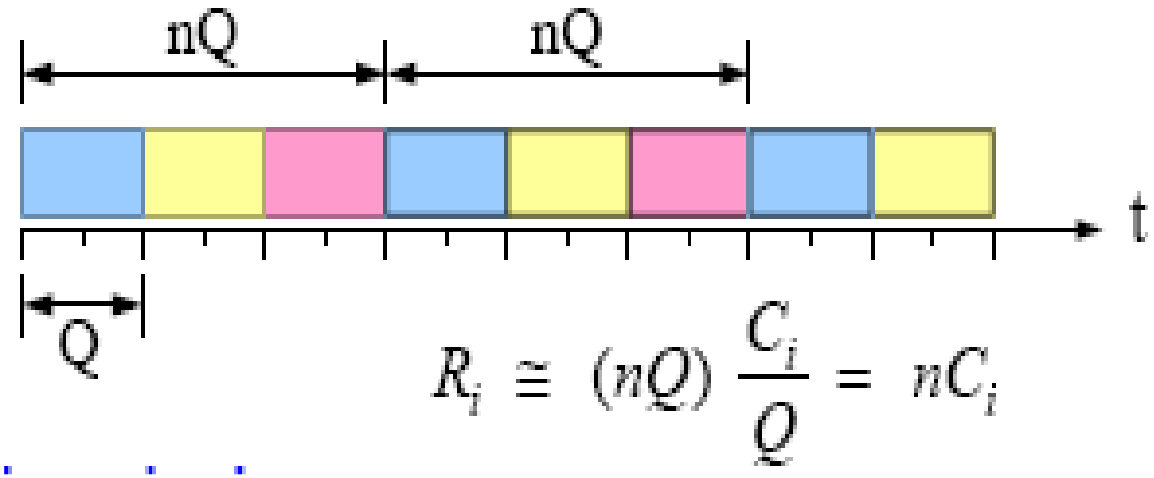
Round-Robin

- ***Niz spremnih*** se servisira ***kao FCFS***, ***ali*** …
- Svaki posao τ_j se može izvršavati ***maksimalno Q*** vremenskih jedinica (Q = ***vremenski kvantum***)
- Nakon isteka vremena Q, τ_j se vraća u red spremnih.



Round-Robin

- **n** = broj poslova u sistemu

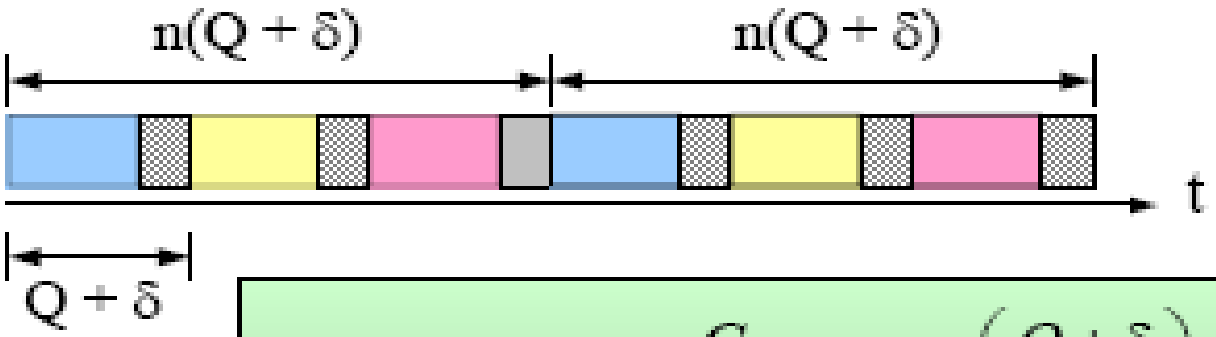


Deljenje vremena **(*Time sharing*):**
Svaki posao kao da se izvršava na posebnom virtuelnom procesoru ***n puta sporijem*** od realnog

Round-Robin

Performanse algoritma RR zavise od veličine vremenskog ***kvantuma Q***:

- **if** $Q > \max(C_i)$ **then** **RR ≡ FCFS**
- **if** $Q \cong$ context switch time (δ) **then**



$$R_i \cong n(Q+\delta)\frac{C_i}{Q} = nC_i\left(\frac{Q+\delta}{Q}\right)$$

Osnovni problem – ***optimalan*** izbor dužine kvantuma Q

Više-nivovsko raspoređivanje