Scheduling (raspoređivanje)

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?

Vrste procesorskog raspoređivanja

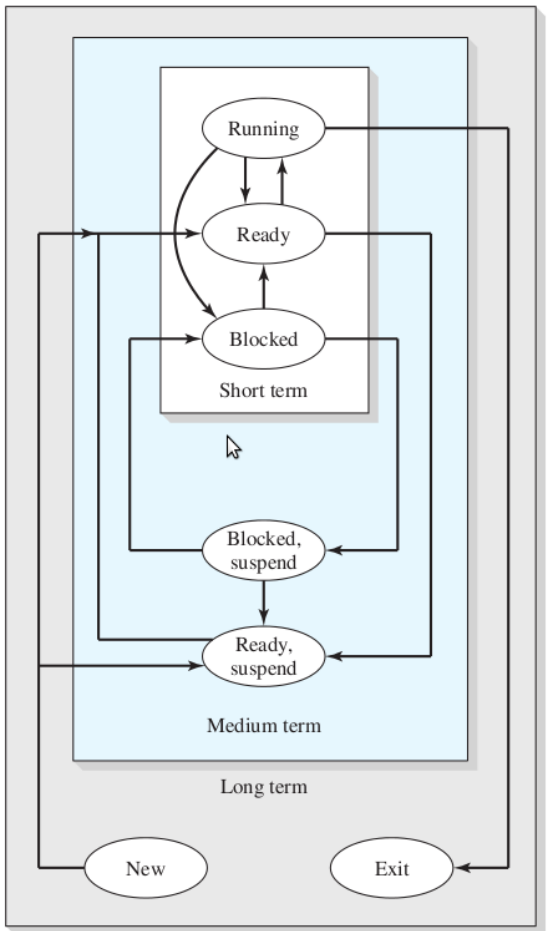
Definicija

U multiprogramskom okruženju, više programa postoji konkurentno u memoriji. Svaki od njih koristi procesor ili čeka neki događaj.

Vrste raspoređivanja

1. Dugoročno raspoređivanje - odluka da se dati proces doda skupu procesa za izvršavanje
2. Srednjeročno raspoređivanje - odluka da se dati proces doda broju procesa koji su delimično ili potpuno u RAM memoriji
3. Kratkoročno raspoređivanje - odluka koji od raspoloživih procesa će da se izvršava
4. Raspoređivanje U/I - odluka koji od nerešenih U/I zahteva će raspoloživi U/I uređaj da opslužи

Vrste procesorskog raspoređivanja
Nastavak



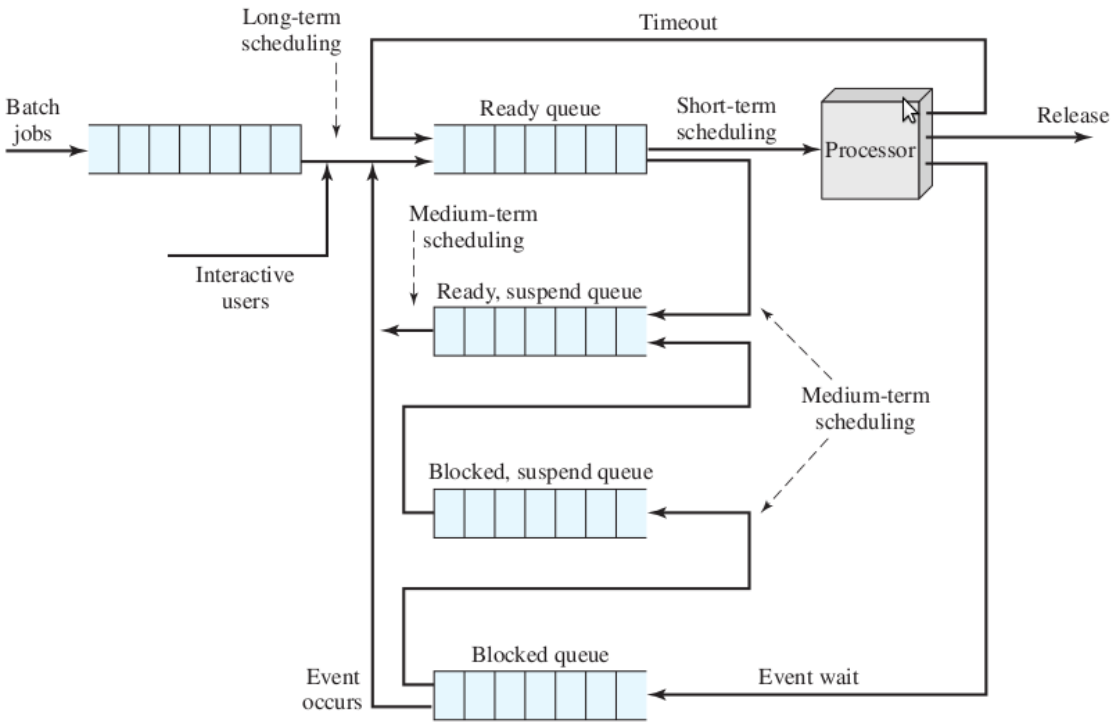


Figure 9.3 **Queuing Diagram for Scheduling**

Kratkoročno raspoređivanje

1. Poziva se mnogo češće od ostala dva
2. Često se naziva i dispečerom
3. Poziva se kada god dođe do događaja koji može da dovede do blokiranja tekućeg procesa
4. Takođe sâm može da uslovi prekid procesa koji se trenutno izvršava u korist nekog drugog procesa

Prekidni događaji u sistemu

1. Prkidi generatora takta (*clock*-a)
2. U/I prekidi
3. Prekidi OS-a
4. Signali (npr. semafori)

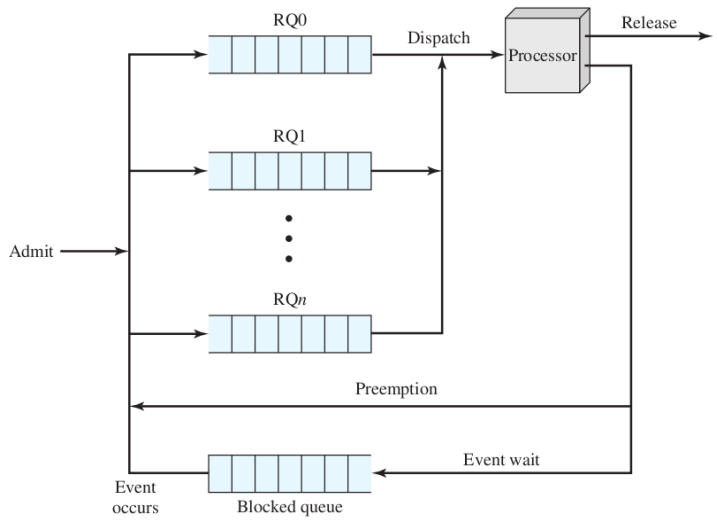
Algoritmi raspoređivanja

Ustanovljava se skup kriterijuma prema kojima se procenjuje kvalitet algoritma raspoređivanja.

- Orijentisani ka korisniku
  1. Vreme prolaska zadatka - interval između podnošenja i završetka posla
  2. Odzivno vreme
  3. Rokovi
  4. Predvidljivost
- Orijentisani ka sistemu
  1. Propusna moć - broj završenih poslova u jed. vremena
  2. Iskorišćenje procesora - procenat vremena u kom je CPU zauzet
  3. Pravičnost
  4. Primena prioriteta - isključiti *starwing* efekat
  5. Uravnoteženje resursa

Algoritmi raspoređivanja
Prioriteti

1. Svaki proces dobija prioritet
2. Umesto jednog reda čekanja *ready* procesa, ima ih više
3. Na UNIX-u, ako je prioritet $[RQ_i] >$ prioriteta $[RQ_j]$ ako je $i < j$, a na *MS Windows*-u obrnuto
4. Najjednostavnije: Opslužе se prvo redovi višeg prioriteta, ali problem može biti *starwing*



Algoritmi raspoređivanja
Politike raspoređivanja

- Funkcija izbora određuje koji će proces da ide sledeći na izvršavanje. Zavisi od 3 veličine:
  1. w - vreme provedeno u sistemu u čekanju i izvršavanju
  2. e - vreme provedeno u izvršavanju
  3. s - ukupno vreme usluživanja koje zahteva proces
  4. Npr. funkcija $max[w]$ ukazuje na FCFS algoritam
- Režim izbora određuje vremenske trenutke u kojima se izvršava funkcija izbora:
  1. Bez prekidanja - Izvršava se sve dok se sâm ne blokira za U/I ili dok se ne završi
  2. Sa prekidanjem - Proces može da prekine OS i stavi neki drugi na izvršenje

Politike raspoređivanja

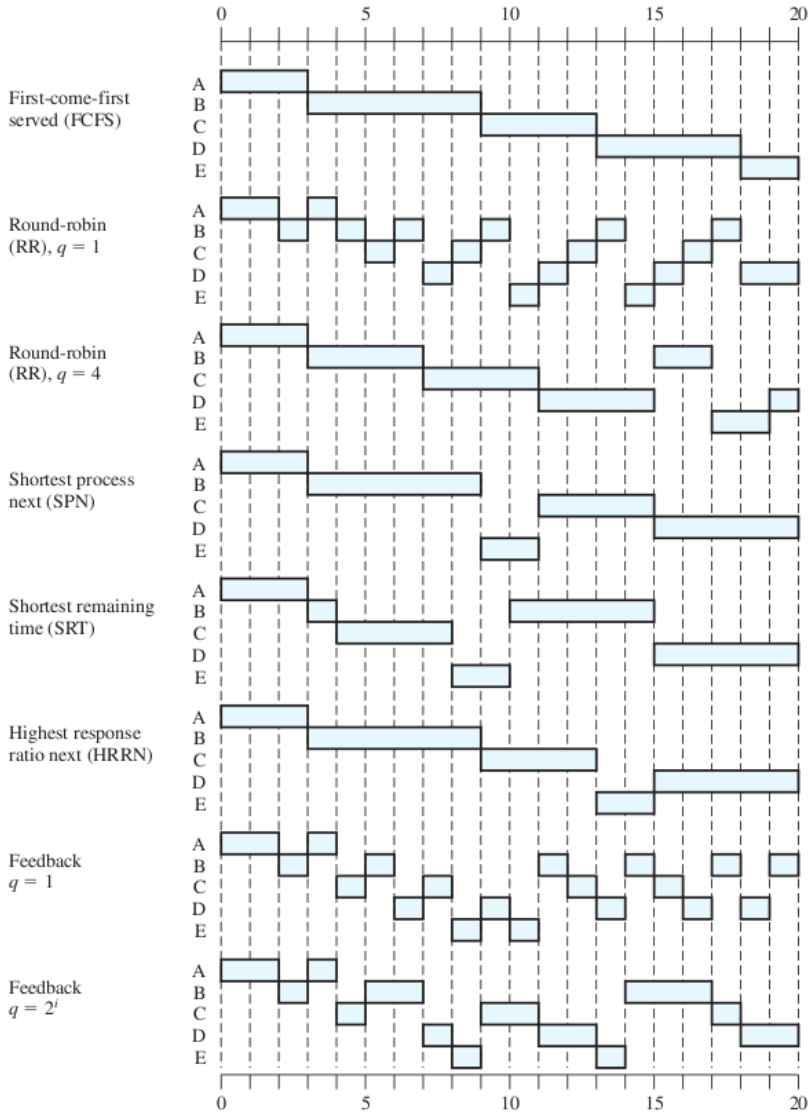
Parametri i primer

- Vreme prolaska zadatka (TAT-*TurnAround Time*) je T_r
- Normalizovano vreme prolaska je $T_r/T_s \geq 1$

Primer:

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Poređenje politika raspoređivanja



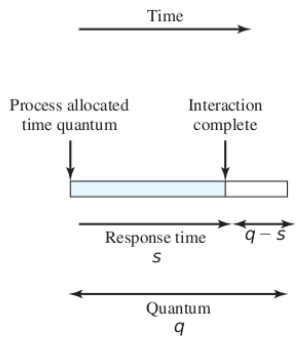
FCFS - *First Come First Served*

- Klasičan FIFO
- Kako koji proces uđe u *ready* stanje, priključuje se redu čekanja
- Favorizuje duge procese u odnosu na kratke
- Favoriuje procese sa intenzivnim korišćenjem CPU-a u odnosu na one sa dosta U/I
- Često se kombinuje sa prioritetnom šemom

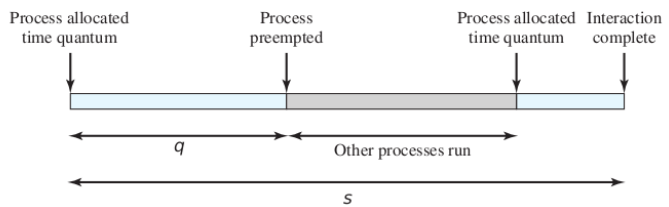
Process	Arrival Time	Service Time (T_s)	Start Time	Finish Time	Turnaround Time (T_r)	T_r/T_s
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1.99
Mean					100	26

Round Robin raspoređivanje

- Prekidanje zanosvano na signalima *clock*-a
- Time slicing algoritam, postoji dodatna režija obrade!
- *Time slice q* bi trebalo da bude malo veći od tipične interakcije



(a) Time quantum greater than typical interaction

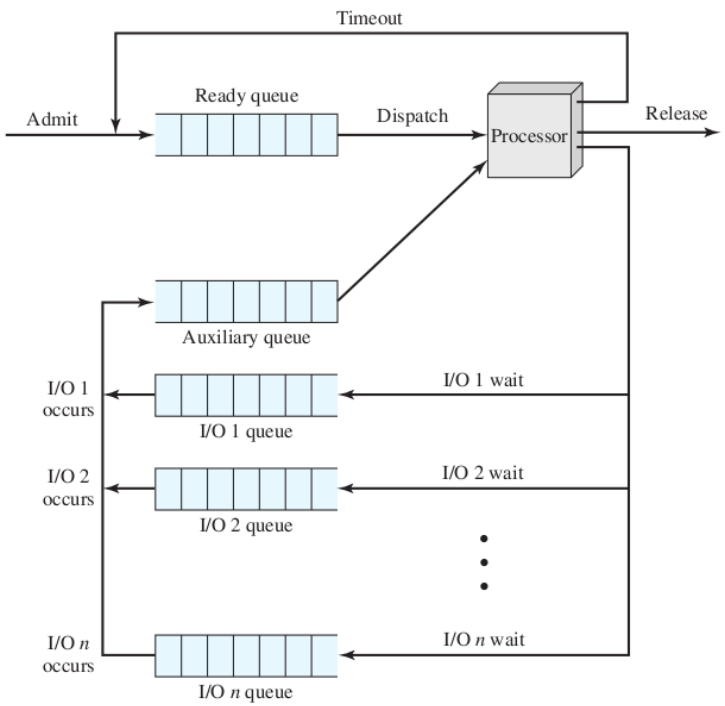


(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

Virtual Round Robin raspoređivanje

- Procesi sa intenzivnom upotrebom CPU-a i dalje su u prednosti u odnosu na procese sa mnogo U/I
- Dodaje se pomoćni red u koji ulaze procese koje otpušta neki U/I
- Taj pomoćni red ima prednost u raspoređivanju



Shortest Process Next raspoređivanje

- Politika bez prekidanja
- Najkraće očekivano vreme usluživanja (*service time*)
- Problem: Procena vremena obrade svakog procesa

Procena T_s za poslove koji se često izvršavaju

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

$$S_{n+1} = \frac{1}{n}T_n + \frac{n-1}{n}S_n$$

Eksponencijalno usrednjavanje za favorizovanje zadnjih

$$S_{n+1} = \alpha T_n + (1-\alpha)S_n$$

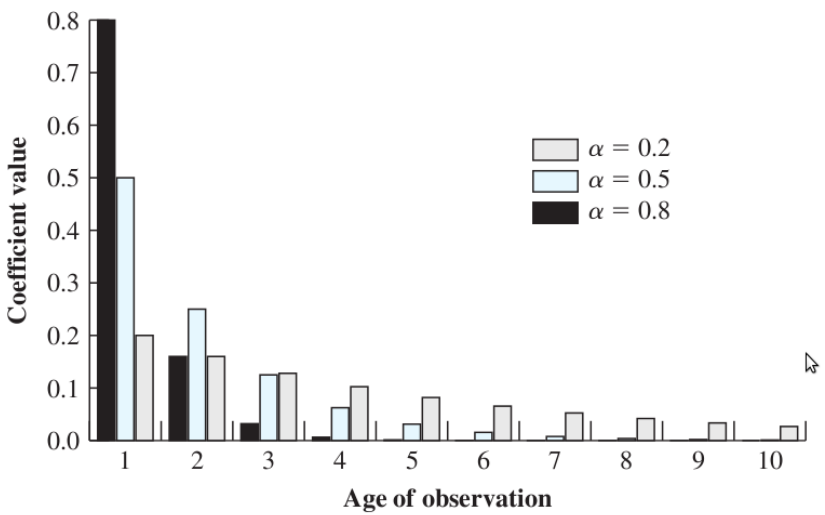
$$S_{n+1} = \alpha T_n + (1-\alpha)\alpha T_{n-1} + \cdots + (1-\alpha)^i \alpha T_{n-i} + \cdots + (1-\alpha)^n S_1$$

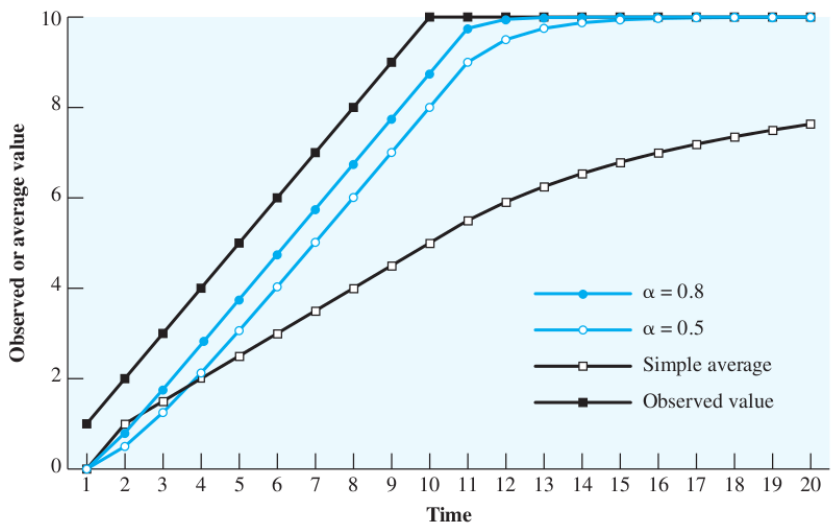
Shortest Process Next raspoređivanje

Nastavak

Problemi

- Kratak poremećaj u vrednosti T_s
- Mogućnost *starwing*-a
- Kažnjen proces Y zbog nemogućnosti prekidanja





Shortest Remaining Time raspoređivanje

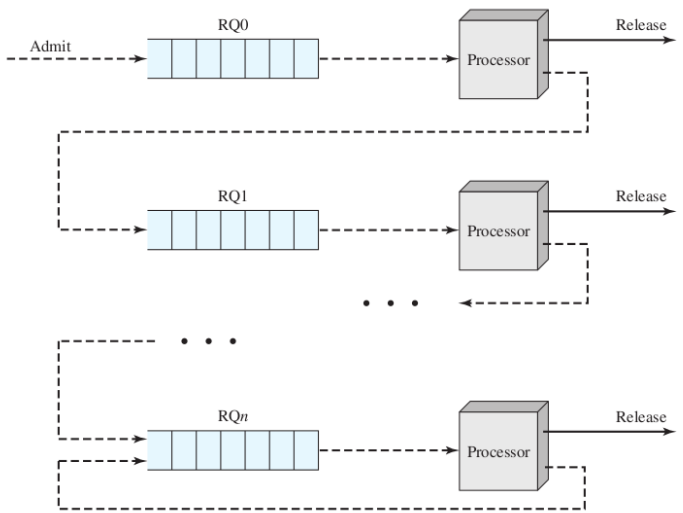
- Prekidna verzija SPN algoritma
- Bira se proces sa najkraćim preostalim vremenom obrade
- Mora da se proceni vreme izvršavanja kao i kod SPN
- Zapisivanje vremena izvršavanja opterećuje sistem

Highest Response Ratio Next raspoređivanje

- Minimizuje se normirano vreme prolaska, tj. veličina $R = \frac{w+s}{s}$ koje je uvek ≥ 1
- Pravičan pristup jer uzima u obzir starost procesa
- Opet mora de se procenjuje vreme usluge s
- Favorizuju se kraći poslovi, ali duži procesi koji čekaju stare, pri čemu se povećava w, pa i oni dolaze na red

FeedBack raspoređivanje

- Ako nema indikacije o dužini procesa, onda ne dolaze u obzir ni SPN, ni SRT, ni HRRN
- Drugi način je da se kazne procesi koji su se već dugo izvršavali
- Kada proces uđe prvi put, stavlja se na $RQ0$
- Svaki put kad se prekine, stavi se u red nižeg prioriteta
- Sa takvom šemom, može da se restegne izvršavanje dugačkog procesa
- Zato se procesu iz RQ_i dozvoljava da se izvršava 2^i vremenskih jedinica



Poisson-ove formule

- Modelovanje događaja prilikom čekanja u redovima može se donekle postići *Poisson*-ovim formulama
- Važi da je

$$\frac{T_r}{T_s} = \frac{1}{1-\rho}, \ \rho \text{ je iskorišćenje procesora}$$

- Teško je modelovati bilo koje druge politike sem FCFS i RR

Table 9.6 Formulas for Single-Server Queues with Two Priority Categories

Assumptions:
1. Poisson arrival rate.
2. Priority 1 items are serviced before priority 2 items.
3. First-come-first-served dispatching for items of equal priority.
4. No item is interrupted while being served.
5. No items leave the queue (lost calls delayed).

(a) General formulas

$$\lambda = \lambda_1 + \lambda_2$$

$$\rho_1 = \lambda_1 T_{s1}; \rho_2 = \lambda_2 T_{s2}$$

$$\rho = \rho_1 + \rho_2$$

$$T_s = \frac{\lambda_1}{\lambda} T_{s1} + \frac{\lambda_2}{\lambda} T_{s2}$$

$$T_r = \frac{\lambda_1}{\lambda} T_{r1} + \frac{\lambda_2}{\lambda} T_{r2}$$

(b) No interrupts; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1} + \rho_2 T_{s2}}{1-\rho_1}$$

$$T_{r2} = T_{s2} + \frac{T_{r1} - T_{s1}}{1-\rho}$$

(c) Preemptive-resume queuing discipline; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1}}{1-\rho_1}$$

$$T_{r2} = T_{s2} + \frac{1}{1-\rho_1}\left(\rho_1 T_{s2} + \frac{\rho T_s}{1-\rho}\right)$$

Poisson-ove formule
Rezultati primera

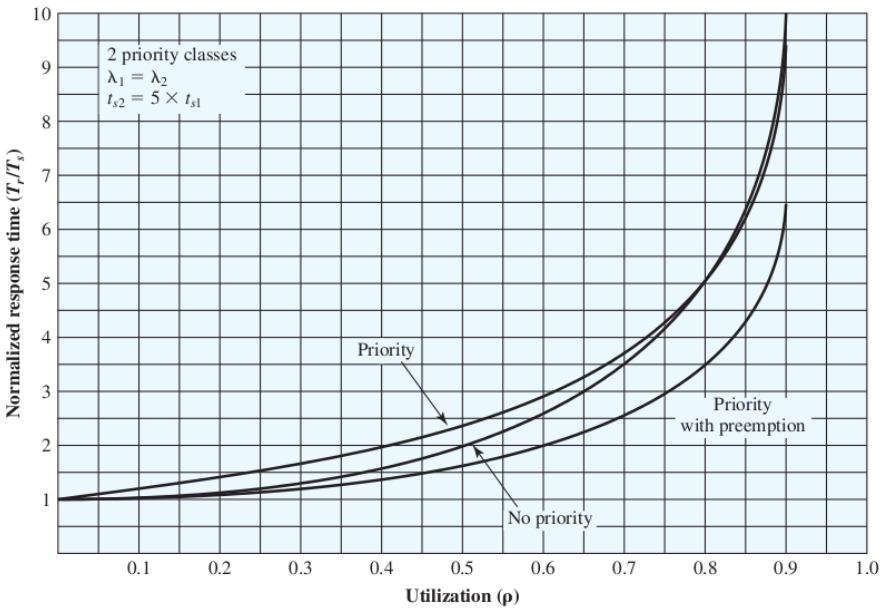


Figure 9.11 **Overall Normalized Response Time**

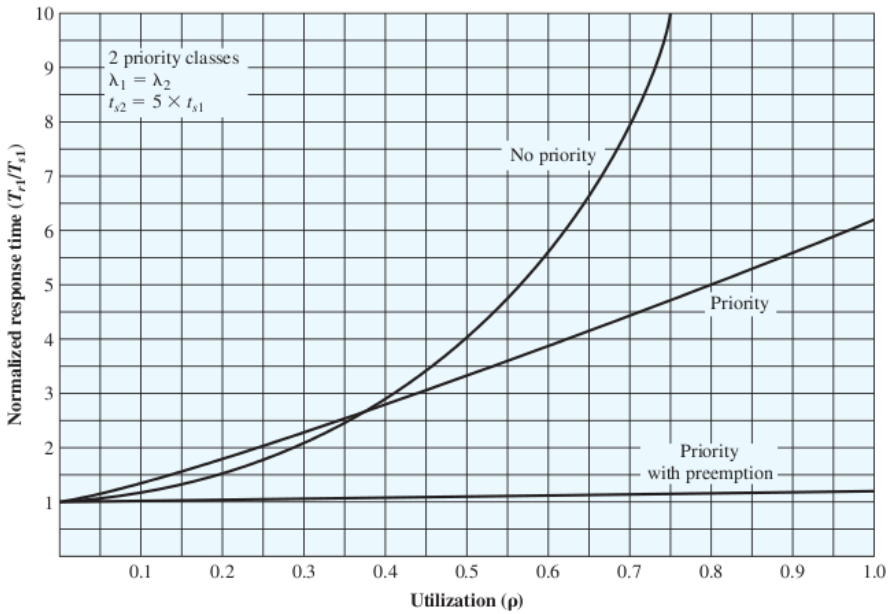


Figure 9.12 **Normalized Response Time for Shorter Processes**

Simulaciono modelovanje

Pretpostavke studije

Simulacija obuhvata 50000 procesa, sa frekvencijom dolaska od $\lambda = 0.8$ i srednjim vremenom opsluživanja $T_s = 1$. Dakle, pretpostavka je da je iskorišćenje procesora $\rho = \lambda T_s = 0.8$. Procesi su grupisani u procente vremena opsluživanja (apscisa).

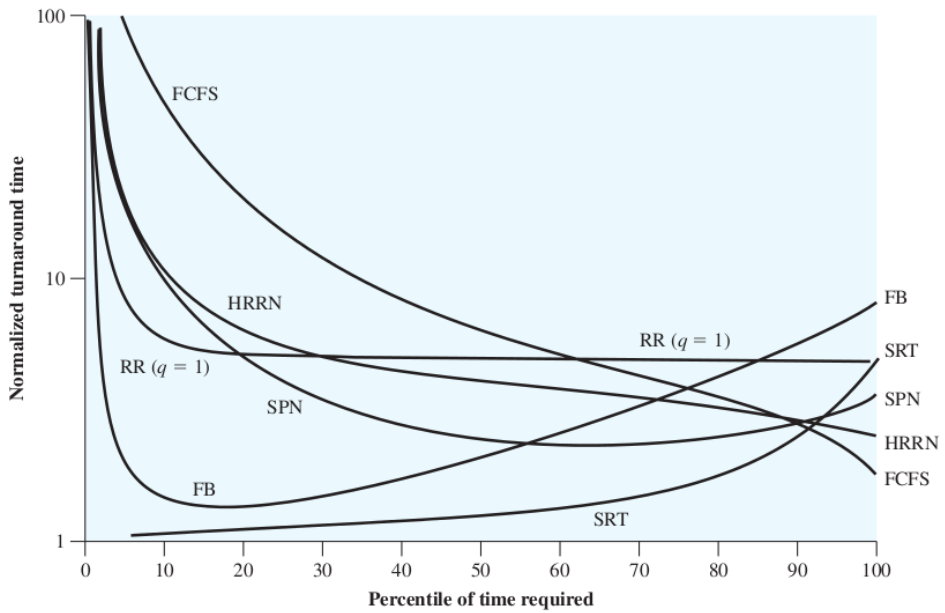


Figure 9.14 Simulation Result for Normalized Turnaround Time

Fairshare raspoređivanje

- U višekorisničkom sistemu više procesa aplikacije/a mogu da budu organizovane u niti ili grupe koje tradicionalni *scheduler* ne prepoznaje
- Svakom korisniku/grupi dodeljuje se težina koja definiše njegov udeo u iskorišćenju ukupnih resursa
- Težine se računaju na duži rok
- Šema se zove FSS i implementirana je na UNIX-ima i nekim klaster *scheduler*-ima (*Maui*)

Faktori FSS-a

1. Osnovni prioritet procesa
2. Njegovo skorašnje iskorišćenje procesora
3. Skorašnje iskorišćenje procesora od strane grupe kojoj proces pripada

Fairshare raspoređivanje

Nastavak

Prioritet procesa j iz grupe k

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$GCPU_k(i) = \frac{GCPU_k(i-1)}{2}$$

$$P_j(i) = Osnova_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

- $CPU_j(i)$ - mera iskorišćenja procesora od strane procesa j u toku intervala i
- $GCPU_k(i)$ - mera iskorišćenja procesora od strane grupe k u toku intervala i
- $P_j(i)$ - prioritet. Manja vrednost - viši prioritet
- $Osnova_j$ - osnovni prioritet procesa j
- W_k - težina grupe k, tako da je $0 \leq W_k \leq 1$

Fairshare raspoređivanje

Primer

Primer

Proces A je u jednoj grupi, a procesi B i C u drugoj. Svaka grupa ima težinski činilac od 0.5. Svi procesi imaju osnovni prioritet od 60. Prioriteti se ponovo izračunavaju jednom u sekundi.

	Process A			Process B			Process C		
Time	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count
0	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0	60	0	0
1	90	30	30	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0 1 2 • • 60
2	74	15 16 17 • • 75	15 16 17 • • 75	90	30	30	75	0	30
3	96	37	37	74	15	15 16 17 • • 75	67	0 1 2 • • 60	15 16 17 • • 75
4	78	18 19 20 • • 78	18 19 20 • • 78	81	7	37	93	30	37
5	98	39	39	70	3	18	76	15	18
	Group 1			Group 2					

Klasično *UNIX* raspoređivanje

Prioritet procesa j iz grupe k

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$P_j(i) = Osnova_j + \frac{CPU_j(i)}{2} + nice_j$$

- $CPU_j(i)$ - mera iskorišćenja procesora od strane procesa j u intervalu i
- $P_j(i)$ - prioritet. Manja vrednost - viši prioritet
- $Osnova_j$ - osnovni prioritet procesa j
- $nice_j$ - faktor za fino podešavanje

Klasično *UNIX* raspoređivanje
Primer

Primer

Procesi A, B i C imaju isti prioritet. Ignoriše se *nice* vrednost.

Time	Process A		Process B		Process C	
	Priority	CPU count	Priority	CPU count	Priority	CPU count
0	60	0, 1, 2, •, •, 60	60	0	60	0
1	75	30	60	0, 1, 2, •, •, 60	60	0
2	67	15	75	30	60	0, 1, 2, •, •, 60
3	63	7, 8, 9, •, •, 67	67	15	75	30
4	76	33	63	7, 8, 9, •, •, 67	67	15
5	68	16	76	33	63	7

Colored rectangle represents executing process

Figure 9.17 **Example of a Traditional UNIX Process Scheduling**