LUENTO 11

VUOROTTAMINEN - YKSI CPU

Stallings, Luku 9

1

Vuorottaminen ja prosessien tilat

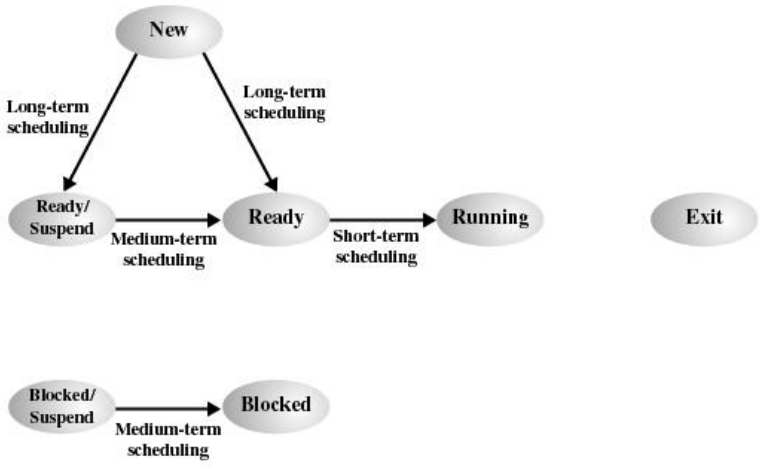


Figure 9.1 Scheduling and Process State Transitions

2

Vuorottamisympäristöt, työkuorma

- Eräajo
  - ajetaan vaikkapa yöllä
  - työn koko osataan arvioida
    - esim. ajetaan joka yö, viikottain, kuukausittain
  - tapahtumaohjattu vuorottaminen OK
    - kunhan kaikki saadaan tehtyä
- Interaktiivinen
  - käyttäjä odottaa vastausta, nopea vastaus hyvä
  - vuorottajalla ei harmainta aavistusta työn kestosta
  - aikaviipaletekniikka
- Reaaliaika
  - aikarajat
  - ohjelmoijakin miettii suorituskykyä ja milloin KJ saa suoritusvuoron

3

Avoin ja suljettu työkuorma

- Deterministinen (suljettu) työkuorma
  - kaikki prosessit tunnetaan
  - heräämistaajuudet tunnetaan
- Avoin työkuorma
  - prosessien joukko vaihtelee ulkoisten tapahtumien perusteella
- Heterogeeninen työkuorma
  - deterministinen + avoin
    - esim. lennon valvonta, lentopintojen ohjaus
    - ”arriving missiles” –poikkeus?
      - vai deterministinen joka 100 ms?
  - eräajo + interaktiivinen + reaaliaika
  - miten sovittaa yhteen
    - jääkö avoimille reaaliaikatöille ”tarpeeksi” aikaa?
    - jääkö KJ:lle tarpeeksi aikaa?

4

Tavoitteita: laatu

Tbl 9.2 [Stal05]

- Samanarvoisille prosesseille sama palvelu
- Priorisointia saa harrastaa
  - turvakontrolli vs. palkanlaskenta
  - KJ prosessit vs. käyttäjän sovellukset
  - reaaliaika prosessit vs. muut
- Interaktiiviset vs. eräajojärjestelmät
- Vastausaika (response time)
  - työ annettu, milloin saadaan vastaus?
- Läpimenoaika (turnaround time)
  - työtä per aikayksikkö
- Ennustettavuus (predictability)
  - *"ei sen näin pitkään pitäisi kestää"*
  - roskien keruu, muu kj-hallinto

Käyttäjän näkökulma

Linux 2.6
O(1) vuoronanto

5

Table 9.2 Scheduling Criteria

User Oriented, Performance Related	
Turnaround time	This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.
Response time	For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.
Deadlines	When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.
User Oriented, Other	
Predictability	A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.

6

System Oriented, Performance Related

Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.

Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.

System Oriented, Other

Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.

Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.

Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

Tbl 9.2 (alaosa)

7

Tavoitteita: suorituskyky

Tbl 9.2 [Stal05]

- Ota järjestelmästä mahdollisimman paljon irti
  - pidä CPU ja erityisesti I/O-laitteet tuottavassa työssä
    - korkea käyttöaste (CPU utilization)
  - tärkeää moniajojärjestelmissä

Ylläpitäjän näkökulma

- Tehokas ja reilu CPU:n käyttö
  - läpimenoaste, läpimenovuo (throughput, work flow)
  - läpimenoaika (turnaround time, response time)
- Reaaliaikajärjestelmä pysyy aikataulussa
  - Aikarajan (deadline) ylitys voi olla harmillista tai vaarallista
    - kuvassa on häiriö, ääni/kuva epäsynkroonisia
    - potilas kuolee, lentokone tippuu, ...

Käyttäjän näkökulma

8

MILLOIN VUOROTETAAN?

9

Milloin?

systeemiin?
- Long-term
  - otetaanko uusi prosessi suoritettavaksi?
  - mahtuuko muistiin? riittääkö swap-tila?

muistiin?
- Medium-term
  - milloin (heittovaihdettu) prosessi muistiin?
  - vapaata muistia?
  - moniajoaste?

suorittimelle?
- Short-term
  - mille prosessille annetaan CPU?

I/O-laitteelle?
- I/O
  - minkä prosessin I/O pyyntö palvellaan ensin?

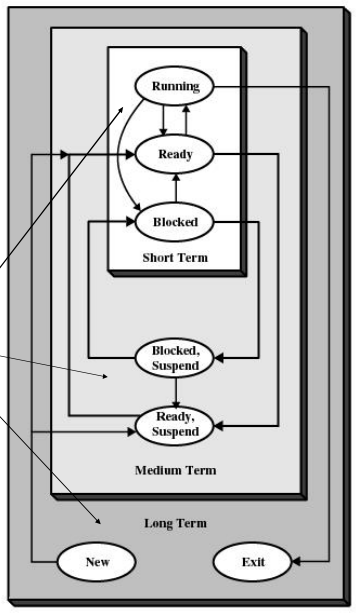


Figure 9.2 Levels of Scheduling

Long-term Scheduling

Fig 9.2 [Stal05]

- Otetaanko uusi työ suoritettavaksi?
  - milloin työstä tulee prosessi?
  - saako koneeseen luoda uuden istunnon?
- Ratkaisevaa: moniajoaste
  - paljon prosesseja à kukin saa harvoin CPU:n
  - jos vähän muistia, niin onko parempi odottaa “long term” vai “medium term”?
  - pyritään takaamaan riittävän tasokas palvelu
  - sopiva suhde: CPU- ja I/O-sidonnaiset työt?
- Milloin?
  - joku prosessi päättynyt / CPU:n käyttöaste pudonnut
- Mikä?
  - First-Come-First-Serviced (FCFS)
  - joskus prioriteetteja: esim. työn koko, I/O-sidonnaisuus

medium term?

Fig 9.3 [Stal05]

11

Medium-Term Scheduling

- Liittyy heittovaihtoon
  - sisäänheiton ajoitus
  - prosessi tilassa Suspend&Ready tai Suspend&Wait
- Milloin muistiin?
  - CPU:n käyttöaste laskenut
  - vapaata muistitilaa runsaasti
- Mikä muistiin?
  - koko (eli odottavan prosessin muistitarve)
  - ulosheittoaika (eli odotusaika levyllä)
  - prioriteetti
- Mikä muistista pois?
  - ei sellainen, jolla tärkeä resurssi hallussa
    - kriittinen vaihe?
  - ei KJ prosessi?

Fig 9.2 [Stal05]

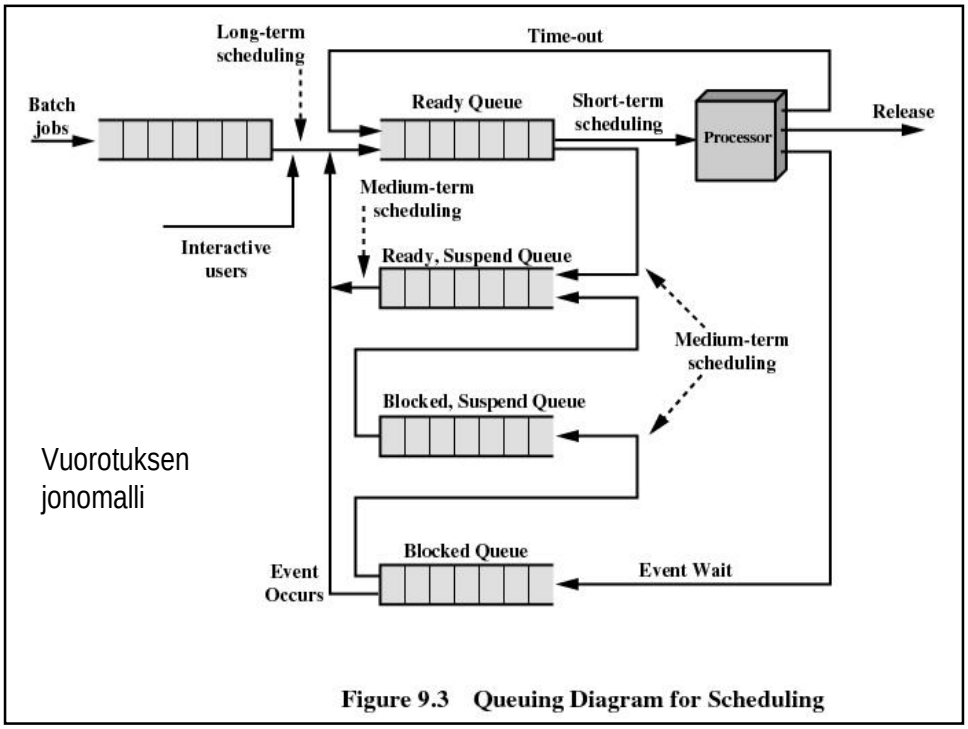
12

Fig 9.2 [Stal05]

Short-Term Scheduling

- CPU:n vuorottaminen (scheduling, dispatching)
  - yleisterminä vuorottaminen tarkoittaa juuri tätä
- Selvästi yleisempi kuin edelliset
- Milloin?
  - keskeytyksen yhteydessä
    - jokainen keskeytys ei aiheuta vuorottamista
  - kun nykyprosessin kyky käyttää suoritinta mennyt
    - joutui Blocked-tilaan: I/O, synkronointi, poissulkeminen
    - poikkeustilanne
    - prosessi käyttänyt oman aikaviipaleensa
  - suuremman prioriteetin työ valmis etenemään
- Kenelle vuoro seuraavaksi?

13



Vuorotuksen jonomalli

Figure 9.3 Queuing Diagram for Scheduling

PRIORITEETTI

15

Prioriteetti

Fig 9.4 [Stal05]

- Suuremman prioriteetin prosessit ensin
  - prioriteetti PCB:ssä / TCB:ssä
- Kullakin prioriteetilla oma Ready-jono
  - lisää aina loppuun
  - haku voi kestää, kun monta jonoa
- Vs. yksi yhteinen Ready-jonossa
  - prioriteetti määrää paikan
  - nopea haku (vain yksi jono)
  - lisäys prioriteetin mukaiseen paikkaan
    - voi olla turhan monimutkaista eli hidasta
- Nälkiintymisvaara
  - vaihteleva prioriteetti torjuu nälkiintymisen?
    - prosessin ikä
    - suoritushistoria
    - vaihtelun rajat?

16

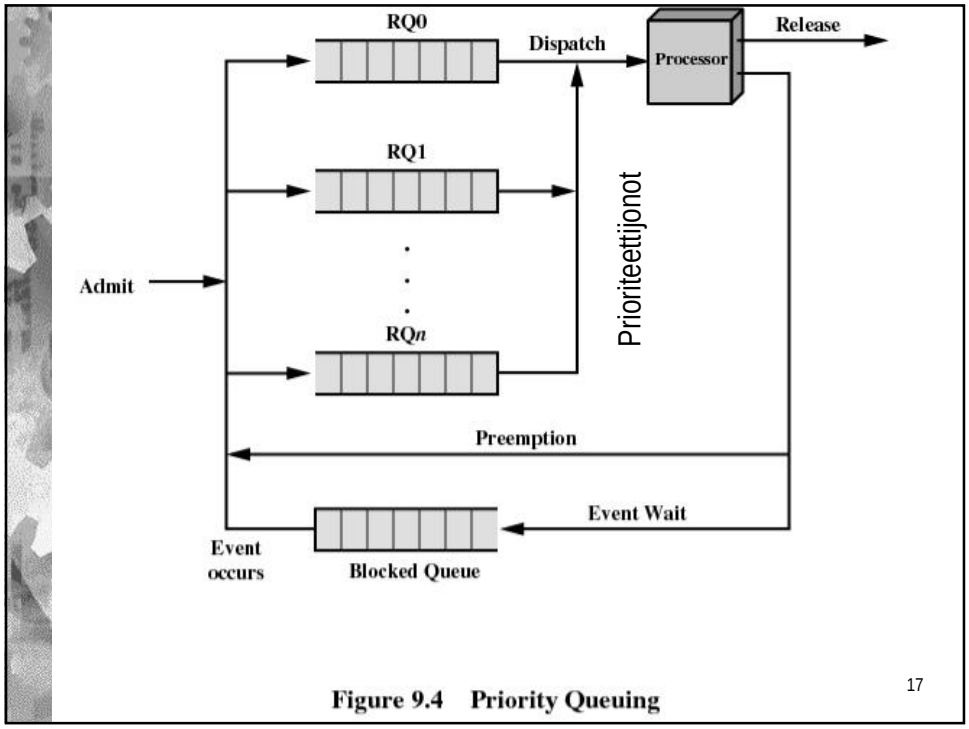
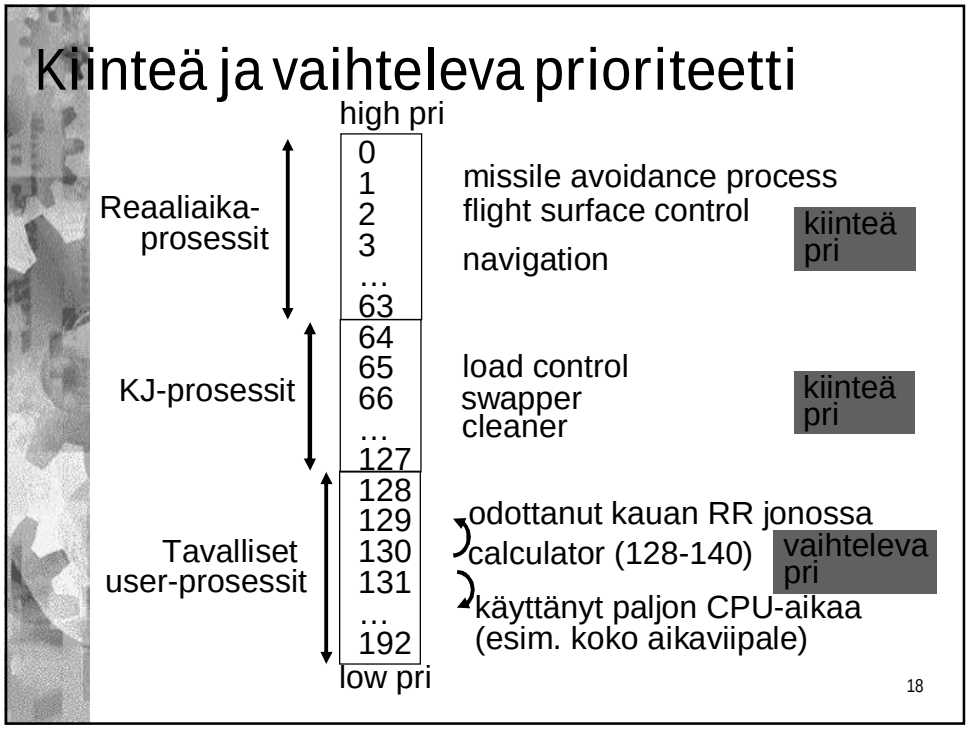


Figure 9.4 Priority Queuing



Milloin vuorotus aktivoituu?

- Nonpreemptive (estävä moniajo)
  - tapahtumaohjattu vuorottaminen
  - prosessi suorituksessa, kunnes se päättyy tai joutuu palvelupyyntönsä vuoksi blocked-tilaan
  - suoritusaikana voi silti olla keskeytyksiä ja KJ työtä!
    - scheduler ei aktivoidu
      - paitsi ehkä KJ-prosesseille?
- Preemptive (keskeytyvä moniajo)
  - keskeyttävä vuorottaminen
    - prosessi ei voi nälkiinnyttää muita
  - suoritus keskeytetään ja prosessi Ready-tilaan, vaikka voisikin käyttää suoritinta
    - aikaviipaletekniikka
    - suuremman prioriteetin prosessi tuli Ready-jonoon
- pre-empt: mennä edelle, ottaa itselleen etuoikeuden nojalla

19

CPU:N VUOROTTAMISALGORITMEJA

20

Algoritmit

- First-Come-First-Served FCFS
- Round Robin RR
- Virtual Round Robin VRR
- Shortest Process Next SPN
- Shortest Remaining Time SRT
- Highest Response Ratio Next HRRN
- Multilevel Feedback feedback

- Fair Share Scheduling FSS

21

Esimerkkiprosessit

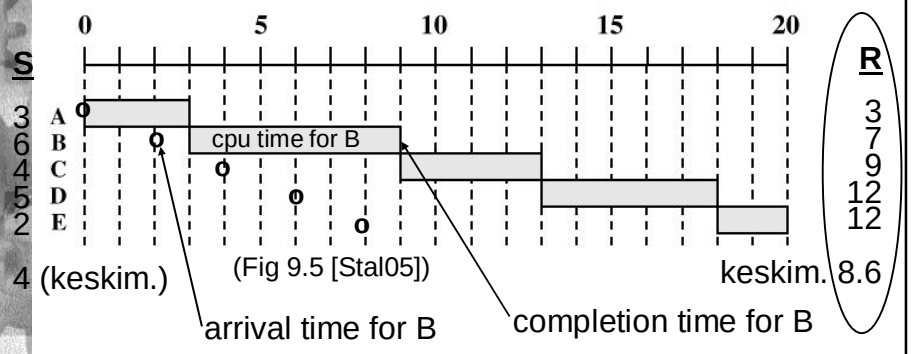
(Tbl 9.4 [Stal05])

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

- Service Time = CPU:ssa kulutettu aika
- Esimerkeissä ei mietitä I/O:n vaikutusta

22

FCFS – First Come First Served



- Eräajo, tapahtumaohjattu, ei prioriteetteja
- Uusi prosessi Ready-jonon hännille
- Kun prosessi luopuu CPU:sta, vuorota seuraava
- Ketä suosii? Ketä ei?

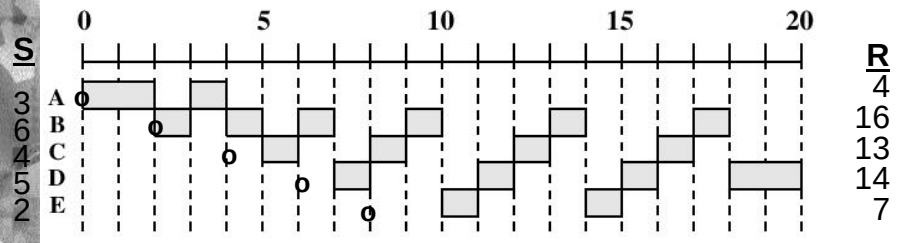
23

FCFS – First Come First Served

- Läpimenoaika — Fig 9.5 [Stal05]
  - riippuu suoritusjärjestyksestä, muiden koosta sekä CPU-sidonnaisuudesta
- Pienikin prosessi voi joutua odottamaan
  - läpimenoajasta valtava osa odotusta — Miksi?
- Suosii CPU-sidonnaisia
  - muille voi tulla pitkä odotusaika
  - I/O-laitteet ehkä turhaan jouten
    - I/O kuitenkin pullonkaula
- Järkevää ottaa mukaan prioriteetit
  - prioriteetin perusta?
    - prosessin koko (suoritin aika)?
    - I/O-sidonnaisuus?

24

RR – Round Robin



(Fig 9.5 [Stal05])

- Aikaviipaletekniikka (q=1), keskeytyvä (preemptive)
- Kukin Ready-prosessi saa vuorollaan aikaviipaleen
- Vuorottaminen, kun viipale käytetty
  tai kun prosessi joutuu Blocked-tilaan

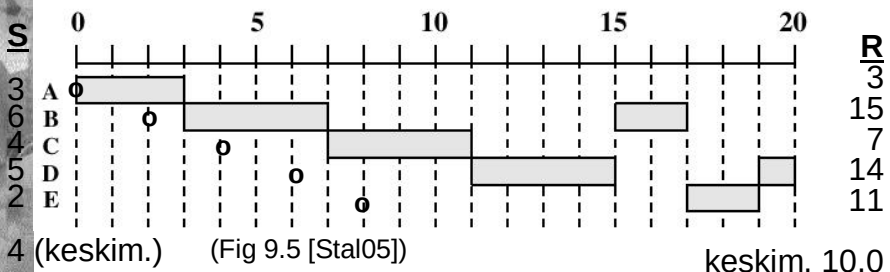
25

RR – Round Robin

- Aikaviipaleen pituus — Fig 9.5 [Stal05]
  - lyhyt: prosessin vaihdot vievät CPU-aikaa
  - pitkä: interaktiivisen työn vastausaika lyhyt, jos yksi aikaviipale riittää
- Suosii hieman CPU-sidonnaisia
  - I/O-sidonnainen ei ehkä käytä koko viipaletta
  - I/O-sidonnainen saa suhteessa harvemmin CPU:n - miksi?
- Virtual RR
  - ready-jonon apujono (Auxiliary Ready Queue), jonne I/O-odotuksesta — Fig 9.7 [Stal05]
    - prioriteetti I/O sidonnaisilla
  - aja ensin apujonossa olevat prosessit
  - aikaviipale vain edellisellä kerralla käyttämättä jäänyt osa, sitten normaaliin Ready-jonoon

26

RR – aikaviipaleen vaikutus



- Aikaviipaletekniikka q = 4
  - prosessin vaihtoaikaa ei huomioitu!
- Suosiiko pitkä aikaviipale cpu- vai I/O-sidonnaisia?
- Suosiiko pitkä aikaviipale lyhyitä vai pitkiä töitä?

27

Virtual RR:n jonomalli

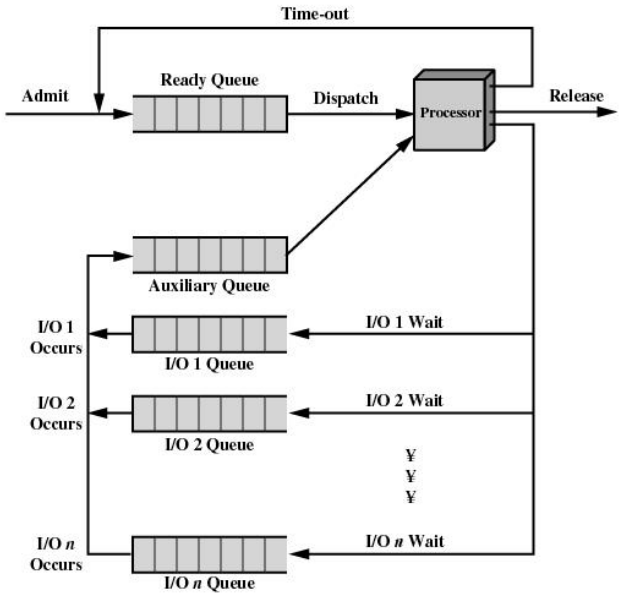
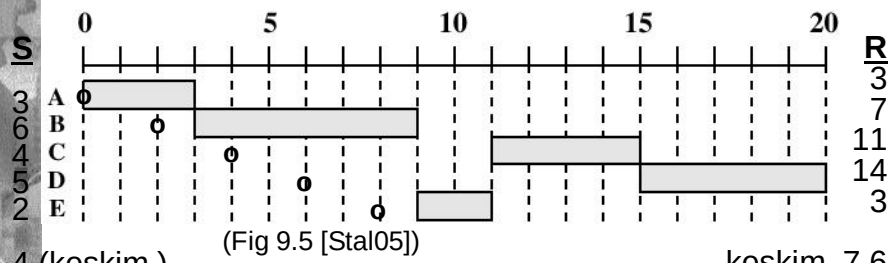


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

28

SPN – Shortest Process Next



(Fig 9.5 [Stal05])

4 (keskim.) keskim. 7.6

- Tapahtumaohjattu (siis non-preemptive)
- Vuorota se, joka käyttää lyhimmän ajan CPU:ta kerrallaan
  - I/O-sidonnaiset ensin
  - mistä tietää?

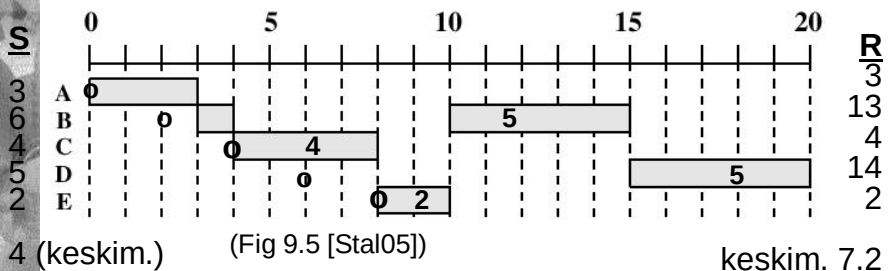
29

SPN – Shortest Process Next

- Nälkiintymisvaara
  - iso jää aina pienten jalkoihin
- Isojen läpimenoaika vaikea ennustaa
- Erätyö: käynnistäjä arvioi työn kestoajan
  - jos työ laitettiin väärään eräajoluokkaan, KJ saattaa katkaista työn
    - miksi? (kulutti liikaa aikaa estimaattiin nähden)
    - käynnistettävä uudelleen isompien luokassa
- Interaktiivinen: KJ laskee keskim. CPU:n käyttöaikaa
  - painottaa viimeksi havaittuja aikoja (T_{n-1})
  - estimoi $S_n^{estim} = \alpha T_{n-1} + (1 - \alpha) S_{n-1}^{estim}$ esim $\alpha = 0.8$
- Ei sovellu osituskäyttöympäristöön

30

SRT – Shortest Remaining Time

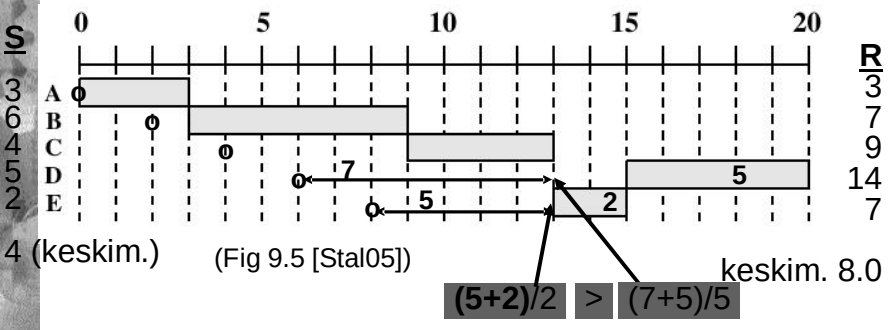


(Fig 9.5 [Stal05])

- Aikaviipaleversio edellisestä (keskeytyvä)
  - tilanne arvioidaan uudelleen joka aikaviipaleelle
- Arvioitava prosessin jäljellä oleva ajantarve
- Ei sovi interaktiiviseen ympäristöön

31

HRRN – Highest Response Ratio Next



(Fig 9.5 [Stal05])

- Tapahtumaohjattu (siis non-preemptive)
- Minimoi läpimenoaikaa (huomioi historia)
- Vuorota se, jolla huonoin suhteellinen vasteaika,
  ts. se, jolla suurin suhdeluku:

$$\text{response ratio} = \frac{\text{time spent waiting CPU + expected service time}}{\text{expected service time}}$$

32

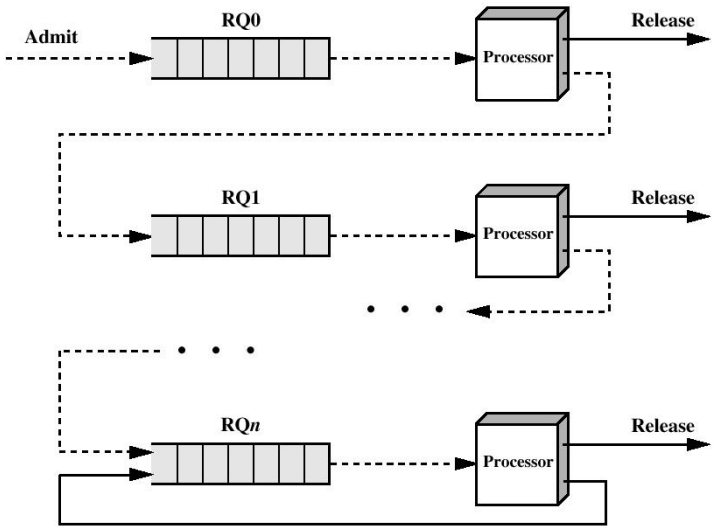
HRRN – Highest Response Ratio Next

- Suosii hieman lyhyitä töitä
  - ei silti nälkiintymisvaaraa
- Dynaaminen prioriteetti
  - ready-jonossa odottelu kasvattaa prioriteettia
- Jäljellä olevaa aikaa ei voi tietää
  - arviot menneisyyden perusteella
  - käyttäjän antama arvio työn koosta
    - ei sovi interaktiiviseen ympäristöön

33

Multilevel Feedback

(Fig 9.10 [Stal05])



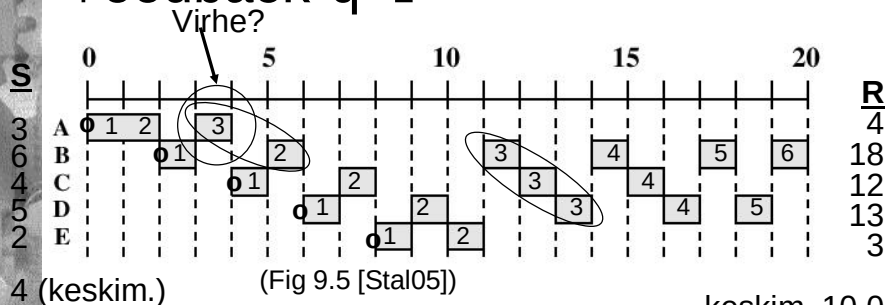
34

Feedback

- Dynaaminen prioriteetti
  - 'Rankaisee' pitkään pörränneitä prosesseja, aikaviipaloitu
- Useita Ready-jonoja
  - RQ0: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
  - RQ2..RQn-1: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
  - RQn: pura aikaviipaleittain, RR, pidä samassa jonossa
- Prosessi kulkeutuu lopulta RQn-jonoon, josta se aikanaan valmistuu
- Nälkiintymisvaara
  - vuorottaa alemmassa jonossa olevat aina ensin
- Useita variaatioita
  - esim. alemmissa jonoissa pitempi aikaviipale (esim. 1, 2, 4, …)
  - palaa blocked-tilasta samaan jonoon

35

Feedback q=1

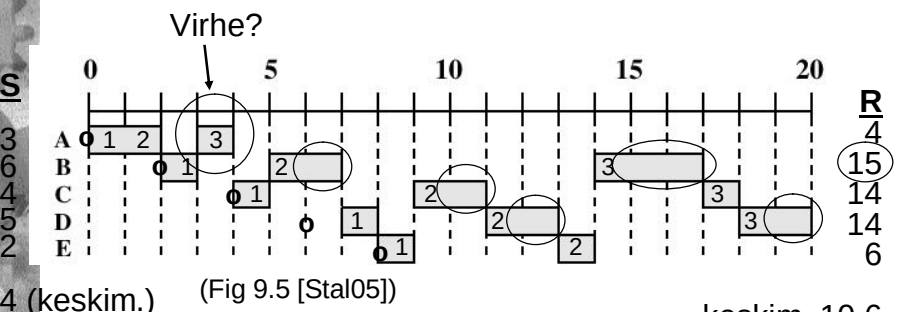


(Fig 9.5 [Stal05])

- Jonot RQ0, RQ1, RQ2, …
- Aikaviipale q=1
- Ei vaadi etukäteisarvioita CPU-ajan tarpeesta
- Pitkät työt voivat kestää kauan

36

Feedback $q=2^i$



(Fig 9.5 [Stal05])

- Jonot RQ0, RQ1, RQ2, ..., RQi
- Aikaviipale $q=2^i$
- Vaikutus, vrt. q=1

Fig 9.5 [Stal05]

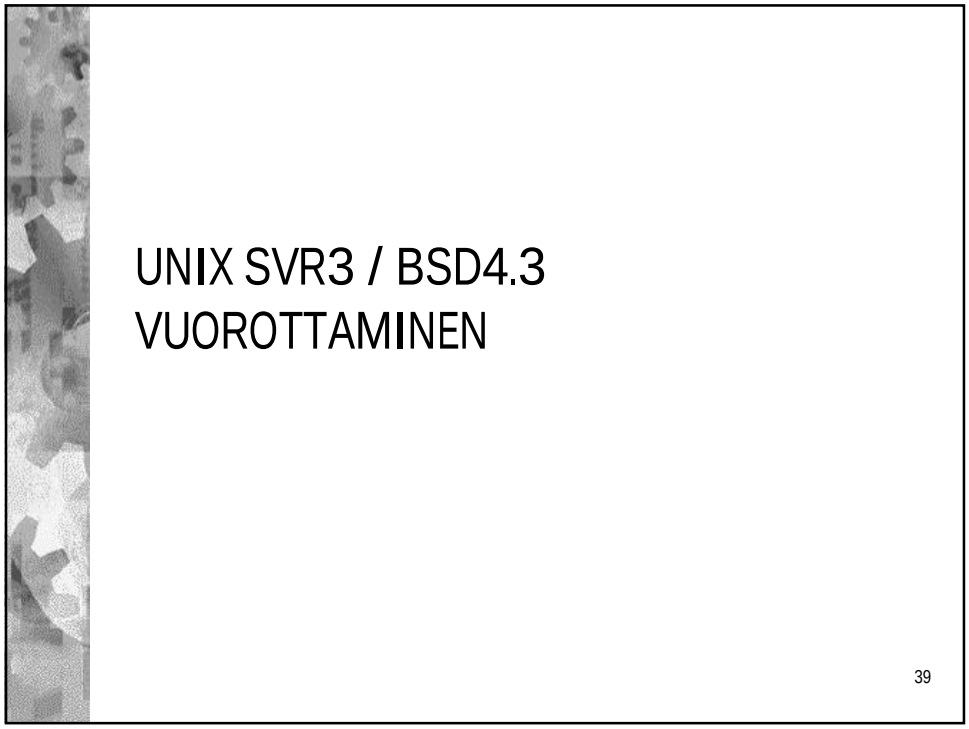
37

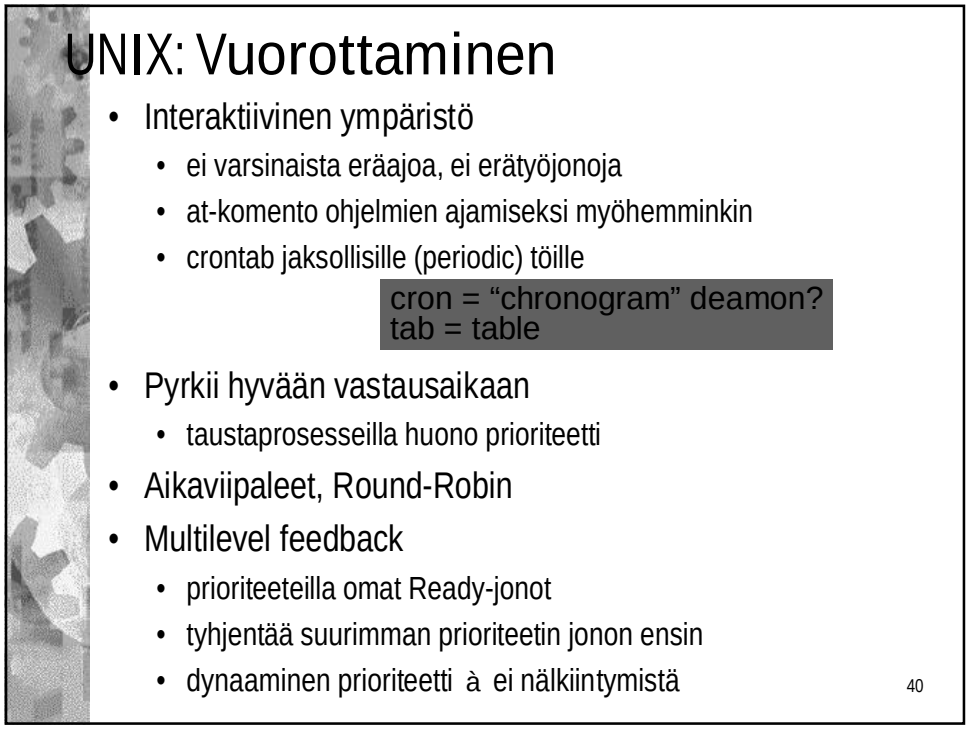
Yhteenveto

Tbl 9.3 [Stal05]

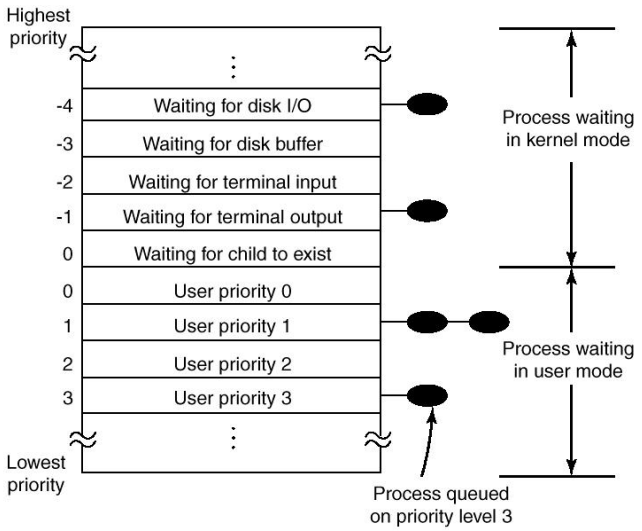
	Selection Function	Decision Mode	Throughput	Response Time	Overhead	Effect on Processes	Starvation
FCFS	max[w]	Nonpreemptive	Not emphasized	May be high, especially if there is a large variance in process execution times	Minimum	Penalizes short processes; penalizes I/O bound processes	No
Round Robin	constant	Preemptive (at time quantum)	May be low if quantum is too small	Provides good response time for short processes	Minimum	Fair treatment	No
SPN	min[s]	Nonpreemptive	High	Provides good response time for short processes	Can be high	Penalizes long processes	Possible
SRT	min[$s-e$]	Preemptive (at arrival)	High	Provides good response time	Can be high	Penalizes long processes	Possible
HRRN	max$\left(\frac{w+s}{s}\right)$	Nonpreemptive	High	Provides good response time	Can be high	Good balance	No
Feedback	(see text)	Preemptive (at time quantum)	Not emphasized	Not emphasized	Can be high	May favor I/O bound processes	Possible

w = time spent ~~in system so far~~, waiting ~~and executing~~
e = time spent in execution so far
s = total service time required by the process, including e


UNIX SVR3 / BSD4.3
VUOROTTAMINEN
39


UNIX: Vuorottaminen
• Interaktiivinen ympäristö
• ei varsinaista eräajoa, ei erätyöjonoja
• at-komento ohjelmien ajamiseksi myöhemminkin
• crontab jaksollisille (periodic) töille
cron = “chronogram” deamon?
tab = table
• Pyrkii hyvään vastausaikaan
• taustaprosesseilla huono prioriteetti
• Aikaviipaleet, Round-Robin
• Multilevel feedback
• prioriteeteilla omat Ready-jonot
• tyhjentää suurimman prioriteetin jonon ensin
• dynaaminen prioriteetti à ei nälkiintymistä
40

UNIX: Ready-jonot



(Fig 10-11 [Tane01])

41

UNIX: Prioriteetti

- Kiinteä perusprioriteetti sekä nice-arvo
  - käyttäjä voi pienentää prioriteettia nice-komennolla
  - pitää prioriteetin siististi tietyllä arvoalueella
- Laskee uuden prioriteetin sekunnin välein
  - iso aikaviipale!
- CPU:n käyttö vaikuttaa uuteen arvoon
  - käyttö: prioriteetti putoaa
  - odottaa kauan: prioriteetti kasvaa
- Suosii I/O-sidonnaisia prosesseja
  - tavoite: I/O-laitteiden tehokas työllistäminen

42

UNIX: multilevel feedback

Fig 9.17 [Stal05]

- CPU_usage = CPU:n käyttö äskettäin
  - laskuri PCB:ssä
- Älä rankaise liikaa aiemmasta käytöstä
  - puolita ennen prioriteetin laskentaa, ja sitten taas

CPU_usage = CPU_usage/2
Pri = Base + (CPU_usage/2)+ Nice
(jos ei cpu_usage = 0, niin ei muutosta)

- Pieni arvo = suuri prioriteetti
- Esimerkki
  - Base=60, Nice = 0
  - päivitä counter (CPU_usage) 60 kertaa/sek
    - kellolaite keskeytys 16.7 ms välein?
    - otanta: kenellä suoritin keskeytyshetkellä?
  - päivitä prioriteetti sekunnin välein

43

Time	Process A Priority	Process A CPU Count	Process B Priority	Process B CPU Count	Process C Priority	Process C 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 Traditional UNIX Process Scheduling

HUOM:
5. Ed [Stal05] joissakin painoksissa väritykset virheellisesti

44

Fair-Share Scheduling

- Tutki myös kuka prosessin omistaa (owner)
  - ettei hulivili voi tukkia järjestelmää…
- Käsittele yhden käyttäjän prosesseja / säikeitä ryhmänä
  - ryhmän vaikutus “nice” termin asemesta
  - vuorottelu edelleen prosessi- / säietasolla
  - pidettävä myös kirjaa paljonko ryhmä saanut CPU:n kokonaisajasta (GCPU_counter)
  - ryhmällä voi olla paino W, joka määrää millaisen osuuden se saa koko (cpu-aika) kakusta
- Käytössä useissa UNIX-järjestelmissä
  - HP-UX, IBM AIX WLM, Sun Solaris SRM
  - ryhmä voi perustua käyttäjään tai prosessiin tai sovellukseen

45

FSS - Fair-Share Scheduling

- Prioriteetin määrittäminen

```
CPU_counter  = CPU_counter/2
GCPU_counter = GCPU_counter/2
Pri = Base + CPU_counter/2 +
GCPU_counter/(4*W_group)
```

iso W_{group}
à pieni painoarvo
GCPU_counter:lla
à pieni Pri arvo
à iso prioriteetti

Esimerkissä

Kts. Fig 9.16 [Stal05]

- Base = 60
- W_A = 0.5 ja W_{B+C} = 0.5 (W_A + W_{B+C} = 1)
- päivitä laskurit 60 kertaa sekunnissa
- päivitä prioriteetti sekunnin välein

46

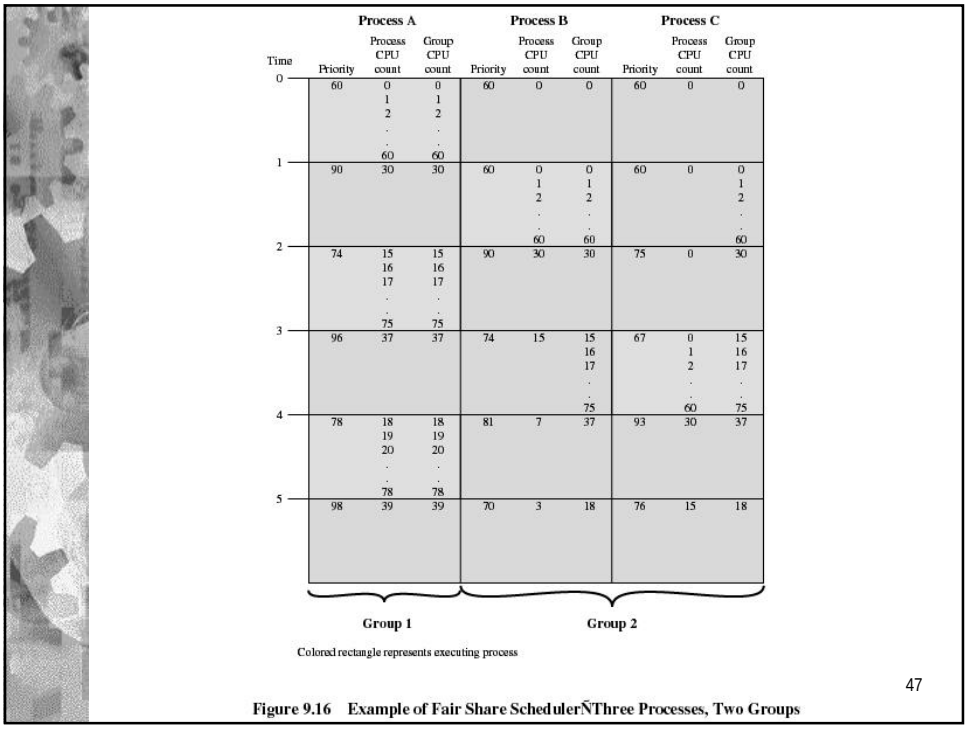


Figure 9.16 Example of Fair Share SchedulerÑThree Processes, Two Groups