

# Käyttöjärjestelmät II

## VUOROTTAMINEN

## YKSI CPU

Stallings, Ch 9 [Stal 05]

# Mitä KJ-I:ssä / KJ-II:ssa?

## KJ-I

- n Ready-jono, valitse ensimmäinen
- n Aikaviipaleteknikka (round-robin)
- n Prioriteetitkin mainittiin

## Seuraavaksi KJ-II:ssa

- n Vuorottamisen tasot
- n CPU:n vuorottamisalgoritmeja
- n Vuorottaminen UNIXissa
- n Yhden CPU:n ympäristössä (Ch 9)
- n SMP:ssä ja reaaliaikajärjestelmissä (Ch 10)

luento 7

luento 8

# Vuorottamisympäristöt, työkuorma

## n Eräajo

- u ajetaan vaikkapa yöllä
- u työn koko osataan arvioida
  - F esim. ajetaan joka yö, viikottain, kuukausittain
- u tapahtumaohjattu vuorottaminen OK
  - F kunhan kaikki saadaan tehtyä

## n Interaktiivinen

- u käyttäjä odottaa vastausta, nopea vastaus hyvä
- u vuorottajalla ei harmainta aavistusta työn kestosta
- u aikaviipaletekniikka

## n Reaaliaika

- u aikarajat
- u ohjelmoijakin miettii suorituskykyä ja milloin KJ saa suoritusvuoron

# Avoim ja suljettu työkuorma

## n Deterministinen (suljettu) työkuorma

- u kaikki prosessit tunnetaan
- u heräämistäajuudet tunnetaan

## n Avoin työkuorma

- u prosessien joukko vaihtelee ulkoisten tapahtumien perusteella

## n Heterogeeninen työkuorma

- u deterministinen + avoin
  - F esim. lennon valvonta, lentopintojen ohjaus
  - F "arriving missiles" –poikkeus?
    - vai deterministinen joka 100 ms?
- u eräajo + interaktiivinen + reaaliaika
- u miten sovittaa yhteen
  - F jääkö avoimille reaaliaikatöille "tarpeeksi" aikaa?
  - F jääkö KJ:lle tarpeeksi aikaa?

# Tavoitteita: laatu

Tbl 9.2 [Stal05]

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

Linux 2.6  
O(1) vuoronanto

# Tavoitteita: suorituskyky

Tbl 9.2 [Stal05]

- n **Ota järjestelmästä mahdollisimman paljon irti**
  - u pidä CPU ja erityisesti I/O-laitteet tuottavassa työssä
    - F korkea käyttöaste (CPU utilization)
  - u tärkeää moniajojärjestelmissä
- n **Tehokas ja reilu CPU:n käyttö**
  - u läpimenoaste, läpimenvuo (throughput, work flow)
  - u läpimenoaika (turnaround time, response time)
- n **Reaaliaikajärjestelmä pysyy aikataulussa**
  - u deadlinen ylitys voi olla harmillista tai vaarallista
    - F kuvassa on häiriö, ääni/kuva epäsynkroonisia
    - F potilas kuolee, lentokone tippuu, ...

# Käyttöjärjestelmät II

## MILLOIN VUOROTETAAN?

# Milloin?

Fig 9.2 [Stal05]

Fig 9.3 [Stal05]

## n Long-term

systemiin?

- u otetaanko uusi prosessi suoritettavaksi?
- u mahtuuko muistiin? riittääkö swap-tila?

## n Medium-term

muistiin?

- u milloin (heittovaihdettu) prosessi muistiin?
- u vapaata muistia?
- u moniajoaste?

## n Short-term

suorittimelle?

- u mille prosessille annetaan CPU?

## n I/O

I/O-laitteelle?

- u minkä prosessin I/O pyyntö palvellaan ensin?



# Long-term Scheduling

Fig 9.2 [Stal05]

## n Otetaanko uusi työ suoritettavaksi?

- u milloin työstä tulee prosessi?
- u saako koneeseen luoda uuden istunnon?

## n Ratkaisevaa: moniajoaste

medium term?

- u paljon prosesseja → kukin saa harvoin CPU:n
- u jos vähän muistia, niin onko parempi odottaa “long term” vai “medium term”?
- u pyritään takaamaan riittävän tasokas palvelu
- u sopiva suhde: CPU- ja I/O-sidonnaiset työt?

Fig 9.3 [Stal05]

## n Milloin?

- u joku prosessi päättynyt
- u CPU:n käyttöaste pudonnut

## n Mikä?

- u First-Come-First-Served (FCFS)
- u joskus prioriteetteja: esim. työn koko, I/O-sidonnaisuus

# Medium-Term Scheduling

## n Liittyy heittovaihtoon

- u sisäänheiton ajoitus
- u prosessi tilassa Suspend&Ready tai Suspend&Wait

Fig 9.2 [Stal05]

## n Milloin muistiin?

- u CPU:n käyttöaste laskenut
- u vapaata muistitilaa runsaasti

## n Mikä muistiin?

- u koko (eli odottavan prosessin muistitarve)
- u ulosheittoaika (eli odotusaika levyllä)
- u prioriteetti

## n Mikä muistista pois?

- u ei sellainen, jolla tärkeä resurssi hallussa
  - F kriittinen vaihe?
- u ei KJ prosessi?

# Short-Term Scheduling

Fig 9.2 [Stal05]

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

# Käyttöjärjestelmät II

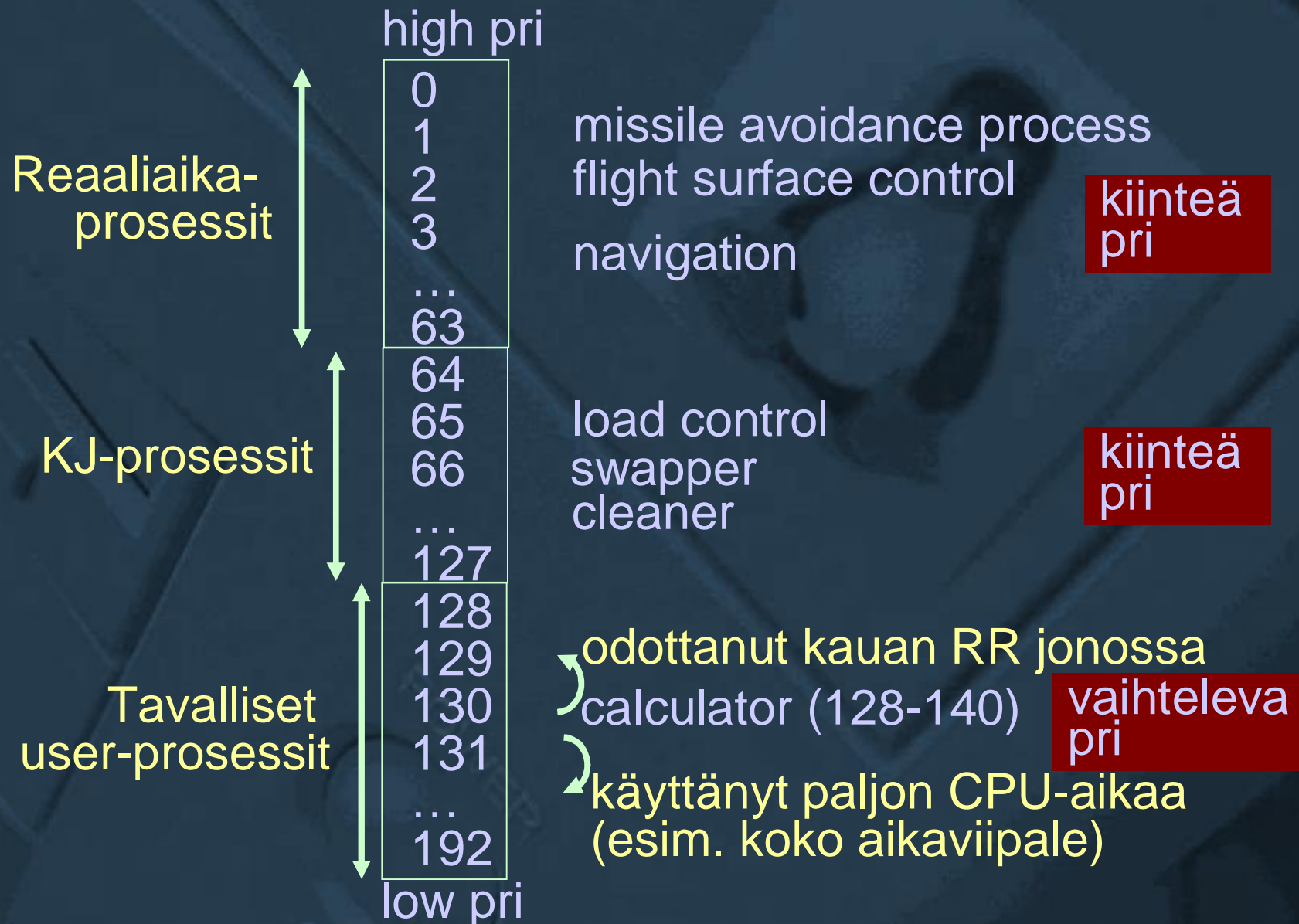
## PRIORITEETTI

# Prioriteetti

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

Fig 9.4 [Stal05]

# Kiinteä ja vaihteleva prioriteetti



# Milloin vuorotus aktivoituu?

## n Nonpreemptive

- u tapahtumaohjattu vuorottaminen
- u prosessi suorituksessa, kunnes se päättyy tai joutuu palvelupyyntönsä vuoksi blocked-tilaan
- u suoritusaikana voi silti olla keskeytyksiä ja KJ työtä!
  - F scheduler ei aktivoidu
    - paitsi ehkä KJ-prosesseille?

## n Preemptive

- u keskeyttävä vuorottaminen
  - F prosessi ei voi nälkiinnyttää muita
- u suoritus keskeytetään ja prosessi Ready-tilaan, vaikka voisikin käyttää suoritinta
  - F aikaviipaletekniikka
  - F suuremman prioriteetin prosessi tuli Ready-jonoon

## n pre-empt: mennä edelle, ottaa itselleen etuoikeuden nojalla

# Käyttöjärjestelmät II

## CPU:N VUOROTTAMIS- ALGORITMEJA



# Algoritmit

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

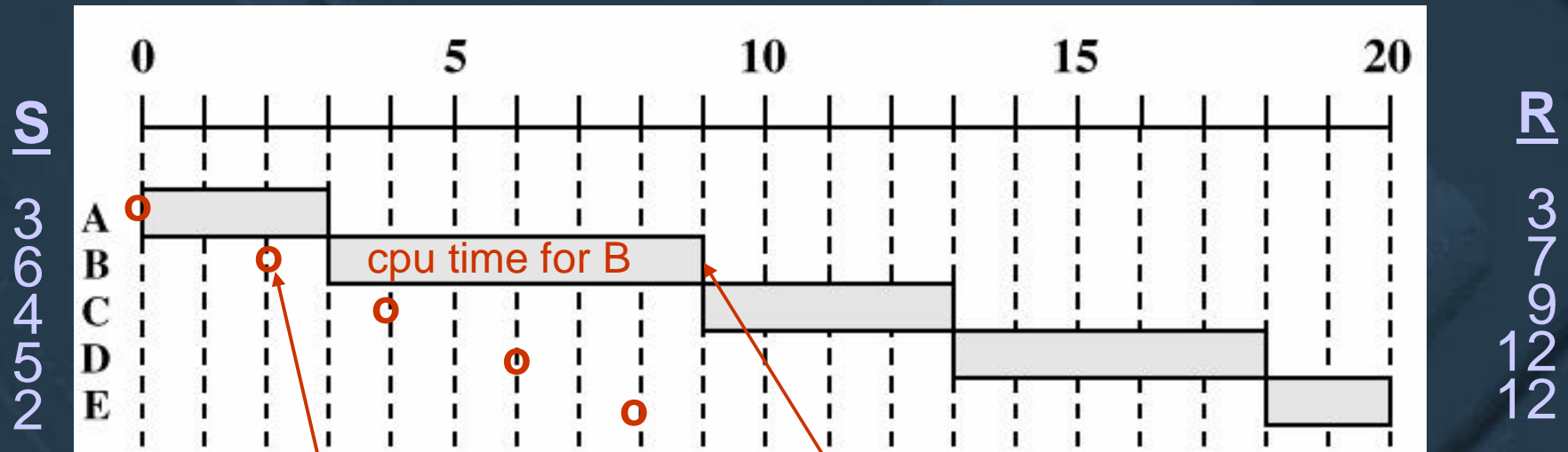
# Esimerkkiprosessit

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

(Tbl 9.4 [Stal05])

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

# FCFS – First Come First Served



4 (keskim.) (Fig 9.5 [Stal05]) keskim. 8.6  
 arrival time for B completion time for B

- n Eräajo, tapahtumaohjattu, ei prioriteetteja
- n Uusi prosessi Ready-jonon hännille Tbl 9.4 [Stal05]
- n Kun prosessi luopuu CPU:sta, vuorota seuraava
- n Ketä suosii? Ketä ei?

# FCFS – First Come First Served

## n Läpimenoaika

- u riippuu suoritusjärjestyksestä,  
muiden koosta sekä CPU-sidonnaisuudesta

Fig 9.5 [Stal05]

## n Pienikin prosessi voi joutua odottamaan

- u läpimenoajasta valtava osa odotusta

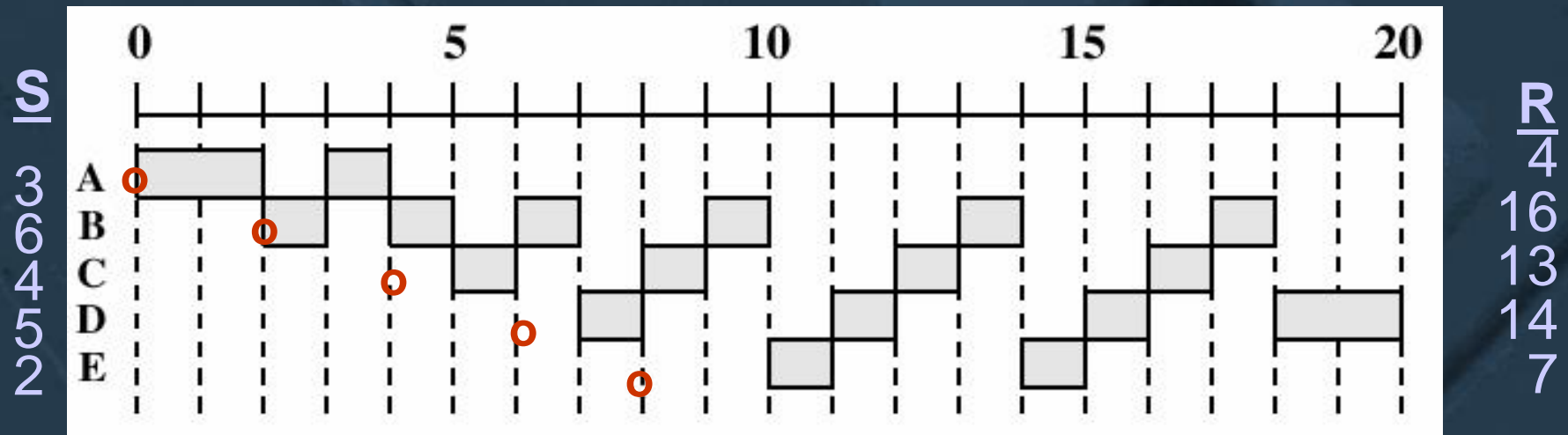
## n Suosii CPU-sidonnaisia

- u muille voi tulla pitkä odotusaika
- u I/O-laitteet ehkä turhaan jouten
  - F I/O kuitenkin pullonkaula

## n Järkevää ottaa mukaan prioriteetit

- u prioriteetin perusta?
  - F prosessin koko (suoritin aika)?
  - F I/O-sidonnaisuus?

# RR – Round Robin



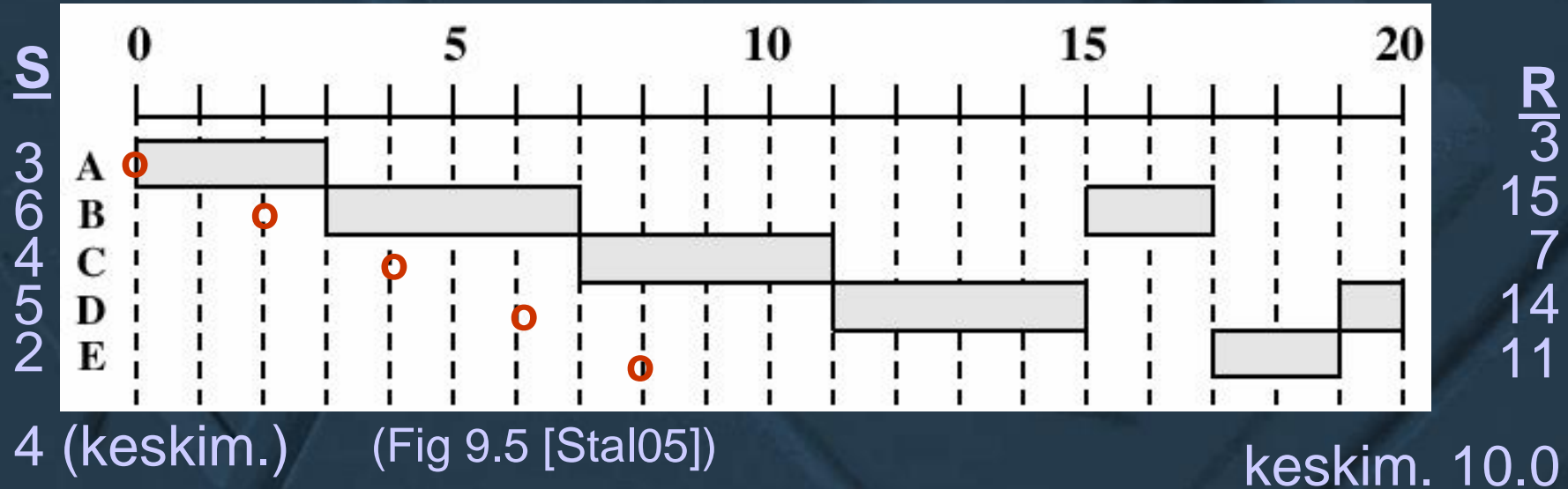
4 (keskim.)

(Fig 9.5 [Stal05])

keskim. 10.8

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

# RR – aikaviipaleen vaikutus



n **Aikaviipaletekniikka  $q = 4$**

u prosessin vaihtoaikaa ei huomioitu!

n **Suosiiko pitkä aikaviipale cpu- vai I/O-sidonnaisia?**

n **Suosiiko pitkä aikaviipale lyhyitä vai pitkiä töitä?**

# RR – Round Robin

## n Aikaviipaleen pituus

- u lyhyt: prosessin vaihdot vievät CPU-aikaa
- u pitkä: interaktiivisen työn vastausaika lyhyt, jos yksi aikaviipale riittää

Fig 9.5 [Stal05]

## n Suosii hieman CPU-sidonnaisia

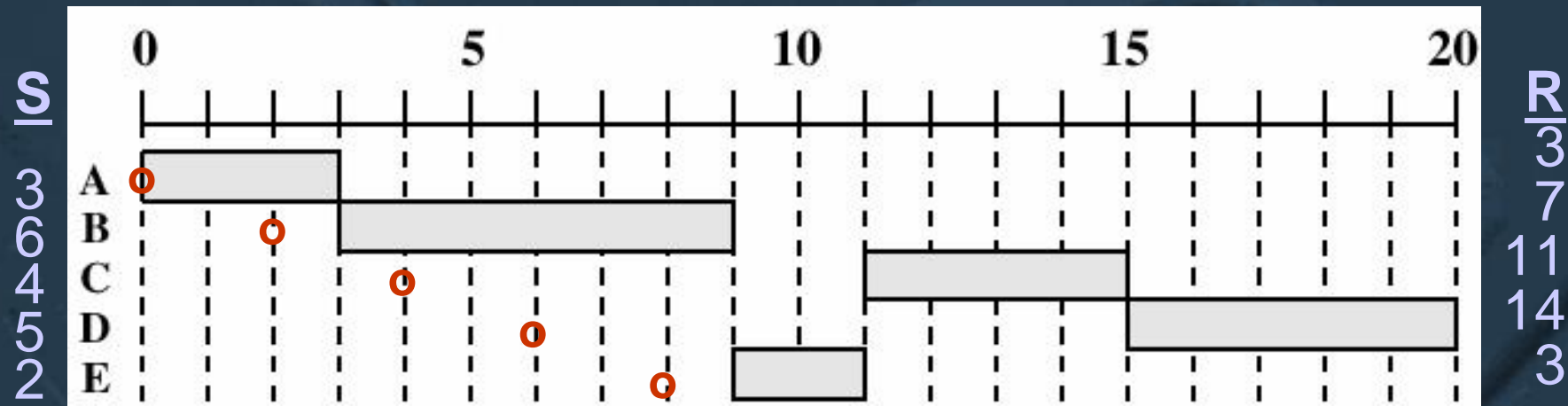
- u I/O-sidonnainen ei ehkä käytä koko viipaletta
- u I/O-sidonnainen saa suhteessa harvemmin CPU:n
  - F miksi?

## n Virtual RR

- u ready-jonon apujono (Auxiliary Ready Queue), jonne I/O-odotuksesta
  - F prioriteetti I/O sidonnaisilla
- u aja ensin apujonossa olevat prosessit
- u aikaviipale vain edellisellä kerralla käyttämättä jäänyt osa, sitten normaaliin Ready-jonoon

Fig 9.7 [Stal05]

# SPN – Shortest Process Next



(Fig 9.5 [Stal05])

4 (keskim.)

keskim. 7.6

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



# SPN – Shortest Process Next

## n Nälkiintymisvaara

- u iso jää aina pienten jalkoihin

## n Isojen läpimenoaika vaikea ennustaa

## n Erätyö: käynnistäjä arvioi työn kestoajan

- u jos työ laitettiin väärään eräajoluokkaan, KJ saattaa katkaista työn
  - F miksi? (kulutti liikaa aikaa estimaattiin nähden)
  - F käynnistettävä uudelleen isompien luokassa

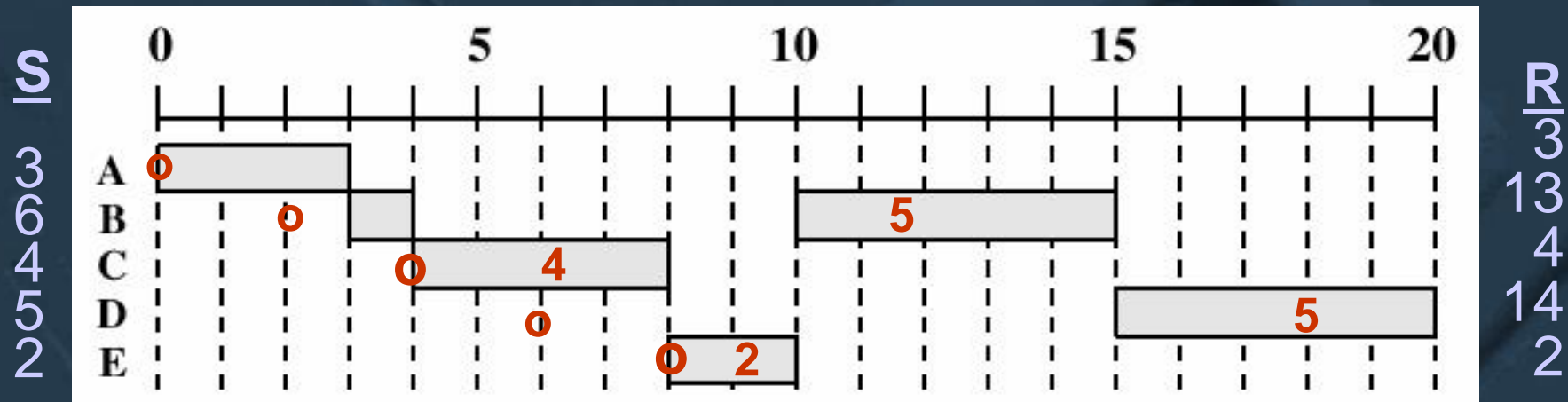
## n Interaktiivinen: KJ laskee keskim. CPU:n käyttöaika

- u painottaa viimeksi havaittuja aikoja ( $T_{n-1}$ )

- u estimoi  $S_n^{\text{estim}} = \alpha T_{n-1} + (1 - \alpha) S_{n-1}^{\text{estim}}$  esim  $\alpha = 0.8$

## n Ei sovellu osituskäyttöympäristöön

# SRT – Shortest Remaining Time



4 (keskim.)

(Fig 9.5 [Stal05])

keskim. 7.2

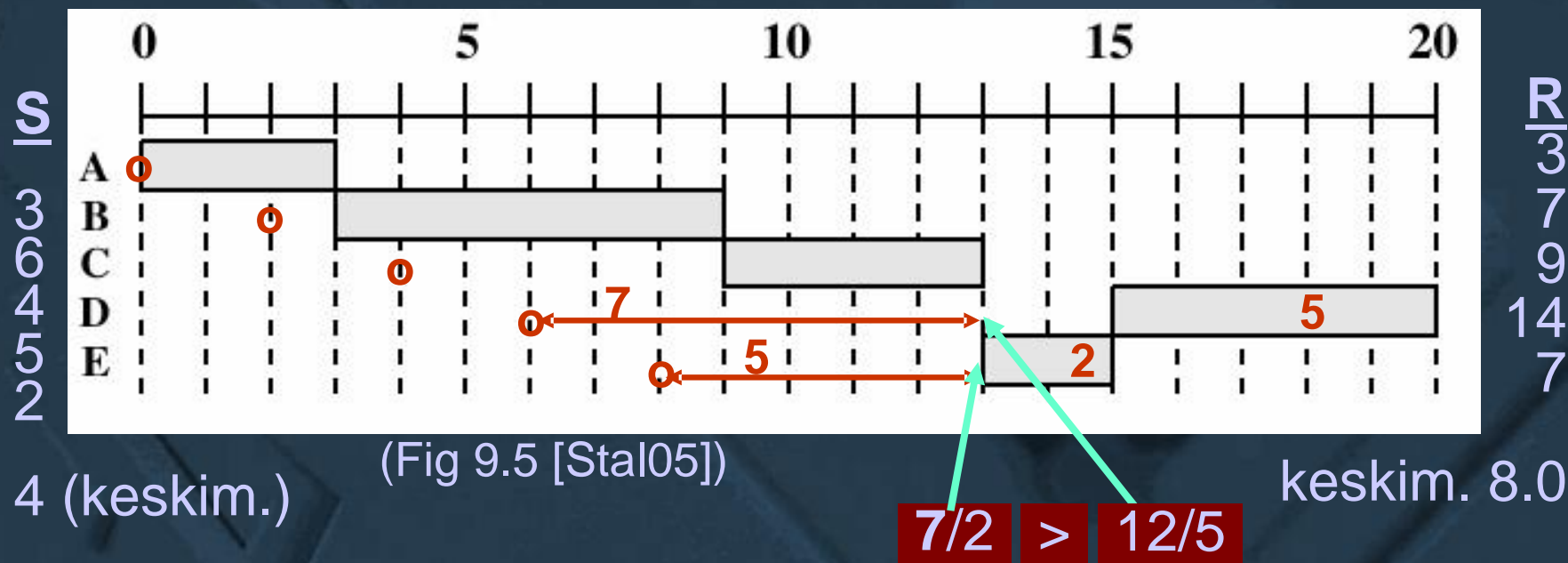
n **Aikaviipaleversio edellisestä (keskeytyvä)**

u tilanne arvioidaan uudelleen joka aikaviipaleelle

n **Arvioitava prosessin jäljellä oleva ajantarve**

n **Ei sovi interaktiiviseen ympäristöön**

# HRRN – Highest Response Ratio Next



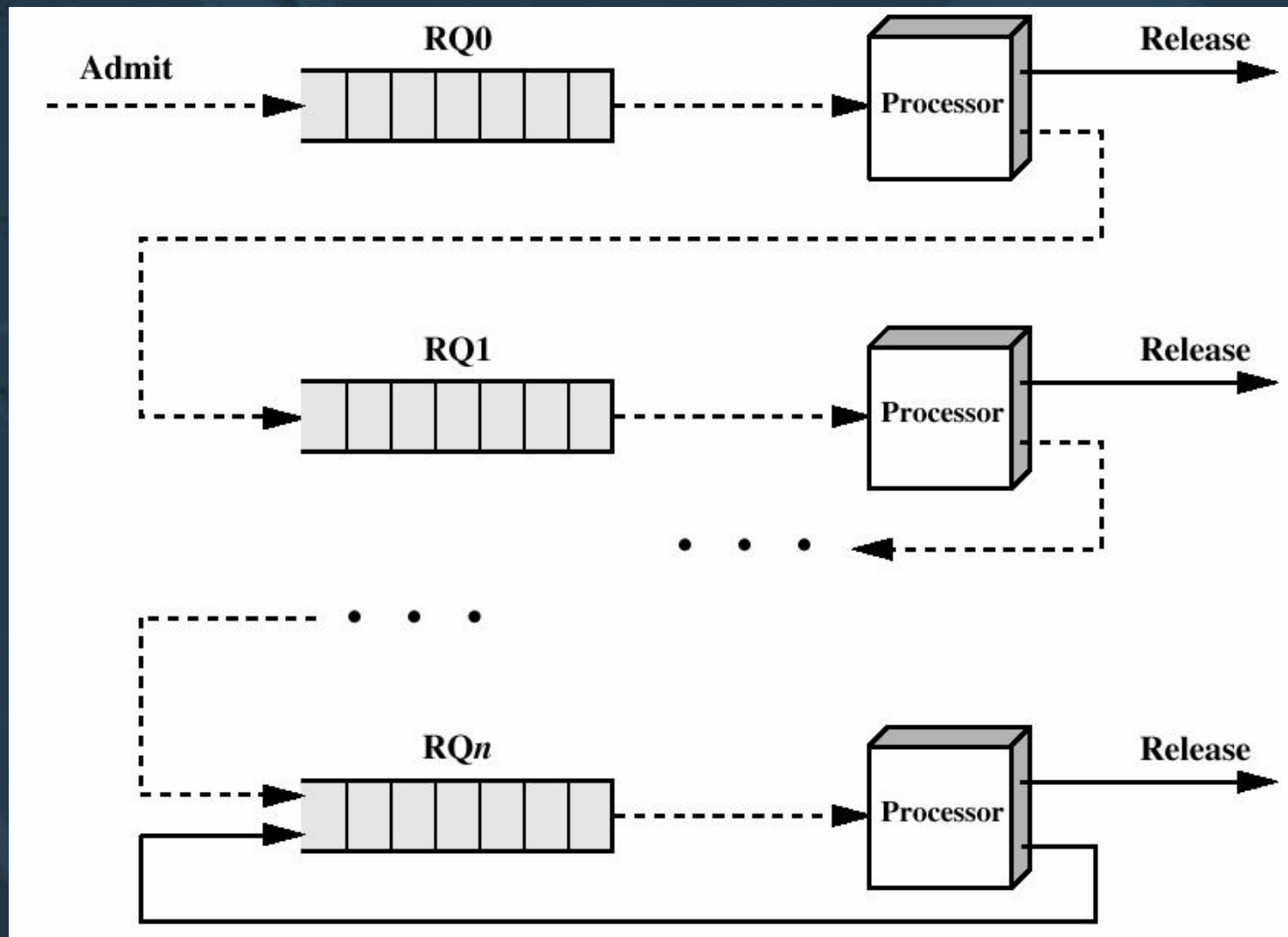
- n Tapahtumaohjattu (siis non-preemptive)
- n Minimoi läpimenoaikaa (huomioi historia)
- n Vuorota se, jolla huonoin suhteellinen vasteaika, ts. se, jolla suurin suhdeluku:

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

# HRRN – Highest Response Ratio Next

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

# Multilevel Feedback



(Fig 9.10 [Stal05])

# Feedback

## n Dynaaminen prioriteetti

- u 'Rankaisee' pitkään pörränneitä prosesseja, aikaviipaloitu

## n Useita Ready-jonoja

- u RQ0: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
- u RQ2..RQn-1: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
- u RQn: pura aikaviipaleittain, RR, pidä samassa jonossa

## n Prosessi kulkeutuu lopulta RQn-jonoon, josta se aikanaan valmistuu

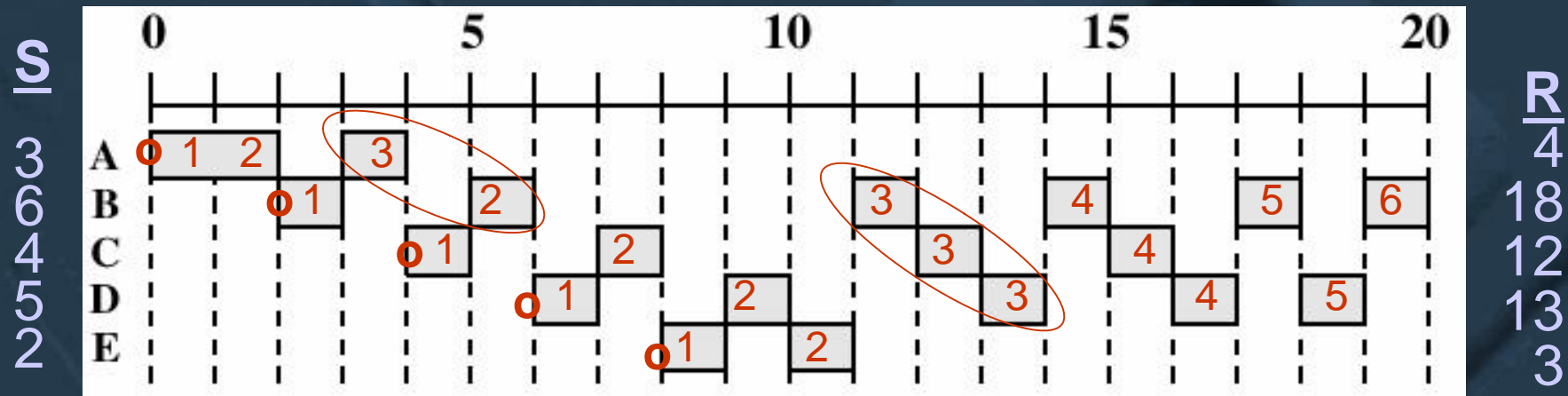
## n Nälkiintymisvaara

- u vuorottaa alemmassa jonossa olevat aina ensin

## n Useita variaatioita

- u esim. alemmissa jonoissa pitempi aikaviipale (esim. 1, 2, 4, ...)
- u palaa blocked-tilasta samaan jonoon

# Feedback $q=1$

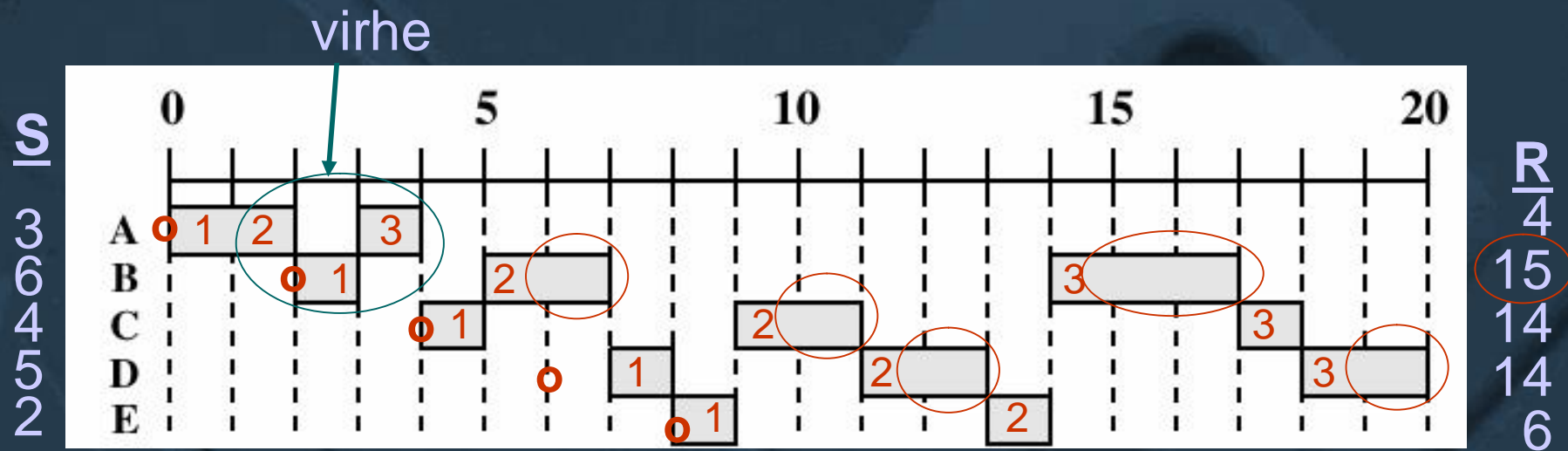


4 (keskim.) (Fig 9.5 [Stal05])

keskim. 10.0

- n Jonot  $RQ_0, RQ_1, RQ_2, \dots$
- n Aikaviipale  $q=1$
- n Ei vaadi etukäteisarvioita CPU-ajan tarpeesta
- n Pitkät työt voivat kestää kauan

# Feedback $q=2^i$



4 (keskim.) (Fig 9.5 [Stal05])

keskim. 10.6

$n$  Jonot  $RQ_0, RQ_1, RQ_2, \dots, RQ_i$

$n$  Aikaviipale  $q=2^i$

$n$  Vaikutus, vrt.  $q=1$

Fig 9.5 [Stal05]



# Yhteenveto

Tbl 9.3 [Stal05]

	Selection Function	Decision Mode	Throughput	Response Time	Overhead	Effect on Processes	Starvation
<b>FCFS</b>	$\max[w]$	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
<b>Round Robin</b>	constant	Preemptive (at time quantum)	May be low if quantum is too small	Provides good response time for short processes	Minimum	Fair treatment	No
<b>SPN</b>	$\min[s]$	Nonpreemptive	High	Provides good response time for short processes	Can be high	Penalizes long processes	Possible
<b>SRT</b>	$\min[s - e]$	Preemptive (at arrival)	High	Provides good response time	Can be high	Penalizes long processes	Possible
<b>HRRN</b>	$\max\left(\frac{w + s}{s}\right)$	Nonpreemptive	High	Provides good response time	Can be high	Good balance	No
<b>Feedback</b>	(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  $e$

# Käyttöjärjestelmät II

UNIX

SVR3 / BSD4.3

VUOROTTAMINEN

# UNIX: Vuorottaminen

## n Interaktiivinen ympäristö

- u ei varsinaista eräajoa, ei erätyöjonoja
- u at-komento ohjelmien ajamiseksi myöhemminkin
- u crontab periodisille töille

cron = “chronogram” daemon?  
tab = table

## n Pyrkii hyvään vastausaikaan

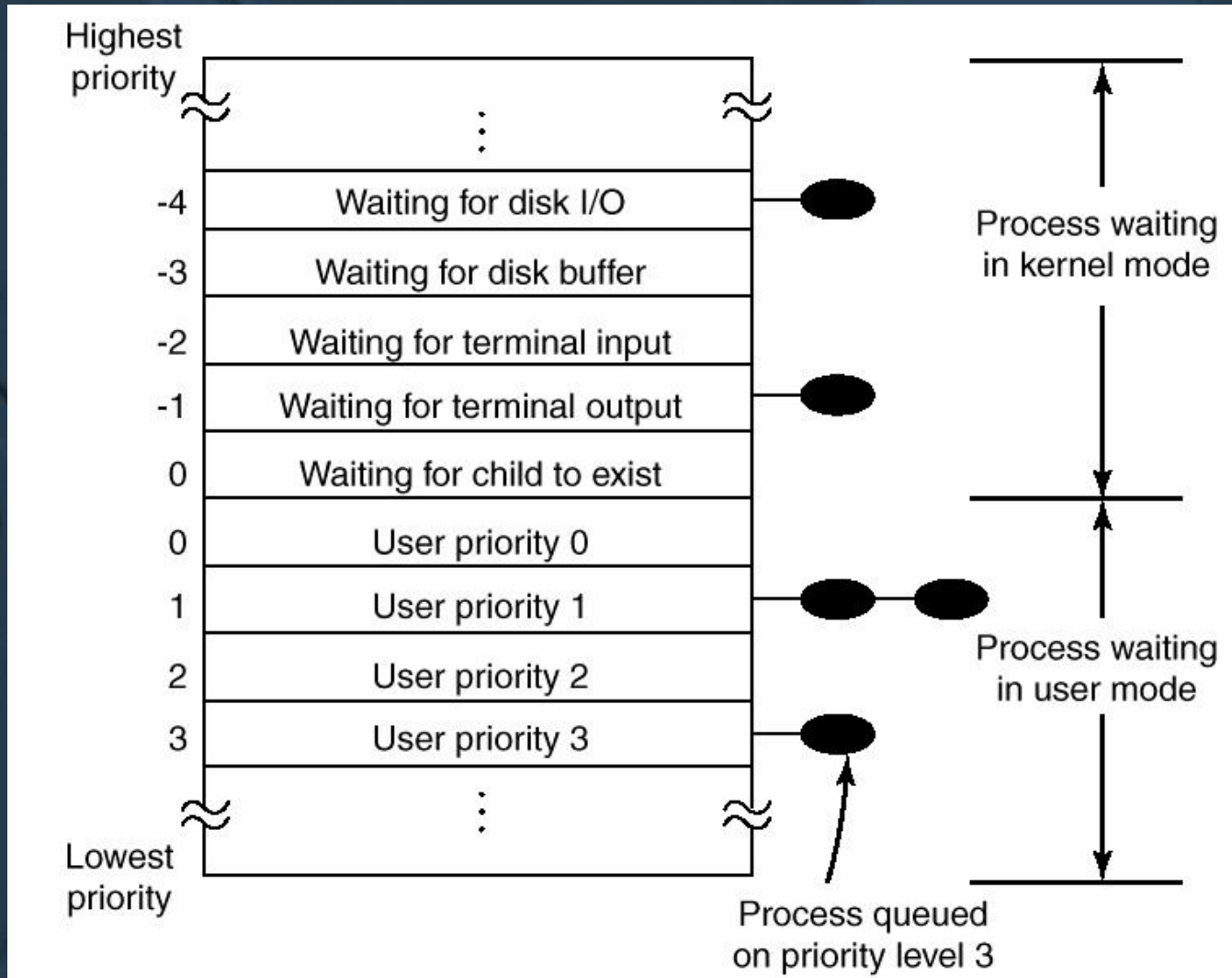
- u taustaprosesseilla huono prioriteetti

## n Aikaviipaleet, Round-Robin

## n Multilevel feedback

- u prioriteeteilla omat Ready-jonot
- u tyhjentää suurimman prioriteetin jonon ensin
- u dynaaminen prioriteetti → ei nälkiintymistä

# UNIX: Ready-queue



(Fig 10-11 [Tane01])

# UNIX: Prioriteetti

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

# UNIX: multilevel feedback

n CPU\_usage = CPU:n käyttö äskettäin

u laskuri PCB:ssä

n Älä rankaise liikaa aiemmasta käytöstä

u puolita ennen prioriteetin laskentaa, ja sitten taas

$$\text{CPU\_usage} = \text{CPU\_usage} / 2$$

$$\text{Pri} = \text{Base} + (\text{CPU\_usage} / 2) + \text{Nice}$$

(jos ei `cpu_usage = 0`, niin ei muutosta)

n Pieni arvo = suuri prioriteetti

n Esimerkki

u Base=60, Nice = 0

u päivitä counter (CPU\_usage) 60 kertaa/sek

F kellolaite keskeytys 16.7 ms välein?

F otanta: kenellä suoritin keskeytyshetkellä?

u päivitä prioriteetti sekunnin välein

Fig 9.17 [Stal05]

# Fair-Share Scheduling

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

# FSS - Fair-Share Scheduling

## n Prioriteetin määrittäminen

$$\text{CPU\_counter} = \text{CPU\_counter} / 2$$

$$\text{GCPU\_counter} = \text{GCPU\_counter} / 2$$

$$\text{Pri} = \text{Base} + \text{CPU\_counter} / 2 + \text{GCPU\_counter} / (4 * W_{\text{group}})$$

iso  $W_{\text{group}}$

à pieni painoarvo

GCPU\_counter:lla

à pieni Pri arvo

à iso prioriteetti

## Esimerkissä

u **Base = 60**

u  **$W_A = 0.5$  ja  $W_{B+C} = 0.5$**  ( $W_A + W_{B+C} = 1$ )

u **päivitä laskurit 60 kertaa sekunnissa**

u **päivitä prioriteetti sekunnin välein**

Fig 9.16 [Stal05]



# Kertauskysymyksiä

- n Milloin vuorotetaan?
- n Tapahtumaohjattu (non-preemptive) VS. keskeytyvä (pre-emptive) vuorottaminen
- n Miten FCSF ja Round-Robin algoritmit eroavat toisistaan?
- n Millaisiin tilanteisiin ne sopivat?
- n Mikä FSS algoritmin perusidea?
- n Mitä hyötyä on käyttää prioriteetteja?
- n Mitä hyötyä on vaihtelevan prioriteetin käytöstä?