Sistemas Operacionais

Escalonamento de Processos Uniprocessador

Capítulo 9

Objetivos do Escalonamento

- É a chave de multiprogramação eficiente
- deve ser transparente ao usuário

- Esolher processos a serem executados por pelo processador(es)
- tempo de resposta
- Throughput/vazão
- eficiência do processador

Níveis de Escalonamento

- Longo Prazo
- Médio Prazo
- Curto Prazo

Escalonamento de Longo Prazo

- Determina que programas são admitidos no sistema para processamento
- Controla o grau de multiprogramação
- quanto mais processos, menor a percentagem de tempo que cada processo é executado
- Quando um processo é aceito, vai para fila de prontos (curto prazo)
- Escalonador chamado quando
  - um processo finaliza
  - processador ocioso (muitos processos bloqueados)

Escalonamento de Médio Prazo

- associado ao swapping the processos
- baseado na necessidade de gerenciar o grau de multiprogramação
  - necessidade de liberar MP para outros processos
  - processador ocioso

Escalonamento de Curto Prazo

- conhecido como despachante
- mais frequentemente executado
- chamado quando os seguintes eventos ocorrem
  - interrupção de relógio
  - interrrupção de E/S
  - chamadas de procedimento
  - sinais

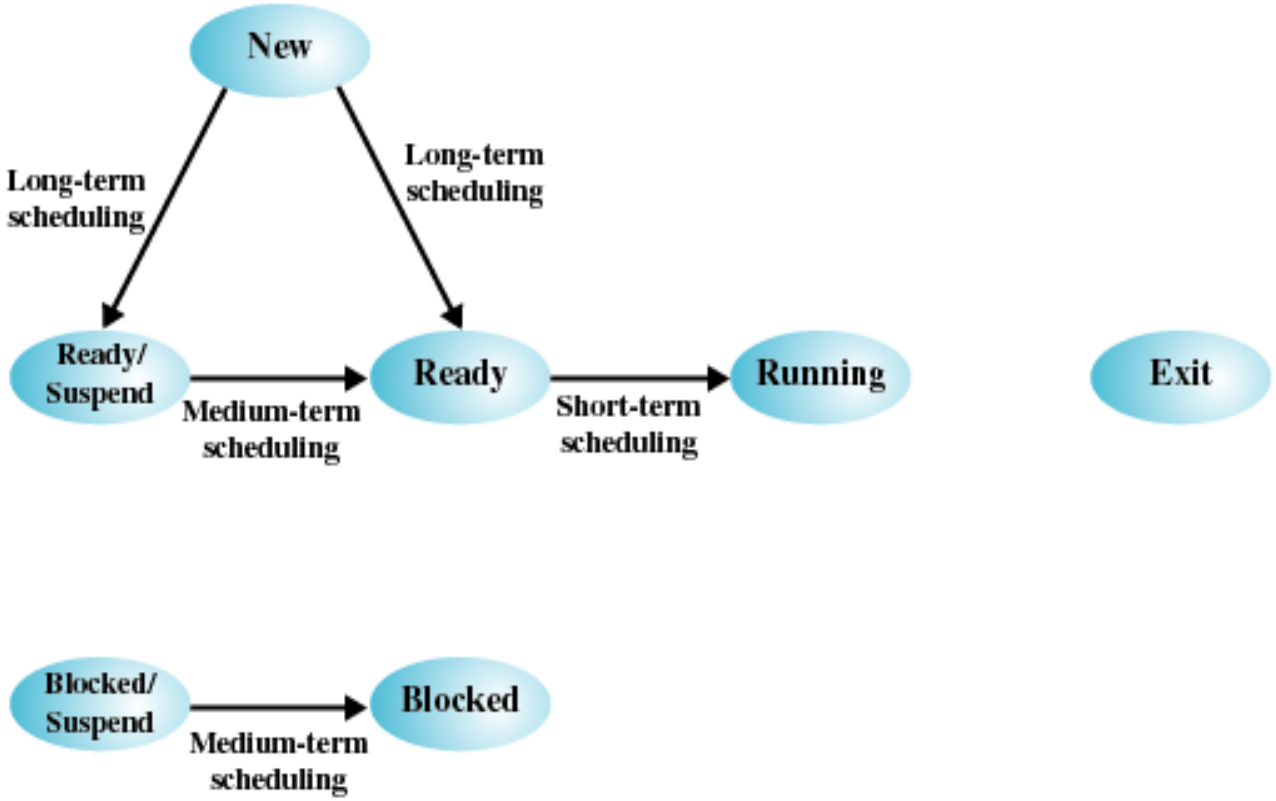


Figure 9.1 Scheduling and Process State Transitions

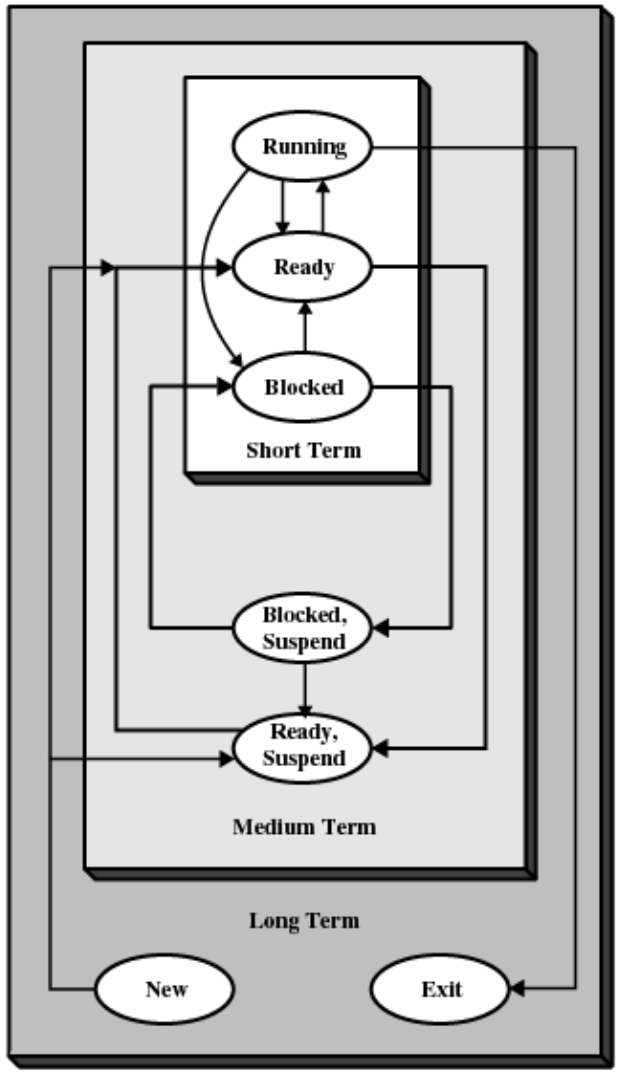


Figure 9.2 Levels of Scheduling

Critérios de escalonamento de curto prazo

- orientado ao usuário
  - voltado para qualidade de serviço associada a sistemas iterativos
  - tempo de resposta
    - período de tempo desde a submissão de um pedido até sua saída
  - minimizar tempo de resposta mesmo quando vários usuários compartilham os recursos
    - eficiência: tempo de resposta em média de 2s
  - neste tipo de sistema, escalonador decide de acordo com este objetivo

Critérios de escalonamento de curto prazo

- orientado ao sistema
  - utilização efetiva e eficiente do sistema
  - maximização de throughput/vazão
    - número de processos por período de tempo

- relacionado ao desempenho
  - quantitativa
  - medições como tempo de resposta e throughput/vazão

Orientado ao usuário e desempenho

- tempo de resposta
  - enquanto executa um processo, pode-se retornar resposta para outro processo
  - processos iterativos
- turnaround time
  - pode ser maior ou igual ao tempo de resposta
  - considerado também para processos em batch
- deadlines
  - política cujas decisões devem respeitar deadlines de processos

Orientado ao sistema e desempenho

- throughput
  - depende do tempo dos processos mas também da política de escalonamento
- utilização do processador
  - processador deve estar sempre ocupado
  - sistemas compartilhados
  - não é o caso de sistemas de tempo real, mesmo que recursos estejam sendo compartilhados

Outros critérios do sistema

- Justeza (fairness)
  - para evitar starvation
- prioridades
- balanceamento na utilização de recursos

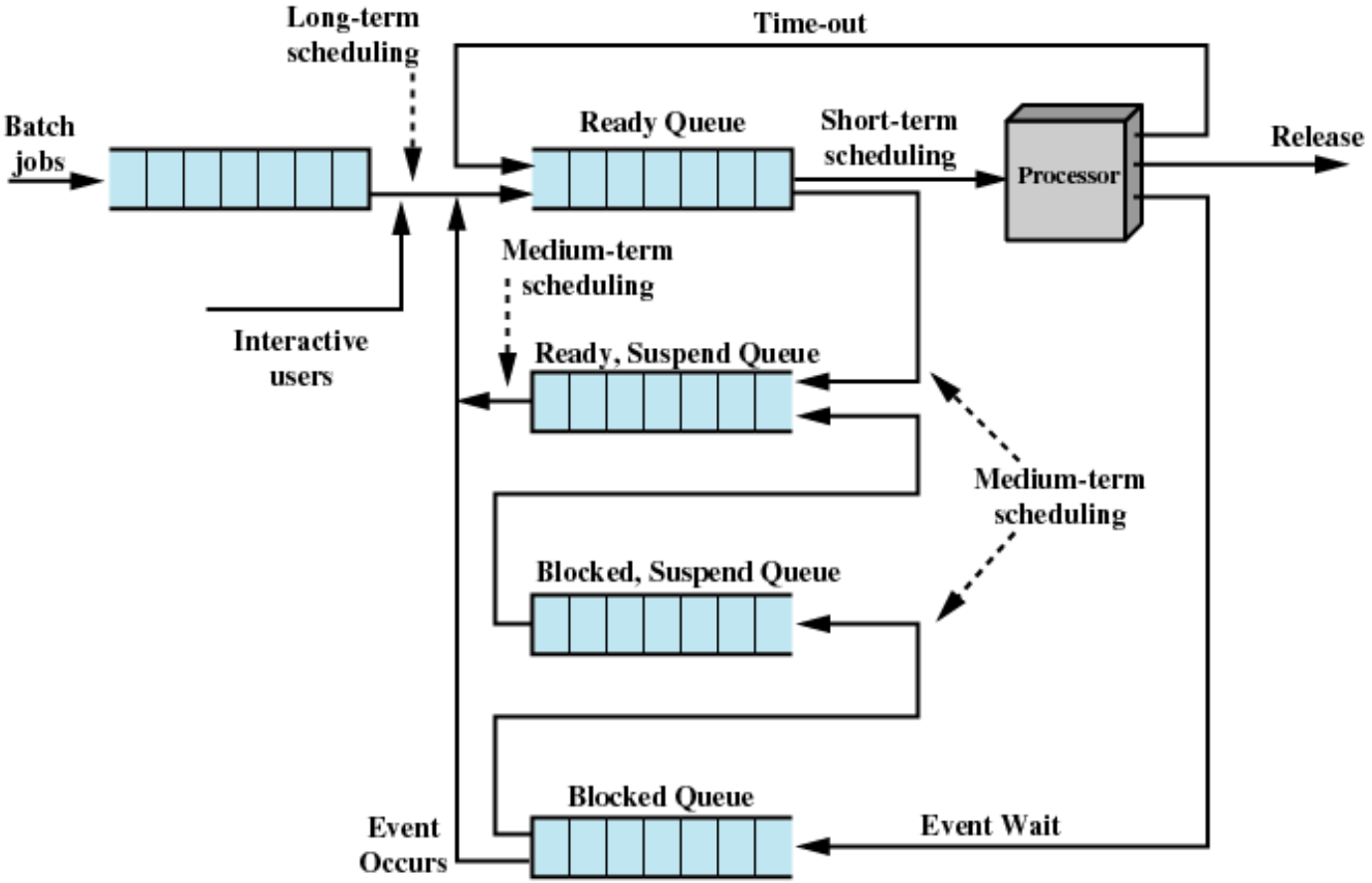


Figure 9.3 Queuing Diagram for Scheduling

Aspectos gerais – Políticas de escalonamento

Dois aspectos principais

- seleção
  - qual processo será selecionado para execução
  - baseado em prioridade/necessidade de recurso/ características de tempo de execução

- modo de decisão

Prioridades como critério de escalonamento

- escolha de processo de maior prioridade
- implementação de filas múltiplas, uma para cada nível de prioridade
- starvation: processo de menor prioridade pode "nunca" ser escolhido
- é necessário modificar prioridades baseado no tempo que o processo passa na fila

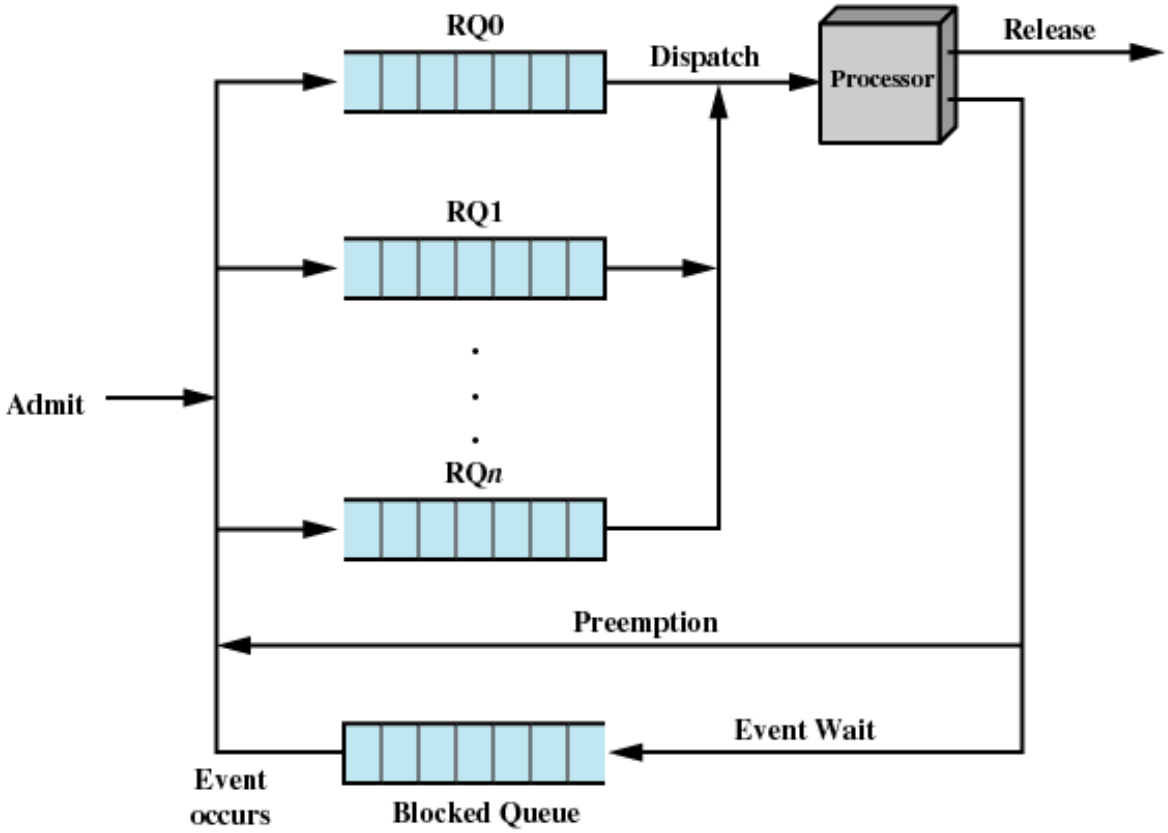


Figure 9.4 Priority Queuing

Modos de decisão

- não preemptivo
- preemptivo
  - o processo sendo executado é interrompido e colocado na fila de prontos pelo SO
  - elimina a monipolização do processador por um processo por muito tempo
  - mas, overheads na troca de contexto
    - importante investir no hardware e software

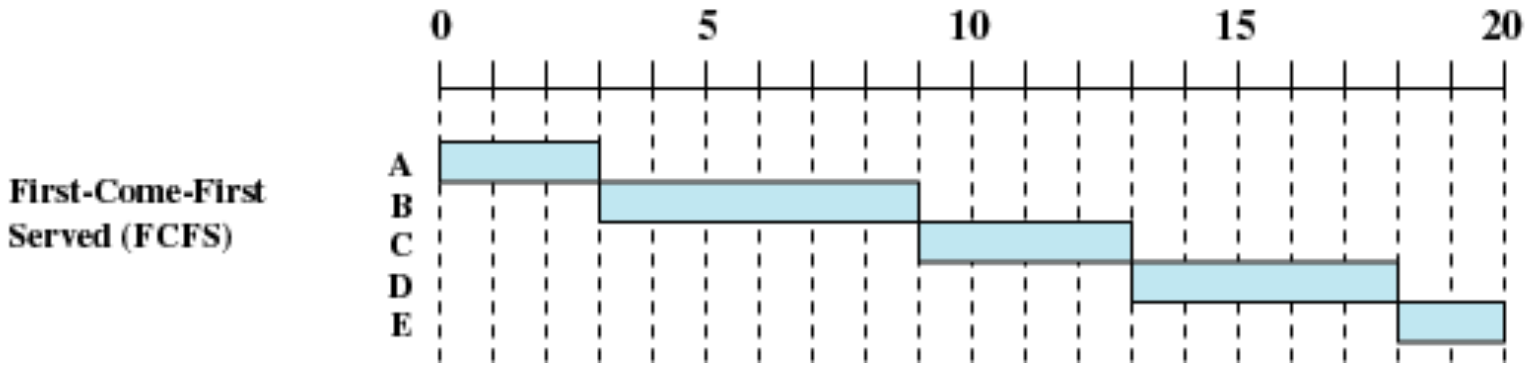
Política:First-Come-First-Served (FCFS)

- seleciona o processo que está a mais tempo na fila de prontos
- exemplo (não preemptivo)

Table 9.4 Process Scheduling Example

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

First-Come-First-Served (FCFS)



- processo curto muito tempo no sistema, pois chegou depois de um processo longo
- favorece processos CPU-bound ao invez de I/O bound (por que?)

First-Come-First-Served (FCFS)

- boa métrica
  - tempo de espera normalizado de cada processo
    - razão entre o turnaround time e o tempo de serviço

$$T_n = T_q / T_s$$

    - eficiente se ∀ processo $T_n \approx 1.9$

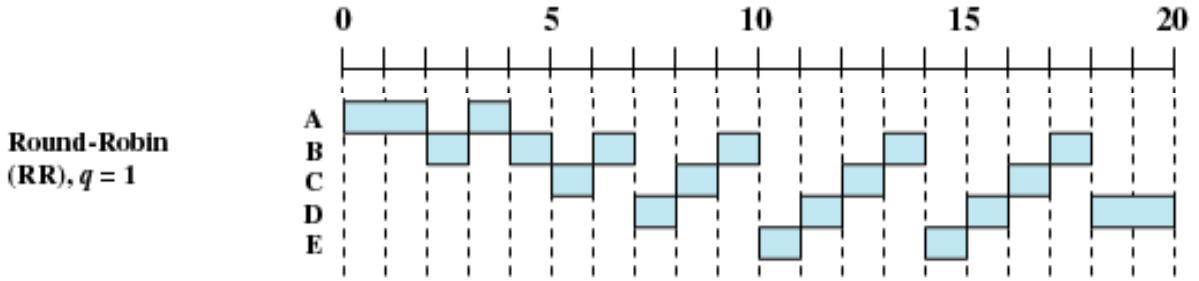
First-Come-First-Served (FCFS)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	3	9	13	18	20
Turnaround time (T_q)	3	7	9	12	12
Tempo em fila normalizado ($T_n = T_q/T_s$)	1	1,17	2,25	2,4	6

Round-Robin (RR)



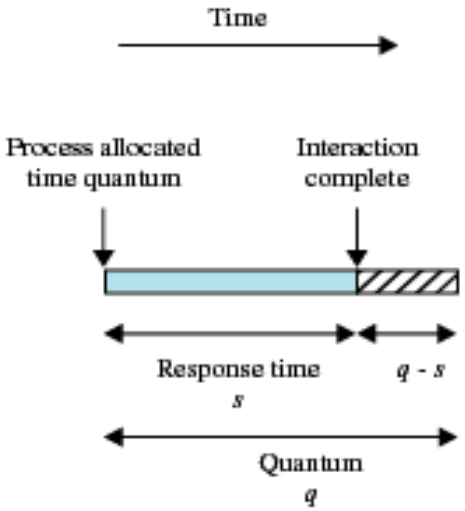
- política preemptiva
- fatia de tempo (time slice) ou quantum de tempo de CPU para cada processo
- interrupção de relógio é gerada a cada quantum

Round-Robin (RR)

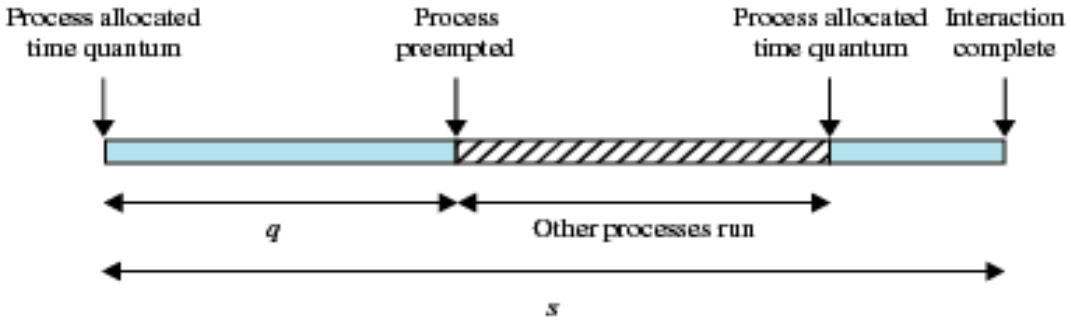
- política preemptiva
- fatia de tempo (*time slice*) ou quantum de tempo de CPU para cada processo
- interrupção de relógio é gerada a cada quantum
- processo em execução → interrompido → fila de prontos
- resolve o problema de processos curtos como no caso de FCFS

Round-Robin (RR)

- qual o melhor valor de quantum?
- pequeno
  - bom para processos curtos
  - mas, freqüência dos overheads maior
    - tratamento de interrupção +
    - funções do escalonamento +
    - funções do despachante



(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

Round-Robin: quantum = 4

Table 9.4 Process Scheduling Example

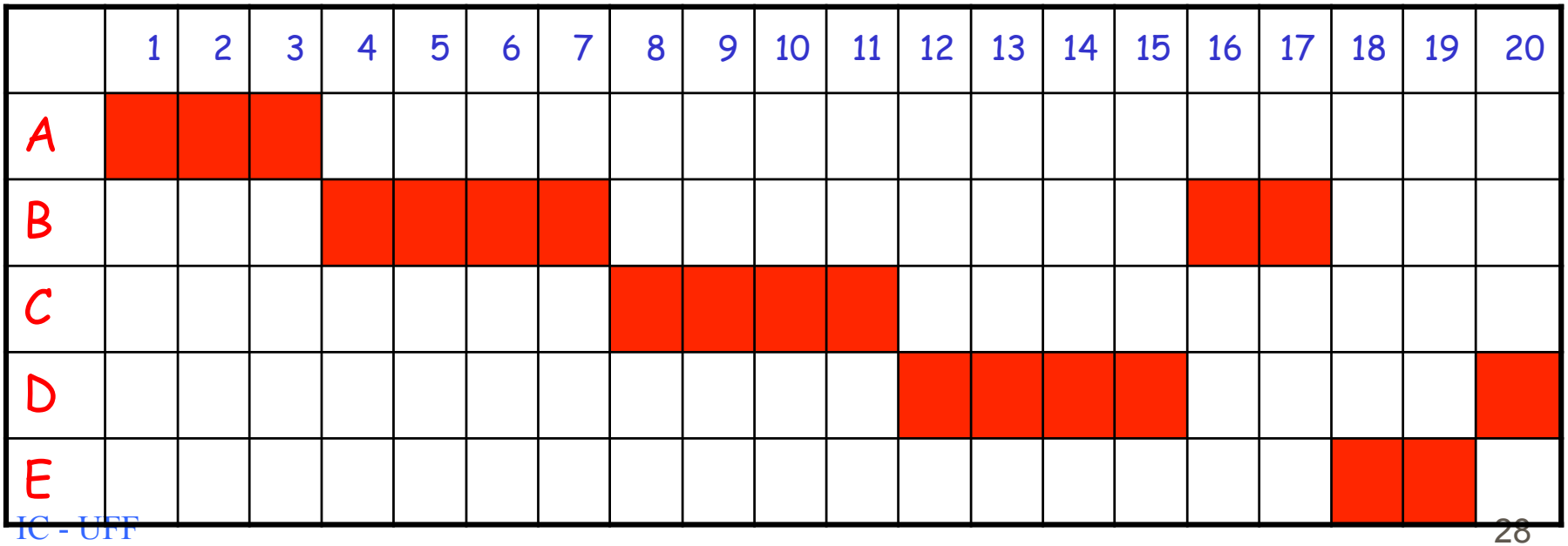
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Round-Robin: quantum = 4

Table 9.4 Process Scheduling Example

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



Round-Robin: quantum = 4

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	3	17	11	20	19
Turnaround time (T_q)	3	15	7	14	11
Tempo em fila normalizado ($T_n = T_q/T_s$)	1	2,5	1,75	2,8	5,5

Round-Robin: quantum = 1

Table 9.4 Process Scheduling Example

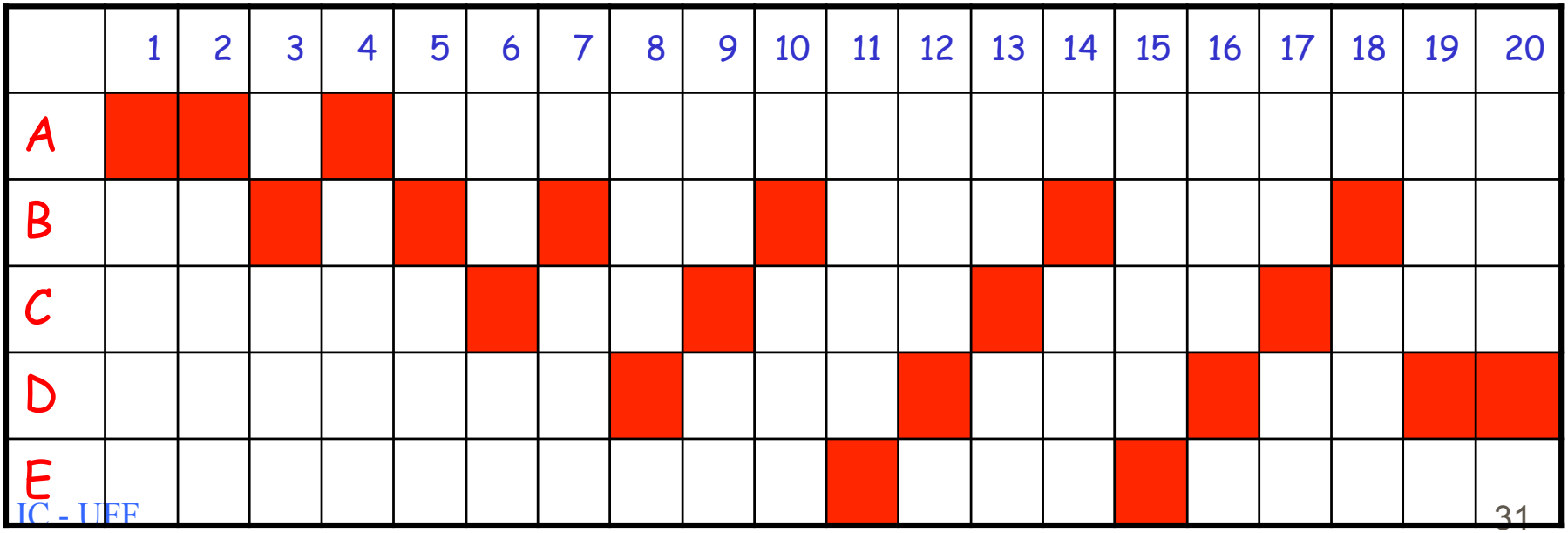
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Round-Robin: quantum = 1

Table 9.4 Process Scheduling Example

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



Round-Robin: quantum = 4

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim					
Turnaround time (T_q)					
Tempo em fila normalizado ($T_n = T_q/T_s$)					

Round-Robin: quantum = 4

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	4	18	17	20	15
Turnaround time (T_q)	4	16	13	14	7
Tempo em fila normalizado ($T_n = T_q/T_s$)	1,33	2,67	3,25	2,8	3,5

Round-Robin (RR)

- vantajoso para sistemas compartilhados em geral
- problemático para processos I/O bound perante CPU bound
  - devido a E/S, o processo perde CPU
  - ao retornar, vem para o final da fila de prontos
- para contornar este problema: RR virtual

Round-Robin virtual (RR virtual)

- processo requisita E/S ➡ vai para fila de bloqueados
- quando evento terminar, processo vai para fila de prontos auxiliar
- quando próximo quantum terminar, o escalonador dá preferência para a fila de prontos auxiliar (pois...?)

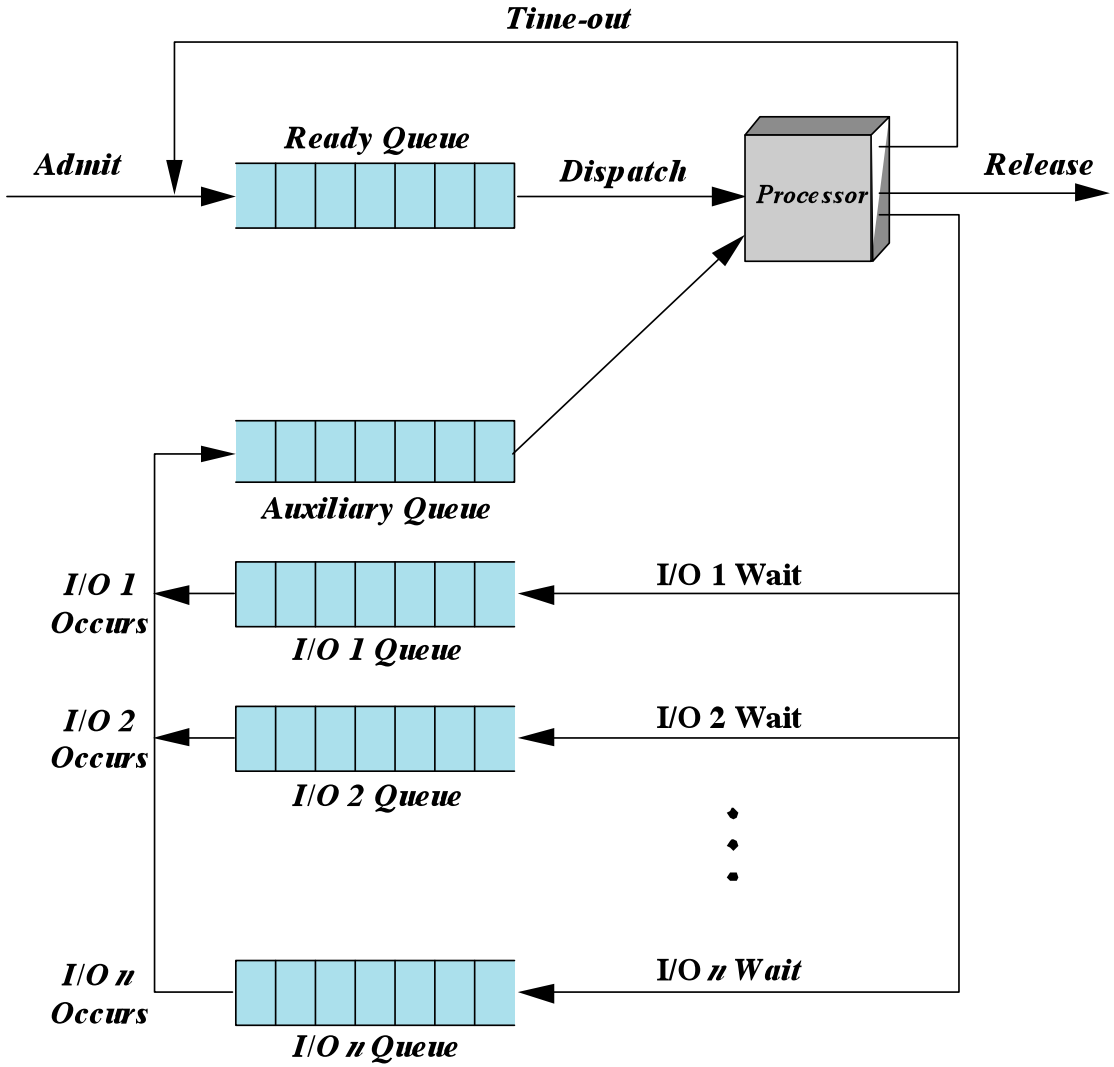


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

Shortest Process Next (SPN)

- não preemptivo
- o processo com o menor tempo de serviço é selecionado como o próximo para ser selecionado
- prioridade para processos curtos
  - turnaround time muito menor para processos curtos do que FCFS
- Mas.....
  - como estimar o tempo de execução dos processos?
  - geralmente, estimativa de processos mais longos não é tão precisa
  - para processos longos: starvation

Shortest Process Next (SPN)

Table 9.4 Process Scheduling Example

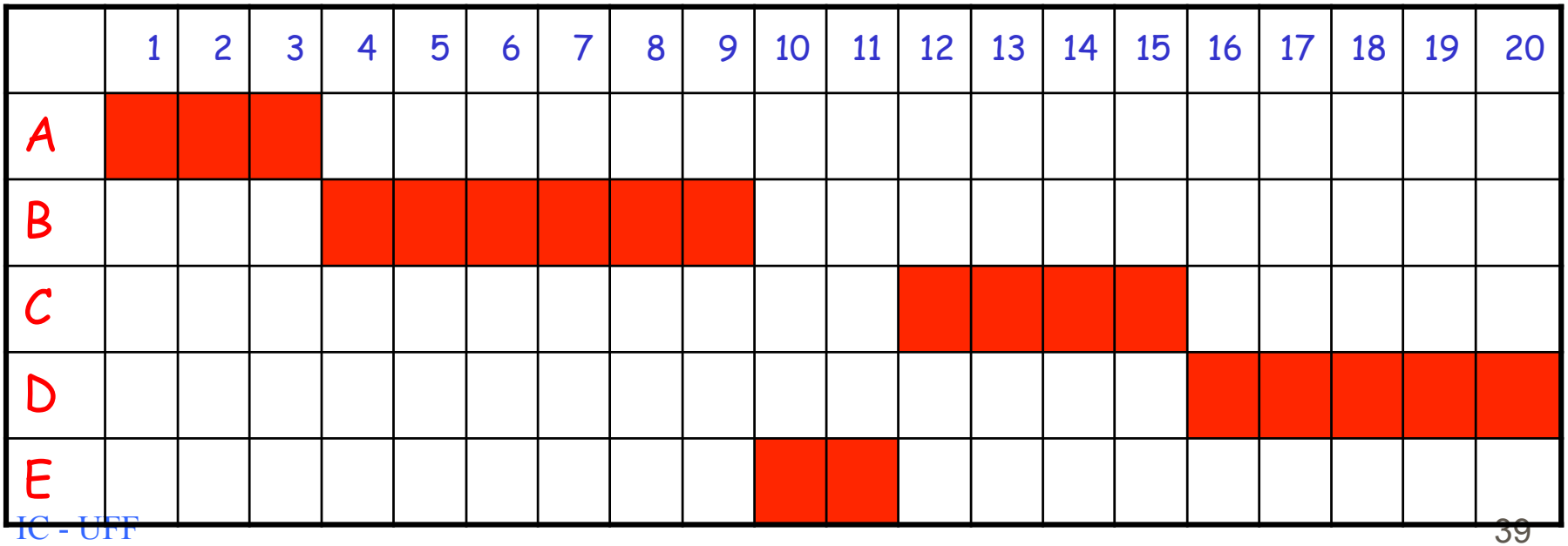
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Shortest Process Next (SPN)

Table 9.4 Process Scheduling Example

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



Shortest Process Next (SPN)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim					
Turnaround time (T_q)					
Tempo em fila normalizado ($T_n = T_q/T_s$)					

Shortest Process Next (SPN)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	3	9	15	20	11
Turnaround time (T_q)	3	7	11	14	3
Tempo em fila normalizado ($T_n = T_q/T_s$)	1	1,17	2,75	2,8	1,5

Shortest Remaining Time (SRT)

- versão preemtiva do SPN
  - o não preemtivo: não aplicável para sistemas compartilhados
  - processos são interrompidos quando um novo processo chega na fila de prontos
- também precisa estimar tempo de execução dos processos
- um novo processo pode ter prioridade quando chegar na fila de prontos
- processos longos → starvation
- menos interrupções que RR

Shortest Remaining Time (SRT)

Table 9.4 Process Scheduling Example

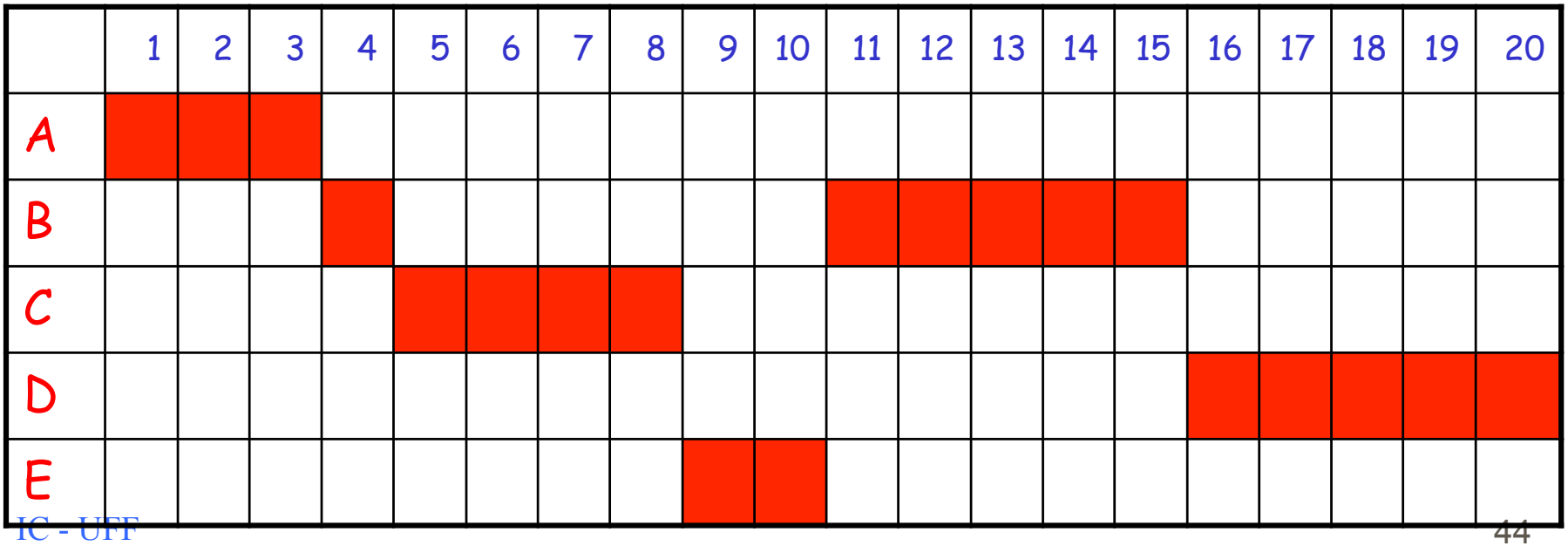
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Shortest Remaining Time (SRT)

Table 9.4 Process Scheduling Example

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



Shortest Remaining Time (SRT)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim					
Turnaround time (T_q)					
Tempo em fila normalizado ($T_n = T_q/T_s$)					

Shortest Remaining Time (SRT)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	3	15	8	20	10
Turnaround time (T_q)	3	13	4	14	2
Tempo em fila normalizado ($T_n = T_q/T_s$)	1	2,17	1	2,8	1

Highest Response Ratio Next (HRRN)

- não preemptivo: seleciona o próximo processo com a maior média de resposta calculado como
- taxa de resposta

$$TR = (w + T_s)/T_s$$

- *jobs* curtos terão prioridade, mas...
  - *jobs* longos terão maior tempo de espera na fila e se tornarão prioritários

Highest Response Ratio Next (HRRN)

Table 9.4 Process Scheduling Example

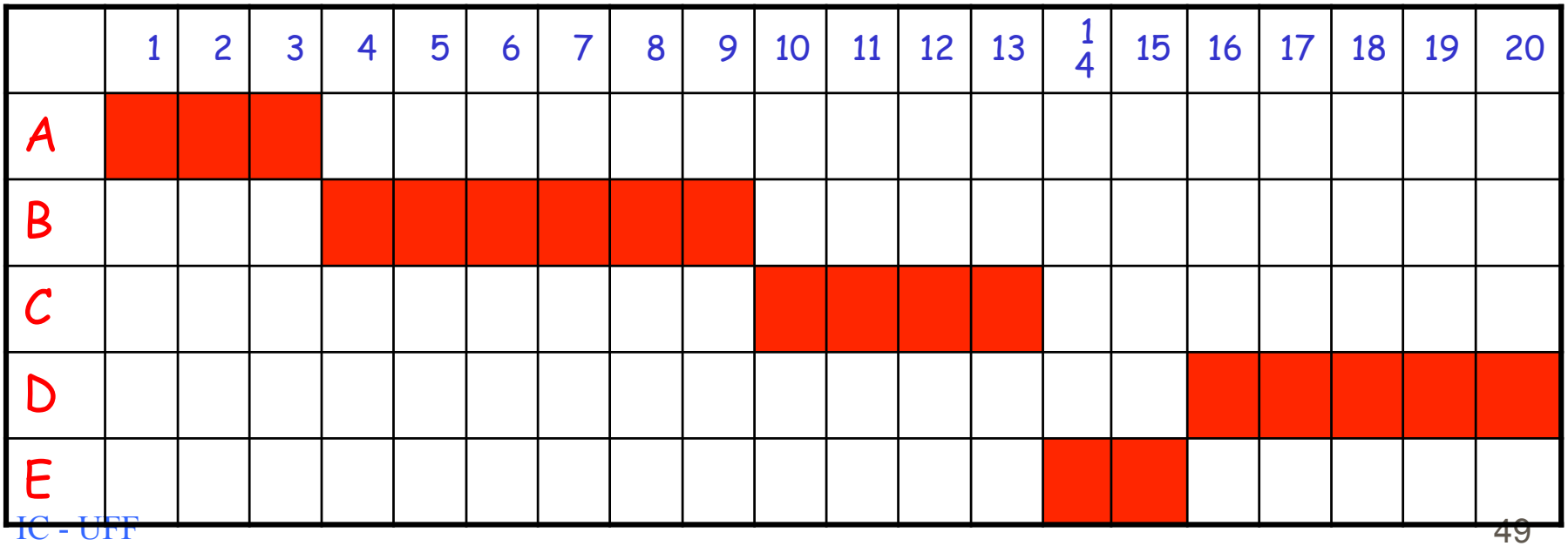
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Highest Response Ratio Next (HRRN)

Table 9.4 Process Scheduling Example

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



Highest Response Ratio Next (HRRN)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim					
Turnaround time (T_q)					
Tempo em fila normalizado ($T_n = T_q/T_s$)					

Highest Response Ratio Next (HRRN)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	3	9	13	20	15
Turnaround time (T_q)	3	7	9	14	7
Tempo em fila normalizado ($T_n = T_q/T_s$)	1	1,17	2,25	2,8	3,5

Prioridade

- prioridades associadas aos processos
  - desempate: FCFS
- prioridades externad ou internas
- pode ser preemptivo
  - chegada de um novo processo a fila de prontos
- ou não preemptivo: CPU ocioso – próximo processo de maior prioridade é selecionado
- problema: *starvation*
- solução: *aging* (envelhecimento)
  - aumentar a prioridade de processos que estão esperando longamente na fila

Feedback

- sem cálculo do futuro (como no caso do SPN, SRT e HRRN)
  - olha para o passado
- preemtivo
- especificação de n filas:
  - P1 entra na fila RQ_0 e é escolhido para execução
  - interrupção: P1 vai para RQ_1
  - .... P1 saiu da fila RQ_i quando foi escolhido para execução
  - interrupção: P1 vai para RQ_{i+1}
- um processo é escolhido da fila de menor índice que contem processos prontos para serem executados

Feedback

- jobs curtos – pouco tempo no sistema
- se um job chegar a fila RQ_n, depois passará para RQ_0
- preempção pode ser por interrupção ao chegada de processos na fila RQ_0 (ou híbrido)
- ou: processo que vem dá fila RQ_i tem 2^i unidades de tempo de processador.

Feedback (quantum = 2)

Table 9.4 Process Scheduling Example

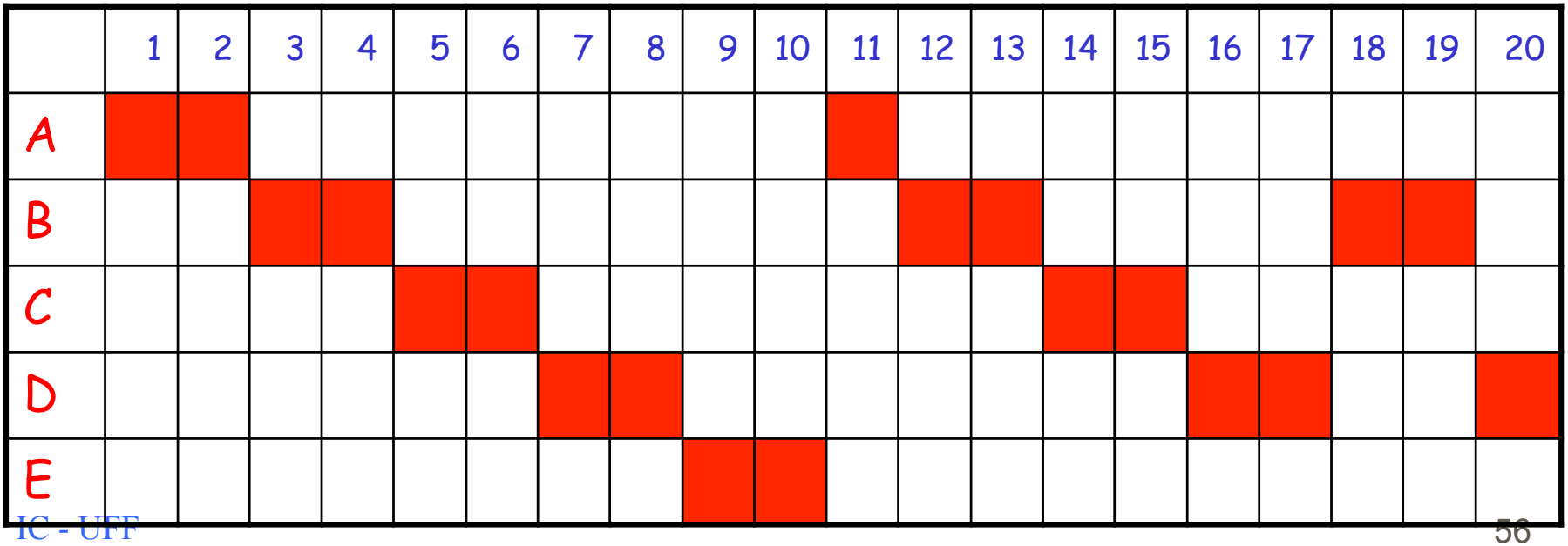
Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A																				
B																				
C																				
D																				
E																				

Feedback (quantum = 2)

Table 9.4 Process Scheduling Example

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



Feedback (quantum = 2)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim					
Turnaround time (T_q)					
Tempo em fila normalizado ($T_n = T_q/T_s$)					

Feedback (quantum = 2)

Table 9.4 Process Scheduling Example

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

	A	B	C	D	E
Tempo de fim	11	19	15	20	10
Turnaround time (T_q)	8	17	11	14	2
Tempo em fila normalizado ($T_n = T_q/T_s$)	2,67	2,83	2,75	2,8	1

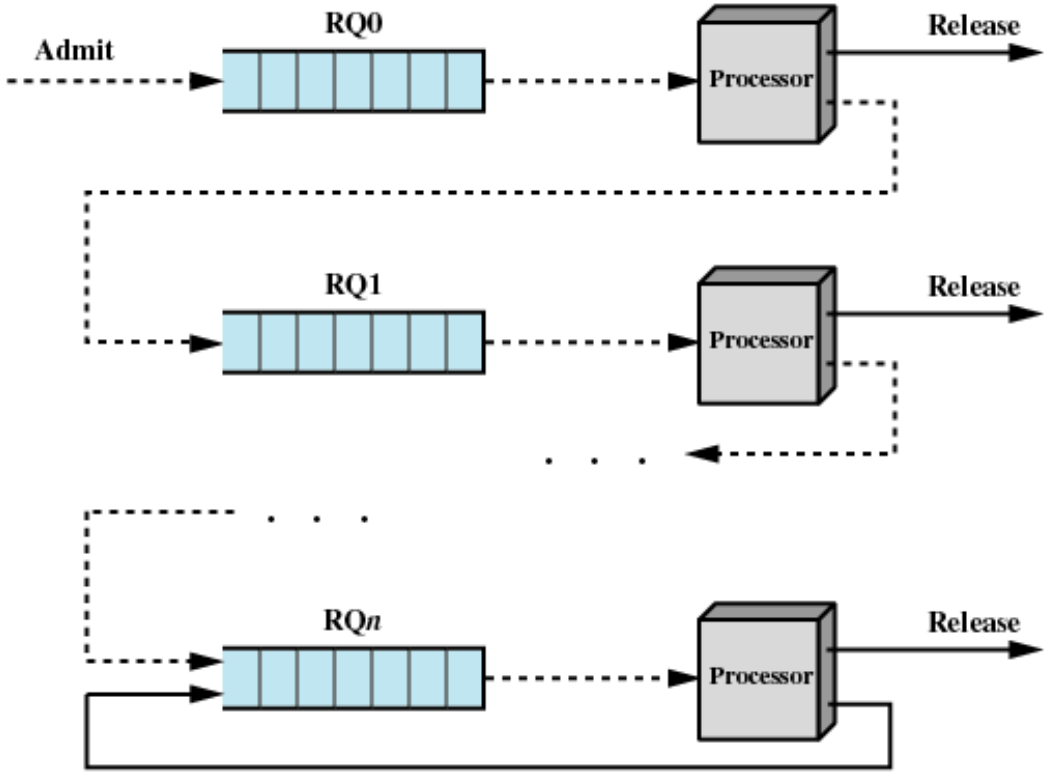


Figure 9.10 Feedback Scheduling

Estimativa de tempo

- grande problema com as políticas que utilizam a estimativa de tempo, é como especificar o tempo de serviço
- Exemplo: função probabilistica baseada no passado (simulando execução de programas com diferentes entradas)
- políticas: SPN (shortest process next)
  SRT (shortest remaining time)]
  HRRN (highest response ratio next)

Estimando o tempo

- devido a dificuldade da previsão, geralmente essa política não é utilizado em escalonamento de curto prazo
- o usuário pode prover o T_s de seu processo
  - se não chegar aquele tempo → abortar
- tentiva de previsão automática do tempo de serviço
  - baseado no histórico de execuções do processo
  - aplicável para processos iterativos
- Previsão: média exponencial dos períodos medidos anteriores

SPN – estimativa de tempo

- t_n → o tempo (real) de serviço da *n*-ésima iteração do processo P
- σ_n → a previsão do tempo d *n*-ésima iteração
- para $0 \leq \alpha \leq 1$

$$\sigma_{n+1} = \alpha\ t_n + (1 - \alpha)\ \sigma_n$$

- geralmente:
  - $\alpha = 0{,}5$
  - σ_0 = constante

SPN – estimativa de tempo

- Exemplo: $\alpha = 0{,}5$ e $\sigma_0 = 10$ u.t. para um processo P

$$\sigma_{n+1} = 0{,}5\ t_n + 0{,}5\ \sigma_n$$

i	0	1	2	3	4	5	6	7...
t_i	6	4	6	4	13	13	13	13...
σ_n	10	8	6	6	5	9	11	12...

Comparação de Desempenho das diversas políticas

- difícil de analisar
  - a distribuição probabilística do tempo de serviço
  - implementação da troca de contexto
  - tipos de E/S e seus desempenhos

- Tentativa de análise: através de teoria de filas
  - análise analítica
  - através de simulações

Exercitando

9.16) Os processos A, B, C, D e E chegam a um computador praticamente ao mesmo tempo. Seus tempos de serviço estimados são 15, 9, 3, 6 e 12, respectivamente. Suas prioridades são: 6, 3, 7, 9 e 4, com o menor valor correspondendo a maior prioridade. Qual é o turnaround time e o tempo de espera normalizado no sistema para cada um dos processos, nas seguintes políticas de escalonamento:

a) RR com quantum 3

b) SRT

c) feedback com 3 filas de prontos e quantum = 2^i, onde i está associada a filha RQ_i

Exercitando

a) RR com quantum 3

Exercitando

a) RR com quantum 3

	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
A																				
B																				
C																				
D																				
E																				

Fair-Share Scheduling

- a aplicação do usuário corresponde a um conjunto de *threads*
- especifica um percentual de processamento a cada processo que participa de uma certa classe de processos

Exemplo: UNIX

- grupos de processos:
  - professores, graduação, pós-graduação, funcionários
  - cada grupo pode receber 25% do temo de processador, por exemplo

Fair-Share Scheduling

- prioridades podem ser utilizadas, que consideram
  - prioridade do próprio processo
  - total de utilização de processador dos processos do grupo ao qual pertence
  - uso médio do processador
- prioridades recalculadas a cada interrupção
- prioridades diminuem cada vez que um processo ganha processador

Fair-Share Scheduling

- Sejam os processos

  A → grupo 1

  B → grupo 2

  C → grupo 2

- Grupo 1 → 50% do processador
- Grupo 2 → 50% do processador
- Suponha um quantum de 1s: a cada 1 s, a seqüência representa as escolhas do processador

  A→ B → A → C → A → B → A → C .....

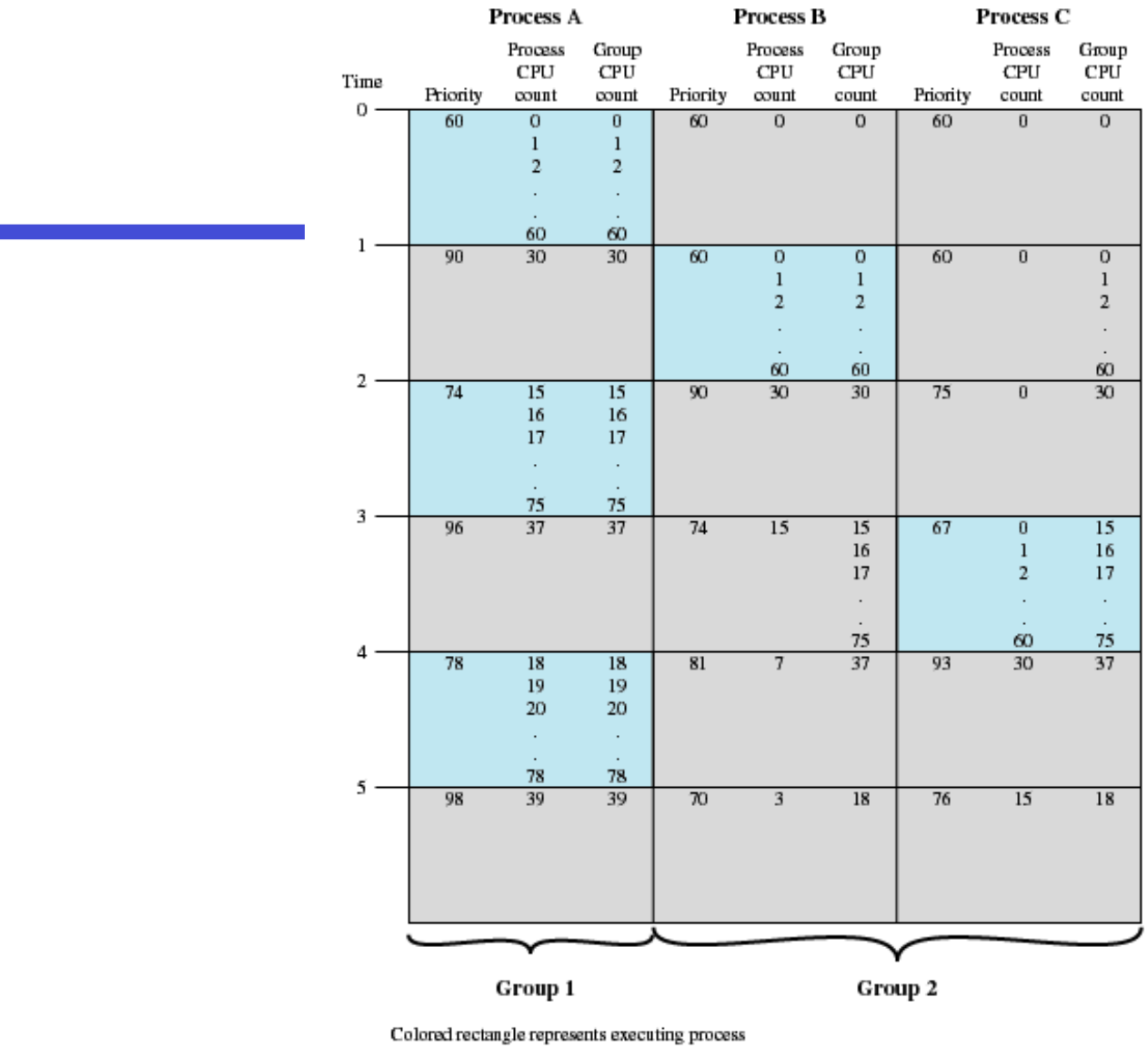
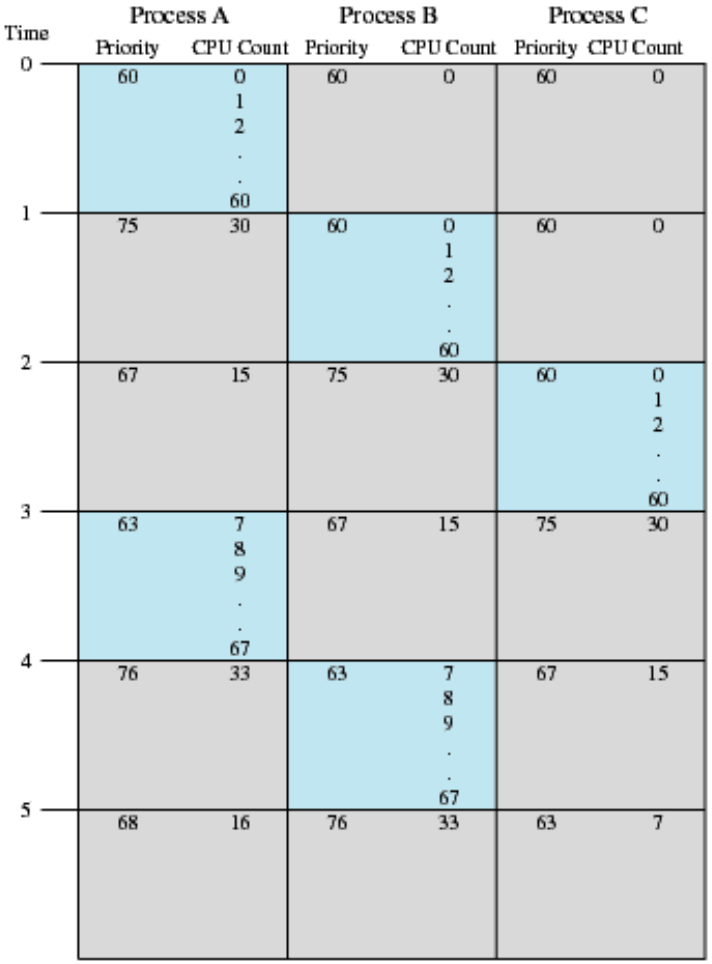


Figure 9.16 Example of Fair Share SchedulerÑThree Processes, Two Groups

Escalonador tradicional do UNIX

- RR em cada fila de prioridade
- preempção: 1s
- prioridades são recomputadas a cada preempção



Colored rectangle represents executing process

Figure 9.17 Example of Traditional UNIX Process Scheduling