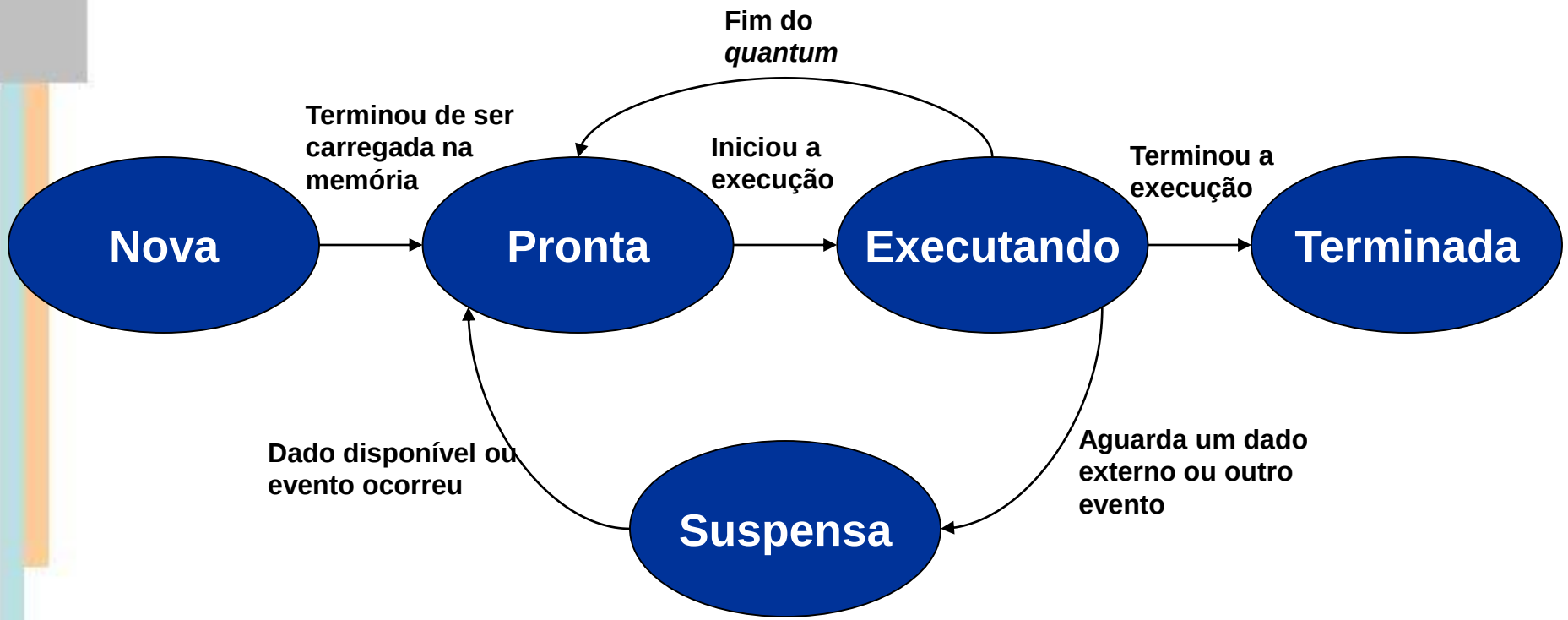
Sistemas Operacionais

Escalonamento de Processos

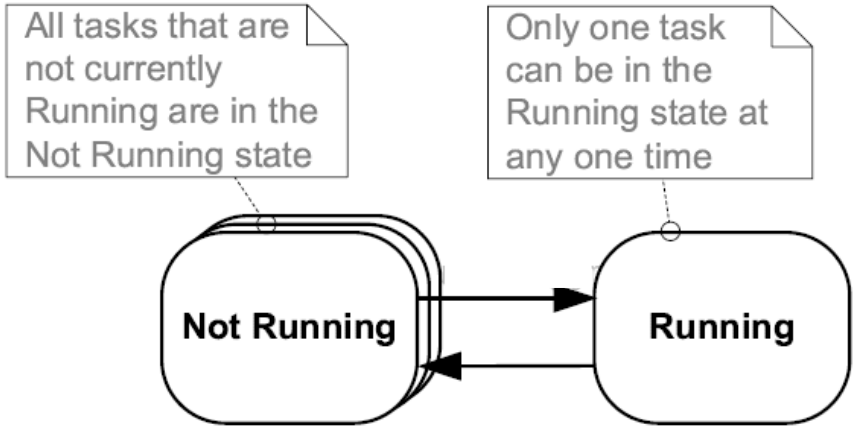
Índice

- Estados de uma tarefa
- Tipos de Tarefas
- Critérios de Escalonamento
- Tipos
  - Escalonamento preemptivo
  - Escalonamento não-preemptivo
  - Escalonamento FCFS ou FIFO
  - Escalonamento Round Robin (RR)
  - Escalonamento Shortest-Job-First (SJF)
  - Escalonamento com prioridades

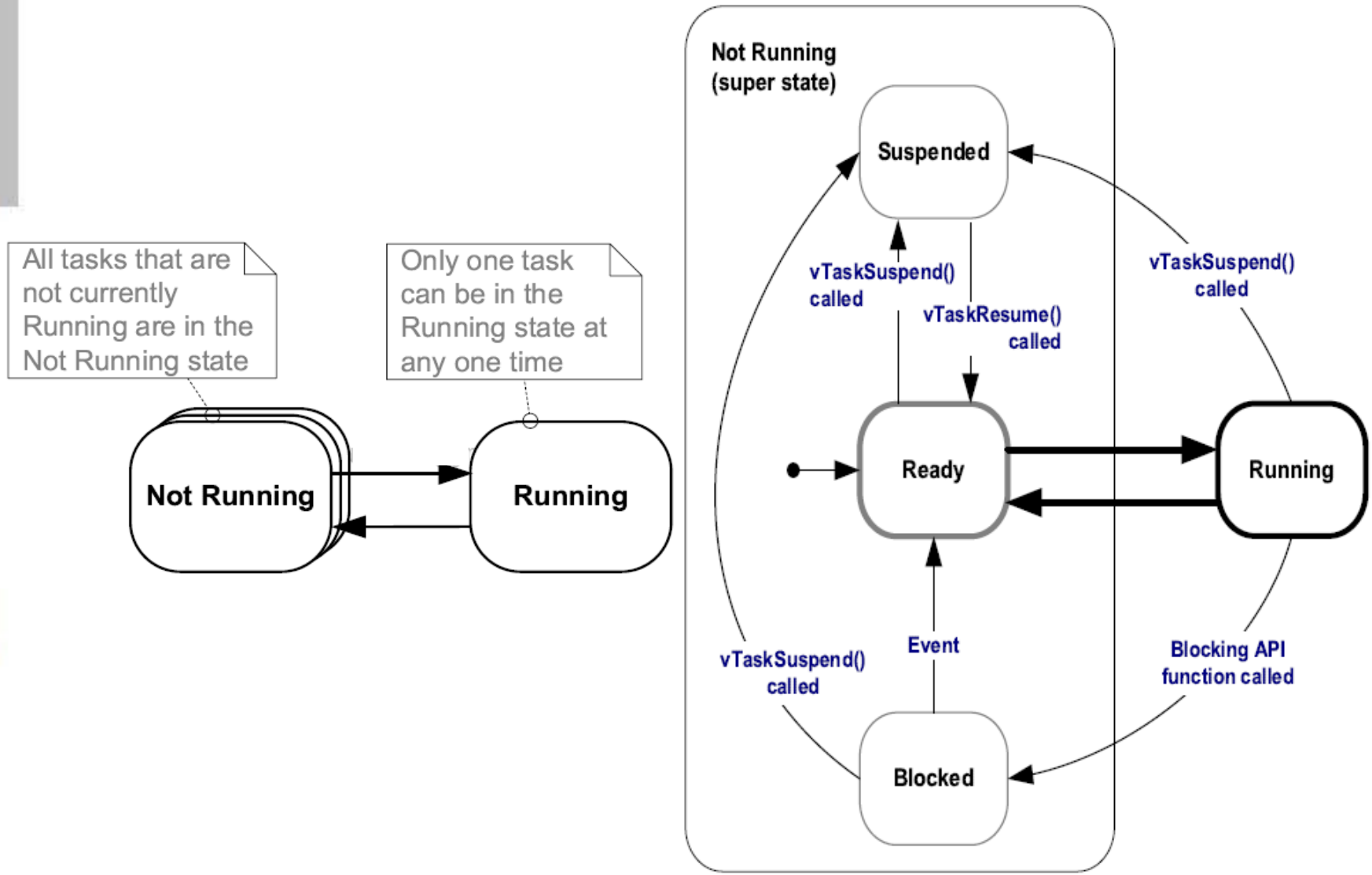
Estados de uma tarefa



Ex.: FreeRTOS


All tasks that are not currently Running are in the Not Running state
Only one task can be in the Running state at any one time
Not Running
Running

Ex.: FreeRTOS


All tasks that are not currently Running are in the Not Running state
Only one task can be in the Running state at any one time
Not Running
Running
Not Running (super state)
Suspended
vTaskSuspend() called
vTaskResume() called
vTaskSuspend() called
Ready
Running
Event
vTaskSuspend() called
Blocking API function called
Blocked

Fila de Processos

Escalonamento de processos

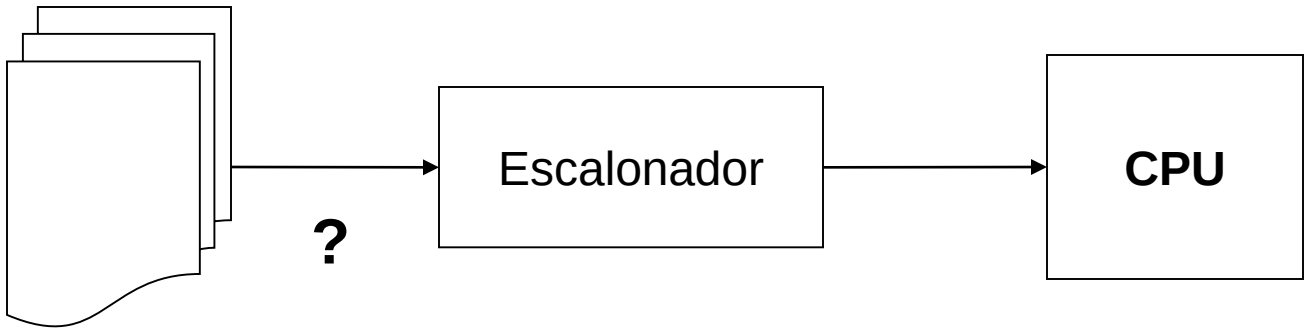
- Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro.

Escalonamento de processos

- Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro.

- A parte do sistema operacional responsável por essa decisão é chamada escalonador, e o algoritmo usado para tal é chamado de algoritmo de escalonamento.

Escolha do processo



Fila de processos prontos

$\Rightarrow$ Problema: Definição da ordem?

Ordem?

Relembrando: Tipos de Tarefas

- Tarefas de tempo real
  - Previsibilidade em seus tempos de respostas (reprodutores de áudio ou vídeo, controle de processos industriais)
- Tarefas interativas
  - Eventos externos solicitados por usuários (editores de texto, navegadores, jogos)
- Tarefas em lote (batch)
  - Sem intervenção do usuário (backup, cálculos numéricos, varredura anti-vírus, renderização de áudio)

Tipos de Tarefas

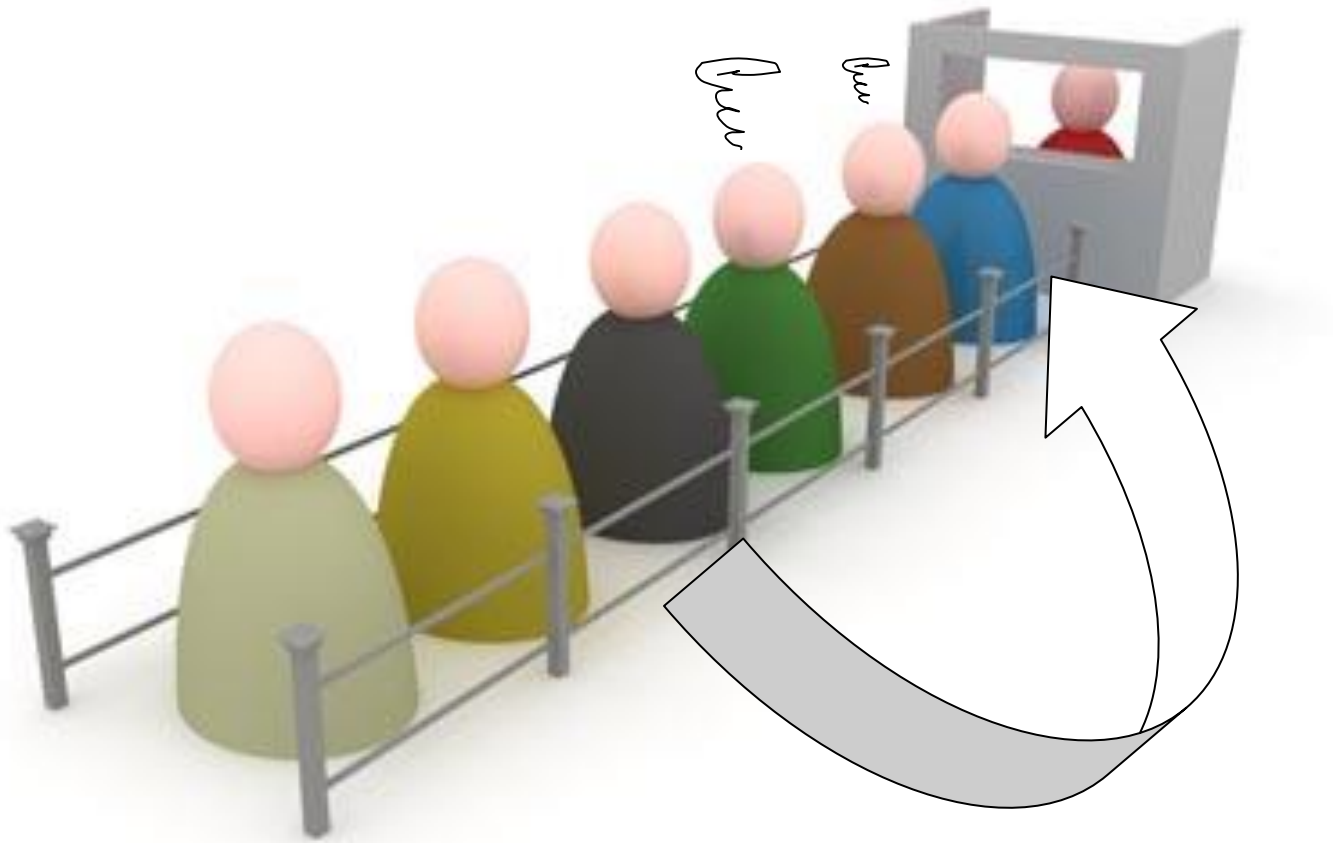
- Tarefas orientadas a processamento (CPU-bound tasks)
  - Uso intensivo do processador
  - Maior parte do tempo nos estados *pronta* e *executando* (ex.:conversão de video, cálculo numérico)
- Tarefas orientadas a entrada/saída (I/O-bound tasks)
  - Passam a maior parte do tempo esperando (estado suspensa) respostas dos dispositivos de entrada/saída (ex.: editores, servidores de rede)

Relembrando...

Uma tarefa pode mudar de comportamento ao longo de sua execução.

Ex.: um conversor de arquivos de áudio WAV→MP3 alterna constantemente entre fases de processamento e de entrada/saída.

Critérios



Como definir o critério?

O desenvolvedor tem de escolher o que priorizar em função do perfil das aplicações.

Muitas vezes as decisões são contraditórias. Ex.: Um Jogo (sistema interativo)

- Quantum baixo $\rightarrow$ tarefa recebe processador mais rapidamente $\rightarrow$ maior interatividade

- Porém: quantum baixo $\rightarrow$ menor η

Critérios de Escalonamento

1. **Justiça**: fazer com que cada processo ganhe seu tempo justo de CPU;
2. **Eficiência**: manter a CPU ocupada “100%” do tempo (se houver demanda);
3. **Tempo de Reposta**: minimizar o tempo de resposta para os usuários interativos;
4. **Tempo de Execução (*turnaround*)**: minimizar tempo decorrido entre a criação da tarefa e seu encerramento, computando todos os tempos de processamento e de espera;
5. **Tempo de espera (*waiting time*)**: minimizar o tempo total perdido pela tarefa na fila de tarefas prontas, aguardando o processador
6. ***Throughput***: maximizar o número de *tarefas* processadas por unidade de tempo.

Critérios de Escalonamento

Muitas vezes, alguns desses critérios são conflitantes:

- Eficiência x Tempo médio de resposta
- Tempo médio de execução x Tempo de espera

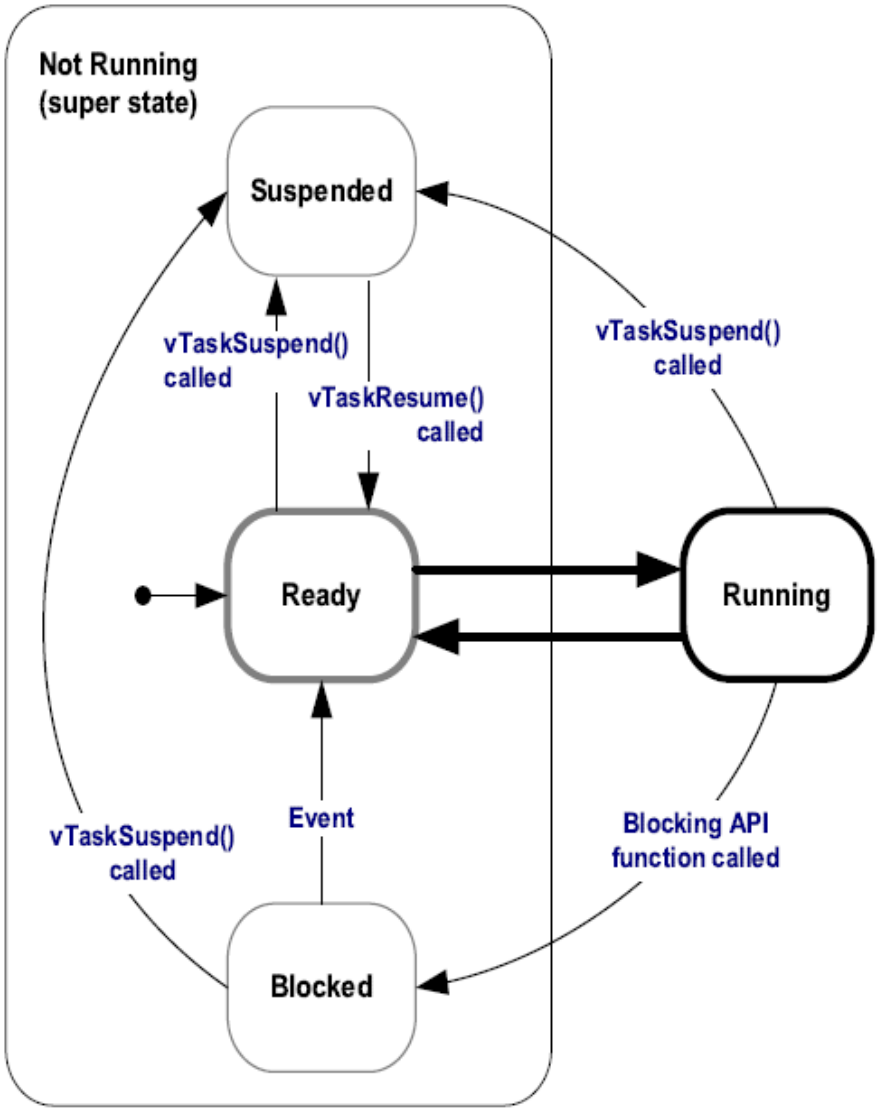
Tipos de Escalonamento

- Preemptivo
- Não preemptivo

Escalonamento preemptivo

- Uma tarefa pode *perder* o processador
  - caso termine seu quantum de tempo ou
  - execute uma chamada de sistema ou
  - caso ocorra uma interrupção que acorde uma tarefa mais prioritária
    - Ex.: tarefa de maior prioridade estava suspensa aguardando um evento

Ex.: FreeRTOS


Not Running
(super state)
Suspended
vTaskSuspend()
called
vTaskResume()
called
vTaskSuspend()
called
Ready
Running
Event
vTaskSuspend()
called
Blocking API
function called
Blocked

Escalonamento não-preemptivo

- A tarefa em execução permanece no processador enquanto tiver condições de executar, só liberando o μP caso:
  - termine de executar;
  - solicite uma operação de entrada/saída;
  - ou libere explicitamente o processador (*sched_yield()*)

Também chamado de cooperativo

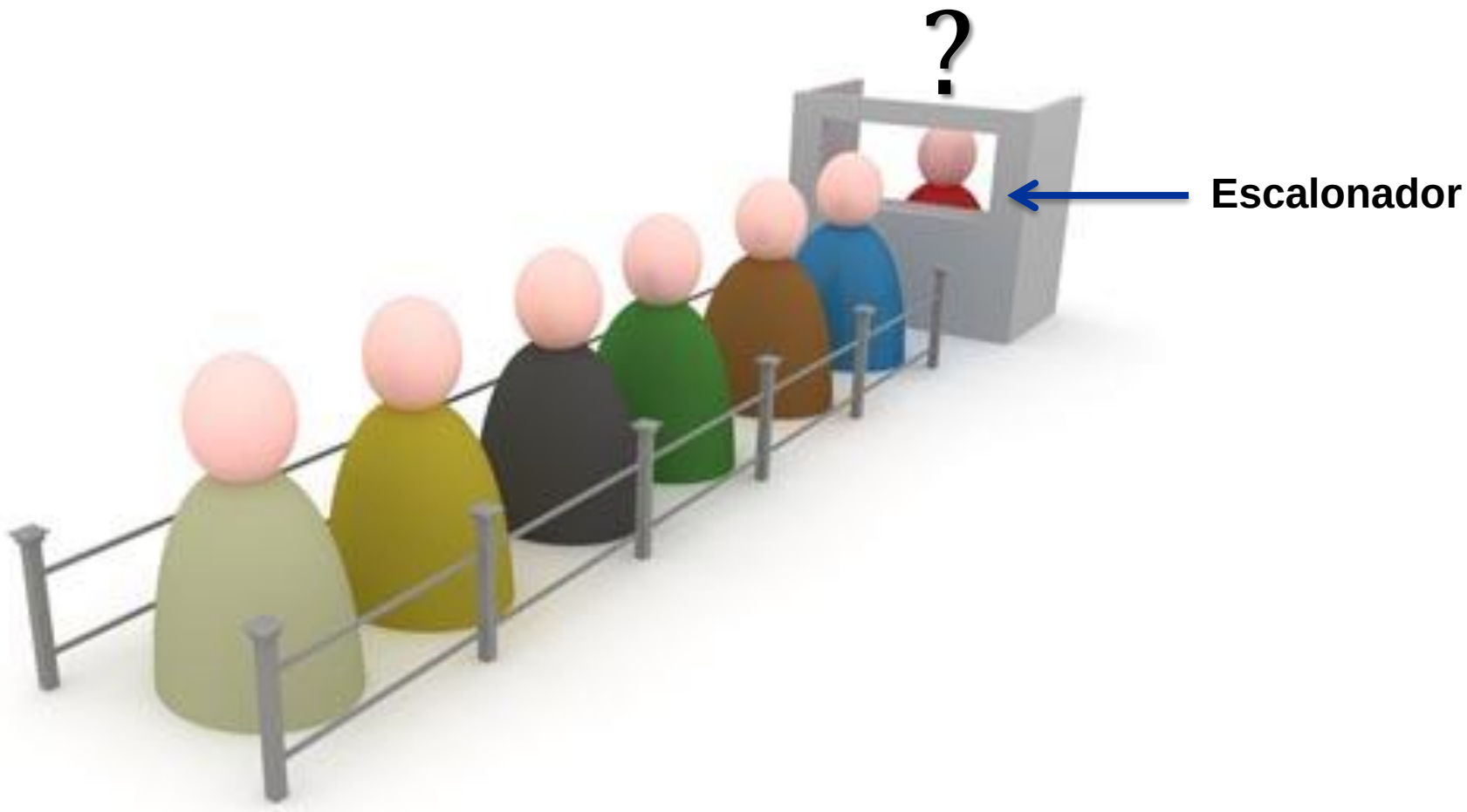
Escalonamento preemptivo

A maioria dos sistemas operacionais de uso geral atuais é preemptiva.

Sistemas mais antigos, como o Windows 3.*, PalmOS 3 e MacOS 8 e 9 operavam de forma cooperativa.

Contiki => Cooperativo (SO embarcado)

Algoritmos de Escalonamento



Algoritmos de Escalonamento

- **FIFO**
- **Round-robin**
- **SJF**
- **Prioridade**

FIFO

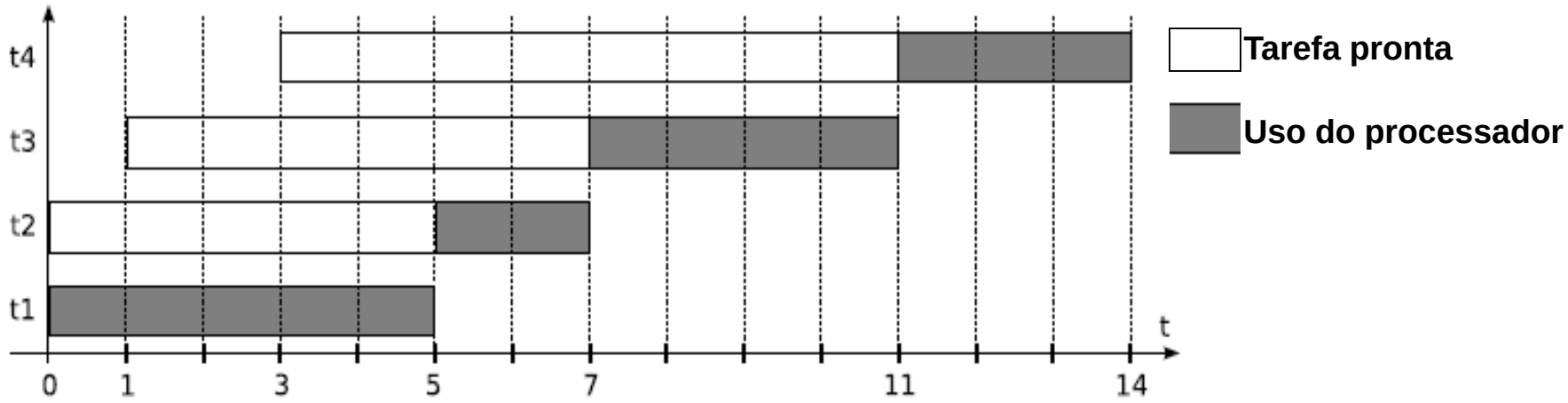
Escalonamento FCFS ou FIFO

- Processos são despachados de acordo com sua ordem de chegada na fila de processos prontos do sistema.
- Uma vez que um processo ganhe a CPU, ele executa até terminar.
- FIFO é uma solução não preemptiva.

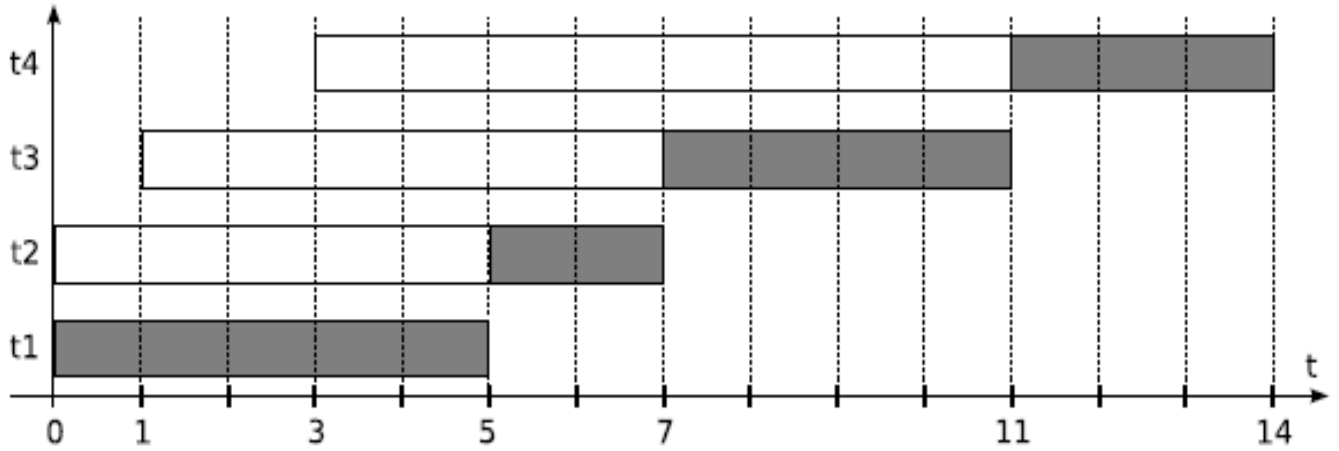
Escalonamento FCFS ou FIFO

Exemplo:

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3



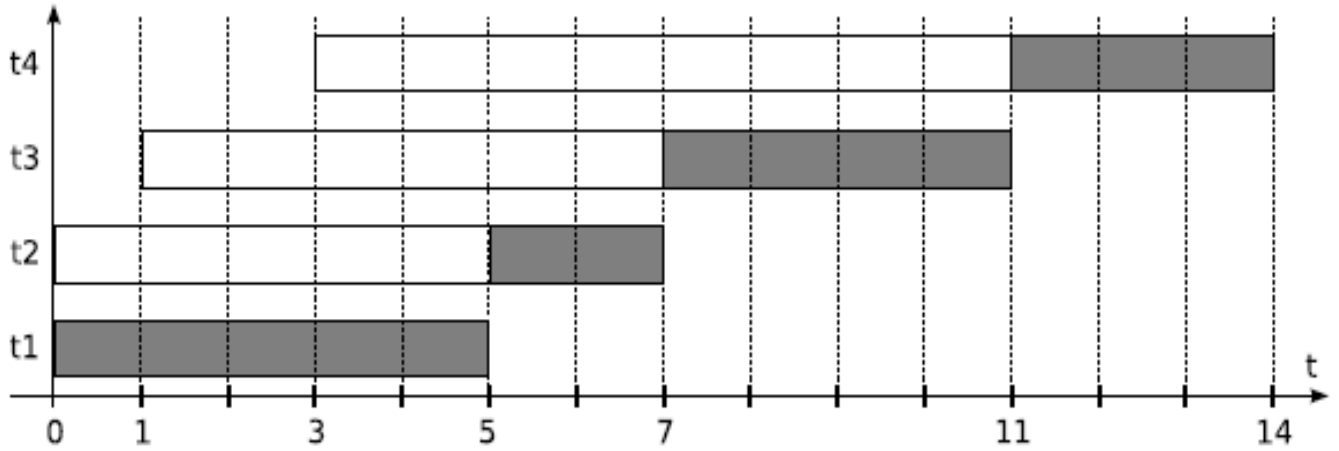
Escalonamento FCFS ou FIFO



Tempo médio de execução

$$T_t = \frac{(5-0)+(7-0)+(11-1)+(14-3)}{4} = 8.25\,s$$

Escalonamento FCFS ou FIFO



Tempo médio de execução

$$T_t = \frac{(5-0)+(7-0)+(11-1)+(14-3)}{4} = 8.25\,s$$

Tempo médio de espera

$$T_\omega = \frac{(0-0)+(5-0)+(7-1)+(11-3)}{4} = 4.75\,s$$

Round-Robin (fila circular)

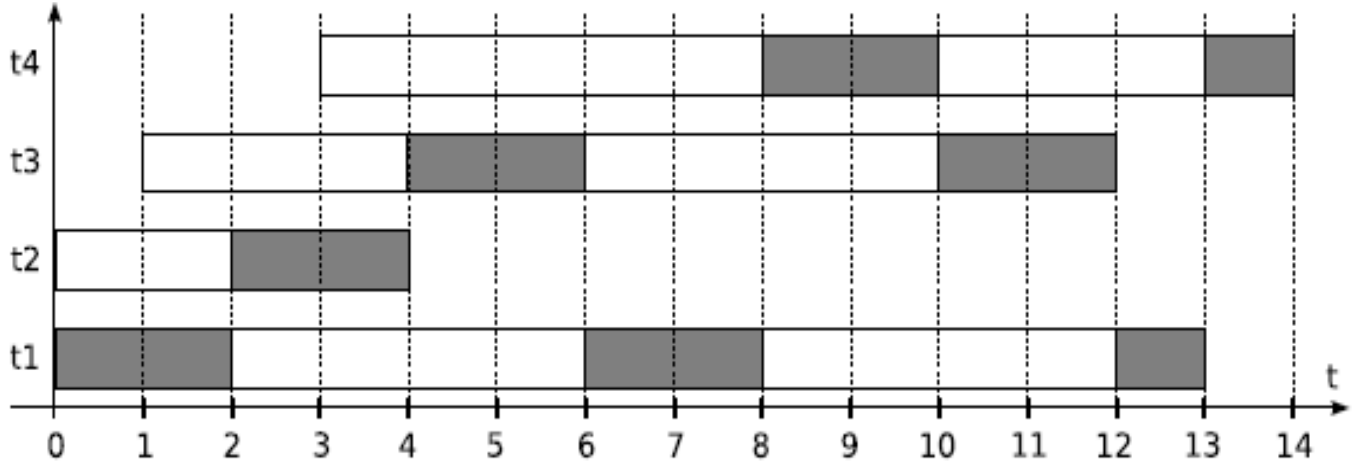
Escalonamento Round Robin (RR)

- Cada processo recebe um intervalo de tempo, chamado quantum, durante o qual ele pode executar.
- Se o processo ainda estiver executando ao final do quantum, a CPU é dada a outro processo.
- Se um processo bloqueou ou terminou antes do final do quantum, a troca de CPU para outro processo é, obviamente, feita assim que o processo bloqueia ou termina.
- É preemptivo.

Escalonamento Round Robin (RR)

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3

Quantum = 2



Tempo médio de execução

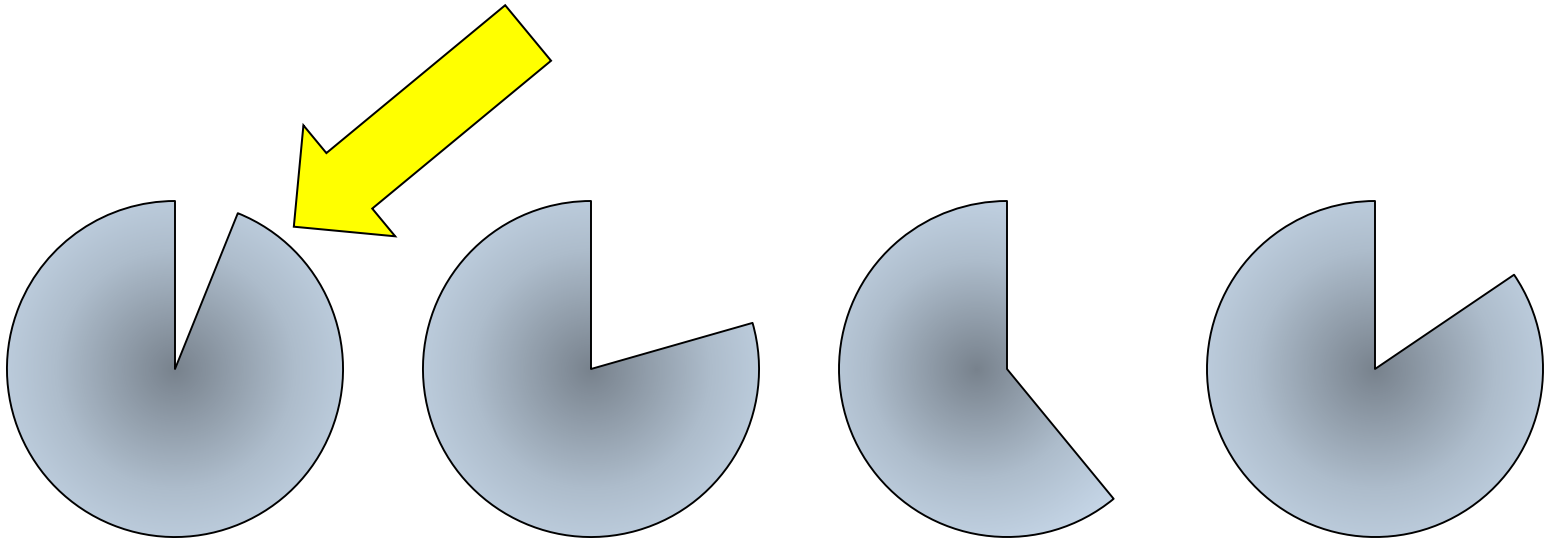
$$T_t = 9.75\,s$$

Tempo médio de espera

$$T_\omega = 6.25$$

Resultado pior que FIFO!!!
Porém distribui melhor o uso do processador
Melhores resultado para aplicações interativas

SJF



Escalonamento Shortest-Job-First (SJF)

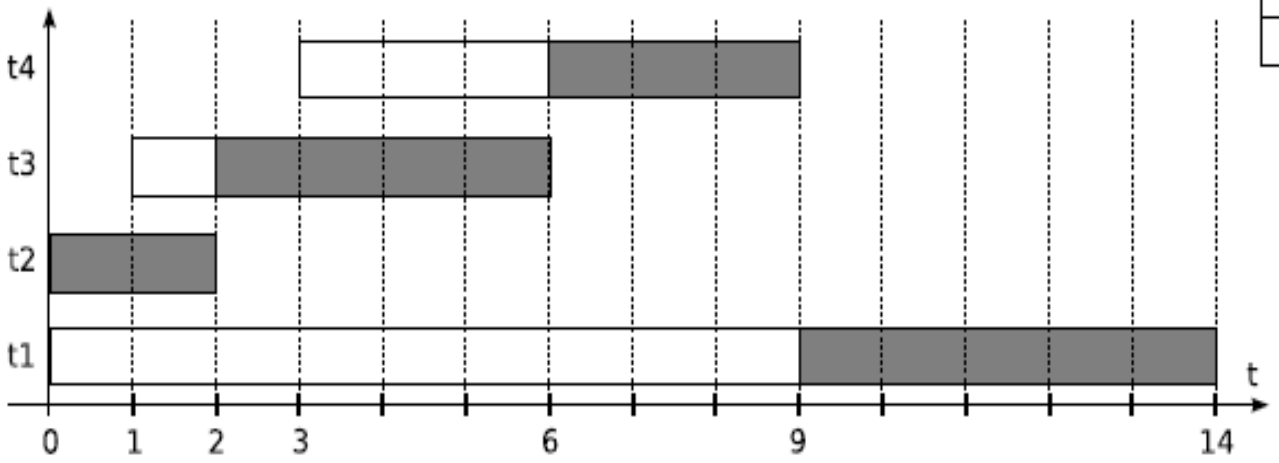
- *Shortest-job-first* é um algoritmo não preemptivo no qual a tarefa (*job*) na fila de espera com o menor tempo total estimado de processamento é executada em seguida.
- SJF reduz o tempo médio de espera sobre o algoritmo FIFO.

Escalonamento Shortest-Job-First (SJF)

- *Shortest-job-first* é um algoritmo não preemptivo no qual a tarefa (*job*) na fila de espera com o menor tempo total estimado de processamento é executada em seguida.
- SJF reduz o tempo médio de espera sobre o algoritmo FIFO.
- Entretanto, os tempos de espera tem uma variância muito grande (são mais imprevisíveis) do que no algoritmo FIFO, especialmente para tarefas longas.
  - Possibilidade de Inanição
- SJF favorece tarefas curtas em *prejuízo* das tarefas maiores.

Escalonamento Shortest-Job-First (SJF)

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3



Tempo médio de execução

$$T_t = 6.75\,s$$

Tempo médio de espera

$$T_\omega = 3.25$$

Prioridade

Escalonamento com Prioridade

Algoritmos vistos anteriormente

-Ordem de chegada (RR, FIFO)
-Duração prevista (SJF)

Prioridade: pode ser definida quantitativamente (número)

-*Escala positiva* – Quanto maior a prioridade maior seu valor
-*Escala negativa* – Quanto menor a prioridade maior seu valor

Escalonamento com Prioridade

Tipos

- Prioridade Fixa sem Preempção
- Prioridade Fixa com Preempção
- Prioridade Dinâmica

Fatores que determinam a prioridade

Fatores Externos (Estático)

Informações providas pelo usuário ou administrador

-Classe (categoria) do usuário (tarefa)
-Valor pago pelo uso do sistema
-Importância da tarefa

Fatores que determinam a prioridade

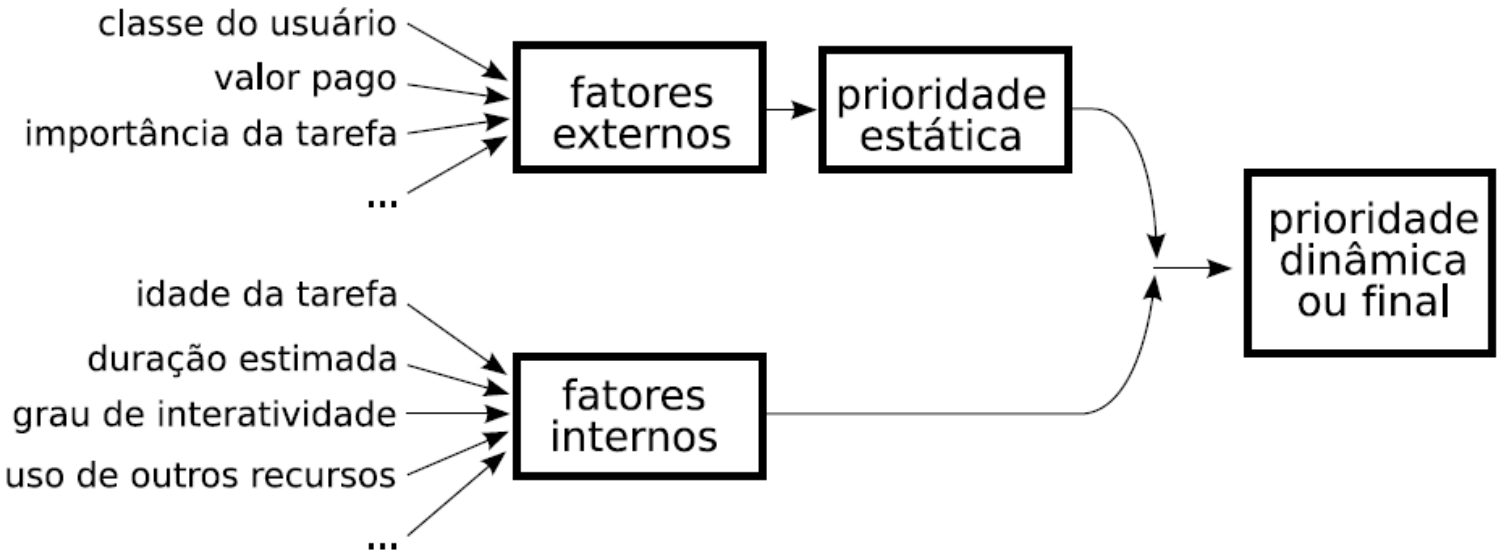
Fatores Internos (Dinâmico)

Informações estimadas pelo *escalonador*

-Idade da tarefa
-Duração (restante) estimada
-Grau de interatividade
-Uso de recursos (memória, arquivos)

Fatores que determinam a prioridade

Fatores Internos e Externos são combinados para definir um valor para a prioridade de cada tarefa



Problemas com prioridades

- Cada processo possui uma prioridade associada, e o processo pronto para executar com a maior prioridade é quem recebe o processador.

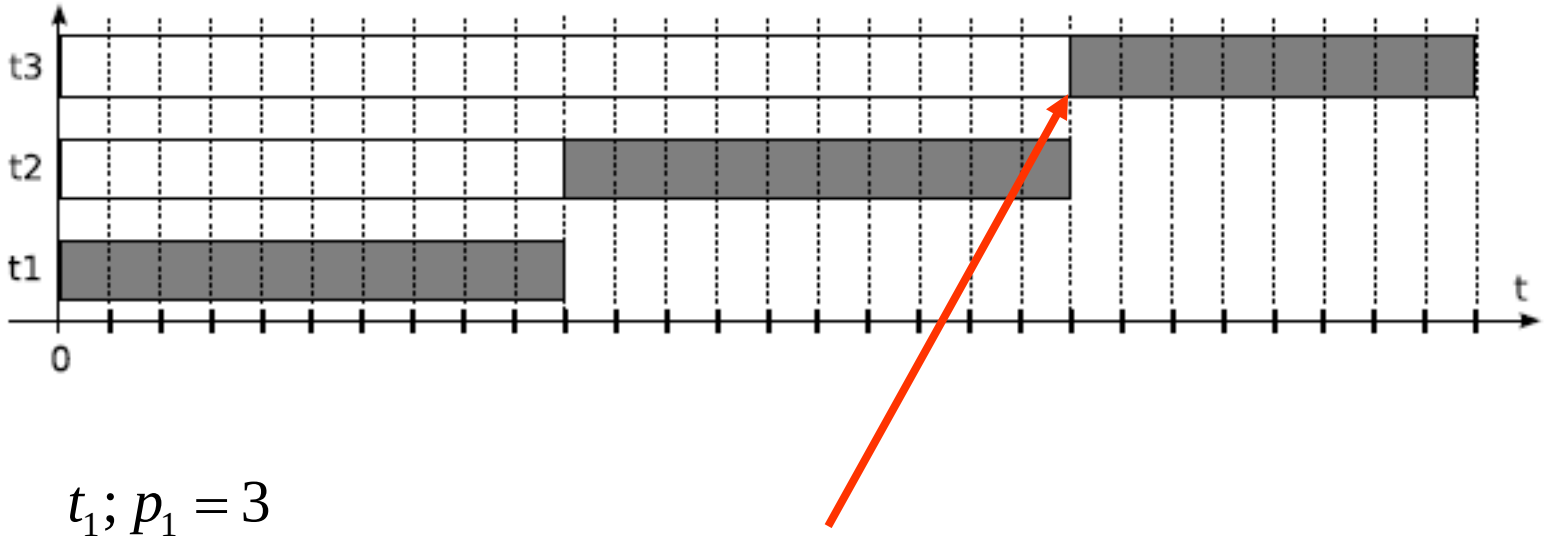
- Para evitar que processos com alta prioridade executem indefinidamente, o escalonador pode decrementar a prioridade do processo atualmente em execução a cada *tick* de relógio.

Problemas com prioridades

- Se as prioridades não forem ajustadas de tempos em tempos, os processos nas classes de prioridades mais baixas podem sofrer o fenômeno que chamamos *starvation*
  - O processo nunca recebe o processador, pois sua vez nunca chega).

Inanição (Starvation)

Se antes de t_3 ser iniciada outra tarefa de mais alta prioridade ficar pronta, t_3 não será executada.



$t_1; p_1 = 3$

$t_2; p_2 = 2$

$t_3; p_3 = 1$

Inanição (Starvation)

Duas coisas podem acontecer:

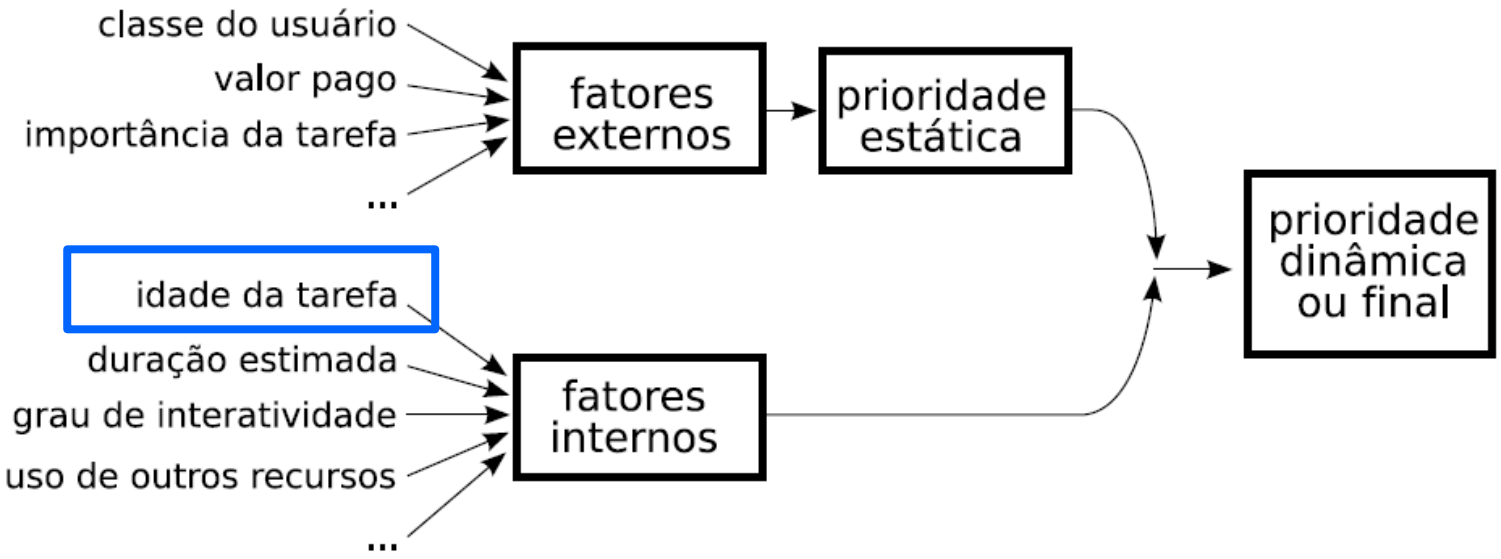
-A tarefa é executada às duas da madrugada de um domingo!

-O sistema trava e perde as tarefas de baixa prioridade não concluídas.

Possível solução → *Task Aging*

Solução

⇒ **Envelhecimento (*Task Aging*)**



Solução

⇒ **Envelhecimento (*Task Aging*)**

Definições:

t_i : tarefa i

pe_i : prioridade estática de t_i

pd_i : prioridade dinâmica de t_i

N : número de tarefas no sistema

Solução

⇒ Envelhecimento (*Task Aging*)

Definições:

t_i : tarefa i
pe_i : prioridade estática de t_i
pd_i : prioridade dinâmica de t_i
N : número de tarefas no sistema

Quando uma nova tarefa t_n ingressa no sistema:

$pe_n \leftarrow$ *prioridade inicial default*
$pd_n \leftarrow pe_n$

Solução

⇒ Envelhecimento (*Task Aging*)

Definições:

t_i : tarefa i
pe_i : prioridade estática de t_i
pd_i : prioridade dinâmica de t_i
N : número de tarefas no sistema

Quando uma nova tarefa t_n ingressa no sistema:

$pe_n \leftarrow$ *prioridade inicial default*
$pd_n \leftarrow pe_n$

Para escolher a próxima tarefa a executar t_p:

escolher $t_p \mid pd_p = max_{i=1}^{N}(pd_i)$
$pd_p \leftarrow pe_p$
$\forall i \neq p : pd_i \leftarrow pd_i + \alpha$ → *fator de envelhecimento*

Atenção: + problemas com prioridades (estáticas)

- Cada processo possui uma prioridade associada, e o processo pronto para executar com a maior prioridade é quem recebe o processador.

- Em sistemas interativos, a intuição associada às prioridade estáticas (fatores externos), que é a de proporcionalidade na divisão do tempo de processamento, pode levar a erro de projeto.

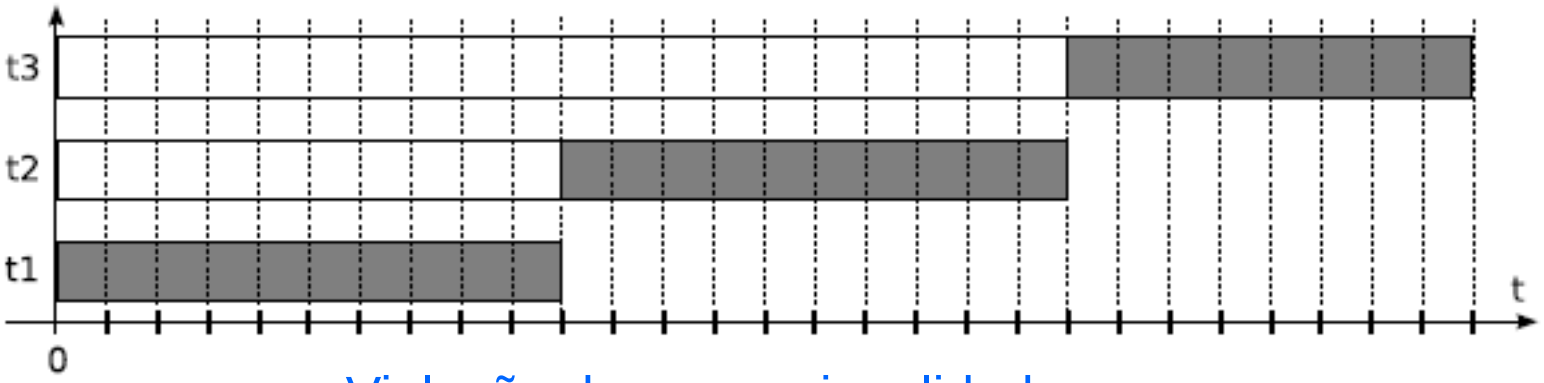
Ex.: Sistemas interativos

Considere: $t_1; p_1 = 3$

$t_2; p_2 = 2$

$t_3; p_3 = 1$

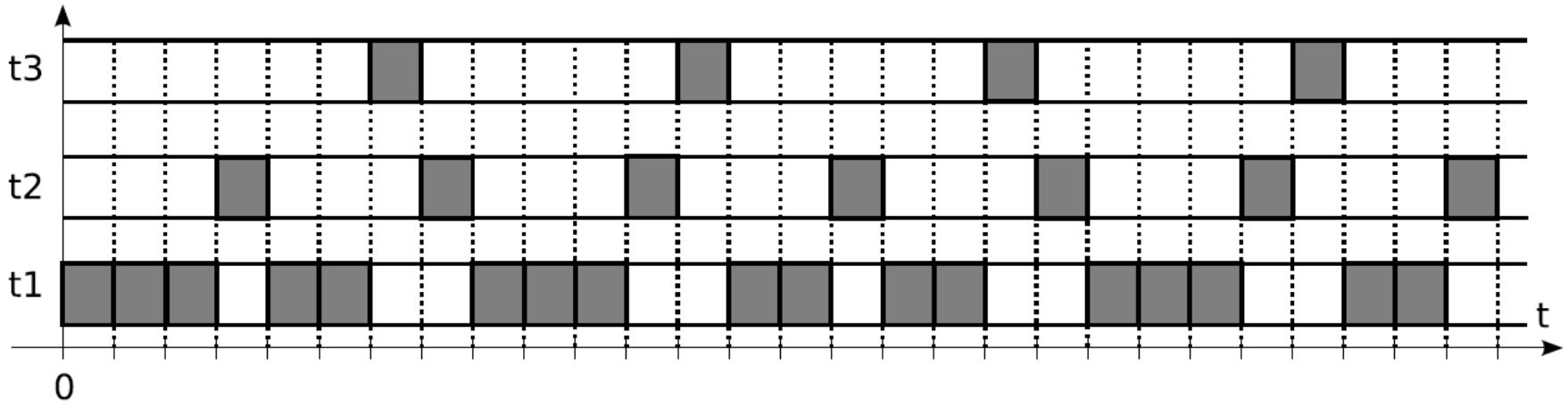
Imagina-se que a divisão do processador será de tal modo que o uso do processador ocorrerá de forma proporcional → t_1:50% - t_2:33.3% - t_3:16.7%



Violação da proporcionalidade

⇒ Envelhecimento (*Task Aging*)

$$p(t_1) > p(t_2) > p(t_3)$$

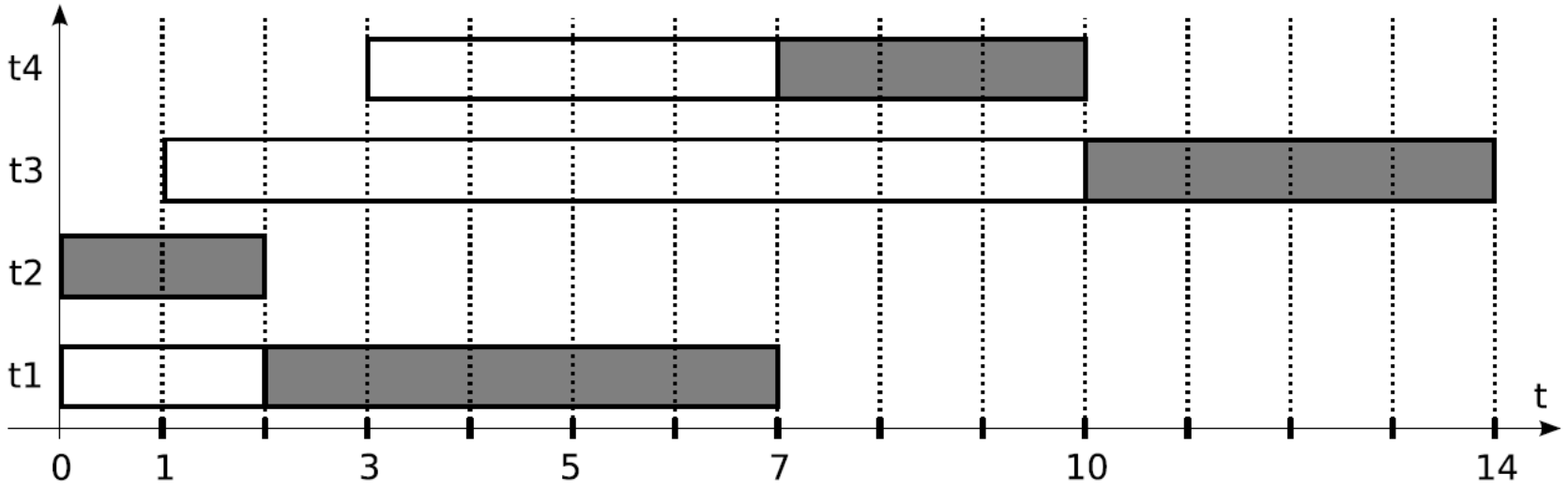


t_1:60% - t_2:25% - t_3:15%

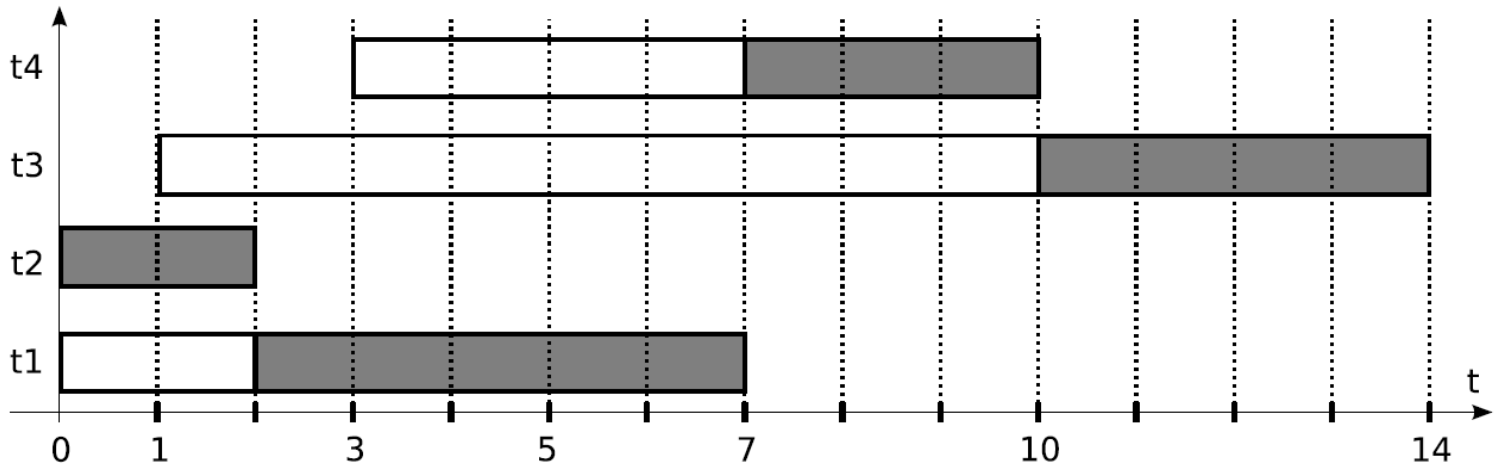
Tempos médios com prioridades

i - Prioridade sem preempção

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3
prioridade	2	3	1	4



Tempos médios com prioridades



$$T_t = \frac{t_t(t_1) + t_t(t_2) + t_t(t_3) + t_t(t_4)}{4} = \frac{(7-0) + (2-0) + (14-1) + (10-3)}{4}$$

$$= \frac{7+2+13+7}{4} = \frac{29}{4} = 7.25s$$

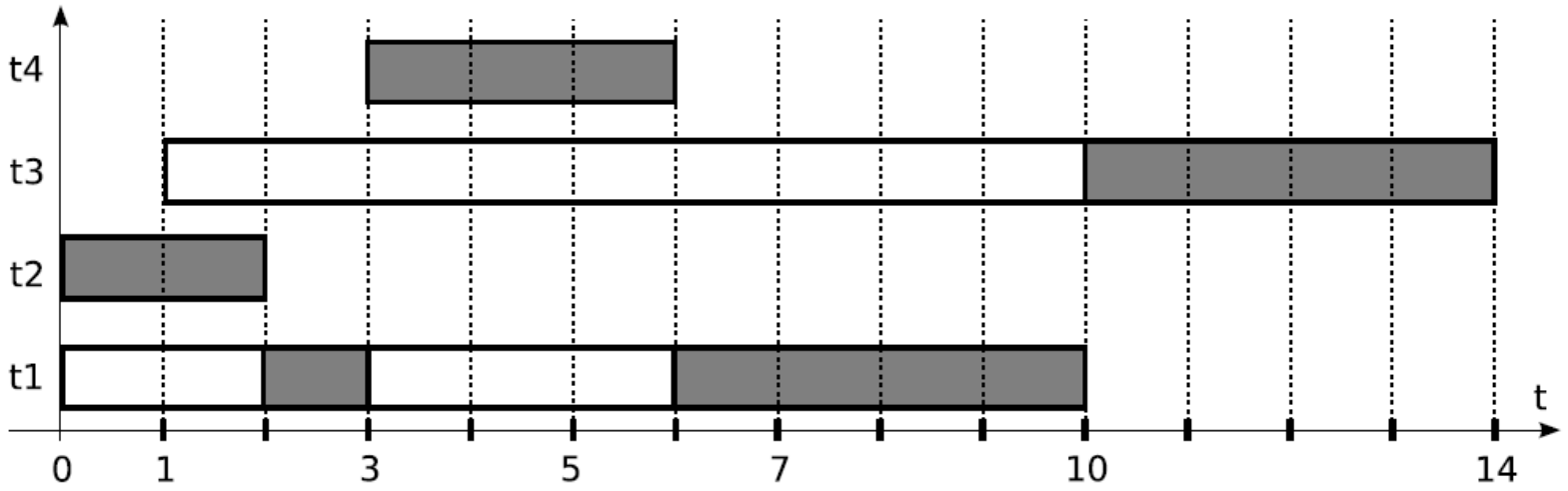
$$T_w = \frac{t_w(t_1) + t_w(t_2) + t_w(t_3) + t_w(t_4)}{4} = \frac{(2-0) + (0-0) + (10-1) + (7-3)}{4}$$

$$= \frac{2+0+9+4}{4} = \frac{15}{4} = 3.75s$$

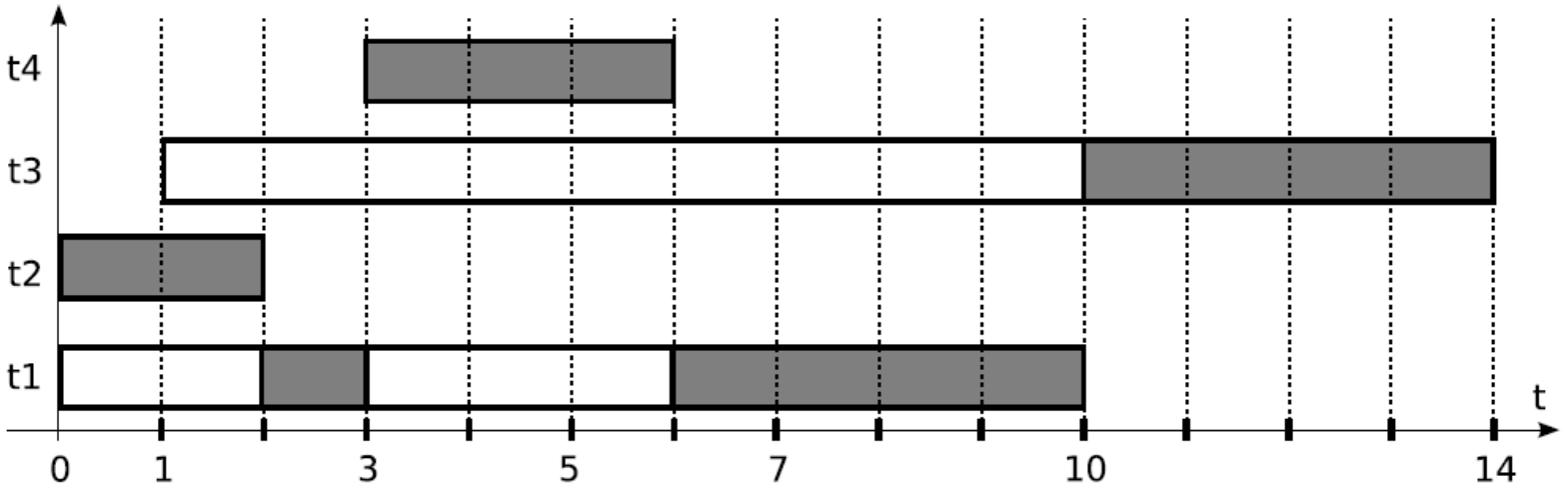
Tempos médios com prioridades

ii - Prioridade com preempção

tarefa	t_1	t_2	t_3	t_4
ingresso	0	0	1	3
duração	5	2	4	3
prioridade	2	3	1	4



Tempos médios com prioridades



$$T_t = \frac{t_t(t_1) + t_t(t_2) + t_t(t_3) + t_t(t_4)}{4} = \frac{(10-0) + (2-0) + (14-1) + (6-3)}{4}$$

$$= \frac{10 + 2 + 13 + 3}{4} = \frac{28}{4} = 7s$$

$$T_w = \frac{t_w(t_1) + t_w(t_2) + t_w(t_3) + t_w(t_4)}{4} = \frac{5 + 0 + 9 + 0}{4} = \frac{14}{4} = 3.5s$$