Inversão de prioridades

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Maio, 2017

Sumário

Estados dos processos

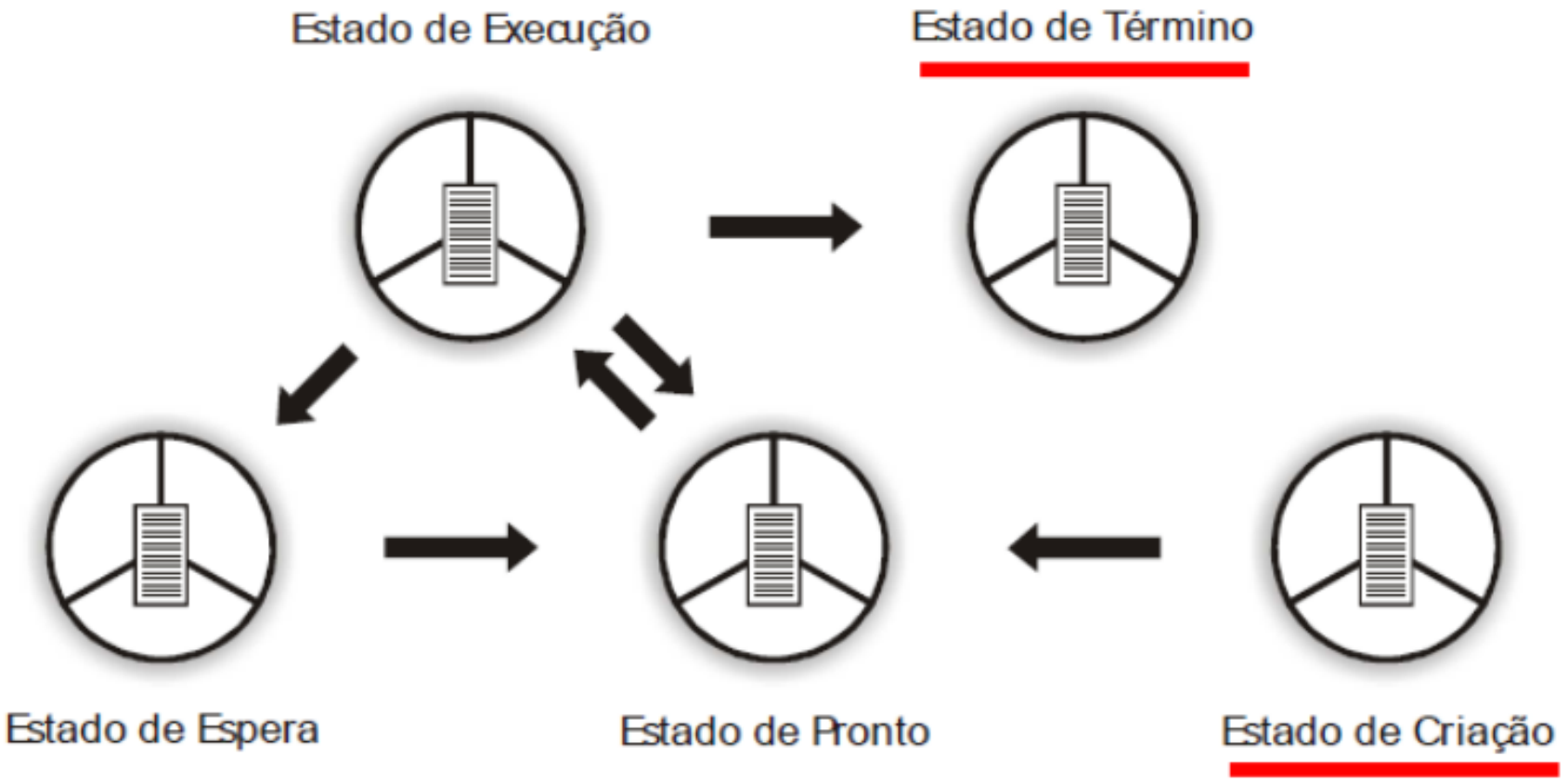

Figura 1.1: Estados dos processos [Chagas, 2016]

Escalonamento

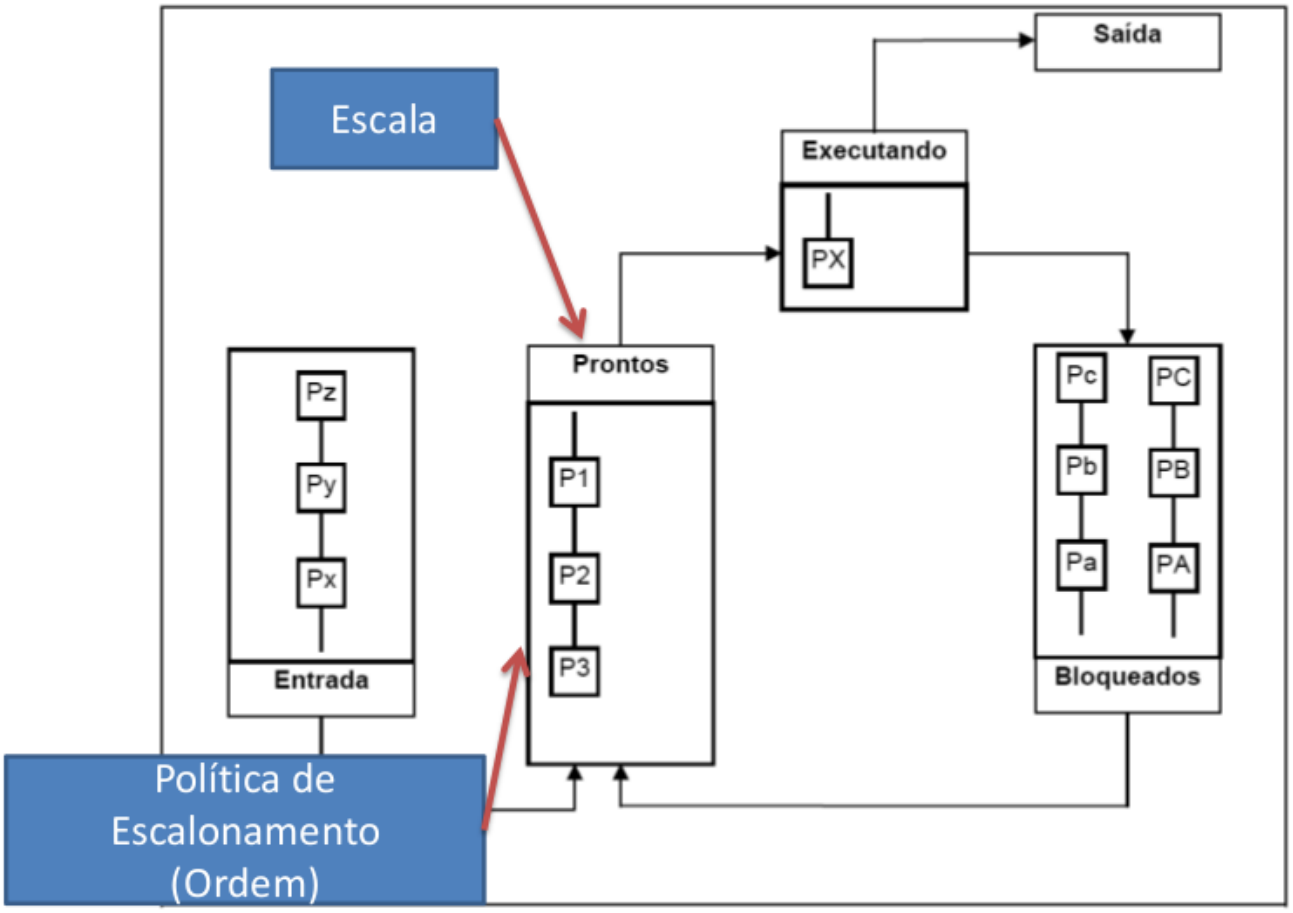

Figura 1.2: Descrição do modelo de escalonamento [Chagas, 2016]

Definição

- Escalonar: Ordenar as tarefas na fila de pronto.

Definição

- Escalonar: Ordenar as tarefas na fila de pronto.
- Os algoritmos de escalonamento podem ser [FARINES and MELO, 2000]:

Preemptivos Tarefas podem ser interrompidas em tempo de execução;

Não preemptivos Tarefas não podem ser interrompidas por outras mais prioritárias;

Estáticos Escalonamento calculado com base em parâmetros fixos atribuídos às tarefas;

Dinâmicos Baseados em parâmetros que mudam em tempo de execução.

Classificação

- Em relação aos parâmetros enviados para as tarefas, os algoritmos podem ser:

  on-line A escala é produzida em tempo de execução;

  off-line A escala é produzida em tempo de projeto.

- Os problemas de escalonamento de tempo real podem ser reduzidos a uma solução polinomial (NP-Completos)

Carga computacional [Chagas, 2016]

- Definição: Somatório dos tempos de computação na fila de pronto.

Carga Estática (Limitada)

- Todas as tarefas são bem conhecidas em tempo de projeto;
- Modeladas através de tarefas periódicas e esporádicas .

Carga Dinâmica (Ilimitada)

- Características de chegada da tarefa não pode ser antecipada;
- Modeladas através de tarefas aperiódicas.

Abordagens

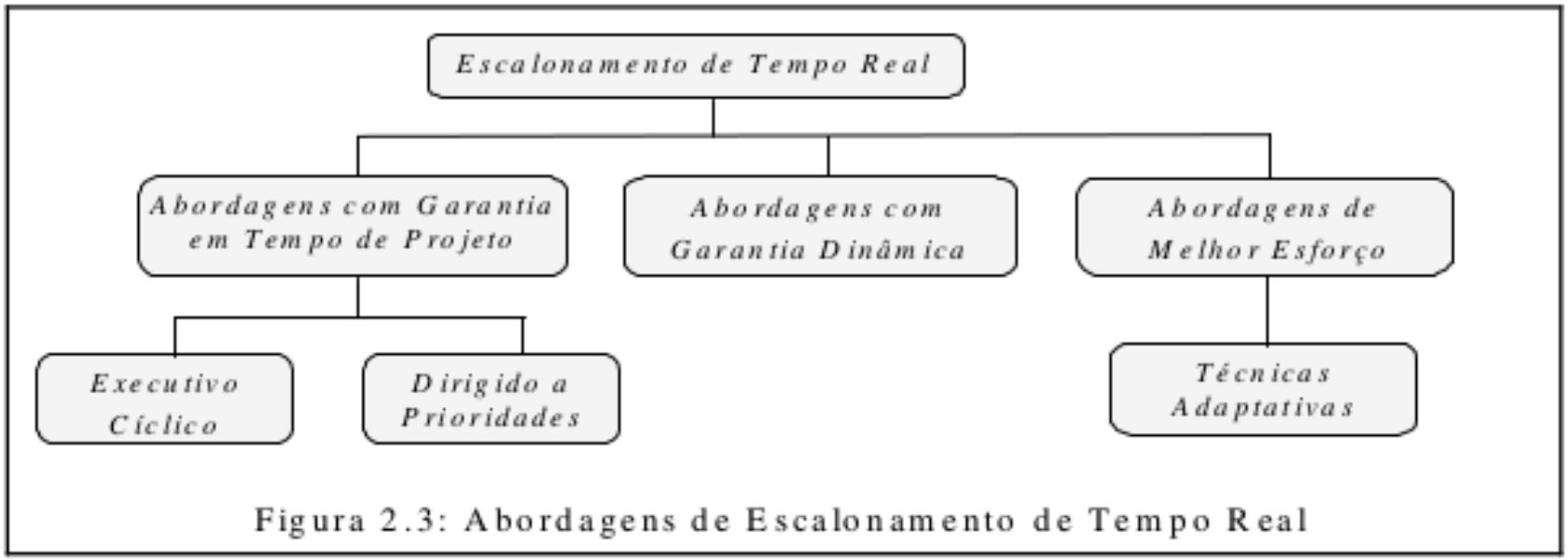

Figura 1.3: Abordagens de escalonamento em tempo real [FARINES and MELO, 2000]

Abordagens de escalonamento [Chagas, 2016]

Garantia em tempo de projeto I

- Carga computacional dinâmica;
- Teste de escalonabilidade em tempo de projeto.

Garantia em tempo de execução II

- Carga computacional dinâmica;
- Sistemas críticos que operam em ambientes não determinísticos.

Abordagens de melhor esforço III

- Teste de escalonabilidade em tempo de execução;
- Não existe previsão de pior caso e não consegue prever recursos para todas as situações de carga.

Escalonamento estático [Chagas, 2016]

- Escalonamento estático ou executivo cíclico;
- A escala é definida durante a fase do projeto (escalonamento *offline*);
- Tempo de processador atribuído a cada tarefa;
- Garantia de escalonabilidade fornecida pela inspeção da lista de escalonamento (*deadline* da tarefa).

Dirigido a prioridades

- Dirigidos de acordo com suas prioridades em tempo de execução;
- Prioridades fixas: RM ou DM;
- Prioridades dinâmicas: EDF;
- Preemptivos ou não preemptivos.

Teste de escalonabilidade

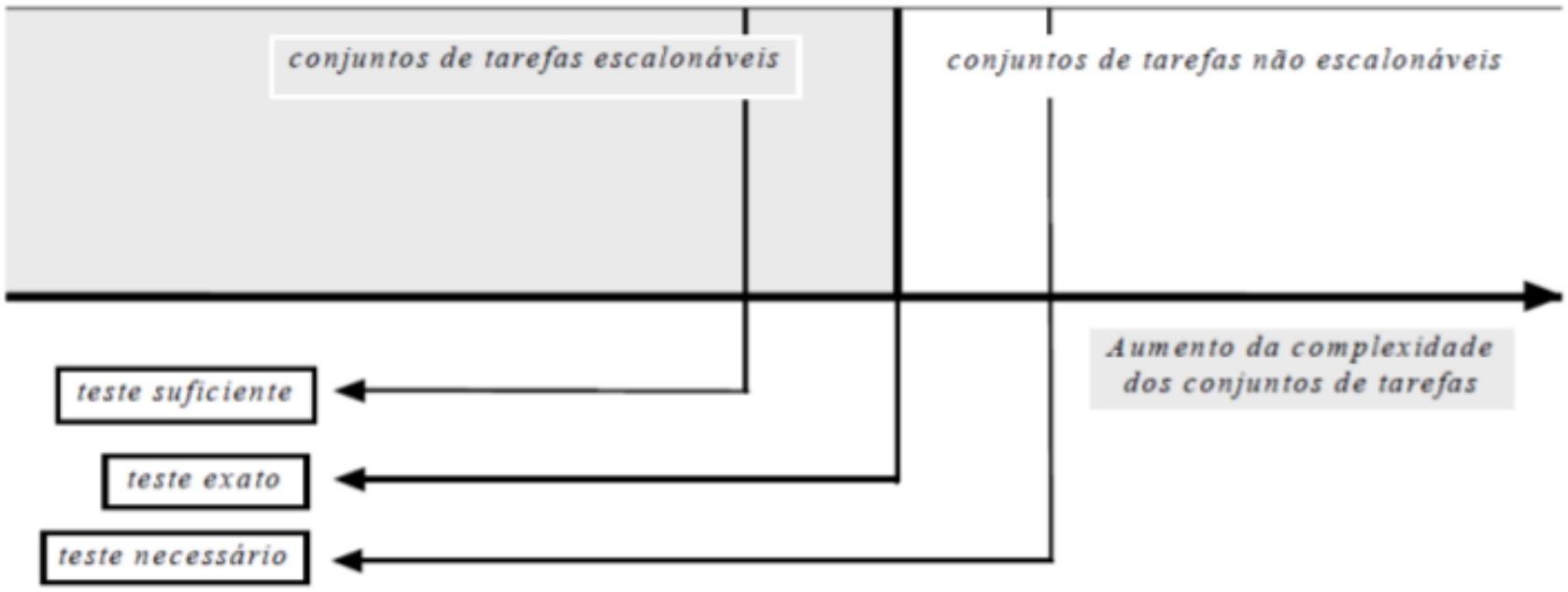

Figura 1.4: Teste de escalabilidade real [FARINES and MELO, 2000]

Definições

- Até o momento consideramos somente tarefas independentes;
- Isso não constitui a realidade da programação em tempo real;
- Em um ambiente multitarefas o compartilhamento de recursos é implícito;
- Técnicas de compartilhamento de recursos;
- Bloqueio em tarefas mais prioritárias: inversão de prioridades;

Introdução

- Os sistemas de computadores têm inúmeros recursos adequados ao uso de somente um processo a cada vez;
- Em ambiente de multiprogramação diversos processos podem competir por um número finito de recursos;
- Um processo em espera não poderá mudar de estado enquanto estiver aguardando algum recurso alocado por outro processo também em espera.

Definição [Favacho, 2009]

- Um conjunto de processos estará em situação de deadlock se todos os processos pertencentes ao conjunto estiverem esperando por um evento que somente um outro processo desse mesmo conjunto poderá fazer acontecer;
- O número de processos, bem como, o número e tipo dos recursos não são importantes;
- Isso é válido para qualquer tipo de recurso, tanto para hardware como para software.

Condições para deadlock

- O *deadlock* acontece em quatro situações [Tanenbaum and Machado Filho, 1995]:
  1. Exclusão mútua: apenas um processo de cada vez pode utilizar o recurso;
  2. Prende e espera: um processo bloqueia os recursos que precisa, e aguarda pelos que estão sendo utilizados por outros processos;
  3. Não preempção: um recurso pode ser liberado apenas voluntariamente pelo processo, após o mesmo ter completado sua tarefa;
  4. Espera circular: cada um dos processos espera um recurso que está sendo usado por outro processo em uma fila.

Deadlock (grafo)

- Exemplo da Figura 20:
  (a) Processo de posse de um recurso
  (b) Processo requisitando um recurso;
  (c) Impasse (*deadlock*).

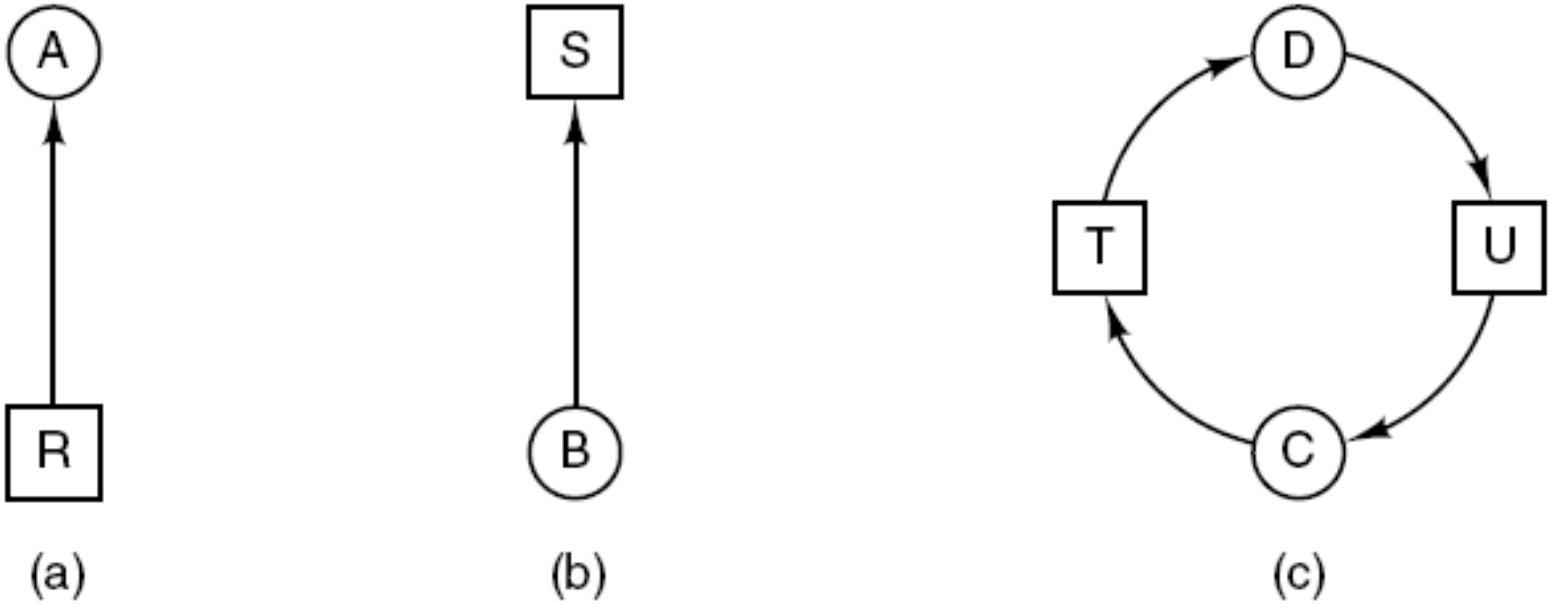

Figura 2.1: Grafo de alocação de recursos [Tanenbaum and Machado Filho, 1995]

Exemplo de deadlock I

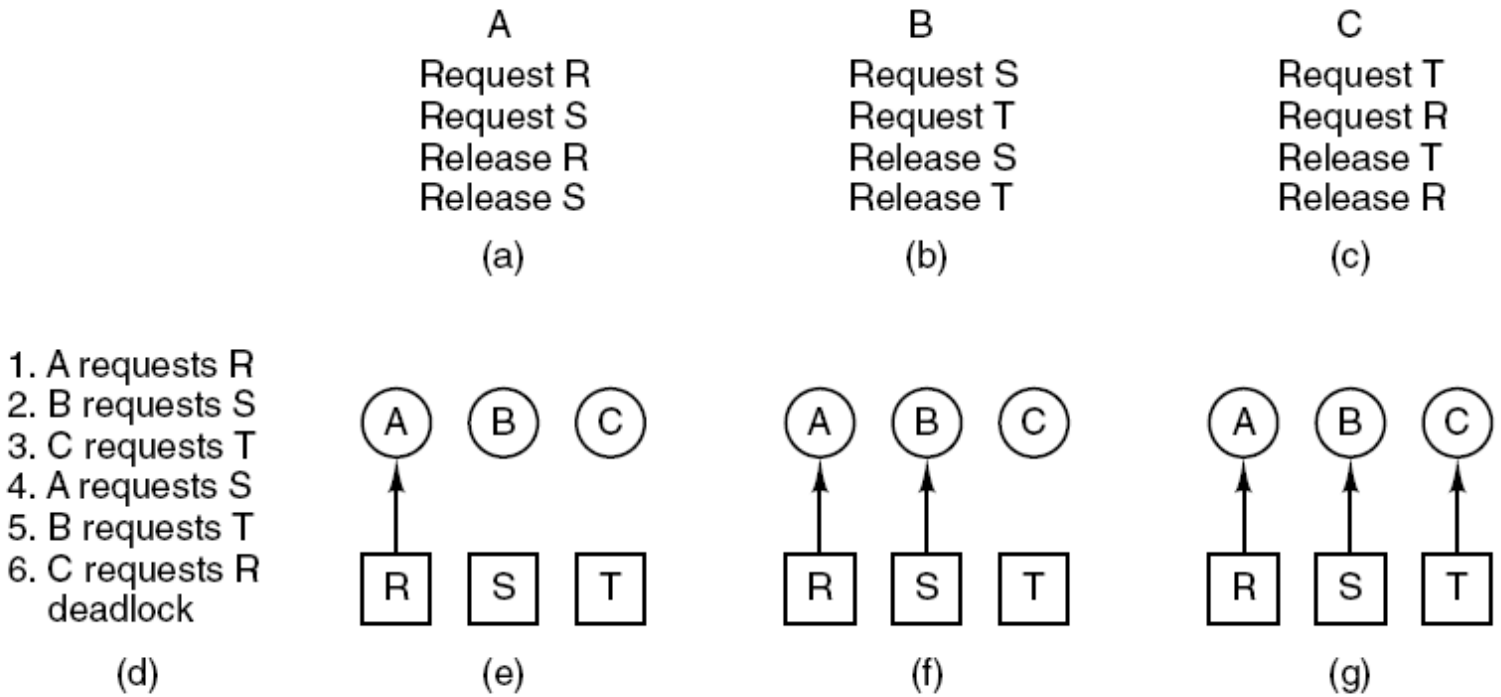

Figura 2.2: Exemplo de como ocorre um impasse e como ele pode ser evitado [Tanenbaum and Machado Filho, 1995]

Exemplo de deadlock II

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
   deadlock

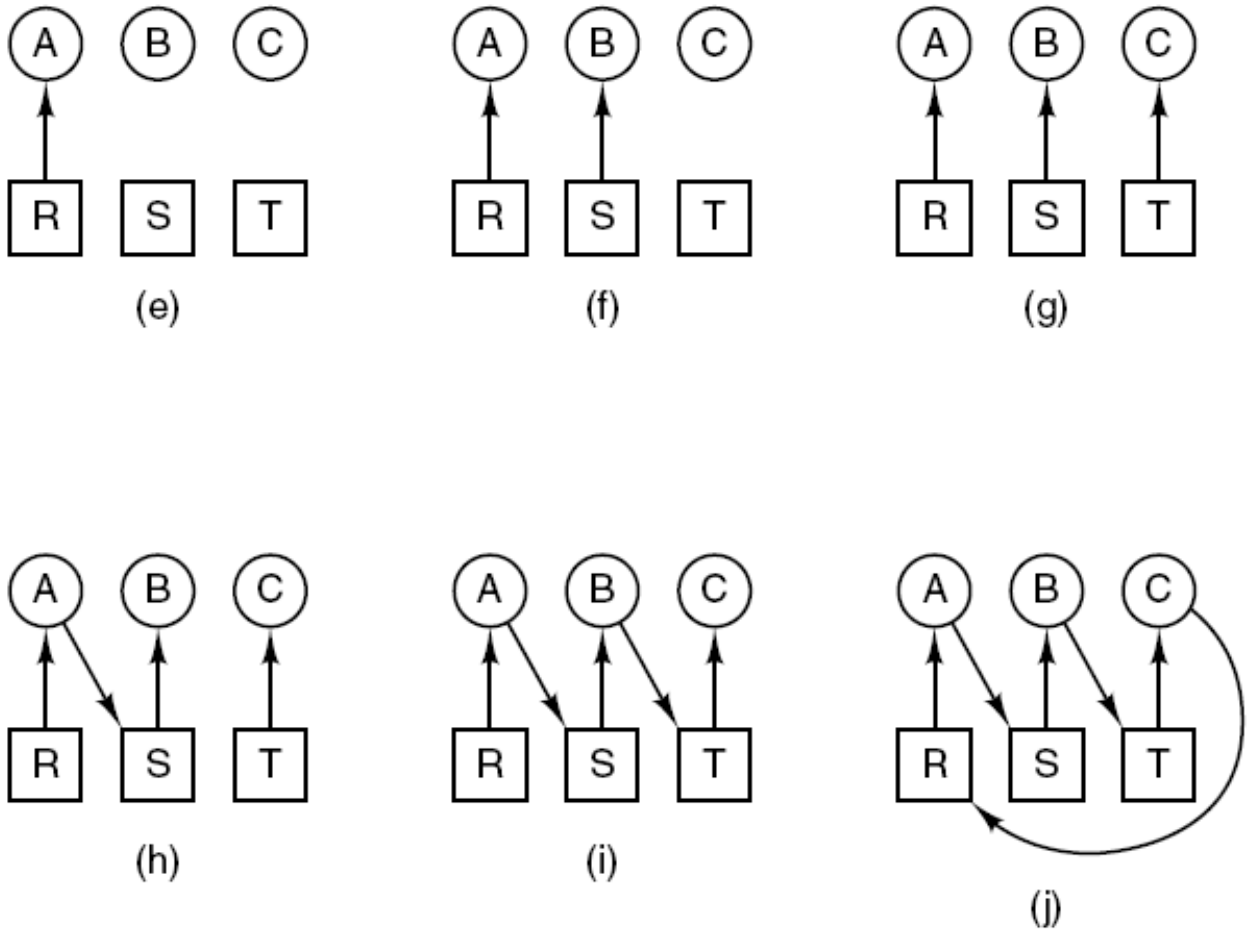

Figura 2.3: Exemplo de como ocorre um impasse e como ele pode ser evitado (cont.) [Tanenbaum and Machado Filho, 1995]

Exemplo de deadlock III

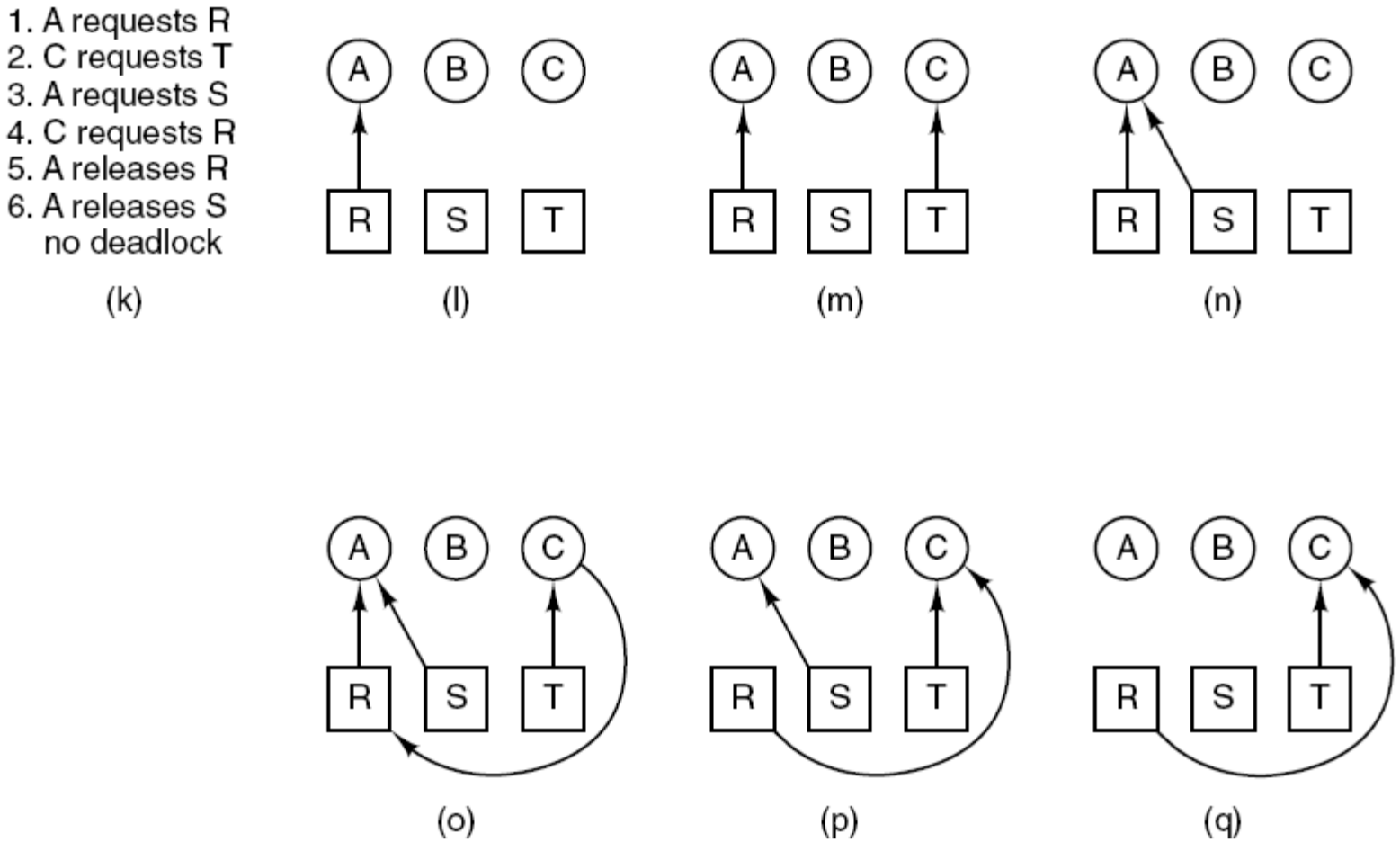

Figura 2.4: Exemplo de como ocorre um impasse e como ele pode ser evitado (cont.) [Tanenbaum and Machado Filho, 1995]

Conceitos

- As tarefas devem ser definidas seguindo uma prioridade nominal ou estática;
- Utilização de algum algoritmo de prioridade fixa: RM;
- Prioridades estáticas versus prioridades ativas.

Inversão de prioridades

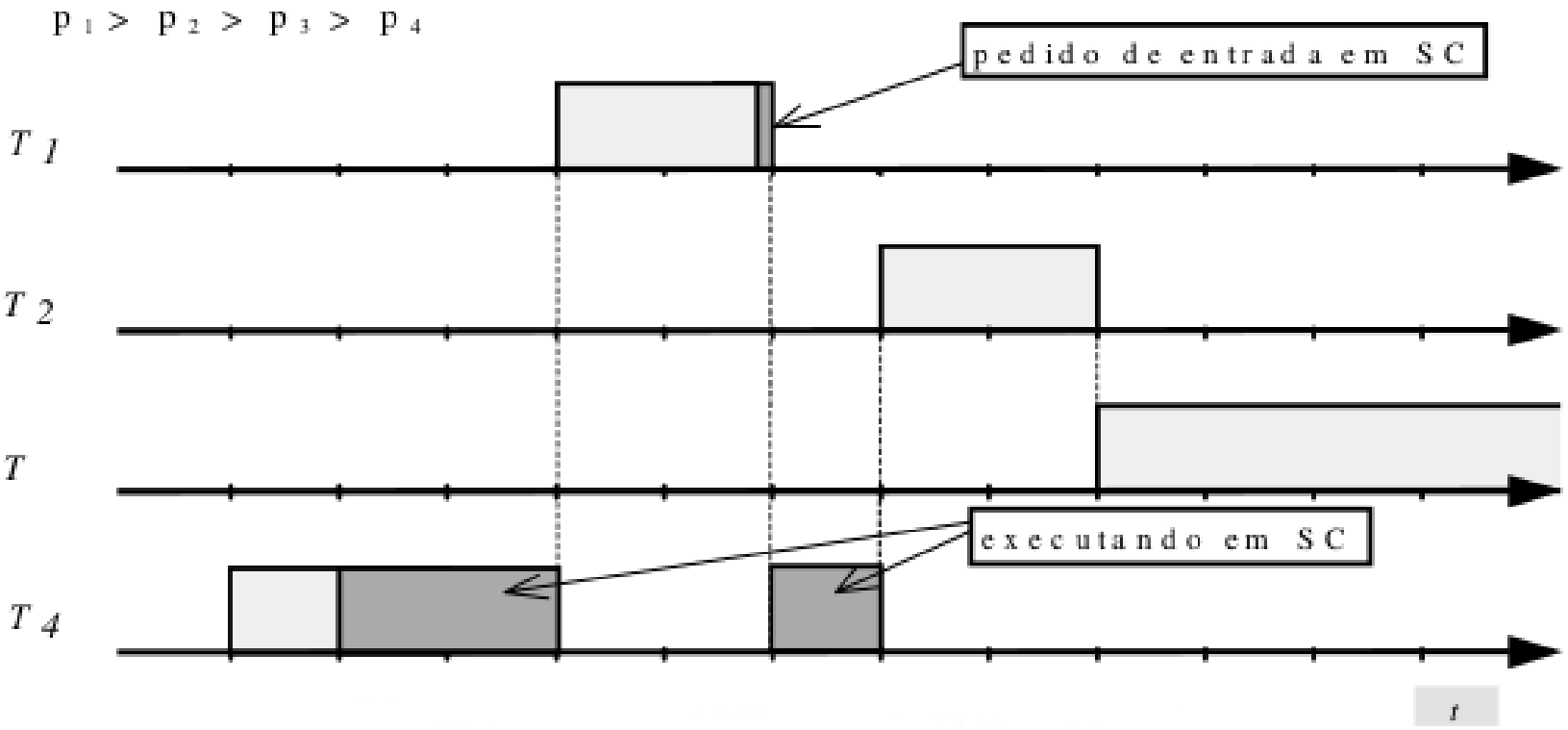

Figura 2.5: Técnica de inversão de prioridades [FARINES and MELO, 2000]

Inversão de prioridades (fluxo)

- Considere que as tarefas T_1, T_2, $T_3 e T_4$ estão em ordem crescente de seus períodos;
- Utilização da escala RM para organizar as prioridades;
- T_1 e T_4 compartilham um recurso guardado por um mecanismo de exclusão mútua;
- O bloqueio que T_1 sofre pelo acesso anterior de T_4 ao recurso compartilhado é chamado de inversão de prioridade;
- Mesmo liberada, a tarefa T_1 não consegue evoluir por conta do bloqueio;
- T_1 sofre interferência de T_2 e T_3 durante o bloqueio
- As preempções de T_2 e T_3 sobre T_4 dificultam o cálculo do tempo de bloqueio para T_1.

Necessidades

- Quando as tarefas são dependentes, a inversão de prioridade é inevitável;
- Limitar a possibilidade de inversão de prioridades é importante;
- Regras de compartilhamento de recursos no tempo de pior caso;
- O tempo de pior caso sempre deve ser conhecido.

Protocolo de herança de prioridade – PHP

- Determina que as tarefas sejam definidas possuindo prioridade nominal ou estática;
- Utilização de política de atribuição de prioridade fixa em conjunto com uma prioridade dinâmica ou ativa;
- Inicialmente todas as tarefas possuem prioridades estáticas coincidindo com as prioridades ativas.

Algoritmo do PHP

- Quando uma tarefa T_i é bloqueada sua prioridade é transferida a uma tarefa T_j;
- Quando T_j reassume executa com a prioridade herdada de T_i ($p_j = p_i$);
- Ao executar, a tarefa herdada sempre mantém a prioridade mais alta de bloqueio;
- Ao liberar a região crítica, T_j novamente passa a ter a prioridade ativa igual à prioridade nominal, ou;
- Assume a prioridade mas alta das tarefas que ainda estejam sob seu bloqueio.

Protocolo PHP

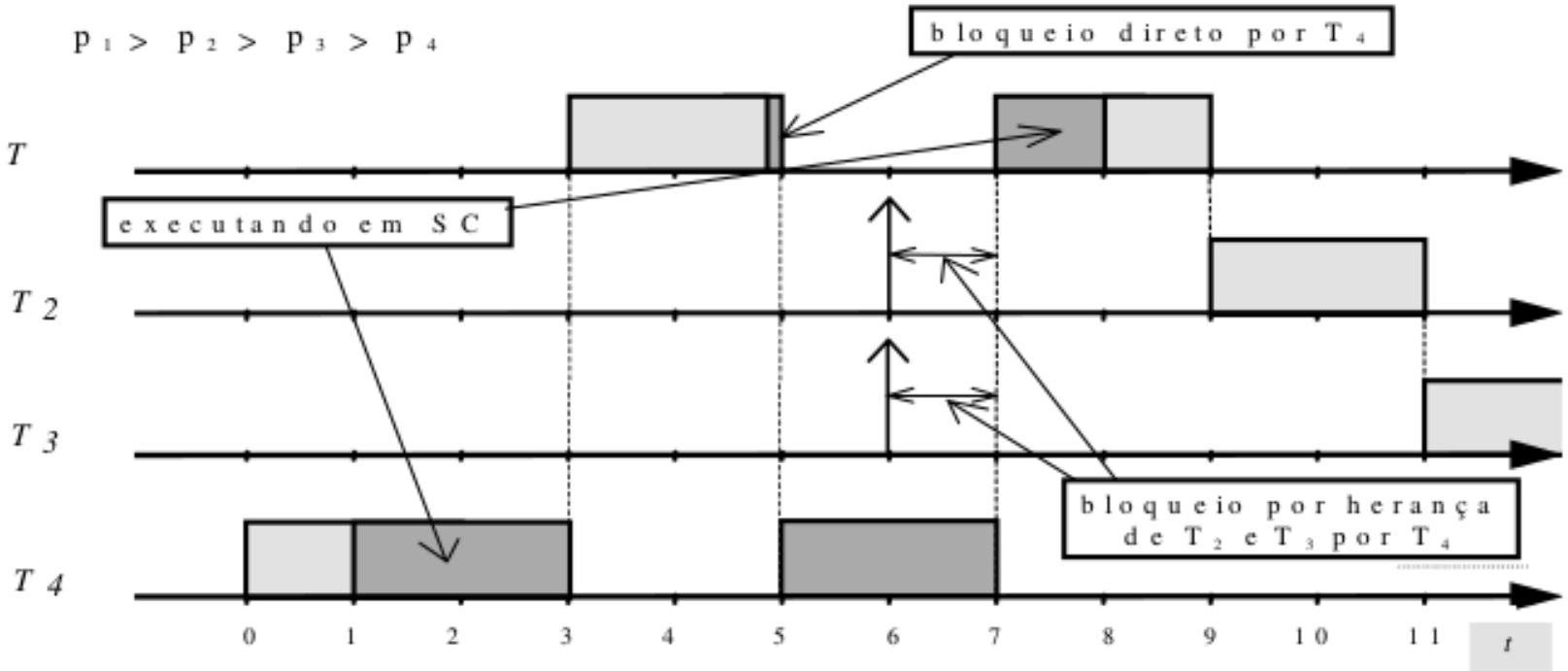

Figura 2.6: Aplicação do protocolo PHP [FARINES and MELO, 2000]

Protocolo PHP (fluxo)

- T_4 não sofre mais preempção de T_2 e T_3;
- Herda a prioridade de T_1 quando $t = 5$;
- Ao sair da região crítica ($t = 7$), T_4 volta ao nível de sua prioridade original.

Tipos de bloqueio [FARINES and MELO, 2000]

Bloqueio direto Ocorre quando a tarefa mais prioritária tenta acessar o recurso compartilhado já bloqueado pela tarefa menos prioritária.

Bloqueio por herança Ocorre quando uma tarefa de prioridade intermediária é impedida de continuar sua execução por uma tarefa que tenha herdado a prioridade de uma tarefa mais prioritária.

O Protocolo de Herança de Prioridade define um limite superior para o número de bloqueios que uma tarefa pode sofrer de outras menos prioritárias.

Protocolo de prioridade de teto – PCP

- Ideia: limitar o número de bloqueios ou inversões de prioridades para evitar a formação de cadeias de bloqueios;
- Cadeias de bloqueio podem levar à formação de deadlocks;
- Dirigido para escalonamentos de prioridade fixa, como o RM;
- Extensão do PHP adicionando uma regra de controle sobre os pedidos de entrada em exclusão mútua.

Descrição do PCP

- Incorpora o mecanismo de herança do PHP, ou seja, a tarefa T_i transfere sua prioridade à tarefa T_j no momento do bloqueio;
- Todos os recursos acessados em exclusão mútua possuem um valor de prioridade teto (*ceiling*) $C(S_k)$: prioridade da tarefa mais prioritária que acessa o recurso;
- Regra para entrada em região crítica [FARINES and MELO, 2000]:

  > *Uma tarefa só acessa um recurso compartilhado se sua prioridade ativa for maior que a prioridade teto (ceiling) de qualquer recurso já previamente bloqueado.*

- São excluídos da comparação recursos bloqueados pela tarefa requerente

Algoritmo do PHP

- Quando uma tarefa T_i é bloqueada sua prioridade é transferida a uma tarefa T_j;
- Quando T_j reassume executa com a prioridade herdada de T_i ($p_j = p_i$);
- Ao executar, a tarefa herdada sempre mantém a prioridade mais alta de bloqueio;
- Ao liberar a região crítica, T_j novamente passa a ter a prioridade ativa igual à prioridade nominal, ou;
- Assume a prioridade mas alta das tarefas que ainda estejam sob seu bloqueio.

Algoritmo do PCP

- Quando uma tarefa T_i é bloqueada sua prioridade é transferida a uma tarefa T_j;
- Quando T_j reassume executa com a prioridade herdada de T_i ($p_j = p_i$);
- Considere S_1 o semáforo de maior prioridade entre todos os bloqueados:
  - T_i só entrará na região crítica se sua prioridade dinâmica for maior que que o *ceiling* $C(S_1)$;
  - Se $p_i \leq S_1$ o acesso é negado a T_i;
- Ao executar, a tarefa herdada sempre mantém a prioridade mais alta de bloqueio;
- Ao liberar a região crítica, T_j novamente passa a ter a prioridade ativa igual à prioridade nominal, ou;
- Assume a prioridade mas alta das tarefas que ainda estejam sob seu bloqueio.

Exemplo do PCP

- Considere as tarefas T_1, T_2 e T_3;
- Acessa os respectivos recursos em exclusão mútua R_1, R_2 e R_3;
- A prioridade teto de cada semáforo é definida da seguinte forma, seguindo esquema de compartilhamento de recursos:
  - $C(S_1) = 1$;
  - $C(S_2) = 1$;
  - $C(S_3) = 2$;
- Novo tipo de bloqueio: ceiling. A tarefa não possui possui prioridade dinâmica superior à maior prioridade teto entre os recursos ocupados.

Exemplo do PCP (gráfico)

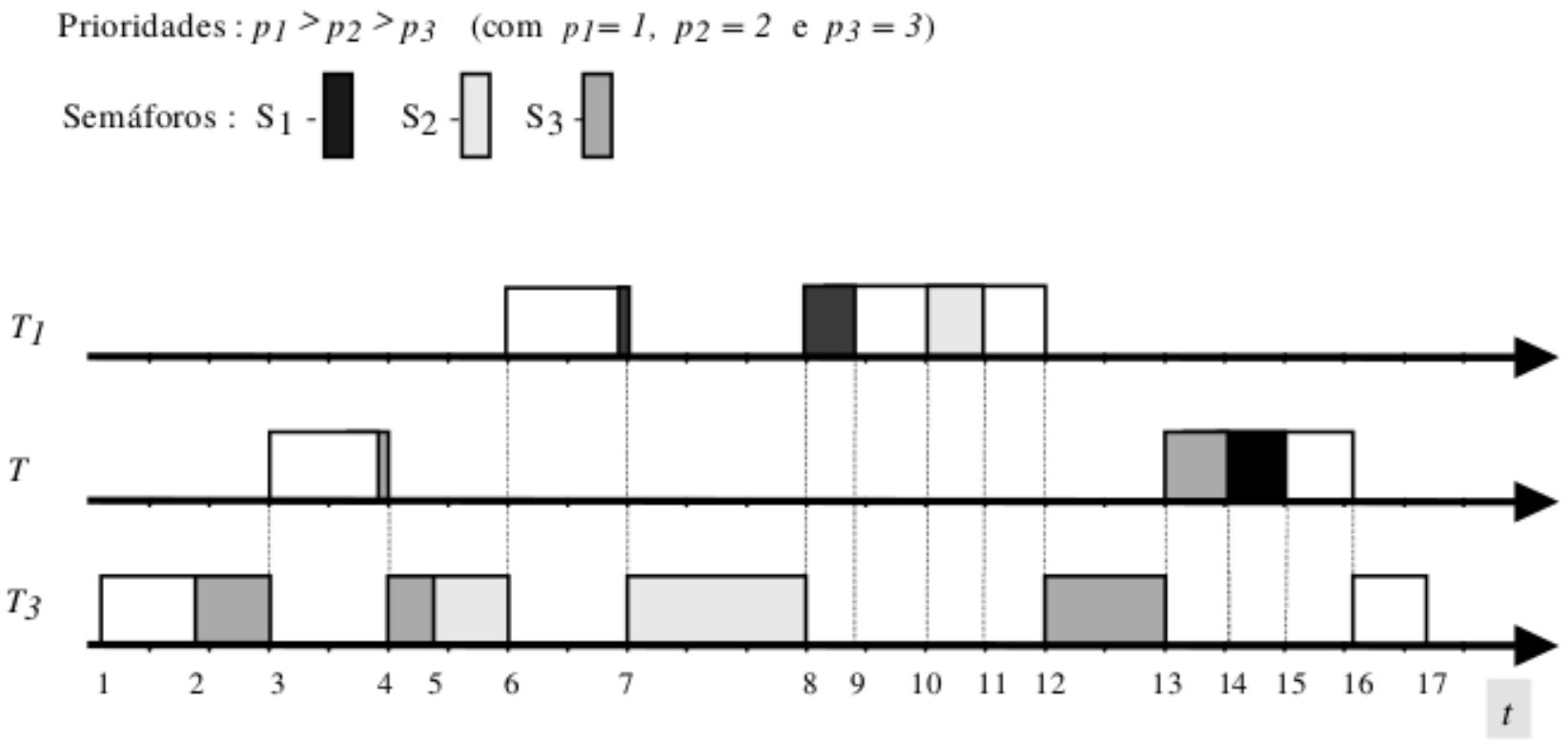

Figura 2.7: Aplicação do protocolo PCP [FARINES and MELO, 2000]

Sequência de eventos [Tanenbaum and Machado Filho, 1995]

1. T_3 começa sua execução;
2. T_3 entra na região crítica (fecha o Semáforo S_3);
3. T_2 inicia seu processamento interrompendo T_3 (preempção de T_3);
4. T_2 sofre bloqueio direto: S_3 "fechado" por T_3 que reassume e herda prioridade de T_2;
5. T_3 entra na região crítica aninhada ao fechar semáforo S_2;
6. T_1 inicia e interrompe T_3. T_1 é mais prioritária que T_3 com a prioridade herdada de T_2;
7. T_1 tenta fechar semáforo S_1, é bloqueado por *ceiling*; possui prioridade igual ao maior *ceiling* de semáforo já fechado pelas outras tarefas ($C(S_2) = 1$);

Sequência de eventos (cont.) [Tanenbaum and Machado Filho, 1995]

8. T_3 libera semáforo S_2. T_1 é reativado e interrompe T_3. T_1 entra em S_1;
9. T_1 libera S_1;
10. T_1 fecha Semáforo S_2;
11. T_1 libera S_2;
12. T_1 termina e T_3 reassume na prioridade herdada de T_2;
13. T_3 libera S_3 retornando a sua prioridade estática. T_2 interrompe T_3 e fecha S_3;
14. T_2 libera S_3;
15. T_2 fecha e libera S_1;
16. T_2 completa e T_3 reassume;
17. T_3 completa.

Sistemas operacionais de tempo real – STOR

- Necessidade de controle do tempo;
- Limitações dos sistemas operacionais de propósito geral:
  - Memória virtual;
  - Mecanismos de acesso ao sistema de arquivos;
  - Algoritmo de escalonamento, etc.
- Tipos de suporte de tempo real:

  NTR Núcleo de tempo real: kernel reduzido para aumentar a previsibilidade temporal.

  STOR Sistema operacional de tempo real: reescrita completa ou otimização de mecanismos de controle do tempo. Ex.: trocar algoritmo de escalonamento.


Chagas, F. (2016).
Notas de aula do Prof. Fernando Chagas.


FARINES, J. M. and MELO, R. (2000).
Sistemas de Tempo Real, volume 1.
IME-USP.


Favacho, A. (2009).
Notas de aula da Profa. Aletéia Favacho.


Tanenbaum, A. S. and Machado Filho, N. (1995).
Sistemas operacionais modernos, volume 3.
Prentice-Hall.

OBRIGADO!!!
PERGUNTAS???