Danton Cavalcanti Franco Junior
falecom@dantonjr.com.br

Comunicação entre Processos

- **Soluções de Hardware**
  - Criam mecanismos para as soluções de software.
  - **Inibição das Interrupções**: Entra na região crítica, inibe as interrupções, habilita quando sai.
  - O processo pode não voltar a habilitar as interrupções – comprometimento do sistema.
  - Útil para compartilhar listas e estruturas de dados.

Comunicação entre Processos

- **Soluções de Hardware**
  - **Instrução Test-and-set:** Lê uma variável, armazena seu conteúdo em outra área e atribui o novo valor.
  - É uma instrução indivisível.
  - Utiliza uma variável lógica.
  - Testa a variável lógica, se falsa, faz a transferência.
  - Sintaxe: Test-and-set(Pode, Bloqueio);

Comunicação entre Processos

- **Soluções de Software**
  - Primeiras soluções não eram tão boas: alto tempo de espera, espera ocupada (starvation) e baixo número de processos.
  - **Semáforos:** Variável inteira e não negativa que só pode ser manipulada por 2 instruções: Down e Up (P-Sleep e V-WakeUp).
  - Quando aplicados ao problema da exclusão mútua, são chamados de Mutex.

Comunicação entre Processos

- **Soluções de Software**
  - **Semáforos:**
    - Inicia com 1
    - Processo Down – 0
    - É 0? Down, entra na fila de espera
    - Processo termina UP
    - Fila anda

Comunicação entre Processos

- **Soluções de Software**
  - **Semáforos:**
  - **P(S):** **Se S = 0 então**

    Bloqueia processo

    Senão

    S := S – 1
  - **V(S):** **S := S + 1**

Comunicação entre Processos

- **Soluções de Software**
  - **Semáforos (tipo vazio e cheio):**
    - Também chamados de contadores.
    - Usados em pool.
    - Determinam a quantidade de recursos disponíveis, sempre decrementando 1.

Comunicação entre Processos

- **Soluções de Software**
  - **Monitores:** Os processos chamam os procedimentos do monitor mas não acessam as estruturas de dados e variáveis internas, ou seja, os procedimentos e a estrutura de dados é definida dentro do módulo.
  - As implementações são realizadas pelo compilador, não pelo programador o que garante mais segurança.

Comunicação entre Processos

- **Soluções de Software**
  - **Monitores:** Os procedimentos dentro do monitor tem sua garantia definida.
  - Implementação da sincronização não é tão simples: Wait e Signal, operando sobre a estrutura de dados (fila de espera por evento).

Comunicação entre Processos

- **Soluções de Software**
  - **Troca de Mensagens:** através das primitivas Send e Receive.
  - Problema de perda de mensagem (receptor envia mensagem de aceite – ACK).
  - Transmissão direta, informando o nome do processo, o buffer é chamado de mailbox.
  - Síncrono: transmissor espera receptor.
  - Assíncrono: nem transmissor nem receptor aguardam a mensagem de confirmação.

Comunicação entre Processos

- **Deadlock:**
  - É a espera de um evento que nunca ocorrerá.
  - Condições para ocorrer o deadlock:
  1. Cada recurso só pode estar alocado a um único processo (exclusão mútua);
  2. Um processo espera por outros recursos, além dos já alocados;
  3. Um recurso não pode ser liberado só porque outros querem.
  4. Um processo espera pelo outro (espera circular).

Comunicação entre Processos

- **Prevenção:**
  - Garantir que uma das quatro condições não ocorra.
  - 1) Retirar a exclusão mútua: problemas de sincronismo.
  - 2) Requisitar os recursos antes de utilizá-los. Grande tempo de uso dos recursos. Possibilidade de ocorrer starvation (espera de recursos). Dificuldade em determinar os recursos antes de utilizar.

Comunicação entre Processos

- **Prevenção:**
  - 3) Retirar o recurso de um processo, pode acarretar em perda de todo o processamento já realizado. Também pode ocorrer starvation.
  - 4) Evitar a referência circular. Forçar o uso de apenas um recurso por vez, se precisar de outro, deve liberar o primeiro.

Comunicação entre Processos

- **Problema dos filósofos (referência circular):**
  - A implementação proposta apresenta cinco (5) filósofos e cinco (5) garfos. A idéia central deste problema é que dado os FILÓSOFOS e os GARFOS em um JANTAR, cada filósofo necessita de DOIS (2) garfos para comer. Os filósofos podem estar em um destes três (3) estados: PENSANDO, FAMINTO ou COMENDO. Se um filósofo está no estado PENSANDO e quer passar para o estado COMENDO, ele tenta pegar dois (2) garfos. Se ele não consegue pegar os dois (2) garfos, passa o estado FAMINTO. Se consegue pegar os dois garfos ele passa para o estado COMENDO. Enquanto no estado FAMINTO, o filósofo permanece tentando pegar os garfos, ou seja, fica esperando a liberação dos garfos.

Comunicação entre Processos

- **Algoritmo do Banqueiro**
- Usado para determinar se um processo pode executar de maneira segura ou não:
  - Todos os processos declaram o máximo de recursos que vão usar durante a execução. A execução é permitida se a soma dos recursos requisitados é menor que os recursos disponíveis no sistema.
  - Recursos requisitados = Recursos declarados por todos os processos já executando + Recursos declarados pelo novo processo.

Comunicação entre Processos

- **Algoritmo do Banqueiro**
- Problema:
  - Número fixo de processos e recursos. É difícil prever esse número.
  - Conservativo: pode recusar processos que não causem deadlock.

Comunicação entre Processos

- **Detecção do deadlock**
  - Implementação de estruturas de dados que identifiquem os recursos do sistema, processos alocando e processos liberando recursos (atualização das listas).
  - Percorrer toda a estrutura, em sistemas tempo real, pode comprometer o desempenho do sistema.

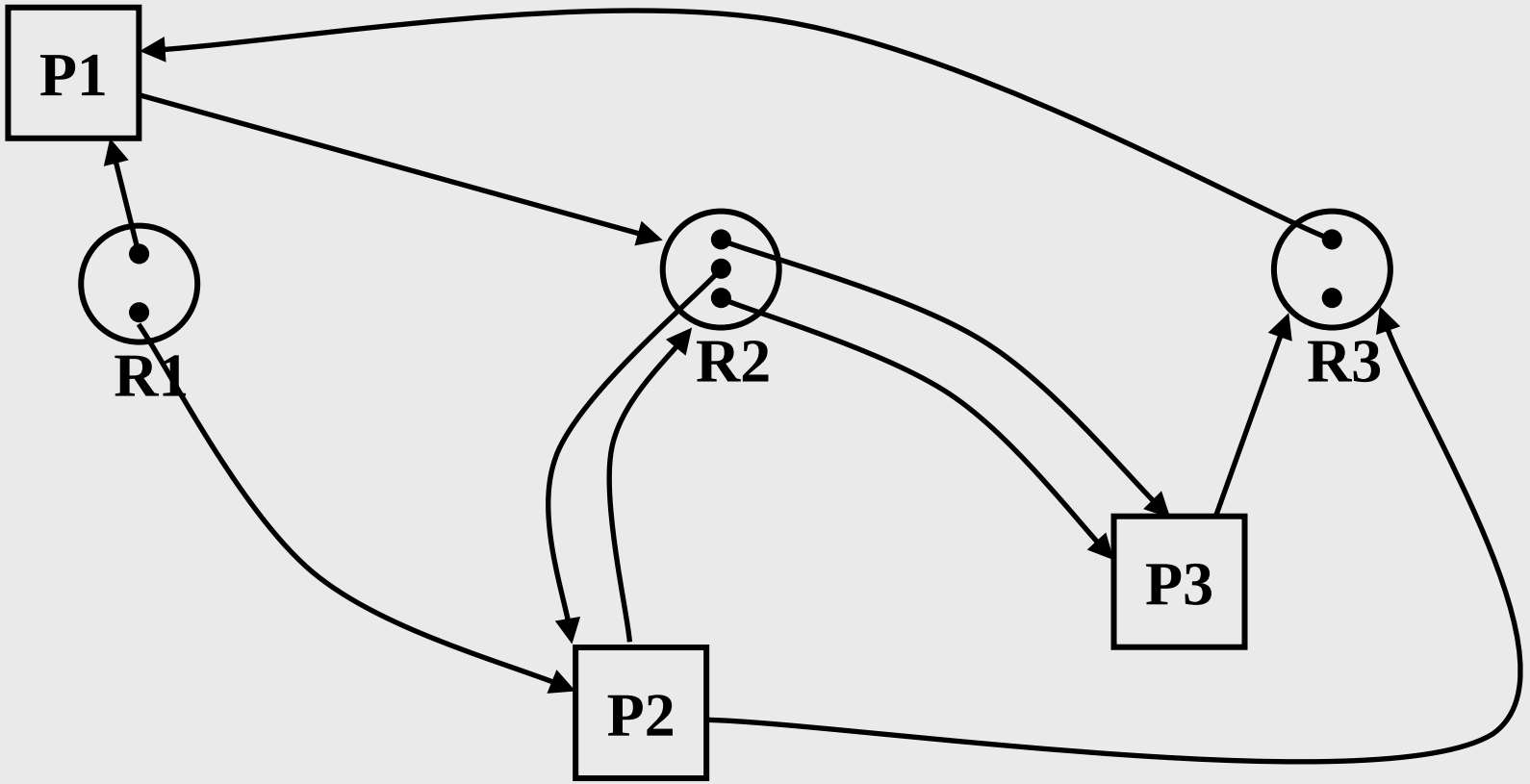
Comunicação entre Processos

- **Correção do deadlock**
  - Eliminação de um processo: problema quando elimina processos críticos (atualização de arquivo ou impressão).
  - Liberação de apenas alguns recursos: o sistema deve liberar os recursos sem comprometer o processamento já realizado. Gera grande overhead.

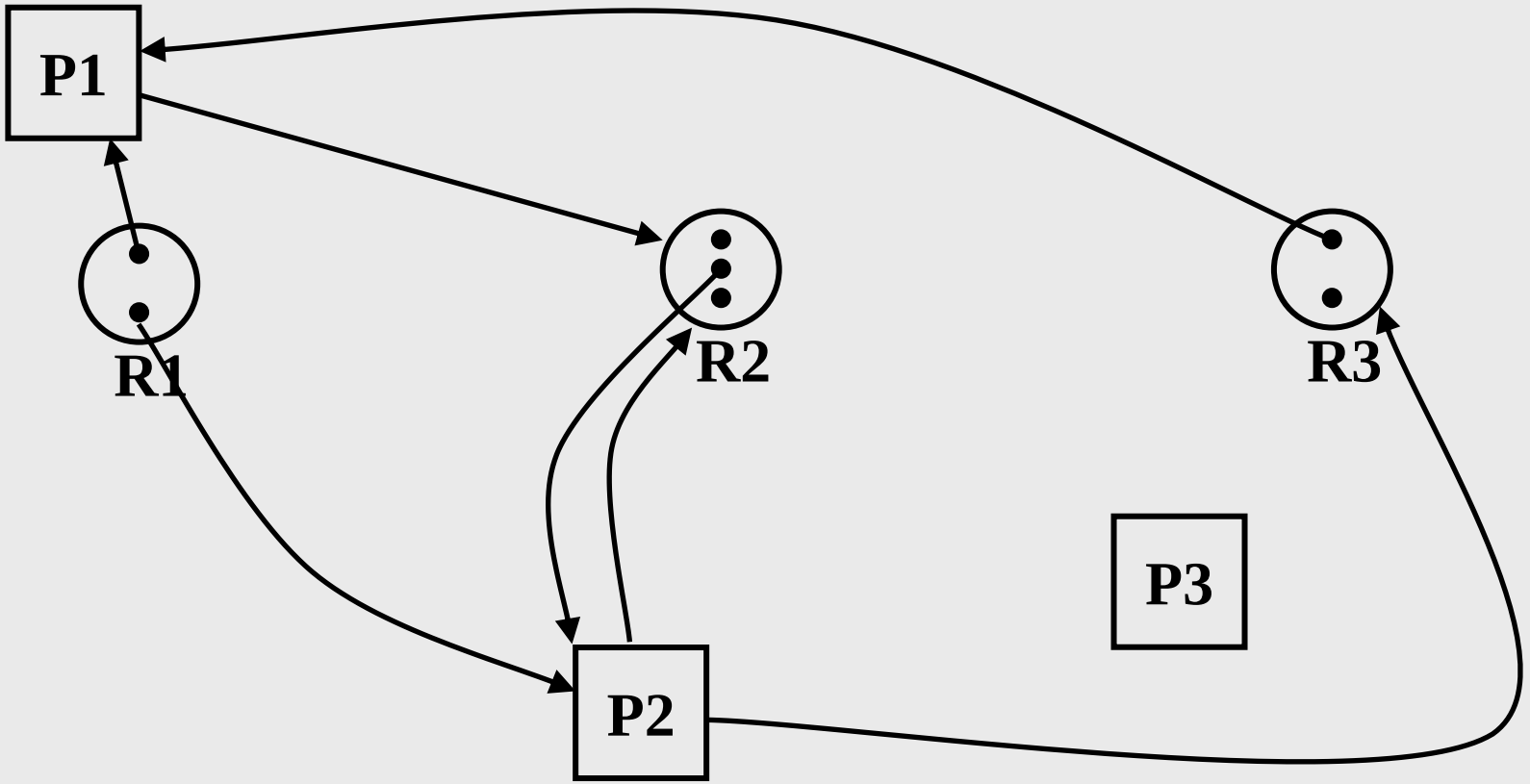
Comunicação entre Processos

- **Estratégia para Correção do deadlock**
  - Redução de grafo;
  - Seta que chega: pedido de recurso
  - Seta que sai: recurso alocado

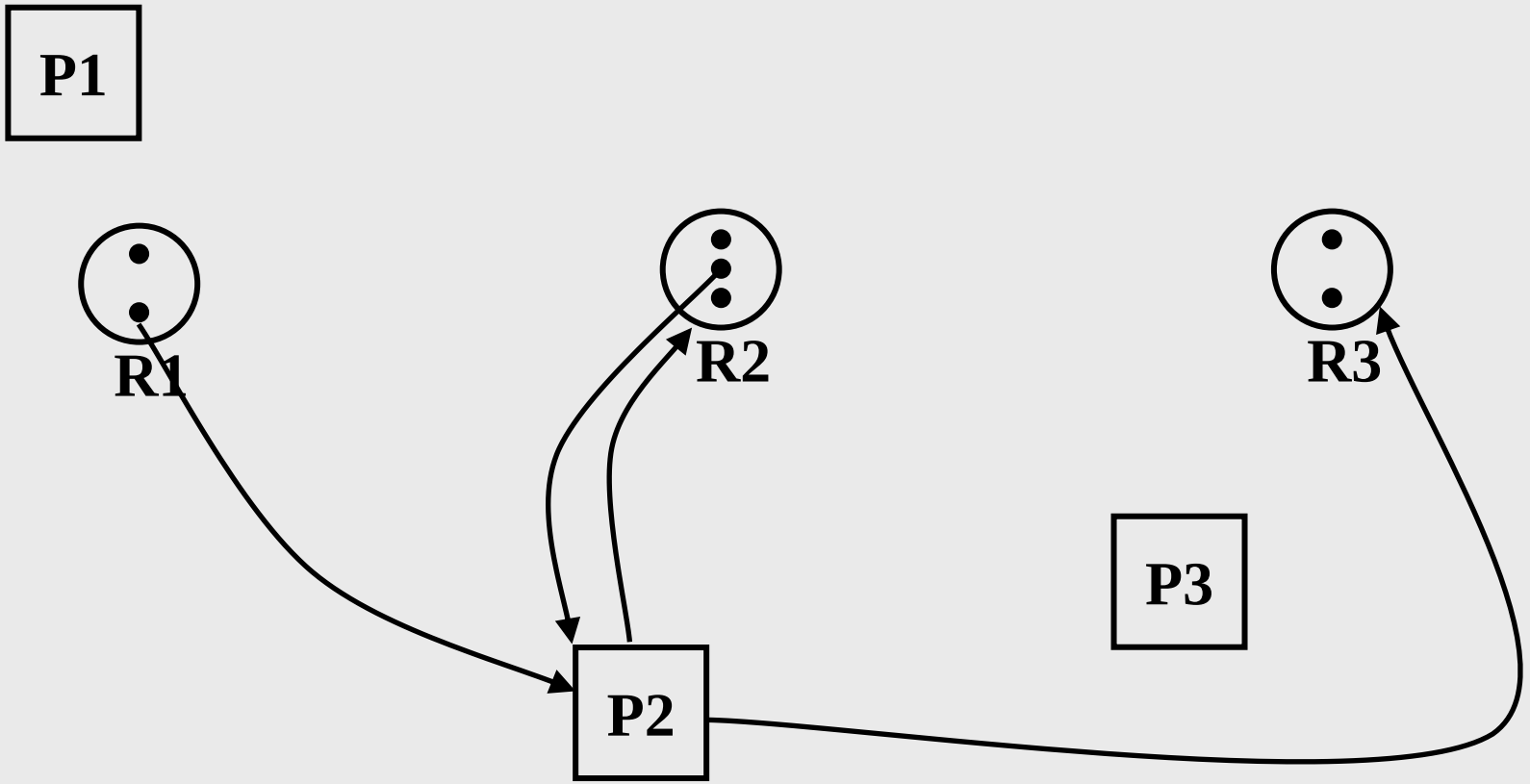
Comunicação entre Processos



Comunicação entre Processos

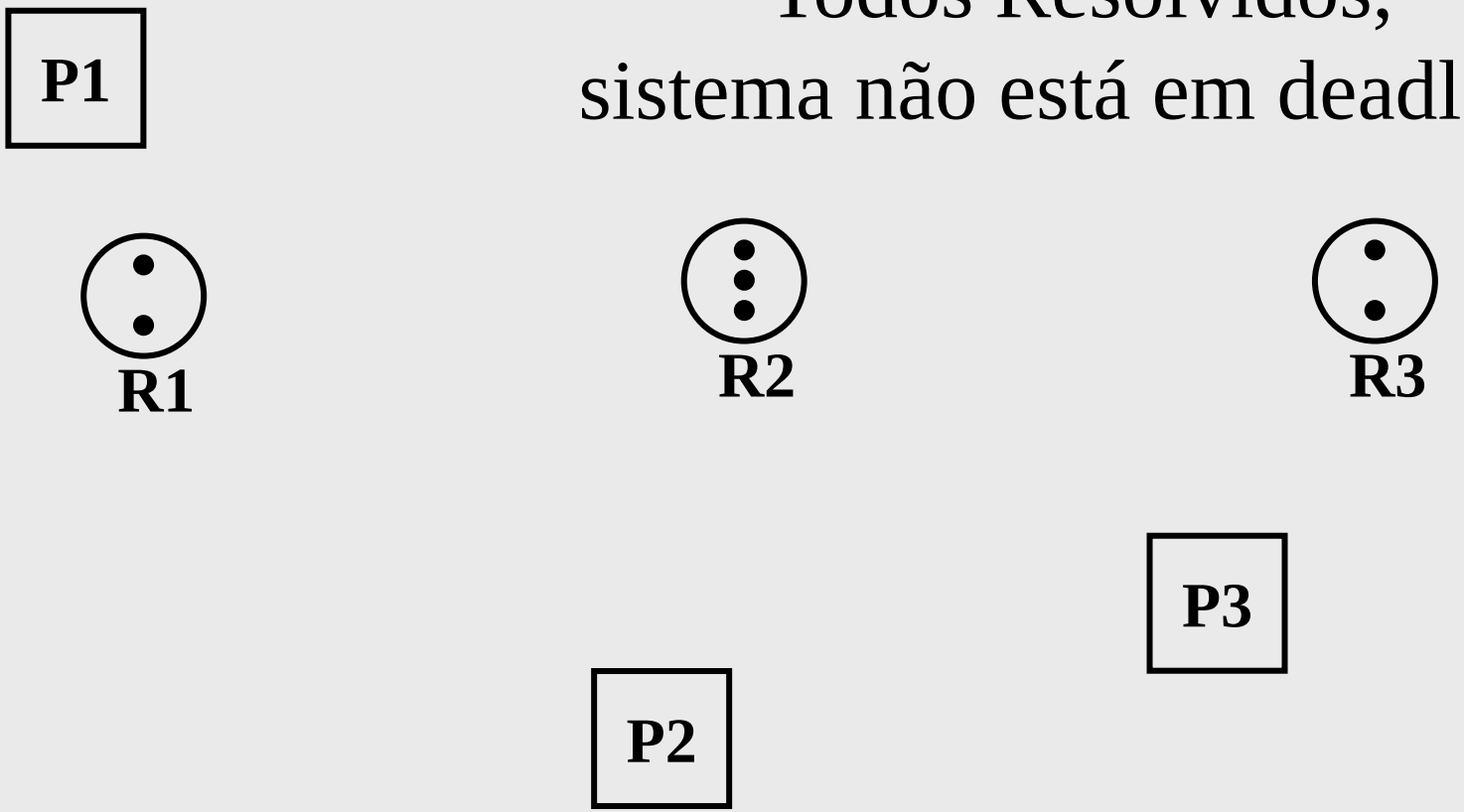


Comunicação entre Processos

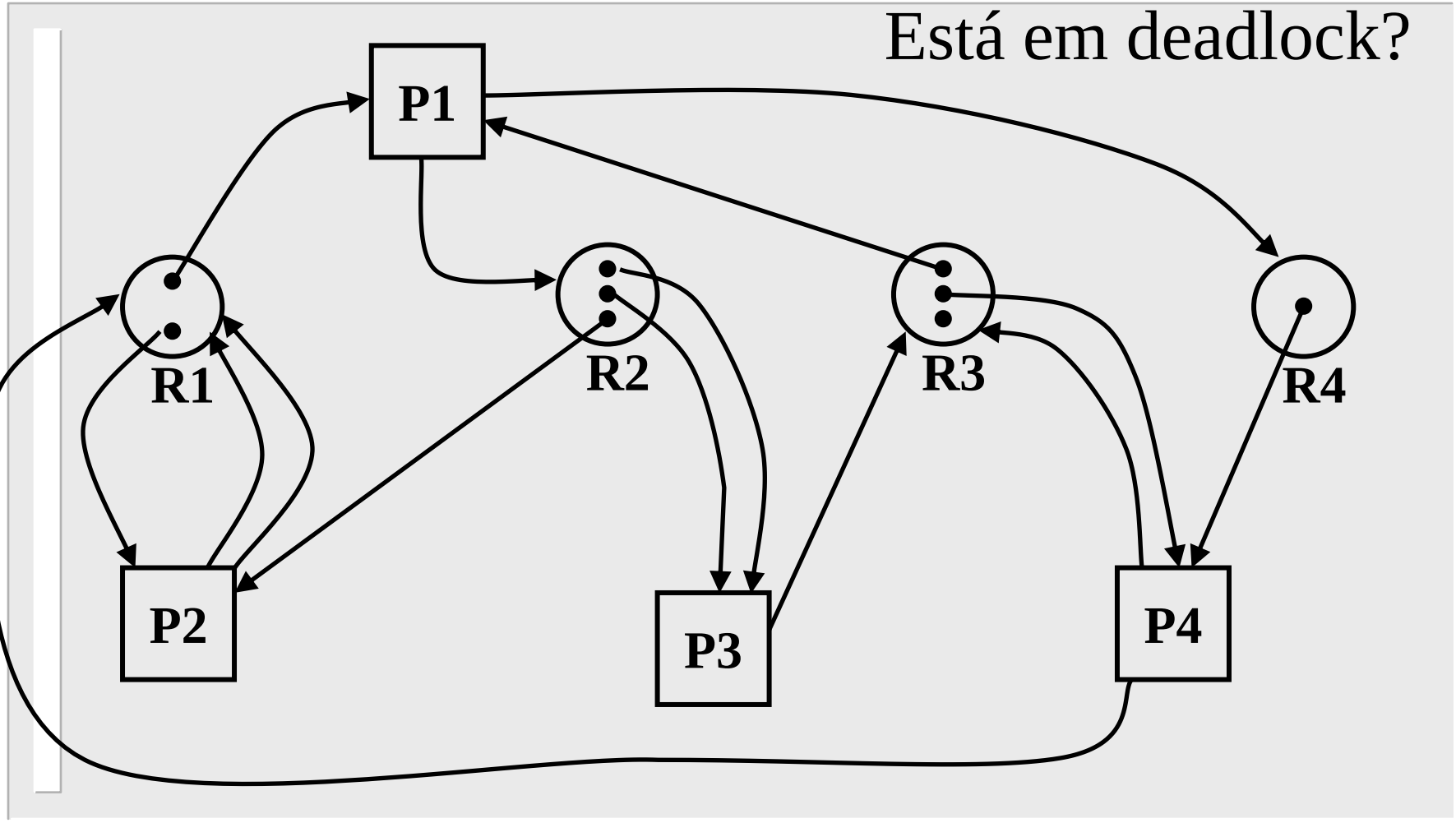


Comunicação entre Processos

Todos Resolvidos,
sistema não está em deadlock

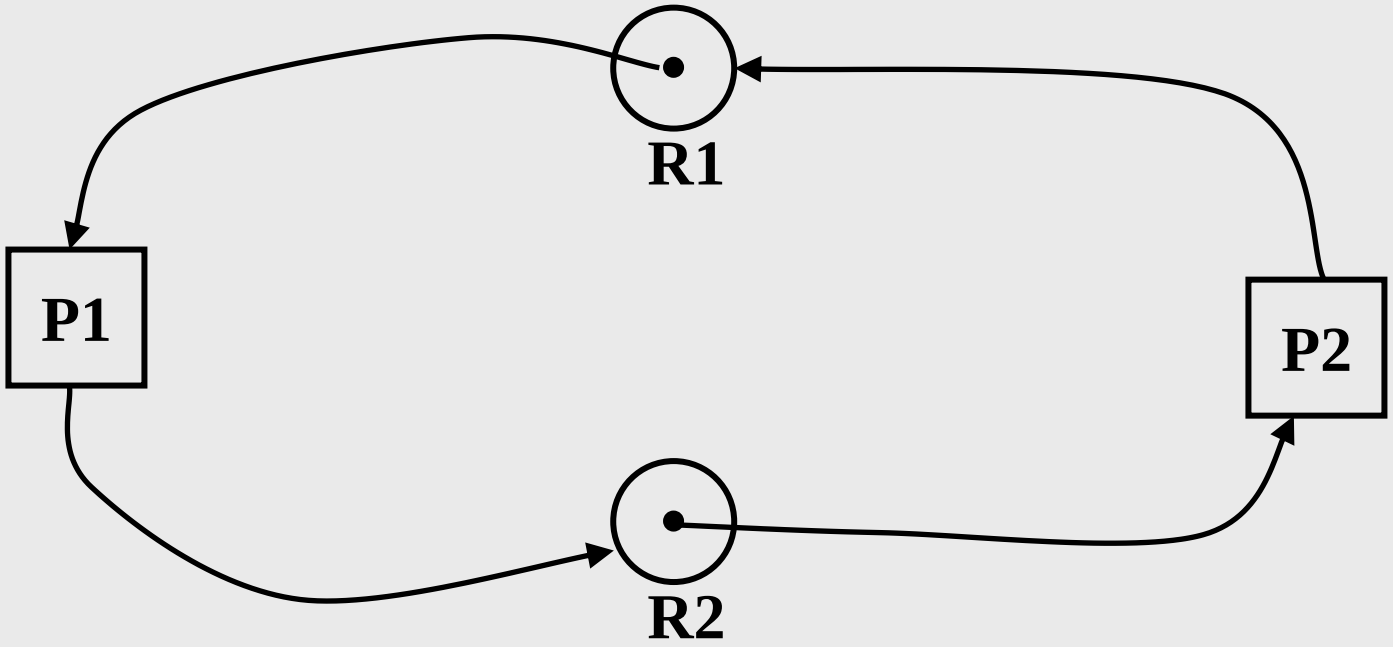


Comunicação entre Processos



Comunicação entre Processos

Está em deadlock?



Comunicação entre Processos

- **Estratégia para Prevenção do deadlock (banqueiro):**
  - Supõe o atendimento da requisição.
  - Supõe que todos os processos não bloqueados (inclusive o pedinte) requisitem o máximo de recursos que ainda possam.
  - Aplica o algoritmo de detecção sobre essa situação (a pior possível).
  - Se não há deadlock, entrega o recurso, senão bloqueia o processo mesmo que o recurso esteja disponível.

Comunicação entre Processos

- **1 recurso (10 unidades) 3 processos (safe):**

Saldo 3

Pr	T	M
A	3	9
B	2	4
C	2	7

Saldo 1

Pr	T	M
A	3	9
B	4	4
C	2	7

Saldo 5

Pr	T	M
A	3	9
B	0	-
C	2	7

Saldo 0

Pr	T	M
A	3	9
B	0	-
C	7	7

Saldo 7

Pr	T	M
A	3	9
B	0	-
C	0	-

P-processo
T-tem
M-máximo

Comunicação entre Processos

- **1 recurso (10 unidades) 3 processos (not safe):**

Saldo 2

Pr	T	M
A	4	9
B	2	4
C	2	7

Saldo 0

Pr	T	M
A	4	9
B	4	4
C	2	7

Saldo 4

Pr	T	M
A	4	9
B	0	-
C	2	7

Comunicação entre Processos

- **1 recurso (10 unidades) 5 processos:**

Saldo 10

A	0	6
B	0	5
C	0	4
D	0	7
E	0	0

Saldo 2

A	1	6
B	1	5
C	2	4
D	4	7
E	0	0

Saldo 1

A	1	6
B	2	5
C	2	4
D	4	7
E	0	0

Quem está safe/not safe?

Comunicação entre Processos

- 1) Criar dois cenários em deadlock e dois cenários em modo safe para o algoritmo da redução por grafo utilizando:
  - No mínimo quatro (4) recursos variando de 1 a 4 unidades;
  - No mínimo 4 processos.
- 2) Criar dois cenários em deadlock e dois cenários em modo safe para o algoritmo do banqueiro utilizando:
  - Um recurso com no mínimo cinco (5) unidades e máximo de quinze (15) unidades;
  - Mínimo de três (3) processos e máximo de oito (8).