Comunicação entre Processos

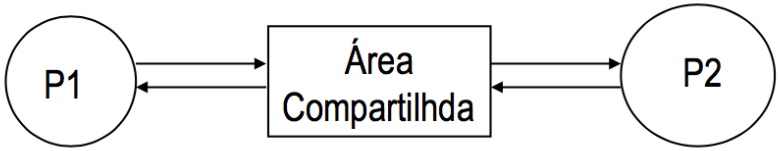
Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

Conteúdo

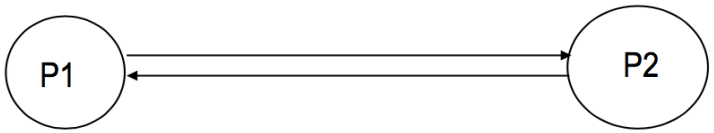
- Memória compartilhada vs Troca de mensagens
- Troca de mensagens:
  - Sincronização
  - Endereçamento
  - Identificação de processos
  - Gerenciamento de buffer (bufferização)
  - Codificação/decodicação de dados
  - Tratamento de falhas

Comunicação entre processos

- Memória Compartilhada:
  - os processo compartilham variáveis e trocam informações através do uso dessas

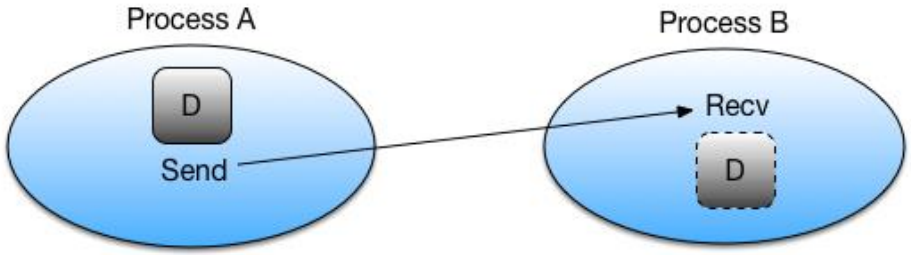


- Sem Memória Compartilhada:
  - os processos compartilham informações através de troca de mensagens
  - o S.O. é responsável pelo mecanismo de comunicação entre os processos



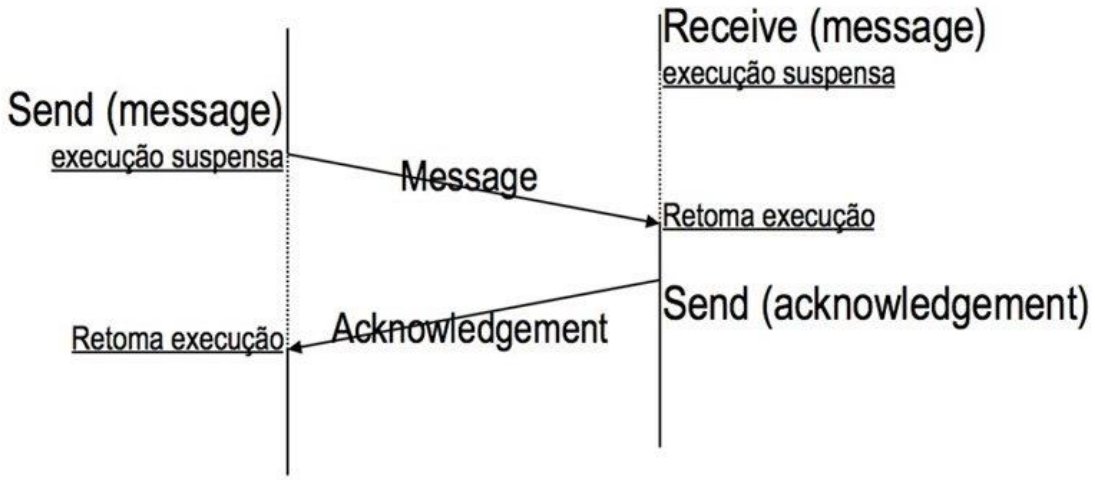
Troca de mensagens

- Método de comunicação em que entidades (threads, processos) trocam informações por meio de mensagens, sem compartilhar memória
- Normalmente utiliza rotinas do tipo send()/recv() ou similar
- Exemplos:
  - Sockets, filas de mensagens do sistema operacional (ex: POSIX message queue), bibliotecas de troca de mensagens (ex: MPI), canais em linguagem de programação (ex: Go e Rust), etc.



Troca de mensagens: sincronização

- Sincronização em operações *send/receive*
  - quando *send* e/ou *receive* são bloqueantes a comunicação é dita **síncrona**
  - senão é dita **assíncrona**



Troca de mensagens: endereçamento

- **Endereçamento explícito:** processo com o qual se quer comunicar é explicitamente identificado através de um parâmetro
- **Endereçamento implícito:** processo com o qual se quer comunicar não é identificado
  - Ex.: *sender* mandando para qualquer *receiver* que desempenhe uma função/serviço
    - *sender* nomeia serviço ao invés de processo
  - Ex: *receiver* quer receber independente do *sender*: modelo cliente/servidor
    - servidor quer poder servir qualquer cliente

Troca de mensagens: endereçamento

- Endereçamento explícito:
  - o *sender* identifica o *receiver*
  - o *receiver* identifica o *sender*

```
/ * task 0 * /
main ( ) {
     -
     -
     -
  send ( t1, m );
}
```

```
/ * task 1 * /
main ( ) {
     -
     -
     -
  receive ( t0, m );
}
```

Troca de mensagens: endereçamento

- Endereçamento implícito do lado *receiver*:
  - *sender* identifica o *receiver*
  - *receiver* recebe de qualquer *sender*

```
/ * task 0 * /
main ( ) {
     -
     -
     -
   send ( t2, m );
 }
```

```
/ * task 1 * /
main ( ) {
     -
     -
     -
   send ( t2, m );
 }
```

```
/ * task 2 * /
main ( ) {
     -
     -
     -
   receive ( -1, m );
 }
```

Troca de mensagens: identificação de processos

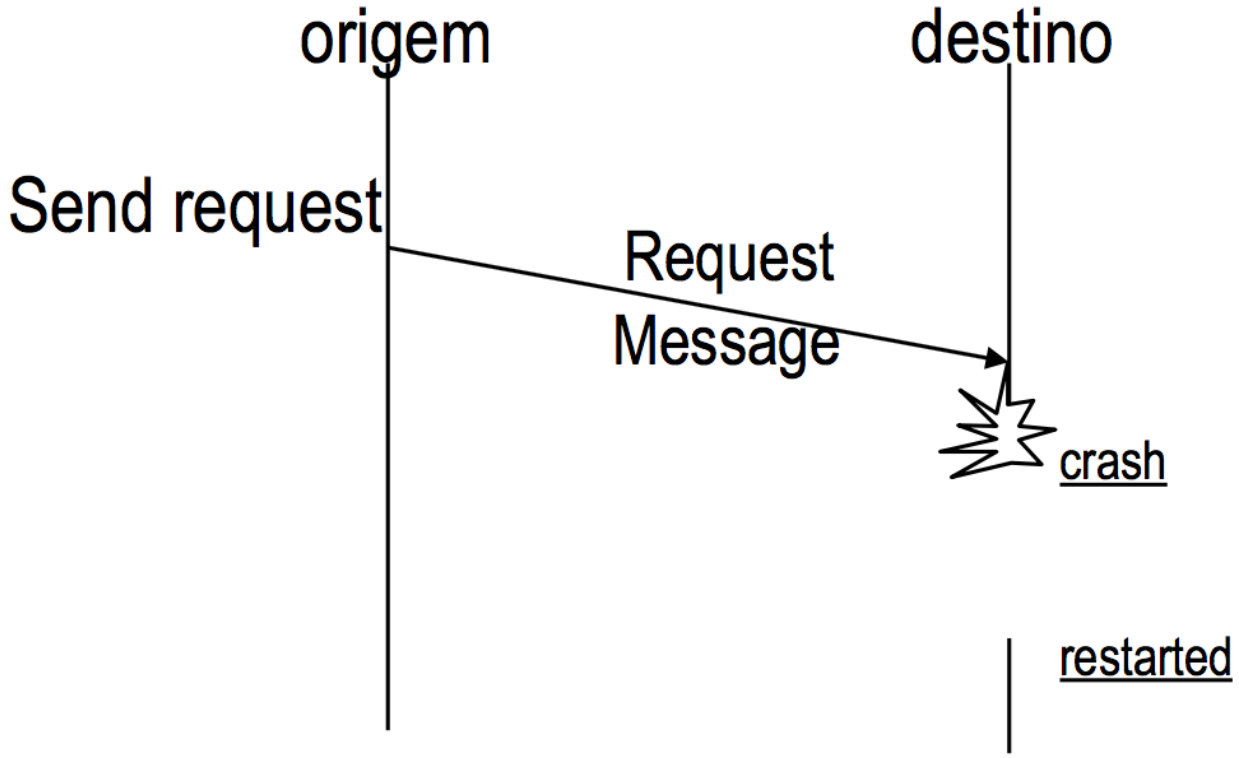
- Problema: como identificar unicamente um processo?
  - Métodos não transparentes: especifica identificador de máquina
    - Exemplo: ip:porta, process_id@machine_id
  - Métodos transparentes: identificador único do processo não deve ter embutida informação de localização do processo
    - Necessita de "tradução" de nomes
    - Exemplo: MPI utiliza um ID numérico para identificar os processos

Troca de Mensagens: Codificação/decodicação

- Problemas de "apresentação" de dados:
  - mensagens entre processos rodando em diferentes arquiteturas
  - transferência de valores de ponteiros para memória perdem significado
  - identificação necessária para dizer o tipo de dado sendo transmitido
- Solução: uso de formato comum de transferência (sintaxe de transferência)
  - representação com tags (tagged): mensagem carrega tipo dos dados
    - Ex.: ASN.1 (abstract syntax notation - CCITT)
  - representação sem tags: receptor tem que saber decodificar mensagem
    - Ex.: XDR (eXternal Data Representation - Sun)

Troca de Mensagens: Tratamento de Falhas

- Problema: execução de um pedido/request no destinatário não tem sucesso

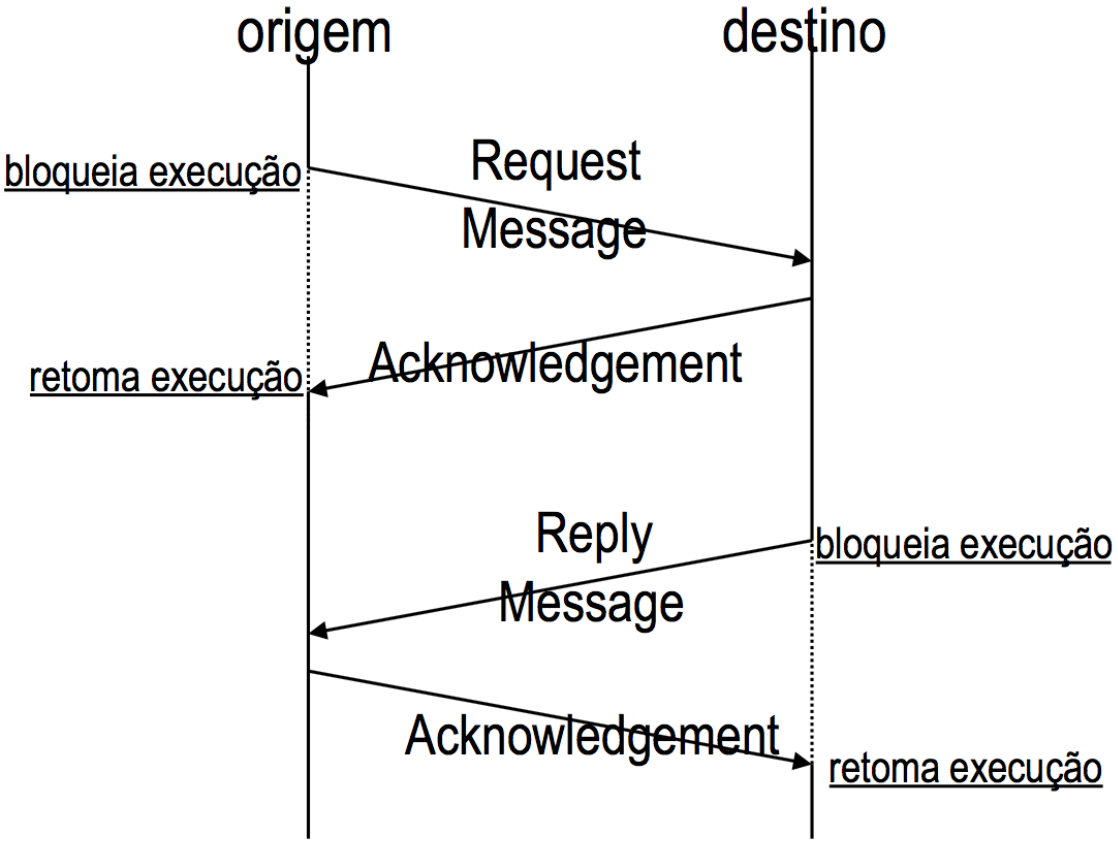


Troca de Mensagens: Tratamento de Falhas

- Estratégias
  - four message: confirmação das mensagens de pedido e resposta
  - three message: confirmação da resposta
  - two message: resposta é confirmação

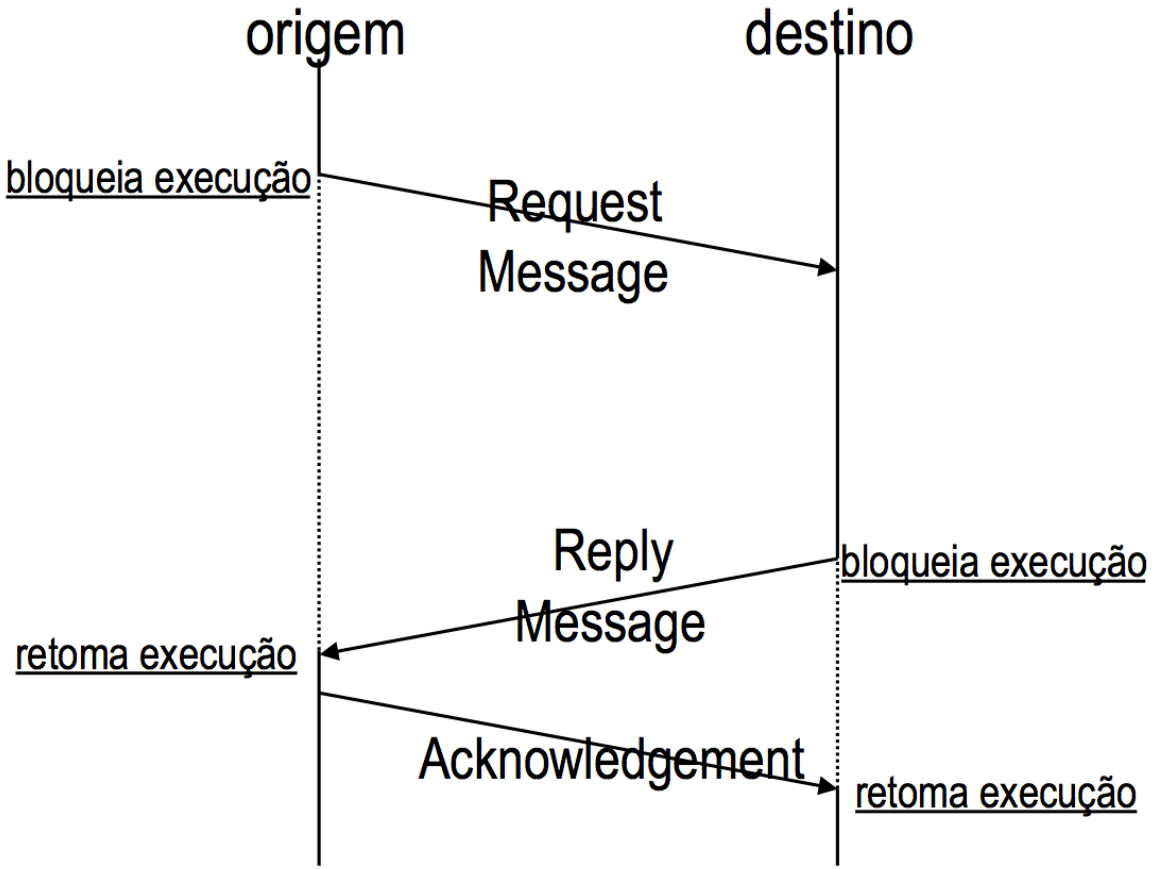
Troca de Mensagens: Tratamento de Falhas

- Estratégia four message: confirmação das mensagens de pedido e resposta



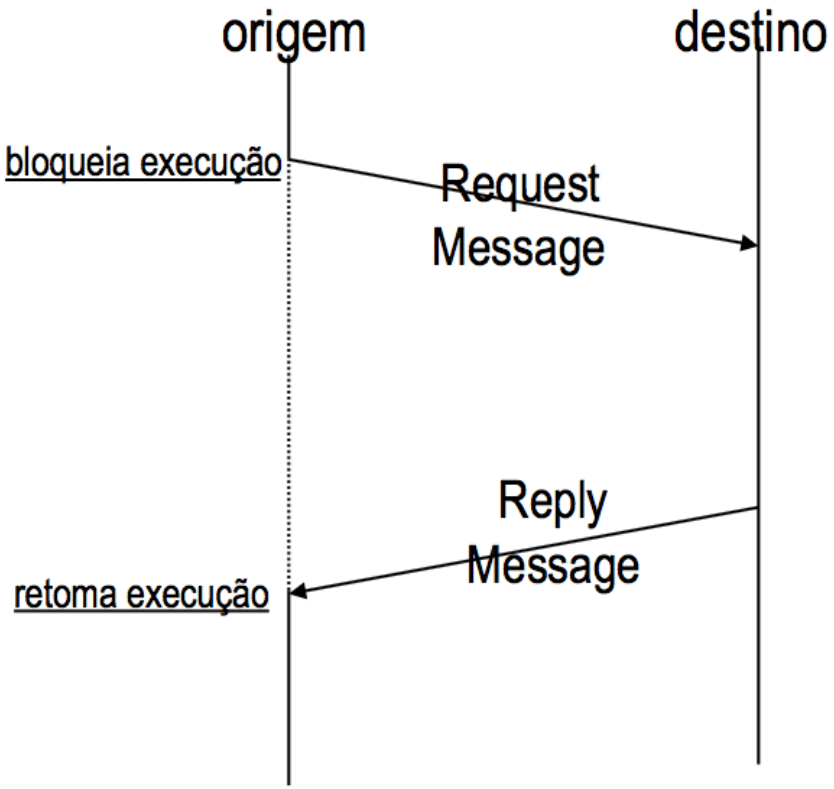
Troca de Mensagens: Tratamento de Falhas

- Estratégia three message: confirmação da resposta



Troca de Mensagens: Tratamento de Falhas

- Estratégia two message: resposta é confirmação



Troca de Mensagens: Tratamento de Falhas

- Idempotência e tratamento de requests duplicados
  - idempotência: repetibilidade
    - operação idempotente pode ser repetida inúmeras vezes sem causar efeitos colaterais ao servidor — ex.: get_time(), sqrt(x)
  - operações não idempotentes necessitam semântica "*exactely-once*"
    - Garante que somente uma execução no servidor é realizada
    - ex.: conta.deposita(valor); conta.retira(valor)
    - Usar identificador único para cada request
    - lado servidor pode guardar *reply cache* para responder mesma resposta a pedido repetido

Troca de Mensagens

- Aspectos importantes para um sistema de troca de mensagens
  - Simplicidade
  - Semântica uniforme
  - Eficiência
  - Confiabilidade
  - Segurança
  - Portabilidade

Troca de Mensagens

- Simplicidade: construção de novas aplicações para interoperar com já existentes deve ser facilitada
- Semântica uniforme: comunicação local (processos no mesmo nodo) e comunicação remota (processos em nodos diferentes) através de funções tão próximas quanto possível
- Eficiência: reduzir número de mensagens trocadas tanto quanto possível
  - economizar fechamento e abertura de conexões;
  - reduzir cópias desnecessárias;

Troca de Mensagens

- Confiabilidade: garantir entrega da mensagem - confirmação, eliminação de duplicatas, ordenação
- Flexibilidade: possibilidade de utilizar somente funcionalidade requerida (em prol de desempenho)
  - necessidade ou não de entrega garantida, ordenada, atomica, etc
- Segurança: suporte a autenticação, privacidade
- Portabilidade: possibilidade de operar em plataformas heterogêneas

Referências

- Material baseado em slides dos Profs. Roland Teodorowitsch, Avelino Zorzo, Celso Costa, Fernando Dotti, Roland Teodorowitsch e Luiz Gustavo Fernandes
- E no livro:
  - Distributed Systems: Principles and Paradigms, Andrew S. Tanenbau