Sistemas Distribuídos

Comunicação e sincronização entre processos

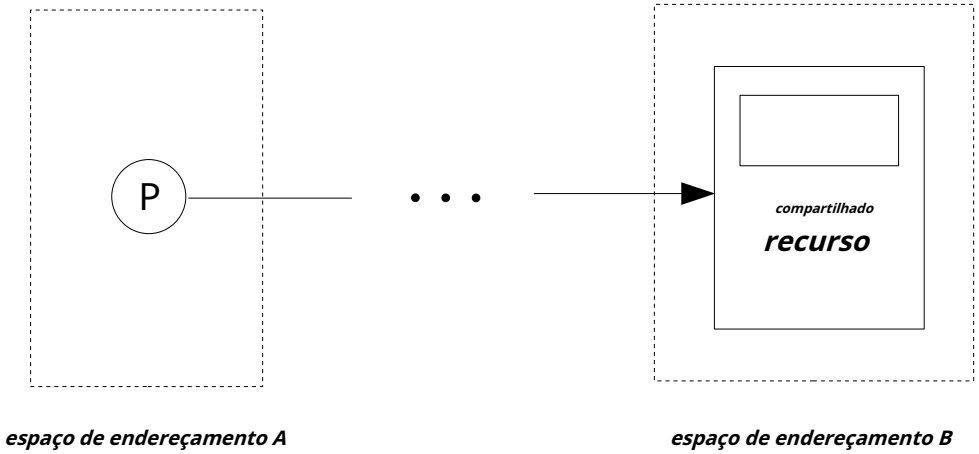
Objetos Remotos - 1

António Rui Borges

Resumo

- *O que é uma chamada de procedimento remoto*
- *Arquitetura de um ambiente de chamada de procedimento remoto*
- *Migração de código*
- *Leitura sugerida*

O que é uma chamada de procedimento remoto – 1



Em um *chamada de procedimento remoto*, o processo que invoca um método em um recurso compartilhado e o próprio recurso compartilhado não compartilham o mesmo espaço de endereçamento. Isso significa que

- no endereçamento *B* ao qual o recurso compartilhado pertence, uma referência a ele deve ser gerada e disponibilizada para outros
- no endereçamento *A* onde reside o processo de chamada, uma referência ao recurso compartilhado deve ser obtida antes que um método possa ser invocado nele.

O que é uma chamada de procedimento remoto - 2

A *controlo remoto* chamada de procedimento não se comporta exatamente como uma *local* chamada de procedimento. Existem três características distintivas das quais devemos estar cientes

- *a chamada pode falhar, mesmo que o código não contenha erros*: isso se deve ao fato de que o controle remoto recurso, b para uma diferença espaço de endereçamento diferente, não é atualmente instanciado, ou a infraestrutura de comunicação que conecta os dois espaços de endereçamento não está operando corretamente
- *todos os parâmetros do procedimento e o valor de retorno, se existir, devem ser passados por valor*: como o processo de invocação e o recurso partilhado residem em espaços de endereçamento diferentes, o único meio de comunicação disponível é através da passagem dos próprios dados relevantes, juntamente com o seu formato e estrutura; portanto, o empacotamento da informação deve ocorrer na origem e o desempacotamento da informação no destino
- *a execução do procedimento remoto leva mais tempo do que a execução do procedimento local*, está implícito um mecanismo de comunicação entre os dois espaços de endereçamento, o que requer que exista algum tipo de troca de mensagens entre eles.

alongamento

O que é uma chamada de procedimento remoto – 3

espaço de endereçamento A

```
/* obtém uma referência remota para o
      recurso compartilhado */
remRef = getRef (namingService);
                  . . .
/* invoca o procedimento xyz no
      recurso compartilhado */
tentar
{remRef.xyz();
}
pegar (erro)
{ /* erro de processo */ }
```

espaço de endereçamento B

```
/* instancia o recurso compartilhado */ locRef =
new SharedResource();

/* gera uma referência remota para o
      recurso compartilhado */
remRef = gerarRef (pubListenAdd);

/* registra-o em um serviço de nomenclatura */
regRef (remRef);
```

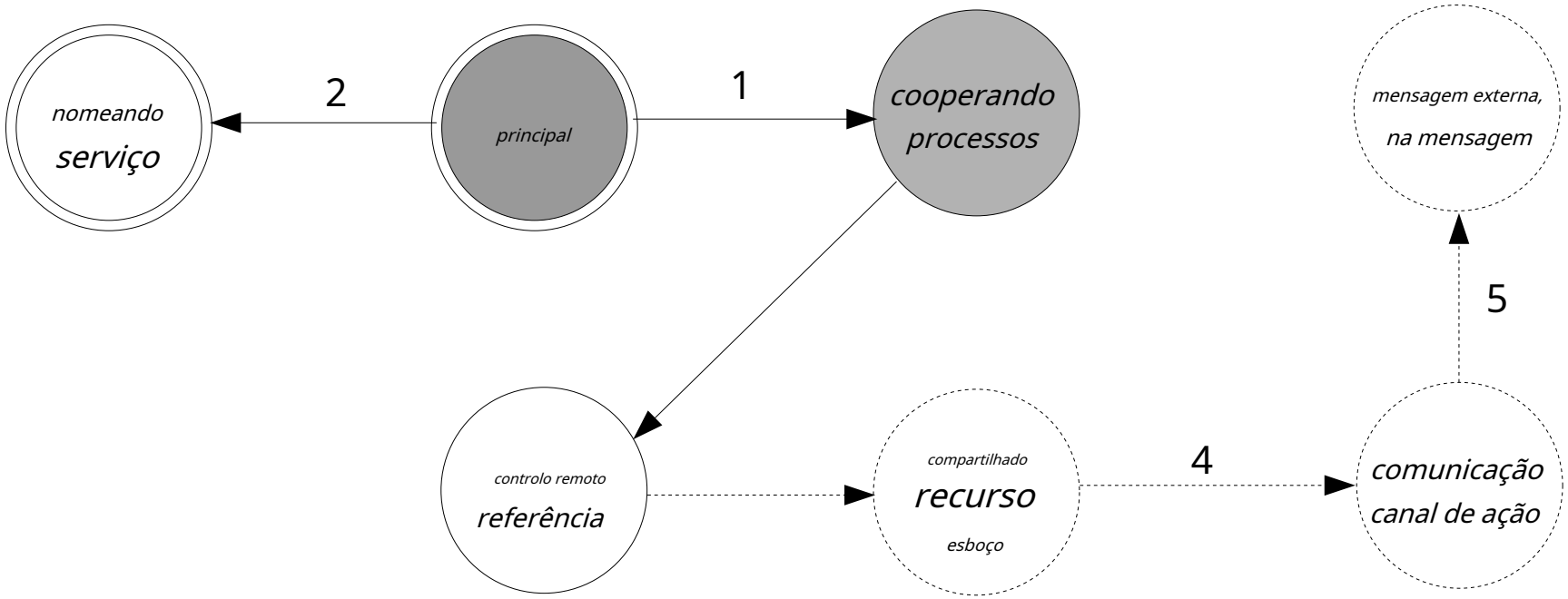
O que é uma chamada de procedimento remoto – 4

Um ambiente que implementa chamadas de procedimento remoto deve fornecer um *serviço de nomenclatura* para registrar recursos compartilhados que serão acessados remotamente. Uma forma de ver isso é assumir que funciona como uma lista telefônica organizada dinamicamente, mapeando, neste caso, o nome publicamente conhecido de um recurso compartilhado para sua localização e recursos de acesso. Assim, o programador da aplicação, para ter transparência na rede, só precisa saber a localização do serviço de nomenclatura e o nome da entrada do recurso compartilhado nele.

Por outro lado, o *controlo remoto* pode-se pensar que a referência ao recurso compartilhado contém o endereço de internet da plataforma onde o recurso compartilhado é instanciado, o número da porta utilizada para estabelecer uma comunicação com ele, a assinatura de todos os procedimentos que podem ser invocados nele e o nome de os arquivos que fornecem uma descrição dos tipos de dados usados como parâmetros de procedimento ou valores de retorno.

Arquitetura de um ambiente de chamada de procedimento remoto – 1

Endereçando o espaço A



1 – instanciar, iniciar, unir 2
– obter referência remota 3
– invocação de remoto
métodos

4 – instanciar, abrir, fechar, escreverObject,
lerObject

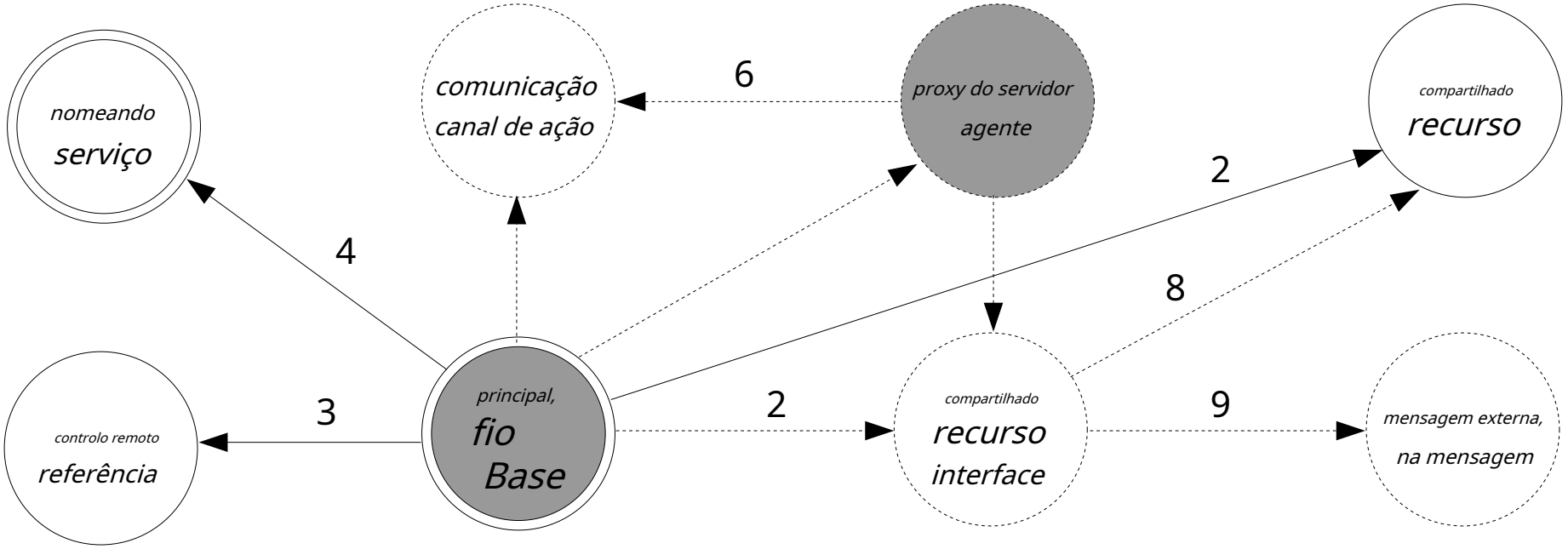
5 – instanciar, obter valores de campo

Arquitetura de um ambiente de chamada de procedimento remoto – 2

- apenas o código associado às entidades descritas por linhas contínuas deve ser escrito pelo programador da aplicação
- o código associado às entidades descritas por linhas tracejadas é gerado pelo ambiente de forma totalmente transparente para o programador
- o programador precisa apenas fornecer a assinatura de todos os procedimentos que podem ser invocados no objeto remoto e uma descrição dos tipos de dados usados como parâmetros de procedimento ou valores de retorno

Arquitetura de um ambiente de chamada de procedimento remoto – 3

Endereçando o espaço B



1 – instanciar, iniciar, terminar, aceitar

2 – instanciar

3 – gerar

4 – cadastre-se

5 – instanciar, iniciar

6 – readObject, writeObject, fechar 7 – processAndReply

8 – invocação de método

9 – instanciar, obter valores de campo

Arquitetura de um ambiente de chamada de procedimento remoto – 4

- apenas o código associado às entidades descritas por linhas contínuas deve ser escrito pelo programador da aplicação
- o código associado às entidades descritas por linhas tracejadas é gerado pelo ambiente de forma totalmente transparente para o programador
- o programador precisa apenas fornecer a assinatura de todos os procedimentos que podem ser invocados no objeto agora local e uma descrição dos tipos de dados usados como parâmetros de procedimento ou valores de retorno
- o agregado das entidades descritas por linhas tracejadas é geralmente chamado *esqueleto* e tem uma execução independente do *tópico principal* do aplicativo
- Assim, o *base de rosca* pode ser pensado como um thread que é instanciado e iniciado quando a referência remota é gerada

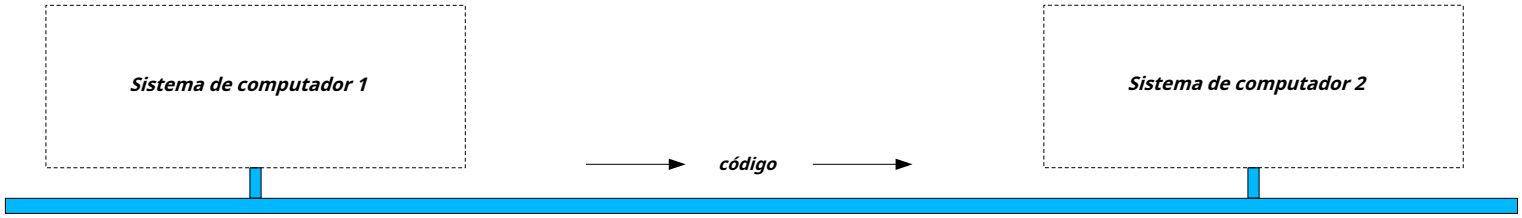
Migração de código – 1

*Migração de código*é um recurso altamente desejado em um sistema distribuído que permite que alguns componentes de uma aplicação sejam transferidos de um nó de processamento do sistema de computador paralelo subjacente para outro durante a execução do programa.

Várias razões podem levar a isso

- *aproveitando o poder de computação de nós de processamento específicos*: nem todos os nós de um sistema computacional paralelo podem ser semelhantes, portanto, partes dos cálculos a serem realizados podem ser movidas para nós específicos, sob demanda e de forma dinâmica, para que o programa seja executado de forma mais eficiente
- *tolerância ao erro*: durante a execução, alguns dos nós de processamento podem falhar, por isso seria importante, se não se quiser travar a aplicação em execução, poder detectar o mau funcionamento e realizar uma reconfiguração dinâmica dos componentes de software previamente atribuídos ao defeituoso nós para novos.

Migração de código – 2



Uma questão com a qual devemos lidar é a forma que o código a ser movido assumirá.

Várias opções estão disponíveis

- *código executável*: esta é a forma mais simples, mas requer que os nós original e de destino sejam relativamente semelhantes para que o código seja executado sem processamento adicional
- *Código fonte*: esta é a forma mais geral, nenhuma suposição precisa ser feita sobre as semelhanças dos nós originais e de destino
- *código intermediário*: isso é típico de situações em que o código a ser movido é executado por meio de um interpretador.

Migração de código – 3

A migração de código, embora seja um recurso bastante desejável, tem *segurança* problemas num contexto geral. Ou seja, deve-se garantir que o código recebido no nó de destino, sendo qualquer, não colocará em risco a integridade dos recursos do sistema computacional onde o nó está localizado.

A forma como se costuma lidar com o problema é introduzir no código um componente que gerencia a migração que monitorará continuamente o acesso aos recursos fornecidos pelo sistema operacional, decidindo caso a caso se os acessos devem ser permitidos ou negados .

Este componente é às vezes chamado de *gerente de segurança*.

Leitura sugerida

- *Sistemas Distribuídos: C uma vez n, 4ªEdi ção, Coulouris*, Dollimore, Kindberg, Addison-Wesley
  - Capítulo 5: *Objetos distribuídos e invocação remota*
    Seções 5.1 a 5.3 e 5.5
- *Sistemas Distribuídos: Princípios e Paradigmas, 2ªEdição,* Tanenbaum, van Steen, Pearson Education Inc.
  - Capítulo 10: *Sistemas distribuídos baseados em objetos* Seções 10.1 a 10.3.4