

Sistemas Distribuídos

Modelos de Programação Distribuída
Prof. Tales Viegas

O que é um Sistema Distribuído?

- ▶ Conjunto de computadores ligados em rede, com software que permita o compartilhamento de recursos e a coordenação de atividades, oferecendo idealmente um sistema integrado.

Computação Distribuída

- ▶ A computação é dividida em várias unidades que executam concurrentemente em diferentes elementos de computação
- ▶ Estes podem ser diferentes processadores em diferentes máquinas, diferentes processadores na mesma máquina ou diferentes cores no mesmo processador

Computação Distribuída

- ▶ Um sistema distribuído é o resultado da interação de vários componentes que atravessam toda a pilha (stack) de computação, desde o Hardware até o Software

Hardware

- ▶ Computador + Infraestrutura de rede
- ▶ Geralmente gerenciado pelo Sistema Operacional (SO), que fornece serviços para:
 - Comunicação entre processos (IPC – Inter-Process Communication)
 - Escalonamento e gerenciamento de processos
 - Gerenciamento de recursos, como o sistema de arquivos e periféricos
- ▶ HW + SO = Plataforma

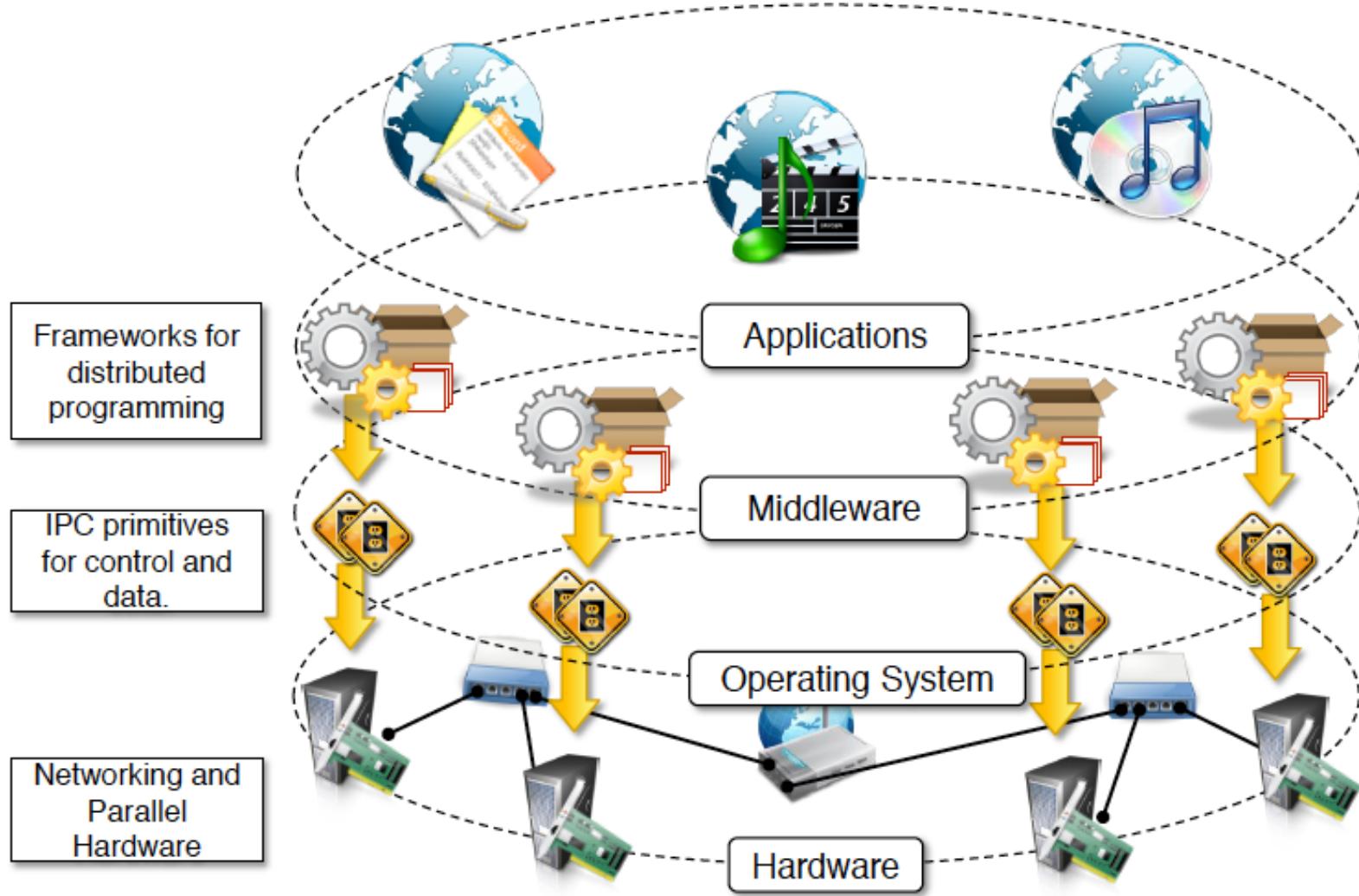
Sistema Operacional

- ▶ A nível de SO, a comunicação entre processos é implementada utilizando protocolos como:
 - TCP/IP (Transmission Control Protocol/Internet Protocol)
 - UDP (User Datagram Protocol)
 - Entre outros

Middleware

- ▶ Acima do SO, o middleware constrói uma camada que permitirá às aplicações abstraírem a heterogeneidade do Hardware e dos Sistemas Operacionais
- ▶ O middleware fornece uma interface uniforme para o desenvolvimento das aplicações

Modelos de Programação Distribuída



Comunicação entre Processos (IPC)

- ▶ A comunicação entre processos é usada tanto para transferir dados entre os processos, como para coordenar a sua atividade
- ▶ Modelos de IPC mais importantes:
 - Memória Compartilhada
 - Comunicação por mensagens

Sistemas de Memória Compartilhada

- ▶ Os processos acessam o mesmo espaço de endereçamento de memória
- ▶ Comunicação através de variáveis compartilhadas
- ▶ Sincronização realizada pelas técnicas clássicas da programação concorrente (semáforos ou monitores)

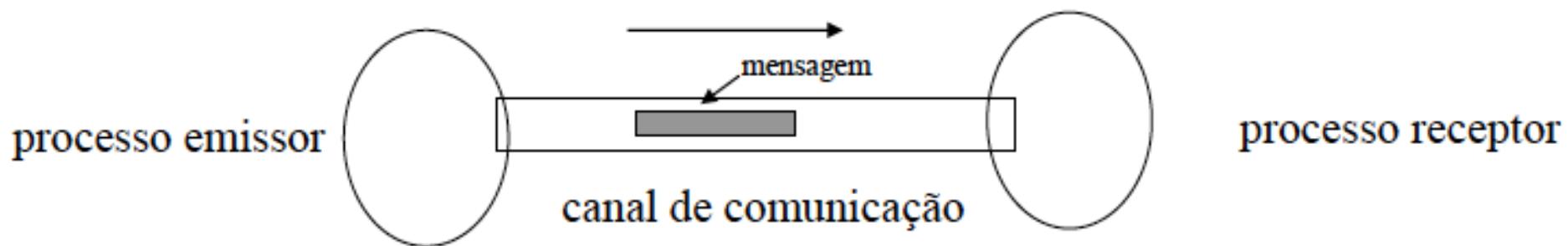
Sistemas de Memória Distribuída

- ▶ Vários espaços de endereçamento disjuntos
 - Cada processador tem a sua própria memória local
- ▶ Comunicação por mensagens
 - Através de um canal de comunicação
- ▶ Comunicação e sincronização integradas em um único conceito

Sistemas de Memória Distribuída

- ▶ O programador utiliza os mecanismos de comunicação por mensagens sem se preocupar com a forma como é feito o armazenamento e a transferência de dados
- ▶ As várias formas de comunicação por mensagens distinguem-se por:
 - Tipo de Sincronização
 - Forma como são especificados os vários intervenientes no processo

Tipos de Sincronização



- ▶ Comunicação Síncrona
- ▶ Comunicação Assíncrona
- ▶ Invocação Remota de Procedimentos

Comunicação Síncrona

- ▶ O envio de uma mensagem é uma operação atômica que requer a participação de 2 processos (emissor e receptor)
- ▶ Se o emissor está pronto para enviar a mensagem, mas o receptor não pode recebê-la, ele a bloqueia
- ▶ Se o emissor está pronto para receber a mensagem, mas o emissor não a envia, o receptor fica bloqueado

Comunicação Síncrona

- ▶ O ato de comunicação sincroniza as sequências de execução dos dois processos
- ▶ O processo emissor aguarda a resposta do processo receptor antes de continuar a sua execução
- ▶ Conceito de mais baixo nível (mais eficiente)

Comunicação Assíncrona

- ▶ O emissor pode enviar a mensagem e continuar a executar sem bloquear, independentemente do estado do processo receptor
- ▶ O receptor pode estar executando quaisquer outras instruções e, mais tarde, testar se tem mensagens a receber
- ▶ Quando ele aceitar a mensagem não tem conhecimento do estado do emissor (que pode até já ter terminado)

Comunicação Assíncrona

- ▶ Permite um maior grau de concorrência
- ▶ Exige que o sistema de execução faça a gestão e o armazenamento das mensagens
 - Um buffer de memória tem de estar preparado para armazenar um número de mensagens potencialmente ilimitado

Invocação Remota de Procedimentos

- ▶ RPC (Remote Procedure Calling)
- ▶ A comunicação entre dois processos é uma chamada de procedimento remoto quando:
 - Um processo (o emissor) envia a mensagem
 - O processo receptor produz uma resposta à mensagem e...
 - O emissor permanece suspenso até a recepção da resposta

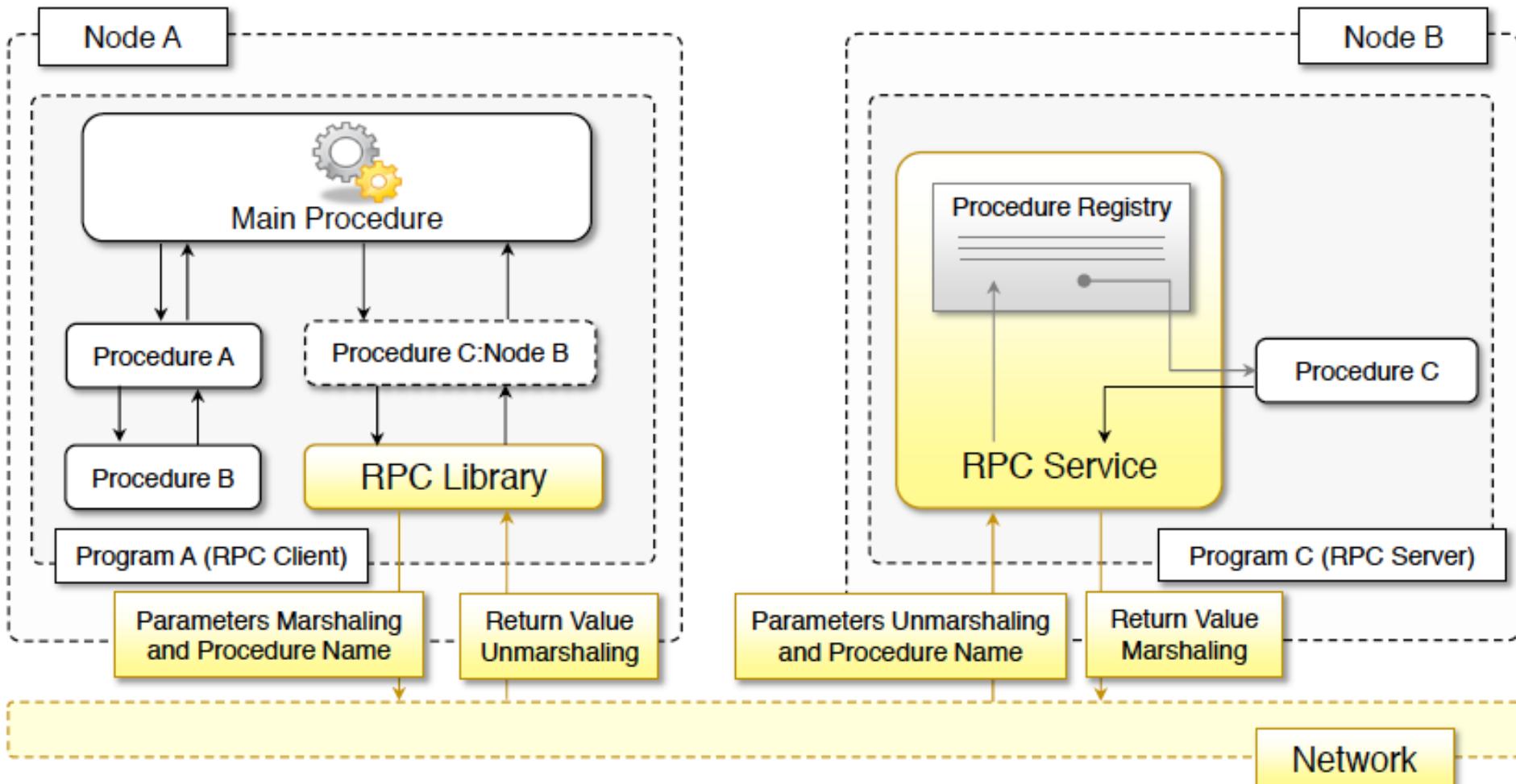
Invocação Remota de Procedimentos

- ▶ Do ponto de vista do processo emissor (cliente) este mecanismo funciona como uma chamada de procedimento
- ▶ O procedimento não vai ser executado no próprio processo, mas sim no receptor (servidor)
- ▶ Os dados comunicados na mensagem funcionam como parâmetros do tipo valor
- ▶ A resposta do receptor pode ter a forma de parâmetros resultado

Invocação Remota de Procedimentos

- ▶ O sistema de execução é responsável por encapsular sob a forma de mensagem qual o procedimento a executar e os respectivos parâmetros
- ▶ Após a execução do procedimento o resultado é uma mensagem do processo remoto para o cliente
- ▶ O estado do procedimento não perdura entre as invocações

Invocação Remota de Procedimentos



Invocação Remota de Procedimentos

- ▶ Variante:
- ▶ Emissor
 - Após o envio da mensagem, o emissor prossegue a execução até que precise do resultado (permite maior concorrência)
 - Se, nesse ponto, a resposta ainda não estiver disponível, o emissor é suspenso

Invocação Remota de Procedimentos

► Receptor

- Quando um processo executa uma instrução de aceitação de mensagem, é suspenso até a chegada da mesma
- Pode acontecer que o receptor pretenda:
 - Escolher uma entre um conjunto de mensagens possíveis
 - Estabelecer condições para o recebimento de uma mensagem

Invocação Remota de Procedimentos

► Receptor (cont)

- Para isto é necessário que exista uma instrução em que o receptor:
 - Selecione uma de um conjunto de mensagens alternativas
 - Cada uma das quais pode ter associada uma condição para a aceitação das mensagens

Especificação dos Intervenientes

- ▶ Identificação dos Processos
- ▶ Criação dos Processos (dinâmica ou estática)
- ▶ Comunicação bi ou uni-direcional

Identificação dos Processos

- ▶ Sistema em que todos os processos tem um nome único
- ▶ O comando de envio pode nomear diretamente o processo receptor
 - ENVIA <mensagem> PARA <nome do processo>
- ▶ Simetricamente no receptor:
 - ESPERA <mensagem> DE <nome do processo>
 - Requer que o receptor saiba o nome de todos os processos possíveis de lhe enviar uma mensagem

Identificação dos Processos

- ▶ Se o receptor apenas estiver interessado em receber determinada mensagem, não importando quem é o emissor:
 - ESPERA <mensagem>
 - O emissor é anônimo, o receptor não

Identificação de Processos

- ▶ Quando não é apropriado um sistema de nomes únicos para todos os processos, definem-se entidades intermediárias
 - Caixas de correio (ou canais)
- ▶ Devem ser conhecidas por ambos os intervenientes na comunicação:
 - ENVIA <mensagem> PARA <caixa de correio>
 - ESPERA <mensagem> DE <caixa de correio>

Identificação de Processos

- ▶ Uma caixa de correio pode ter várias formas.
Pode ser usada por:
 - Vários emissores e vários receptores
 - Um emissor e vários receptores (difusão ou “broadcasting”)
 - Vários emissores e um receptor
 - Um receptor e um emissor
- ▶ Pode ser ainda estruturada para enviar informação em ambas as direções ou apenas em uma

Formas de Criação de Processos

- ▶ Podem ser criados de forma estática ou dinâmica
- ▶ Definição Estática
 - Todos os processos são criados no início da execução
 - A atribuição de recursos (memória, canais de comunicação, etc) é feita em tempo de compilação
 - Mais eficiente

Formas de Criação de Processos

- ▶ Definição Dinâmica
 - Permite maior flexibilidade
 - O sistema ajusta-se às necessidades da aplicação ao longo do tempo
 - Permite mecanismos de balanceamento de carga (“load balancing”)
- ▶ A criação estática de processos é apropriada para sistemas dedicados, onde a configuração do sistema é conhecida no momento da execução

Modelos de Interação por Mensagens

► Comunicação Ponto-a-Ponto

- Cada mensagem é enviada de um componente para outro
- O processo que envia a mensagem tem de conhecer o endereço do receptor
- A comunicação pode ser síncrona ou assíncrona
- A comunicação assíncrona implica um sistema de armazenamento de mensagens

Modelos de Interação por Mensagens

▶ Publish-and-Subscribe

- Os processos podem desempenhar um de dois papéis
- O publisher, que permite aos outros processos subscreverem um determinado tópico ou evento
- Quando o evento ocorre no lado do publisher, é desencadeada a criação de uma mensagem associada ao evento e que ficará disponível para todos os subscriptores deste evento

Modelos de Interação por Mensagens

- ▶ Para os subscriptores serem notificados do evento, há duas estratégias possíveis
 - Push strategy: O publisher tem a responsabilidade de notificar todos os subscribers
 - Pull strategy: O publisher cria a mensagem, mas são os subscriptores que tem de verificar se há mensagens para um dado evento

Modelos de Interação por Mensagens

▶ Request–Reply

- Inclui todos os modelos para os quais para cada mensagem enviada há uma resposta correspondente