Comunicação em Java

Autor

- **Autor**
  - Cláudio Geyer
- **Local**
  - Instituto de Informática
  - UFRGS
  - disciplinas: POD e PDP
  - Versão
    - v4.1
    - 2013-1

Súmula

- **Súmula**
  - Sockets
  - RMI
  - CORBA: IIOP
  - JavaSpaces
  - J2EE
  - JMS
  - Web Services

- **Introdução**
  - ➢Objetivos
    - Apresentar uma visão geral dos mecanismos atuais para comunicação em Java
    - Ênfase em mecanismos “comerciais”

Java Sockets

- troca de mensagens ponto-a-ponto e multicast
- protocolos (sockets) TCP e UDP
- vantagens gerais
  - mais eficiente
  - mais flexível (com relação a algoritmos e padrões de comunicação)
  - apreendizado (mais) rápido
  - API uniforme (sobre outras API sockets)
- desvantagens gerais
  - menos legível
  - não orientado a objetos em termos de chamada
    - Mas usa classes e objetos

- **Java RMI**
  - chamada de método remota
  - vantagens
    - (mais) orientado a objetos
    - (quase) transparente com relação à chamada local
      - sintática e semanticamente
    - mais legível (versus sockets)
    - atendimento concorrente a vários clientes
      - Uso de máquinas multiprocessadores / multicore
      - Melhor atendimento de chamadas curtas
    - transferência automática de código (por classes)

Java RMI

- desvantagens
  - menos flexível (com relação a algoritmos e padrões de comunicação)
  - menos eficiente (versus sockets)
  - somente entre programas Java
  - síncrono: espera execução do método
  - modelo de relacionamento cliente/servidor em nível de objetos
    - Simples, único
    - 1 servidor para vários clientes
    - criação (gerência) de servidores pelo servidor

- **Java CORBA/IIOP**
  - chamada de método remota
  - vantagens
    - orientado a objetos
    - (quase) transparente com relação à chamada local
      - sintática e semanticamente
    - mais legível (versus sockets)
    - entre Java e outras linguagens
    - CORBA: síncrono e opções de assincronismo

- **Java CORBA/IIOP**
  - desvantagens
    - menos flexível (com relação a algoritmos e padrões de comunicação)
    - menos eficiente (versus sockets)
  - Modelo CORBA
    - Padrão com inúmeros funcionalidades adicionais
      - Transações, persistência, chamada dinâmica, ...
    - Java oferece somente a chamada estática de objetos

- **JavaSpaces**
  - comunicação por memória compartilhada distribuída
  - similar ao modelo Linda
  - vantagens
    - independente da vida dos parceiros
    - facilita tratamento da concorrência
    - suporta transações
    - suporta eventos (comunicação assíncrona)
  - desvantagens
    - menos eficiente que RMI
    - somente entre programas Java
    - chamada síncrona (sem eventos)

- **J2EE/EJB**
  - similar a RMI
  - menos eficiente em termos de comunicação
    - implementação sobre RMI
  - Mas oferece recursos para gerência de objetos servidores
    - melhor desempenho que RMI
  - mais transparente
  - outras vantagens de EJB e J2EE
    - suporta transações
    - diferentes tipos de objetos servidores (sessão, persistência, ...)
    - diferentes tipos de sincronização

- **JMS**
  - troca de mensagens ponto-a-ponto
  - conceito de canais (mail box)
    - independência de local e tempo
  - diversas funcionalidades e opções
  - menos eficiente que sockets
  - eficiência X RMI?
  - usada em J2EE

- **Web Services**
  - diversas APIs
  - diversos fornecedores
  - sobre diversos protocolos (possível ao menos)
  - independentes de plataformas (SO, ...) e linguagens
  - similar a RPC:
    - cliente chama um serviço remoto com ou sem resposta
    - mas oferece variações na sincronização

Novos Paradigmas

- **Novos paradigmas (modelos) computacionais**
  - Exemplos de novos paradigmas
    - Computação em grade
      - Inúmeras variações
    - P2P
    - Computação ubíqua
    - Tolerância a falhas (nem tão novo ...)
    - IA, multiagentes
  - Geram novos requisitos de PD
  - Novos requisitos de PD exigem novas soluções de comunicação (APIs, protocolos, ...)

Novos Paradigmas

- **Novos paradigmas (modelos) computacionais**
  - Exemplos de novos requisitos
    - Grid
      - Latências e bandas diversas
      - Diferentes hw para redes -> diferentes protocolos de baixo nível
      - Mais ênfase em escalabilidade

Novos Paradigmas

- **Novos paradigmas (modelos) computacionais**
  - Opcionalmente novos níveis de abstração
    - Comunicação por chamada de método remoto
    - Com funcionalidades de mais alto nível
    - Por exemplo, em computação voluntária, pedir uma tarefa
  - Exemplos
    - JXTA: solução Java (Sun) para sistemas p2p

- **Pesquisa**
  - Muitos projetos de pesquisa oferecem outros mecanismos
  - Em instituições de pesquisa
  - Em comunidades específicas (sw livre)
  - Middlewares e frameworks com variações em
    - Confiabilidade
    - Estabilidade
    - Documentação
  - Exemplos
    - Projeto ProcAtive: ênfase em grid
    - Projeto ISAM/Exehda (Gradep): ênfase em Ubicomp

Comunicação em Java