Seminário de MAC5755

Jini e JavaSpaces

Novembro de 2001

Márcio Carneiro <carneiro@ime.usp.br>

Professor Fábio Kon

Motivação

Redes
- São heterogêneas e complexas
- Pretendemos utilizá-las para interligar todo equipamen- to imaginável: computação ubíqua
- Configuração é bastante complicada e manual: leigos?

Java
- Multiplataforma e pronto para as redes
- Já se falava no começo como um sistema "embedded"
- Características como OO, RMI, carga dinâmica, etc

Redes + Java ▷ JINI
- Solução para serviços distribuídos
- JavaSpaces (serviço do Jini)
  - Especificação para persistência distribuída

JINI - Descrição

O que é?

- Jini é um sistema distribuído baseado em federações de usuários, recursos e serviços disponíveis
- É um middleware (entre Java e serviços)

Composto de:

- Infra-estrutura para a federação de serviços (API, Core)
- Um modelo de programação, para a criação de serviços
- Serviços que oferecem funcionalidades à federação

O que não é:

- Um sistema operacional (pode ser usado em qualquer máquina Java)
- Uma linguagem (baseia-se na linguagem Java)
- Um aplicativo (é uma estrutura para criação de serviços)

JINI - Objetivos e Base

Principais objetivos:

- Prover aos usuários compartilhamento de recursos e serviços numa rede heterogênea
- Permitir fácil acesso a esses recursos e permitir mobilidade do usuário na rede
- Simplificar a criação, manutenção e alteração de uma rede de dispositivos, serviços e usuários

Uso do Java:

- A idéia é criar uma rede de máquinas virtuais
- Possibilidade de mover dados e código
- Segurança interna da JVM
- Verificação forte de tipos da linguagem

Resumo ▷ Sistemas distribuídos: a federação de componentes e serviços aparentam um único sistema para o usuário, com simplicidade e rapidez na configuração e utilização

JINI - Conceitos

Serviços

- É uma entidade que pode ser utilizada por um usuário, um programa ou outro serviço (são participantes da federação)
- São algoritmos, aplicativos, dispositivos, ou até usuários
- Os serviços se comunicam por um protocolo de servi- ço. Alguns crí ticos já estão definidos no sistema.
- A interface dos serviços deve ser previamente definida

Serviço de procura (lookup service ou LS)

- É o organizador do sistema: os serviços se registram e os clientes buscam serviços nos LS disponíveis
- Para se registrar, os serviços utilizam dois protocolos: o discovery (procurar os LS) e o join (se registrar).
- LS pode conter objetos de outros LS, ou mesmo de outros serviços distribuídos (nome, diretório, etc)

JINI - Elementos

Java RMI

- É usado como base para mover os objetos e comunicação
- Pode não ser utilizado na implementação de um serviço

Contratos (lease)

- Acesso para a maioria dos serviços é baseado em contratos
- Têm duração ou podem ser vitalícios
- Podem ser exclusivos ou não-exclusivos

Transações

- Pode-se agrupar operações em transações atômicas

Eventos

- Jini inclui eventos distribuídos entre objetos do sistema

JINI - Discovery

Três protocolos de descobrimento

Multicast request protocol

- Utilizado para que entidades (serviços) descubram os LS disponíveis na rede (UDP)

Multicast announce protocol

- Utilizado para que LS possam se pronunciar como ativos numa rede (UDP)

Unicast discovery protocol

- Utilizano na comunicação entre um LS e um serviço para finalização do processo de descobrimento (TCP)

Multicast request protocol

No processo de descobrimento

- Serviço (entidade "descobridora" - discovering entity)
  - Multicast request client: faz pedido para procurar serviços de busca (LS)
  - Multicast response server: espera respostas dos LS presentes
- Serviço de busca (LS)
  - Multicast request server: espera pedidos que procuram serviços de busca
  - Multicast response client: responde com proxy que permite acesso ao LS

Utiliza UDP (pacotes de até 512 bytes)

- Formato:
  - Versão do protocolo (int)
  - Porta do multicast response server (int)
  - Número de LS conhecidos (int)
  - Vetor de LS conhecidos (net.jini.core.lookup.ServiceID[])
  - Número de grupos (int)
  - Vetor de grupos (java.lang.String[])
- Os dados devem ser independentes de plataforma (uso de UTF, e java.io.ByteArrayOutputStream)

Multicast request protocol

Na entidade descobridora

- Inicia o cliente de pedido, que envia mensagens UDP para ume porta pr é-definida
- Inicia um soquete TCP na porta escolhida (servidor de resposta). Resposta do LS pelo unicast discovery protocol
- Envia pedidos incluindo os recebidos por um período
- Se não for suficiente, inicia o multicast announce protocol

No LS

- Um soquete UDP espera na porta pré-definida
- Se o pedido puder ser atendido (pertence ao grupo e não aos IDs), se conecta à porta TCP indicada
- Inicia o unicast discovery protocol

Multicast announce protocol

No LS

- Inicia um soquete TCP para esperar o unicast discovery protocol de alguma entidade em reposta ao anúncio
- Envia pacotes UDP para uma porta conhecida
- Formato:
  - Versão do protocolo (int)
  - Domínio para o unicast (java.lang.String)
  - Porta para resposta (int)
  - ID do LS de origem (net.jini.core.lookup.ServiceID)
  - Número de grupos (int)
  - Vetor de grupos que representa (java.lang.String[])

Na entidade descobridora

- Inicia um soquete UDP numa porta conhecida
- A cada anúncio de LS, verifica se já não conhece ou se está nos grupos de interesse, e então inicia o protocolo unicast para recebimento do objeto do LS

Unicast discovery protocol

Usado na segunda parte do descobrimento automático (multi- cast), ou manualmente pelo usuário (e.g., serviços distantes)

Uma das duas partes inicia uma conexão TCP com a outra ponta, na porta indicada no início, ou pelo usuário

A entidade descobridora:

- Envia apenas a versão do protocolo

O LS:

- Responde com o objeto net.jini.core.lookup.Ser- viceRegistrar (o objeto enviado é um java.rmi.Mar- shalledObject do original)

A partir disso, pode-se passar para o join protocol.

JINI - Join

Cada serviço deve conter dados persistentes:

- Uma identificação (service ID)
- Um conjunto de atributos que identificam o serviço
- Um conjunto de grupos a qual deseja participar
- Um conjunto de LS a qual deseja se registrar

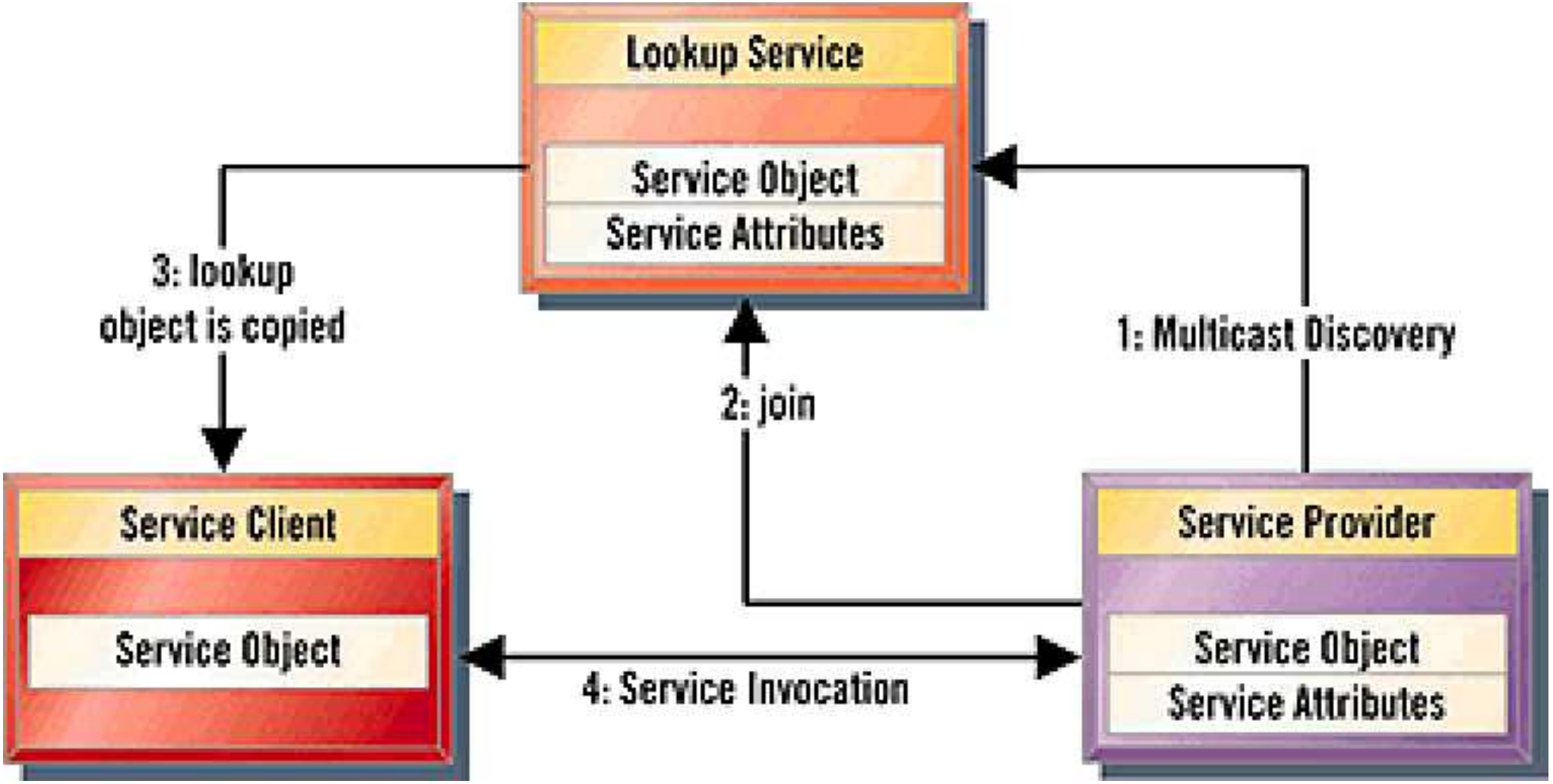
O serviço pode se registrar no LS através do método register, enviando um ServiceItem:

- Entry[] attributeSets
- java.lang.Object service
- ServiceID serviceID

Cada serviço iniciado faz uma pausa aleatória para evitar um congestionamento em redes com muitos serviços

JINI - Serviço e cliente

Visão geral da relação entre os elementos



net.jini.discovery.LookupDiscovery

LookupDiscovery(java.lang.String[] groups)

- Construtor, recebe grupos para pesquisa
  - static java.lang.String[] ALL_GROUPS

void addGroups(java.lang.String[] newGroups)

java.lang.String[] getGroups()

void removeGroups(java.lang.String[] oldGroups)

void setGroups(java.lang.String[] newGroups)

- Funções para manipular os grupos do objeto

ServiceRegistrar[] getRegistrars()

- Função para pegar o resultado (referências aos LS)

void discard(ServiceRegistrar reg)

JINI - Entradas e modelos

Para o registro e busca de serviços há dois tipos de classes

Entry (entradas)

- Deve ser utilizada para os serviços se registrarem
- Contém atributos do serviço

ServiceTemplate

- Entry[] attributeSetTemplates
- ServiceID serviceID
- java.lang.Class[] serviceTypes

Pode-se utilizar campos nulos para representar qualquer valor

Há um pacote específico com padrões de tributos e repre- sentantes para JavaBeans (net.jini.lookup.entry)

JINI - Contratos (lease)

A maioria dos serviços do Jini utilizam esse paradigma

- O contradado define o período, mas pode ser negociado
- O contrato pode ser cancelado pelo contratante
- O contratante pode renovar o contrato
- Passado o período, o contrato expira

Período

- O contratante pode requerir um determinado tempo
- O contradado pode aceitar um período menor
- Pode ser requerido um contrato sem período (FOREVER)
- Pode-se representar o tempo absoluto ou relativo

Mapeamento

- Os contratos podem ser mapeados em grupos
- As implementações definem se o contrato pode ser reunido com outro (canBatch) ▷ reflexivo, simétrico e associativo

JINI - Eventos

Jini define interfaces para a manipulação de eventos distribuídos

Eventos são alterações que podem ocorrer determinado objeto

Elementos
- Objeto interessado em eventos de um outro objeto
- O objeto gerador de eventos
- Um receptor dos eventos (listener) gerados pelo anterior

O interessado deve registrar um receptor de eventos no gerador, e recebe um contrato de validade do registro

Conteúdo dos eventos
- Um identificador do tipo de evento
- Uma referência ao objeto que gerou o evento
- Um número de seqüência da instância do evento
- Um objeto que o interessado registrou interesse

JINI - Transações

Jini define uma estrutura mínima para a realização de tran- sações: conjuntos de operações, que devem ter a garantia de ocorrer em todos participantes, ou não ocorrer em nenhum

Utiliza o modelo de duas fases: two-phase commit

Gerente (Manager)

- Um objeto deve ser o responsável pela organização
- Ele aceita a transação, fornecendo ID único
- TransactionFactory cria a classe com a transação
- Cliente envia para os participantes

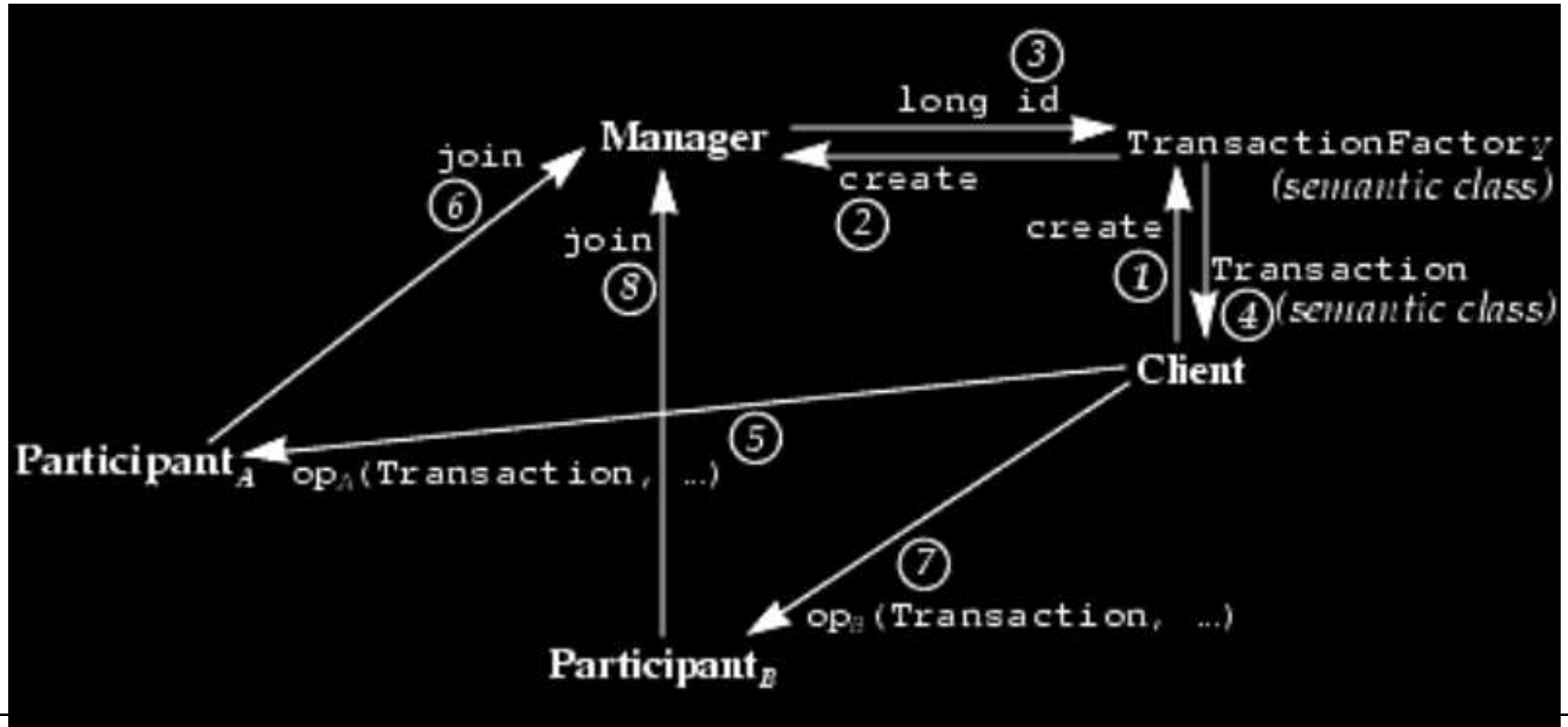
Transação

- Participantes votam sim (prepared), nulo (not changed), não (aborted)
- Gerente autoriza os que disseram sim, se não houve não

JINI - Transações

As propriedades ACID são deixadas a cargo da implementação dos objetos, bem como a semântica das transações

Uma semamântica padrão é implementada, e essa respeita as propriedades ACID



Serviço JavaSpaces

Especificação para um sistema distribuídos de dados persistentes

Um serviço JavaSpaces guarda objetos que implementam a interface net.jini.core.entry.Entry

Buscas podem ser feitas utilizando os modelos (templates)

Utiliza os contratos, transações e eventos

Não é um banco de dados: a intenção é de auxiliar na construção de sistemas distribuídos

Implementação comercial: http://www.gigaspaces.com/

JINI - Análise

Vantages

- É Java
- Simplicidade e flexibilidade
- Suporte a eventos, transações
- Transparência para as implementações
- "Plug-and-Play networks"

Desvantagens

- É Java
- As interfaces devem ser previamente conhecidas
- Dependência de padronização para uso genérico
- Não tem muito suporte a segurança
- Não é tão escalável
- Não tornou-se tão popular (ainda?)

JINI - Casos

eko systems, inc.

- Utilizaram Jini para inteligar equipamentos médicos: http://www.ekosystems.com/

Energy's Pacific Northwest National Laboratory (PNNL)

- Estão desenvolvendo com Jini sensores para ambientes de alto risco: http://www.pnl.gov/

Padrões

- ServiceUI
  - Padronização de um serviço para interfaces de usuários
- Print
  - Definir uma API padrão para impressoras
- Surrogate
  - Definir um padrão para viabilizar a participação em serviços Jini de equipamentos que não podem se comunicar diretamente com o serviço

JINI - Informações

Sítio da Sun http://www.sun.com/jini

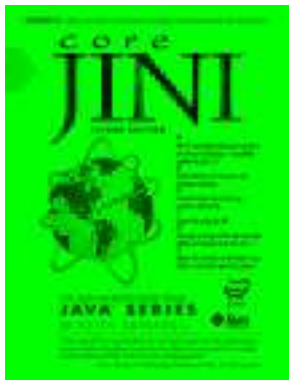
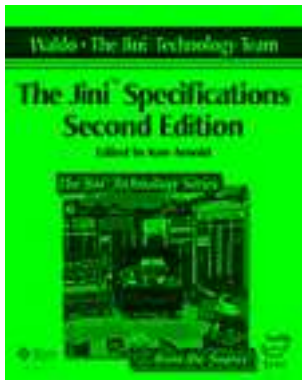
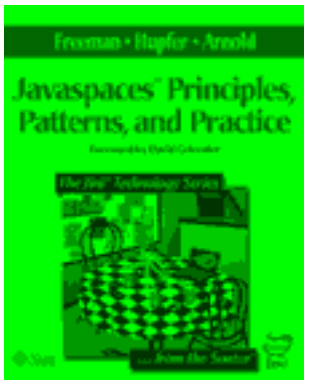
- Textos, casos, e especificações

Jini.org http://www.jini.org

- Pesquisas, projetos, e padrões

Livros

Core Jini, W. Edwards, Prentice Hall, 2ª edição, Dez/2000

The Jini Specifications, Ken Arnold (editor), Addison-Wesley, 2ª edição, Dez/2000

Javaspaces Principles, Patterns, and Practice, Eric Freeman, Addison-Wesley, Jun/1999