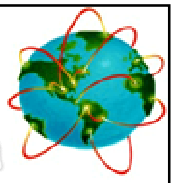
Tecnologia JINI



Profª Ana Cristina Barreiras Kochem Vendramin

Tecnologia Jini

- Baseada na linguagem de programação Java ➨ independência de plataforma;
- Implementação de diferentes serviços em uma infra-estrutura de rede distribuída;
- Dispensa *drivers* ➨ sistema *plug-and-play*;
- Comunicação entre dispositivos (ex.: TVs, DVDs, impressoras, etc).
- Flexibilidade – a falha de um dispositivo não afeta os demais.
- Permite que um assinante em um JVM receba notificações de eventos de um objeto em outra JVM, normalmente em outro computador.

Serviços

- Definidos por uma ou mais interfaces Java;
- Para que um serviço se torne disponível, ele precisa se registrar com cada servidor *lookup* que faça parte da comunidade Jini da qual deseja participar;
- Possuem um objeto *proxy* que implementa suas interfaces.
- É através do *proxy* que os clientes poderão se comunicar com os serviços desejados.

  Ex.: serviço de impressão ➨ o *proxy* implementa a interface padrão *Printer*.

Serviços

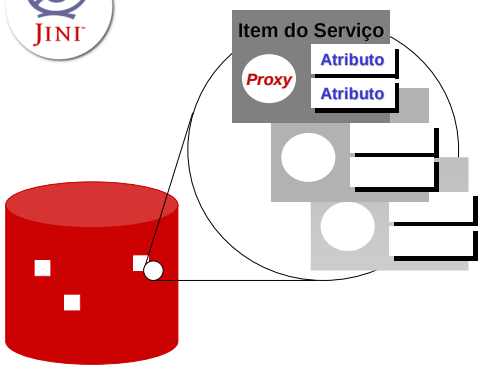
Exemplos de Serviços Jini:

- Serviço de Impressão ➨ imprime aplicações Java;
- Serviço *JavaSpaces* ➨ serviço de armazenamento persistente de objetos Java;
- Gerente de uma Transação ➨ agrupa várias operações em uma transação Jini para serem executadas simultaneamente.

Servidor *Lookup*

- Representa os serviços disponíveis em uma comunidade Jini
- Fornece habilidades para pesquisar e encontrar tais serviços
- Quando um serviço se registra com um servidor *lookup* que faça parte da comunidade da qual deseja participar, ele passa a esse servidor um item de serviço que é único para um determinado serviço.

Servidor *Lookup*



Protocolos de Comunicação

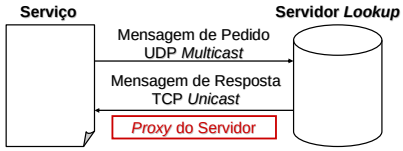
Protocolos *Discovery*

- Utilizados pelos serviços para encontrarem os servidores *lookup* adequados e obter o *proxy* desses servidores.
- Responsável pela construção de comunidades Jini
- Três protocolos *discovery*:
  - Protocolo de pedido *multicast;*
  - Protocolo de anunciamento *multicast;*
  - Protocolo *Discovery Unicast*.

Protocolos *Discovery*

Protocolo de Pedido *Multicast*

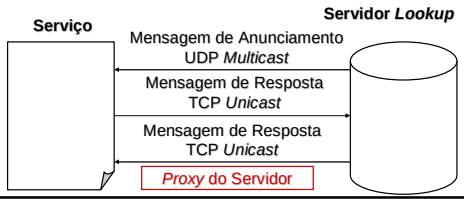
- Serviço envia uma mensagem para um endereço *multicast* conhecido e porta padrão onde todos os servidores *lookup* estarão escutando
- Mensagem especifica a comunidade Jini de interesse
- Servidor responsável pela comunidade se conectará diretamente ao serviço provendo o seu *proxy*



Protocolos *Discovery*

Protocolo de Anunciamento *Multicast*

- Utilizado durante toda a vida útil de um servidor *lookup* para anunciar sua presença na rede
- Quando um serviço receber um anúncio da existência de um servidor *lookup* de seu interesse, ele pode se conectar diretamente a ele através do TCP *unicast*



Protocolos *Discovery*

Protocolo *Discovery Unicast*

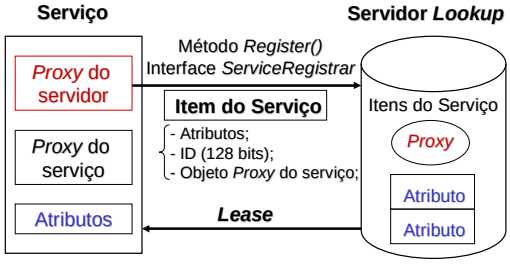
- Utilizado durante o curso dos dois protocolos *multicast*;
- Utilizado para estabelecer comunicações com um servidor *lookup* específico em uma rede;
- Informações necessárias:
  - Nome do *host* que abriga o servidor *lookup*;
  - Número da porta.

Protocolo *Join*

- Uma vez que um serviço tenha obtido o *proxy* de um servidor *lookup*, ele estará pronto para executar o protocolo *Join,* tornando-se parte do grupo de serviços registrados nesse servidor
- Serviço invocará o método *register()* da interface *ServiceRegistrar* implementada pelo *proxy* do servidor recebido
  - O item do serviço será passado como argumento
- Após o registro, o serviço receberá um *lease* → determina o tempo em que ficará registrado nesse servidor

Protocolos de Comunicação

Protocolo *Join*

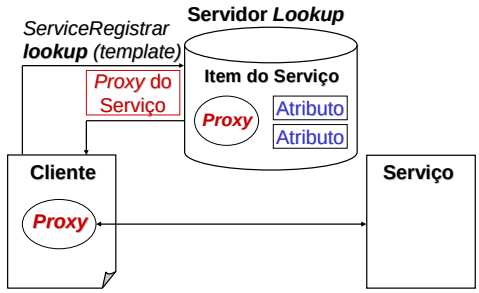


Profª. Ana Cristina B. Kochem
DAINF/CEFET-PR

Protocolo *Lookup*

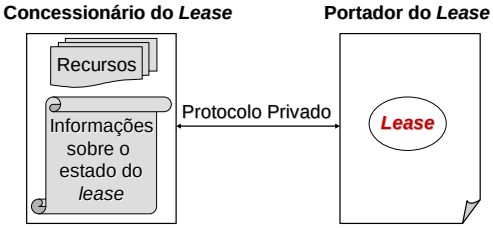
- Permite que um cliente pesquise nos servidores *lookup*, após obtido o *proxy* deste, os serviços que possam auxilia-los na realização de suas metas
- Invoca o método *lookup()* declarado na interface *ServiceRegistrar* implementada pelo *proxy* do servidor
- Argumento → *template*
  - Pode ser usado para procurar um *proxy* do serviço que implemente uma determinada interface
  - A procura também pode ser pelo ID do serviço ou pelos atributos que o descrevem
- Se um serviço for encontrado, o cliente receberá o seu *proxy* podendo invocar métodos nesse *proxy* para interagir com o serviço.

Protocolo *Lookup*



Leasing

- *Lease* → objeto que permite a concessão de recursos por certos períodos de tempo



Leasing

Um *lease* pode ser solicitado por um objeto para:

- Obter acesso a um recurso;
- Receber notificações de eventos;
- Utilizar um serviço de armazenamento persistente (*JavaSpaces*);
- Anunciar disponibilidades nos servidores *lookup*.

Os *leases* podem ser:

- Cancelados;
- Renovados antes que expirem.

Eventos Remotos

- Evento → algo que acontece em um objeto correspondendo a alguma mudança no estado desse objeto

Exemplos de Eventos:

- Um novo serviço disponível na rede;
- Um serviço que não está mais registrado em um servidor *lookup*;
- Uma impressora sem papel.

Eventos Remotos

Objetos envolvidos na especificação de evento distribuídos Jini

- Evento Remoto: objeto que representa a notificação
- Gerador de eventos:
  - Objeto no qual o evento ocorre;
  - Registra assinantes interessados em seus eventos e gera notificações
- Ouvinte de eventos remotos: Recipiente de notificações de eventos.
- *Third-party-agents*: observadores interpostos entre um gerador de evento e um assinante. Podem armazenar notificações para assinantes até que eles possam recebe-las.

Profª. Ana Cristina B. Kochem
DAINF/CEFET-PR

Eventos Remotos

- Java RMI é usado para enviar notificações do gerador de eventos para os assinantes, com possibilidade de ser via *third-party agents*
- Eventos em Jini são fornecidos por meio das seguintes classes e interfaces:
  - ***RemoteEventListeners***
    - Fornece um método chamada *notify*
    - Assinantes e *third-party agents* implementam esta interface para receber notificações quando o método *notify* for invocado

Eventos Remotos

Classe RemoteEvent

- Classe que representa uma notificação (passado ao *remote event listener* no método *notify*).
- Possui as seguintes variáveis:
  - Referência ao gerador de eventos
  - Um ID de evento, especificando seu tipo
  - Um número de seqüência aplicado sobre os eventos daquele tipo.
    - A seqüência cresce a medida que eventos ocorrem
    - É usada para ordenar eventos no assinante
  - Um objeto *marshaled* já fornecido quando o assinante subscreveu para àquele tipo de evento
    - Geralmente, abriga qualquer informação necessária para o recipiente identificar o evento e reagir a sua ocorrência

Eventos Remotos

Interface Event Generator

- Fornece o método *register* usado para subscrever eventos no gerador de eventos
- **Os argumentos do método *register* especificam:**
  - Um identificador de evento (especifica seu tipo)
  - Um objeto montado para ser devolvido com cada notificação
  - Uma referência remota para um *event listener object*
  - Período de *lease* requisitado pelo assinante que especifica o tempo que o mesmo vai receber a notificação.
    - Porém, o período de *lease* real concedido é retornado como resultado do método *register*

Transações

- Transações são úteis para solucionar um grande problema em SDs – as falhas parciais.
- Agrupam um conjunto de operações para execução simultânea de modo que todas sejam executadas com sucesso ou que todas falhem
- Serviço Jini que utiliza transações – serviço de armazenamento *JavaSpaces*.

Transações

Componentes de uma Transação:

- Cliente da Transação
  Pede ao gerente para criar uma transação
- Gerente da Transação
  Cria uma transação e a gerencia enviando a cada participante uma mensagem indicando para qual estado esse participante deve mudar
- Participante da Transação
  Executa as operações em uma transação e interage com o gerente para completa-la corretamente

Serviço *JavaSpaces*

- Armazena objetos Java;
- Encontrado através do protocolo *lookup*;
- Interface *JavaSpace*;
- Suporta Transações.

Operações:

- *write*: armazena um objeto Java em um serviço *JavaSpaces;*
- *read*: procura um objeto armazenado
- *take*: remove um objeto do serviço
- *notify*: emite notificações de eventos para um objeto específico quando um novo objeto Java for armazenado no serviço

Profª. Ana Cristina B. Kochem
DAINF/CEFET-PR

Referências Bibliográficas

[ARNOLD 99] Arnold, Ken et al. The Jini Specification. 1999.