Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e informática
Unidade Acadêmica de Engenharia Elétrica

Trabalho de Conclusão de Curso

Protocolos para Configuração de Serviços em Ambientes Pervasivos

Ádrian Lívio Vasconcelos Guedes
adrian@compor.net

Orientador:
Angelo Perkusich
perkusich@dee.ufcg.edu.br

Campina Grande, Março de 2006

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e informática
Unidade Acadêmica de Engenharia Elétrica

Trabalho de Conclusão de Curso

Protocolos para Configuração de Serviços em Ambientes Pervasivos

Ádrian Lívio Vasconcelos Guedes
(Aluno)

Angelo Perkusich
(Orientador)

Campina Grande, Março de 2006

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

API - Application Programming Interface
DHCP - Dynamic Host Configuration Protocol
DNS - Domain Name Service
GENA - General Event Notification Architecture
HTTP - HyperText Transfer Protocol
HTTPMU -HTTP-over-Multicast
HTTPU - HTTP-over-UDP
IANA - Internet Assigned Numbers Authority
IETF - Internet Engineering Task Force
IP - Internet Protocol.
IPP - Internet Printing Protocol
JVM - Java Virtual Machine
mDNS - Multicast DNS
NAT - Network Address Translation
RMI - Remote Method Invocation
SCSL - Sun Community Source License
SLPv2 - Service Location Protocol, Version 2
SOAP - Simple Object Access Protocol
SSDP - Simple Service Discovery Protocol
TCP - Transport Control Protocol.
UDP - User Datagram Protocol
UPnP - Universal Plug and Play
URL - Universal Resource Locator
XML - Extensible Markup Language

Capítulo 1

Introdução

Em ambientes pervasivos as diversas aplicações e dispositivos precisam estabelecer comunicação entre si onde nem sempre existe uma infraestrutura física de rede definida de forma fixa. Nestes ambientes é preciso a adoção de protocolos que permitam o estabelecimento de uma comunicação entre os dispositivos, permitindo com que o dispositivo obtenha as configurações necessárias para entrar em contato com a rede, além de poder prover e utilizar serviços da rede.

Tecnologias como UPnP [6], Zeroconf [3] e Jini [5] provêm mecanismos necessários para que estes dispositivos funcionem, como configuração de rede e detecção de dispositivos e serviços

O UPnP é uma arquitetura de rede aberta e distribuída passível de integração com redes TCP/IP e outras tecnologias de rede, permitindo conectividade peer-to-peer entre Aplicações Inteligentes, Dispositivos Wireless, PCs, etc.

Zeroconf é um conjunto de técnicas que automaticamente cria uma rede IP sem configuração ou servidores especiais.

Jini é uma arquitetura que permite que vários serviços (software ou dispositivos) comuniquem-se sem configurações por parte de um administrador.

Neste trabalho inicialmente é feita uma apresentação dos três protocolos supracitados mostrando suas características e especificações. Posteriormente como

estudo de caso é apresentada a arquitetura de uma implementação simples de um dispositivo UPnP utilizando o modelo de componentes COMPOR e a biblioteca CyberLink.

Capítulo 2

Zeroconf

2.1 Introdução

Zeroconf ou Zero Configuration Network [3] é um conjunto de técnicas que permitem a criação de uma rede IP de forma automática sem a configuração de servidores específicos. Isto permite que usuários sem conhecimento técnico conectem computadores, impressoras e outros dispositivos em rede de forma que estes possam ser utilizados.

2.2 Especificação

A especificação do padrão é feita pelo IETF Zeroconf Working Group criado em setembro 1999. Em maio 2002 uma versão de Zeroconf foi lançada pela Apple com o nome comercial de Rendezvous, em agosto do mesmo ano é lançada uma versão para Windows, e em abril de 2005 Apple anuncia "Rendezvous 2"com um novo nome comercial, Bonjour. Bonjour inclui APIs para Java, além de uma versão completa para Windows.

O IETF apresenta como requisitos para o Zeroconf:

- Alocação automática de endereços sem a presença de um servidor DHCP

(Link-Local Addressing).

- Tradução entre nomes e IP sem a presença de um servidor DNS (Multicast DNS)
- Localizar serviços, como impressoras, sem um servidor de diretorio (DNS Service Discovery)

Um outro requerimento é que as soluções acima devem coexistir com redes configuradas. Os protocolos Zeroconf não podem causar dano à rede quando uma máquina Zeroconf é conectada a uma rede maior.

2.2.1 Alocação automática de endereços

Em uma rede IP cada ponto de comunicação seja ele fonte ou destino de uma mensagem deve possuir um endereço IP único no escopo em que este endereço será usado. A autoconfiguração de endereços permite a um host

- Configurar sua interface com um endereço único;
- Determinar qual submáscara de rede utilizar;
- Determinar se houve alocação duplicada de endereços;
- Lidar com colisões.

Quando um novo dispositivo é adicionado na rede, Zeroconf configura o dispositivo utilizando a técnica chamada de link-local addressing, se um servidor DHCP estiver disponível Zeroconf irá utilizar o IP fornecido pelo DHCP.

Com a técnica link-local addressing o dispositivo escolhe um IP de forma aleatória, dentro da faixa de valores 196.254.xxx.xxx determinada pela IANA (Internet Assigned Numbers Authority). O dispositivo então envia uma mensagem para a rede de forma a determinar se o endereço já esta em uso, se o

endereço estiver em uso o dispositivo aleatoriamente escolhe outro IP na mesma faixa, repetindo o processo até que encontre um endereço livre. Após conseguir um IP o dispositivo já está pronto para enviar e receber tráfego IP através da rede.

2.2.2 Tradução Nome-Endereço

Uma aplicação IP tipicamente identifica os dispositivos da rede por nome ao invés de endereço, isto provê uma maior estabilidade operacional pois no caso de uma mudança de endereço o nome permanece o mesmo. O Zeroconf fornece mecanismos para:

- Obter o endereço IP associado com um nome.
- Determinar o nome associado com um endereço IP.

Isto pode ser feito através de servidores DNS na internet e em grandes redes com gerência os administradores mantêm estes servidores, em redes onde não existe DNS Zeroconf se utiliza de um variante do DNS chamado multicast DNS (mDNS). Uma notificação mDNS pergunta e recebe o tipo do serviço (impressora IPP) o nome do serviço (como "Printing"), IP e endereço da porta além de outras informações opcionais. Cada dispositivo da rede recebe uma notificação e armazena a informação. Aplicativos rodando nos dispositivos podem usar estas informações para criar uma lista de serviços e os oferecer ao usuário.

2.2.3 Localização de serviços

A IETF estabeleceu como padrões dois mecanismos de descoberta de serviços sobre redes IP. O primeiro SLPv2 (service location protocol version 2) que provê a descoberta de serviços tanto pelo tipo de serviço como pelos atributos, então um cliente pode localizar um determinado serviço especificando as características

desejadas. Na figura 2.1 é ilustrada a localização de serviços através de SLPv2, onde um aplicativo cliente requisita a localização do serviço "Bar"e os agentes SLPv2 respondem, anunciando os serviços que se enquadram a requisição.

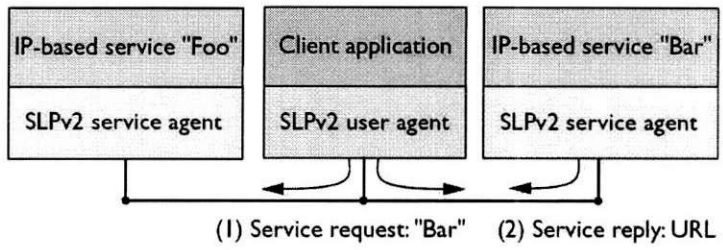

Figura 2.1: Localização de serviços através de SLPv2.

O segundo mecanismo é o DNS SRV Resource Record, que permite clientes procurarem serviços via DNS. Os clientes especificam o tipo de serviço, o protocolo de transporte e domínio no qual procurar. A resposta contem uma lista de hosts que satisfazem a requisição. Uma requisição por dispositivos que sejam impressoras IPP pode tem a forma.

- `_ipp._tcp.local.PTR?`

Enquanto que a lista de respostas pode ser

- `Marketing._ipp._tcp.local.`
- `Engineering._ipp._tcp.local.`
- `Sales._ipp._tcp.local.`
- `CopyRoom._ipp._tcp.local.`

Onde a estrutura destas mensagens é apresentada na figura 2.2

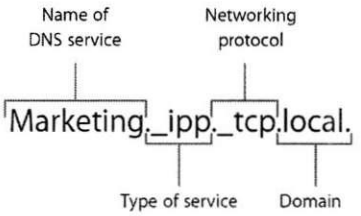

Figura 2.2: Estrutura de uma mensagem DNS SRV Resource Record

Isto é, temos que o serviço Marketing possui um serviço de impressão do tipo IPP utilizando o protocolo TCP no domínio local.

Capítulo 3

Jini

3.1 Introdução

A tecnologia de rede Jini [5] é uma arquitetura aberta que permite a os desenvolvedores em Java criar serviços centrados em rede facilmente adaptados a mudanças. Jini pode ser usada para criar sistemas com escalabilidade e flexibilidade.

Jini foi originalmente criada pela Sun Microsystems, tendo contribuições da Jini Community desde 1999 [4]. O código fonte de Jini é disponível sob a Sun Community Source License (SCSL).

3.2 Especificação

Jini é um sistema distribuído baseado na idéia de grupos federados de usuários e de recursos requisitados por estes usuários. O objetivo geral é transformar a rede em uma ferramenta flexível e facilmente administrável, onde os recursos possam ser encontrados pelos clientes. Estes recursos podem ser implementados tanto por dispositivos de hardware como por programas ou uma combinação de ambos.

Jini é construída sobre Java de forma que os objetos Java posam circular pela rede, estendendo os benefícios da programação orientada a objetos a rede. Desta

forma podem interagir entre si através de interfaces bem definidas.

A especificação da tecnologia Jini [11] descreve:

- Mecanismos de descobertas de serviços em tempo de execução, permitindo aos clientes localizar e conectar-se a serviços específicos.
- Um modelo de programação que estende o modelo de programação da linguagem Java de forma a acomodar as características específicas da rede, como por exemplo uma falha parcial.
- Serviços chave como memória compartilhada e serviços de gerenciamento de transações, que fornecem importantes funcionalidades a serem disponibilizadas em um ambiente distribuído.

Como pré requisito Jini exige de cada dispositivo:

- Uma máquina virtual Java (JVM), com acesso a todos os pacotes necessários para executar um software escrito de acordo com as especificações Jini.
- Uma pilha de protocolos de rede propiamente configurada.

Jini define uma infraestrutura de execução residente na rede que provê mecanismos que permitem a adição, remoção e acesso a serviços. Esta infraestrutura de execução reside na rede em três pontos:

- Nos *lookup services* situados na rede.
- Nos provedores de serviços, como por exemplo dispositivos que possuam Jini habilitado.
- Nos clientes

Os *lookups services* são o mecanismo central de organização nos sistemas Jini, quando novos serviços são disponibilizados na rede eles se registram através de um *lookup service* e quando um cliente deseja localizar um serviço ele utiliza um *lookup service.*

A infraestrutura de execução usa um protocolo a nível de rede chamado *discovery* e dois protocolos a nível de objeto chamados de *join* e *lookup. Discovery* permite aos clientes e serviços localizar os *lookup services. Join* permite ao serviço registrar-se em um *lookup service. Lookup* permite a um cliente requisitar serviços a um *lookup service.*

Uma impressora recém conectada a rede, por exemplo precisa descobrir o *lookup service* apropriado através do *discovery* então registrar-se através do *join.* Ao registrar-se a a impressora recebe um *lease* (escritura de aluguel) e este deve ser periodicamente renovado, este processo permite que o *lookup service* perceba que a impressora foi retirada da rede e retire o serviço por ela provido.

Um cliente que deseje fazer uso do serviço de impressão precisa, também, se registrar em um *lookup service* e então pode fazer um *lookup* para localizar um serviço de impressão e receber uma instância da classe que representa o serviço, neste momento o *lookup service* deixa de participar do processo, o cliente então pode invocar, por exemplo, o método *print* da classe que representa o serviço de impressão, passando como argumento a imagem que deseja imprimir.

As chamadas aos métodos são feitas através de RMI (Remote Method Invocation) que permite a um Objeto fazer uma chamada a uma interface remota de um objeto remoto.

Capítulo 4

UPnP: Universal Plug and Play

4.1 Introdução

UPnP é uma arquitetura de rede aberta e distribuída, passível de integração com redes TCP/IP e outras tecnologias de rede. UPnP é uma arquitetura que permite a conectividade peer-to-peer entre aplicações inteligentes,dispositivos wireless e PCs.

A tecnologia UPnP tem como objetivos:

- Prover uma conectividade entre redes ad-hoc, independente das suas arquiteturas, de uso fácil e flexível.
- Prover a transferência de dados entre redes de diferentes arquiteturas.
- Substituição dos Drivers de dispositivos por protocolos de uso comum.
- Ser uma rede independente de mídias, sistemas operacionais e de linguagem de programação.

Os dispositivos UPnP apresentam como principais características:

- Suporte a configuração zero.

- "Rede invisível", isto é, a rede é enxergada de forma transparente para os dispositivos.
- Descoberta automática de uma ampla categoria de dispositivos dos mais diversos fabricantes.
- Um dispositivo UPnP tem a capacidade de unir-se a uma dada rede dinamicamente, obtém endereço IP, apresentando seus serviços e detectando a presença e as capacidades de outros dispositivos.
- Servidores de DHCP e DNS são opcionais, usados apenas se disponíveis.
- Capacidade de sair da rede sem deixar nenhuma inconsistência na mesma.

4.2 Especificação

A arquitetura dos dispositivos UPnP [10] define os protocolos para comunicação entre controladores, ou pontos de controle e dispositivos. Para descoberta, descrição, controle, tratamento de eventos e apresentação, o UPnP usa a pilha mostrada na figura 4.1.

O serviço de rede UPnP consiste dos seguintes passos:

0. Endereçamento
1. Descoberta
2. Descrição
3. Controle
4. Tratamento de Eventos
5. Apresentação

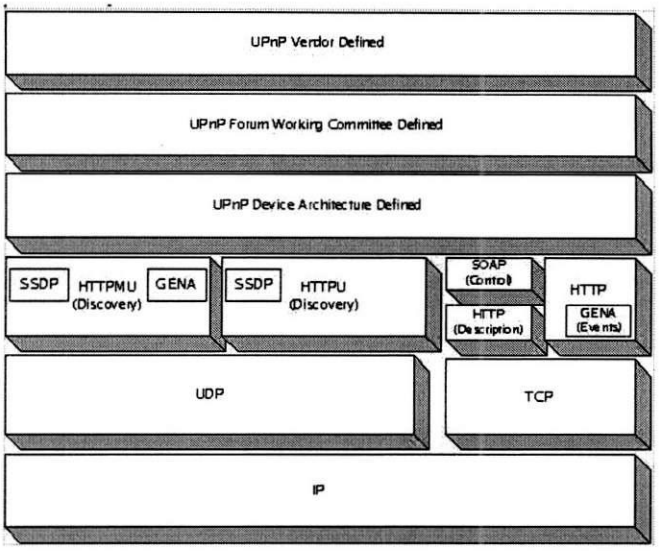

Figura 4.1: Pilha de Protocolos utilizada no UPnP.

4.2.1 Endereçamento

A base do serviço de rede UPnP é o endereçamento IP. Cada dispositivo deve ter um cliente DHCP e procurar por um servidor DHCP quando se conecta pela primeira vez na rede. Se o servidor estiver disponível o dispositivo deve usar o endereço IP fornecido. Caso contrário o dispositivo deve usar a técnica chamada de Auto IP para obter um endereço.

O Auto IP define como um dispositivo, de forma inteligente, escolhe um IP de um conjunto de endereços reservados, tornando o dispositivo capaz de se mover facilmente entre redes com ou sem gerenciamento (DHCP). Se durante a negociação com o DHCP o dispositivo recebe um nome de domínio (DNS), o dispositivo deve usar este nome em sua operações.

4.2.2 Descoberta

A descoberta é o primeiro passo propriamente dito em um sistema UPnP. Quando um dispositivo é adicionado à rede, o protocolo de descoberta do UPnP permite

que o dispositivo anuncie seus serviços aos pontos de controle na rede. De forma semelhante, quando um ponto de controle é adicionado à rede, o protocolo de descoberta do UPnP libera o ponto de controle para procurar por dispositivos de interesse na rede.

Em ambos os casos o ponto fundamental é a mensagem de descoberta, esta contém um pequeno conjunto de parâmetros essenciais sobre o dispositivo, como:

- Tipo;
- Identificador;
- Um apontador para detalhes;

Quando o novo dispositivo é adicionado na rede (*root device 1* na figura 4.2), ele faz multicast de mensagens de descoberta anunciando (*advertise*) dispositivos e serviços que contém. Qualquer ponto de controle interessado pode esperar por mensagens no endereço de multicast padrão, estam mensagens informam sobre novas capacidades adicionadas a rede. De forma semelhante, quando um novo ponto de controle é adicionado a rede (*control point 3* na figura 4.2) ele envia mensagens em multicast em busca dos dispositivos e serviços de seu interesse.

Todos os serviços devem esperar por mensagens vindas do endereço de multicast padrão, e devem responder caso algum de seus dispositivos ou serviços correspondem ao critério de busca da mensagem (*unicast responses* do *root device 2* na figura 4.2).

Quando um dispositivo é removido da rede ele deve novamente enviar mensagens em multicast informando que aquele dispositivo está sendo retirado da rede e que ele e seus serviços não estarão mais disponíveis.

O endereço de multicast padrão, assim como os mecanismos de anuncio, busca e remoção de dispositivos são definidos pelo *Simple Service Discovery Protocol* (SSDP). O SSDP é usado sobre HTTPMU, responsável pelo multicast sobre UDP e HTTPU responsável por unicast sobre UDP.

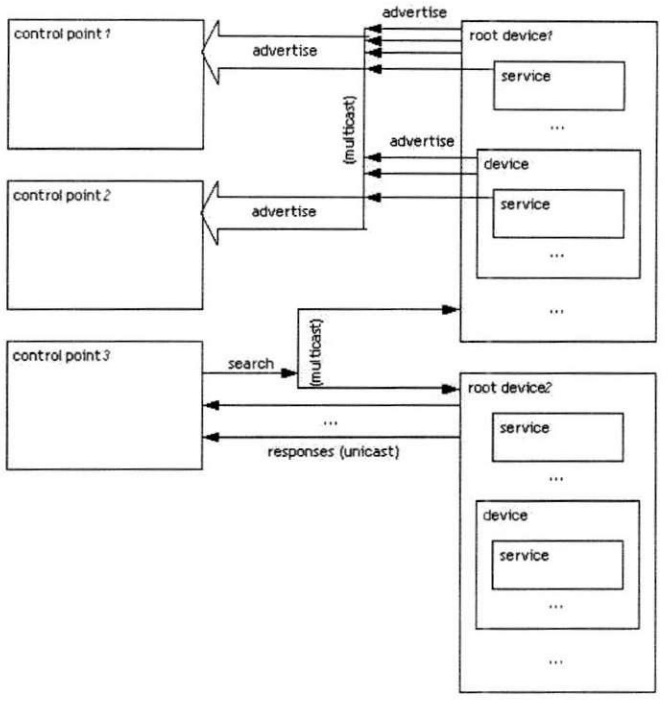

Figura 4.2: Etapa de Descoberta.

4.2.3 Descrição

No momento que um ponto de controle descobre um dispositivo ele ainda sabe muito pouco sobre ele, para que isto seja sanado, deve haver uma requisição da descrição através da URL passada ao ponto de controle no passo anterior, como mostrado na figura 4.3.

Um dispositivo UPnP pode conter outros dispositivos lógicos ou serviços. Estas informações estão contidas na descrição do dispositivo. A descrição UPnP para um dispositivo é expressa em XML e é composta de:

- Especificações do fabricante como modelo, número de série, nome e URL do fabricante
- Uma lista de todos dispositivos embutidos ou serviços.
- URLs para controle tratamento de eventos e apresentação.

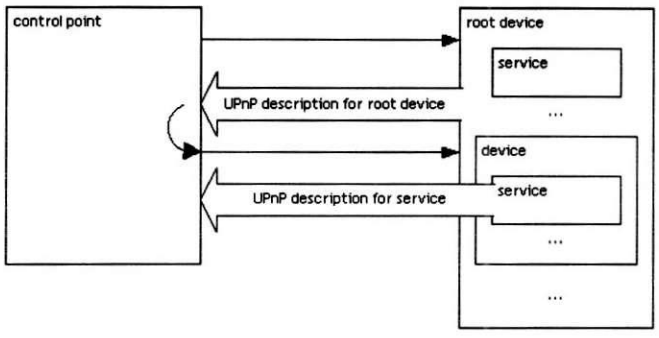

Figura 4.3: Etapa de Descrição

Após receber a descrição do dispositivo o ponto de controle pode fazer uma nova requisição para cada serviço ou dispositivo embutido (4.3). Para cada serviço, a descrição inclui uma lista de comandos ou ações, parâmetros ou argumentos e variáveis.

4.2.4 Controle

Após a etapa de descrição, o ponto de controle pode enviar ações à um serviço de um dispositivo remoto. Para enviar uma ação, o ponto de controle envia uma mensagem de controle ao endereço contido na URL de controle previamente adquirida (*action request* na figura 4.4). Uma mensagem de controle é enviada em XML, o protocolo usado é o SOAP (*Simple Object Access Protocol*).

Como resposta a uma mensagem de controle o serviço pode retornar algum resultado ou um erro. Uma chama a um serviço também pode provocar uma alteração nas variáveis de estado do serviço, uma alteração no valor destas variáveis provoca a emissão de eventos para todos os pontos de controles interessados, como mostrado na seção 4.2.5.

Caso um ponto de controle deseje determinar o estado de alguma variável de estado ele pode requisitar o valor desta variável, o serviço estão fornece o valor da variável (*query variable* na figura 4.4).

Enquanto a publicação feita na descoberta ainda estiver válida o ponto de

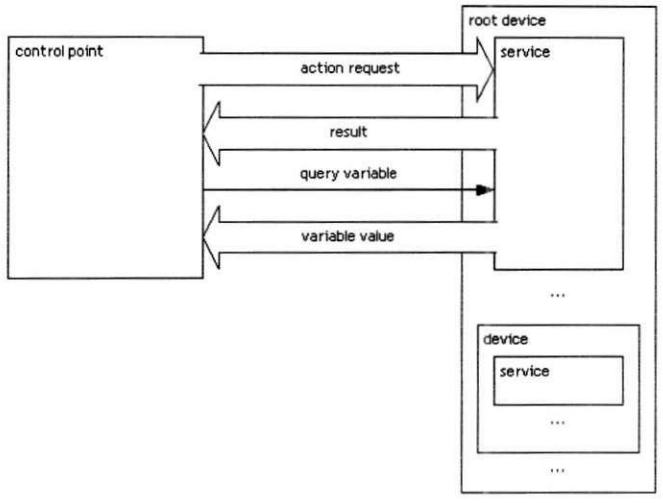

Figura 4.4: Etapa de Controle.

controle pode assumir que o dispositivos e seus serviços ainda estão disponíveis.

4.2.5 Tratamento de Eventos

Após um ponto de controle ter descoberto um dispositivo (seção 4.2.2) e recebido a sua descrição (seção 4.2.3), o ponto de controle possui as informações essenciais para o tratamento de eventos. A descrição UPnP para um serviço inclui:

- Uma lista das ações.
- Uma lista das variáveis que modelam o serviço em tempo de execução.

Uma mensagem especial é enviada assim que o ponto de controle se une à rede. Esta mensagem contém os nomes e valores de todas as variáveis que podem ser tratadas, permitindo a inicialização do modelo de estado do serviço. Em uma rede com múltiplos pontos de controle, o tratamento de eventos é destinado a manter todos os pontos de controle igualmente informados sobre os efeitos de qualquer ação.

O serviço publica, via mensagens, as atualizações quando as variáveis são alteradas. Qualquer ponto de controle pode observar esta mudança. Estas men-

sagens contém o nome e o valor atual da variável ou variáveis. As mensagens são expressas em XML e formatadas pelo GENA (*General Event Notification Architecture*).

Para poder observar os eventos um ponto de controle pode "assinar"os eventos de um deterninado serviço, isto é, o ponto de controle se tornará um observador do estado daquele serviço e sempre que uma alteração numa variável de estado do serviço ocorrer o ponto de controle será informado sobre essa mudança. Isto é feito através de uma mensagem (*subscription message* na figura 4.5).

Periodicamente a requisição deve ser renovada (*renewal request* na figura 4.5) de forma que caso o ponto se controle ou o serviço venha a ser retirado de forma inesperada da rede, desta forma a rede consegue identificar que aquele componente não está mais disponível.

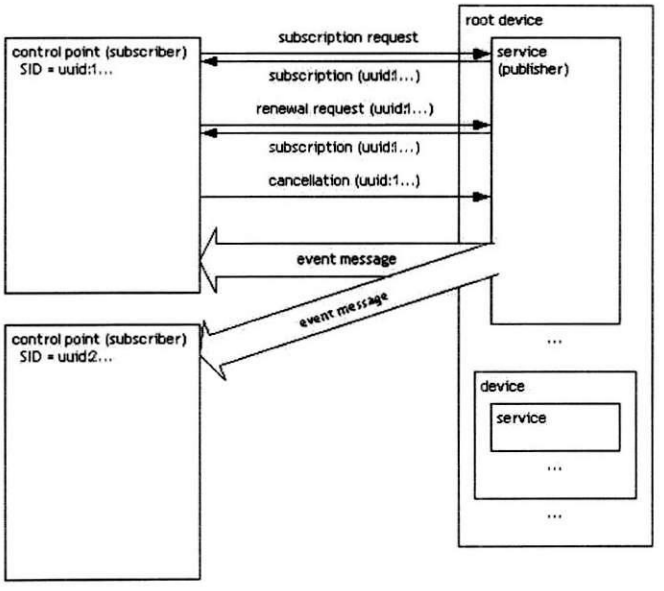

Figura 4.5: Etapa de Tratamento de Eventos.

No momento em que o ponto de controle não necessita mais dos dados de um determinado serviço ele pode mandar uma mensagem de cancelamento (*cancellation* na figura 4.5)), desta forma o ponto de controle não irá mais receber eventos

do serviço.

4.2.6 Apresentação

Um dispositivo pode ter uma URL para apresentação. Caso tenha, o ponto de controle pode carregar esta URL em um browser (figura 4.6) e dependendo das capacidades da página, ela permite ao usuário controlar o dispositivo e/ou ver o status do mesmo. O grau de controle permitido depende das capacidades especificas da página de apresentação e do dispositivo.

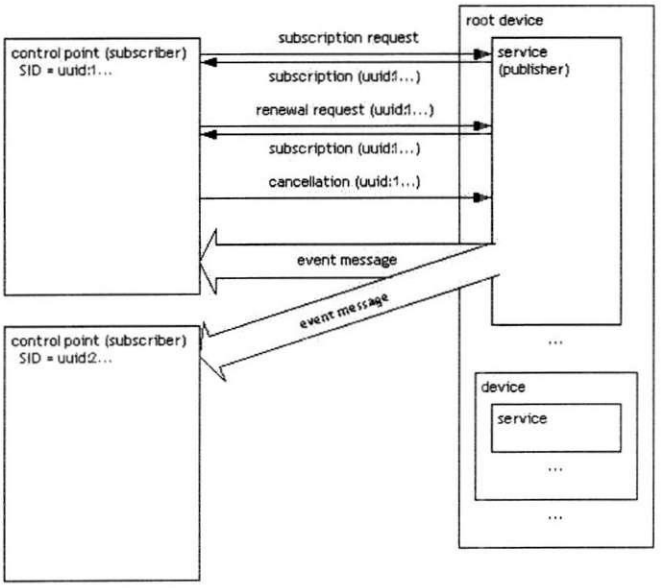

Figura 4.6: Etapa de Apresentação.

Capítulo 5

Implementação de um Aplicativo UPnP

5.1 Introdução

Como um estudo de caso foi feita uma implementação de uma aplicação UPnP simples. A aplicação consiste na implementação de um dispositivo UPnP de acordo com o *Binary Light Device Template* [9] definido pelo Fórum UPnP [6].

A aplicação usa como base a biblioteca *CyberLink* voltada para desenvolvedores de aplicativos UPnP [2] e o modelo de desenvolvimento baseado em componentes COMPOR [1].

5.2 CyberLink

CyberLink é um pacote para o desenvolvimento de aplicações UPnP em C++, Java, C ou Pearl. CyberLink controla automaticamente todos os protocolos que rodam por trás da arquitetura UPnP (HTTP, SOAP, GENA, SSDP e HTTPU).

O CyberLink é um arcabouço que permite ao programador implementar suas aplicações sem se preocupar com os protocolos que compõem o UPnP, dando

suporte para o programador criar pontos de controle e dispositivos UPnP.

O guia de programação do CyberLink [8] provê as informações necessárias para a implementação de dispositivos e pontos de controle.

5.2.1 Implementação de um Dispositivo UPnP

Um diagrama de classes simplificado de um dispositivo UPnP segundo a biblioteca CyberLink é apresentado na figura 5.1.

Observando o diagrama temos que um dispositivo pode conter serviços e outros dispositivos. Os serviços possuem Ações e Variáveis de Estado.

A etapa de descrição de um dispositivo é feita por vários arquivos XML então antes de programar o dispositivo, é preciso criar vários arquivos XML com a descrição e os serviços do mesmo. Na URL de descrição deve-se haver apontadores para estes arquivos. Os arquivos devem englobar: serviços, ícone e apresentação do dispositivo como mostrado na figura 5.2.

A próxima etapa é a criação do dispositivo propiamente dito isto é feito quando se faz uma instanciação da classe `Device` esta é associada à URL de descrição previamente criada. Este dispositivo é do tipo raiz. Quando ele é inicializado através do método `Device.start()`, é imediatamente avisado à toda rede UPnP.

Uma vez criado, um dispositivo raiz retorna automaticamente todas as requisições feitas por pontos de controle. As configurações padrão são:

Parâmetro	Padrão	Método	Detalhes
HTTP Port	4004	`setHTTPPort()`	O servidor http usa a porta no dispositivo raiz.
Description URI	/description.xml	`setDescriptionURI()`	A URL de descrição do dispositivo raiz.
Lease time	1800	`setLeaseTime()`	O *lease time* do dispositivo raiz.

Tabela 5.1: Configurações padrão para um dispositivo raiz.

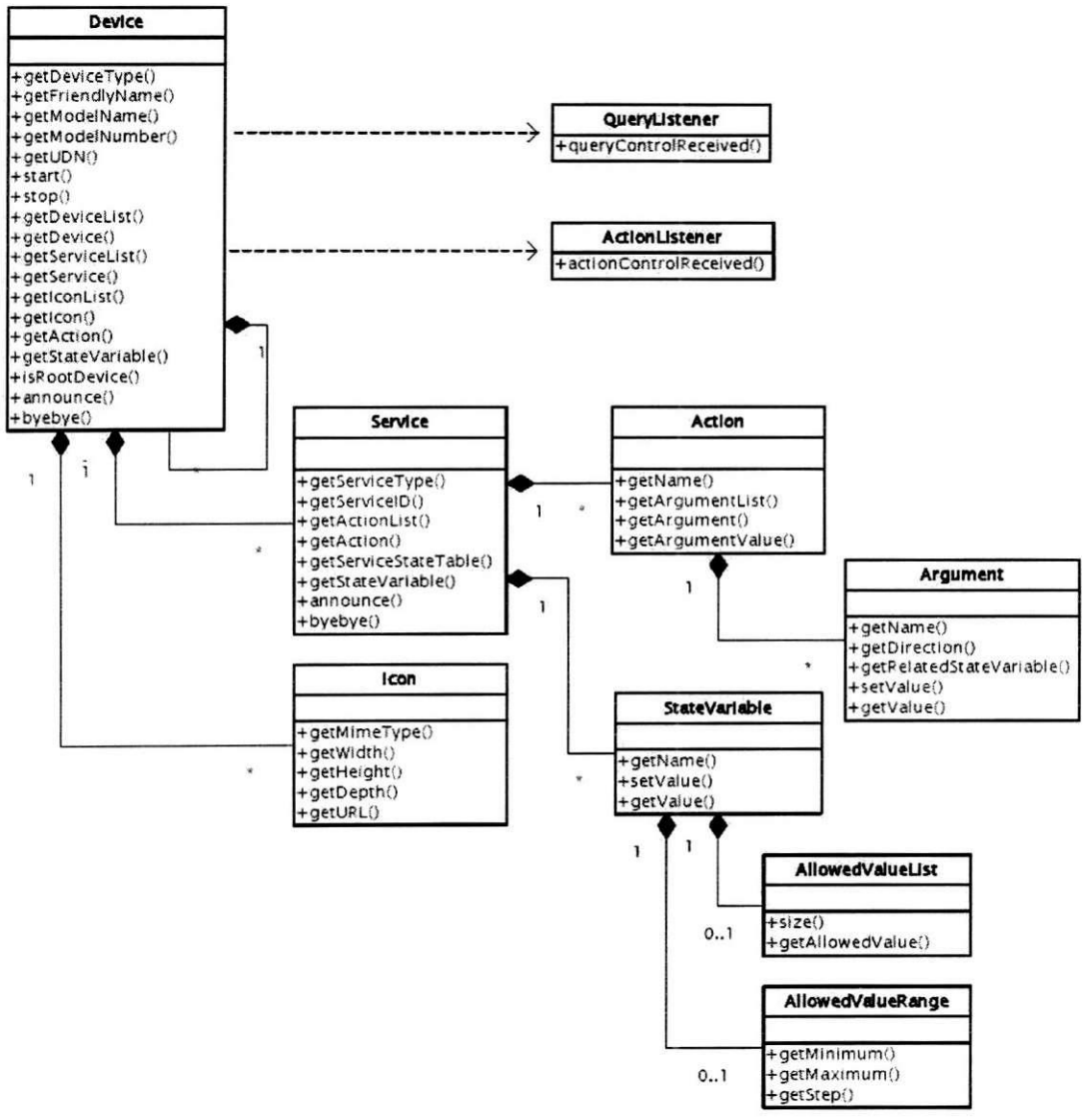

Figura 5.1: Diagrama de classes de um dispositivo UPnP.

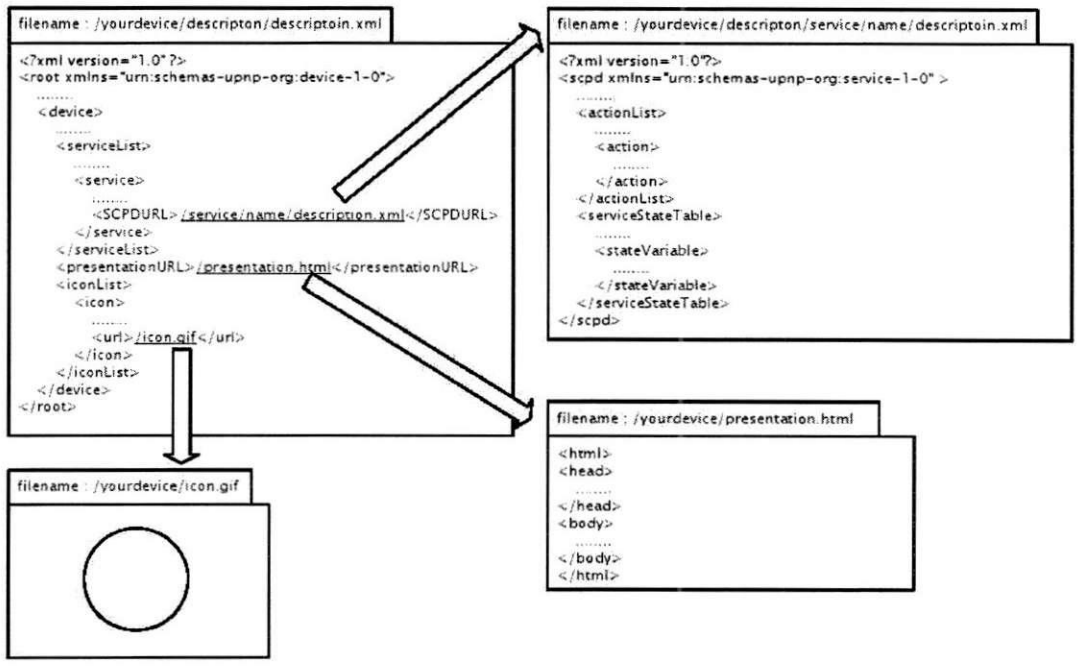

Figura 5.2: Arquivos de descrição de um dispositivo UPnP.

O dispositivo é anunciado a toda rede, no momento da inicialização, `Device.start()`, postando uma mensagem de notificação com `ssdp::alive`. Quando o dispositivo é desativado, `Device.stop()`, posta-se uma mensagem de notificação com `ssdp::byebye`.

O dispositivo repete sua mensagem de notificação no *lease time* automaticamente. Pode-se anunciar o dispositivo usando `Device.announce()` ou `Device.byebye()`.

Um dispositivo raiz pode ter vários dispositivos embarcados que podem ser acessados por `Device.getDeviceList()` que retorna uma lista de todos os dispositivos embarcados naquele dispositivo raiz.

Uma vez que um dispositivo, seja ele um dispositivo raiz ou embarcado, esteja disponível é possível acessar seus serviços através do método `Device.getServiceList()`.E a partir de um serviço é possível descobrir seu estado, e que ações este pode realizar com os métodos `Service.getServiceStateTable()` e `Service.get ActionList()`.

Para que um dispositivo raiz possa aceitar eventos de controle, de requisições

de ações, estes têm que:

- implementar a interface *ActionListener*;
- implementar o método `actionControlReceived()`;
- implementar a interface *QueryListener*;
- implementar o método `queryControlReceived()`;

Estes métodos são chamados pelo CyberLink quando o dispositivo recebe uma requisição de ação ou evento.

Um dispositivo pode mudar suas variáveis de estado, se for necessário que um dispositivo envie para um ponto de controle o valor de uma variável de estado do dispositivo usa-se `ServiceStateVariable.setValue()`, o evento é então enviado automaticamente para os outros pontos de controle.

5.3 COMPOR

O COMPOR é um modelo de componente para composição dinâmica de aplicações. A arquitetura de componentes do modelo COMPOR possui dois tipos de entidades [7]: contêineres e componentes funcionais. Os componentes funcionais implementam as funcionalidades do sistema, disponibilizando-as em forma de serviços. Os componentes funcionais não são compostos por outros componentes, ou seja, não possuem "componentes-filhos". Os contêineres, por sua vez, não implementam funcionalidades, apenas gerenciam o acesso aos serviços dos seus "componentes-filhos". No modelo não existem referências diretas entre os provedores de funcionalidades, desta forma eles podem ser adicionados e removidos de maneira mais flexível.

5.4 Implementação

A aplicação a ser implementada basicamente consiste na criação de um componente funcional que publique determinados serviços providos por um outro componente, na forma de serviços em uma rede UPnP.

5.4.1 LampDeviceComponent

O primeiro componente a ser criado é `LampDeviceComponent` que proverá os serviços e as variáveis de estado do componente a serem publicadas pelo `UPnPPublisherComponent`. O `LampDeviceComponent` será um `FunctionalComponent` e o serviço e a variável de estado serão então os dois `ProvidedServices` `Switch` e `LampState` respectivamente, como mostrado na figura 5.3.

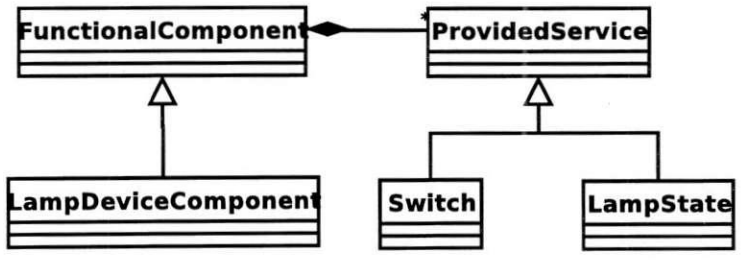

Figura 5.3: Diagrama de classes do `LampDeviceComponent`.

Quando o serviço `Switch` é invocado o estado da lâmpada mude de "aceso" para "apagado" ou visse-versa, dependendo do estado atual da lâmpada. O serviço `LampState` retorna o estado atual da lâmpada de forma que este possa ser publicado na rede UPnP pelo `UPnPPublisherComponent`.

5.4.2 UPnPPublisherComponent

O `UPnPPublisherComponent` é responsável por, a partir dos `ProvidedServices` fornecidos pelo `LampDeviceComponent` gerar e tornar disponível para uma rede UPnP um dispositivo UPnP que represente a lâmpada modelada pelo `LampDeviceComponent`.

A figura 5.4 mostra o diagrama de classes para o `UPnPPublisherComponent`.

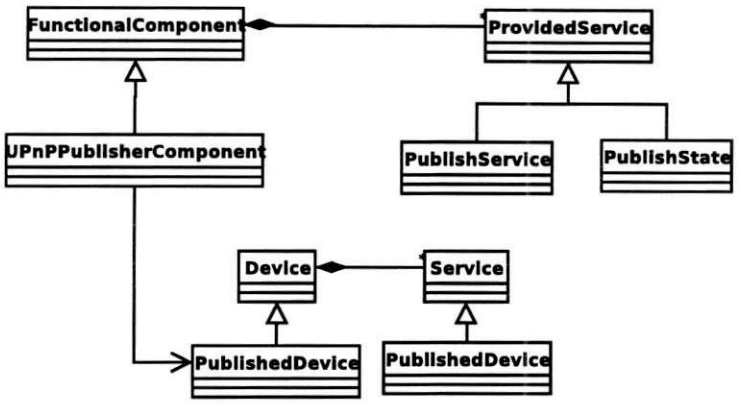

Figura 5.4: Diagrama de classes do `UPnPPublisherComponent`.

Quando os serviços `PublishService` e `PublishState` são chamados o `UPnPPublisherComponent` cria uma instância da classe `Device` da biblioteca *CyberLink* e atribui a seu serviço a funcionalidade fornecida por `PublishService` e adiciona a variável de estado obtida em `PublishState`.

A partir daí o `UPnPPublisherComponent` cria as descrições para o dispositivo e os serviços criados, além de uma URL de apresentação simples. Do ponto de vista da rede UPnP um novo dispositivo é adicionado, possuindo as funcionalidades providas pelo `LampDeviceComponent`.

Capítulo 6

Conclusão

Os protocolos mostrados neste trabalho apresentam diferente soluções para o problema de configuração automática de serviços em ambientes onde nem sempre a estrutura da rede é fixa, podendo variar com o tempo.

Jini e UPnP modelam os serviços e dispositivos de maneira bastante semelhante, usando a idéia de objetos. Jini porém assume que estes objetos sejam objetos Java, desta forma atrelando o sistema a existência de máquinas virtuais Java (JVM). UPnP por outro lado apenas define um protocolo comum, que pode ser implementado em qualquer linguagem, o que fornece maior flexibilidade, temos como exemplo a biblioteca *CyberLink* que possui implementações em Java, C, C++ e Pearl.

Zeroconf e UPnP possuem algumas características semelhantes em termos de configuração de rede, obtendo um IP de maneira automática quando o dispositivo é adicionado na rede, ao contrário de Jini que presume a existência de um mecanismo de configuração de rede já definido, como DHCP por exemplo. Zeroconf e UPnP diferem porém no fato de Zeroconf ser focado em serviços e UPnP em dispositivos.

O UPnP também tem como característica a possibilidade de apresentação de seus serviços na forma de uma página de apresentação, permitindo que um

dispositivo possa ser controlado diretamente de um browser.

A associação com o COMPOR mostrada na implementação faz com que um dispositivo UPnP se beneficie da arquitetura de composição dinâmica do modelo, permitindo que uma aplicação anteriormente escrita segundo o modelo de componentes possa ser utilizada em uma rede UPnP bastando que para isso seja adicionado um componente a aplicação.

Bibliografia

[1] *COMPOR - Software Composition.* http://www.compor.net.

[2] *CyberLink Development Package for UPnP Devices.* www.cybergarage.org.

[3] *IETF Zeroconf Working Group.* http://www.zeroconf.org.

[4] *Jini Community.* http://www.jini.org.

[5] *Jini Network Technology.* http://www.sun.com/software/jini.

[6] *UPnP Forum.* http://www.upnp.org.

[7] E. de Barros Costa H. O. de Almeida, A. Perkusich. Composição dinâmica de componentes para aplicações com mudanças freqüentes de requisitos. In *4º Workshop de Desenvolvimento Baseado em Componentes, João Pessoa, Brasil*, 2004.

[8] Satoshi Konno. *CyberLink for Java Programming Guide*, version 1.7 edition, 2005.

[9] Christoph Sahm; Hans J. Langels. *BinaryLight:1 Device Template.* UPnP Forum, http://www.upnp.org/standardizeddcps, version 1.0 edition, November 2003.

[10] Contributing Members of the UPnP Forum. *UPnP Device Architecture.* http://www.upnp.org/download/UPnPDA10_20000613.htm, version 1.0 edition, Jun 2000.

[11] Sun Microsystems, Inc. *Jini(TM) Architecture Specification*, 2003.