Instituto de Computação

PGC – Pós-Graduação em Computação

MONOGRAFIA DE

CONCEPÇÃO E PROJETO DE SISTEMAS DISTRIBUÍDOS

Análise de Tecnologias para Computação Pervasiva

MARCIO BELO RODRIGUES DA SILVA
mbelo@ic.uff.br

PROFESSOR: Dr. ORLANDO LOQUES

Niterói, RJ – Brasil
2003

Análise de Tecnologias para Computação Pervasiva

Marcio Belo Rodrigues da Silva

Universidade Federal Fluminense
Instituto de Computação
Departamento de Pós-Graduação
Niterói, Brasil

mbelo@ic.uff.br

Introdução

A crescente demanda pela integração de dispositivos, sejam eles desde complexos PCs até simples equipamentos eletrônicos, tanto no ambiente empresarial quanto no doméstico, traz à tona preocupações com a infra-estrutura que dará suporte a esse cenário de maneira eficiente.

Atualmente, esse cenário conectado é perfeitamente tangível, porém demanda um grande esforço de instalação e configuração, o que o torna inviável para um usuário leigo. O que se deseja num cenário futuro é tornar as facilidades eletrônicas cada vez mais disponíveis e integradas, de forma a colaborarem para o enriquecimento da experiência humana com o mundo computacional, de forma transparente e segura. Este ambiente computacional pretendido tem sido atualmente classificado como pervasivo [5].

Um ambiente computacional pervasivo pode ser definido como aquele no qual a presença de dispositivos inteligentes não é notada pelo usuário [8], por mais complexas que sejam as operações que estão sendo feitas para realizar uma determinada tarefa solicitada por ele.

O objetivo deste trabalho é identificar os principais requisitos, analisar as principais tecnologias de infra-estrutura disponíveis e demonstrar o uso prático de uma dessas tecnologias que poderão tornar a computação pervasiva uma realidade.

Na seção *Necessidades de um Ambiente Pervasivo* serão listadas as principais necessidades introduzidas pela computação pervasiva. A *seção Tecnologias Atuais* realiza uma análise das principais tecnologias disponíveis no mercado que podem ser úteis para a composição da infra-estrutura de suporte necessária. Na seqüência, a seção *A opção por UPnP* explica os motivos que fazem o autor apontar essa tecnologia como a mais adequada ao ambiente pervasivo. Como forma de demonstrar um ambiente pervasivo em ação, a seção *Projeto Prático* apresenta um estudo de caso usando um kit SDK UPnP. A seção *O Problema da Segurança* debate aspectos relativos ao tema e propõe de forma geral uma implementação de segurança na arquitetura UPnP. Finalmente, a seção *Conclusão* fecha o artigo apontando expectativas futuras com relação ao cenário pervasivo.

Necessidades de um Ambiente Pervasivo

- Conectividade

Uma das principais necessidades num ambiente pervasivo é a possibilidade de conectar dispositivos, dentro e fora de ambientes fechados. Neste aspecto, a grande preocupação é convencionar padrões de rede, de forma a facilitar o estabelecimento da comunicação entre dispositivos. Vários padrões despontam como os principais atores no cenário pervasivo e, provavelmente, vários deles prevalecerão, desde que não restrinjam a interoperabilidade entre si. Exemplo de tecnologias: Ethernet, 802.11, Bluetooth, etc.

- Padrão unificado para descoberta, configuração e controle de dispositivos

Como uma das características básicas de uma ambiente pervasivo é a utilização por pessoas sem conhecimentos técnicos, devem ser utilizadas tecnologias que permitam um alto grau de autoconfiguração, sem a necessidade de intervenção humana. Isso se refere, por exemplo, à habilidade de um dispositivo descobrir a presença de outros dispositivos e serviços na rede, e identificar suas funcionalidades.

- Padronização de formatos de mídia e protocolos de "streaming"

Uma das demandas que a integração de dispositivos trará para o ambiente a ser utilizado é a necessidade de utilizar padrões conhecidos de formato de mídia uma vez que a mera capacidade de trocar informações entre dispositivos não é suficiente. Por exemplo, de nada adiantaria um dispositivo que toca MP3 se o meu servidor de músicas só possui arquivos digitais no formato WAV. Como em muitas das decisões que são tomadas em torno de quais padrões utilizar, o mais importante de fato é a definição do padrão a ser usado por vários fabricantes, mesmo que o escolhido não seja o tecnicamente melhor.

- Segurança

Esse talvez seja um dos pontos menos explorados tecnicamente, embora não menos importante no ambiente pervasivo. Com um grande número de dispositivos disponíveis para acesso

na rede, é fundamental a preocupação com relação à autenticação e autorização [18] de pessoas e dispositivos para acessar algum recurso dessa rede.

No ambiente do futuro, iremos lidar com redes pervasivas com um grande número de dispositivos conectados, passíveis tanto de ataques internos quanto externos.

Esse ponto introduz uma discussão que talvez seja a mais importante e polêmica no ambiente pervasivo: a dualidade entre permissividade e segurança. Sendo esses aspectos antagônicos, será de fundamental importância definir um ponto de conciliação, sem obviamente impedir que um determinado grau de segurança seja aplicado num determinado ambiente que exige um maior grau de proteção.

Tecnologias Atuais

WSDL

O que são Web Services ?

Web Services é uma tecnologia emergente que permite a integração de sistemas distribuídos. Baseada em tecnologias consagradas, como TCP/IP, HTTP, XML e SOAP, os Web Services tornam-se uma opção simples de disponibilizar um serviço que pode ser invocado remotamente por um cliente da rede.

Por exemplo, uma organização que vende matéria prima pode disponibilizar um serviço que possibilite seus compradores submeterem pedidos de compra automaticamente, através da integração de seus sistemas.

Um Web Service é disponibilizado em um servidor Web, na forma de um aplicativo servidor. Esse aplicativo é responsável por receber a mensagem de invocação do serviço remoto, realizar o processamento e responder ao cliente. Todas as mensagens utilizadas nessa conversação são formatadas num protocolo específico que é o SOAP (Simple Object Access Protocol), definido pelo W3C.

Dessa forma, poderíamos disponibilizar um Web Service e invocá-lo usando a URL a seguir, por exemplo:

http://meuhost/servico.cgi

WSDL

WSDL é a sigla para Web Services Description Language. Como o nome diz, uma linguagem para descrição de serviços Web.

WSDL é um padrão, definido pelo W3C, que define o formato para especificar a interface de um serviço Web, da mesma forma como temos o IDL (Interface Definition Language) em CORBA e COM.

O WSDL é escrito na linguagem XML, utilizando elementos e atributos específicos para definir a interface que um serviço oferece. WSDL pode ser entendido como um contrato para colaboração entre um cliente e o servidor que hospeda o serviço Web.

Com o arquivo WSDL de um determinado serviço, podemos produzir um Proxy, que permite um cliente invocar o serviço Web descrito. Um Proxy é equivalente a um stub na linguagem CORBA. A obtenção dessa interface pode ser extremamente facilitada pelo mecanismo padrão de consulta disponibilizada pelo Web Service. Por exemplo, se eu desejo obter a interface do Web Service endereçado pela URL descrita acima, eu posso obter o WSDL correspondente executando o seguinte comando:

http://meuhost/servico.cgi?WSDL

Na estrutura do arquivo WSDL, encontramos 5 seções distintas, sendo estas divididas em 2 grupos:

O grupo principal é conhecido como grupo de definições abstratas, contém as seções Types, Messages e PortTypes. Esse grupo define elementos da interface independentes de plataforma e de linguagem.

O segundo grupo é conhecido como descrições concretas e contém as seções Bindings e Services. Esse grupo contem definições específicas de determinado site.

- Seção Types: definições de tipos independentes de plataforma;
- Seção Messages: contém parâmetros de funções (os de entrada separados dos de saída) e descrições do documento;
- Seção PortTypes: define as assinaturas das funções, unindo o nome da função, os parâmetros de entrada e os de saída;
- Seção Bindings: especifica ligações entre cada operação a ser exposta e sua respectiva assinatura em PortTypes;
- Seção Services: especifica o endereçamento das portas de cada ligação.

UDDI

Introdução

A tecnologia UDDI (Universal Description, Discovery and Integration) define uma especificação [10] que permite uma empresa publicar e descobrir serviços disponibilizados por parceiros de tal forma a facilitar a integração de sistemas distribuídos.

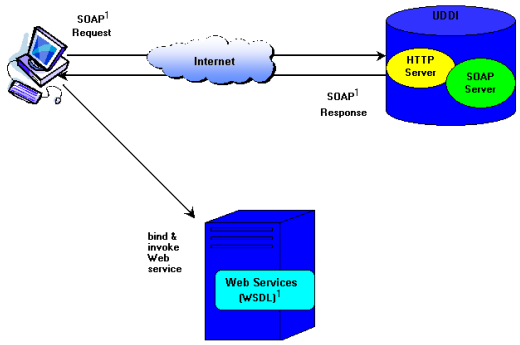
Resumidamente, o UDDI pode ser entendido como um serviço de "Páginas Amarelas", onde uma empresa publica suas informações e os serviços (Web Services) oferecidos por ela. Outras empresas acessam esse catálogo a procura de parceiros de negócios e descobrem as funcionalidades das quais podem usufruir para integrar seus sistemas de informação.

Arquitetura UDDI

Um serviço UDDI pode ser entendido como um grande banco de dados de registros, cujo objetivo é manter referências de fácil manipulação e recuperação. Este serviço é disponibilizado em um nó (UDDI node), que consiste em um site Internet que pode ser acessado pela interface gráfica HTML ou programaticamente.

O UDDI é classificado como um sistema de registros, não um repositório, uma vez que seu objetivo não é manter tudo que é necessário para o funcionamento ou integração com o serviço, mas apenas vínculos que apontam para onde estão localizadas essas informações.

O esquema gráfico [11] a seguir mostra como funciona uma consulta a um sistema UDDI:

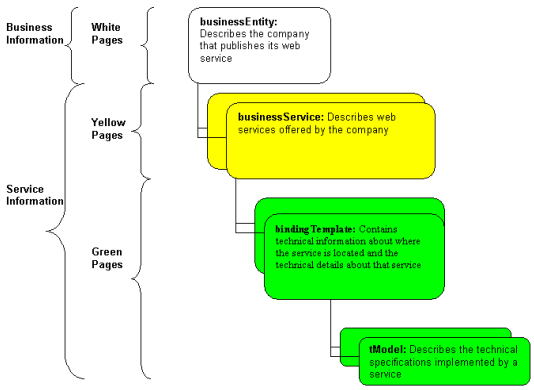


As informações disponibilizadas em um servidor UDDI são classificadas de acordo com tabelas padronizadas internacionalmente (chamadas Taxomonies), permitindo que uma empresa se identifique de forma precisa e que seja entendida de forma ampla por vários parceiros de negócios em várias partes do mundo.

Como exemplos de tabelas de classificação podemos citar:

- ISO 3166 Geographic Taxonomy;
- Universal Standard Products and Services Codes (NAICS-1997).

As informações disponibilizadas são organizadas de acordo com a seguinte estrutura de dados [11]:



Considerando o grande número de empresas interessadas em disponibilizar seus serviços para seus parceiros de negócio, ficaria cada vez mais difícil o descobrimento de novas oportunidades de integração. O UDDI vem agregar valor à tecnologia dos Web Services uma vez que promove sua utilização por um número maior de parceiros.

Location-aware Applications

Introdução

O artigo "An Approach to Providing a Seamless End-User Experience for Location-Aware Applications" [12] aborda um conceito de computação pervasiva onde a localização de uma pessoa ou objeto é usada para focar a aplicação ou o conjunto de serviços a serem oferecidos naquele contexto.

Um dos fatores atuais que estimulam o desenvolvimento de tais aplicativos é a demanda imposta pela FCC (Federal Communication Comission) americana para que telefones móveis sejam capazes de determinar sua localização aproximada devido à necessidade de prover, de forma transparente, os serviços de emergência daquela região (como de ambulância, polícia, etc.)

Além disso, o desenvolvimento atual de tecnologias de GPS (Global Positioning System) também estimula o aparecimento de sistemas computacionais sensíveis à localização do usuário, devido ao barateamento deste mecanismo.

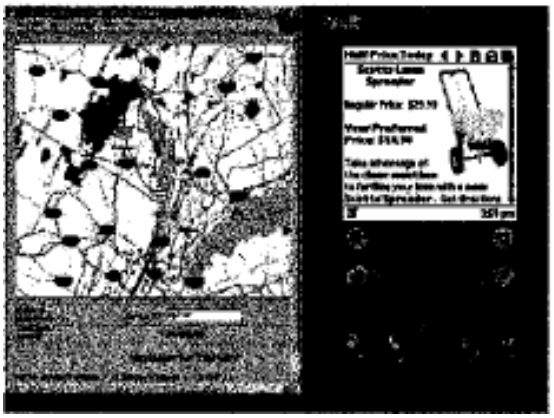
Para realizar a proposta de prover serviços ao usuário de relevância e sensíveis à localização, dois parâmetros básicos são necessários: localização do consumidor - por uma tecnologia qualquer de localização - e as preferências do consumidor - para oferecer uma maior precisão na escolha dos serviços que serão relevantes para o usuário.

Como exemplo de aplicativos onde este tipo de tecnologia seria altamente relevante, podemos citar algumas:

- Páginas amarelas localizadas;
- Navegação automotiva com assistência de rota;
- Localizador pessoal: pessoas com deficiências físicas poderiam ser altamente beneficiadas por essa tecnologia;
- Propaganda personalizada.

Simulação

Foi desenvolvido um protótipo que simulasse o comportamento descrito no artigo. Um usuário, portando um PalmPDA, possui informações embutidas no dispositivo que correspondiam a um cupom eletrônico de uma loja. Ao se aproximar da localidade onde está estabelecida a loja, na tela do Palm seria exibida a seguinte interface:



Na parte a direita da tela, seria exibido ao usuário um mapa que o orientaria a chegar até a loja. Na parte a esquerda o cupom que o usuário possui para compra com desconto nesta loja. Ao entrar na loja, a seguinte interface seria exibida:

Nesta tela o usuário teria indicações internas na loja sobre as prateleiras e corredores e a seta indicaria onde o produto desejado se encontra.

Desafios

Alguns desafios devem ser transpostos em nível de infra-estrutura para suportar tal tecnologia. Em nível de tecnologia podemos citar:

- Portabilidade de sessão sobre diferentes tecnologias wireless: usando o exemplo da simulação proposta do artigo, poderíamos citar a dificuldade na mudança ocorrida quando o usuário estava fora da loja – usando uma determinada tecnologia de localização – e quando ele entrou na loja – provavelmente usando alguma infra-estrutura disponibilizada pela empresa. Durante essa transição, a sessão do usuário (dados sobre sua localização) deverá ser mantida;

- Disponibilidade de tecnologia de localização interna (indoor): ainda é raro encontrar empresas que disponibilizem em seus ambientes equipamentos capazes de suportar a localização de dispositivos móveis;

- Transição transparente entre diferentes tecnologias wireless: a transição de um ambiente ou localidade para outra deve ser suportada de forma transparente quanto à tecnologia adotada. Exemplo: num ambiente a tecnologia é GPS, noutro a tecnologia é Bluetooth.

Em nível de sistema podemos citar as seguintes dificuldades:

- Estruturas eficientes de procura de serviços;
- Interoperabilidade dos serviços participantes.

Tecnologias de Localização

- Externa (outdoor)
  - GPS (Global Positioning System), desde 1980;
  - Baseados nos celulares:
    - Angle of Arrival (AOA);
    - Time Difference of Arrival (TDOA);

- Interna (indoor):
  - Não estão maduras;
  - Receptores GPS não funcionam;
  - Celulares: 100 metros de imprecisão;
  - Wireless LAN access points (802.11b). Bluetooth Local Positioning beacons e Assisted-GPS: atendem, mas não estão amplamente disponíveis.

Objetivos

Entre os objetivos para se alcançar aplicações sensíveis à localização do usuário, existem alguns aspectos devem ser verificados:

- Mobilidade de comunicação sem fio: esta será uma característica fundamental para permitir tais aplicações, como por exemplo, para manutenção de sessões IPs, porém não foram analisados no artigo citado [12];

- Integração de tecnologias de localização: tecnologias como GPS (Global Positioning System) e Bluetooth deverão ser capazes de se integrar de forma transparente, preferencialmente fazendo uso de uma API única para facilitar os sistemas que a eles se integrarem;

- Transição transparente através dos limites das organizações e/ou empresas: essa é uma característica fundamental visto que será um desafio permitir que as organizações tenham acesso a dados de preferência dos usuários que se encontram na sua localidade, porém garantir ao usuário seu direito de privacidade;

- Descoberta automática de serviços;

Propostas

Dynamic Bookmarks

É um mecanismo cujo objetivo é capturar requerimentos do usuário. O conceito de dymanic bookmarks é similar ao conceito de bookmarks existente nos navegadores Internet existentes. A diferença é que dado um conjunto de bookmarks pertencente a um usuário, os links relativos a cada um desses bookmarks irá variar dependendo da localidade onde esse usuário se encontra.

A cada Dynamic Bookmark é associado um conjunto de atributos que define características padronizadas, usando valores tabelados por instituições como a NAICS por exemplo, usados para encontrar URLs relevantes numa determinada localidade. Veja o exemplo de uma lista de Dynamic Bookmarks e seus respectivos atributos:

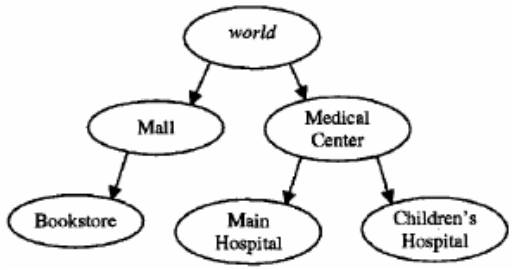
Bookmark Name	Attributes	Preferences	URL
Garden Center	Type = Nursery and Garden Centers Services = product reservation Capabilities = routes, store locator	Profile release = no Partial match = yes	http://gardenworld.com
Store Guide	Type = Nursery and Garden Centers Services = store guide Capabilities = product locator	Profile release = no Partial match = yes	NONE

Location Domain

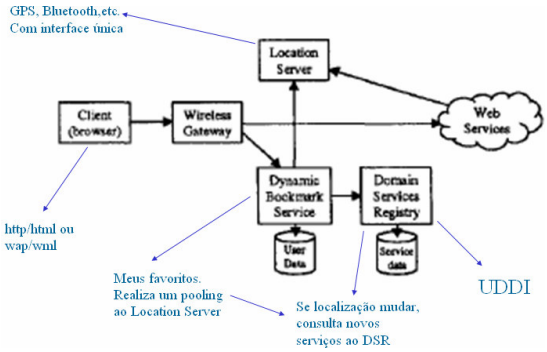
É um mecanismo para prover serviços relevantes baseados na localização. Cada Location Domain contemplará:

- uma área geográfica bem delimitada;
- hospeda um conjunto de recursos relacionados àquela área;
- provê uma lista de referências onde cada recurso se localiza;
- oferece serviços como computação da distância entre pontos e determinação da rota de um ponto a outro.

Cada Domain tem uma forma especifica de descrever localidade. Exemplo: a localidade World pode ser endereçada em termos de localização através de coordenadas de latitude e longitude. O interior de uma loja ou magazine pode ser endereçado através do número do corredor onde o usuário se encontra. A figura a seguir exemplifica uma típica organização hierárquica de um Location Domain:



Arquitetura



SLP

Introdução

SLP é a sigla para Service Location Protocol, que define um protocolo para descobrimento de serviços numa rede. SLP é um padrão independente mantido para IETF (Internet Engineering Task Force) e empregado por diversas empresas do ramo, entre as quais: Sun, Axis, HP, Novell, Apple, Lexmark, IBM, Madison River Tech, etc. SLP é um protocolo projetado para redes TCP/IP e se propõe a ser escalável para redes empresariais largas.

Arquitetura

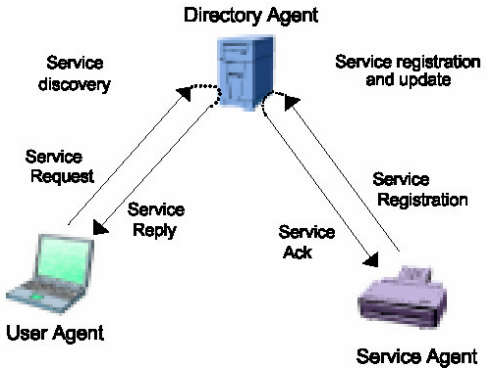
A arquitetura do protocolo SLP define três elementos principais: User Agents (UA), Service Agents (SA) e Directory Agents (DA).

- User Agents (UA): realiza a descoberta de serviços, em nome do cliente (usuário ou aplicação);

- Service Agents (SA): divulga a localização e características dos serviços, em nome dos serviços (provedores/servidores);

- Directory Agents (DA): reúne em uma base de dados endereços e informações sobre serviços recebidos dos Service Agents (SAs), e responde às requisições de serviços dos User Agents (UAs).

O esquema a seguir [13] demonstra alguns detalhes de interação entre os componentes do sistema:



Quando um novo serviço conecta a rede, o SA contata o DA para anunciar a existência do novo serviço (Service Registration). Quando um usuário necessita de um certo serviço, o UA pergunta por serviços disponíveis na rede ao DA (Service Request). Após receber o endereço e características do serviço desejado, o usuário pode finalmente utilizar o serviço.

Antes de um cliente (UA ou SA) ser capaz de contatar o DA, é necessário descobrir a existência do DA. Existem 3 métodos para a descoberta do DA: estática, ativa e passiva.

Com a descoberta estática, os agentes SLP (UA e SA) obtém o endereço do DA através de um DHCP (Dynamic Host Configuration Protocol). Existem algumas configurações especiais que devem ser defi-

nidas no servidor DHCP para servir ao protocolo SLP. Os servidores DHCP distribuem os endereços dos DAs para os hosts que os requisitam.

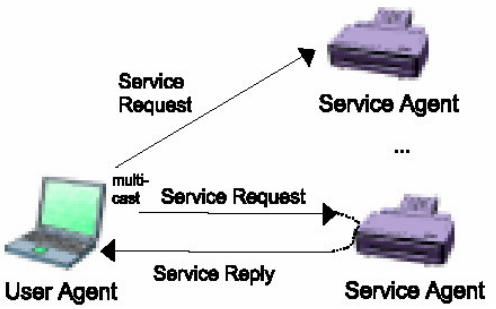
Na descoberta ativa, UAs e SAs enviam requisições de serviços para o endereço de grupo multicast SLP (239.255.255.253). Um DA escutando este endereço irá eventualmente receber uma requisição de serviço e responder diretamente (via unicast) para o agente requerente.

No caso de descoberta passiva, DAs periodicamente enviam mensagens de anúncio multicast para os serviços. UAs e SAs descobrem através dessa mensagem o endereço do DA e são a partir de então capazes de contatar diretamente o DA (via unicast).

É importante ressaltar que a presença do Directory Agent (DA) não é obrigatória num sistema SLP. De fato, ele é usado especialmente em redes largas com muitos serviços, uma vez que seria possível categorizar os serviços em diferentes grupos (escopos). Em redes menores (redes domésticas ou de interior de carros) é mais indicado usar o SLP sem o DA.

Portanto, o SLP tem dois modos operacionais, dependendo da existência ou não do Directory Agent (DA). Se o DA existe na rede (como mostrado na figura acima), ele irá coletar todas as informações dos serviços anunciados pelos SAs. Os UAs irão enviar requisições de serviços para os DAs e receberão a informação desejada sobre o serviço.

Caso o Directory Agent (DA) não esteja presente, o seguinte esquema [13] se aplica:



Sem a existência de um DA, UAs mandam repetidamente requisições de serviços para o endereço SLP multicast. Todos os SAs escutam por estas requisições multicast e, se eles publicam (possuem) o serviço requisitado, eles enviam mensagem diretas (unicast) para o UA. Além disso, SAs enviam mensagens multicast se anunciando periodicamente, para que os UAs possam saber da existência de novos serviços.

Os serviços são anunciados usando um Service URL e um Service Template. O Service URL contém o endereço IP, a porta e o caminho do serviço. Service Template especifica os atributos que caracterizam o serviço e seus valores padrão.

Como exemplo, podemos considerar a descrição a seguir [13], que define a especificação de um serviço de uma impressora na rede:

```
service:printer://lj4050.tum.de:1020/queue1
scopes = tum, bmw, administrator
printer-name = lj4050
printer-model = HP LJ4050 N
printer-location = Room 0409
color-supported = false
pages-per-minute = 9
sides-supported = one-sided, two-sided
```

A primeira linha corresponde ao Service URL. As outras linhas correspondem ao Service Template.

O SLP suporta procura de serviço baseada em critérios, usando operadores como AND, OR e outros, além de possibilitar consulta à substrings.

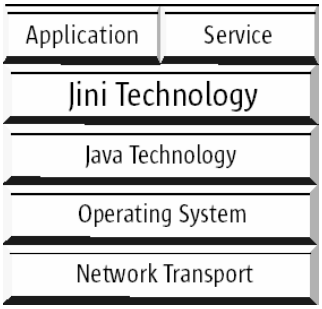
Comparando com outras arquiteturas [13]:

Feature	SLP	Jini	UPnP
Developer	IETF	Sun	Microsoft
License	open source	open license; but fee for commercial use	open (only for members)
Version	2	1.0	0.91
Network transport	TCP/IP	independent	TCP/IP
Programming language	independent	Java	independent
OS and platform	dependent	independent	dependent
Code mobility	no	yes (Java RMI)	no
Srv attributes searchable	yes	yes	no
Central cache repository	yes (optional)	Optional using SLP	no
Operation w/o directory	yes	Lookup Table required	
Leasing concept	yes	Java based	yes
Security	IP dependent		IP dependent

JINI

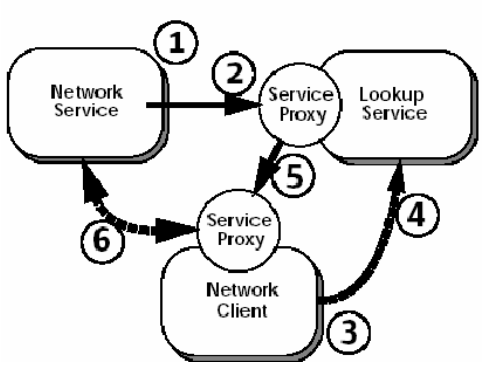
JINI é uma tecnologia para ambientes de computação dinâmicos – classificado neste artigo como pervasivos – que permite a interação espontânea de serviços e dispositivos numa rede.

A arquitetura da JINI é exibida na figura a seguir:



Como podemos observar, o JINI roda sobre a máquina virtual JAVA (run-time). Apesar dessa característica permitir uma neutralidade de sistema operacional e de protocolo de transporte, coloca o JINI dependente de linguagem de desenvolvimento.

A figura a seguir mostra a seqüência de comunicação definida no protocolo JINI:



1. O serviço de rede descobre o Lookup Server;
2. O serviço de rede envia o Proxy do serviço que ele oferece para o Lookup Server;
3. O cliente de rede descobre o Lookup Server;
4. O cliente de rede envia uma requisição para o Lookup Server procurando por serviços disponíveis que são de seu interesse;
5. O Lookup Server envia ao cliente de rede os proxies dos serviços encontrados no seu repositório que atendem ao critério de busca definido;
6. O cliente de rede interage diretamente com o serviço de rede.

Um ponto importante deve ser observado nesse protocolo de comunicação: a dependência de um repositório central, no caso o Lookup Server. Essa dependência causa transtornos em ambientes que exigem configuração mínima.

UPnP

Introdução

A tecnologia Universal Plug and Play é uma arquitetura para permitir o estabelecimento de uma rede pervasiva ponto-a-ponto para dispositivos inteligentes em geral. Como dispositivos inteligentes devemos entender não somente os tradicionais PCs, mas também dispositivos sem fio genéricos que podem estar oferecendo ou usufruindo recursos de todos os participantes de uma rede.

Em 1999, foi fundado o Universal Plug And Play Group, cujo objetivo é promover a disseminação da tecnologia através de um esquema participativo. Para permitir a cooperação de várias empresas interessadas nesta tecnologia, foi criado um fórum [4] (www.upnp.org) através do qual várias empresas trocam conhecimentos sobre a tecnologia e ajudam na definição e acertos dos padrões.

A tecnologia UPnP é mais do que uma simples extensão do modelo Plug And Play para periféricos que se estabeleceu como padrão no começo desta década. UPnP é projetado para suportar redes sem necessidade de configuração e invisíveis, com descoberta automática de dispositivos de vários fornecedores. Isto significa que um dispositivo pode dinamicamente entrar numa rede, descobrir os serviços oferecidos por outros dispositivos, aprender como utilizá-los e dizer para os outros dispositivos quais serviços ele oferece. Serviços como DHCP e DNS são opcionais na tecnologia e utilizados somente se existem na rede. Ao final, o dispositivo pode sair suavemente e sem deixar nenhum estado de inconsistência na rede.

UPnP abraça padrões Internet, largamente aceitos, como IP, TCP, UDP, HTTP e XML. A escolha do IP como base para a interconexão é estratégica, visto sua habilidade de suportar diferentes mídias físicas e permitir uma fácil integração de dispositivos de vários fabricantes. Além disso, via uma ponte, é possível acomodar dispositivos que não suporte IP.

Objetivos

- Sem drivers de dispositivos

Essa característica confere ao UPnP a facilidade de conexão de dispositivos sem a necessidade da árdua tarefa de configurá-los. Essa característica é possibilitada pelo uso de protocolos comuns. Uma vez que todos os dispositivos suportem esses padrões, a própria tecnologia UPnP se encarrega da preparação para a comunicação entre os dispositivos.

- Independente do sistema de comunicação

A própria característica aberta dos padrões Internet utilizados na UPnP fornecem a capacidade da utilização de vários meios para o sistema de comunicação: linha telefônica, linha de força, ethernet, rádio freqüência, etc.

- Independência de Plataforma

É possível desenvolver um dispositivo UPnP usando qualquer sistema operacional e qualquer linguagem de programação, bastando para tanto apenas abraçar os padrões estabelecidos pela arquitetura. Por exemplo, podemos ter um dispositivo UPnP baseado no sistema operacional PalmOS utilizando C++ como linguagem de desenvolvimento interagindo com um dispositivo baseado no sistema Windows CE usando Visual Basic como linguagem de programação. Essa característica, a propósito, não pode ser encontrada na proposta JINI, concorrente da UPnP.

- Controle por interface direta com usuário ou programática

Os dispositivos UPnP que oferecem serviços podem disponibilizar seus serviços através de uma interface própria ao usuário ou um controle programático dos seus serviços, permitindo o desenvolvedor personalizar a interface para um determinado dispositivo.

- Protocolos base comuns associados às tecnologias Internet

Através do uso das tecnologias já consagradas na Internet para conexão de dispositivos (IP, TCP, UDP, HTTP e XML), agregam-se um conjunto de protocolos definido pelo grupo UPnP que permitem a integração dos dispositivos.

- Extensível

Através de serviços adicionais agregados numa camada específica no topo da arquitetura básica, novos recursos podem ser anexados à tecnologia UPnP por interesse de algum fabricante específico.

Aplicações

A tecnologia UPnP visa atender às necessidades de um mundo cada vez mais conectado, onde conteúdos digitais de entretenimento e até pessoais podem ser acessados de vários dispositivos, no cenário doméstico ou empresarial, não importando onde este conteúdo está armazenado.

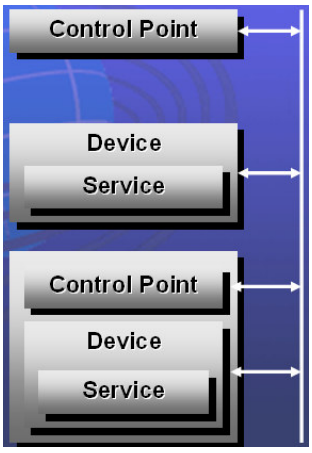
No cenário doméstico, a família pode se divertir compartilhando jogos baseados na Internet, projeção de fotos do álbum de família e indivíduos com dificuldades físicas podem interagir através de uma interface personalizada com máquinas de auto-atendimento atendendo suas necessidades especiais.

À medida que os custo dos sistemas de computação tornam-se acessíveis, organizações podem construir dispositivos inteligentes no projeto de muitos eletroeletrônicos comuns para o nosso dia-a-dia. Desta forma, estarão surgindo novas oportunidades para novos produtos e funcionalidades baseadas na conectividade natural e transparente entre os dispositivos.

A conectividade pode ser usada para controlar remotamente outros dispositivos, para mover conteúdo digital na forma de áudio, vídeo e imagens, compartilhar informação entre os dispositivos e com a World Wide Web, e trocar conteúdo estruturado e seguro para suportar comércio eletrônico.

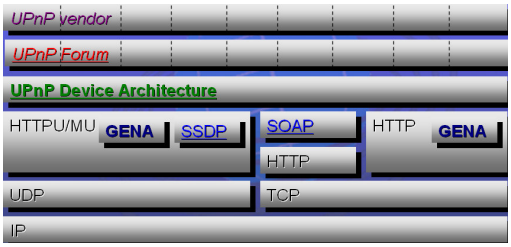
Especificação da Arquitetura

A arquitetura UPnP define protocolos para comunicação entre controladores (Control Points) e dispositivos (Devices), conforme representado na figura a seguir [3]:



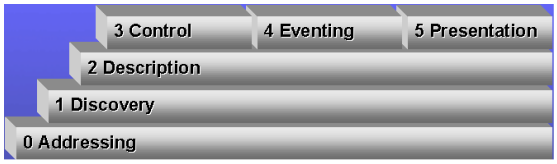
Um Controlador (ou Control Point) pode ser entendido como o cliente de serviços disponibilizados pelos dispositivos provedores (Devices).

Para suprir as necessidades de comunicação definidas pela arquitetura, a tecnologia UPnP usa a seguinte pilha de protocolo [4]:



- Camada UPnP vendor: contém logicamente apenas informações específicas de cada fornecedor sobre seus dispositivos;
- Camada UPnP Forum: o conteúdo do fornecedor é suplementado com informações definidas pelo comitê do Fórum UPnP;
- Camada UPnP Device Architecture: protocolos específicos da UPnP. Serão descritos com mais detalhes na continuação deste documento;
- Camadas HTTP: podem ser definidas como Multicast (HTTPMU, vários receptores) ou Unicast (HTTPU, um para um). Além disso, as mensagens nesta camada são formatadas de acordo com o tipo de conteúdo que carregam, sendo eles:
    - SSDP (Simple Service Discovery Protocol);
    - GENA (General Event Notification Architecture);
    - SOAP (Simple Object Access Protocol);
- Camada UDP ou TCP;
- Camada IP.

O protocolo UPnP define os seguintes passos de conversação numa rede [4]:



O objetivo desses passos de conversação é permitir que um dispositivo ganhe um endereço IP para poder participar da rede, descubra outros dispositivos e suas capacidades, e disponibilize para outros dispositivos interessados os serviços que ele oferece. Cada um desses passos cumpre um papel definido dentro deste propósito amplo, sendo descritos a seguir:

Passo 0: Addressing

O endereçamento IP é básico para a comunicação no UPnP. É neste passo que um dispositivo ganhará um número IP e poderá participar desta rede. Como um serviço DHCP na rede é opcional, veremos a seguir os sub passos e possibilidades de interação do dispositivo com a rede.

0.1. Determinando se usa Auto-IP ou não

Neste sub passo, o dispositivo deve descobrir se existe um serviço de designação de IP (DHCP). Ela faz isso enviando uma mensagem (DHCPDISCOVER)

interrogando um possível serviço DHCP. Durante um determinado tempo, o dispositivo deve esperar por uma resposta DHCPOFFER. Caso receba essa resposta, o dispositivo deve continuar o processo de designação automático de IP. Caso contrário, o dispositivo deverá escolher um IP para utilizar.

0.2. Escolhendo um endereço

Para se autoconfigurar com um IP, o dispositivo usa um algoritmo para escolher um endereço do intervalo 169.254/16.

0.3. Testando o endereço

Uma vez tendo um endereço IP escolhido, o dispositivo deve verificar se ele está em uso. Ele faz isso usando o protocolo ARP (Address Resolution Protocol). Caso o dispositivo descubra que o endereço já está em uso, ele tenta novamente escolhendo outro IP e repete a operação.

0.4. Teste periódico de disponibilidade de um sistema dinâmico de endereço

Caso o dispositivo tenha usado um Auto-IP, ele deve periodicamente verificar se um serviço DHCP foi disponibilizado na rede. Em caso afirmativo, ele deve abandonar o uso do endereço que se auto designou e utilizar o disponibilizado pelo DHCP.

0.5. Utilizando um servidor de nomes

Depois que o dispositivo ganha um endereço IP, podem existir situações onde seja interessante utilizar um serviço de tradução de nome de dispositivos para endereços IP. Nesta situação, o dispositivo deve se registrar no serviço de nomes usando o IP que lhe foi atribuído ou por intermédio do próprio serviço DHCP.

0.6. Mapeamento de nomes para endereços IPs

Uma vez o dispositivo registrando-se num serviço DNS, outros dispositivos são capazes de encontrá-lo, interrogando esse serviço (conhecido estática ou dinamicamente através do DHCP) e recebendo com resposta o endereço IP do procurado.

Passo 1: Discovery

Após obter um endereço IP, o Device está pronto para anunciar seus serviços para os Control Points na rede. De forma similar, quando um Control Point entra na rede, o protocolo permite que este procure Devices de interesse na rede. A mensagem de troca, em ambos os casos, contém algumas informações sobre o dispositivo e seus serviços, um identificador, e um "ponteiro" para informações mais detalhadas.

Quanto um novo Device é adicionado na rede, ele dispara mensagens alertando os Control Points dos serviços que ele oferece. Todos os Control Points podem escutar um endereço multicast padrão para essas mensagens. Abaixo esta a mensagem de notificação [2]:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: search target
NTS: ssdp:alive
SERVER: OS/version UPnP/1.0 product/version
USN: advertisement UUID
```

O dispositivo, depois de anunciado, tem seus serviços considerados disponíveis até vencer o tempo de expiração (max-age), ou até que uma mensagem de ByeBye seja enviada:

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: search target
NTS: ssdp:byebye
USN: advertisement UUID
```

Da mesma forma, quando um Control Point é adicionado na rede, ele dispara uma mensagem multicast procurando Devices, serviços, ou ambos, de seu interesse. Todos os Devices devem escutar um endereço padrão de multicast por essas mensagens e devem responder se quaisquer de seus serviços combinam com o critério de busca na mensagem de Discovery. Exemplo dessa mensagem é exibido abaixo [2]:

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
```

Uma vez recebida uma consulta por um Device, o mesmo deve responder ao Control Point passando detalhes sobre sua existência e características [2]:

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/1.0 product/version
ST: search target
USN: advertisement UUID
```

Passo 2: Description

Após a descoberta de vários Devices da rede, um dispositivo Control Point deverá obter detalhes sobre quais são as capacidades desse determinado Device. Para tanto, ele deverá interrogar o Device com uma mensagem de consulta. Isto é feito utilizando o protocolo HTTP GET sobre a URL (LOCATION) fornecida no passo de Discovery. A seguir esta mensagem de GET [2]:

```
GET path to description HTTP/1.1
HOST: host for description:port for description
ACCEPT-LANGUAGE: language preferred by control point
```

Como resposta, o Control Point recebe uma mensagem conhecida como Device Description, onde detalhes do dispositivo são expostos. A seguir a estrutura da mensagem [2]:

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
        <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      Declarations for other services defined by a UPnP Forum working committee (if any)
        go here
      Declarations for other services added by UPnP vendor (if any) go here
    </serviceList>
    <deviceList>
      Description of embedded devices defined by a UPnP Forum working committee (if any)
        go here
      Description of embedded devices added by UPnP vendor (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

Como se pode reparar, a mensagem Device Description disponibiliza as seguintes informações relevantes:

- Informações de versão do dispositivo;
- Informações do fabricante do dispositivo;
- Lista de serviços oferecidos, cada um disponibilizando uma URL para o passo Control (invocação de serviços) ou o passo Eventing (consulta de variáveis de estado);

Com essas informações, o Control Point já conhece toda a lista de serviços que um Device oferece, a partir daí o Control Point já é capaz de interrogar o Device para obter maiores detalhes sobre um serviço específico oferecido. Ele faz isso realizando um novo HTTP GET, usando como URL de busca a informação fornecida na tag SCPDURL. Como resposta, o Control Point recebe a mensagem Service Description, conforme exibida a seguir [2]:

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>actionName</name>
      <argumentList>
        <argument>
          <name>formalParameterName</name>
          <direction>in or out</direction>
          <retval />
          <relatedStateVariable>stateVariableName</relatedStateVariable>
        </argument>
        Declarations for other arguments defined by UPnP Forum working committee (if any)
          go here
      </argumentList>
    </action>
    Declarations for other actions defined by UPnP Forum working committee (if any)
      go here
    Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueList>
        <allowedValue>enumerated value</allowedValue>
        Other allowed values defined by UPnP Forum working committee (if any) go here
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueRange>
        <minimum>minimum value</minimum>
        <maximum>maximum value</maximum>
        <step>increment value</step>
      </allowedValueRange>
    </stateVariable>
    Declarations for other state variables defined by UPnP Forum working committee
      (if any) go here
    Declarations for other state variables added by UPnP vendor (if any) go here
  </serviceStateTable>
</scpd>
```

Nesta mensagem, o Control Point recebe informações mais detalhadas sobre um determinado serviço oferecido pelo Device, como por exemplo a forma de invocação do serviço com a lista de parâmetros a ser usada.

Passo 3: Control

É nesta etapa que um Control Point invoca um serviço no Device. Para fazer isso, o Control Point envia uma mensagem Invoke ao Device, como a seguir [2]:

```
POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:serviceType:v#actionName"

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:actionName xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
      <argumentName>in arg value</argumentName>
      other in args and their values go here, if any
    </u:actionName>
  </s:Body>
</s:Envelope>
```

Essa mensagem é formatada usando o padrão SOAP (Simple Object Access Protocol), que define um padrão para invocação de serviços sobre HTTP.

Como resultado, o Control Point pode receber uma mensagem de sucesso, como a seguir [2]:

```
HTTP/1.1 200 OK
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:actionNameResponse xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
      <argumentName>out arg value</argumentName>
      other out args and their values go here, if any
    </u:actionNameResponse>
  </s:Body>
</s:Envelope>
```

Caso ocorra algum erro, o Control Point receberá uma mensagem de erro no seguinte formato [2]:

```
HTTP/1.1 500 Internal Server Error
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Client</faultcode>
      <faultstring>UPnPError</faultstring>
      <detail>
        <UPnPError xmlns="urn:schemas-upnp-org:control-1-0">
          <errorCode>error code</errorCode>
          <errorDescription>error string</errorDescription>
        </UPnPError>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

Repare que o padrão SOAP define tags específicas para classificar os tipos de erros ocorridos (error code) e descrevê-los.

Outra possibilidade de invocação é a consulta a uma variável de estado do Device. Isto é feito enviando uma mensagem Query Invoke, como exibida abaixo [2]:

```
POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:control-1-0#QueryStateVariable"

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:QueryStateVariable xmlns:u="urn:schemas-upnp-org:control-1-0">
      <u:varName>variableName</u:varName>
    </u:QueryStateVariable>
  </s:Body>
</s:Envelope>
```

Como resposta, da mesma forma de quando da invocação de um serviço, pode ser retornada uma mensagem de sucesso ou erro. A de sucesso é exibida a seguir [2]:

```
HTTP/1.1 200 OK
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:QueryStateVariableResponse xmlns:u="urn:schemas-upnp-org:control-1-0">
      <return>variable value</return>
    </u:QueryStateVariableResponse>
  </s:Body>
</s:Envelope>
```

Caso ocorra algum erro, a seguinte mensagem de erro é retornada:

```
HTTP/1.1 500 Internal Server Error
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Client</faultcode>
      <faultstring>UPnPError</faultstring>
      <detail>
        <UPnPError xmlns="urn:schemas-upnp-org:control-1-0">
          <errorCode>error code</errorCode>
          <errorDescription>error string</errorDescription>
        </UPnPError>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

Passo 4: Eventing

Nesta etapa, um Control Point pode se manifestar como interessado em conhecer uma variável de estado de um determinado Device. O Control Point sabe da existência dessa variável de estado – e como consultá-la – através do Device Description, obtido na etapa Description.

Para ser informado sobre qualquer mudança numa variável de estado no Device, o Control Point pode se inscrever neste dispositivo, usando uma mensagem Subscribe:

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
CALLBACK: <delivery URL>
NT: upnp:event
TIMEOUT: Second-requested subscription duration
```

Enquanto o tempo estipulado na mensagem de Subscription não expirar, o Device notificará qualquer mudança de estado na variável em que o Control Point manifestou interesse. Caso o Control Point deseje manter a Subscription, deverá repetir o envio da mensagem acima num período inferior ao tempo de expiração, mantendo a Subscription.

Quanto houver alguma mudança na variável de estado do Device, o mesmo enviará uma mensagem comunicando o novo valor dessa variável a cada um dos Control Points que se registraram como interessados. A mensagem que comunica essa mudança é exibida a seguir [2]:

```
NOTIFY delivery path HTTP/1.1
HOST: delivery host:delivery port
CONTENT-TYPE: text/xml
CONTENT-LENGTH: Bytes in body
NT: upnp:event
NTS: upnp:propchange
SID: uuid:subscription-UUID
SEQ: event key

<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
  <e:property>
    <variableName>new value</variableName>
  </e:property>
  Other variable names and values (if any) go here.
</e:propertyset>
```

Passo 5: Presentation

Uma vez que se conhece os Devices, todos os serviços que eles oferecem e como invocá-los, um Control Point terá a capacidade de exibir uma interface ao usuário permitindo este controlar os Devices de forma facilitada. Para isto, basta o Control Point enviar uma requisição HTTP GET ao Device solicitando a interface que ele oferece. Como resposta, o Control Point recebe uma página HTML, que será exibida ao usuário.

Nesta página HTML, estará exposta a interface aos serviços do Device através de links para as URLs que servem para invocar os serviços do dispositivo. Além disso, nesta página HTML, o Device poderá exibir dados sobre suas variáveis de estado.

Abaixo é exibido um exemplo de página HTML com a interface de um Device do tipo Ar-Condicionado:

Ar Condicionado

Ações:

- Aumentar temperatura
- Diminuir temperatura
- Ligar
- Desligar

Variáveis de estado:

- Temperatura atual: 28ºC
- Temperatura desejada: 25ºC

A opção por UPnP

Após a análise dos vários padrões de conectividade existentes no mercado e analisar seus pontos positivos e negativos, acredito que o padrão UPnP é o que mais se aproxima do ambiente pervasivo definido no início deste artigo.

Entretanto, devo considerar que grande parte dessa minha crença deva-se meramente à observação do mercado em torno desse padrão, a despeito dos aspectos técnicos que preferencialmente deveriam fundamentar todas as decisões de escolha de padrões. Considerando que o foco principal do ambiente pervasivo é a integração, é de grande interesse que os principais fornecedores de tecnologia apóiem o padrão de forma integral e que até colaborem para o seu enriquecimento.

A aceitação massiva do UPnP por fabricantes de dispositivos eletroeletrônicos e de computadores deve-se principalmente à característica aberta e extensível da tecnologia.

Ao contrário do padrão JINI, cujos padrões são definidos pela Sun – empresa que detém a marca registrada da tecnologia – o UPnP foi concebido por um consórcio de empresas, que montou uma organização dedicada ao desenvolvimento e promoção da tecnologia – o UPnP.org.

O UPnP.org [4] estimula participação de qualquer empresa que possa agregar valor às especificações da arquitetura. Este fato fica claro na própria especificação atual da arquitetura que define, na sua pilha de protocolo, uma camada (UPnP vendor) exclusiva para implementações particulares de um determinado fornecedor.

Projeto Prático

Como forma de demonstrar o ambiente UPnP, analisei dois kits de desenvolvimento para o ambiente UPnP: o Linux SDK for UPnP [21] e o Intel Tools for UPnP Technologies [22].

O SDK para Linux disponibiliza uma série de bibliotecas em C com primitivas específicas para estabelecer a conexão e conversação de dispositivos UPnP. Além disso, esse SDK exige uma série de requisitos de configuração de rede do computador onde reside o SDK. Depois de muitas tentativas frustradas de configuração do sistema operacional para que atendesse os requisitos do SDK, acabei decidindo abandonar a intenção de utilizar este SDK.

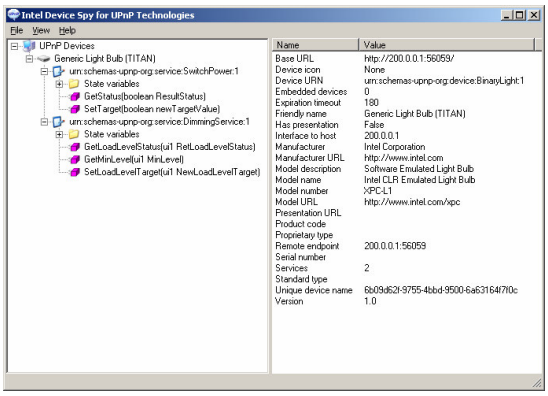
O SDK disponibilizado pela Intel é baseado na máquina virtual da Microsoft (.NET runtime) e que atualmente só possui versão para o ambiente Windows. Foi incrível a facilidade com que o ambiente UPnP de teste pode ser configurado. Por isso, optei por realizar meu experimento prático utilizando esse SDK.

O Intel Tools for UPnP Technologies [22] é um pacote de ferramentas para ajudar programadores e projetistas de hardware a acelerar o desenvolvimento, testes e disponibilização de dispositivos compatíveis com a tecnologia UPnP. Entre as ferramentas disponibilizadas por esse SDK, podemos citar as seguintes:

- Um Control Point (UCP) universal: ele exibe através de uma interface genérica qualquer tipo de Device que entre na rede. Ele também permite analisar os pacotes enviados para os Devices;
- Network Light: a implementação gráfica de exemplo de um Device que simula uma lâmpada UPnP. Ela pode ser ligada e desligada, aumentar ou diminuir a intensidade da luz, e ler os estados de suas variáveis (se o Device está ligado ou desligado e qual a intensidade atual da luz);
- Sniffer: permite a captura de pacotes de descobrimento do UPnP (fase Discovery). Permite também enviar mensagens de procura (search), depurar mensagens de resposta (response) e notificações (notify);
- AV Media Controller: Um Control Point universal para dispositivos de mídia;
- AV Media Server: permite compartilhar arquivos de mídia na rede UPnP;
- AV Renderer: permite a exibição de streaming de mídia em uma interface gráfica.

O experimento apresentado consiste em utilizar um Device e um Control Point. Para isso o simulador de device Network Light disponibilizado no SDK foi usado. Ao ser iniciado, ele exibe na tela do computador uma lâmpada apagada, como mostra a figura a seguir:

Depois disso, o Control Point universal foi iniciado. Após alguns segundos, o Device lâmpada foi reconhecido e pode então ser utilizada. A tela seguir exibe a interface do Control Point genérico:



Deve ser percebido que todos os passos de Addressing, Discovery e Description ocorreram, conforme descrito na seção UPnP deste artigo.

Finalmente, posso acionar um serviço disponibilizado pelo Device lâmpada que permite acendê-la. O nome do serviço é o SetTargetBoolean. Após invocar esse serviço informando como parâmetro true, a lâmpada se acende. Veja:

O problema da segurança

Introdução

Como posto no tópico "Necessidades de um ambiente pervasivo", a segurança é um requisito essencial em ambientes pervasivos. Afinal, imaginemos o seguinte cenário: uma pessoa tem em sua casa vários dispositivos, um deles permite fazer ligações à longa distância através de uma operadora local e que está conectado a uma rede pervasiva pessoal. Um vizinho com más intenções poderia, utilizando seu telefone sem fio, realizar ligações através do dispositivo da primeira pessoa. Este exemplo é colocado num ambiente doméstico, imaginemos então os problemas de segurança que poderiam ocorrer num ambiente empresarial.

É evidente que algum nível de segurança deve ser embutido no sistema pervasivo para que situações, como a descrita acima, não ocorram. Embora não existam soluções perfeitas hoje e provavelmente nunca existirão [18], podemos propor um nível de segurança que, para um ambiente pervasivo convencional, será suficiente.

No ambiente pervasivo, devemos considerar não só os dispositivos fixos de uma determinada rede, mas também aqueles que acessam essa rede remotamente ou dispositivos móveis introduzidos numa rede por intermédio de convidados.

Tipos de problemas relacionados à segurança

Podemos dividir as preocupações com segurança em três aspectos: autenticação, autorização e confiabilidade na troca de dados.

- Autenticação

Através do mecanismo de autenticação, podemos ter certeza que um determinado usuário ou dispositivo é quem realmente alega ser.

A forma mais comumente utilizada para ter tal garantia é utilizando a encriptação, na qual o remetente da mensagem tem uma chave secreta e a utiliza para codificar os dados que devem ser enviados para um determinado destinatário. O recebedor da mensagem usa essa mesma chave secreta para decodificar a mensagem. Esse método de autenticação é conhecido como algoritmo simétrico, pois a mesma chave é usada para encriptar e decriptar a mensagem.

Outra forma de autenticar através de encriptação é utilizando métodos de chaves públicas, conhecido também como assinaturas digitais. Nesse método o remetente da mensagem possui uma chave secreta e a utiliza para codificar a mensagem a ser enviada. O receptor utiliza a chave pública associada ao remetente para decodificar a mensagem, pois apenas essa chave pública é capaz de decifrar a mensagem. Neste caso é utilizado um algoritmo assimétrico, no qual uma mensagem codificada usando uma chave privada só pode ser decodificada por meio da chave pública associada ao mesmo proprietário da chave privada.

Apesar do método assimétrico (chave pública) ser mais seguro, esse mecanismo é muito mais demorado do que o método simétrico. Como não se pode onerar o sistema com custosos algoritmos de encriptação, o ideal é usar um método híbrido, onde a chave secreta utilizada para a comunicação é trocada através do método assimétrico e usada por um determinado período de tempo.

- Autorização

Identificar quem enviou a mensagem, através da autenticação, é essencial, mas não é suficiente. Por exemplo, imagine que existe numa casa um dispositivo de alarme. Num determinado momento chega, de um dispositivo identificado na rede, uma ordem para que ele seja desligado. Será que realmente o dispositivo que enviou a ordem tem autoridade para realizá-la ?

Desta forma, devemos garantir que, para cada dispositivo da rede, tenhamos uma lista de atividades permitidas para cada um desses dispositivos.

- Confiabilidade na troca de dados

Essa preocupação está relacionada com a confiança que devemos dar à rede quanto a possíveis alterações nos dados trafegados por ela. Imagine que um hacker mau intencionado pudesse alterar uma mensagem legítima enviada de um dispositivo para o outro, ordenando que ao invés de uma ação X, fosse executada a ação Y. Isso obviamente precisa ser evitado.

Da mesma forma que foi utilizada na autenticação, os dados que comandam ordens devem ser protegidos pelo método de encriptação utilizado, de tal forma que se alguma alteração for realizada na mensagem original, o destinatário receberia uma mensagem inválida e a ignoraria.

Outra cautela que devemos ter é quanto a um ataque conhecido como reenvio de mensagem [18] (Message Replay). Esse ataque consiste em pegar uma mensagem válida para executar uma ação X num dispositivo Y e reenviá-la sempre que o hacker desejar realizar a mesma ação legítima que a mensagem original executou. Para evitar esse tipo de problema, o protocolo deve prever a inserção de algum código serial na mensagem para que o destinatário certifique-se que não executou anteriormente essa mesma mensagem.

Uma proposta para UPnP

Como a proposta desse artigo é dar ênfase à tecnologia UPnP pelos motivos expostos no tópico "A opção por UPnP", iremos propor uma implementação de segurança como extensão dela.

Por ser a forma mais segura de encriptar dados, a proposta de codificação será usar o método assimétrico de chaves púbicas [20]. Para implementar esse esquema, precisamos ter na nossa rede um dispositivo que armazene as chaves públicas. Esse papel pode ser designado a um PC conectado nessa rede. Chamaremos esse dispositivo especial de *Controlador de Segurança*.

Cada dispositivo, por sua vez, viria com uma chave secreta e uma pública embutida especificamente pelo fabricante. Na primeira vez que um dispositivo qualquer fosse conectado na rede, o Controlador de Segurança receberia a chave pública do dispositivo e permitiria que o administrador da rede designasse um identificador para este dispositivo recém agregado. Esse identificador seria algo que permitisse uma fácil identificação do dispositivo, algo como "Sistema de Alarme".

Assim que esse dispositivo fosse registrado no Controlador de Segurança, ele estaria em condições de participar dessa rede segura. Desta forma, cada dispositivo que entrasse na rede (fase Discovery), seria identificado pelo Controlador de Segurança e teria acesso a todas as chaves públicas de todos os dispositivos registrados na rede.

Quando algum dispositivo enviar qualquer mensagem definida pelo protocolo UPnP [3], primeiramente essa mensagem seria criptografada usando a chave privada do dispositivo. Em acréscimo a mensagem codificada, seria enviada o identificador do dispositivo. O receptor ou receptores dessa mensagem recorreria ao Controlador de Segurança para descobrir a chave pública associada ao respectivo dispositivo remetente. De posse da chave pública enviada pelo Controlador de Segurança, o dispositivo realizaria a decodificação dos dados e validaria ou não essa mensagem.

No Controlador de Segurança, haveria um console onde o administrador do sistema listaria, para cada dispositivo, as autorizações dadas a cada Control Point dessa rede.

É fácil de perceber o overhead embutido no sistema com o uso desse mecanismo. Citamos os principais, como constituem desvantagens significativas desse modelo proposto:

- A necessidade de um PC assumindo o papel de controlador;
- Cada dispositivo deveria possuir o hardware necessário para realizar a codificação e/ou decodificação dos dados enviados e/ou recebidos;
- Em acréscimo às mensagens trocadas pelo UPnP, deveriam ser criadas mensagens específicas, listadas a seguir:
  - Identification: mensagem contendo a chave pública do dispositivo e o seu identificador;
  - GetPublicKey: mensagem de requisição contendo o identificador do dispositivo e a ação que ele deseja realizar;
  - PublicKeyResponse: mensagem contendo a chave pública de um determinado dispositivo (Control Point) registrado, caso ele tenha autorização para executar o serviço desejado;
  - AuthorizatonError: mensagem enviada pelo Controlador de Segurança ao Device indicando que o dispositivo (Control Point) não possui permissão para executar o serviço desejado. Em contrapartida, o dispositivo responde ao Control Point que originou a requisição com a mensagem UPnP padrão (SOAP) de erro.
- Essa proposta, para ser válida, deveria ter uma ampla aceitação pelos fornecedores do grupo UPnP.

Conclusão

Ambientes pervasivos serão cada vez mais presentes nos cenários futuros e as tecnologias que permitirão esse ambiente prosperar estão em pleno amadurecimento, permitindo a descoberta automática de serviços e redes de configuração zero. SLP, JINI, Salutation e UPnP são exemplos dessas tecnologias.

Por convergir o maior número de características desejáveis em um ambiente pervasivo ideal, a tecnologia UPnP encontra-se em vantagem diante das demais tecnologias, porém, essa posição pode mudar frente aos acontecimentos que estão por vir, influenciados pela concorrência mercadológica.

Entretanto, uma questão vital para o sucesso dos ambientes pervasivos ainda não está completamente madura: a segurança. Várias propostas isoladas estão sendo colocadas, embora nenhuma delas tenha sido considerada como um padrão para ser adotado amplamente.

Este artigo propõe, em linhas gerais, algumas características e forma de implementação de um esquema de segurança na arquitetura UPnP.

É esperado que em pouco tempo as lacunas que nos separam do ambiente pervasivo ideal sejam preenchidas, e a experiência humana com o uso de dispositivos eletrônicos mude radicalmente.

Agradecimentos

Agradeço principalmente ao meu professor, Doutor Orlando Loques, pelas valiosas orientações para a realização deste trabalho, e agradeço também a minha namorada, Adriana, por toda sua tolerância frente as minhas ausências nos nossos poucos momentos de disponibilidade.

Referências

[1] W3C Web Services Description Language (WSDL) Version 1.2. July, 9 2002.
http://www.w3.org/TR/wsdl12/

[2] Microsoft, Web Services Description Language Explained.
July, 2001.
http://msdn.microsoft.com/library/en-us/dnwebsrv/html/wsdlexplained.asp

[3] Microsoft Corporation. Universal Plug And Play Device Architecture.
Version 1.0. 2000.
www.upnp.org

[4] Universal Plug And Play Forum. www.upnp.org

[5] Mark Weiser, R.Gold, J.S.Brown. The origins of ubiquitous computing research at PARC in the late 1980s, IBM Systems Journal, vol.38, no. 4, 1999.

[6] Jini Architectural Overview, Technical White Paper, Sun Microsystems, 1999.

[7] Jini Network Technology Datasheet, Sun Microsystems, 1999.

[8] A.C. Huang, B.C.Ling, J.Barton, A.Fox. Making Computers Disappear: Appliance Data Services, MobiCom 2001 Rome, Italy

[9] uddi.org. UDDI Executive White Paper. November 14, 2001.
www.uddi.org

[10] uddi.org. UDDI Technical White Paper. September 6, 2000.
www.uddi.org

[11] Sandip H. Mandera. Web Services and UDDI: The New Wave of E-Business Renaissance
www.devx.com

[12] Sastry Duri, Alan Cole, Jonathan Munson, Jim Christensen. An Approach to Providing a Seamless End-User Experience for Location-Aware Applications, IBM, 2001.

[13] Christian Bettstetter and Christoph Renner. A comparison of service discovery protocols and implementation of the Service Location Protocol, Technische Universität München.

[14] Choonhwa Lee and Sumi Helal. Protocols for service discovery in dynamic and mobile networks, University of Florida, International Journal of Computer Research, Volume 11, Number 1, pp. 1-12.

[15] Robert E. McGrath. Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing, National Center for Supercomputing Applications, Department of Computer Science, University of Illinois, Urbana-Champaign.

[16] Golden G. Richard III. Service Advisement and Discovery: Enabling Universal Device Cooperation, University of New Orleans, Internet Computing, September, 2000.

[17] Yasser Rashed, Jim Edwards, Charlie Tai. Home Interoperability Framework for the Digital Home, Intel Technology Journal, Volume 6, Issue 4, pp. 5-16, November 15, 2002.

[18] Carl M. Ellison. Home Network Security, Intel Technology Journal, Volume 6, Issue 4, pp. 37-48, November 15, 2002.

[19] Sun Microsystems. Jini Network Technology Datasheet, 2001.

[20] SPKI e recursos relacionados.
http://world.std.com/~cme/html/spki.html

[21] UPnP Software Development Kit for Linux
http://upnp.sourceforge.net/

[22] Intel Tools for UPnP Technologies
http://www.intel.com/labs/connectivity/upnp/

Biografia do Autor

Marcio Belo (mbelo@ic.uff.br) é Analista de Sistemas da Empresa Municipal de Informática (IplanRio), professor de informática e aluno do curso de mestrado em ciência da computação da Universidade Federal Fluminense (UFF).
Sua página pessoal pode ser acessada pelo endereço http://www.ic.uff.br/~mbelo

* * * FIM * * *