Descoberta Semântica de Dispositivos em Ambientes de Computação Ubíqua

Rafael Besen, Frank Siqueira

Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

{rbesen,frank}@inf.ufsc.br

Abstract. *This paper describes a mechanism for semantic discovery of mobile devices in ubiquitous computing environments based on web services technology, called SeMoSD (Semantic Mobile Service Discovery). Through this mechanism, mobile devices are able to profit from de use of semantic technologies for service discovery. Ontologies are employed to describe services, which may be executed either on mobile or on fixed nodes. A semantic reasoner is responsible for processing service queries, being able to obtain more accurate results than syntactic searches.*

Resumo. *Este artigo descreve um mecanismo para descoberta semântica de dispositivos móveis em um ambiente de computação ubíqua baseado na tecnologia de serviços Web, denominado SeMoSD (Semantic Mobile Service Discovery). Por meio desse mecanismo, dispositivos móveis são capazes de tirar proveito do uso de tecnologias semânticas para descoberta de serviços. Ontologias são empregadas para descrever os serviços, que podem ser executados em nós móveis ou estacionários. Uma máquina de inferência semântica é responsável por processar as requisições de serviços, de forma a obter resultados mais precisos que os originados por buscas sintáticas.*

1. Introdução

Recursos providos por dispositivos computacionais podem ser compartilhados com outros dispositivos, com o objetivo de realizar tarefas cooperativamente. Para permitir a interoperabilidade em um ambiente computacional ubíquo [Weiser, 1993], dispositivos devem ser aptos a localizar uns aos outros na rede. Portanto, aplicações distribuídas em ambientes ubíquos requerem mecanismos de descoberta de serviços.

Uma abordagem amplamente empregada no compartilhamento de recursos é a arquitetura SOA (*Service Oriented Architecture*) [Erl 2005], criada para que dispositivos computacionais possam compartilhar recursos modelados na forma de serviços que podem ser invocados remotamente. No entanto, para um serviço ser compartilhado ele precisa ser descrito. Uma descrição básica de um serviço, indicando métodos e parâmetros sintaticamente, como ocorre usualmente, acaba limitando o entendimento do que o serviço realmente faz. Essa maneira simples de descrever recursos dificulta a busca de um serviço, visto que os termos da busca devem ser exatamente os mesmos utilizados para registrar o serviço, e que um mesmo termo pode possuir diferentes significados.

Criar uma descrição semântica de um serviço possibilita a utilização de conceitos comuns, tornando a busca mais eficaz. O uso de ontologias para descrição de serviços torna a descrição muito mais rica, porém essa tecnologia ainda é extremamente nova no ambiente de dispositivos móveis. Uma descrição semântica necessita de uma grande quantidade de recursos para ser processada, quantidade esta muitas vezes indisponível em dispositivos móveis, tornando a aplicação desta tecnologia em ambientes computacionais ubíquos um desafio.

O presente artigo descreve um mecanismo de descoberta de serviços que visa utilizar os benefícios da tecnologia semântica em ambientes de computação móvel e ubíqua. Este mecanismo se baseia no padrão DPWS (*Devices Profile for Web Services*) [OASIS 2009] para realizar a interação entre dispositivos. A descrição semântica dos serviços é feita utilizando ontologias baseadas em WSMO (*Web Services Modeling Ontology*) [ESSI 2006] e processada pelo ambiente de execução semântico WSMX (*Web Service Modeling eXecution environment*) [WSMX 2008].

O restante desse artigo é organizado da seguinte maneira: na seção 2 aborda o padrão DPWS, que possibilita o uso da tecnologia de serviços Web em dispositivos móveis. As tecnologias semânticas utilizadas no presente trabalho são apresentadas na seção 3. A arquitetura do mecanismo de descoberta semântica, sua dinâmica de execução e o protótipo desenvolvido são apresentados na seção 4. A seção 5 descreve e compara pesquisas semelhantes ao presente trabalho. A seção 6 apresenta os resultados obtidos nos testes realizados com o protótipo do mecanismo proposto e, por fim, a seção 7 analisa as contribuições do trabalho e apresenta sugestões para seu aprimoramento.

2. DPWS

A tecnologia de serviços Web vem sendo amplamente adotada para a construção de aplicações distribuídas e integração com sistemas legados. O uso de serviços Web em ambientes com dispositivos móveis, apesar de recente, vem se tornando cada vez mais comum devido à popularização desses dispositivos e à necessidade de interação destes com outros equipamentos conectados à Internet. A adoção desta tecnologia em dispositivos móveis ganhou um impulso com a definição do padrão DPWS (*Device Profile for Web Services*).

O DPWS, desenvolvido pela Microsoft e publicado pela OASIS (2009), define um modelo de comunicação baseado em serviços web com foco em dispositivos móveis. O padrão torna possível que dispositivos conectados a uma rede forneçam e divulguem serviços descritos sintaticamente para outros dispositivos. Com isso, qualquer dispositivo com conectividade e recursos de processamento – como smartphones, tablets, sensores e outros – pode implementar o padrão e ter seus serviços descobertos dinamicamente e invocados por outros dispositivos que entendam a sua descrição sintática.

DPWS define dois tipos básicos de serviços: *hosting services* e *hosted services*. O primeiro tipo representa o dispositivo propriamente dito; e o segundo corresponde aos serviços computacionais hospedados pelo *hosted service*. Cada *hosted service* tem seu próprio endereço de rede, que pode ser utilizado por outros dispositivos para contatá-lo e invocar seus *hosted services*.

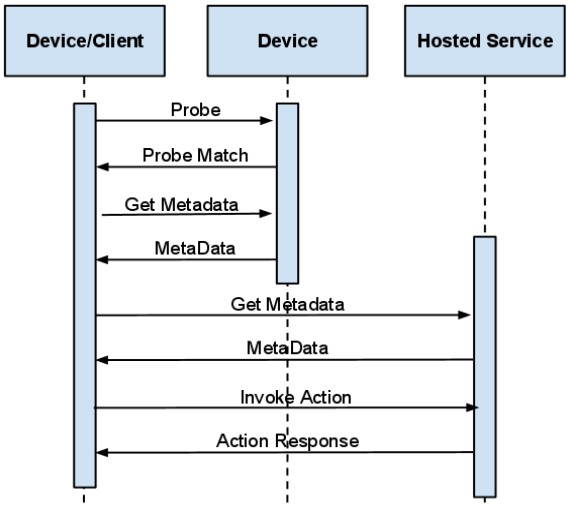

Figura 1. Fluxo DPWS

Na Figura 1 podemos observar o fluxo de interações entre um dispositivo cliente e um *hosting service* que possui um *hosted service*. Inicialmente o cliente envia uma requisição *Probe* para a rede através de *broadcast*, informando as características do serviço desejado. O dispositivo que possuir um serviço compatível responde à requisição com uma mensagem *Probe Match*. Ao receber o *Probe Match* de um ou mais dispositivos, o cliente envia uma mensagem *Get Metadata* diretamente para o *hosted service* escolhido para requisitar as informações necessárias para invocação do serviço. Depois de receber essas informações, o dispositivo pode finalmente invocar o serviço.

3. Serviços Web Semânticos

A utilização de tecnologia semântica pode otimizar a precisão da busca de serviços através do uso de ontologias. Serviços descritos através de ontologias são a base dos serviços web semânticos.

Conceitos pertencentes a um domínio específico e seus relacionamentos podem ser descritos através de uma ontologia. Com isso, os conceitos definidos na ontologia podem ser utilizados para descrever serviços e as mensagens trocadas entre eles. O diferencial de utilizar esses conceitos para descrever serviços é a possibilidade de prover o significado exato dos dados e especificar melhor o contexto de uma busca. Especificando esses conceitos dessa maneira, os tornamos interpretáveis por máquinas, eliminando a necessidade da intervenção humana em um processo de descoberta.

3.1. WSMO

Web Services Modeling Ontology [ESSI 2006] é uma tecnologia criada para a descrição de serviços web semânticos, que foi escolhida para uso neste trabalho por proporcionar uma expressividade superior em relação a outras tecnologias para o mesmo fim.

As interações e os dados trocados com um serviço são descritas usando ontologias, criadas utilizando a linguagem WSML (*Web Services Modeling Language*), que é a linguagem padrão para criação de artefatos WSMO. WSML é uma linguagem baseada em XML e utiliza URIs (*Uniform Resource Identifiers*) para identificar recursos compartilhados.

Artefatos WSMO devem definir o significado dos dados trocados nas mensagens enviadas entre serviços. Conceitos, relações e instâncias de uma ontologia são utilizados para a descrição de um serviço. WSMO possui quatro tipos de artefatos, sendo cada um responsável por descrever um aspecto de um serviço. São eles:

- *Ontology*: usada para definir os conceitos e relações empregados para descrever semanticamente os demais aspectos;
- *Service*: artefato que utiliza o que foi modelado na ontologia para descrever um serviço semanticamente;
- *Goal*: artefato utilizado para buscar um serviço empregando os conceitos e relações da ontologia para descrever semanticamente as características do serviço desejado;
- *Mediator*: tem o objetivo de garantir a interoperabilidade entre diferentes artefatos WSMO, lidando automaticamente com suas diferenças.

Conceitos e papéis são descritos separadamente em WSMO, permitindo que serviços e requisições sejam especificados em documentos diferentes, assim como a descrição e a implementação de um serviço. Essa estratégia visa promover o reuso e manter a compatibilidade com serviços sem características semânticas.

Para encontrar um serviço baseado em sua descrição semântica, deve ser criado um artefato WSMO chamado *goal*, o que contém as características desejadas em um serviço. O *goal* é processado por um *reasoner* semântico, que compara a descrição do *goal* com as descrições de serviços disponíveis. Como resultado, um conjunto com zero, um ou mais serviços compatíveis com a busca é retornado a quem realizou a busca.

3.2. WSMX

O *Web Services Modeling eXecution environment* [WSMX 2008] é o ambiente de execução do WSMO. O WSMX possui uma arquitetura que permite integrar sistemas existentes, através da criação de componentes que façam a intermediação necessária para integração, evitando assim alterações nos softwares existentes e mantendo o ambiente o máximo possível desacoplado.

O WSMX é composto pelo WSMX *Core*, que possui a base do sistema, e no qual *plugins* podem ser acoplados; por uma interface de gerenciamento de recursos, que é responsável por gerenciar cada novo recurso adicionado; e pelo *WSML2Reasoner Framework* [Grimm et al 2007], que é um *framework* de processamento semântico responsável por processar os *goals* e encontrar os serviços desejados. O usuário dispõe da ferramenta WSMT [DERI 2005], baseada no IDE Eclipse, cuja interface gráfica permite acessar as funcionalidades do ambiente de execução e criar elementos WSMO.

4. SeMoSD

O mecanismo proposto provê repositórios nos quais serviços semanticamente descritos podem ser armazenados e localizados, levando em consideração a existência de dispositivos com recursos computacionais limitados. Tais são incapazes de processar algoritmos de interpretação semântica, e ainda precisam lidar com o problema da intermitência da conexão. Por esse motivo, os repositórios semânticos devem ser mantidos em dispositivos com mais recursos e com conexão estável.

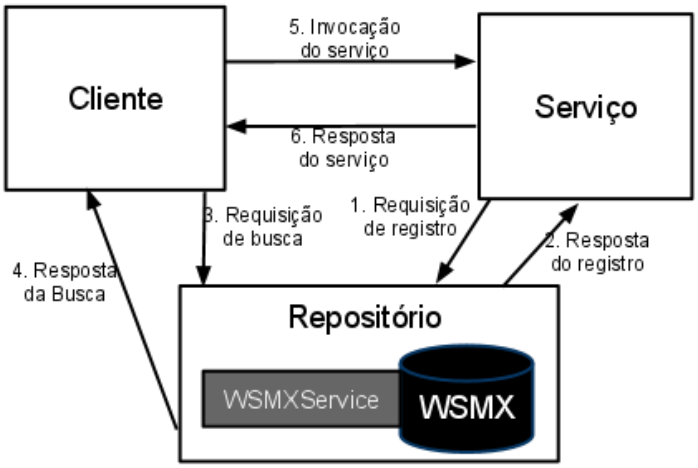

Figura 2. Funcionamento do Modelo

A estrutura básica para a troca de mensagens entre os serviços é o padrão DPWS, que provê mecanismos de comunicação para interação entre dispositivos móveis. O repositório que pode haver um ou mais presentes na rede se trata de uma instância do WSMX juntamente com um serviço DPWS que foi definido como "WSMXService" e tratará as requisições e vindas de clientes e outros serviços convertendo-as para WSML e enviando as requisições para o WSMX efetuar a operação necessária, ou ainda caso alguma outra operação seja necessária como uma verificação se um serviço está disponível ou ranking de serviços também pode ser realizado no WSMXService. A dinâmica do mecanismo pode ser melhor entendida na Figura 2.

Um serviço é semanticamente descrito através de conceitos e relacionamentos definidos em uma ontologia criada por meio do WSMO. O serviço envia sua descrição para o WSMXService (passo 1 na Figura 2), que trata essa requisição, armazena a descrição semântica do serviço no WSMX e responde ao serviço (passo 2). Para evitar que seja necessário efetuar qualquer processamento semântico no cliente, cujos recursos computacionais podem ser escassos, sugere-se adicionar uma descrição semântica para cada operação de um serviço, fazendo com que posteriormente uma requisição retorne não só o serviço desejado, mas também a operação específica que atende as necessidades do cliente.

A busca por serviços no repositório é realizada através de *goals*, que são entidades do WSMO. Os *goals* contêm uma descrição semântica da busca que será processada pelo repositório e comparada aos serviços armazenados. Um cliente que deseja buscar por um serviço deve enviar um *goal* ao WSMXService (passo 3 na Figura 2), que por sua vez repassa a requisição ao WSMX. Este executa o *matching* semântico e retorna os serviços compatíveis (passo 4).

No WSMXService os serviços encontrados são verificados e ordenados conforme dados históricos de disponibilidade, e então retornados ao cliente para que este possa executar a invocação. A política de invocação fica a critério do desenvolvedor do cliente, que pode escolher um ou mais serviços para serem invocados (passo 5 na Figura 2), e usar a primeira resposta obtida ou executar um algoritmo para seleção da resposta adequado ao tipo de aplicação desenvolvida (passo 6).

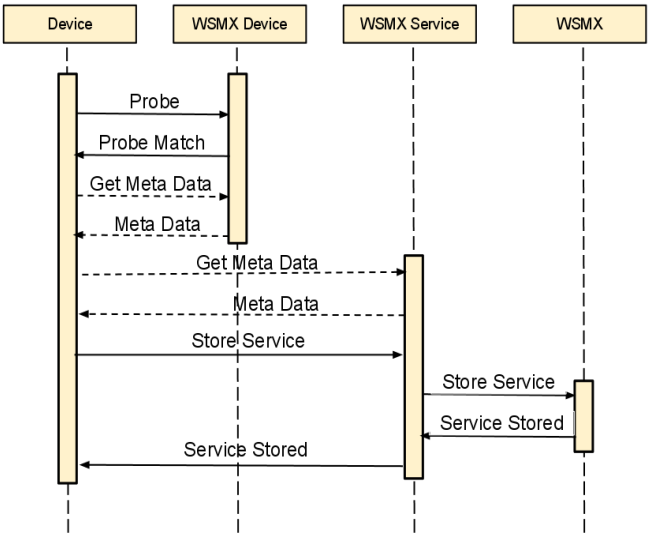

Figura 3. Serviço ao Entrar na Rede

Ao estabelecer uma política de invocação deve-se levar em consideração que em dispositivos móveis cada requisição tem um custo para a bateria do equipamento. Portanto, quanto maior o número de requisições enviadas, maior será o consumo da bateria do dispositivo.

A arquitetura criada é totalmente compatível com os serviços DPWS sintáticos, permitindo que estes invoquem e sejam invocados por serviços criados para o presente modelo. Isso permite que o sistema continue funcionando mesmo que de maneira sintática caso não seja encontrado nenhum mecanismo de descoberta semântica na rede.

4.1. Dinâmica de Execução

No modelo criado temos algumas sequências de atividades que devem ser seguidas para cada componente criado: o processo de um novo serviço entrando na rede; a entrada de um cliente na rede; e a busca e invocação de um serviço.

Ao entrar em uma rede, um serviço deve registrar sua descrição semântica em um repositório para que possa ser encontrado pelos clientes. O primeiro passo para execução dessa tarefa é emitir um DPWS *Probe* buscando pelo WSMXService. Ao encontrar o serviço do repositório, o serviço emite uma requisição para obter os metadados do WSMXService. De posse desses metadados, o serviço invoca a operação para armazenar sua descrição semântica no repositório. Ao receber essa requisição o WSMXService converte os parâmetros recebidos para o formato requerido pelo WSMX e envia a requisição. Assim, o serviço é armazenado e fica passível de uma busca posteriormente. Essas etapas podem ser visualizadas na Figura 3.

Já um cliente, ao entrar na rede, deve agir da mesma maneira que um serviço. Primeiramente, o cliente precisa encontrar pelo menos um repositório disponível. Portanto, o processo inicial é o mesmo, ou seja, o cliente envia a mensagem *Probe* buscando pelo WSMXService e, a cada resposta recebida, a referência do repositório fica armazenada para uma invocação posterior. A Figura 4 ilustra o processo do cliente buscando o WSMXService.

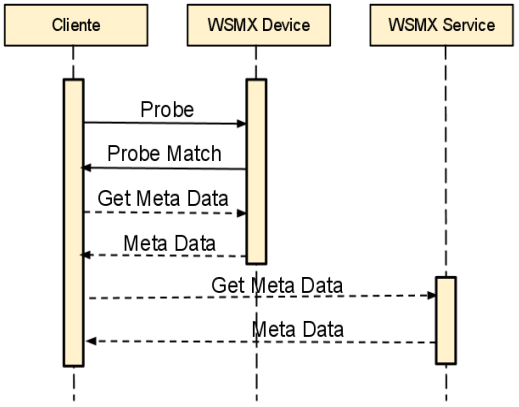

Figura 4. Cliente ao Entrar na Rede

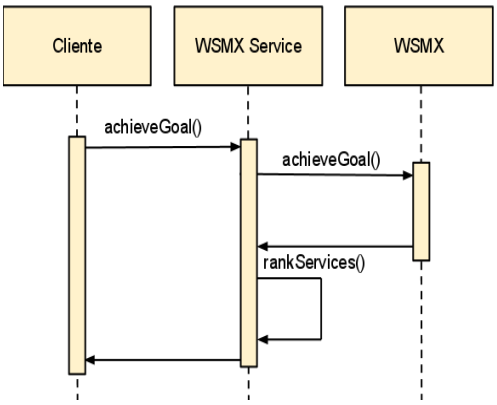

Figura 5. Busca por um Serviço

Depois de localizar os repositórios presentes na rede, o cliente fica apto a realizar a busca semântica de serviços, ilustrada na Figura 5. Na busca o cliente faz uma requisição ao WSMXService, passando por parâmetro o *goal* com as características de serviço desejado. O WSMXService, por sua, vez converte essa requisição para a linguagem do WSMX e requisita que o *goal* seja executado. Caso existam serviços compatíveis, esses serão retornados ao WSMXService que realiza o processo de ordenação dos serviços baseado em sua disponibilidade. A lista com os serviços ordenados é encaminhada para o cliente, que poderá então realizar a invocação destes. Já o processo de invocação do serviço consiste em uma requisição DPWS comum, ilustrada na Figura 1 do presente artigo.

4.2. Implementação do Protótipo

O protótipo implementado foi desenvolvido na linguagem Java, versão 6, escolhida devido ao vasto suporte a dispositivos móveis e à fácil integração com o WSMX. A integração com o WSMX é realizada através de *web services* e de uma API de integração em Java. Foi utilizada a versão 0.5 do WSMX.

A *WSMX Integration API* é usada para realizar as requisições ao repositório. As requisições são traduzidas para WSML pelo WSMXService utilizando a biblioteca WSMO4J. Ao obter uma resposta do WSMX contendo um ou mais serviços, o WSMXService verifica quais desses serviços estão disponíveis. Os dados de disponibilidade são armazenados e levados em consideração na ordenação dos serviços, colocando no topo da lista os serviços com maior disponibilidade.

Os clientes e serviços DPWS foram implementados utilizando a biblioteca Java do WS4D, que implementa o DPWS utilizando JMEDS. Essa biblioteca foi escolhida por ser a mais utilizada e melhor documentada.

5. Trabalhos Relacionados

Existem diversos protocolos já estabelecidos para descoberta de serviço, como Jini [Sun 2003], OSGi [OSGi Alliance, 2010], SLP [SLP Working Group, 2002], UPnP [UPnP Forum, 2003], etc. Contudo, estes protocolos possuem diversas limitações. Existem diversos esforços no sentido de estender a capacidade destes protocolos para aprimorar o processo de descoberta. A seguir são apresentados alguns destes esforços.

Tabela 1. Comparação dos Trabalhos Relacionados

Trabalho	Comunicação	Restrições de linguagem/ambiente	Escopo	Suporte a Semântica	Suporte a Mobilidade
AIDAS	WS	Java	Qualquer	Parcial	Limitado
Home SOA	OSGi	Java	Mídia residencial	Total	Total
P2P-Based Semantic Service	SLP	Suporte a SLP	Redes ad-hoc	Total	Requisições centralizadas
SeMoSD	WS	Nenhuma	Qualquer	Total	Total

Home SOA [Bottaro e Gérodolle, 2008] foi projetado para ser utilizado em ambientes com poucos dispositivos, principalmente dispositivos multimídia. Utiliza descrição semântica para descrição dos dispositivos e serviços oferecidos. A comunicação do modelo é realizada através da tecnologia OSGi, pois no trabalho julga-se que *web services* não são ideais para tal ambiente visto que a comunicação gera um grande tráfego. Contudo, fazendo essa escolha o modelo fica limitado a linguagens suportadas pela máquina virtual Java.

O projeto AIDAS [Toninelli et al., 2007] utiliza dados semânticos para descrever serviços, porém para o objetivo de criar um ambiente ubíquo ele possui algumas limitações. A primeira delas é o suporte limitado a dispositivos móveis, os quais são usados apenas para obter informações dos usuários e criar contextos. Portanto, a descoberta e a invocação de serviços em dispositivos móveis não são viabilizadas. Outra limitação do modelo é que, assim como Home SOA, ele se limita à plataforma Java, não suportando serviços em ambientes heterogêneos.

Por fim, o trabalho chamado *P2P-Based Semantic Service* [Baumung et al., 2006] apresenta um modelo semântico de gerenciamento de serviços. O modelo se trata de uma rede *ad-hoc* se comunicando através de protocolos P2P. Por ser limitado a redes *ad-hoc*, o modelo não suporta uma grande quantidade de dispositivos. As requisições são centralizadas em um *pool*, portanto todas as requisições são centralizadas. O *pool* é construído com base no protocolo SLP.

Conforme ilustrado na Tabela 1, os trabalhos descritos não são flexíveis o suficiente ou não levam em consideração algum aspecto importante para a criação de um ambiente ubíquo. Já o mecanismo proposto no presente artigo mostra-se capaz de cobrir os aspectos mais importantes apresentados na tabela.

6. Resultados Experimentais

Para validar o mecanismo criado foram realizados alguns testes, não apenas para verificar o seu correto funcionamento, mas também para medir os tempos de execução de cada etapa do processo.

O ambiente utilizado para rodar os testes é composto por diversos componentes. O WSMX, versão 0.5, foi executado um uma máquina virtual com Windows XP com 1Gb de memória RAM e 1 CPU, na qual foram armazenadas 25 ontologias, 40 serviços e 36 *goals*.

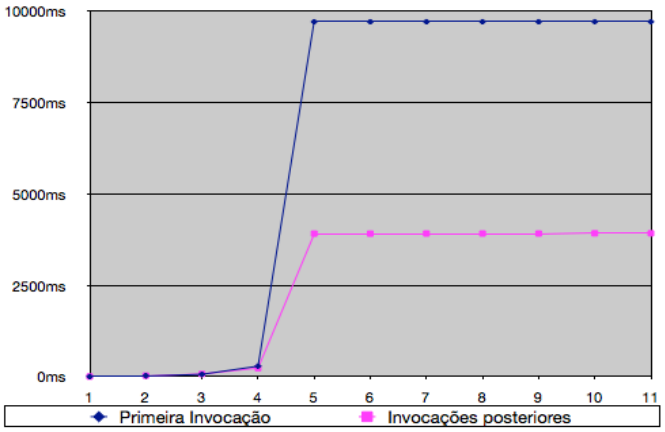

Figura 6. Resultados dos testes

O WSMXService e a aplicação cliente foram executadas em um notebook com 4Gb de memória RAM e processador Intel Core2Duo 2.4GHz. Outros serviços, inclusive o serviço a ser localizado, foram executados em um notebook com 3Gb de memória RAM e processador Intel Core2Duo 2.26GHz. Os dispositivos móveis foram emulados nos notebooks utilizando recursos da biblioteca WS4D. Toda a comunicação foi realizada via rede wireless IEEE 802.11g.

Os testes realizados visaram medir o tempo de descoberta do WSMXService, o tempo de invocação do WSMXService, o tempo de invocação e processamento semântico do WSMX, o tempo de busca do serviço retornado e o tempo de invocação do mesmo. Nas medições foram observadas algumas questões importantes, principalmente referentes ao WSMX. Na primeira vez que um *goal* é executado, o tempo de processamento é cerca de três vezes mais alto do que das execuções posteriores. Tal fato é justificado pela existência de uma *cache* no WSMX para diminuir o processamento necessário e acelerar o processo de busca. Na Figura 6 essa característica pode ser visualizada.

O ponto 1 mostrado no gráfico representa o início da busca pelo WSMXService; no ponto 2 o WSMXService foi encontrado; no ponto 3 a requisição DPWS é enviada para invocação do serviço; no ponto 4 o WSMX é invocado; e no ponto 5 o WSMXService recebe a resposta do WSMX. O ponto 6 representa quando o cliente recebe a resposta do WSMXService; no ponto 7 inicia a busca pelo serviço; no ponto 8 as configurações do serviço são obtidas; e no 9 o serviço é invocado. O ponto 10 corresponde à invocação do serviço e o ponto 11 representa a resposta sendo entregue ao cliente, encerrando a execução.

Os resultados obtidos representam a média dos testes, primeiramente executados com a configuração descrita anteriormente. Ao adicionar mais 40 serviços DPWS na rede, o tempo médio de invocação não teve alterações significativas. Para tentar diminuir o tempo de resposta do WSMX foi dobrada a quantidade de memória RAM da máquina virtual na qual este foi executado, mas o tempo de execução não teve alterações.

Como pode ser observado no gráfico, a maior parte do tempo consumido na operação se deve ao processamento semântico. Todo o restante da operação leva em média apenas 300 milissegundos.

7. Conclusões e Trabalhos Futuros

O presente trabalho fornece uma infraestrutura para descoberta semântica de serviços em ambientes com dispositivos móveis. As características semânticas do WSMO, combinadas com a facilidade de comunicação do DPWS, possibilitam a criação de um mecanismo de descoberta semântica para um ambiente de computação ubíqua.

Um possível aprimoramento do mecanismo proposto consiste na otimização do algoritmo de ordenação de serviços por disponibilidade. No entanto, a principal questão a ser aprimorada é a diminuição do tempo de processamento semântico. Possivelmente a versão 1.0 do WSMX, que durante o desenvolvimento deste trabalho estava em versão beta, deverá melhorar esse tempo de resposta, pois teve seu código em grande parte reformulado. Esta versão beta foi utilizada devido à falta de documentação e a problemas de estabilidade, mas deve trazer importantes aprimoramentos, como a ordenação dos serviços localizados e um componente para monitoramento dos serviços.

Referências

Erl, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.

OASIS. DPWS v 1.1, July 2009. Available at http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

ESSI WSMO working group. WSMO version 1.3, October 2006. Available at http://www.wsmo.org/TR/d2/v1.3/20061021/

WSMX version 0.5, May 2008; Available at http://www.wsmx.org/.

DERI, WSMT v0.1, January 2005, Disponível em http://www.wsmo.org/TR/d9/d9.1/v0.1/20050127/

Grimm, S.; Keller, U.; Lausen, H. and Nagypal, G. A Reasoning Framework for Rule-Based WSML. In Proceedings of 4th European Semantic Web Conference (ESWC), Innsbruck, Austria, 2007

Sun Microsystems. Jini Technology Core Platform Specification, v. 2.0, June 2003. Available at www.sun.com/software/jini/specs/core2_0.pdf

OSGi Alliance. OSGi Specification R4 Enterprise Version 4.2, March 2010. Available at http://www.osgi.org/Release4/HomePage

SLP Working Group. Service Location Protocol(SLPv2), August 2002. Available at http://srvloc.sourceforge.net/new_drafts/draft-guttman-svrloc-rfc2608bis-03.txt

UPnP Device Architecture 1.0, UPnP Forum, Dec. 2003; Available at http://www.upnp.org/resources/documents/CleanUPnPDA10120031202s.pdf

A. Bottaro and A. Gérodolle. Home SOA – Facing Protocol Heterogeneity in Pervasive Application ; ICPS'08, July 6–10, 2008, Sorrento, Italy.

A. Toninelli, A. Corradi, and R. Montanari, "Semantic-based discovery to support mobile context-aware service access", Computer Comunication, 31: pp. 935-949, January 2008, DOI:10.1016/j.comcom.2007.12.026.

P. Baumung, S. Penz and M. Klein, "P2P-Based Semantic Service Management in Mobile Ad-hoc Networks", MDM '06 Proceedings of the 7th International Conference on Mobile Data Management, 2006, Washington, DC, USA.

M. Weiser. Hot Topics: Ubiquitous Computing. IEEE Computer, 26(10):71– 72, October 1993.