UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Um Protocolo de Descoberta de Serviço para Sistemas Sensíveis ao Contexto

RICARDO RIOS MONTEIRO DO CARMO

Vitória, Maio de 2006

RICARDO RIOS MONTEIRO DO CARMO

Um Protocolo de Descoberta de Serviço para Sistemas Sensíveis ao Contexto

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do título de Mestre em Informática.
Orientador: Prof. Dr. José Gonçalves Pereira Filho

Vitória
Maio de 2006

RICARDO RIOS MONTEIRO DO CARMO

Um Protocolo de Descoberta de Serviço para Sistemas Sensíveis ao Contexto

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 12 de maio de 2006, por:

Prof. Dr. José Gonçalves Pereira Filho
UFES - Universidade Federal do Espírito Santo

Prof. Dr. Wanderley Lopes de Souza
UFSCar - Universidade Federal de São Carlos

Prof. Dr. José Laurindo Campos dos Santos
INPA - Instituto Nacional de Pesquisas da Amazônia

Prof. Dr. Sérgio Andrade Freitas
UFES - Universidade Federal do Espírito Santo

Universidade Federal do Espírito Santo
Vitória - Espírito Santo
Maio de 2006

Dedico este trabalho a todos que de modo direto ou indireto têm me auxiliado a crescer profissional e espiritualmente.

Agradecimentos

Agradeço a Deus por ter permitido e me auxiliado a conseguir realizar este trabalho.

À minha mãe e meu irmão Alessandro, pelo amor, dedicação, orações e apoio incondicional.

À Luciana, por seu amor, compreensão, dedicação, incentivo e apoio, que me ajudaram a vencer mais esta etapa.

A Palmiro, Yasmin, Carolina, Vivian, Yasmin Filha, Conceição e Tânia. Pessoas queridas que reencontrei e que fazem parte de minha família espiritual. Sem o auxílio delas não teria conseguido ultrapassar muitas barreiras e angariar mais experiência de vida durante minha estada em Vitória.

À Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) pelo suporte financeiro.

Durante a realização deste trabalho tive a grande oportunidade de fazer parte da equipe do Laboratório de Pesquisas em Redes e Multimídia da UFES, onde pude fazer novas amizades e desfrutar de momentos de crescimento acadêmico e profissional.

Ao Prof. José Gonçalves Pereira Filho, pelo seu trabalho de orientação e de apoio. Agradeço sua disponibilidade e incentivo, principalmente nos momentos finais do trabalho. Obrigado pela oportunidade de trabalho em conjunto. Mais que um orientador, mostrou-se um amigo, sempre disposto a me ajudar em momentos cruciais durante minha estada em Vitória.

Ao Prof. Laurindo Campos, amigo e incentivador de meu crescimento acadêmico e profissional. Sem o seu apoio não teria conseguido vencer mais esta etapa de minha educação.

Aos amigos Rodrigo Bonfá e André Costa, pessoas honestas, íntegras e companheiras, que me auxiliaram na materialização deste trabalho, participando e contribuindo com idéias muito importantes nos vários momentos que tivemos para conversar sobre o trabalho.

Aos amigos Cristiano Biancardi, Gustavo Gaburro e Leonardo Silvestre pelo companheirismo e pelos momentos de alegria.

Aos amigos do LPRM: Renzo Quedevez, Camilo Calvi, Rodrigo Mantovaneli, Roberto Menequini, Adilson Cruz, Filipe Pandolfi, Thiago Nico, Luiz Rodrigo, Natália Quirino, Magnos Martinello, Roberta Lima Gomes e muitos outros que passaram pelo laboratório deixando sua contribuição para o crescimento do grupo.

Aos professores Álvaro Barbosa, Anilton Garcia e Saulo Bortolon pelo incentivo, pelas conversas e contribuições que direta e indiretamente deram.

A todos os professores do Departamento de Informática, a Dirlene, Ivani e Johnathan que estiveram sempre disponíveis a me auxiliar nos vários momentos que os procurei.

Finalmente e não menos importante, agradeço a todas as pessoas com as quais pude conviver, tanto no plano profissional, quanto no pessoal, que me acolheram de braços abertos durante o tempo que morei em Vitória.

Muito obrigado e que Deus abençoe a todos.

“O homem só é grande, só tem valor pelo seu pensamento; por ele suas obras irradiam e se perpetuam através dos séculos.”

Léon Denis

Resumo

O avanço da computação móvel e da tecnologia da comunicação sem fio facilitaram o surgimento de novos paradigmas. Um destes é a *Computação Ubíqua* ou *Pervasiva*, caracterizada por um ambiente altamente dinâmico, ocasionado pela mobilidade de seus usuários e pela utilização de dispositivos móveis, como PDAs e aparelhos de telefonia celular.

A Computação Ubíqua permite explorar um novo conjunto de aplicações, dentre elas as *Aplicações Sensíveis ao Contexto*. Essas aplicações manipulam informação contextual possibilitando o desenvolvimento de serviços mais flexíveis, adaptáveis, ricos em funcionalidades e centrados no usuário. Entretanto, para que as aplicações sensíveis ao contexto possam se beneficiar de tais serviços, é necessária uma infra-estrutura de suporte preparada para lidar com a natureza da informação contextual.

Dentre as várias infra-estruturas propostas, a *Infraware*, em desenvolvimento no Laboratório de Pesquisas em Redes e Multimídia da UFES, propõe um modelo de gerência integrada de serviços que enfatiza a utilização de descrições semânticas para composição e descoberta dinâmica de serviços. Em particular, a descoberta de serviços tem um papel fundamental, pois é através dela que aplicações encontram os serviços necessários para executar suas tarefas.

Nos ambientes caracterizados pela computação pervasiva, a descoberta de serviços deve considerar a natureza da informação contextual. Poucos mecanismos e protocolos de descoberta consideram informação contextual como atributo de seleção. Além disso, alguns deles não consideram segurança e autenticação de dados. Este trabalho propõe um protocolo de descoberta de serviços para a plataforma *Infraware*, seguro e sensível ao contexto, denominado *SCaSDP - Secure Context-aware Service Discovery Protocol*, parte do módulo *Gerente de Serviços*. Este protocolo apresenta uma arquitetura genérica que permite a sua utilização em cenários diversos de descoberta de serviços.

Abstract

Advances in mobile computing and wireless technology are allowing the raising of new computing paradigms. One of them is the *Ubiquitous* or *Pervasive Computing*, which is characterised by a highly dynamic environment caused by the user mobility and by the massive use of small mobile devices like PDAs and cellular phones.

Ubiquitous Computing allows to explore a new set of applications, the *Context-Aware Applications*. Theses applications use contextual information, allowing more flexible, adaptable, rich in functionality and user-centred services to be developed. However, context-aware applications require an infrastructure prepared to deal with the nature of the contextual information is needed.

Among the many proposed context-aware infrastructures, Infraware has been developed in the Network and Multimedia Research Laboratory of the Federal University of Espírito Santo. Infraware proposes an integrated management model of services that emphasises composition and discovery of services based on semantic description. In particular, the service discovery has an important role because it is used by the applications to find the right services to execute their tasks.

In Pervasive Computing environments, service discovery must consider the nature of the contextual information. Few discovery mechanisms and protocols consider contextual information in the selection process. Moreover, some of them do not consider security and authentication of data. This master thesis proposes a secure and context-aware service discovery protocol, named SCaSDP, which is part of the Service Manager Module of the Infraware platform. This protocol presents a generic architecture, allowing its use in many situations of service discovery.

Lista de Siglas

ARPANet	*Advanced Research Projects Agency Network*
CA	*Certification Authority*
DA	*Directory Agent*
DAML	*DARPA Agent Markup Language*
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name System*
DTD	*Document Type Definition*
FIPA	*Foundation for Intelligent Physical Agents*
GUA	*Graphical User Authentication*
HTTP	*Hyper Text Transfer Protocol*
IETF	*Internet Engineering Task Force*
INPA	Instituto Nacional de Pesquisas da Amazônia
IP	*Internet Protocol*
IrDA	*Infrared Data Association*
LBA	*Experimento de Grande Escala da Biosfera-atmosfera da Amazônia*
LDAP	*Light Weight Directory Access Protocol*
LPRM	Laboratório de Pesquisas em Redes e Multimídia
MAC	*Media Access Control*
MCT	Ministério de Ciência e Tecnologia
MTU	*Maximum Transfer Unit*
NIC	*Network Information Center*
OO	Orientação Objeto
OWL	*Ontology Web Language*
OWL-S	*Ontology Web Language - Service*
PDA	*Personal Data Assistant*
PIPE	*Platform Independent Petri Net Editor*
PKC	*Public-Key Cryptography*
PKI	*Public-Key Infrastructure*
PKIX	*Public-Key Infrastructure X.509*
QoS	*Quality of Service*
RFC	*Request for Comments*
RMI	*Remote Method Invocation*
SCaSDP	*Secure Context-aware Service Discovery Protocol*
SGBD	Sistema Gerenciador de Banco de Dados
SLM	*Service Locating Manager*
SLP	*Service Location Protocol*
SPDP	*Secure Pervasive Discovery Protocol*
SPKI	*Simple Public-Key Infrastructure*

SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SSDS	*Secure Service Discovery Service*
TCP	*Transfer Control Protocol*
TCP/IP	*Transfer Control Protocol/Internet Protocol*
UA	*User Agent*
UDDI	*Universal Description, Discovery, and Integration*
UDP	*User Datagram Protocol*
UFES	Universidade Federal do Espírito Santo
UMLS	*Unified Medical Language System*
UPnP	*Universal Plug-and-Play*
URL	*Unified Resource Location*
WASP	*Web Architecture for Service Platform*
WSDL	*Web Service Definition Language*
XML	*eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Motivação

A evolução dos sistemas computacionais pode ser divida em três eras: era do *mainframe*, era do computador pessoal e a era da *Internet*. Esta última incentiva a proliferação da computação distribuída, permitindo a disseminação da informação e a colaboração e interação entre as pessoas. Com o avanço da computação móvel e a tecnologia da comunicação sem fio, a computação distribuída ganha uma nova dimensão, propiciando o surgimento de novos paradigmas.

A última década possibilitou o surgimento de um desses novos paradigmas computacionais que vem ganhando espaço em relação ao paradigma tradicionalmente estático, relativamente previsível e baseado em estações de trabalho. Esse novo paradigma é caracterizado por um ambiente altamente dinâmico, ocasionado pela mobilidade de seus usuários e pela utilização de dispositivos multi-funcionais, como PDAs e aparelhos de telefonia celular. Esta mudança permite o aparecimento da *Computação Ubíqua*, introduzida por Mark Weiser [Weiser 1991]. Weiser vislumbrou sistemas e ambientes onde recursos computacionais são capazes de prover serviços e informações quando e onde sejam requeridos pelos usuários. Isto propõe uma integração contínua (*everywhere, everytime*) entre ambiente e tecnologia, onde as pessoas são auxiliadas por dispositivos computacionais nas suas mais variadas atividades e são capazes de interagir com eles mais naturalmente, ou seja, abstraindo os detalhes operacionais característicos desses dispositivos.

A Computação Ubíqua, também referenciada como computação pervasiva (*Pervasive Computing*) ou tecnologia calma (*calm technology*), traz consigo a possibilidade de explorar um novo conjunto de aplicações, as *aplicações sensíveis ao contexto*. Essas aplicações utilizam não apenas as informações providas por usuários, mas também as informações implícitas (não conhecidas pelos usuários), altamente dinâmicas, extraídas do ambiente onde se encontram. Essas informações, ditas contextuais, são utilizadas no processamento e influenciam na tomada de decisões das aplicações. Assim, ao invés de tratar mobilidade como um problema, as aplicações sensíveis ao contexto exploram essa característica, levando em consideração a natureza dinâmica e contextual da informação presente no ambiente, com o objetivo de criar serviços mais flexíveis, adaptáveis, ricos em funcionalidade e centrados no usuário.

A criação de aplicações sensíveis ao contexto pressupõe a existência de uma infra-estrutura de suporte preparada para lidar com a característica dinâmica da informação contextual. Há na literatura propostas de plataformas e de infra-estruturas de *mid-*

dleware voltadas ao desenvolvimento de aplicações sensíveis ao contexto, como [Dey 2000, Indulska et al. 2001, Chen e Kotz 2002, Huang 2002, Costa 2003, Gu et al. 2004]. Essas plataformas têm por objetivo auxiliar o projetista a conceber aplicações que utilizem serviços, mecanismos e interfaces que escondam a complexidade introduzida pela manipulação da informação contextual.

Um desses projetos é o *Infraware*, um *middleware* de suporte às aplicações sensíveis ao contexto, em desenvolvimento no Laboratório de Pesquisas em Redes e Multimídia (LPRM) da Universidade do Espírito Santo (UFES). O *Infraware* é uma extensão do projeto holandês *WASP* [WASP Project 2003] e se utiliza de conceitos e tecnologias relacionados à *Web* Semântica, particularmente a utilização de ontologias para manipulação e tratamento da informação contextual. Além disso, introduz novos componentes responsáveis por tratar questões relacionadas com a aquisição e a integração de dados contextuais heterogêneos e distribuídos, a resolução de conflitos e coordenação entre aplicações, a privacidade e segurança e a gerência integrada de serviços com descrição semântica.

No *Infraware* as descrições semânticas dos serviços se baseiam na utilização de ontologias para especificar modelos formais extensíveis, os quais descrevem também o domínio das aplicações. Esta abordagem permite prover meios de configurar as interações entre as aplicações e a plataforma em tempo de execução. Novos serviços e entidades podem ser adicionados através da extensão das ontologias existentes e da inclusão de novas ontologias, que descrevem os modelos utilizados, o que conseqüentemente estende a *Infraware*. A gerência dos serviços na *Infraware* é uma atividade dinâmica, que envolve a composição e a descoberta de serviços.

Em um ambiente dinâmico, como o caracterizado pela computação sensível ao contexto, a descoberta de serviço tem um papel fundamental, pois facilita o gerenciamento dinâmico dos serviços disponíveis nos diversos dispositivos computacionais presentes. Algumas questões, entretanto, ainda requerem maior atenção. Em particular, apesar de alguns considerarem a segurança dos dados de usuários e serviços, muitos mecanismos e protocolos de descoberta propostos até hoje são pobres em lidar com as características peculiares da computação sensível ao contexto, não considerando a natureza contextual da informação e não permitindo a utilização de descrições de serviços mais ricas em semântica. Este fato justifica o desenvolvimento de pesquisas na área e abre espaço para o surgimento de outras propostas de protocolos, mecanismos e algoritmos que incorporem funcionalidades para lidar com a descrição semântica e a segurança dos serviços e aplicações sensíveis ao contexto.

1.2 Objetivo Geral

Este trabalho tem por objetivo desenvolver o módulo de descoberta de serviços do Gerente de Serviços da plataforma *Infraware*. Em particular, visa desenvolver um protocolo de descoberta semântica de serviços que utilize a natureza contextual da informação e que proveja segurança das informações de usuários e serviços. Uma característica desta proposta é a utilização de ontologias para descrever semanticamente os serviços providos pelos dispositivos computacionais. A segurança é garantida pela integridade, confidencialidade e não-repúdio das mensagens trocadas e pela utilização de mecanismos de autenticação entre as entidades.

1.3 Metodologia

O desenvolvimento do trabalho está estruturado sobre as seguintes etapas/atividades:

1. Um estudo inicial sobre Computação Sensível ao Contexto e conceitos relacionados;

2. Levantamento e análise de requisitos de protocolos de descoberta de serviços, visando identificar os principais requisitos funcionais desta classe de protocolos. Para isso, alguns conceitos relacionados à descoberta de serviço e alguns protocolos de descoberta são analisados, bem como outras propostas de infra-estruturas e ambientes de descoberta que contribuam para o desenvolvimento do trabalho. As características inerentes à computação ubíqua são particularmente observadas com o intuito de determinar características específicas do protocolo proposto. Para auxiliar na identificação dos requisitos é tomado como cenário de uso um dos sítios de pesquisa do Projeto de Grande Escala da Biosfera-Atmosfera da Amazônia (LBA), localizado em Manaus, Amazonas. Esse sítio retrata muito bem um ambiente *ad hoc*, onde as informações possuem característica altamente dinâmica;

3. Projeto e validação do protocolo. O projeto utiliza os requisitos definidos na fase de análise de requisitos e a validação é feita através de metodologias propostas pelas Redes de Petri e máquinas de estados finitos comunicantes;

4. Implementação, testes e análise dos resultados.

1.4 Estrutura da Dissertação

O restante deste trabalho está assim organizado:

O capítulo 2 apresenta uma introdução geral ao tema *Computação Sensível ao Contexto*, apresentando seus principais conceitos e tecnologias.

O capítulo 3 discute questões que envolvem a descrição, a organização, a classificação e a seleção de serviços, bem como alguns conceitos relacionados à segurança de informações.

O capítulo 4 introduz alguns protocolos e propostas de descoberta de serviços existentes na literatura, cujas características são usadas para o entendimento dos mecanismos de descoberta de serviços.

O capítulo 5 apresenta a proposta de um novo protocolo de descoberta de serviços, baseado nos requisitos definidos no capítulo anterior, seguro e sensível ao contexto, para a arquitetura *Infraware*.

O capítulo 6 apresenta a especificação formal do protocolo proposto, utilizando Redes de Petri e Máquinas de Estados Finita.

No capítulo 7 descreve a implementação de um protótipo e apresenta o cenário de testes utilizado.

O capítulo 8 conclui o trabalho, apresentando as considerações finais e relacionando as propostas de trabalhos futuros.

Capítulo 2

Sensibilidade ao Contexto

2.1 Introdução

Este capítulo apresenta uma breve introdução ao tema contexto. São discutidos seus conceitos básicos e como vem sendo utilizado no projeto de plataformas de serviços, em particular da plataforma *Infraware*, uma infra-estrutura de suporte ao desenvolvimento de aplicações sensíveis ao contexto. A descrição semântica de serviços é suportada através da utilização de ontologias, que são brevemente apresentadas ao final do capítulo.

2.2 Contexto

A palavra contexto tem origem no latim, da palavra *contextu*, que significa encadeamento das idéias de um texto. Da gramática vem a definição de "enquadramento sintagmático de uma unidade do discurso; situação de comunicação; argumento". Essas definições são muito bem observadas no dia-a-dia do homem.

O ser humano sempre procurou se comunicar para expressar seu pensamento. Esta expressão é direcionada segundo um determinado assunto e/ou contexto, que dita o curso da conversa. Por exemplo, quando um grupo de pessoas assiste a um programa de TV, a conversa pode estar direcionada para o assunto tratado no programa naquele momento; em uma conversa entre amigos, uma idéia, um assunto, é foco das atenções.

Dependendo do contexto, a comunicação é feita adaptando-se o modo de agir, de se expressar, de acordo com a necessidade. Em uma sala de cinema, por exemplo, as pessoas provavelmente falam entre si sussurrando. O contexto, portanto, não determina apenas sobre o que se está conversando, mas também a forma de expressão.

Há vários anos o tema contexto tem sido um assunto muito pesquisado em algumas comunidades científicas, como a Lingüística e a Psicologia Congnitiva. Na área da Computação as primeiras pesquisas, feitas pela comunidade de Inteligência artificial, datam da década de 90 e contribuíram para o entendimento e a formalização do termo [Guha 1991, McCarthy e Buvac 1994]. Além desta, várias outras áreas investigaram sua utilização na computação, como a de Interação Homem-Máquina [Hao e Selker 2000], a de Computação Móvel [Huang 2002], a de Computação Pessoal e Ubíqua [K.Dey et al. 2001], a de Recuperação de Informação [Brewington et al. 1999, Friedrich, Shimkat e Küchlin 2002] e a de Sistemas Operacionais [Bulusu]. Um exemplo da aplicação de contexto em computação pode ser observado na utilização de páginas de busca na *Web*. Uma pesquisa feita com

base em informações providas por um usuário será mais precisa quanto mais informações forem utilizadas. Esse tipo de consulta é chamada de *consulta contextual*.

Apesar de muito explorado, o termo contexto ainda não possui uma definição comumente aceita. Uma definição amplamente utilizada é a fornecida por [Dey. e Abowd 1999]: *contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade*. Outras propostas se utilizam da mesma abordagem ou enumeram características do ambiente para definir contexto.

Neste trabalho, contexto é interpretado como o estado de um ambiente em um determinado momento, caracterizado por um conjunto de informações das entidades envolvidas e do inter-relacionamento entre essas entidades. Essas entidades podem ser pessoas, dispositivos computacionais ou qualquer outro objeto relevante para caracterizar o estado do ambiente, que pode se repetir no decurso do tempo.

2.3 Informação Contextual

A informação contextual é utilizada para caracterizar o estado de um ambiente em um determinado momento. Alguns exemplos de informação contextual são: luminosidade, nível de barulho, conectividade de rede e custo de comunicação [Schilit, Adams e Want 1994].

Com base em uma classificação chamada de "cinco Ws", [Abowd e Mynatt 2000] classificam a informação contextual da seguinte forma:

- *who (quem)*: identidade do usuário, pessoas próximas;
- *what (o que)*: atividade do usuário;
- *where (onde)*: localização;
- *when (quando)*: tempo; e
- *why (por que)*: informações que caracterizam o estado de espírito do usuário.

Em [Chen e Kotz 2000] os tipos de informação contextual são classificados em três categorias:

- *computacional*: conectividade de rede, custos de comunicação, largura de banda e recursos próximos, tais como impressoras, monitores e estações de trabalho.
- *de usuário*: perfil do usuário, localização, pessoas próximas e a situação social.
- *físico*: luminosidade, nível de barulho, condições de tráfego e temperatura.
- *de tempo*: hora do dia, semana, mês e estação do ano.

Para se obter um histórico, por exemplo, são utilizadas informações contextuais físicas e de usuário durante um período de tempo.

Além de classificar os tipos, Chen define informação contextual como *o conjunto de estados e opções de ambiente que determinam um comportamento da aplicação ou nos quais um evento da aplicação ocorre e é interessante para o usuário.* Distingue também dois tipos:

1. *informação contextual ativa*: influencia o comportamento de uma aplicação; e
2. *informação contextual passiva*: relevante mas não crítica para a aplicação.

Em [Lei et al. 2002] são definidos dois tipos de informação contextual: *persistente* e *transiente*. Esta última reflete o ambiente em um momento exato, enquanto que a persistente representa um padrão recorrente do contexto transiente.

Em [Dey. e Abowd 1999] os tipos de informação de contexto são classificados em quatro categorias:

- *localização*: objetos ou pessoas próximos da entidade.
- *identidade*: números de telefone, endereço, endereço eletrônico, data de nascimento e lista de amigos.
- *atividade*: atividade ocorrendo no momento.
- *tempo*: dia, hora, semana, ano.

O tipo *atividade* indica o que está ocorrendo em uma situação, a qual é definida pelas informações das quatro categorias descritas acima. Neste trabalho a informação contextual é interpretada e classificada do seguinte modo:

Qualquer informação, ativa ou passiva, que pode ser utilizada para caracterizar uma situação, sendo classificada em:

- *primária*
  - *física: localização, luminosidade, nível de barulho, temperatura*
  - *identidade: números de telefone, endereço, endereço eletrônico*
  - *tempo: segundos, minutos, hora*
  - *computacional: largura de banda, capacidade de armazenagem, conectividade de rede*
- *derivada*
  - *atividade: atividade ocorrendo no momento*
  - *computacional: custos de comunicação, taxa de utilização do recurso*

2.3.1 Natureza da Informação Contextual

Segundo [Henricksen, Indulska e Rakotonirainy 2002], informações sensíveis ao contexto possuem características importantes que determinam vários requisitos no projeto de componentes de plataformas sensíveis ao contexto:

- Temporal

  A informação contextual pode ser dinâmica ou estática. A informação contextual dinâmica, mais freqüente em sistemas sensíveis ao contexto, descreve as constantes mudanças ocorridas no contexto. Um exemplo de deste tipo é a localização de uma entidade. A informação contextual estática descreve aspectos invariantes. O dinamismo da informação contextual influencia a freqüência com que é obtida.

- Imperfeita

  A característica dinâmica de ambientes, como o da computação sensível ao contexto, a obtenção de informações incorretas providas por sensores, algoritmos ou usuários e as possíveis falhas de conexão entre as entidades influenciam na qualidade da informação contextual. Assim, se uma informação não reflete o estado corrente do mundo, ela é dita incorreta; se contém informação contraditória, é inconsistente; e se alguns aspectos do contexto não são conhecidos, é incompleta. A imperfeição pode ser atenuada através da utilização de interpretadores de contexto, que inferem novas informações contextuais a partir de informações de mais baixo nível. Entretanto, não é uma tarefa fácil desenvolver tais interpretadores devido à complexidade inerente ao processo de manipulação e coordenação dos provedores de informação contextual.

- Representações Alternativas

  A informação contextual possui representações alternativas devido às várias possibilidades de derivação a partir de provedores de informação contextual. Um tipo de informação que exemplifica muito bem essa situação é a localização de uma entidade, que pode ser representada por latitude e longitude, ou uma descrição textual como rua, prédio, escritório. Essa diversidade pode ser um problema caso a forma de representação desejada seja diferente da que é provida. Uma maneira de contornar essa situação é utilizar mecanismos de processamento de informação para transformar uma representação em outra desejada pela aplicação.

- Dependência

  Uma informação contextual pode ser uma composição de outras informações contextuais. A atividade corrente de uma pessoa pode ser obtida, por exemplo, da composição de sua localização e do histórico de atividades anteriormente realizadas. Neste caso, a atividade corrente é uma informação contextual derivada, dependente de informações utilizadas para obtê-la, o que caracteriza uma relação de dependência.

2.4 Computação Sensível ao Contexto

Conforme mencionado, um dos objetivos da computação ubíqua é fazer com que dispositivos interajam com o ambiente e automaticamente adaptem-se às mudanças com base em informações, como necessidades de usuários, de outros dispositivos, tempo e espaço. Com isso, a sensibilidade ao contexto passa a ser um dos seus maiores requisitos.

No modelo computacional tradicional, ilustrado pela Figura 2.1, um serviço recebe explicitamente uma entrada quando é invocado, realizando processamento com base nessa entrada e ao final retornando um resultado. Todas as vezes que for executado e submetido à mesma entrada, o serviço retorna o mesmo resultado.

O modelo de computação sensível ao contexto, ilustrado pela Figura 2.2, possui uma fonte de dados adicional, a informação contextual que, de maneira implícita e transparente para o usuário, é obtida a partir de informações fornecidas por provedores de informação contextual, como um sensor de temperatura ou de batimentos cardíacos.

Na computação sensível ao contexto as aplicações se caracterizam pela mobilidade. Sistemas sensíveis ao contexto procuram lidar com a característica móvel das aplicações

Figura 2.1: Sistema computacional tradicional.

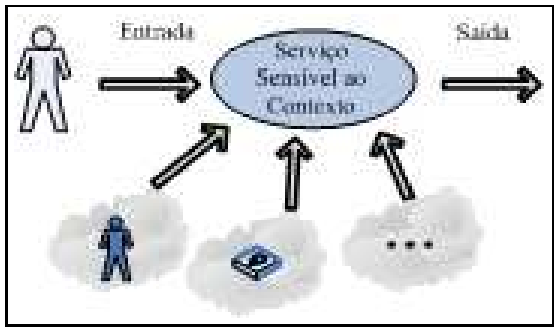

Figura 2.2: Sistema computacional sensível ao contexto.

não como um problema a ser resolvido, mas como um meio de dar suporte à criação e ao gerenciamento de uma nova geração de aplicações, as *aplicações sensíveis ao contexto*. Tais aplicações são mais portáteis e utilizam a característica contextual da informação para permitir uma melhor interação com os usuários, através de uma maior sensibilidade do ambiente.

Várias propostas procuram caracterizar e definir melhor a computação sensível ao contexto. Uma delas foi proposta por [Schilit e Theimer 1994], que define esse tipo de computação como *a habilidade de aplicações móveis de usuário de descobrirem e reagirem às mudanças ocorridas no ambiente onde estão inseridas - levando-se em consideração sua localização, as pessoas e objetos próximos e mudanças nesses objetos durante o tempo*. Desta forma, qualquer alteração no ambiente poderá levar a uma mudança de comportamento da aplicação.

[Dey. e Abowd 1999] definem computação sensível ao contexto como *um sistema que usa o contexto para prover informação relevante e/ou serviços para o usuário, onde relevância depende das tarefas do usuário*.

Em [Schilit, Adams e Want 1994] as aplicações sensíveis ao contexto são classificadas de acordo com dois pontos em duas dimensões ortogonais, conforme mostrado na tabela 2.1.

	Manual	**Automático**
Informação	Seleção de proximidade	reconfiguração automática contextual
Comando	Informação contextual e comandos contextuais	Ações de contexto disparado.

Tabela 2.1: Classificação de contexto segundo [Schilit, Adams e Want 1994].

- *Seleção de proximidade*: é uma técnica de interface de usuário onde os objetos localizados na proximidade são enfatizados ou são fáceis de serem escolhidos. A

classe de aplicações que se utilizam dessa técnica estão interessadas em localizar objetos com base na informação de localização;

- *Reconfiguração contextual automática*: é o processo de adicionar novos componentes, remover componentes existentes, ou alterar as conexões entre eles. As aplicações que se enquadram nesta classe utilizam esse processo para automaticamente reconfigurar componentes (servidores, clientes e canais de comunicação entre eles) com base em informações contextuais;

- *Informação e comandos contextuais*: as aplicações que se enquadram nesta classe se utilizam de informações contextuais para parametrizar comandos. Por exemplo, se um usuário está em uma sala de biblioteca e deseja imprimir um documento, o comando utilizado para impressão é executado com os parâmetros de localização do usuário para que o documento seja impresso pela impressora mais próxima dele; e

- *Ações de contexto disparado*: as aplicações desta classe se utilizam de regras IF-THEN para especificar como podem se adaptar.

Em [Chen e Kotz 2000] a definição baseia-se na perspectiva de como uma aplicação pode tirar vantagem do contexto, da seguinte forma:

- *Percepção de contexto ativo*: uma aplicação se adapta automaticamente ao contexto descoberto, mudando o comportamento da aplicação;

- *Percepção de contexto passivo*: uma aplicação apresenta o contexto novo ou atualizado para um usuário interessado, ou torna o contexto persistente para que o usuário o recupere depois.

Neste trabalho, computação sensível ao contexto é interpretada como a habilidade de aplicações, tanto de usuário quanto do sistema, descobrirem e adaptarem-se às mudanças no ambiente onde estão situadas. Em relação ao ambiente são levados em consideração os objetos próximos, as mudanças nesses objetos durante o tempo, a localização e as pessoas. A adaptação pode ser automática ou feita pelo usuário através de solicitação. A solicitação é feita pelo usuário interessado que informa o sistema da mudança de contexto, ou armazena as informações sobre o contexto para posterior utilização.

2.5 Arquiteturas Sensíveis ao Contexto

Nos últimos anos, com o intuito de tratar e superar vários desafios teóricos e tecnológicos impostos pela computação sensível ao contexto, têm surgido várias propostas e pesquisas voltadas para o projeto e a construção de infra-estruturas de suporte a aplicações sensíveis ao contexto. Algumas das mais referenciadas são:

- *Solar System*: desenvolvido no *Dartmouth College*, é uma arquitetura distribuída de componentes, organizada de maneira semelhante ao sistema solar, que oferece apoio à interação entre aplicações sensíveis ao contexto [Chen e Kotz 2002]. A arquitetura do *Solar System* enfatiza características voltadas para a mobilidade do usuário, flexibilidade e escalabilidade, tendo estruturas direcionadas para o paradigma distribuído de computação;

- *SOCAM* : desenvolvida na *Computing School of Singapure National University*, a arquitetura *Service-Oriented Context-Aware Middleware (SOCAM)* [Gu et al. 2004] utiliza um modelo formal de contexto baseado em ontologia, cujo objetivo é prover prototipação rápida de serviços sensíveis ao contexto em ambientes de computação pervasiva. As informações contextuais são descritas em uma linguagem de marcação semântica (*Ontology Web Language - OWL*), permitindo, com isso, o compartilhamento de conhecimento contextual entre entidades diferentes, além de ser possível raciocinar sobre tais informações. A possibilidade de especificação e de raciocínio contextual constitui um dos principais aspectos da arquitetura;

- *CoBrA*: desenvolvida na *University of Maryland* [Chen 2004], é uma arquitetura com distribuição híbrida, composta de um componente central denominado *context broker*, cujas funções são:

  1. compartilhar com os outros agentes da arquitetura uma ontologia comum para a representação das informações contextuais;
  2. realizar inferências a partir dos dados sensoriais; e
  3. tratar questões relativas a privacidade dos usuários e segurança do sistema.

  Além de seu componente central (*broker*), a arquitetura *CoBrA* também incorpora outros agentes configuráveis para notificar às aplicações acerca de mudanças de contexto; e

- *WASP*: desenvolvida pelo Instituto de Telemática da Universidade de Twente na Holanda e pela Ericson [Costa 2003]. Posteriormente teve continuidade através de uma parceria institucional com o Departamento de Informática da Universidade Federal do Espírito Santo [Santos 2004]. A arquitetura da plataforma WASP (*Web Architectures for Service Platforms*) tem como característica particular o uso da tecnologia de distribuição de Serviços *Web* e é executada sobre uma rede móvel 3G. Na época em que foi definida, a WASP era uma das poucas plataformas que usava serviços *Web* como infra-estrutura de distribuição. Este foi o principal motivo que a levou a ser tomada como base para a construção da plataforma *Infraware*, descrita a seguir.

Há outras plataformas que se propõem a superar os desafios teóricos e tecnológicos impostos pela computação sensível ao contexto. Uma descrição mais detalhada sobre as vantagens e desvantagens de algumas propostas pode ser encontrada em [Filho et al. 2005].

2.5.1 Plataforma *Infraware*

A *Infraware* é um *middleware* baseado na tecnologia de serviços *Web*. Ela visa dar suporte arquitetural para o desenvolvimento, construção e execução de aplicações móveis sensíveis ao contexto, bem como auxiliar o projetista a conceber aplicações utilizando serviços, mecanismos e interfaces que escondam a complexidade introduzida pela manipulação da informação contextual [Filho et al. 2005]. O tratamento e a manipulação da informação contextual são fundamentados em conceitos, teorias, linguagens e modelos da *web* semântica, em especial no uso de ontologias (vide seção 2.6).

A arquitetura conceitual da *Infraware* estende as funcionalides a plataforma WASP e foi definida com base nos seguintes requisitos funcionais:

- Aquisição de informações contextuais utilizando diferentes métodos/estratégias;
- Um modelo de subscrição para que aplicações possam se inscrever para o recebimento de mensagens sobre a mudança de um contexto específico;
- Um modelo de representação que descreva as informações contextuais de forma estruturada;
- Interpretação de informações contextuais através de métodos para gerar diferentes níveis de abstração e/ou informações contextuais derivadas para atender as restrições conceituais e estruturais das aplicações interessadas;
- Equilíbrio entre distribuição e integração de dados com o objetivo de facilitar o crescimento modular e a distribuição de tarefas, permitindo que mecanismos centralizados interajam com módulos distribuídos, de modo a atender aos requisitos de desempenho, simplicidade, abrangência e robustez;
- Segurança e privacidade de transações e execuções de tarefas de usuários que tenham determinadas restrições de privacidade;
- Coordenação entre aplicações utilizando políticas de prioridades e cooperação com o objetivo de evitar condições de corrida no que diz respeito ao acesso de recursos da plataforma;
- Descoberta de serviços utilizando métodos mais elaborados de comparação de descrições semânticas de serviços. Este requisito é explorado com maior profundidade neste trabalho, sendo o tema central de interesse da presente pesquisa.

Com base nesses requisitos a arquitetura conceitual da plataforma foi concebida. Seis módulos funcionais compõem a arquitetura da plataforma *Infraware*, ilustrada na Figura 2.3.

A *Infraware* interage com três entidades: *aplicações sensíveis ao contexto*, *provedores de serviço* e *provedores de informação contextual*. As aplicações sensíveis ao contexto são os clientes da plataforma. Elas utilizam os serviços previamente disponíveis, as informações contextuais provenientes dos provedores de informação contextual e da plataforma para se adaptarem de acordo com o contexto no qual estão inseridas. Os provedores de serviço são entidades que fornecem serviços à plataforma. Os provedores de informação contextual são entidades, de *hardware* ou *software*, que provêem à plataforma informações contextuais.

Os módulos da plataforma possuem as seguintes responsabilidades:

1. *Gerente de Subscrição*: responsável por gerenciar assinaturas de serviços previamente disponíveis aos clientes da plataforma. As subscrições são feitas através de uma linguagem voltada para a troca de sentenças e expressões entre as aplicações sensíveis ao contexto e a plataforma. Essa linguagem é utilizada para resolver, interpretar e gerenciar as requisições enviadas pelas aplicações. A interação entre as aplicações e a plataforma deve suportar modelos de requisição do tipo *request-response*, *time-driven* e *event-driven*, os quais determinam o comportamento da plataforma para determinadas requisições de subscrição;

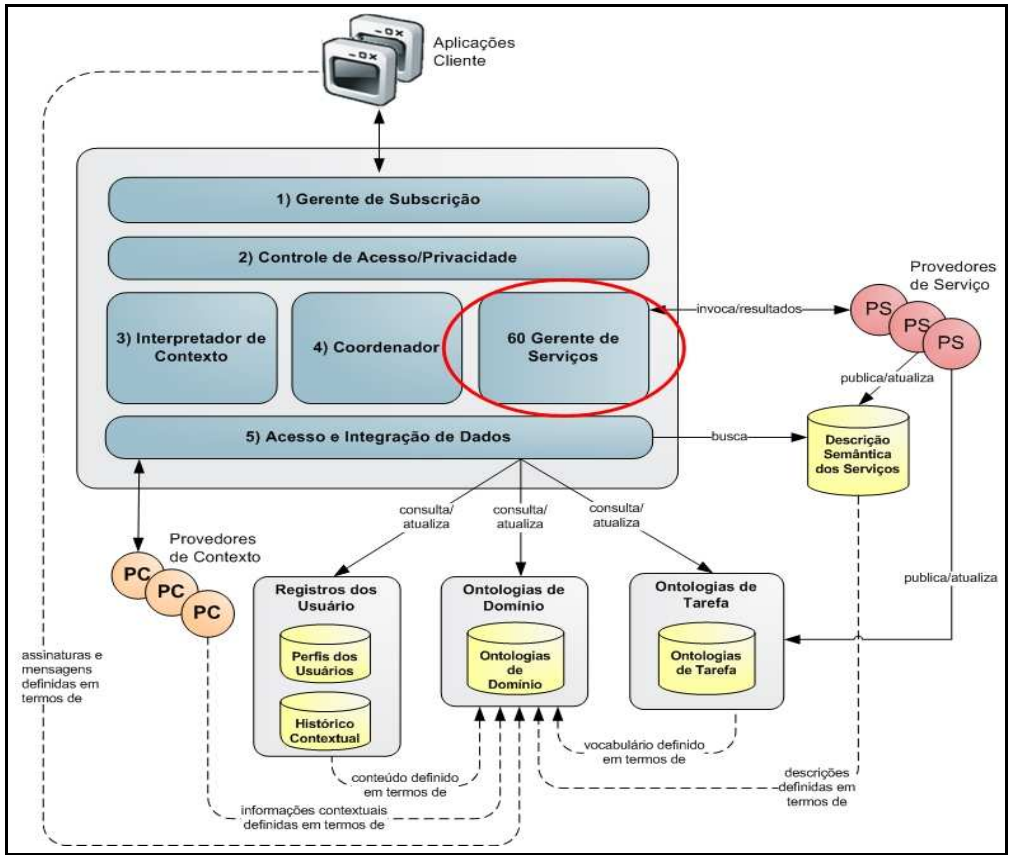

Figura 2.3: Arquitetura *Infraware*.

2. *Controle de Acesso e Privacidade*: responsável por filtrar o fluxo de dados entre a plataforma e as aplicações. Para isso utiliza um conjunto de restrições envolvendo preferências de privacidade dos usuários e políticas de privacidade das aplicações, que permitem estabelecer limites de visibilidade para os dados coletados pela plataforma;

3. *Interpretador de Contexto*: responsável pela manipulação, derivação, refinamento e inferência de informações contextuais mais elaboradas a partir de informações contextuais primitivas provenientes de diferentes fontes, com a finalidade de torná-las disponíveis às aplicações de forma transparente;

4. *Coordenador*: responsável pelo gerenciamento, geração e disparo de planos de ação e atividades executados pelos outros componentes da plataforma. Quando necessário, resolve conflitos entre as tarefas e atividades realizadas na plataforma e entre as requisições das aplicações. Interage com os demais componentes para executar as requisições feitas pelas aplicações;

5. *Acesso e Integração de Dados*: responsável por prover uma interface única de acesso a dados, proporcionando transparência no que diz respeito à distribuição e à heterogeneidade das fontes de dados. Este componente é um *middleware* baseado em componentes e *frameworks* [Barbosa, Porto e Melo 2002]; e

6. *Gerente de Serviços*: é responsável pelas atividades de composição e descoberta de serviços. Esta última contempla as atividades de publicação e seleção de serviços que

utilizam uma abordagem baseada em descrições semânticas dos serviços. Essas descrições apresentam a vantagem de permitir a utilização de informações contextuais para selecionar os serviços que atendam às características desejadas. A descoberta garante que o serviço descoberto está disponível no momento da solicitação.

A concepção do Gerente de Serviços da *Infraware* se baseou em uma proposta feita por [Santos 2004]. Esta proposta estende o Gerente de Serviços proposta na WASP através da utilização de ontologias para descrever semanticamente os serviços. O Gerente de Serviços proposto por Santos é composto por cinco módulos funcionais, conforme ilustra a Figura 2.4.

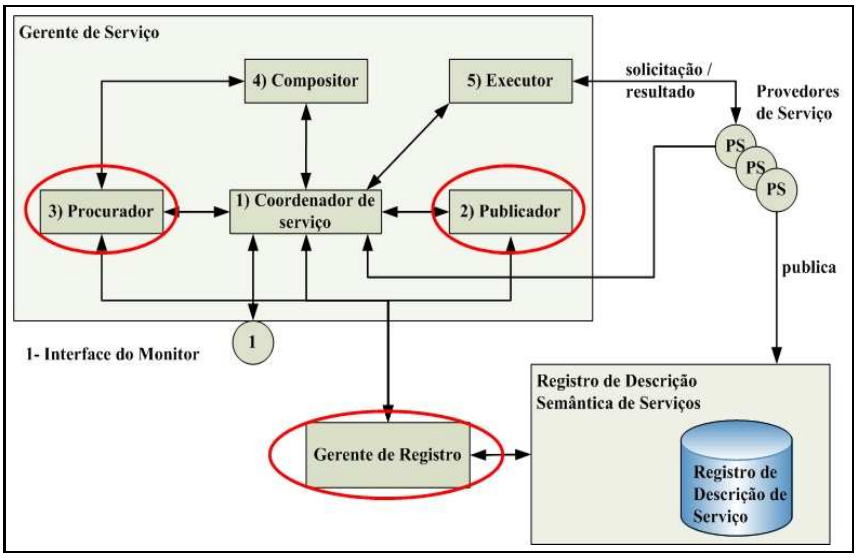

Figura 2.4: Gerente de Serviços - Extensão da WASP.

Os componentes do Gerente de Serviços possuem as seguintes responsabilidades:

1. *Coordenador de Serviços*: é responsável por encaminhar as mensagens recebidas dos provedores de serviço e do *Coordenador* da plataforma aos demais componentes do Gerente de Serviços ou ao módulo *Acesso e Integração de Dados*;

2. *Publicador*: é responsável pelo anúncio das descrições dos serviços, que são verificadas. Se são válidas, são adicionadas no *Registro de Descrição Semântica de Serviços*;

3. *Procurador*: é responsável por descobrir descrições semânticas dos serviços disponíveis na plataforma;

4. *Compositor*: é responsável pelo processo de composição semântica de serviços; e

5. *Executor*: é responsável por executar e monitorar os serviços. A partir de um plano de execução gerado pelo compositor os serviços são invocados de forma ordenada.

O Gerente de Serviços tem um papel fundamental na plataforma *Infraware*, pois é ele o responsável por dar suporte à utilização de serviços. Por exemplo, sem o componente de descoberta de serviços não é possível prover às aplicações os serviços desejados, tampouco

permitir ao componente de composição ser abastecido dos serviços necessários para a execução de sua tarefa. É também muito importante para o módulo de interpretação de contexto, pois provedores de informação contextual são também encontrados através dele.

O protocolo proposto neste trabalho implementa as funcionalidades previstas para os módulos *Descobridor*, *Publicador* e *Gerente de Registro*, que são descritos com maiores detalhes no capítulo 5.

2.6 Ontologias

Como mencionado anteriormente, a plataforma *Infraware* utiliza amplamente o conceito de ontologias em alguns de seus módulos, como no módulo Gerente de Serviços para a descoberta, a composição e execução de serviços; no módulo Interpretador de Contexto para a inferência de novas informações contextuais; e no módulo de Acesso e Integração de Dados para a armazenagem e distribuição de dados. Esta seção tem por objetivo introduzir o tema e destacar o potencial de uso de ontologias em sistemas sensíveis ao contexto.

O termo Ontologia vem sendo usado há alguns anos pelas comunidades de Inteligência Artificial e de Representação do Conhecimento. Sua essência parte da premissa de que a formalização do conhecimento começa com uma conceituação, que por sua vez consiste de um conjunto de entidades sobre o qual o conhecimento e a relação entre essas entidades é expresso.

Uma ontologia é uma especificação explícita de uma conceituação, isto é, uma descrição (como uma especificação formal de um programa) dos conceitos e relacionamentos que podem existir em um determinado domínio [Gruber 1993]. Uma conceituação é uma visão abstrata e simplificada do domínio que se deseja representar para algum objetivo [Gruber 1995], tendo em vista que é impossível representar o mundo real, ou mesmo uma parte dele, em sua completa riqueza de detalhes [Guizzardi 2000].

Ontologia é mais que uma taxonomia, pois inclui a expressão do domínio específico do conhecimento. A possibilidade de compartilhar o entendimento comum do conhecimento é uma das principais razões para sua utilização. O reuso do conhecimento é outro ponto importante, pois ontologias anteriormente definidas podem ser reutilizadas e enriquecidas com novos conceitos e relacionamentos. Ela é mais que um vocabulário padrão, pois assegura que os termos escolhidos sejam suficientes para especificar e definir conceitos e permitir relacionamentos adequados a partir da escolha terminológica. Além disso, por se apresentarem como modelos formais, permitem analisar o conhecimento de um domínio através de inferências explícitas e com isso novos fatos e relacionamentos entre as entidades podem ser descobertos.

As ontologias ultrapassaram o domínio da inteligência artificial e estão sendo utilizadas e desenvolvidas por especialistas de diversos domínios da computação. Alguns deles são [Carrara e Guarino 1999]:

- Linguagem Natural: é utilizada para caracterizar o significado das palavras.
- Banco de Dados: é utilizada em esquemas conceituais, para permitir interoperabilidade semântica entre bases de dados heterogêneas.
- Orientação a Objetos: é utilizada para especificar sistemas.

- Recuperação de Informação: é utilizada para especificar o significado dos conceitos a serem procurados.

No caso específico dos sistemas sensíveis ao contexto, uma das dificuldades encontradas diz respeito à modelagem da informação contextual, devido às suas características particulares (vide subseção 2.3.1). Essas características não são facilmente modeladas utilizando as técnicas tradicionais de engenharia de *software* e de sistemas de informação, como UML e ER. Uma abordagem que vem sendo defendida como uma maneira conveniente de se modelar as características da informação contextual é a utilização de ontologias. Algumas propostas, como SOCAM [Gu et al. 2004] e CoBra [Chen 2004], têm se favorecido das facilidades oferecidas pelas ontologias para permitir maior interoperabilidade entre serviços e dispositivos e fazer comparações mais elaboradas para descobrir serviços mais adequados às necessidades das aplicações sensíveis ao contexto.

As ontologias podem ser classificadas, com base em seu conteúdo, nas seguintes categorias [Guarino 1998]:

- **Ontologias Genéricas**

  Descrevem conceitos bastante gerais, tais como, espaço, tempo, matéria, objeto, evento, ação, etc. São independentes de um problema ou domínio particular.

- **Ontologias de Domínio**

  Expressam conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, tal como Medicina ou Direito.

- **Ontologias de Tarefas**

  Expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como, diagnóstico de doenças ou vendas de produtos.

- **Ontologias de Aplicação**

  Descrevem conceitos dependentes do domínio e da tarefa particulares. Estes conceitos freqüentemente correspondem a papéis desempenhados por entidades do domínio quando da realização de uma certa atividade.

- **Ontologias de Representação**

  Explicam as conceituações que fundamentam os formalismos de representação do conhecimento. As ontologias de domínio são construídas para serem utilizadas em um micro-mundo. São o tipo mais comum desenvolvido. Diversos trabalhos encontrados na literatura, enfocam áreas como química, medicina (*Unified Medical Language System (UMLS)*), modelagem de processos de sistema, biologia molecular e bioquímica (*GENSIM*) e ciência dos materiais (*PLINIUS*).

As ontologias, portanto, constituem uma ferramenta poderosa para suportar a especificação e a implementação de sistemas computacionais. Ao se usar esta abordagem na fase de análise e especificação do domínio, vários benefícios são obtidos como a comunicação entre as pessoas para obtenção de consenso sobre um vocabulário; formalização e conseqüente eliminação ambiguidades; inferência de novas informações; representação do conhecimento em um alto nível de abstração; e reutilização do conhecimento.

No caso particular de protocolos de descoberta de serviços, ontologias podem ser utilizadas para descrever os serviços e permitir uma maior precisão ao processo de descoberta, sendo esta a abordagem utilizada no presente trabalho. O capítulo 5 descreve em maiores detalhes a utilização de ontologias no protocolo proposto.

2.7 Conclusão

O termo contexto tem sido alvo de investigações há várias décadas nas mais diversas áreas da ciência. Na computação, a comunidade de Inteligência Artificial foi uma das primeiras a fazer contribuições para o entendimento e a formalização do termo. Atualmente, com o advento da computação móvel, contexto é um tema bastante explorado, especialmente pelas comunidades de Computação Ubíqua e Interação Homem-Computador.

Apesar de muitas definições, ainda não há um consenso no uso do termo, que depende, inclusive, da área de pesquisa. Neste trabalho contexto é interpretado como o estado de um ambiente em um determinado momento e é caracterizado por um conjunto de informações contextuais.

A interação entre aplicações sensíveis ao contexto e o ambiente, bem como a adaptação de tais aplicações às mudanças ocorridas nele, são dois dos objetivos da Computação Sensível ao Contexto. Para que possam ser alcançados, entretanto, há algumas dificuldades a serem vencidas, como a modelagem das informações contextuais e seus interrelacionamentos. Para isso, o emprego de ontologias é uma abordagem que vem sendo utilizada.

A plataforma *Infraware* se favorece das facilidades oferecidas pelas ontologias genéricas, de domínio, de tarefas, de aplicação e de representação para permitir maior interoperabilidade entre serviços e dispositivos, principalmente no que diz respeito à descoberta de serviços mais adequados às necessidades das aplicações sensíveis ao contexto, tema central deste trabalho.

Os serviços possuem um importante papel no desenvolvimento de plataformas sensíveis ao contexto. Aspectos como segurança, modelo de descrição, estratégias de seleção, distribuição e armazenagem são requisitos importantes no projeto do componente de gerência de serviços dessas plataformas e são discutidos no próximo capítulo.

Capítulo 3

Serviços

3.1 Introdução

Como visto, uma das mais importantes contribuições da Computação Sensível ao Contexto é a possibilidade de permitir que aplicações se adaptem às mudanças ocorridas no ambiente. Utilizando-se de informações contextuais, fornecidas por serviços sensíveis ao contexto, tais aplicações podem tomar decisões e alterar seu comportamento de modo automático.

A realização da adaptação automática requer que as aplicações encontrem serviços que provejam as informações solicitadas, o que não é uma tarefa fácil devido à característica dinâmica do ambiente onde esses serviços e aplicações estão inseridos.

Alguns requisitos precisam ser contemplados para garantir a eficiência da descoberta de serviços, como uma descrição de serviço que permita a realização de inferências semânticas, uma seleção de serviços que leve em consideração a natureza da informação contextual, uma organização das descrições de serviços que possibilite a realização de pesquisas avançadas e a existência de políticas de segurança e autenticação que restrinjam o acesso de determinadas classes de aplicação à alguns serviços. Esses são alguns dos requisitos discutidos neste capítulo.

3.2 Definição

Segundo [McGrath 2000], não há uma padronização na terminologia utilizada na área de descoberta de serviço no que se refere aos termos *dispositivo* e *serviço*. Em computação, normalmente, utiliza-se o termo dispositivo para identificar uma entidade física, concreta, como uma impressora, um PDA, um *mouse* ou um computador pessoal, os quais podem ser designados, respectivamente, de dispositivo de impressão, dispositivo móvel, dispositivo apontador e dispositivo de computação.

Serviços caracterizam-se por prover funcionalidades aos seus clientes, por exemplo serviço de impressão, serviço de monitoramento através de câmeras de vídeo. A confusão quanto ao emprego dessas designações existe porque muitos dispositivos são identificados como serviços em uma rede de computadores [McGrath 2000]. Além disso, McGrath explica que um simples dispositivo de rede pode implementar muitos serviços, como um *pen drive* que pode ser utilizado como um dispositivo de armazenamento e como um tocador de músicas.

Há uma correlação entre serviços e dispositivos, através da qual verifica-se que tanto um como o outro provêem funcionalidades para alcançar um objetivo. Além disso, eles podem ser concretos (dispositivos físicos) ou implementados por meio de alguma linguagem de programação. No mundo *web*, por exemplo, as funcionalidades são providas por sítios, que oferecem informações dinâmicas, geradas a partir de alguma ação ou mudança do mundo. Esses sítios são denominados *serviços web*, considerados os mais importantes entre os vários recursos disponíveis na *web* [Martin et al. 2005].

Ambos - dispositivos e serviços - são, portanto, entidades que provêem mecanismos ou um conjunto de meios dispostos para atingir um objetivo. Esse objetivo pode ser a impressão de documentos, no caso de uma impressora; ou disponibilização de páginas *web*, no caso de um servidor *web*. Neste trabalho, dispositivos são também designados como serviços.

3.3 Descrição de Serviço

Um serviço precisa ser identificado de alguma forma. Dependendo do tipo de identificação empregada existem implicações que precisam ser tratadas como a duplicação ou a complexidade da identificação utilizada, ou a limitação imposta pelo tipo de descrição ou mesmo pelo protocolo utilizado para prover o serviço.

Uma forma de descrever um serviço baseia-se em uma descrição nominal, que tem a vantagem de facilitar ao usuário memorizar e identificar rapidamente o tipo do serviço oferecido. Por exemplo, uma aplicação apresenta ao usuário um serviço de vídeo, descrito como VÍDEO. Há, porém, o problema do mesmo serviço ter uma descrição diferente em um outro ambiente computacional, por exemplo, VÍDEO-CÂMERA. Apesar de ser o mesmo serviço, a descrição diferente dificulta ao usuário encontrar o serviço desejado. Além disso, há a possibilidade de existirem várias descrições para o mesmo serviço em um mesmo ambiente, o que não é desejável.

Uma alternativa para resolver esse problema é utilizar uma descrição baseada em número. Apesar de interessante, essa alternativa é um pouco inconveniente para o usuário, pois há a necessidade de se memorizar números de serviços, o que não facilita a sua identificação (falha de naturalidade). Além disso, não há garantias de que a identificação do serviço será única, ou seja, há a possibilidade de o serviço possuir outros números para ser identificado em outros ambientes computacionais, visto que cada um desses ambientes pode adotar um padrão de numeração diferente.

A descrição baseada em URL resolve esse problema. Nessa abordagem, um serviço pode ser definido da seguinte forma:

```
service:printer:lpr://hostname
```

Dessa forma, a possibilidade de um serviço possuir diferentes identificações é eliminada, pois o par "protocolo:máquina" torna o serviço único no ambiente em que está disponível. No exemplo dado acima, *lpr* identifica o serviço de impressão e *hostname* identifica univocamente o servidor que provê o serviço. Esta abordagem, apesar de ser útil, apresenta o inconveniente de não permitir uma descrição mais rica, como a especificação da localização física do serviço, a capacidade de processamento, se está em uso ou disponível, entre outras.

A descrição baseada em atributos ou *template* permite descrições mais ricas. Utilizando-se de um "molde" é possível descrever o serviço com vários atributos e, assim, o usuário pode realizar pesquisas mais avançadas, definindo atributos desejáveis para o serviço que procura.

Há várias possibilidades na utilização da descrição baseada em atributos, podendo ser especificada uma estrutura que contenha vários campos que descrevem o serviço. Entretanto, essa descrição fica limitada pela quantidade de campos definida no pacote do protocolo. Os atributos abaixo são exemplos do que pode ser definido no pacote de algum protocolo.

```
nome:
tipo:
localização:
capacidade:
```

A pesquisa é limitada pelos atributos definidos. Caso o usuário queira encontrar um serviço informando o atributo "fila de impressão" (que define a quantidade de trabalhos na fila de impressão), a pesquisa não poderá ser realizada porque o protocolo não saberá tratar esse atributo.

Uma outra possibilidade é utilizar uma linguagem de descrição, como XML, em conjunto com uma URL. Um exemplo de descrição do serviço utilizando essa abordagem é:

```
<chapter id="CUPS-printing">
<chapterinfo>
   <author>
      <firstname>Kurt</firstname><surname>Pfeifle</surname>
      <affiliation>
         <orgname>Danka Deutschland GmbH </orgname>
         <address><email>kpfeifle@danka.de</email></address>
      </affiliation>
   </author>
   <author>
      <firstname>Ciprian</firstname><surname>Vizitiu</surname>
      <affiliation>
         <address><email>CVizitiu@gbif.org</email></address>
      </affiliation>
      <contrib>drawings</contrib>
   </author>
   <author>&person.jelmer;<contrib>drawings</contrib></author>
   <pubdate> (27 Jan 2004) </pubdate>
</chapterinfo>
<title>CUPS Printing Support</title>
```

Como pode se observado no exemplo acima, é definida uma sintaxe para a descrição do serviço de impressão. O autor, por exemplo, é identificado por uma *tag* <*author*>.

Uma vantagem dessa descrição é que ela não é limitada pelos atributos definidos no pacote do protocolo. O serviço de impressão pode agora prover uma descrição mais flexível, ou seja, quando necessário novos atributos podem ser incluídos quando necessários. XML, entretanto, permite apenas representar os dados em um formato, mas não garante a semântica da descrição. Para representar a semântica da descrição do serviço pode-se utilizar ontologias.

Ontologias permitem descrever semanticamente um serviço e compartilhar o conhecimento, neste caso a descrição. Além disso, o processo de descoberta pode ser melhor trabalhado, ou seja, é possível utilizar restrições semânticas em pesquisas mais complexas para encontrar um serviço desejado.

A descrição semântica do serviço pode ser representa através de uma linguagem de descrição semântica como OWL. Esta linguagem é recomendada pela W3C e é empregada para representar ontologias. OWL é uma evolução da linguagem DAML+OIL e possui uma forma de serialização em RDF [W3C 2004], que é representada através de XML [W3C 2004]. Um exemplo da descrição do serviço de impressão usando OWL é:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XLMSchema#"
   xmlns="file:/home/seipel/DisLog/projects/Ontologies/
                                  Examples/printer.owl#">
   <owl:Ontology rdf:about="">
      <owl:VersionInfo>My example version 1.2, 17 October 2002
      </owl:VersionInfo>
   </owl:Ontology>
   <owl:Class rdf:ID="product">
      <rdfs:comment>Products form a class</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="padid">
      <rdfs:comment>Printing and digital imaging devices
                    form a subclass of products
      </rdfs:comment>
      <rdfs:label>Device</rdfs:label>
      <rdfs:subClassOf rdf:resource="#product"/>
   </owl:Class>
   <owl:Class rdf:ID="hpProduct">
      <rdfs:comment>HP products are exactly those products
                  that are manufactured by Hewlett Packard
      </rdfs:comment>
      <owl:intersectionOf>
         <owl:Class rdf:about="#product"/>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#manufactured-by"/>
            <owl:hasValue>
               <xsd:string rdf:value="Hewlett Packard"/>
            </owl:hasValue>
```

```
        </owl:Restriction>
      </owl:intersectionOf>
   </owl:Class>
```

Nesse exemplo, a descrição possui uma semântica, pois são definidas classes, propriedades e o relacionamentos entre essas classes. Por exemplo, a classe *hpProduct* é definida como um produto fabricado pela *Hewlett Packard*.

Com OWL um domínio pode ser formalizado através da definição de classes e suas propriedades. Permite também realizar inferência sobre as classes e indivíduos para o grau permitido pela semântica formal da linguagem [Buccella, Penabad e Rodriguez 2004]. Os serviços descritos com OWL podem ser localizados, selecionados, utilizados, compostos e monitorados de modo automático. Para o desenvolvimento de sistemas sensíveis ao contexto essas são características indispensáveis, pois permitem uma manipulação mais complexa de descrições de serviço e auxiliam a aumentar a precisão do processo de descoberta.

3.3.1 OWL-S: uma Ontologia de Serviços

Através da utilização de uma ontologia padrão, consistindo de classes e propriedades básicas e dos mecanismos de estruturação de ontologias, os serviços podem ser descritos e compartilhados. A OWL-S é uma proposta para alcançar esse objetivo.

OWL-S é uma ontologia de serviços *web* desenvolvida de modo colaborativo por vários pesquisadores. Algumas das tarefas esperadas que OWL-S deve permitir são:

- descoberta automática de serviços *web*;
- invocação automática; e
- interoperabilidade e composição automáticas.

A ontologia de serviços OWL-S, ilustrada pela Figura 3.1, é construída para prover três tipos essenciais de conhecimento sobre o serviço:

- *What it does* (O que faz);
- *How to access it* (Como pode ser acessado); e
- *How it works* (Como funciona).

A classe *Service* provê uma referência de organização do serviço *web*. Existe uma instância dessa classe para cada serviço declarado. A classe *ServiceProfile* provê a informação necessária para que o serviço seja descoberto, ou seja, essa classe descreve o que o serviço faz. As classes *ServiceModel* e *ServiceGrounding* juntas provêem informações para que o serviço seja utilizado por algum agente. A primeira descreve como o serviço pode ser utilizado, enquanto a segunda especifica os detalhes de como o serviço pode ser acessado.

Neste trabalho, OWL-S é utilizada para prover semântica às descrições de serviços e melhorar a precisão dos mecanismos de comparação empregados.

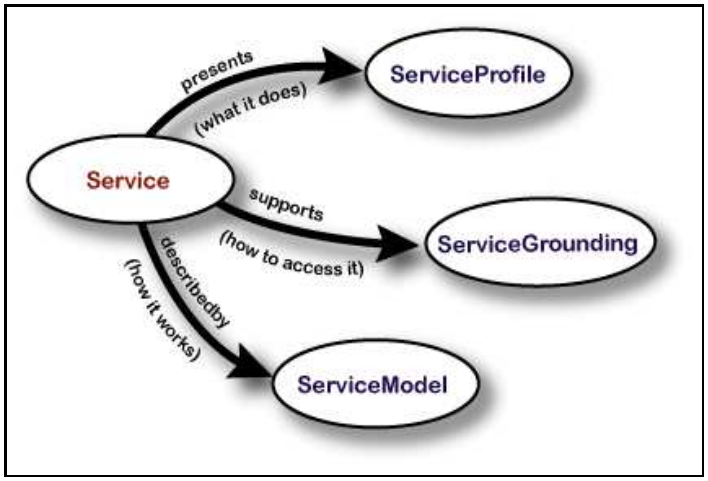

Figura 3.1: Nível mais alto da OWL-S [Martin et al. 2005].

3.4 Seleção de Serviços

Há várias abordagens para se selecionar um serviço. Por exemplo, um serviço pode ser selecionado com base em algumas informações providas pelo usuário; pode ser provida uma lista e deixar que o usuário escolha o que melhor o atende; pode também ser utilizado um processo de comparação de atributos ou descrições de serviços (*service matching*) para obter o serviço que melhor, senão o que exatamente, atende ao usuário. De modo geral, a seleção de serviço pode ser assim classificada:

- Feita pelo usuário

  O usuário escolhe um serviço de uma lista de serviços encontrados pelo protocolo.

- Feita por aplicações

  As aplicações, através de processos automáticos e com base em informações providas pelo usuário e outras implícitas, seleciona o(s) serviço(s) que melhor atende(m) às necessidades do usuário.

Em sistemas sensíveis ao contexto, além das abordagens mencionadas, o emprego de informações contextuais é muito importante para selecionar o serviço adequado, pois elas permitem descrever com mais detalhes os serviços desejados.

Em [Broens 2004] é proposto um método de seleção baseado em uma classificação de graus de similaridade. Segundo essa proposta, *grau de similaridade é uma medida de qualidade que indica o quão próxima uma descrição de serviço é de uma requisição de serviço.*

O grau de similaridade depende da comparação entre dois conceitos. Por exemplo, entre uma descrição de serviço e uma requisição pode haver quatro tipos de comparação [Li e Horroks 2003]. A Figura 3.2 ilustra um exemplo onde **OutR** é a saída do requisitante, e **OutA** a do serviço anunciado.

Os graus de similaridade são:

(1) *Exato*: se **OutR** e **OutA** são iguais, pois não há propriedade faltando. Outra possibilidade ocorre quanto **OutR** é uma sub-classe (sub-tipo imediato) de **OutA**. Este é o primeiro melhor grau de similaridade.

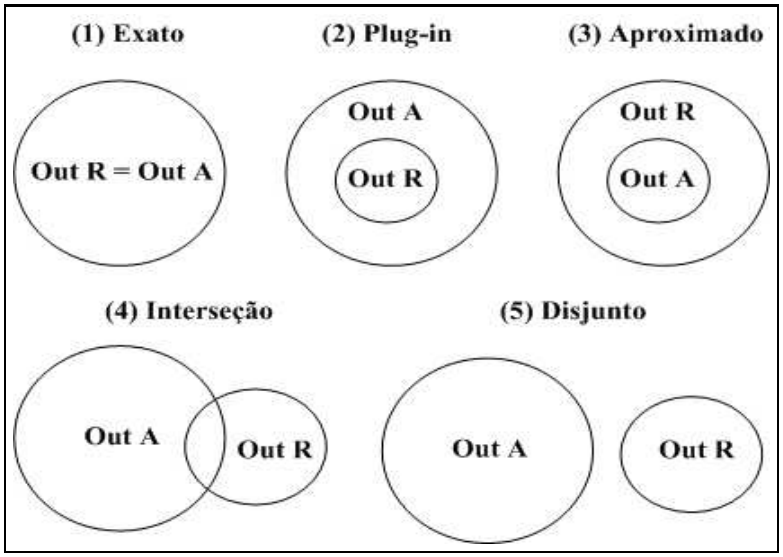

Figura 3.2: Graus de similaridade.

(2) *Plug-in*: é o segundo melhor grau de similaridade. Acontece quando **OutA** engloba **OutR**. Neste caso, o serviço provê mais propriedades do que o requisitante deseja, então **OutA** pode substituir (*plugged in*) **OutR**. Pode ser considerado um casamento exato porque os requisitos do solicitante são atendidos.

(3) e (4) *Subsume (Aproximado ou Interseção)*: acontece quando **OutA** é um sub-conjunto de **OutR**, ou **OutA** e **OutR** possuem uma interseção. O serviço fornece apenas parte do que o solicitante quer. O grau de similaridade é inferior ao *Plug-in*.

(5) *Falha*: acontece quando **OutR** e **OutA** não possuem relações entre si.

Como pode ser observado, é utilizada a teoria de conjuntos para classificar o grau de similaridade. Com base nisso, aplicações têm mais possibilidade de, com mais precisão, selecionar serviços para usuários.

A seleção de serviços é empregada por alguns protocolos de descoberta, que se utilizam de informações providas pelo usuário e de mecanismos de seleção, como escopo administrativo e qualidade de serviço. Entretanto, dos protocolos de descoberta referenciados na literatura, nenhum utiliza informação contextual na seleção do serviço desejado [Zhu, Mutka e Ni 2002].

O processo de seleção de serviço deve, portanto, utilizar informações e mecanismos baseados em:

- Informação Contextual

  A comparação pode utilizar informações contextuais computacionais, de usuário, do mundo físico, tempo e atividade para selecionar um serviço.

- Escopo Administrativo

  O mecanismo de seleção pode se utilizar do escopo administrativo, ou seja, de uma classificação baseada no domínio da rede a qual o serviço pertence, para selecionar o serviço desejado. Por exemplo, dados dois serviços S_1 e S_2 semelhantes, pertencentes cada um às redes R_1 e R_2, respectivamente. O processo de seleção baseado no escopo

administrativo seleciona o serviço que esteja na mesma rede do usuário, ou seja, se o usuário estiver na rede R_1, o serviço selecionado será o S_1. Esse processo é muito útil para uma grande quantidade de serviços espalhados por vários ambientes computacionais.

- Qualidade de Serviço

  Está associada à comparação de atributos, à descrição do serviço e ao anúncio dos provedores de serviço. Sendo a descrição do serviço constantemente atualizada pelo provedor de serviço, é possível ao usuário obter aquele que melhor atende às suas necessidades. Além disso, o mecanismo pode determinar que provedor de serviço está disponível em um dado momento.

- Comparação de Descrições de Serviço (*Service Matching*)

  Uma forma de comparar serviços é verificar se o que é desejado é o que é anunciado pelo provedor de serviço. Utilizando descrições semânticas de serviços, baseadas em uma ontologia básica, OWL-S por exemplo, o mecanismo de seleção pode, atribuindo a cada serviço encontrado um grau de similaridade, selecionar com mais precisão os serviços que atendem às necessidades do usuário.

3.5 Serviço de Diretório

A medida que a quantidade de informação aumenta, cresce também a necessidade de gerenciá-la. Antes era suficiente uma agenda para guardar números de telefone e endereços. Agora, entretanto, é indispensável um mecanismo mais robusto e expansível, como uma agenda eletrônica ou mesmo um PDA, para guardar e organizar números de telefone, endereços eletrônicos, endereços físicos e outras informações relevantes.

Os sistemas computacionais sofrem com o mesmo problema. No início da *Advanced Research Projects Agency Network (ARPANet)* uma lista com os poucos computadores podia ser mantida utilizando um arquivo contendo todas as informações desses computadores. Além disso, uma entidade central, o *Network Information Center (NIC)*, era suficiente para manter essa lista. Com a proliferação da *Internet*, porém, a manutenção do *HOSTS.TXT* tornou-se uma tarefa difícil de ser realizada. Antes que todos recebessem uma versão atualizada, novas informações eram inseridas no arquivo. A solução encontrada foi distribuir o gerenciamento dessas informações. O *Domain Name System*, um dos serviços de diretório de sucesso já implementados, surgiu como uma solução para gerenciar, utilizando distribuição e delegação de responsabilidade, informações dos computadores da *Internet* [Albitz e Liu 1998].

Um serviço de diretório pode ser definido como um *software*, ou um conjunto de aplicações, que armazena e organiza informação sobre uma rede, seus recursos e usuários, permitindo que essas informações sejam gerenciadas e acessadas [Wikipedia 2005]. É utilizado para localizar, gerenciar, administrar e organizar recursos computacionais, como impressoras, usuários, grupos de usuários, entre outros.

O serviço de diretório pode ser centralizado ou distribuído. Sendo centralizado apresenta desvantagens como indisponibilidade e congestionamento no acesso a ele. Por outro lado, o acesso é direcionado, o que causa menos tráfego na rede. Se o serviço de diretório utiliza uma abordagem distribuída, a resposta às solicitações de pesquisas realizadas pode ser mais eficientes, pois a quantidade de informação a ser consultada é menor. Uma

desvantagem dessa abordagem é a demora quando é necessários consultar vários serviços de diretório para obter a informação desejada [Zhu, Mutka e Ni 2002].

Uma forma de tornar o serviço de diretório distribuído mais eficiente é replicar os dados em todos os diretórios. Essa replicação pode ser total ou parcial. Na replicação total todas as informações são duplicadas para todos os servidores. Isto pode ocasionar duas situações, dependendo da quantidade de dados e da freqüência com que a atualização das réplicas é feita:

- base de dados grandes e freqüência de atualização muito alta pode gerar tráfego muito intenso da rede, levando a um possível congestionamento; ou
- se a freqüência de atualização das réplicas for baixa as informações ficam desatualizadas por um período de tempo muito grande, o que para aplicações que manipulam informações altamente dinâmicas, como é o caso de aplicações sensíveis ao contexto, não é desejável.

Na replicação parcial apenas as informações novas e são replicadas. Apesar de reduzir a quantidade de informação a ser replicada, há ainda a possibilidade de haver congestionamento se existir um número muito grande de servidores-réplicas. A situação mais adequada é que haja um equilíbrio entre a freqüência de atualizações parciais e o tamanho da base de dados.

A organização da informação utilizada pelos serviços de diretório obedece a uma classificação, como listas amarela, verdes, ou brancas. As descrições de serviço suportadas são pobres, no que diz respeito à semântica. O *Light Weight Directory Access Protocol (LDAP)*, por exemplo, utiliza uma classificação baseada em valores e atributos, que representam objetos como país, organização, pessoas e grupos [Johner et al. 1999]. Já o *Universal Description, Discovery and Integration (UDDI)* utiliza a linguagem XML para descrição dos objetos [UDDI 2000]. Apesar de ser um grande avanço em relação a outras propostas, esta forma de descrever serviços ainda é insuficiente para garantir uma seleção que usa informação semântica. Para melhorar isso, é necessário que seja suportada uma descrição mais rica, baseada em uma ontologia, por exemplo OWL-S.

Há ainda questões relacionadas com a estruturação e distribuição dos diretórios. Eles podem estar todos no mesmo nível, ou dispostos hierarquicamente em vários níveis, semelhante a estrutura do DNS na Internet. Uma vantagem dessa estruturação é a escalabilidade do serviço.

Devido à sua característica altamente dinâmica, sistemas sensíveis ao contexto requerem mecanismos de descoberta que permitam encontrar os serviços desejados pelas aplicações. Uma forma de auxiliar o processo de descoberta é utilizar as funcionalidades dos serviços de diretório, como um modelo distribuído para armazenagem, capacidades de pesquisa avançada e replicação de informação, para melhor atender às solicitações de descoberta feitas pelas aplicações.

3.6 Segurança

Há informações que são consideradas não confidenciais, como a temperatura ou a hora local. Há também as que são consideradas confidenciais, como informações de usuário, informações médicas, entre outras. Para estas é muito importante que haja algum meio de torná-las seguras, inacessíveis a pessoas não autorizadas. Uma maneira de se conseguir

isso é através de políticas de segurança que codifiquem informações e exijam autenticação de usuários, ou autorização através de chaves de criptografia.

No que diz respeito à descoberta de serviços há poucos protocolos que se utilizam das funcionalidades de segurança e autenticação [Zhu, Mutka e Ni 2002]. Esses protocolos adotam esquemas de segurança muito simples, devido à necessidade de serem automáticos, leves e de minimizar o uso da rede.

A seguir algumas abordagens para garantir a confidencialidade, a integridade, o não-repúdio e autenticação de usuários e informações são apresentadas.

3.6.1 Criptografia

Criptografia, do grego *kryptós* que significa escondido e *graphos*, escrita, é a ciência de proteger dados. Ela envolve quatro grandes aspectos: confidencialidade, autenticação, integridade e não-repúdio. A cifragem, transformação de dados em uma forma ilegível utilizando uma chave de criptografia, garante a privacidade e a confidencialidade de dados, mantendo a informação escondida de qualquer um sem autorização [Parmar 2005].

Para cifrar dados é utilizada uma metodologia, que utiliza:

- um ou mais algoritmos de criptografia, baseados em uma fórmula matemática;
- chaves de criptografia;
- um sistema gerenciador de chaves de criptografia (nem sempre necessário);
- os dados; e
- os dados cifrados.

O processo consiste em cifrar os dados empregando um algoritmo e uma chave de criptografia para gerar os dados cifrados. Para decifrar utiliza-se o mesmo algoritmo com a mesma chave para obter os dados na forma original.

Dependendo da forma como as chaves são criadas, a criptografia pode ser dividida em dois grandes grupos: simétrica e assimétrica. A simétrica utiliza uma mesma chave privada para cifrar e decifrar dados. Com esse método são resolvidos o não-repúdio e a autenticação. A autenticação é conseguida de modo implícito, pois se uma entidade consegue decifrar os dados enviados por outra é porque as duas entidades possuem a mesma chave privada [Parmar 2005]. Para o mesmo nível de proteção, os métodos simétricos são muito mais rápidos que os assimétricos, pois utilizam chaves cujo tamanho é pequeno e podem cifrar grandes quantidades de dados. Há, entretanto, o problema de distribuição da chave privada, comum a todos os sistemas de criptografia. Uma abordagem de distribuição empregado é o envio da chave privada dividida em várias partes.

A criptografia assimétrica, também chamada de criptografia de chave pública (*Public Key Cryptography - PKC*), utiliza duas chaves diferentes para cifrar e decifrar dados. Uma dessas chaves, a chave pública, é divulgada e a outra, a privada, é mantida secreta. Ambas as chaves podem ser empregadas para cifrar e decifrar dados, mas se uma for usada para cifrar, apenas a outra pode ser usada para decifrar [Parmar 2005, Stallings 1995]. A distribuição da chave privada é realizada cifrando, com a chave pública, essa chave privada e anexando-a aos dados cifrados. A distribuição da chave pública pode ser conseguida através da utilização de autoridades certificadoras, de quem as partes envolvidas na comunicação solicitam uma cópia da chave pública usada para cifrar a chave privada.

Uma Autoridade de Certificação (*Certification Authorities - CA*) é um ponto comum de confiança entre as partes envolvidas na comunicação e garante que o certificado digital é válido e confiável. A principal função de uma CA é gerar e gerenciar certificados digitais, que são documentos eletrônicos assinados digitalmente. Esses certificados fazem a associação da identidade de uma pessoa ou entidade a uma chave pública, ou seja, um certificado digital é um pedaço de dados binários assinados que contêm uma ou mais chaves públicas [Zwicky, Cooper e Chapman 2000]. Algumas partes são legíveis para os seres humanos, outras não. As assinaturas em um certificado são feitas de modo a não serem forjadas. Um certificado contém, em geral, as seguintes informações:

- nome da pessoa ou entidade a ser associada à chave pública;
- período de validade do certificado;
- chave pública;
- nome e assinatura da entidade que assinou o certificado; e
- um número de série.

3.6.2 Autenticação

Autenticação é o processo pelo qual se verifica a identidade do usuário e se é permitido a ele ter acesso a um sistema ou recurso. O método de autenticação consiste de uma política de segurança que solicita do usuário informações que apenas ele conhece. Essas informações podem ser alguma coisa que o usuário possui, como um cartão magnético, ou alguma coisa que o usuário conhece, como uma senha, ou mesmo alguma coisa que torna o usuário único, como a sua impressão digital, retina, etc. Quanto mais dessas informações são utilizadas, mais é garantido que a identidade do usuário é verdadeira [Parmar 2005].

Alguns mecanismos de autenticação utilizados atualmente são:

- baseado em senha;
- baseado em dispositivos eletrônicos ou magnéticos, como cartões magnéticos;
- baseado em biometria;
- gerador de senha;
- baseado em imagens; e
- baseado em senhas não reutilizáveis.

Um dos mais utilizados mecanismos é o par conta/senha. Entretanto, este mecanismo está longe de ser o mais seguro, pois apresenta algumas questões que o tornam um dos mais vulneráveis.

A criação de senhas é um processo antes de tudo subjetivo, pois cada usuário possui um método peculiar para escolher uma senha. Normalmente, os usuários tendem a criar senhas que são fáceis de serem lembradas e também fáceis de serem adivinhadas. Para contornar esta situação, geradores automáticos de senhas são utilizados para criar senhas a partir de caracteres aleatórios ou randômicos. Para os sistemas este mecanismo é muito

útil. Para os usuários, entretanto, não é interessante, porque as senhas criadas são tão difíceis de serem lembradas que os usuários procuram guardá-las em lugares de fácil acesso. A escolha de uma senha mais adequada deve, portanto, equilibrar uma senha fácil de lembrar e difícil de ser descoberta (manualmente ou automaticamente).

3.7 Conclusão

Serviços provêem mecanismos, ou um conjunto de meios dispostos para atingir um objetivo. Na computação sensível ao contexto, serviços são utilizados para informar sobre mudanças ocorridas no ambiente às aplicações, que podem então se adaptar ao novo contexto.

As aplicações sensíveis ao contexto, para que possam utilizar os serviços, necessitam de um mecanismo de descoberta provido de funcionalidades, que automaticamente façam inferências sobre as descrições dos serviços e levem em consideração a natureza da informação contextual. Uma maneira de se conseguir isto é através da utilização ontologias, que além de permitirem descrever semanticamente os serviços, possibilitam utilizar processos de comparação mais complexos para selecioná-los.

Os serviços de diretório desempenham um papel fundamental no processo de descoberta, pois possuem características e funcionalidades muito úteis como um modelo distribuído para armazenamento, organização e gerenciamento, capacidades de pesquisas avançadas e replicação de informação.

Algumas classes de serviços devem ter o acesso permitido a determinados grupos de aplicações. Serviços médicos, monitoração de batimentos cardíacos de um paciente, por exemplo, devem ser descobertos apenas por aplicações médicas. Isto sugere a utilização de políticas de segurança e métodos de autenticação que garantam a confidencialidade, integridade e o não-repúdio dos dados trocados entre aplicações e sistemas sensíveis ao contexto.

Essas características são exploradas pelo protocolo proposto neste trabalho e empregadas em seu projeto, apresentado no capítulo 5 desta dissertação.

Capítulo 4

Descoberta de Serviço

4.1 Introdução

Descoberta de serviço é um termo genérico utilizado para descrever os protocolos e mecanismos que permitem a um dispositivo ou um serviço se tornarem sensíveis à rede a qual estão conectados e descobrirem outros serviços disponíveis [Rusnak 1999].

Descoberta de serviço não é um assunto novo. O primeiro protocolo de descoberta de serviço desenvolvido foi o *Service Location Protocol*, proposto pela IETF, em 1997. Atualmente há outros protocolos em utilização no mercado, tais como o *Simple Service Discovery Protocol* da *Microsoft*, que é utilizado no UPnP, o *JINI* da *Sun Micrsystems*, o *Salutation* do *Salutation Consortium* e o *Secure Service Discovery Service* de Berkeley.

Em computação ubíqua as pessoas estão rodeadas de dispositivos computacionais, que por sua vez provêem informações e serviços. O número desses dispositivos pode ser muito grande, o que dificulta o gerenciamento de todos eles. A descoberta de serviço tem aqui um papel fundamental, pois facilita o gerenciamento dos serviços disponíveis através de mecanismos dinâmicos. Há, entretanto, algumas questões que ainda merecem mais atenção.

Muitos mecanismos de descoberta propostos até hoje são pobres no suporte a descrição semântica de serviços e na segurança provida ao usuário e aos serviços. Além disso, não utilizam a informação contextual e não consideram sua natureza no processo de descoberta. Em sistemas sensíveis ao contexto, por exemplo, é fundamental um mecanismo de descoberta de serviços que considere tais requisitos.

Neste capítulo, inicialmente são discutidas as diferenças existentes entre processos de localização e de descoberta. Posteriormente são analisados alguns protocolos e mecanismos de descoberta existentes na literatura. Finalmente é feita uma avaliação dessas abordagens com base nos requisitos desejáveis de um protocolo de descoberta de serviços para sistemas sensíveis ao contexto.

4.2 Localização *versus* Descoberta

Segundo [McGrath 2000], a localização é referida como a ação de encontrar um objeto através de pesquisas em uma ou várias bases de dados. A localização é estática, ou seja, é necessário consultar diretamente uma base de informação sobre o objeto desejado, ou utilizar um agente que intermedie a consulta nessa base. A descoberta, ao contrário, é um processo dinâmico. Através dela entidades descobrem-se umas às outras sem a necessidade

de que bases de dados sejam consultadas. Além disso, essas entidades se apresentam com informações atualizadas, o que pode não acontecer no processo de localização.

Há uma diferença entre esses dois processos: enquanto na localização uma base de dados é necessária para armazenar e tornar disponíveis as informações sobre as entidades procuradas, na descoberta a base de dados não é obrigatória, o que permite às próprias entidades se apresentarem e proverem as informações requisitadas. No caso da descoberta utilizar uma base de dados, o que é interessante em ambientes compostos por várias redes e onde há políticas de segurança que restringem o acesso à rede local, é necessário que as informações contidas na base sejam periodicamente atualizadas.

Tanto a localização quanto a descoberta podem obedecer a um processo passivo e/ou ativo. No processo passivo o cliente recebe atualizações periódicas sobre os serviços disponíveis. No ativo, o cliente inicia a descoberta ou localização do serviço desejado.

Como exemplo de protocolo de localização pode-se citar o utilizado no *Domain Name System* [Albitz e Liu 1998], que provê uma base de dados com informações sobre computadores disponíveis na *Internet*. Neste a pesquisa é ativa e é realizada nessa base. Como exemplo de um protocolo de descoberta pode-se citar o *Universal Plug-and-Play* [Forum 2003], que descobre as entidades enviando solicitações, através de *broadcast*, aos dispositivos presentes no ambiente e estes dispositivos respondem às solicitações enviando suas informações ao requisitante. No UPnP não é empregada uma base de dados de informações sobre os dispositivos.

A tabela 4.1 resume as características dos processos de localização e descoberta.

Processo	**Característica**
Localização	- Utiliza base de dados/arquivos estáticos; - É mantido por administradores com privilégios; - Não garante disponibilidade de objetos registrados; - Não garante anúncio quando recursos são registrados ou removidos das bases de dados.
Descoberta	- Descoberta espontânea e configuração de entidades na rede; - Baixo (preferencialmente nenhum) requisito de interação humana; - Adaptação automática à disponibilidade móvel e esporádica; - Interoperabilidade entre fabricantes e plataformas.

Tabela 4.1: Classificação dos processos de descoberta e localização.

Devido às suas características, o processo de descoberta se apresenta como o mais adequado para encontrar serviços em sistemas sensíveis ao contexto. Mas, para que atenda às necessidades de tais sistemas, alguns requisitos, como anúncio, consulta, taxa de utilização da rede, interoperabilidade e disponibilidade do serviço, precisam ser melhor investigados.

Todos os protocolos de descoberta de serviços possuem dois mecanismos básicos: anúncio e consulta. Esses mecanismos são os responsáveis por disponibilizar os serviços aos usuários. Estão presentes tanto em protocolos que utilizam serviço de diretório, quanto naqueles que não o utilizam. Os usuários empregam-nos para aprender sobre quais serviços estão disponíveis na rede. Os anúncios e as consultas podem ser feitos por meio de *unicast*, *multicast* e *broadcast*. Cada um desses meios possui vantagens e desvantagens. Por exemplo, *broadcast* é muito útil quando não existe serviço de diretório presente na rede, entretanto possui a desvantagem de gerar tráfego, aumentando a taxa de utilização da rede. O *multicast* elimina a possibilidade de clientes que não utilizam o protocolo

receberem mensagens, mas muitos administradores bloqueiam pacotes *multicast* em seus roteadores. Com *unicast* o cliente precisa saber o endereço do serviço de diretório e do provedor de serviço. Assim, as mensagens são diretamente enviadas para eles.

Dependendo de como e da freqüência com que são feitos os anúncios dos serviços, o tráfego na rede pode ser muito alto. Uma boa abordagem é utilizar UDP, pois é um protocolo de transporte rápido, leve e muito eficiente para troca de mensagens entre servidores e um número muito grande de clientes.

A interoperabilidade é outra questão importante. Os protocolos existentes atualmente não adotam um padrão, não sendo possível a comunicação entre eles. Uma conseqüência direta é a impossibilidade de, em uma mesma rede, clientes que utilizam um protocolo conseguirem descobrir serviços disponíveis através de outro protocolo. Por exemplo, um usuário utiliza uma aplicação que faz uso de um protocolo "A" de descoberta de serviços. Esse usuário, em visita a uma empresa, depara-se com a impossibilidade de encontrar serviços porque no ambiente computacional da empresa é utilizado outro protocolo de descoberta, identificado como "B". Como entre os dois protocolos não há interoperabilidade, cabe ao usuário, caso queira encontrar e utilizar os serviços desejados, instalar a aplicação que implementa o protocolo "B" escolhido pela empresa. Caso houvesse algum mecanismo de compatibilização, como um anúncio padrão para os protocolos de descoberta, o usuário poderia utilizar sua aplicação, ou seja, para ele seria transparente encontrar serviços. Há uma iniciativa proposta por [Livingstone 2003], que implementa uma ponte entre os protocolos *JINI* e *Twine*, mas no mercado não há qualquer utilização dessa proposta.

A disponibilidade do serviço, ou seja, a garantia de que o serviço está ativo, é também tratada por muitos protocolos de descoberta. Há duas maneiras de se obter informação sobre a disponibilidade do serviço: consultar o provedor de serviço, ou este se anunciar periodicamente. Na primeira, há a vantagem de não aumentar a taxa de utilização da rede, pois o provedor é consultado somente quando alguma informação sua for solicitada. Para o usuário, entretanto, é necessário que conheça o endereço do provedor, o que não é uma tarefa simples quando existe um número muito grande de provedores. Na segunda, apesar da desvantagem de aumentar a taxa de ocupação da rede caso o número de provedores de serviço seja muito grande, o usuário não precisa se preocupar em ter uma lista de todos os provedores existentes no ambiente onde está inserido. Uma alternativa, que une as vantagens das duas abordagens, é fazer com que todos os provedores de serviço se anunciem diretamente a um serviço de diretório. Assim, mesmo havendo um número grande de provedores de serviço, os anúncios são direcionados para uma única entidade na rede e, além disso, o usuários têm apenas que conhecer o endereço dessa entidade e fazer uso das funcionalidades providas pelo serviço de diretório provido por ela.

4.3 Protocolos de Descoberta

Como visto anteriormente, os protocolos de descoberta de serviço provêem meios de automaticamente descobrir serviços para usuários, que podem selecionar os serviços desejados ou deixar que o protocolo realize essa tarefa. Uma outra funcionalidade é a configuração automática de dispositivos de rede, que auxilia administradores nessa difícil tarefa quando a rede assume grandes proporções.

Vários protocolos foram propostos e cada um dá ênfase a aspectos específicos. Por exemplo, o *Salutation* se propõe a ser uma abordagem independente de processador, sis-

tema operacional e protocolo de comunicação; o SLP permite que aplicações descubram a existência, localização e configuração de serviços; o *JINI* se propõe a habilitar dispositivos a se comunicarem entre si sem qualquer planejamento, instalação ou intervenção humana; o UPnP se baseia em uma arquitetura para conectividade de redes ponto-a-ponto; e o SSDS dá ênfase na segurança.

A seguir esses protocolos são analisados para verificar se sistemas sensíveis ao contexto podem fazer uso de suas funcionalidades no processo de descoberta de serviços.

4.3.1 *Salutation*

A arquitetura *Salutation* foi a primeira a prover um protocolo de descoberta de recursos comercialmente disponível. Sua proposta é resolver problemas de descoberta e utilização de serviços de um conjunto de dispositivos e equipamentos em um ambiente com conectividade e mobilidade amplas [Pascoe 1998]. Tem a característica de ser uma abordagem independente de processador, sistema operacional e protocolo de comunicação, além de permitir sua implementação em dispositivos com poucos recursos computacionais [Miller 1999]. Isso é conseguido através de uma central de serviços que abstrai as características de cada entidade para aplicações, serviços e dispositivos. Pode também operar em qualquer tipo de ambiente computacional, seja em uma rede sem fio ou com fio.

A proposta de uma arquitetura totalmente independente do tipo de tecnologia surgiu devido à grande variedade de formatos de apresentação da informação. Um grande número de dispositivos e aplicações, com suas restrições tecnológicas, ditam aos usuários uma dependência no que diz respeito ao modo de acesso à informação. Um usuário com um dispositivo móvel como um aparelho celular, por exemplo, não tem condições de visualizar um documento no formato PDF. Para poder ter acesso a esse documento é necessária uma transformação para o formato suportado pelo aparelho. *Salutation* veio poupar o usuário dessa e de outras restrições tecnológicas, como a incompatibilidade entre os meios de comunicação, provendo meios de codificação da informação.

A arquitetura *Salutation* procura atender aos requisitos de um protocolo de descoberta de serviços, provendo um método padrão para que aplicações e recursos descrevam e anunciem suas informações a outras aplicações e recursos. Permite também que informações sejam determinadas através de consulta a outros recursos.

O modelo abstrato definido possui três componentes:

- Cliente;
- Serviço/dispositivo (recurso); e
- *Salutation Manager* (SLM).

Na arquitetura *Salutation* os clientes e recursos são também chamados de entidades de rede [Miller 1999]. A Figura 4.1 ilustra a arquitetura *Salutation*.

O SLM é o intermediário entre aplicações/serviços e o meio de comunicação. As aplicações acessam os serviços providos pelo SLM através de uma interface, a *Salutation Manager Application Program Interface (SLM-API)*. O SLM é, portanto, uma facilidade de gerenciamento que abstrai para o desenvolvedor de aplicações e serviços a manipulação direta dos protocolos da arquitetura.

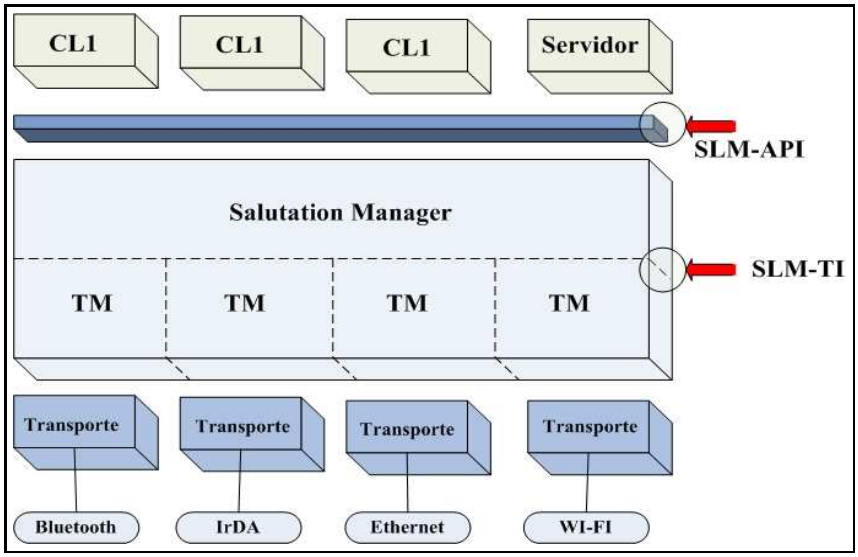

Figura 4.1: Arquitetura *Salutation*.

Um SLM se comunica com outro SLM para executar seus serviços como um *service broker*. Conforme é ilustrado pela Figura 4.2, essa comunicação é feita através de um protocolo de comunicação definido pela arquitetura, chamado *Salutation Manager Protocol*.

O *Transport Manager* é utilizado para conectar entidades SLM e abstrair os protocolos de comunicação específicos utilizados. Há um gerente de transporte para cada protocolo de comunicação, por exemplo, gerente de transporte TCP/IP; gerente de transporte *Bluetooth*; gerente de transporte *IrDA*. O gerente de transporte oferece uma interface denominada *Salutation Manager Transport Interface (SLM-TI)*. Assim, um dispositivo que utiliza *Bluetooth* pode descobrir um serviço situado em uma rede *Ethernet*.

O SLM permite que entidades de rede descubram e utilizem as capacidades de outras entidades de rede. O modelo abstrato pode ser implementado com ou sem um serviço de diretório. Se for implementado sem utilizar um serviço de diretório, as entidades de rede podem se localizar utilizando consulta por difusão (*broadcast*). Neste caso, a descoberta é permitida apenas na rede local, restrição essa causada pelas regras de segurança impostas pelos equipamentos de borda (roteadores, *firewalls*) da rede. Se for implementado utilizando um serviço de diretório, as entidades de rede registram-se nesse serviço. Os diretórios podem ser organizados como uma hierarquia ou um grafo cooperativo.

Os serviços básicos providos pelo SLM são de registro, descoberta, disponibilidade e gerenciamento de sessão. O serviço de descoberta é realizado através de comparação nominal de tipos de serviço e atributos. Os clientes podem encontrar serviços enviando requisições solicitando todos os serviços; solicitando serviços de u tipo específico; ou solicitando serviços com atributos específicos.

4.3.2 *Service Location Protocol (SLP)*

O *Service Location Protocol (SLP)* permite que aplicações descubram a existência, localização e configuração de serviços em uma rede de computadores [Systems 2005]. Até o surgimento do SLP nenhum protocolo de descoberta havia sido padronizado. Atualmente ele está na versão 2, sendo que há um esboço da versão 3.

Inicialmente desenvolvido pela *Sun Microsystems*, o SLP tornou-se um padrão IETF

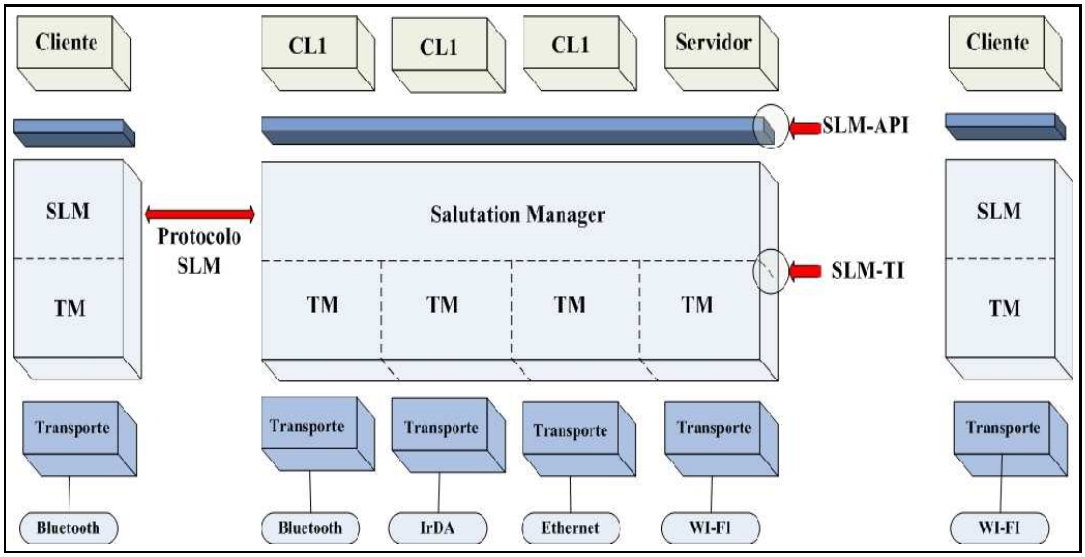

Figura 4.2: Modelo do SLM.

para descoberta de serviços. Antes desse protocolo, o usuário tinha de saber o nome do computador provedor do serviço para encontrar o serviço desejado. Com o SLP o usuário provê o tipo de serviço desejado e um conjunto de atributos que descrevem o serviço e, com base nessas informações, o protocolo retorna o endereço de rede do serviço para o usuário.

Há quatro conceitos importantes no SLP:

- Agentes;
- Mensagens;
- A maneira como as mensagens são enviadas; e
- SLP API.

Existem três tipos de agentes no SLP: *User Agent (UA)*, *Service Agent (SA)* e *Directory Agent (DA)*. O UA é um agente que procura pela localização de um ou mais serviços; o SA é um agente que anuncia a localização de um ou mais serviços; e o DA é um agente que atua como um repositório central para informação de localização de serviço.

As mensagens enviadas entre as entidades SLP são um total de 11 diferente tipos:

- *Service Request (SrvRqst)*
- *Service Reply (SrvRply)*
- *Service Registration (SrvReg)*
- *Service Deregister (SrvDeReg)*
- *Service Acknowledge (SrvAck)*
- *Attribute Request (AttrRqst)*
- *Attribute Reply (AttrRply)*

- *Service Type Request (SrvTypdRqst)*
- *Service Type Reply (SrvTypeRply)*
- *DA Advertisement (DAAdvert)*
- *SA Advertisement (SAAdvert)*

A maneira pela qual as mensagens são enviadas é outro conceito importante. Apesar de suportar *broadcast*, SLP é basicamente um protocolo que utiliza *unicast* e *multicast* [Guttman et al. 1999]. Os DAs e SAs aceitam requisições *unicast* e *multicast*. A descoberta se baseia em *multicast*. *Broadcast* funciona em redes isoladas no lugar de *multicast*. Dessa forma, os agentes que não suportam *multicast* podem utilizar o SLP nessas redes. Dependendo do número de dispositivos de rede, *broadcast* pode gerar muito tráfego, por isso requisições *unicast* e *multicast* são preferíveis.

A *SLP Application Programmer's Interface (SLP-API)* é a mais importante parte do SLP. Com ela os programadores podem utilizar o SLP em suas aplicações. Abaixo estão listadas algumas chamadas de funções:

- *SLPReg()*: registra uma URL e os atributos do serviço;
- *SLPDeReg()*: cancela o registro de um serviço anteriormente registrado;
- *SLPFindSrvs()*: encontra serviços com base no tipo ou nos atributos do serviço;
- *SLPFindAttrs()*: obtém uma lista de atributos dos serviços registrados;
- *SLPFindSrvTypes()*: obtém uma lista de tipos de serviços registrados.

O SLP foi projetado com o objetivo de anunciar serviços em um ambiente onde não há preocupação com confidencialidade e por isso não a provê. A RFC 2608 garante que, quando devidamente implementado, os agentes SLP suportam uma assinatura digital, que pode ser registrada no DA. As assinaturas são utilizadas para autenticar mensagens e devem ser geradas utilizando métodos externos ao protocolo. A autenticação, apesar de não ser especificado como é feita, é utilizada para garantir a integridade das mensagens.

O SLPv2 foi projetado para ser uma solução escalável para redes grandes, com roteadores separando os diversos segmentos de uma rede [Systems 2005].

Há duas possíveis configurações para o SLP: uma passiva e outra ativa, sendo que esta última sem o agente de diretório (DA). Na configuração passiva os DAs periodicamente anunciam serviços utilizando *multicast*. Sem os DAs, em uma rede *ad hoc* por exemplo, os UAs e SAs implementam todas as funções de um DA e se anunciam utilizando *multicast*. A Figura 4.3 ilustra a forma passiva e ativa do protocolo.

É fácil de verificar que a forma passiva é mais eficiente, pois os UAs e SAs aprendem sobre os DAs através de anúncios periódicos destes, sem ser necessário enviar mensagem para a descoberta, diminuindo assim o número de mensagens enviadas pela rede.

Como foi projetado para anunciar serviços, o SLP utiliza mensagens que cabem em um pacote com MTU de 1500 *bytes*. Por isso, o protocolo UDP da camada de transporte é preferencialmente utilizado. Entretanto, caso a descrição do serviço exceda o tamanho do pacote, o UA pode solicitar ao DA que a descrição seja enviada utilizando TCP.

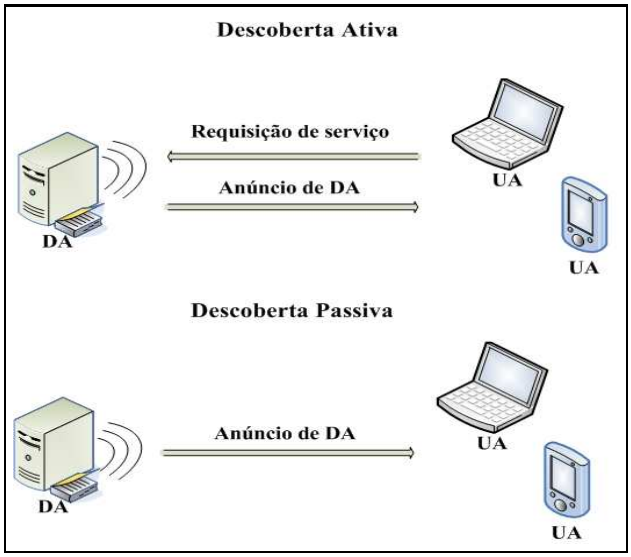

Figura 4.3: Métodos de descoberta do DA.

Existem várias implementações do protocolo, entre elas destacam-se os da *Sun Microsystems*, *Novel*, *IBM*, *Axis Communications*, *Lexmark*, *Madison River Technologies* e *HP* que adotaram o SLPv1 [Guttman 1999]. Em [Cha et al. 2003] foi implementada uma versão para IPv6, baseada na versão 2 do SLP.

4.3.3 JINI

JINI é uma infra-estrutura/modelo de programação que permite o desenvolvimento e a configuração dinâmica de sistemas distribuídos [Jini 2003]. Foi desenvolvido pela *Sun Microsystems* com o objetivo de habilitar dispositivos a se comunicarem entre si sem qualquer planejamento, instalação ou intervenção humana. Além disso, cada dispositivo provê serviços que outros dispositivos podem utilizar [Pascoe 1999]. Os dispositivos se comunicam utilizando *Java Remote Method Invocation - RMI*. A descoberta é feita através da movimentação de objetos Java entre as máquinas virtuais. Na terminologia *JINI*, os dispositivos (que podem ser *software* ou *hardware*) são denominados de serviços.

Existem três componentes no *JINI*:

- Clientes;
- Serviços; e
- *Service Lookup* (Serviço de Descoberta).

O serviço de descoberta é utilizado para descobrir os serviços requisitados pelos clientes. Após isso a comunicação se dá somente entre cliente e o serviço requisitado. A seqüência de passos no processo de descoberta, ilustrada pela Figura 4.4, é a seguinte:

1. O dispositivo conecta-se à rede através do protocolo *Discovery and Join*;
2. O provedor de serviço procura o Serviço de Descoberta para realizar seu registro;

3. Após encontrar o Serviço de Descoberta o provedor de serviço envia um objeto *Service Registrar* a ele. Este objeto funciona como um *proxy* local para o Serviço de Descoberta. Além disso, ele implementa a interface do serviço de descoberta e o protocolo de comunicação entre o cliente e o Serviço de Descoberta;

4. O cliente por sua vez precisa encontrar o serviço antes de utilizá-lo, para isso conecta-se ao Serviço de Descoberta e requisita o objeto do serviço (*proxy*).

5. A comunicação entre o cliente e o serviço requisitado é feita pelo objeto de serviço, sem a interação do Serviço de Descoberta.

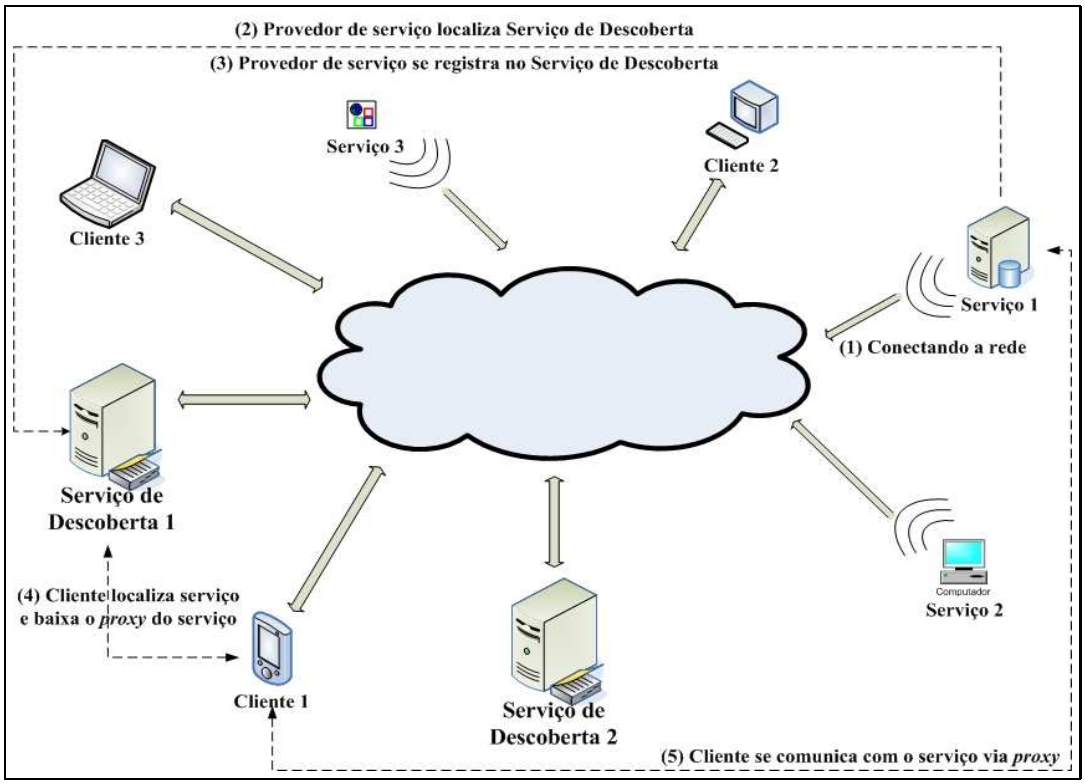

Figura 4.4: Sistema *JINI* típico.

A infra-estrutura tecnológica do *JINI* é composta por:

- *Lookup Service* (Serviço de Descoberta);
- *Service Join Protocol* (Protocolo de Junção de Serviço);
- Mecanismo *Client Lookup* (Descoberta do Cliente); e
- *Leasing*, (um período de tempo calculado para determinar a validade de um serviço).

O protocolo de junção de serviço é responsável por registrar os serviços no serviço de descoberta. Esse protocolo foi implementado utilizando *Java Remote Method Invocation - RMI*.

O mecanismo *Client Lookup* é a maneira pela qual os clientes realizam a pesquisa por serviços. Clientes se comportam da mesma maneira que os serviços quando se anunciam

ao serviço de descoberta. O serviço de descoberta responde aos clientes enviando um *proxy* de objeto de serviço ou um conjunto de itens de serviço, contendo cada um deles um *proxy*, como também qualquer atributo associado com o *proxy*. Os clientes podem também pedir ao serviço de descoberta que envie notificações de mudança de serviços, aparecimento de novos serviços ou a indisponibilidade de serviços já registrados.

O acesso aos serviços no *JINI* é baseado em *leasing*, que é negociado entre o cliente e o provedor do serviço. Com esse mecanismo o serviço de descoberta pode gerenciar a liberação de entradas de serviços que não podem se remover do registro por si mesmas, após o tempo limite expirar. Com isso os clientes não precisam perder tempo com serviços que não estão ativos.

Os clientes e os serviços encontram um serviço de descoberta tanto através de *unicast* como *multicast*. Após isso, os clientes registram-se no serviço de descoberta.

JINI provê os seguintes protocolos:

- O *Multicast Request Protocol*, utilizado pelos serviços quando precisam descobrir um serviço de descoberta em uma rede local;
- O *Multicast Announcement Protocol* é utilizado para anunciar periodicamente a presença de serviços de descoberta em uma rede local; e
- O *Unicast Discovery Protocol* é utilizado para fazer requisições diretamente a um serviço de descoberta. Este protocolo funciona apenas se o cliente ou serviço sabem de antemão onde o endereço do serviço de diretório está funcionando. É empregado comumente para acessar serviços de descoberta que não estão na rede local.

4.3.4 *Simple Service Discovery Protocol (SSDP)*

A tecnologia *Universal Plug and Play* define uma arquitetura para conectividade de rede ponto-a-ponto pervasiva de dispositivos sem fio, computadores pessoais e serviços [Forum 2003]. Ela estende o *Plug and Play* para suportar redes e descoberta e configuração ponto-a-ponto [Pascoe 1999]. Para realizar a descoberta o *Simple Service Discovery Protocol (SSDP)* utiliza o HTTP sobre UDP através de *multicast* e *unicast* com duas funções:

- *OPTIONS*: que determina se um serviço existe na rede; e
- *ANNOUNCE*: utilizado pelos serviços para anunciar sua existência.

O SSDP é responsável por realizar a descoberta e deixa descrições adicionais de serviço e/ou negociação para camadas mais altas da arquitetura. É definido para um ambiente IP, requerendo portanto *multicast* e *unicast* UDP e sintaxe de mensagem HTTP. É limitado a uma rede local pequena como de uma casa ou de um escritório. Esse escopo é devido ao objetivo do protocolo que é prover descoberta para redes locais e não para toda *Internet*.

Com a utilização do *multicast* é mais fácil uma solução que não exija intervenção humana para configuração. Dessa forma, qualquer um pode conectar-se à rede e utilizar o serviço de descoberta [Goland et al. 1999].

É feita uma classificação dos dispositivos em dois tipos: os dispositivos controlados e os pontos de controle. O dispositivo controlado funciona como um servidor e responde às requisições dos pontos de controle.

Como é baseado em IP, os dispositivos de uma rede UPnP devem possuir um cliente *Dinamic Host Configuration Protocol - DHCP* e pesquisar com um servidor DHCP quando o dispositivo se conecta à rede. No caso de não encontrar um servidor DHCP, o dispositivo deve atribuir a si mesmo um IP, ou seja utilizar o *Auto IP*.

Após isso o passo seguinte é o processo de descoberta. O protocolo de descoberta do UPnP permite ao dispositivo anunciar seus serviços aos pontos de controle. Cada ponto de controle, por sua vez, utiliza o protocolo para descobrir dispositivos na rede UPnP. A troca de mensagem realizada por ambos os tipos de dispositivos é fundamental. A mensagem contém essencialmente o tipo, o identificador único universal e uma URL para qualquer informação mais detalhada.

O próximo passo é a descrição. O ponto de controle não sabe muito sobre o dispositivo quando o descobre. Para saber mais é preciso que acesse as informações contidas na URL informada na mensagem do protocolo de descoberta. Essas informações são descritas em XML e contêm informação específica do dispositivo. Os dispositivos podem ser compostos por dispositivos lógicos, unidades funcionais ou serviços. Para cada serviço há uma lista de comandos e argumentos para cada comando. A descrição do serviço também inclui uma lista de variáveis que modelam o estado do serviço em tempo de execução.

Após obter uma descrição mais detalhada do dispositivo e do serviço provido por ele, o ponto de controle pode enviar comandos ao serviço do dispositivo. Para realizar isto é utilizado o protocolo SOAP, através do qual são enviadas mensagens de controle escritas em XML. A resposta são valores específicos referentes aos comandos executados.

Após a execução de comandos, o estado do dispositivo muda. Quando isto acontece, o serviço publica a mudança de estados através da atualização das variáveis que controlam o estado do dispositivo. Os pontos de controle podem se inscrever para receber notificações, feitas através de mensagens de eventos, sobre as mudanças ocorridas. Estas mensagens são expressas em XML e contêm uma ou mais variáveis de estado e o valor corrente dessas variáveis. Em um cenário com muitos pontos de controle, todos eles recebem mensagens de eventos. Assim, todos os pontos de controle são igualmente informados sobre os resultados das execuções de comandos.

Caso o dispositivo tenha uma URL para apresentação, o ponto de controle pode ler a página dessa URL e permitir ao usuário, dependendo das propriedades da página, controlar e/ou visualizar o estado do dispositivo.

UPnP é semelhante ao *JINI* no que diz respeito ao anúncio dos dispositivos, tanto *multicast* como *unicast*. Entretanto, provê uma descrição mais rica, pois utiliza XML, enquanto que *JINI*, *Salutation* e SLP são pobres nesse quesito, pois utilizam descrições baseadas em objetos Java, nomes e atributo-valor, respectivamente. Outro aspecto importante é que não é necessário um servidor central para que a descoberta seja realizada. UPnP foi projetado para que não seja necessária muita intervenção humana.

UPnP depende da infra-estrutura de rede e da *Web* para que possua algum mecanismo de segurança, ou seja, a segurança é fornecida por mecanismos externos ao protocolo, como políticas de segurança implementadas em equipamentos, ou uma camada de segurança provida pelo SSL ao HTTP.

4.3.5 *Secure Service Discovery Service (SSDS)*

Secure Service Discovery Service (SSDS) [Czerwinski et al. 1999] foi desenvolvido em Java e utiliza XML para descrição dos serviços. SSDS é parte do projeto *Ninja Service Disco-*

very Service, uma plataforma escalável de serviços, tolerante a falhas e distribuída. SSDS é um repositório seguro e provê aos clientes acesso a todos os serviços através de uma estrutura de diretório.

Há dois tipos de informação armazenadas no SSDS:

- descrições de serviços, que estão disponíveis para execução em recursos computacionais na rede; e
- serviços que estão em execução em algum lugar da rede.

São suportadas descoberta passiva e ativa. A primeira é realizada através de anúncios *multicast*; a segunda é realizada utilizando consultas aos servidores SSDS. As mensagens *multicast* contêm URLs para o servidor SSDS disponível.

Os clientes têm de requisitar informações de pelo menos um servidor SSDS, pois são eles os responsáveis por armazenar informações sobre os serviços. Em um ambiente *ad hoc* torna-se necessário que uma das entidades seja um servidor SSDS, mas isso não é definido na documentação.

A arquitetura SSDS é composta por cinco componentes:

- Servidores;
- Serviços;
- Certificados de Autorização;
- Gerente de Capacidade; e
- Clientes.

A grande contribuição do SSDS, em relação aos outros protocolos de descoberta, é no que diz respeito a segurança, confiabilidade e escalabilidade que são implementadas utilizando criptografia de todas as informações trocadas entre as entidades do sistema. Dessa forma, o protocolo procura evitar ataques maliciosos. A comunicação entre clientes e servidores e entre serviços e servidores é protegida e a criptografia utilizada é um método tradicional assimétrico e simétrico. Além disso, é oferecido um mecanismo de autenticação entre as entidades através do uso de certificados. Estes certificados são assinados por autoridades certificadoras (CA).

A escalabilidade é conseguida através de uma estrutura hierárquica entre os servidores SSDS. A comunicação entre os servidores é feita com mensagens *multicast* enviadas periodicamente. Por exemplo, no caso de algum servidor ser reiniciado, ele pode popular sua base de dados ouvindo os anúncios dos serviços. Dessa forma é feita uma recuperação, não sendo necessária a implementação de um mecanismo de recuperação explícito [Czerwinski et al. 1999].

4.4 Outras Propostas de Infra-estruturas de Descoberta de Serviços

Algumas propostas de ambientes, arquiteturas e protocolos de descoberta de serviço têm sido feitas. Muitas procuram utilizar descrições de serviço mais ricas, serviços de diretório

para organizar e gerenciar informações, bem como adotar políticas de segurança e autenticação para o acesso às informações sobre serviços. A seguir, algumas dessas propostas são analisadas com o objetivo de verificar que requisitos, dos anteriormente discutidos, são atendidos.

4.4.1 *Service Locating Manager*

Em [Gu et al. 2003] é proposto um sistema gerenciador de localização de serviço (*Service Locating Manager - (SLM)*) que procura tratar questões de escalabilidade, disponibilidade, dinamismo e suporte a múltiplos mecanismos de comparação, por exemplo, de nomes, números e atributos.

O SLM adota uma estrutura hierárquica dinâmica baseada em árvore e agregação de serviço, como mostrado na Figura 4.5.

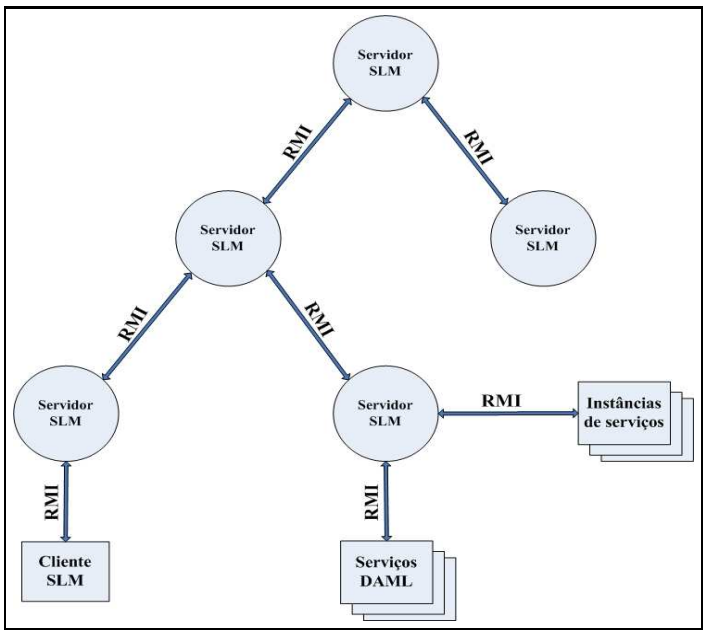

Figura 4.5: Sistema SLM [Gu et al. 2003].

O sistema possui três componentes: servidor SLM que é um repositório de informação de serviços; serviços, descritos em XML; e clientes SLM que pesquisam por serviços para os usuários. A comunicação entre os servidores e clientes é feita através de Java RMI.

A descoberta de serviços nessa proposta é feita através de consultas a uma base de informação sobre serviços, os quais se registram e têm suas informações propagadas em uma hierarquia de servidores SLM em forma de árvore. É uma abordagem que utiliza um serviço de diretório, com a diferença de que em vez de um único servidor são consultados vários.

O protocolo *JINI*, versão 1.2, foi tomado como modelo para projetar o SLM. Para garantir que os serviços requisitados estejam disponíveis, o SLM implementa o mecanismo de *leasing*, que indiretamente provê dinamismo ao serviço de descoberta.

Devido a sua característica hierárquica, o SLM define o *Server Connection Manager* que, utilizando *multicast* e *unicast* do *JINI*, procura por outros SLMs e constrói a árvore de servidores SLM.

Dois aspectos importantes que não são abordados pela proposta são segurança e autenticação.

Uma maneira de tratar a disponibilidade de serviços no SLM é a duplicação dos registros deles em todos os SLMs, ou seja, cada servidor SLM possui informações sobre os serviços disponíveis em todos os outros servidores SLM. Isto é uma vantagem importante para o usuário, que pode utilizar o serviço independentemente de onde esteja. Entretanto, um problema não tratado na proposta é o grau de preferência dada aos serviços encontrados. Por exemplo, ao serem encontrados dois serviços de impressão disponíveis, sendo um local e outro remoto, não fica claro qual dos dois é escolhido. Uma forma de tratamento é deixar que o usuário selecione aquele que melhor atenda as suas necessidades, ou o sistema utilizar as informações contextuais do usuário e dos provedores de serviço para selecionar o serviço que melhor atenda aos requisitos impostos por essas informações.

4.4.2 *Secure Pervasive Discovery Protocol - SPDP*

O trabalho apresentado em [Almenárez e Campo 2003] investiga algumas questões referentes à segurança e à descoberta de serviços em redes *ad hoc*. Propõe um protocolo, o *Secure Pervasive Discovery Protocol (SPDP)*, que não utiliza um servidor central para organizar as descrições dos serviços. Cada entidade gerencia os serviços que provê, anunciando-os e autorizando acesso a eles.

A proposta se baseia nos protocolos SLP, *JINI*, *Salutation*, *Splendor*, SSDP, SSDS e em um modelo de confiança anárquico, o qual não requer um servidor central e uma arquitetura hierárquica para tratar os desafios impostos por redes *ad hoc*. de infra-estrutura de chave pública. A principal motivação de propor esse protocolo foi a carência dos protocolos existentes para tratar segurança. Mesmo os poucos existentes, como o SSDS [Czerwinski et al. 1999], que foi projetado para redes fixas, possuem algumas deficiências quando vistos pelo prisma de redes móveis *ad hoc*.

O protocolo proposto adota dois mecanismos de anúncio: *push* e *pull*. No primeiro o serviço é anunciado e os dispositivos-cliente o selecionam se estiverem interessados nele. No segundo, os dispositivos-cliente solicitam o serviço desejado e os dispositivos-provedor que oferecem o serviço respondem à solicitação. Aqui, cabe ao cliente escolher aquele que melhor atende às suas necessidades.

Uma vez solicitado um serviço, vários dispositivos poderão responder à solicitação. Isso obriga o dispositivo solicitante realizar uma escolha, que demanda processamento. Se a lista de serviços retornados for grande, o dispositivo solicitante poderá ter de empregar muito processamento, consumindo mais energia, o que é crucial para dispositivos móveis. Essa questão não é tratada no trabalho.

Outro objetivo do protocolo é compartilhar serviços de forma segura, adotando um modelo de confiança entre eles, que nada mais é do que uma autoridade de certificação (*Certification Authority - CA*).

Uma outra característica importante desse protocolo é que cada dispositivo possui uma lista de todos os outros dispositivos confiáveis, juntamente com o grau de confiabilidade associado a eles. Dependendo do grau de confiabilidade, um dispositivo decide armazenar o serviço oferecido por outro. Quando o dispositivo acessa os serviços, são primeiramente selecionados os serviços cujos provedores possuem maior grau de confiabilidade.

4.4.3 *Carmen*

Carmen é uma arquitetura de descoberta de serviços que une as características de serviços de descoberta, limitados às redes locais,x às características dos que utilizam servidores de diretório centralizados, que contêm informações desatualizadas e que não são sensíveis aos recursos locais [Marti e Krishnan 2002]. *Carmen* se propõe, portanto, a manter informações de serviços atualizadas e a permitir a qualquer provedor de serviço inserir descrições de seu serviço. Essa arquitetura é escalável a proporções globais,ou seja, os *proxies* e os clientes podem estar espalhados na *Internet*. É um mecanismo de descoberta de serviços destinado a ambientes onde os usuários são nômades, ou seja, que mudam de lugar constantemente.

A arquitetura pode ser descrita como uma aplicação de rede com três tipos de nós:

- Cliente *Carmen*: agente de usuário que solicita recursos. Podem ser móveis e conectam-se e desconectam-se com freqüência da rede.

- *Proxy Carmen*: provê uma conexão entre clientes e serviços. Estão localizados em servidores estáticos. O *proxy Carmen* é semelhante a um DA do SLP ou ao serviço de busca do *JINI*.

- Provedor de serviço *Carmen*: provedor de serviço. É também cliente do *proxy*, anuncia seu serviço ao *proxy*, que por sua vez propaga o anúncio na rede *Carmen*.

A seleção de serviço é feita através de consultas aos *proxies*, que por sua vez propagam a consulta na rede. Quando recebem a requisição, os provedores de serviço *Carmen* verificam se podem atender a ela e respondem diretamente ao requisitante, ou seja, ao cliente *Carmen*, com a informação desejada ou com uma URL do serviço. O resultado retornado pelo *proxy Carmen* pode ser uma lista de provedores de serviço e cabe ao cliente *Carmen* escolher qual o que atende às suas necessidades.

Quando um provedor de serviço deixa a rede, o *proxy* atualiza a lista de provedores de serviço e remove a informação do que deixou a rede. Desta forma, é garantido que os provedores de informação de serviço presentes na lista do *proxy* estão sempre disponíveis, o que faz de *Carmen* uma abordagem dinâmica.

O anúncio é composto do nome do serviço, uma descrição curta e uma URL. O atributo do serviço não é incluído no anúncio. As descrições são feitas em XML, de acordo com um esquema ou DTD gerais.

A consulta de serviços é feita através dos *proxies*. Cada mensagem de consulta possui um campo onde é especificado o número de pulos (*hopcounts*), que informa a distância na árvore *Carmen* permitida para a propagação da requisição.

A topologia é hierárquica. Essa estrutura permite que um grande número de serviços e clientes formem um serviço de descoberta distribuído, o que faz de *Carmen* uma arquitetura escalável.

Os clientes de *Carmen* localizam o *proxy* mais próximo através de uma configuração estática ou dinâmica. Na estática, o endereço do *proxy* é manualmente configurado. Pode ser utilizado um nome universal, como *http://local-carmen-proxy/* e assim, em qualquer ambiente onde exista um *proxy*, o cliente faz consultas a ele. Na configuração dinâmica o cliente obtém o endereço através de mensagens *multicast* ou da opção 78 do protocolo DHCP.

Uma das contribuições de *Carmen* é uma disposição hierárquica dos *proxies* para melhorar a eficiência da consulta por serviços. Para isso, utiliza papéis contextuais nos *proxies* e assim o tráfego pode ser diminuído. A Figura 4.6 ilustra essa hierarquia.

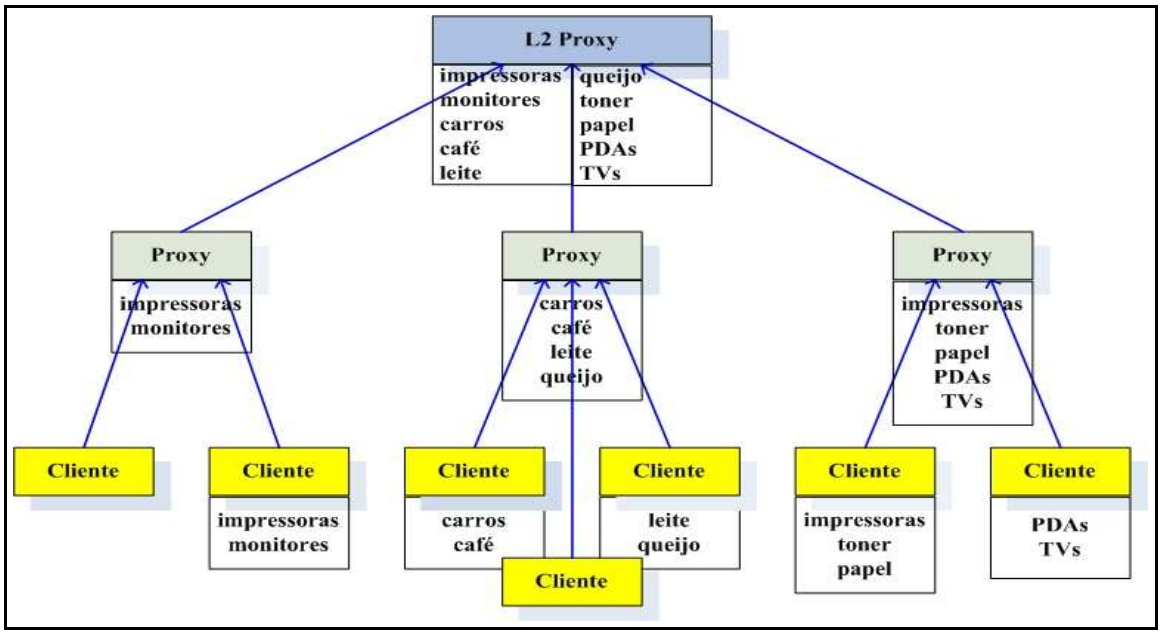

Figura 4.6: Topologia hierárquica de *proxies Carmen* [Marti e Krishnan 2002].

A segurança é garantida através de políticas implementadas nos *proxies*, que são configurados para permitir o anúncio de alguns serviços para fora do domínio, como também das requisições que podem entrar no domínio. Para os clientes e para o serviço de descoberta não são tratadas de questões como autenticação e mecanismo de segurança.

4.4.4 *Splendor*

O *Splendor*, proposto em [Zhu, Mutka e Ni 2003], é um modelo de descoberta de serviço que suporta usuários e serviços nômades em um ambiente público, enfatizando segurança e autenticação. É utilizada a localização como informação importante na descoberta de serviço.

A localização é obtida através de *tags*, ou seja, etiquetas enviadas pelos dispositivos. Outras informações podem ser enviadas, mas a ênfase da proposta é na localização.

O modelo proposto possui quatro componentes: *clientes*, *serviços*, *diretórios* e *proxies*, como ilustra a Figura 4.7.

O protocolo de descoberta de serviço é seguro, pois permite que ambas as partes autentiquem-se e troquem informação através de um canal seguro, utilizando chaves públicas de criptografia e o protocolo de autenticação de duas vias X.509. Os dados são cifrados utilizando técnicas de chave simétrica. Para o gerenciamento de chave e assinatura é utilizada criptografia de chave pública.

Os clientes e os provedores de serviço identificam a mudança de ambiente de três formas: através da mudança do endereço IP; através das mensagens de anúncio dos diretórios, as quais contêm o endereço do diretório; ou através de coordenadas informadas por etiquetas eletrônicas.

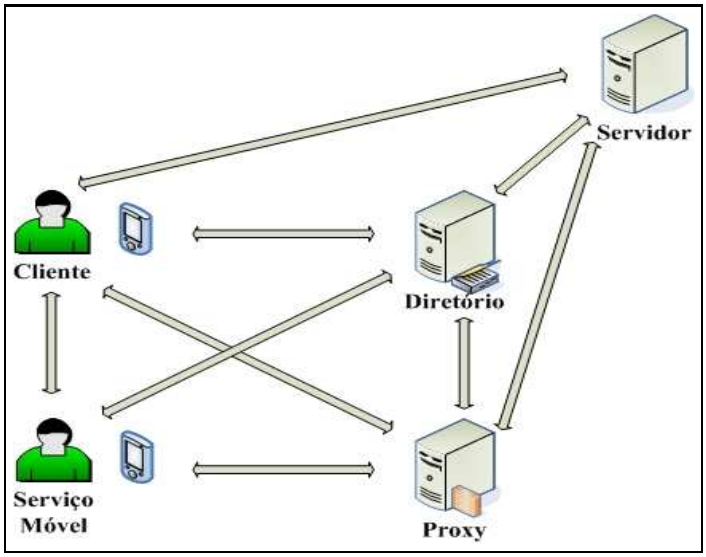

Figura 4.7: Modelo de descoberta de serviço *Splendor* [Zhu, Mutka e Ni 2003].

A comunicação inicial entre clientes, serviços e diretórios é feita através de endereços *multicast*, conhecidos antecipadamente por todas as entidades. Os anúncios e a busca são feitos utilizando comunicação *unicast* entre clientes, provedores de serviços, *proxies* e diretórios, envolvendo apenas as partes interessadas, o que diminui a possibilidade de congestionamento do meio de comunicação.

Para ilustrar o funcionamento do protocolo, foi explorado um exemplo onde há dois médicos de um mesmo hospital em um *shopping center*. Um usuário solicita a descoberta de um serviço de atendimento médico e o protocolo encontra a descrição dos dois médicos. O protocolo, ao receber uma requisição de descoberta, escolhe um dos serviços de atendimento e envia sua descrição ao usuário. A escolha é feita através da agregação, que agrupa os dois serviços de atendimento, e da filtragem, que escolhe um dos dois serviços, para enviar ao usuário.

Os *proxies* são os responsáveis por gerenciar toda negociação para registro, autenticação, autorização e gerenciamento de chaves de criptografia. No exemplo dado, o serviço móvel do médico, o *proxy* negocia com o cliente e informa os serviços móveis de emergência disponíveis. Os serviços móveis possuem seu registro no *proxy* durante o tempo que estiverem disponíveis no ambiente coberto por ele.

4.5 Avaliação

As propostas apresentadas neste capítulo procuram atender às questões mais essenciais de um protocolo de descoberta de serviço, como dinamismo através da descoberta espontânea e configuração automática, seleção com possibilidade de utilização de semântica, baixa interação humana, adaptação automática à disponibilidade móvel e interoperabilidade entre fabricantes e plataformas. Entretanto, se conseguem ser espontâneos, não dão suporte a uma seleção utilizando informação semântica; se têm baixa interação humana, não conseguem suportar adaptação automática; e a interoperabilidade entre fabricantes e plataformas é outro ponto não muito explorado. Além disso, poucos se preocupam com a segurança de dados e autenticação de usuários. Com relação à sensibilidade ao contexto

o máximo que fazem é utilizar a localização para identificar o contexto.

Nesta seção é feita uma avaliação das propostas apresentadas anteriormente, procurando destacar os requisitos principais de um protocolo de descoberta serviço seguro e sensível ao contexto: descrição de serviço, suporte de um serviço de diretório, seleção de serviço, segurança, autenticação e suporte a manipulação de informações sensíveis ao contexto.

Dependendo da configuração adotada, os mecanismos de descoberta que não utilizam um serviço de diretório, como *JINI*, SSDP, SPDP e *Splendor* podem ficar restritos a uma rede local, ou mesmo a um domínio administrativo. Essa situação é típica das redes domésticas, das redes de pequenos escritórios, das redes de sensores, ou mesmo das redes *ad hoc*. Nessa abordagem, os serviços estão descritos nos próprios provedores e as aplicações utilizam um mecanismo de descoberta para verificar a existência de um serviço e acessá-lo. Os serviços são anunciados por difusão. Os provedores de serviço podem ser vistos como uma espécie de diretório e por isso, no ambiente descrito, pode-se dizer que há um *diretório distribuído*, embora não possa ser chamado de um serviço de diretório, pois não há uma classificação, armazenagem das descrições dos serviços, nem métodos bem definidos para gerenciamento e consulta dessas descrições.

A arquitetura *Salutation* se propõe a operar em qualquer ambiente computacional através dos *Transport Managers*, que abstraem os detalhes operacionais para os desenvolvedores de aplicação. Além disso, pode operar com ou sem um serviço de diretório. No caso de não ser implementada com serviço de diretório, as entidades de rede têm condições de se localizarem através de consultas feitas utilizando *broadcast*. No caso de haver um número muito grande de entidades, esse método pode causar um tráfego intenso na rede, podendo degradar todo o ambiente. É utilizada uma comparação nominal de tipos de serviços procurados com os existentes para encontrar serviços. Segurança e autenticação não são tratadas no trabalho.

O SLP utiliza uma descrição simples para os serviços, limitada a uma pesquisa do tipo atributo-valor, por isso não possui um mecanismo de comparação muito eficiente. Pode atuar com ou sem um serviço de diretório. Utiliza *unicast* e *multicast* para descoberta de serviço, o que é muito importante, pois direciona as mensagens apenas para as entidades interessadas. A segurança provida é mínima, pois é apenas garantida a integridade das mensagens e são deixadas de lado outras questões como confidencialidade e não-repúdio. A autenticação das mensagens é feita através da verificação da assinatura digital da entidade. Um aspecto interessante a ser investigado é há possibilidade da utilização de descrições feitas em XML, RDF, ou OWL para enriquecer a descrição dos serviços, já que a resposta do protocolo a uma solicitação de descoberta é uma URL.

JINI é um protocolo dinâmico de descoberta de serviços, pois utiliza uma técnica chamada *leasing* para garantir que todos os serviços registrados na base de dados estejam disponíveis. Quando o tempo limite é alcançado, o protocolo remove as informações do serviço da base, inserindo-as novamente apenas no próximo anúncio. É utilizada uma base de dados de serviços, definidos como *proxies* de objetos, que são carregados nos clientes e utilizados por estes para acessarem os serviços.

O SSDP utiliza HTTP sobre UDP para descobrir serviços. Tem a limitação de prover descoberta apenas para redes locais. Possui a característica importante de prover maior descrição dos serviços através de XML. Além disso, há um campo na mensagem do protocolo onde é possível inserir uma URL para qualquer informação mais detalhada, inclusive em outras linguagens como RDF ou OWL.

SSDS utiliza XML para descrição de serviços. Provê tolerância à falhas, pois caso um servidor SSDS falhe, outros assumem a responsabilidade pelos provedores de serviço. É também um protocolo seguro, confiável e escalável. A segurança é conseguida através da criptografia das informações trocadas entre as entidades.

Dentre as propostas aqui apresentadas as que adotam um serviço de diretório são o *Service Locating Manager*, *Carmen* e *Splendor*. A descrição de serviços é levada em consideração no *Service Locating Manager* e *Carmen*.

A seleção de serviço é feita de forma total ou parcial. Utilizam também parâmetros de QoS, escopo administrativo ou comparação de atributos. Comparação de atributos é utilizada pelo *Service Locating Manager*. Os outros se baseiam em comparações de nomes ou identificadores. Apenas *Splendor* utiliza a localização da entidade como informação contextual para selecionar serviços.

Segurança e autenticação são dois requisitos atendidos por SSDS, SPDP, *Splendor*, e SLP. SSDS utiliza certificados para prover autenticação das entidades. SPDP adota uma abordagem baseada em grau de confiabilidade, enquanto *Splendor* utiliza chaves públicas para troca segura de mensagens e não permite que o cliente tenha acesso às informações do provedor de serviço. SLP utiliza assinaturas digitais para autenticação. *Carmen* procura atender ao requisito segurança limitando o anúncio de serviços à rede local, entretanto em um ambiente local não dá ênfase a qualquer tipo de segurança para os clientes.

A tabela 4.2 resume as características dos protocolos discutidos nesta seção.

	Descrição de Serv.	**Serv. de Diretório**	**Seleção de Serv.**	**Segurança**	**Autenticação**
Salutation	por nome	com ou sem	por nome	não tratada	não provê
SLP	atributo-valor	com ou sem	atributo-valor	provê (mínima)	provê (mínima)
JINI	objetos Java	sem	usuário	limitada à linguagem	não provê
SSDP	XML	sem	baseada na linguagem	não provê	não provê
SSDS	XML	com	baseada na linguagem	provê	provê
SLM	XML	com	atributo-valor	não provê	não provê
SPDP	por nome	sem	não provê	provê	não provê
Carmen	XML	com	não provê	não provê	não provê
Splendor	por nome	sem	baseada no protocolo	provê	provê

Tabela 4.2: Características dos protocolos de descoberta.

4.6 Conclusão

Descoberta de serviço é um processo dinâmico, com baixa interação humana, utiliza informações atuais, garante disponibilidade e provê suporte para descoberta passiva e ativa. Nos processos de descoberta, principalmente os requisitados em sistemas sensíveis ao contexto, algumas questões devem ser levadas em consideração: uma descrição de serviços

mais rica, que torne o processo de descoberta mais eficiente através da utilização de mecanismos de comparação semântica; um suporte de um serviço de diretório para organizar as descrições de serviços segundo alguma classificação, de modo a auxiliar no processo de descoberta através da restrição de consultas, por exemplo, às classes e/ou aos atributos dos serviços; a utilização da informação contextual para selecionar o serviço mais adequado para o usuário; e a utilização de políticas e mecanismos de segurança e autenticação para permitir que clientes e serviços tenham suas informações protegidas contra acesso indesejáveis.

Os protocolos apresentados neste capítulo tratam de maneira muito tímida algumas dessas questões. Alguns, como o SLP, apresentam-se pobres em semântica no que diz respeito à descrição dos serviços. Outros não utilizam qualquer tipo de classificação das descrições, como o *JINI*. A seleção é ainda limitada por linguagens ou comparação nominal, como o *Salutation*. Segurança e autenticação são ainda muito pouco consideras. A utilização de informação contextual é também outro requisito que merece ser melhor tratado, pois apenas uma proposta utiliza localização como informação contextual, que não é suficiente para caracterizar um contexto. São deixados de lado informações como hora, perfil do usuário e de aplicações e dispositivos computacionais presentes no ambiente. Dos mecanismos de descoberta de serviço referenciados na literatura nenhum utiliza informação contextual [Zhu, Mutka e Ni 2002].

Em sistemas sensíveis ao contexto há, portanto, uma carência de um mecanismo de descoberta de serviço que seja dinâmico, seguro e sensível ao contexto, que permita a usuários e aplicações encontrarem serviços que atendam às suas necessidades e que os auxiliem a executar suas atividades.

Capítulo 5

Projeto do Protocolo SCaSDP

5.1 Introdução

Sistemas sensíveis ao contexto permitem que aplicações, tanto de usuário, quanto do sistema, adaptem-se automaticamente às mudanças ocorridas no ambiente onde estão inseridas. A adaptação indica a existência de uma interação entre ambos, possibilitada por serviços que manipulam informações contextuais de usuários, de dispositivos e de outras aplicações. Para serem utilizados, os serviços precisam se tornar conhecidos e uma maneira de fazer isso é através de anúncios. Entretanto, devido a possibilidade de haver um número muito grande de serviços e uma grande variedade de características associadas a cada um deles, descobrir o serviço adequado não é uma tarefa fácil.

Para descoberta de serviços sensíveis ao contexto várias questões devem ser levadas em consideração, como o contexto onde usuários, aplicações e serviços estão inseridos e a natureza da informação contextual. Na literatura há uma carência de propostas de descoberta de serviço que considerem essas questões. Além disso, pouquíssimas abordagens se utilizam de descrições de serviço mais ricas, que possibilitem a utilização de mecanismos de comparação mais complexos, como os que consideram a semântica das descrições.

Neste capítulo é proposto um protocolo de descoberta que contempla as questões acima expostas, considerando as informações contextuais no processo de descoberta e utilizando mecanismos de comparação semântica de descrições mais ricas de serviços. Além disso, permite organizar e classificar as informações de usuários, dispositivos e serviços utilizando um serviço de diretório. Considera também a segurança exigida por determinadas classes de aplicação para garantir a integridade e confidencialidade de suas informações.

5.2 Descrição Geral

O *Secure Context-aware Service Discovery Protocol (SCaSDP)* é um protocolo de descoberta de serviços seguro e sensível ao contexto, responsável pelo registro, anúncio e descoberta de serviços. É uma proposta de descoberta de serviços para o módulo *Gerente de Serviços* da plataforma *Infraware*. Este módulo possui a arquitetura ilustrada pela Figura 5.1.

O SCaSDP implementa o componente *Gerente de Descrições*, cuja arquitetura é ilustrada pela Figura 5.2. Sua arquitetura é composta por quatro componentes que acessam um serviço de diretório:

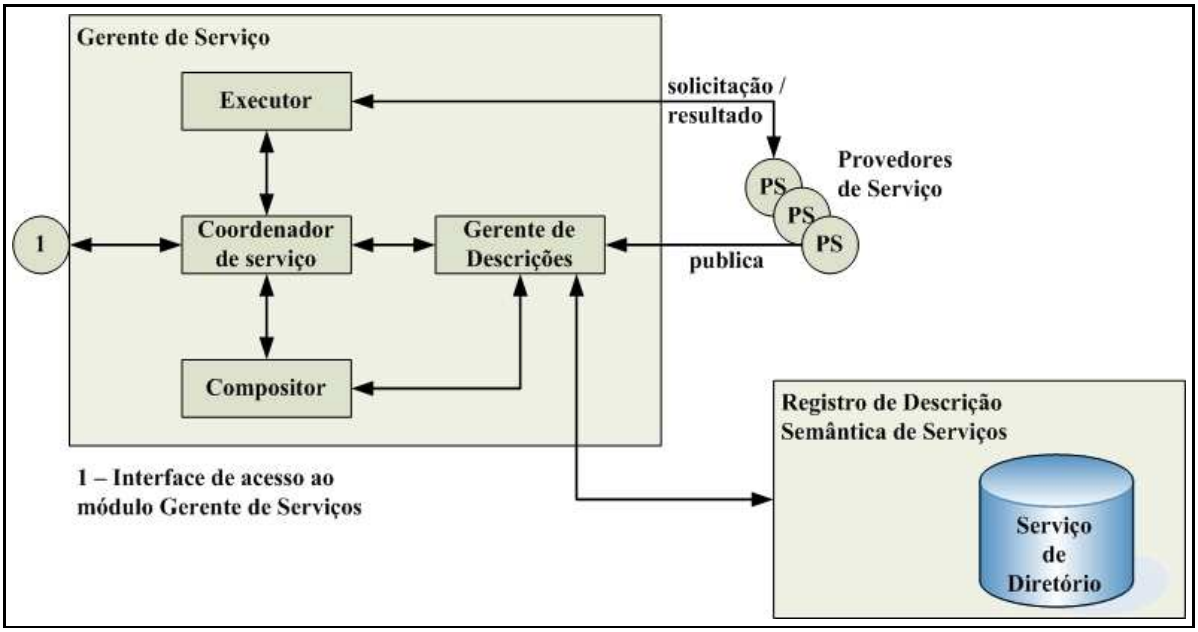

Figura 5.1: Proposta para o Gerente de Serviços da *Infraware*.

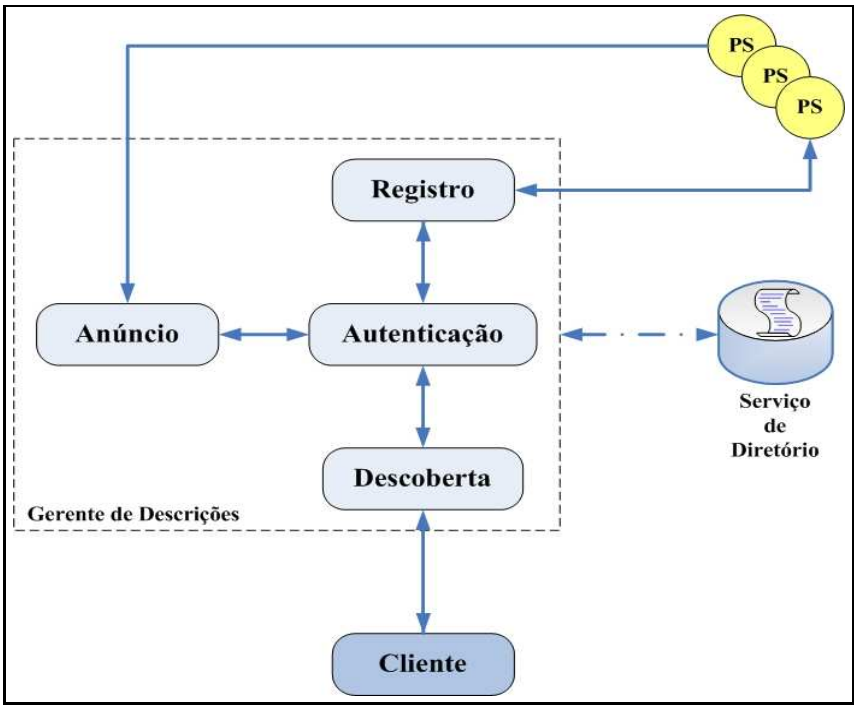

Figura 5.2: Arquitetura do SCaSDP.

1. *Componente de Registro*: responsável por registrar e cancelar o registro de provedores de serviço;

2. *Componente de Anúncio*: responsável por receber os anúncios de provedores de serviço e atualizar os dados armazenados no serviço de diretório;

3. *Componente de Descoberta*: responsável por receber solicitações de descoberta de serviço dos clientes. Este componente utiliza as descrições de serviços armazenadas no serviço de diretório para encontrar o serviço requisitado pelo cliente;

4. *Componente de Autenticação*: responsável por autenticar as solicitações de registro, cancelamento de registro, anúncio e descoberta de serviço. Este componente acessa o *Serviço de Diretório* para obter informações para autenticação de entidades. Apenas após a resposta deste componente os outros podem dar prosseguimento às suas tarefas; e

5. *Serviço de Diretório*: responsável por gerenciar e organizar as descrições de serviço e as informações de autenticação dos provedores de serviço e clientes. O serviço de diretório é acessado pelos componentes de registro, anúncio e descoberta para realizarem suas tarefas.

O protocolo se propõe a atender aos requisitos discutidos no capítulo 4, ou seja, provê um mecanismo de descoberta que utiliza seleção com base na descrição semântica do serviço; utiliza um serviço de diretório para organizar e gerenciar informações de clientes e provedores de serviço; e conta com mecanismos de segurança para garantir a confidencialidade, integridade, não-repúdio de dados e autenticação de entidades.

As subseções seguintes apresentam uma descrição informal do funcionamento do protocolo. A especificação formal do SCaSDP, utilizando Redes de Petri e Máquinas de Estados Finita, é apresentada no capítulo 6.

5.2.1 Funcionamento do SCaSDP

O SCaSDP permite que provedores de serviço se anunciem e que clientes descubram serviços através de uma entidade central, denominada *proxy*. Como no SLP, vários *proxies* podem se comunicar para propagar as informações e fazer consultas sobre serviços, o que conseqüentemente aumenta as chances de um cliente encontrar o serviço desejado, mesmo que este esteja fora do domínio do *proxy* ao qual é feita a solicitação de descoberta. Os clientes, para descobrirem serviços, contatam o *proxy* e enviam informações para ele encontrar o serviço desejado.

Os provedores de serviço, antes de se anunciarem a um *proxy*, precisam se registrar nele e para isso, utilizam descoberta passiva para determinar o endereço do *proxy*. O provedor do *Serviço 06* da Figura 5.3, por exemplo, descobre o *proxy* da rede e em seguida solicita o registro enviando alguns dados como sua localização, conta e senha de acesso, sua chave pública de criptografia para uso posterior pelo *Proxy C*, o intervalo entre anúncios e uma descrição do serviço provido, que é feita com base em uma ontologia de serviços e contém informações contextuais do provedor e do serviço. O *Proxy C* recebe esses dados, processa-os e responde ao provedor com uma autorização de registro, que inclui sua chave pública de criptografia e uma identificação, que o provedor deve utilizar para fazer seus anúncios periódicos. O provedor pode cancelar seu registro enviando uma solicitação ao *proxy*.

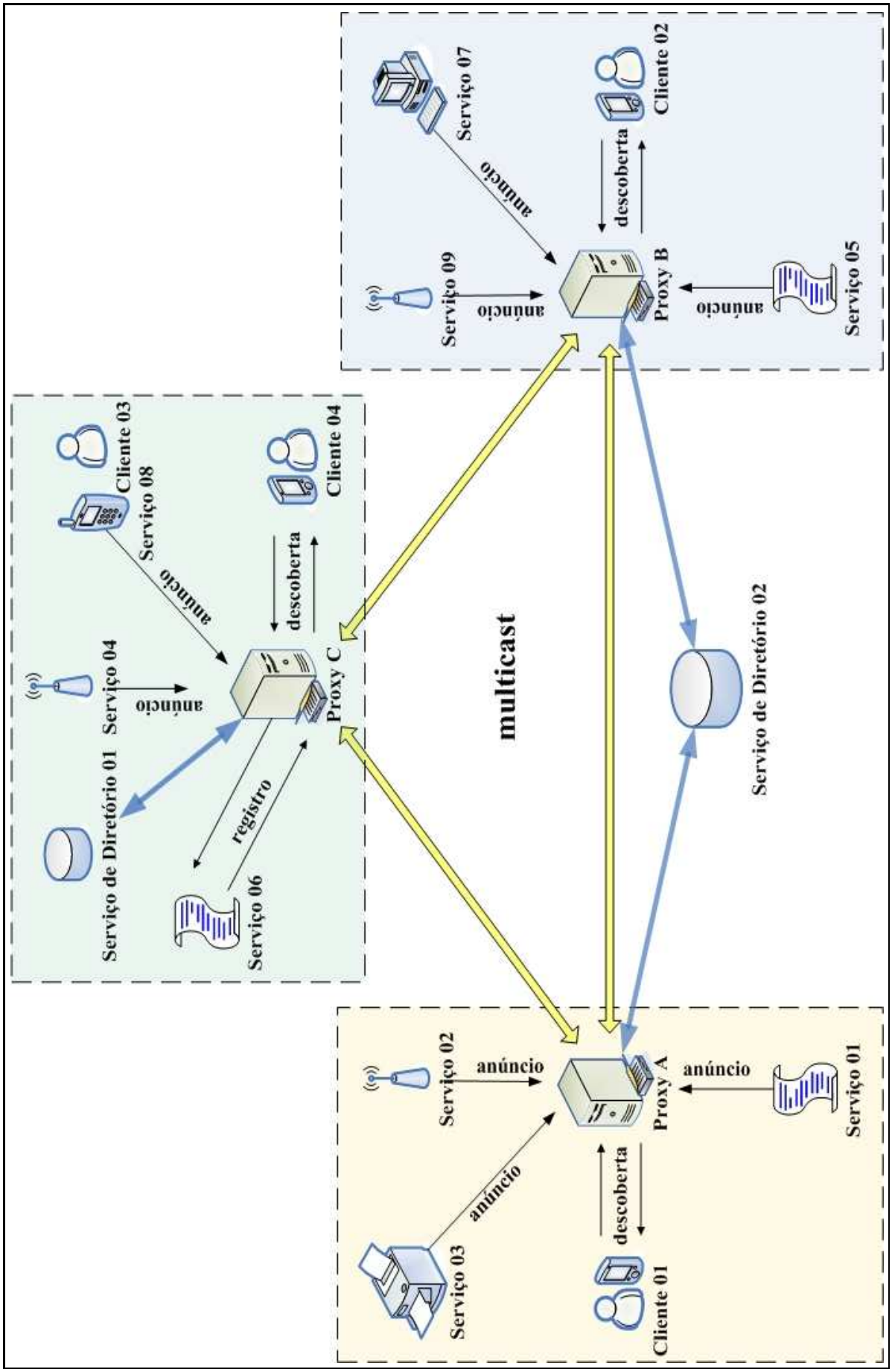

Figura 5.3: Funcionamento do SCaSDP.

Com isso a identificação do provedor é removida, mas suas informações são mantidas com o objetivo de gerar dados estatísticos.

Os anúncios de serviços devem ser feitos periodicamente ao *proxy*, obedecendo a intervalos de tempo regulares, para garantir a disponibilidade do serviço. Os anúncios são feitos utilizando os protocolos da camada de transporte TCP ou UDP. A preferência por um dos dois é determinada pelo tamanho da mensagem, ou seja, se uma mensagem de anúncio possuir tamanho superior a 1400 *bytes* o TCP é escolhido, caso contrário é utilizado o UDP. O tamanho de 1400 *bytes* é determinado tomando como base a MTU, que tem tamanho de 1500 *bytes* para o *Ethernet*. Isto permite aumentar a possibilidade dos anúncios serem feitos via UDP, o que elimina o tempo despendido pelo TCP para o estabelecimento de conexão. Os 100 primeiros *bytes* do pacote do protocolo são ocupados pelos campos de controle da camada de enlace, da camada de rede e do protocolo da camada de transporte.

O *proxy* tem abrangência local, ou seja, cada rede possui um único *proxy*. É responsável por se anunciar e por receber os anúncios dos provedores e as solicitações de descoberta dos clientes. Quando recebe uma mensagem, seja de registro, cancelamento de registro, anúncio de provedor ou solicitação de descoberta feita por um cliente, o *proxy* antes de mais nada autentica o solicitante. A autenticação é realizada através de verificação de conta, senha e chave pública no caso de provedores. Para clientes apenas conta e senha são utilizadas. O passo seguinte é processar a requisição. O provedor do *Serviço 07* da Figura 5.3, por exemplo, envia um anúncio ao *Proxy B*. Este autentica o provedor e em seguida verifica se a mensagem é válida, ou seja, se foi enviada dentro do intervalo de tempo entre anúncios definido pelo provedor. Se for válida, o anúncio do provedor é aceito e as informações são armazenadas no serviço de diretório, ficando disponíveis durante um período de tempo especificado pelo provedor. Uma solicitação de descoberta feita por um cliente obedece ao mesmo procedimento, exceto o último passo, que é substituído pelo processo de descoberta.

A descoberta de serviços feita pelo *proxy* utiliza um algoritmo de comparação semântica de descrições de serviço obtidas do serviço de diretório e/ou de *proxies* de outras redes. Um serviço de diretório pode ser utilizado por mais de um *proxy*, conforme é ilustrado pela figura. O *proxy* descobre serviços da seguinte forma: o *Proxy A* recebe de seu cliente uma requisição de descoberta de um serviço de monitoração de temperatura. São executados os procedimentos para descoberta, que envolvem consultas a outros *proxies*. O *Proxy A* encontra o *Serviço 02* e o *Proxy C* o *Serviço 04*, que também atende ao cliente. O protocolo escolhe qual dos dois serviços melhor atendo ao cliente com base nas informações contextuais do cliente e dos serviços. Por exemplo, se o cliente solicitou um serviço de medição de temperatura da sala onde se encontra, o protocolo utiliza as informações de localização do cliente para escolher o serviço adequando, o que neste caso é o *Serviço 02*.

Se vários serviços de temperatura próximos ao cliente são encontrados, o protocolo envia a descrição de todos eles e, então, cabe ao cliente escolher qual o que melhor atende às suas necessidades.

Entre os *proxies* não há qualquer hierarquia que identifique o principal, ou seja, se um *proxy* solicita descoberta para outro, este é visto como servidor e o primeiro como cliente. A autenticação entre *proxies* é realizada utilizando conta, senha e a chave pública do *proxy* solicitante.

Um dispositivo pode atuar tanto como provedor de serviço, quanto como cliente. Por exemplo, o *Serviço 08* da Figura 5.3 é e um provedor porque anuncia sua localização,

capacidade de armazenamento, qualidade do canal de comunicação e outras informações relevantes. É um cliente porque utiliza o protocolo para solicitar descoberta de serviços ao *proxy*.

5.2.2 Multicast, Broadcast e Portas TCP e UDP

Como visto anteriormente, os provedores de serviço utilizam os protocolos de transporte TCP ou UDP para o anúncio das descrições de serviço. Os clientes utilizam TCP para se comunicarem com o *proxy* e solicitarem descoberta de serviços. A porta alocada nos dois protocolos da camada de transporte para o *proxy* receber as requisições de anúncio e descoberta é a de número *30000*. Os provedores de serviço utilizam a porta de número *30001* do UDP para determinarem o *proxy* da rede, através de descoberta passiva.

Os *proxies* se comunicam entre si através de *IP multicast*, que tem a vantagem de permitir que apenas os dispositivos que possuem esse tipo de IP recebam as mensagens. O anúncio na rede é feito sobre UDP por *broadcast*, ou por *multicast*. A escolha por um ou por outro é feita no momento da instalação e configuração do SCaSDP.

5.2.3 Descrição de Serviço

As descrições de serviços devem ser feitas utilizando a ontologia de serviço OWL-S, que permite ao SCaSDP utilizar mecanismos de comparação semântica, e a ontologia de dispositivos FIPA (*Foundation for Intelligent Physical Agents* [Agents 2002], que define classes e propriedades interpretadas pelo protocolo como informações contextuais. Essas ontologias permitem descrever serviços que possam ser analisados pelo SCaSDP durante o processo de descoberta do serviço mais adequado às necessidades dos clientes. Por exemplo, se o cliente necessita utilizar um serviço que garanta *QoS* e que tenha poucos processos em execução, a comparação deve levar em conta essas restrições.

A descrição do serviço deve conter as seguintes informações:

- *Identificação*: identificação do serviço gerada pelo *proxy*;
- *Localização*: localização física do provedor do serviço composta das coordenadas geográficas *latitude* e *longitude*. Estas são informações contextuais e são importantes para que o protocolo, com base nelas, possa selecionar o serviço mais próximo do cliente;
- *Tipo de serviço*: tipo de serviço solicitado pelo cliente;
- *Saída*: informação sobre o que o serviço fornece;
- *Entrada*: informação necessárias para que o serviço seja oferecido. Esta propriedade é classificada em:
  - *Obrigatória*: necessária para que a descoberta de serviço seja feita;
  - *Opcional*: quando fornecidas refinam a descoberta de serviço, melhorando a qualidade do resultado obtido.
- *URL*: URL onde o serviço pode ser acessado;

- *Atributos*: facilidades que o usuário deseja que o serviço possua. Dependendo da facilidade, o usuário deve fornecer algumas informações contextuais, por exemplo localização, perfil de *hardware* ou *software*, etc. Esta propriedade é utilizada para enriquecer a descrição e melhorar a precisão da descoberta de serviço.

Um exemplo de descrição de serviço pode ser:

- Identificação: identificação do serviço.
  - Id: 1923834848483 (gerada pelo *proxy*)
  - Nome: jurema
  - Fabricante: CUPs (fabricante/desenvolvedor)
- Tipo: impressão
- Saída: (documento impresso)
- Entrada: (arquivo a ser impresso)
- Atributos: atributos do serviço e/ou do provedor de serviço.
  - Localização: localização física do dispositivo.
    - Latitude: -45.4354
    - Longitude: -30.3456
  - Conexão: conexão de rede do dispositivo (informação contextual).
    - Tipo: *Ethernet*
    - Banda: 10000
    - IP: 10.0.0.1;
    - *Netmask*: 255.255.255.0
    - *URL*: http://jurema:461/printers
  - Memória: descrição de memória do serviço (informação contextual).
    - Tipo: RAM
    - Capacidade: 128
    - Unidade: MB
    - Utilização: 96
  - CPU (utilização): taxa de utilização da CPU (informação contextual).
    - Máxima: 0.28
    - Média: 0.06
    - Mínima: 0.02
  - Trabalhos na fila de impressão: 10 (informação contextual).

Outros atributos podem ser acrescentados de acordo com as características de cada serviço.

5.2.4 Seleção de Serviço

Para encontrar um serviço, um cliente envia ao *proxy* a descrição do serviço desejado. O *proxy* por sua vez, consulta o serviço de diretório e seleciona os serviços que atendem ao que o cliente deseja. A seleção das descrições está vinculada à situação do serviço, ou seja, apenas é selecionada a descrição cujo anúncio é atual. Isto é verificado através do *leasing*, que é calculado utilizando a hora de anúncio do serviço ($hora_{envio}$), a hora que o anúncio foi recebido pelo *proxy* ($hora_{recebimento}$) e o intervalo entre anúncios (*pooling*) informado pelo provedor, conforme é mostrado pela inequação abaixo:

$$(hora_{recebimento} - pooling) \leq hora_{envio} \leq hora_{recebimento}$$

Se a inequação acima for falsa, significa que as informações estão desatualizadas e conseqüentemente que o serviço pode estar inativo. O *leasing* caracteriza o dinamismo da descoberta de serviço, pois mesmo que possa atender ao que o cliente deseja, o serviço não é selecionado pelo algoritmo. Desta forma, o dinamismo é indiretamente conseguido, garantindo que os serviços selecionados estejam ativos na rede.

A comparação de descrições é feita utilizando as seguintes informações:

- *Requisição de serviço*: uma descrição semântica do serviço desejado pelo cliente;
- *Descrições de serviço*: um conjunto de descrições semânticas de serviços disponíveis, obtidas de um serviço de diretório;
- *Ontologias de serviços*: utilizadas para obter uma compreensão comum sobre os serviços e permitir a realização de inferências semânticas sobre as descrições; e
- *Informação contextual*: obtida de provedores de informação contextual quando o cliente não é capaz de fornecer explicitamente informações necessárias para descobrir o serviço desejado.

5.2.5 Algoritmo de Comparação Semântica

A comparação entre as descrições de serviços é executada por um algoritmo desenvolvido por [Costa 2005]. Este algoritmo se baseia nos graus de similaridade (vide seção 3.4) para classificar a comparação.

O algoritmo compara as descrições segundo os parâmetros seguintes, nessa ordem:

1. Tipo de serviço e *leasing*: o tipo de serviço indica a classificação do serviço segundo uma taxonomia. O *leasing* é utilizado para selecionar os serviços ativos no ambiente;
2. Saída: o resultado do serviço solicitado pelo usuário;
3. Entrada: o que o serviço precisa para realizar seu processamento;
4. Atributos: requisitos do serviço, desejados pelo usuário. Esses atributos caracterizam as informações contextuais.

Os dois primeiros itens são fundamentais, ou seja, o serviço que não for do tipo que o cliente solicitou e que não possuir algumas das saídas definidas, não é selecionado. A

entrada garante uma comparação exata, ou seja, se for fornecida pelo cliente ou se o algoritmo a obtiver de provedores de informação contextual a comparação é considerada exata. Caso contrário, é aproximada. É feita uma ordenação decrescente dos serviços selecionados utilizando o número de atributos, o que tiver maior número de atributos compatíveis com a descrição do serviço desejado é o primeiro da lista.

O algoritmo de comparação possui quatro etapas, conforme é ilustrado pela Figura 5.4.

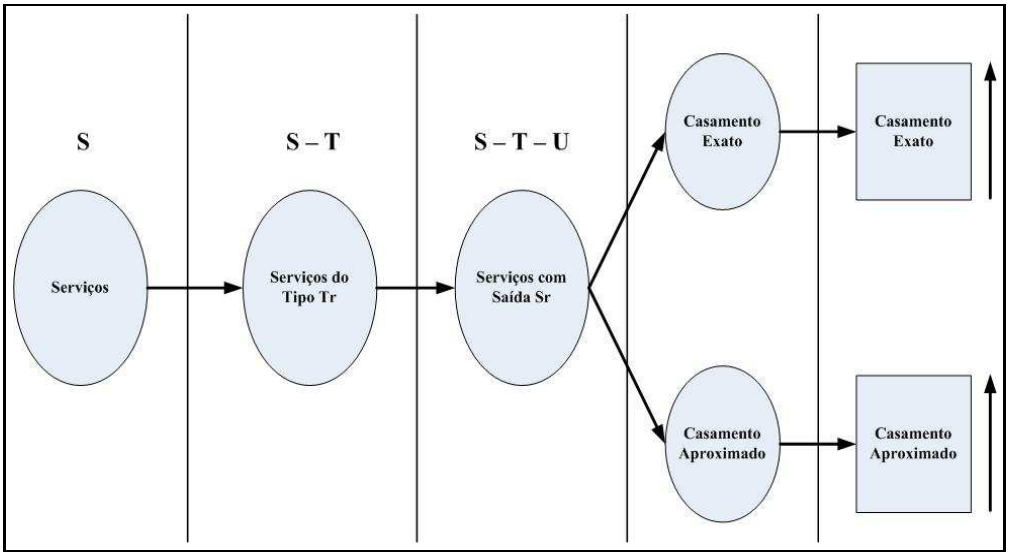

Figura 5.4: Etapas do algoritmo de comparação.

O processo compara uma requisição de serviço R fornecida pelo cliente, que contém o tipo de serviço T_r requisitado, as saídas S_r desejadas, as entradas E_r providas, e os atributos A_r com um conjunto de descrições de serviço S, obtidos do serviço de diretório. Cada descrição s ($s \in S$) possui o tipo de serviço T_s, a saída provida S_s, as entradas E_s que precisam e os atributos fornecidos A_s. Durante a primeira etapa do processo de comparação são selecionadas descrições de serviço do conjunto S que são do tipo T_r solicitado e cujo *leasing* não tenha expirado. Isso gera um conjunto menor de descrições $S - T$. Na segunda etapa, as descrições que oferecem a saída S_r são selecionadas, o que gera um outro conjunto $S - T - U$ menor que o anterior. Na terceira etapa, as descrições deste novo conjunto são examinadas para verificar se as entradas E_s são fornecidas pelo cliente ou se podem ser obtidas de algum provedor de informação contextual. Se isto for verificado, a comparação é exata e a descrição é inserida no conjunto de descrições de serviços que atendem totalmente ao que é requerido. Caso contrário, a descrição é inserida no conjunto de descrições de serviços que aproximadamente atendem ao que é requerido. Na última etapa os dois conjuntos são ordenados e enviados ao cliente.

5.2.6 Serviço de Diretório

Conforme definido em 3.5, um serviço de diretório permite que administradores gerenciem o acesso aos recursos de uma rede através de um *software* ou um conjunto de aplicações. O SCaSDP utiliza um serviço de diretório para armazenar as descrições de serviço, informações sobre os usuários, contas e senhas de acesso.

Toda comunicação dos provedores de serviço e clientes com o *proxy* é antes de mais nada autenticada. Isto é feito pelo componente de autenticação do *Gerente de Descrições*.

Este componente acessa o serviço de diretório para obter informações (senha e conta de acesso, chave pública de criptografia) sobre os provedores e os clientes para verificar se a comunicação pode ser estabelecida. O componente *Registro* acessa o serviço de diretório para registrar e/ou atualizar as informações e descrição de serviço de provedores. O componente *Anúncio*, da mesma forma, atualiza as informações e descrição de provedores. O componente *Descoberta* consulta o serviço de diretório em busca de descrições de serviço que atendam às necessidades dos clientes.

No SCaSDP os *proxies* não são organizados segundo uma hierarquia, o que permite uma independência para que cada um utilize uma classificação diferente para organizar as informações. Além disso, podem ser utilizadas políticas de acesso diferentes e adaptadas à realidade do domínio. Por exemplo, no domínio de uma empresa são adotadas regras de acesso mais robustas do que no domínio de um ambiente doméstico. Dois domínios com características semelhantes podem utilizar o mesmo servidor de diretório. Um exemplo disso é observado na Figura 5.3. Essa situação é válida para, por exemplo, uma empresa composta de escritórios espalhados por diversos locais.

O serviço de diretório suporta descrições semânticas de serviço e provê funcionalidades para pesquisas avançadas, que se utilizam da semântica das descrições para melhor atender às solicitações de descoberta feitas pelas aplicações.

5.2.7 Segurança e Autenticação

O SCaSDP provê segurança através de mecanismos de criptografia, autenticação e verificação de mensagens antigas. O *proxy* e os provedores de serviço trocam chaves de criptografia, geradas utilizando um método de criptografia de chave pública (*Public Key Cryptography (PKC)*) durante a fase de registro. Os provedores recebem a chave pública do *proxy* e este as chaves públicas dos provedores. Os provedores utilizam suas chaves privadas para cifrar suas mensagens de anúncio, e o *proxy* utiliza as chaves públicas dos provedores para decifrar as mensagens de anúncio deles.

As mensagens de anúncio dos provedores, antes de serem cifradas, são compactadas com o objetivo de aumentar a criptografia e as chances de serem enviadas em um único pacote UDP. Esta abordagem permite ao SCaSDP ser um pouco mais rápido, eliminando o tempo utilizado pelo TCP no estabelecimento de conexão. Entretanto, se isso não for possível, o anúncio é feito através de um canal seguro sobre o TCP.

Além de garantir a integridade dos dados, a criptografia das mensagens permite verificar se o provedor que envia o anúncio é de fato quem se diz ser, ou seja, é possível garantir o não-repúdio do anúncio. O *proxy* verifica isto utilizando o IP do provedor para obter a chave pública correspondente e decifrar a mensagem de anúncio. Se não conseguir decifrar, a mensagem é silenciosamente descartada.

Todas as mensagens de anúncio recebidas pelo *proxy* têm seu tempo de validade verificado através do *leasing*, sendo deste modo garantida a não aceitação de anúncios antigos.

Antes de todo e qualquer processamento de registro, cancelamento de registro, descoberta e anúncio de serviço o *proxy* autentica as entidades solicitantes de tais serviços. Para isso, são utilizados métodos de autenticação baseados em conta e senha, informações como endereço MAC, comparação de chaves públicas, e outros mecanismos providos por serviços de diretório.

5.3 Primitivas de serviço

O SCaSDP provê quatro primitivas de serviço. São elas:

- **Register**: utilizada para o registro de um provedor de serviço no *proxy*. Esta primitiva requer os seguintes parâmetros:
  - *IP do proxy*: número IP do *proxy*;
  - *Conta*: conta administrativa para registro de serviço;
  - *Senha*: senha da conta administrativa;
  - *Senha do provedor*: senha do provedor a ser cadastrada;
  - *Chave pública de criptografia*: chave pública de criptografia do provedor de serviço; e
  - *Descrição do serviço*: descrição semântica do serviço.

- **Deregister**: utilizada para cancelar o registro de um provedor de serviço no *proxy*. Esta primitiva requer os seguintes parâmetros:
  - *IP do proxy*: número IP do *proxy*;
  - *Senha do provedor*: senha do provedor a ser cadastrada; e
  - *Identificação do provedor*: identificação gerada pelo *proxy*

- **Advert**: utilizada por provedores de serviço e *proxy* para se anunciarem. Esta primitiva requer os seguintes parâmetros para o provedor:
  - *IP do proxy*: número IP do *proxy*;
  - *Senha do provedor*: senha do provedor a ser cadastrada;
  - *Identificação do provedor*: identificação gerada pelo *proxy*; e
  - *Descrição de serviço*: descrição semântica do serviço.

  Para o *proxy* é exigido o seguinte parâmetro:
  - *IP broadcast*: número IP para *broadcast*.

- **Discover**: utilizada por um cliente para solicitar descoberta de serviço ao *proxy* e obter descrições do serviço desejado. Esta primitiva requer os seguintes parâmetros:
  - *IP do proxy*: número IP do *proxy*;
  - *Conta de acesso*: conta de acesso do cliente;
  - *Senha de acesso*: senha de acesso do cliente; e
  - *Descrição do serviço*: descrição semântica do serviço desejado.

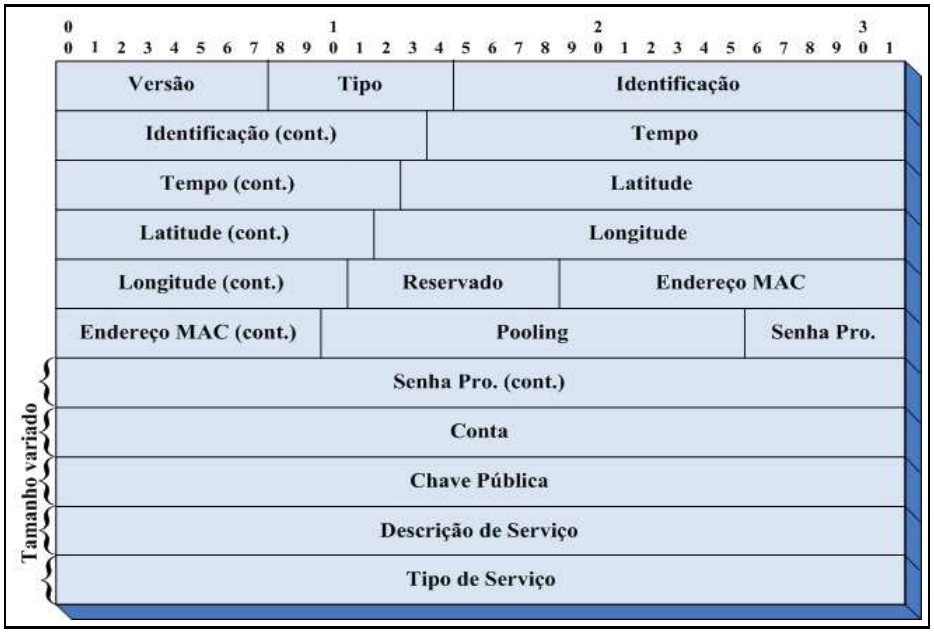

Figura 5.5: Cabeçalho de mensagens SCaSDP.

5.4 Formato das Mensagens do SCaSDP

Todas as mensagens do SCaSDP são encapsuladas na parte de dados do TCP ou UDP e são do formato mostrado na Figura 5.5.

Os campos da mensagem são listados abaixo:

- **Versão**: versão do protocolo;
- **Tipo**: o tipo da mensagem enviada;
- **Identificação**: identificação do provedor de serviço. Só será preenchido se a mensagem for enviada por um provedor de serviço;
- **Tempo**: tempo em segundos. É utilizado para verificar a validade da mensagem;
- **Latitude**: latitude. Coordenada geográfica utilizada para localizar a entidade em conjunto com o campo *Longitude*;
- **Longitude**: longitude. Coordenada geográfica utilizada para localizar a entidade em conjunto com o campo *Latitude*;
- **Reservado**: campo reservado para uso futuro;
- **Endereço MAC**: endereço MAC da entidade. Utilizado no processo de autenticação da entidade;
- **Pooling**: intervalo entre anúncios. É utilizado pelo provedor de serviço e pelo *proxy* para informar o intervalo entre os anúncios dessas entidades; e
- **Senha Pro.**: senha do provedor de serviço cadastrada durante o registro do provedor;

- **Conta**: conta e senha de acesso do administrador do provedor, ou do usuário do cliente durante o processo de registro e cancelamento de registro do provedor ou de solicitação de descoberta do cliente;

- **Chave pública**: chave pública de criptografia do provedor ou do *proxy*. É utilizado durante o processo de registro do provedor;

- **Descrição**: descrição semântica de serviço. É utilizado durante o registro e anúncio do provedor de serviço. O cliente utiliza este campo para enviar ao *proxy* a descrição do serviço desejado;

- **Tipo de Serviço**: tipo do serviço provido pelo provedor de serviço. É utilizado no registro e descoberta de serviço.

As mensagens possuem a abreviação e identificação mostradas na tabela 5.1.

Tipo de mensagem	**Abreviação**	**Id.**
Anúncio de *proxy*	PrAdvert	0
Registro de serviço	SrvReg	1
Resposta de registro de serviço	SrvRegRply	2
Resposta negativa de registro de serviço	SrvRegRplyNeg	3
Requisição de descoberta	DiscReq	4
Resposta de requisição de descoberta	DiscReqRply	5
Resposta negativa de requisição de descoberta	DiscReqRplyNeg	6
Cancelamento de registro de serviço	SrvDeReg	7
Resposta de cancelamento de registro de serviço	SrvDeRegRply	8
Resposta negativa de cancelamento de registro de serviço	SrvDeRegRplyNeg	9
Anúncio de serviço	SrvAdvert	10

Tabela 5.1: Tipos de mensagens do protocolo SCaSDP.

Provedores de serviço devem suportar *SrvReg*, *SrvRegRply*, *SrvRegRplyNeg*, *SrvDeReg*, *SrvDeRegRply*, *SrvDeRegRplyNeg*, *PrAdvert* e *SrvAdvert*. Clientes devem suportar *DiscReq*, *DiscReqRply* e *DiscReqRplyNeg*.

Os campos do cabeçalho utilizados por todas as mensagens são, *Versão*, *Tipo*, *Identificação*, *Tempo*, *Latitude* e *Longitude*. O campo *Identificação* possui valor diferente de 0 para as mensagens do tipo *SrvRegRply*, *SrvDeReg* e *SrvAdvert*. O campo *Endereço MAC* é preenchido nas mensagens do tipo *SrvReg*, *SrvDeReg* e *SrvAdvert*.

5.4.1 Anúncio de *Proxy*

O *proxy* se anuncia utilizando a primitiva *Advert* com mensagens do tipo *PrAdvert* aos provedores de serviço a cada *Pooling* segundos. Tais mensagens têm o campo *Tipo* preenchido com valor 0. As Figuras 5.6 e 5.7 ilustram a mensagem e o anúncio do *proxy*, respectivamente.

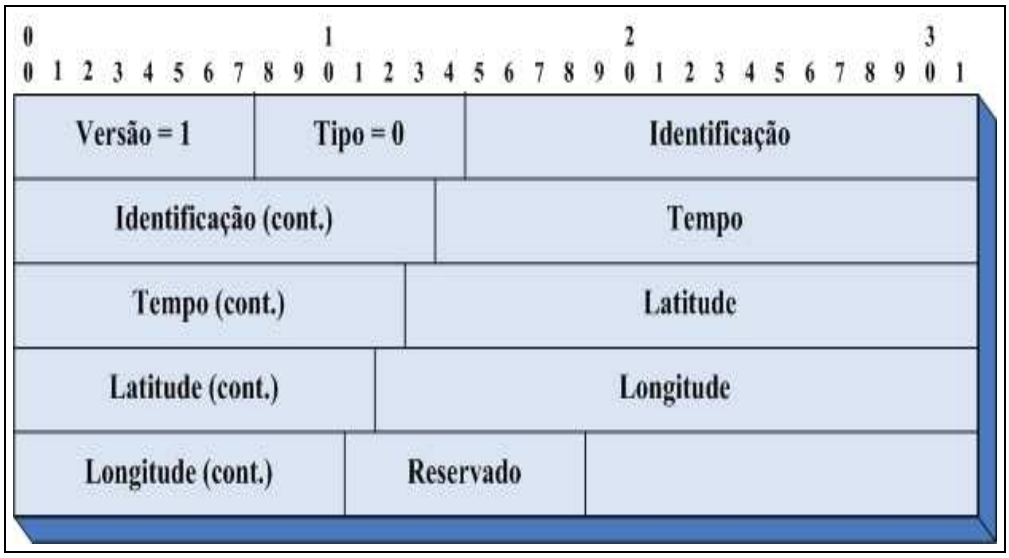

Figura 5.6: Mensagem de anúncio do *proxy*.

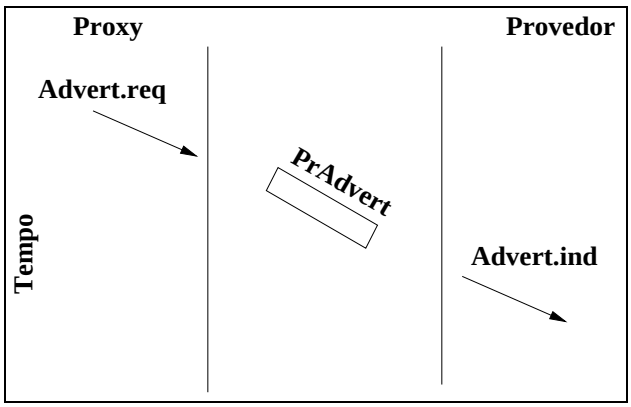

Figura 5.7: Anúncio do *proxy*.

5.4.2 Registro de Serviço

Os provedores de serviço utilizam a primitiva *Register* para se registrarem no *proxy*. A mensagem utilizada é *SrvReg* com os campos *Tipo* com valor 1, *Endereço MAC* com o endereço MAC do provedor, *Pooling* com o intervalo de tempo entre anúncios, *Senha Pro.* com a senha do provedor a ser cadastrada, *Conta* com a conta de administrador, *Chave Pública* com a chave pública do provedor, *Descrição de Serviço* com a descrição semântica do serviço e *Tipo de Serviço* com o tipo de serviço provido. As Figuras 5.8 e 5.9 ilustram a mensagem e o registro de serviço.

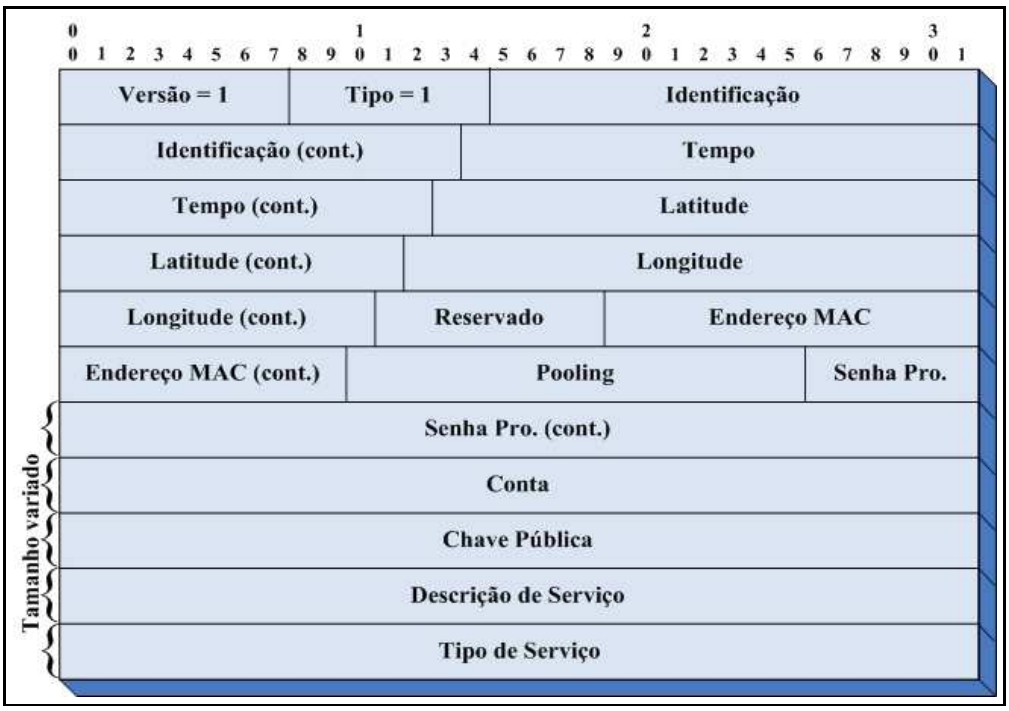

Figura 5.8: Mensagem de registro de serviço.

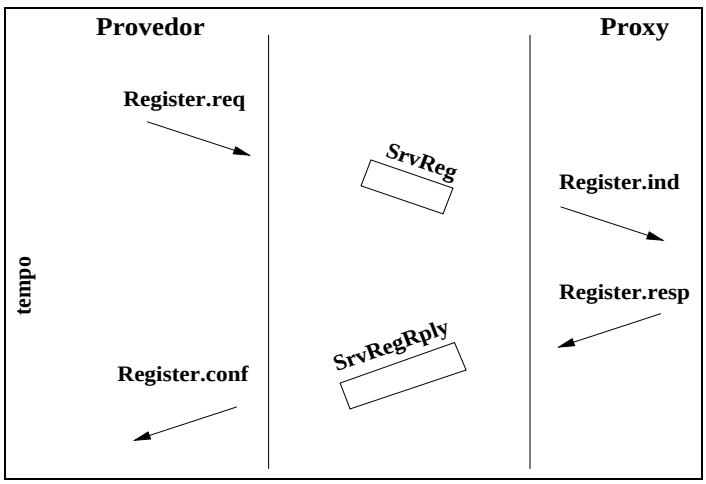

Figura 5.9: Registro de serviço.

Para realizar o registro o provedor envia uma mensagem do tipo *SrvReg*. Se for registrado, o provedor recebe uma mensagem do tipo *SrvRegRply*, que inclui a chave pública

do *proxy* e uma identificação que deve utilizar para se anunciar.

Se o registro não for realizado, o provedor recebe uma mensagem do tipo *SrvRegRply-Neg* que indica erro no processo de autenticação. Se não receber resposta por um período de tempo superior a *TIMEOUT*, o registro é cancelado.

5.4.3 Resposta de Registro de Serviço

Uma vez tendo solicitado registro, o provedor espera por uma resposta do *proxy*, que processa a autenticação e registra no serviço de diretório as informações do provedor de serviço. Em seguida envia uma mensagem do tipo *SrvRegRply*, com os campo *Tipo* com valor 2, *Identificação* com uma identificação gerada e *Chave pública* com sua chave pública de criptografia. A Figura 5.10 ilustra a mensagem.

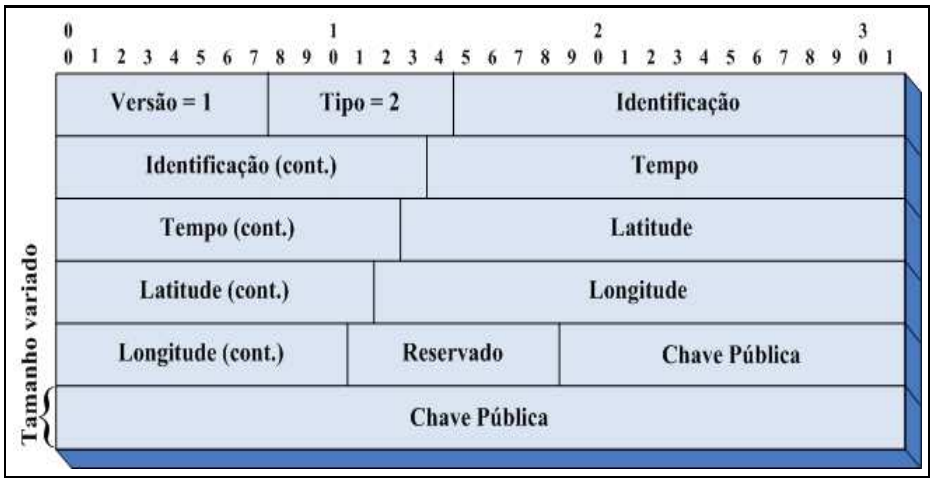

Figura 5.10: Resposta de registro de serviço.

5.4.4 Resposta Negativa de Registro de Serviço

Caso o registro do provedor de serviço não seja efetuado, o *proxy* envia uma mensagem do tipo *SrvRegRplyNeg* ao provedor com o campo *Tipo* com valor 3. A Figura 5.11 ilustra a mensagem.

5.4.5 Requisição de Descoberta

O cliente, para descobrir um serviço, utiliza a primitiva *Discover*. A mensagem utilizada é do tipo *DiscReq*, que contém conta, senha e a descrição serviço desejado. Os campos *Tipo*, *Conta*, *Descrição de Serviço* e *Tipo de Serviço* são preenchidos com valor 4, conta e senha, a descrição e tipo do serviço desejado, respectivamente. As Figuras 5.12 e 5.13 ilustram a mensagem e a requisição de descoberta de serviço.

Após isso, espera uma resposta do *proxy*, que poderá ser descrições de serviços encontrados ou a negação de autenticação. Se não receber resposta por um período de tempo superior a *TIMEOUT*, a requisição de descoberta é cancelada.

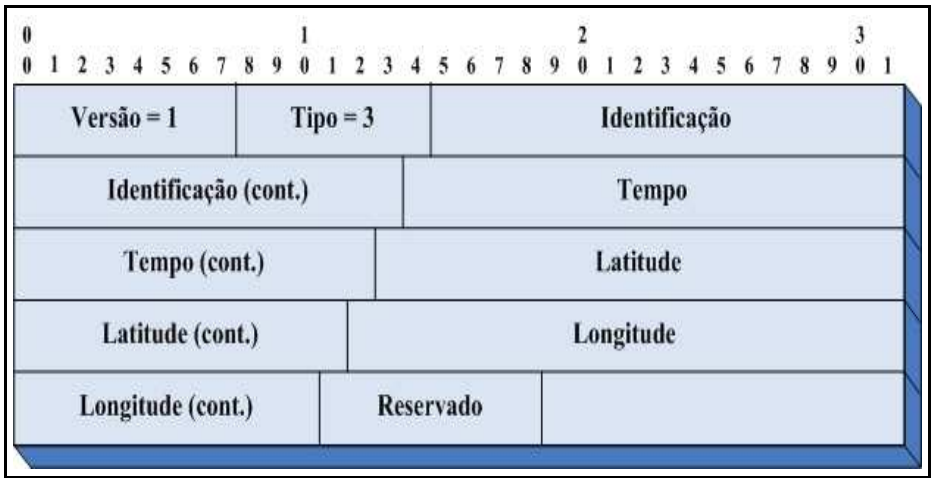

Figura 5.11: Resposta negativa de registro de serviço.

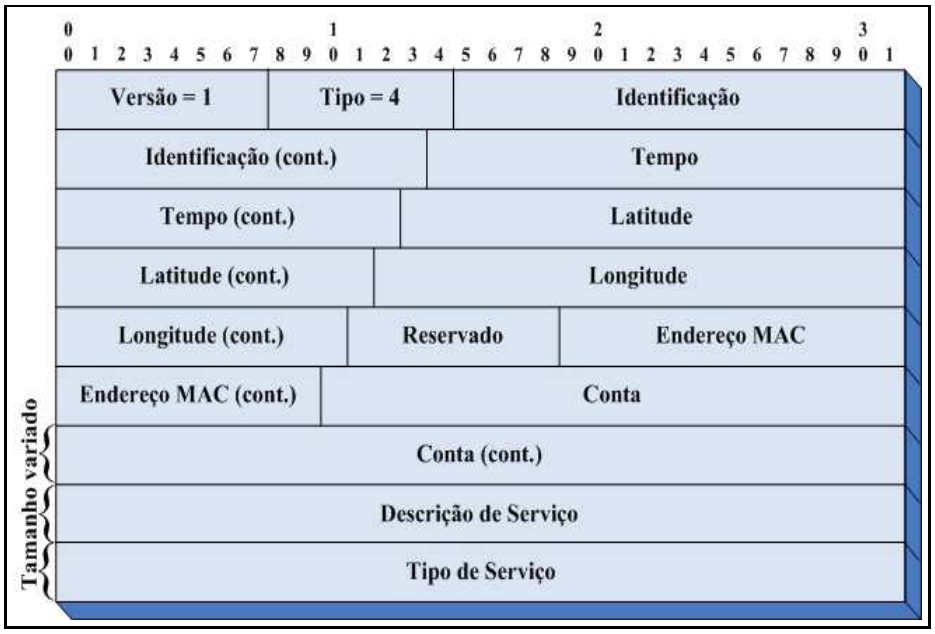

Figura 5.12: Mensagem de requisição de descoberta.

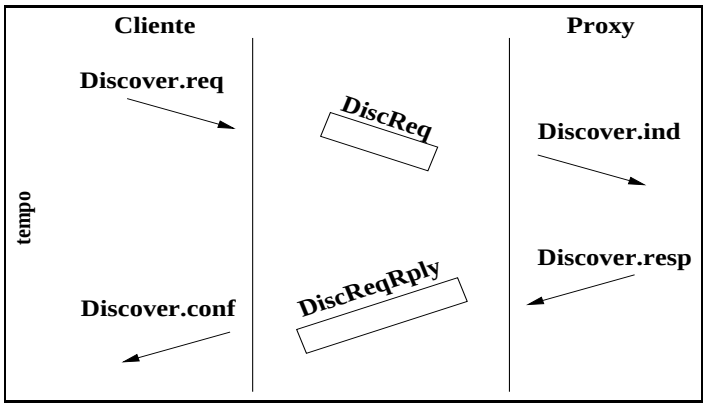

Figura 5.13: Requisição de descoberta.

5.4.6 Resposta de Requisição de Descoberta

Quando recebe uma solicitação de descoberta, o *proxy* a processa e responde ao cliente com uma mensagem do tipo *DiscReqRply*, que tem os campos *Tipo* com valor 5 e *Descrição de Serviço* com as descrições de serviços encontrados, respectivamente. A Figura 5.14 ilustra essa mensagem.

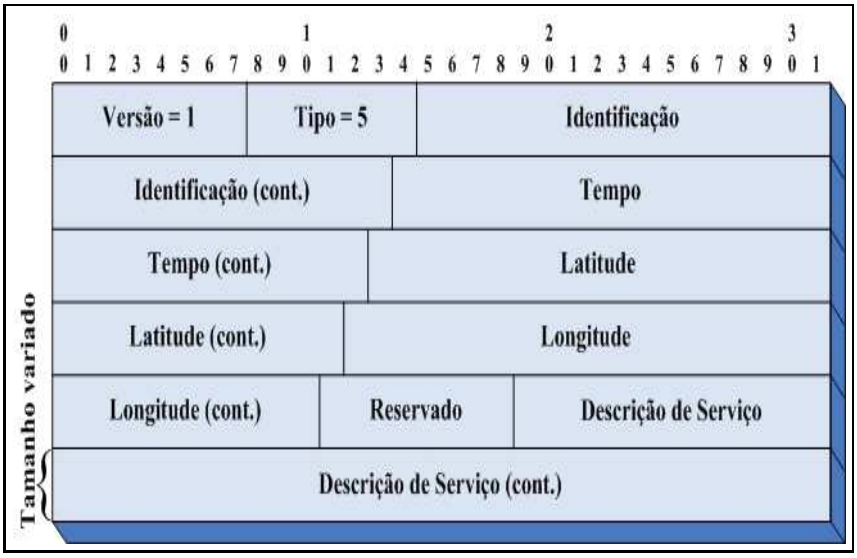

Figura 5.14: Resposta de requisição de descoberta.

5.4.7 Resposta Negativa de Requisição de Descoberta

Caso a autenticação não seja realizada, o *proxy* envia uma mensagem do tipo *DiscReq-RplyNeg* ao cliente com o campo *Tipo* com valor 6. A Figura 5.15 ilustra a mensagem.

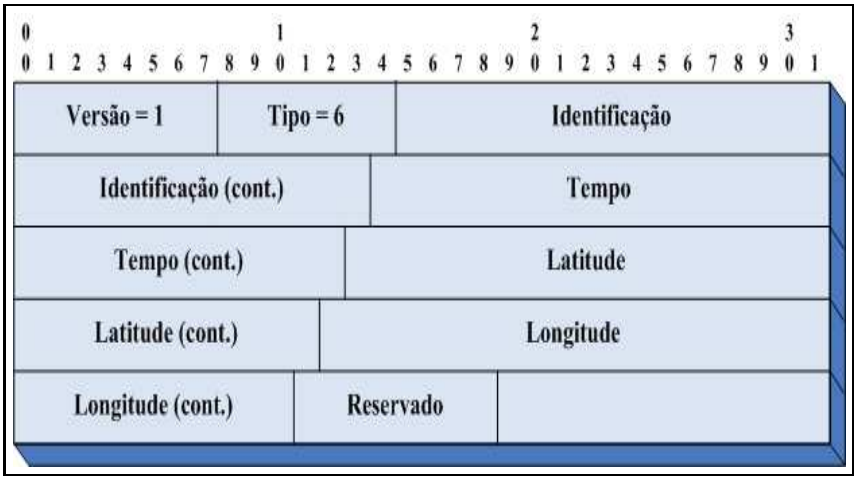

Figura 5.15: Resposta negativa de requisição de descoberta.

5.4.8 Cancelamento de Registro de Serviço

O provedor de serviço cancela seu registro no *proxy* utilizando a primitiva *Deregister*. A mensagem utilizada é do tipo *SrvDeReg* com os campos *Tipo* com valor 7, *Identificação*

com o número de identificação do provedor e *Senha Pro.* com a senha do provedor. As Figuras 5.16 e 5.17 ilustram a mensagem e o cancelamento de registro.

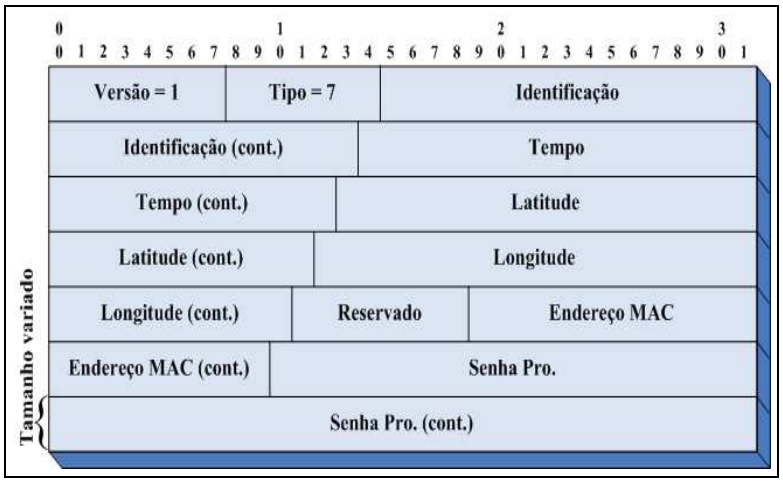

Figura 5.16: Mensagem de cancelamento registro.

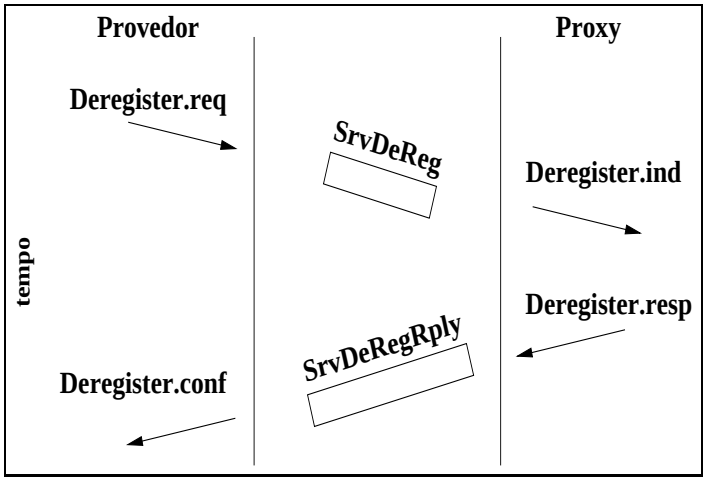

Figura 5.17: Cancelamento de registro

O cancelamento de registro é cancelado se o provedor não receber uma resposta do *proxy* dentro de um intervalo de tempo não superior a *TIMEOUT*.

5.4.9 Resposta de Cancelamento de Registro de Serviço

Ao receber uma mensagem de cancelamento de registro, o *proxy* a processa e responde ao provedor com uma mensagem do tipo *SrvDeRegRply*, enviada com o campo *Tipo* com valor 8. A Figura 5.18 ilustra essa mensagem.

5.4.10 Resposta Negativa de Cancelamento de Registro de Serviço

Caso a autenticação não seja realizada, o *proxy* envia uma mensagem do tipo *SrvDeRegRplyNeg* ao provedor com o campo *Tipo* com valor 9. A Figura 5.19 ilustra essa mensagem.

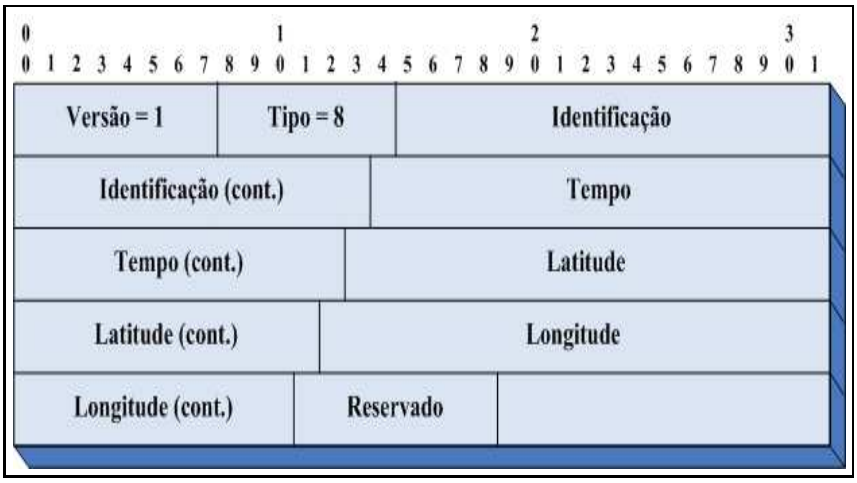

Figura 5.18: Resposta de cancelamento de registro de serviço.

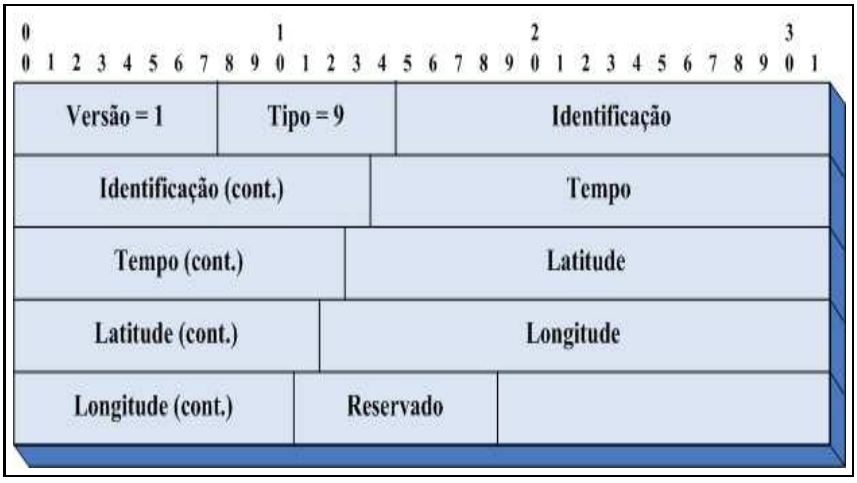

Figura 5.19: Resposta negativa de cancelamento de registro.

5.4.11 Anúncio de Serviço

O provedor de serviço, após ter se registrado no *proxy*, anuncia-se utilizando a primitiva *Advert*. A mensagem utilizada é do tipo *SrvAdvert* e contém os campos *Tipo* com valor 10, *Identificação* com a identificação do provedor, *Pooling* com o intervalo de tempo entre anúncios, *Senha Pro.* com a senha do provedor, *Descrição de Serviço* com a descrição semântica do serviço e *Tipo de Serviço* com o tipo de serviço. As Figuras 5.20 e 5.21 ilustram a mensagem e o anúncio de serviço.

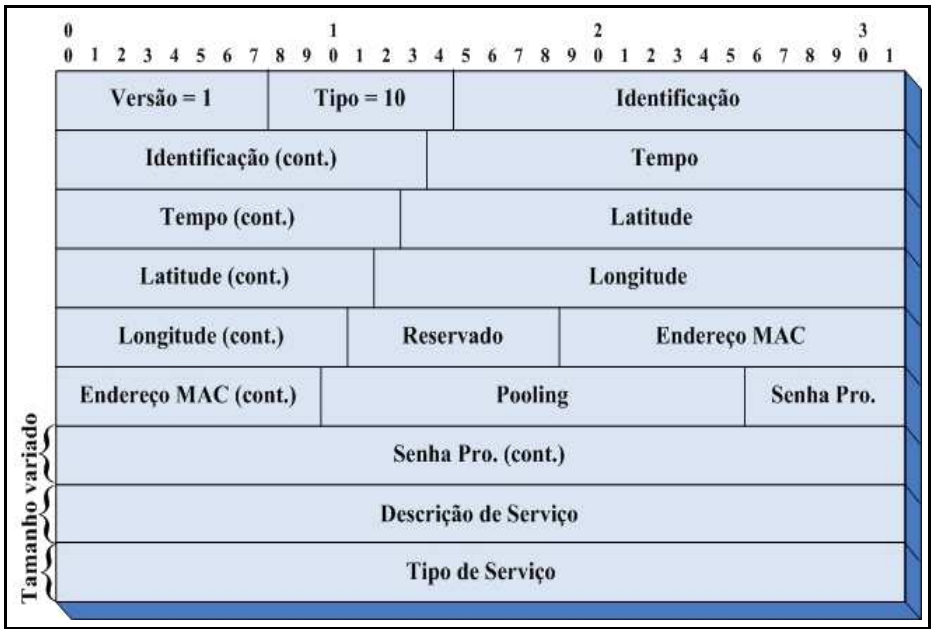

Figura 5.20: Mensagem de anúncio serviço.

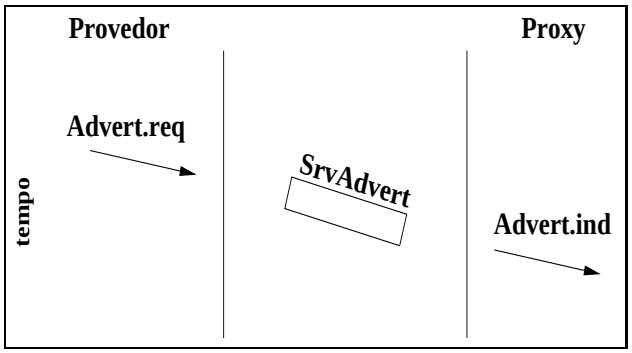

Figura 5.21: Anúncio de serviço.

5.5 Erros

As mensagens *SrvReg*, *DiscReq* e *SrvDeReg* são as únicas que recebem notificação de erro através de mensagens do tipo *SrvRegRplyNeg*, *DiscRegRplyNeg* e *SrvDeRegRplyNeg*, respectivamente. Todos os outros tipos de mensagem não possuem resposta. Um anúncio de provedor de serviço, por exemplo, consiga o *proxy* processá-lo ou não, não possui confirmação de recebimento. Os anúncios não processáveis de provedores são silenciosamente

descartados e os provedores de serviço, por sua vez, descartam anúncios de *proxy* não processáveis. O não envio de confirmação do recebimento de tais mensagens é justificado pelo fato de novos anúncios serem feitos a cada intervalo de tempo. Assim, se o *proxy* ou algum provedor de serviço não conseguir processar um anúncio recebido, um outro será recebido mais tarde e com informações atualizadas.

Os erros previstos no protocolo são:

- *Erro de versão*: indica que a versão do protocolo não é suportada.
- *Erro de autenticação*: indica que conta e/ou senha de acesso, ou senha de provedor são inválidas;
- *Erro de registro*: indica que registro do serviço falhou causado pela falha na autenticação; e
- *Erro de descoberta*: indica que a descoberta falhou causado pela falha na autenticação.

5.6 Conclusão

O SCaSDP é uma proposta de descoberta de serviço para sistemas sensíveis ao contexto. Possui características importantes que outras propostas não exploram, como a utilização de informação contextual e de ontologias no processo de descoberta. É dinâmico, seguro, adaptável à disponibilidade móvel, requer baixa interação humana e faz uso de um serviço de diretório para organizar e classificar informações sobre os serviços, clientes e provedores.

A informação contextual é suportada pelo protocolo através de descrições de serviço não restritivas, ou seja, com atributos é possível aumentar a quantidade de informações para caracterizar o contexto do serviço, pois a cada anúncio feito tais informações são atualizadas. A quantidade de atributos não é fixa, o que permite a cada serviço ter uma descrição particular.

OWL-S e a ontologia de dispositivos FIPA são utilizadas para compartilhar, inferir conhecimento e selecionar serviços. Além disso, devido a sua semântica formal, permitem uma manipulação automática e complexa das descrições e auxiliam no aumento da precisão do processo de descoberta.

A descoberta de serviços utiliza um algoritmo de comparação que faz uso descrições semânticas e informações contextuais para selecionar o serviço desejado. Esse algoritmo utiliza graus de similaridade para classificar a comparação em exata e aproximada, além de ordenar os serviços selecionados segundo o número de atributos que possuem.

A segurança é garantida com a utilização de um mecanismo de criptografia das mensagens trocadas entre provedores de serviço, clientes e *proxies*. Como no SPDP e no *Splendor*, a comunicação entre essas entidades é cifrada. Similarmente ao SSDS e SLP, utiliza mecanismos de autenticação para permitir que clientes solicitem descoberta de serviços e que provedores de serviço registrem-se e anunciem-se.

As entidades cliente e provedor de serviço utilizam um *proxy* para solicitar descoberta e anunciar descrições de serviços, respectivamente. Como no *Salutation*, SLP, SSDS, SLM e *Carmen*, o *proxy* do SCaSDP faz uso de um serviço de diretório para organizar e classificar as informações de serviços, provedores e clientes. O protocolo, como em *Carmen* e SLM,

também prevê a comunicação entre vários *proxies* durante a descoberta de serviços não existentes em um ambiente no qual um cliente esteja inserido.

O SCaSDP pode ser muito útil em ambientes onde há um alto dinamismo das informações tratadas e mobilidade de usuários, como em hospitais, universidades, empresas e em sítios de pesquisa. O Experimento de Grande Escala da Biosfera-atmosfera da Amazônia (LBA), por exemplo, possui sítios de pesquisa espalhados por toda a região amazônica. Nesses sítios são realizados experimentos sobre as mudanças pelas quais tem passado toda aquela região. Esses experimentos se utilizam de sensores e sistemas localizados em vários pontos estratégicos. Para um pesquisador do LBA que atua em várias frentes de pesquisa e tem de lidar com as constantes mudanças de contexto, causadas pelas características peculiares de cada sítio, uma plataforma como a *Infraware* e um protocolo de descoberta como o SCaSDP podem facilitar a descoberta e a utilização de serviços necessários à monitoração dos experimentos, à coleta e processamento de dados relevantes para a compreensão das mudanças climáticas, físicas e químicas pelas quais passa a região amazônica. No apêndice A podem ser encontradas maiores informações sobre o LBA que justificam a utilização do SCaSDP.

No capítulo seguinte são descritas as Redes de Petri e as Máquinas de Estados de processos considerados mais importantes do SCaSDP.

Capítulo 6

Validação Formal do SCaSDP

6.1 Introdução

Segundo [Holzmann 1991] um protocolo é melhor entendido como uma máquina de estados. Para validar algumas propriedades do SCaSDP e melhor entender os estados pelos quais passa durante sua utilização, foram utilizados dois modelos formais de especificação de protocolos: Redes de Petri e Máquinas de Estados Finita.

Redes de Petri são uma ferramenta de modelagem gráfica e matemática aplicável a muitos sistemas [Murata 1989]. O modelo de Redes de Petri foi proposto por Carl Adam Petri em sua dissertação "Kommunikation mit Automaten" em 1962, na faculdade de Matemática e Física da Universidade Técnica de Darmstadt. O objetivo do trabalho era modelar a comunicação entre autômatos, utilizados para representar sistemas a eventos discretos [Cardoso e Valette 1997].

Devido à sua natureza geral e não restritiva, Redes de Petri podem ser utilizadas em várias áreas de aplicação, dentre elas a de processos concorrentes, assíncronos, distribuídos, paralelos, não determinísticos e estocásticos. É fundamentada em um formalismo matemático, o que a torna uma ferramenta formal de modelagem. Protocolos de comunicação de rede podem ser modelados e validados utilizando Redes de Petri, garantindo, através das propriedades matemáticas, que possui boas propriedades como ausência de *deadlocks*, ciclos ruins, *livelocks* e serem reiniciáveis.

Neste capítulo são descritas as Redes de Petri de processos considerados mais importantes do SCaSDP. Em seguida são analisados os autômatos gerados a partir das Redes de Petri.

6.2 Redes de Petri do SCaSDP

A partir da especificação do protocolo SCaSDP, feita no capítulo 5, foram elaboradas duas redes de Petri, uma para registro de serviço e outra para descoberta de serviço. Para verificar e simular o comportamento do protocolo nessas duas situações foi utilizado o simulador PIPE (*Platform Independent Petri-Net Edit*) versão 2.0 [Bloom et al. 2004]. Esta ferramenta possui vários módulos que permite classificar, comparar, analisar invariantes e simular o comportamento da rede de petri especificada e da máquina de estados associada a essa rede.

6.2.1 Registro de Serviço

O protocolo SCaSDP oferece um serviço confirmado para o registro de provedores de serviços. As mensagens trocadas entre as entidades são *SrvReg*, *SrvRegRply*, *SrvRegRply-Neg* e *PrAdvert*, definidas em 5.4.2, 5.4.3, 5.4.4 e 5.4.1, respectivamente. É mostrado que esta especificação possui boas propriedades: não há *deadlock*, não há *livelock*, é reiniciável e é k-limitada.

As Figuras 6.1 e 6.2 mostram a rede de Petri para o registro de serviço e sua classificação feita utilizando ferramenta PIPE.

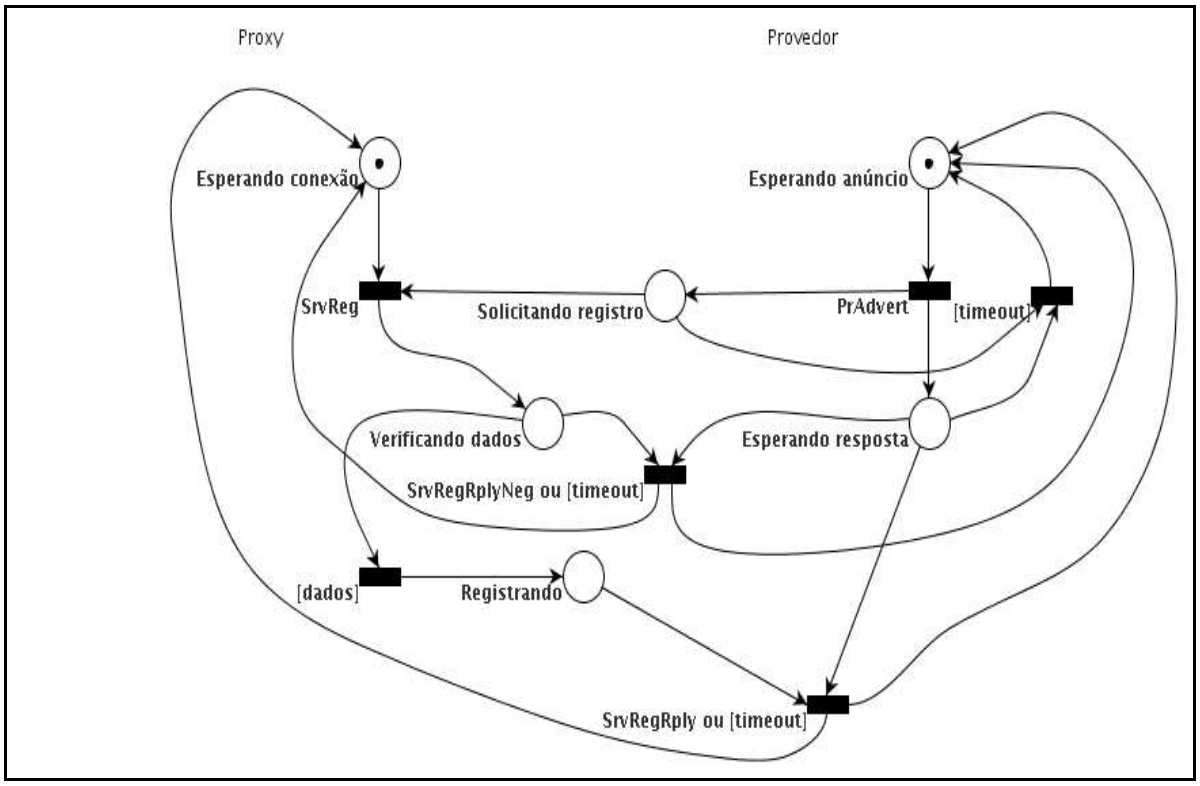

Figura 6.1: Rede de Petri para registro de serviço.

6.2.2 Descoberta de Serviço

Para encontrar serviços os clientes fazem requisições ao *proxy* da rede através de conexões seguras. Informando alguns parâmetros os clientes especificam em que tipo de serviços estão interessados. As mensagens utilizadas durante a descoberta de serviços são *DiscReq*, *DiscReqRply* e *DiscReqRplyNeg*, definidas em 5.4.5, 5.4.6 e 5.4.7, respectivamente. É mostrado que esta especificação possui boas propriedades: não há *deadlock*, não há *livelock*, é reiniciável e é k-limitada.

As Figuras 6.3 e 6.4 mostram a rede de Petri para descoberta de serviço e sua classificação feita utilizando a ferramenta PIPE.

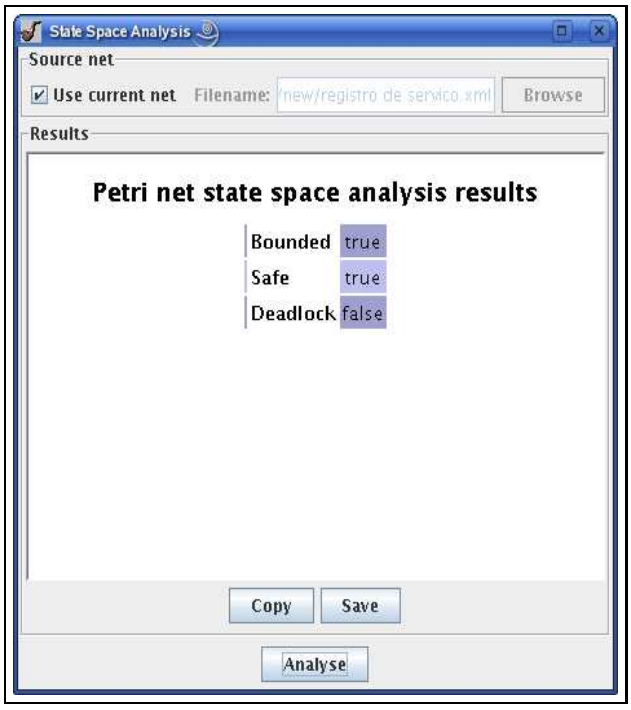

Figura 6.2: Classificação da rede de Petri para registro de serviço.

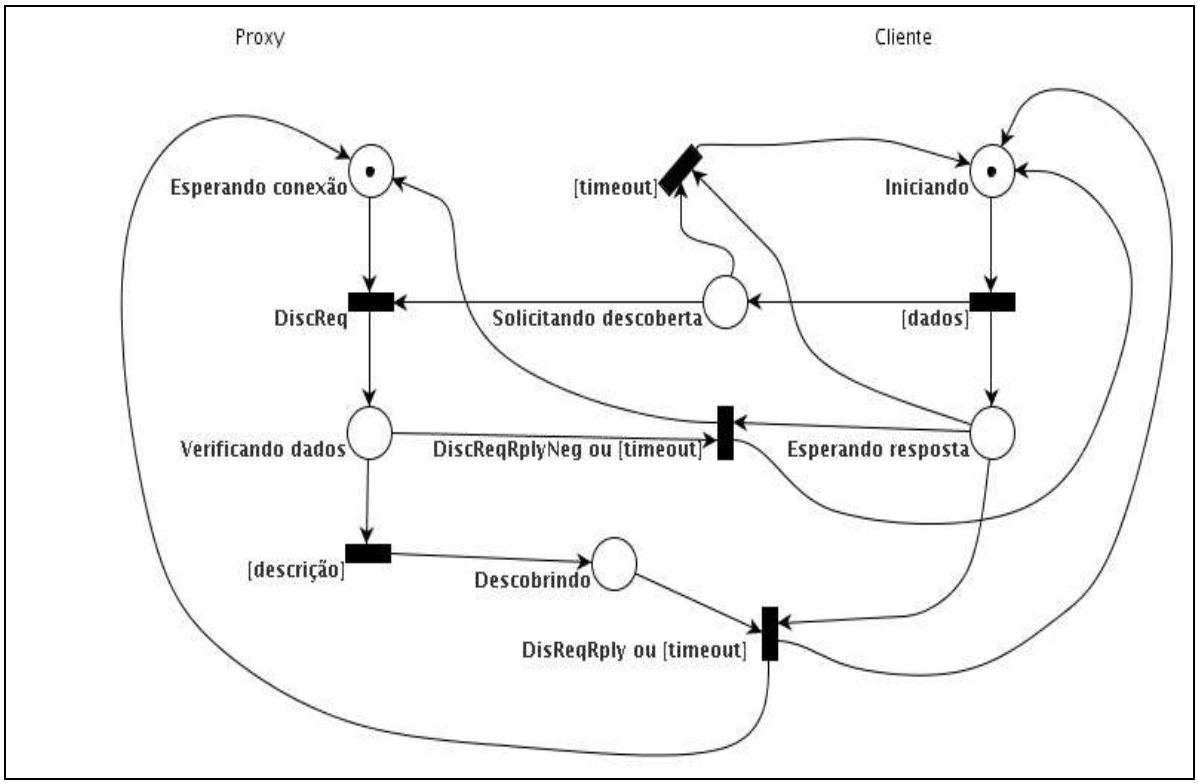

Figura 6.3: Rede de Petri para descoberta de serviço.

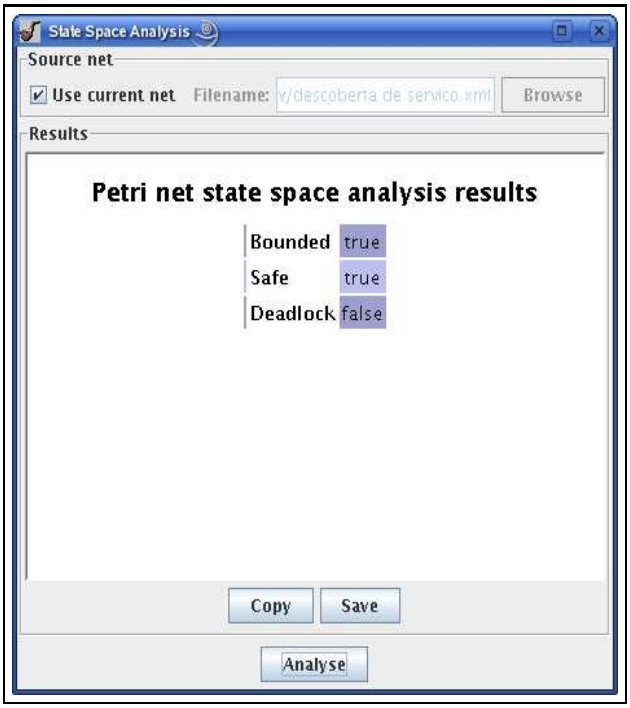

Figura 6.4: Classificação da rede de Petri para descoberta de serviço.

6.3 Autômatos relativos às fases do SCaSDP

A partir das Redes de Petri elaboradas na seção 6.2 são gerados os autômatos das entidades *proxy*, provedor de serviço e cliente referentes ao registro e descoberta de serviço. Há outras partes do protocolo que não foram aqui abordadas por serem mais simples e de mais fácil verificação.

6.3.1 *Proxy*

As tabelas 6.1, 6.2 e 6.3 apresentam as máquinas de estados do *proxy* no que diz respeito ao registro de serviço, anúncio de provedor de serviço e descoberta de serviço, respectivamente. As Figuras 6.5, 6.6 e 6.7 apresentam os diagramas de transição de estados para registro de serviço, anúncio de provedor e descoberta de serviço, respectivamente.

Estado Corrente	**Entrada**	**Saída**	**Próximo Estado**
Esperando conexão	-	SrvReg	Verificando dados
Verificando dados	SrvReg	SrvRegRplyNeg	Esperando conexão
Verificando dados	SrvReg	[timeout]	Esperando conexão
Verificando dados	SrvReg	[dados]	Registrando
Registrando	[dados]	SrvRegRply	Esperando conexão
Registrando	[dados]	[timeout	Esperando conexão

Tabela 6.1: Máquina de estados - *Proxy*: Registro de serviço.

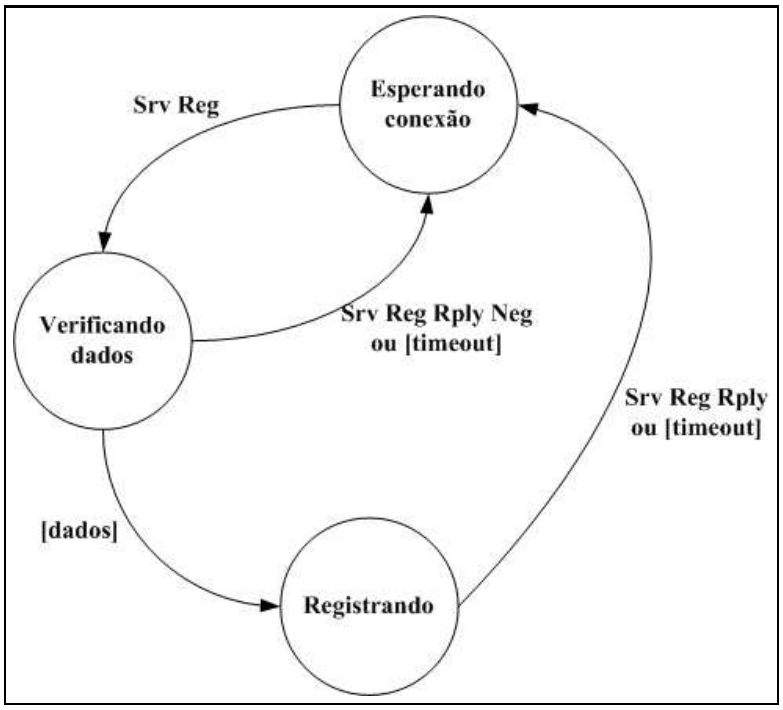

Figura 6.5: Diagrama de transição de estados - *Proxy*: Registro de Serviço.

Estado Corrente	**Entrada**	**Saída**	**Próximo Estado**
Esperando conexão	-	SrvAdvert	Processando
Processando	SrvAdvert	[erro]	Esperando conexão
Processando	SrvAdvert	[dados]	Atualizando serviço
Atualizando serviço	[dados]	-	Esperando conexão

Tabela 6.2: Máquina de estados - *Proxy*: Anúncio de provedor.

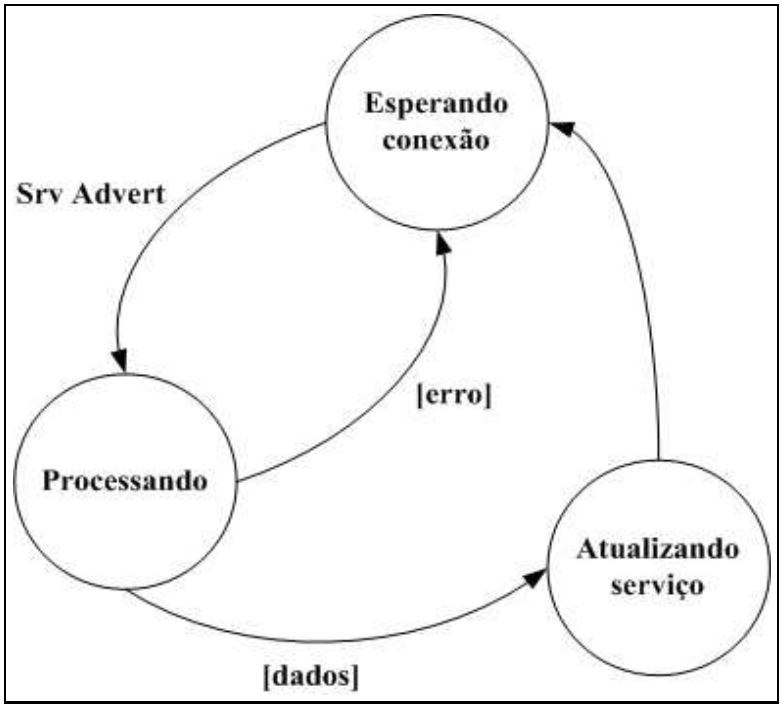

Figura 6.6: Diagrama de transição de estados - *Proxy*: Anúncio de Provedor.

Estado Corrente	**Entrada**	**Saída**	**Próximo Estado**
Esperando conexão	-	DiscReq	Verificando dados
Verificando dados	DiscReq	DiscReqRplyNeg	Esperando conexão
Verificando dados	DiscReq	[timeout]	Esperando conexão
Verificando dados	DiscReq	[descrição]	Descobrindo
Descobrindo	[descrição]	DiscReqRply	Esperando conexão
Descobrindo	[descrição]	[timeout]	Esperando conexão

Tabela 6.3: Máquina de estados - *Proxy*: Descoberta de serviço.

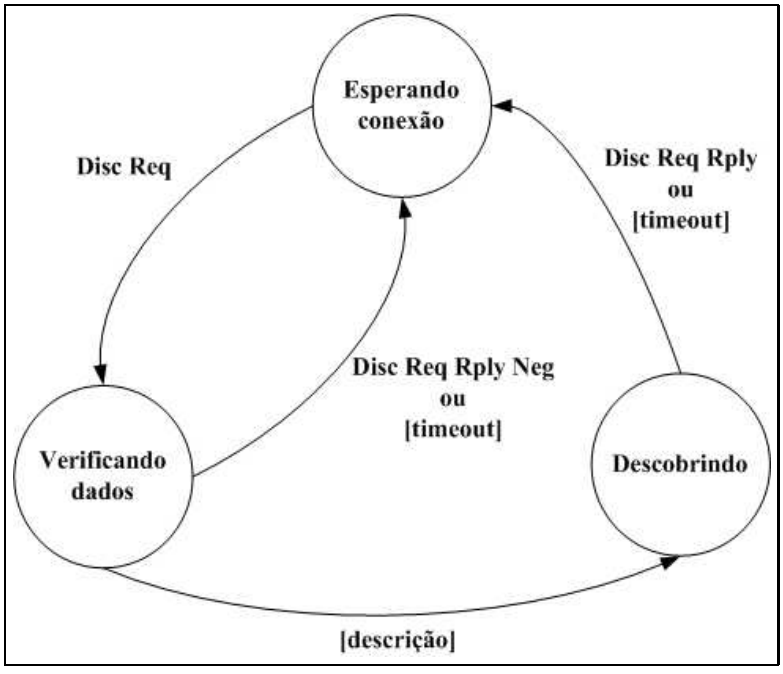

Figura 6.7: Diagrama de transição de estados - *Proxy*: Descoberta de Serviço.

6.3.2 Provedor

A tabela 6.4 apresenta a máquina de estados do provedor para registro de serviço junto ao *proxy* e a Figura 6.8 mostra seu diagrama de transição de estados.

Estado Corrente	**Entrada**	**Saída**	**Próximo Estado**
Esperando anúncio do *proxy*	-	PrAdvert	Solicitando registro
Solicitando registro	PrAdvert	SrvReg	Esperando resposta
Esperando resposta	SrvReg	SrvRegRplyNeg	Esperando anúncio do *proxy*
Esperando resposta	SrvReg	[timeout]	Esperando anúncio do *proxy*
Esperando resposta	SrvReg	SrvRegRply	Esperando anúncio do *proxy*

Tabela 6.4: Máquina de estados - Provedor: Registro de serviço.

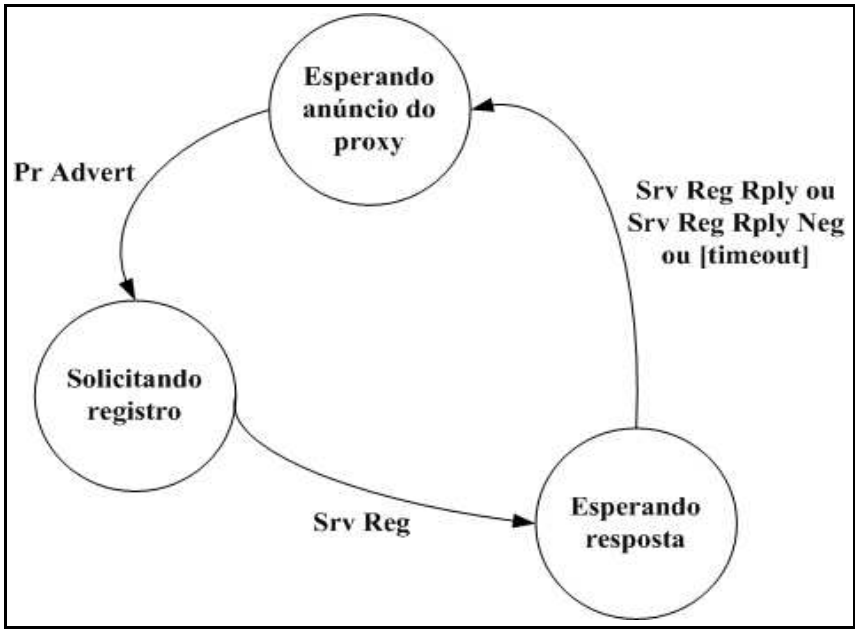

Figura 6.8: Diagrama de transição de estados - Provedor: Registro de Serviço.

6.3.3 Cliente

A tabela 6.5 apresenta a máquina de estados do cliente para descoberta de serviço junto ao *proxy* e a Figura 6.9 mostra seu diagrama de transição de estados.

Estado Corrente	**Entrada**	**Saída**	**Próximo Estado**
Iniciando	-	[descrição]	Solicitando descoberta
Solicitando descoberta	[descrição]	*Discover*	Esperando resposta
Esperando resposta	*Discover*	DiscReqRply	Terminando
Esperando resposta	*Discover*	DiscReqRplyNeg	Terminando
Esperando resposta	*Discover*	[timeout]	Terminando

Tabela 6.5: Máquina de estados - Cliente: Descoberta de serviço.

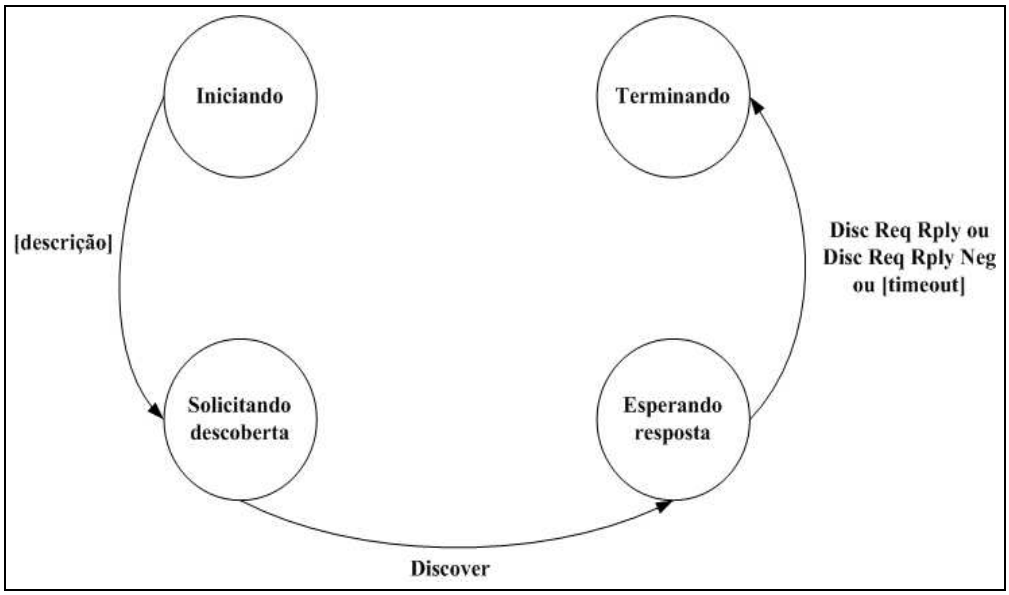

Figura 6.9: Diagrama de transição de estados - Cliente: Descoberta de Serviço.

6.4 Conclusão

A validação formal é uma maneira de garantir que um protocolo possui boas propriedades, como ausência de *deadlocks* e *livelocks*. Uma forma de mostrar essas e outras propriedades importantes é utilizando o modelo de Redes de Petri.

Foram modeladas as Redes de Petri para registro e descoberta de serviço e em seguida, a partir da validação do protocolo, foram geradas as máquinas de estados juntamente com seus diagramas de transição de estados.

No capítulo seguinte é descrita a implementação do protótipo e avaliados os testes realizados em laboratório.

Capítulo 7

Implementação do Protocolo SCaSDP

7.1 Introdução

Para verificar a funcionalidade do SCaSDP foi implementado um protótipo, desenvolvido sobre uma rede local *Ethernet*. Este não é um requisito essencial, pois o protocolo visa ser adaptável a qualquer meio de comunicação que suporte os protocolos de transporte TCP e UDP.

Neste capítulo são especificados o ambiente de desenvolvimento, o algoritmo de comparação semântica e as abordagens para criptografia e autenticação utilizados.

7.2 Configuração do Protocolo

O protótipo desenvolvido compreende a implementação do protocolo para as três entidades, ou seja, cliente, provedor de serviço e *proxy*. A interação entre *proxies* de diferentes domínios não foi contemplada pelo protótipo, pois não traz grande impacto para demonstrar seu funcionamento. Foi utilizado o sistema operacional Linux, distribuição *SuSE 9.1* [SuSE Linux 2006] e a plataforma de desenvolvimento *NetBeans* versão 4.1 [Microsystems 2006], ambos de distribuição livre. A linguagem de programação utilizada foi Java, versão 1.5.0_06, para permitir uma fácil portabilidade para outras plataformas.

O cliente SCaSDP, para ser utilizado, requer que o usuário proveja um arquivo que contenha a descrição em OWL do serviço desejado, o endereço do *proxy* (IP ou nome da máquina), a conta e a senha de acesso. Essas informações são enviadas, via SSL, ao *proxy* SCaSDP, que autentica o cliente e dá início à descoberta com base na descrição do serviço desejado. Como resposta é retornado ao cliente um único arquivo OWL que contém o(s) serviço(s) encontrados. A apresentação do resultado ao usuário não é de responsabilidade do cliente do protocolo e sim de alguma aplicação que interprete o formato OWL. Para isso podem ser utilizados navegadores *Internet*, ou mesmo alguma aplicação desenvolvida especialmente para este propósito.

Para que sejam iniciados, o provedor e o *proxy* SCaSDP precisam ser configurados. A configuração é feita através dos comandos: *java Proxy -conf* e *java Provider -conf*, para *proxy* e provedor, respectivamente. As Figuras 7.1 e 7.2 ilustram o processo de configuração para o provedor e o *proxy*, respectivamente.

Na configuração do provedor o usuário deve fornecer as seguintes informações:

```
Gerando configuracao.
Porta de comunicacao TCP [30000]:
prompt> Porta de comunicacao UDP [30000]:
prompt> Porta de anuncio do Proxy [30001]:
prompt> Tempo de espera [300 s]:
prompt> Tempo entre anuncios [300 s]:
prompt> Semente para geracao de chaves de criptografia [1024]:
prompt> Endereco MAC [00:00:00:00:00:00]:
prompt>  -> Nome do host: yara
      Deseja alterar o nome do host? [S/N]:
prompt> Dominio do host [local]:
prompt> Conta de acesso ao Proxy [null]:
prompt> Senha da conta de acesso [null]:
prompt> Identificacao do provedor no Proxy [0]:
prompt> Senha do provedor [null]:
prompt> Senha do certificado [null]:
prompt> Comando para gerar a descricao OWL:
prompt> Criando arquivos de configuracao:
 -> Gerando arquivo de configuracao. Gerado.
 -> Criando diretorio das chaves de criptografia. Criado.
 -> Gerando chaves de criptografia. Geradas.
 -> Griando diretorio de descricao de servico. Criado.
 -> Gerando arquivo de descricao de servico. Gerado.
 -> Criando diretorio de programas. Criado.
```

Figura 7.1: Processo de configuração da entidade Provedor.

- Porta de comunicação TCP: porta TCP utilizada para anúncios do provedor. Por padrão é 30000;
- Porta de comunicação UDP: porta UDP utilizada para anúncios do provedor. Por padrão é 30000;
- Porta de anúncio do *proxy*: porta UDP utilizada para receber anúncios do *proxy*. Por padrão é 30001;
- Tempo de espera: tempo de espera antes que o protocolo cancele uma conexão TCP. Este tempo é também utilizado pelo provedor para verificar a atividade do *proxy*. Por padrão é 5 minutos;
- Tempo entre anúncios: intervalo de tempo entre um anúncio e outro. Por padrão é 5 minutos;
- Semente para geração das chaves de criptografia: número inteiro utilizado pelo método de criptografia para gerar as chaves pública e privada. Por padrão é 1024;
- Endereço MAC do dispositivo: o usuário deve informar o endereço MAC do dispositivo;
- Nome do dispositivo: é obtido automaticamente, mas pode ser alterado pelo usuário;
- Domínio do dispositivo: domínio segundo a terminologia utilizada na *Internet*. O usuário deve informar o domínio;
- Conta de acesso ao *proxy*: conta de acesso para que o *proxy* autentique e registre o provedor;

- Senha da conta de acesso: senha da conta de acesso para que o *proxy* autentique e registre o provedor;
- Identificação do provedor no *proxy*: se o provedor já possui uma identificação o usuário deve informá-la. Caso contrário deve deixar esse campo inalterado;
- Senha do provedor: senha do provedor utilizada pelo *proxy* para autenticar as mensagens de anúncio do provedor;
- Senha do certificado: senha do certificado de segurança para que o provedor possa estabelecer conexões via SSL com o *proxy*. O certificado deve ser obtido antes do processo de configuração junto ao administrador do sistema; e
- Comando para gerar descrição: comando utilizado pelo provedor para gerar a descrição OWL do serviço. Cabe ao usuário definir esse comando, que pode ser um programa que gere automaticamente a descrição OWL do serviço provido.

Após a configuração, o provedor deve ser registrado através do comando *java Provider -reg*. Se o registro for realizado com sucesso, o provedor pode ser iniciado com o comando *java Provider -start*, que a partir de então faz anúncios periódicos ao *proxy*. Se não receber anúncios do *proxy* por um período de tempo superior ao especificado durante a configuração, o provedor suspende os anúncios até que receba um novo anúncio do *proxy*. Quando isto acontece, o provedor atualiza o registro e reinicia o envio de anúncios. Isto permite que dispositivos computacionais com poucos recursos não os desperdicem.

Além de registrar um provedor é possível também cancelar seu registro utilizando o comando *java Provider -dereg*. Isto foi definido para permitir que provedores de serviço que estejam fora de operação possam ter seu registro cancelado.

Na configuração do *proxy*, ilustrada pela Figura 7.2, o administrador deve fornecer as seguintes informações:

- Porta de comunicação TCP: porta TCP utilizada para receber anúncios de provedores de serviço. Por padrão é 30000;
- Porta de comunicação UDP: porta UDP utilizada para receber anúncios de provedores. Por padrão é 30000;
- Porta de anúncio do *proxy*: porta UDP utilizada para anúncios do *proxy*. Por padrão é 30001;
- Tempo de espera: tempo de espera antes que o protocolo cancele uma conexão TCP. Por padrão é 5 minutos;
- Tempo entre anúncios: intervalo de tempo entre um anúncio e outro. Por padrão é 5 minutos;
- Semente para geração das chaves de criptografia: número inteiro utilizado pelo método de criptografia para gerar as chaves pública e privada. Por padrão é 1024;
- Endereço MAC do dispositivo: o usuário deve informar o endereço MAC do dispositivo;

```
Gerando configuracao.
Porta de comunicacao TCP [30000]:
prompt> Porta de comunicacao UDP [30000]:
prompt> Porta de anuncio do Proxy [30001]:
prompt> Tempo de espera [300 s]:
prompt> Tempo entre anuncios [300 s]:
prompt> Semente para geracao de chaves de criptografia [1024]:
prompt> Endereco MAC [00:00:00:00:00:00]:
prompt>  -> Nome do host: yara
      Deseja alterar o nome do host? [S/N]:
prompt> Dominio do host [local]:
prompt> Endereco broadcast [192.168.0.255]:
prompt> Senha do certificado [null]:
prompt> Criando arquivos de configuracao:
 -> Gerando arquivo de configuracao. Gerado.
 -> Criando diretorio das chaves de criptografia. Criado.
 -> Gerando chaves de criptografia. Geradas.
 -> Criando diretorio de programas. Criado.
```

Figura 7.2: Processo de configuração da entidade *Proxy*.

- Nome do dispositivo: é obtido automaticamente, mas pode ser alterado pelo usuário;
- Domínio do dispositivo: domínio segundo a terminologia utilizada na *Internet*. O usuário deve informar o domínio;
- Endereço de *broadcast*: endereço, obtido automaticamente, é utilizado pelo *proxy* para fazer seus anúncios. Pode ser alterado pelo administrador; e
- Senha do certificado: senha do certificado de segurança para que o *proxy* possa estabelecer conexões com provedores de serviço. O certificado deve ser obtido antes do processo de configuração junto ao administrador do sistema.

O *proxy* pode ser iniciado com o comando *java Proxy -start*, que a partir de então passa a estar disponível aos provedores de serviço e clientes. As descrições de serviço recebidas através dos anúncios dos provedores de serviço são autenticadas, decifradas, descompactadas e só então são armazenadas. Quando um cliente solicita descoberta de algum serviço, o *proxy* autentica a requisição e dá início ao processo de descoberta consultando as descrições de serviços cujos provedores estão ativos. Em seguida responde ao cliente com o resultado encontrado.

7.2.1 Descrição de Serviço

O SCaSDP utiliza descrições de serviço em OWL. Foi utilizada a ontologia para dispositivos da *Foundation for Intelligent Physical Agents - FIPA* e uma ontologia de informações contextuais básicas definida para este trabalho. A Figura 7.3 ilustra as ontologias criadas utilizando a ferramenta *Protégé* [Informatics 2005].

O *Protégé* é um editor de ontologias desenvolvido pela Universidade de Stanford, nos Estados Unidos. Ele é um *software livre*, de código aberto, desenvolvido em Java e

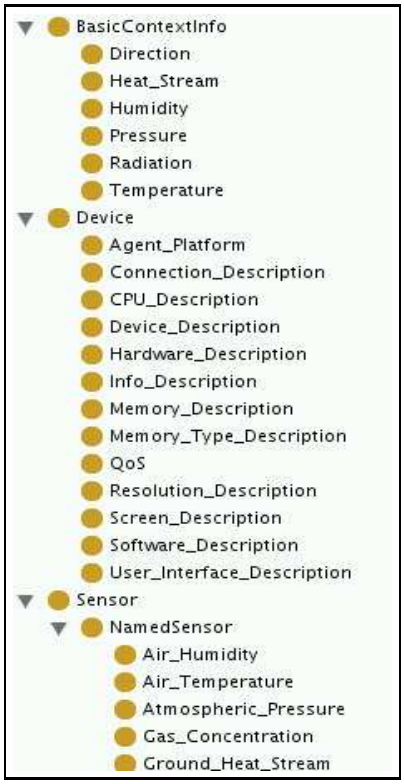

Figura 7.3: Ontologias de dispositivos, informações contextuais e sensores.

com possibilidades de extensão através de *plug-ins*. Um desses é o *Protégé OWL Plugin* [Informatics 2005], utilizado para definir as ontologias.

A ontologia de dispositivos da FIPA provê suporte às aplicações para que analisem informações do dispositivo e tomem decisões. Por exemplo, se a aplicação requerer que o serviço forneça certo nível de *QoS*, ou que não tenha muitos processos sendo atendidos, ela pode especificar essas restrições no arquivo de descrição do serviço desejado e durante o processo de descoberta elas são consideradas. Essas informações são contextuais, pois caracterizam a situação do serviço no momento do envio da mensagem e são utilizadas no processo de inferência feito pelo algoritmo de comparação.

As descrições tomam por base as informações definidas em 5.2.3 e são geradas por um programa externo ao protocolo. Como foi mencionado anteriormente, o SCaSDP não gera as descrições dos serviços. Isto torna o protocolo versátil, permitindo que o administrador, utilizando as ontologias que achar necessárias, crie a descrição adequada para o serviço.

No exemplo abaixo é mostrada uma descrição de serviço para o serviço de diagnóstico por ultrassom.

```
<actor:Actor rdf:ID="informacao_contato">
  <actor:phone rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >33406508</actor:phone>
  <actor:physicalAddress rdf:datatype="http://www.w3.org/2001/
                                              XMLSchema#string">
  R. Professor Tiago Soares Torres, 117</actor:physicalAddress>
  <actor:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Servico de Ultrassonografia do Hospital Praia da Costa</actor:name>
</actor:Actor>
```

```
<profile:ServiceCategory rdf:ID="paginas_amarelas">
  <profile:categoryName rdf:datatype="http://www.w3.org/2001/
                                              XMLSchema#string">
   Saude</profile:categoryName>
  <profile:value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  Servico de Diagnostico</profile:value>
</profile:ServiceCategory>
<process:Output rdf:ID="consulta">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                                               XMLSchema#anyURI">
   http://www.owl-ontologies.com/unnamed.owl#Ultrasom
  </process:parameterType>
</process:Output>
<process:Input rdf:ID="planoDeSaude">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                 XMLSchema#anyURI">
    http://www.owl-ontologies.com/unnamed.owl#PlanoDeSaude
  </process:parameterType>
  <process:parameterValue rdf:parseType="Literal">SMS
  </process:parameterValue>
</process:Input>
<process:Input rdf:ID="cidade">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                                                XMLSchema#anyURI">
    http://www.owl-ontologies.com/unnamed.owl#Cidade
  </process:parameterType>
  <process:parameterValue rdf:parseType="Literal">Vila_Velha
  </process:parameterValue>
</process:Input>
<profile:ServiceParameter rdf:ID="estacionamento">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
          2001/XMLSchema#boolean">
  true</profile:serviceParameterName>
</profile:ServiceParameter>
<profile:ServiceParameter rdf:ID="publico">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
          2001/XMLSchema#boolean">
  true</profile:serviceParameterName>
</profile:ServiceParameter>
<profile:ServiceParameter rdf:ID="horas24">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
          2001/XMLSchema#boolean">
  true</profile:serviceParameterName>
</profile:ServiceParameter>
<profile:ServiceParameter rdf:ID="localizacao">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
          2001/XMLSchema#string">
```

```
    -20.25897,-40.33325</profile:serviceParameterName>
  </profile:ServiceParameter>
```

Nessa descrição são definidos:

- O tipo de serviço: serviço de diagnóstico;
- A entrada exigida pelo serviço: plano de saúde e a cidade atendida;
- Informação contextual: localização, estacionamento, atendimento público e 24 horas

Como mencionado anteriormente, o cliente solicita a descoberta enviando uma descrição do serviço desejado. Abaixo segue o exemplo de uma requisição de descoberta do serviço de diagnóstico :

```
<profile:ServiceCategory rdf:ID="categoria">
  <profile:value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Servico de Diagnostico</profile:value>
</profile:ServiceCategory>
<process:Output rdf:ID="saida">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#anyURI">
    http://www.owl-ontologies.com/unnamed.owl#ExameLaboratorial
  </process:parameterType>
</process:Output>
<process:Input rdf:ID="entrada1">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#anyURI">
    http://www.owl-ontologies.com/unnamed.owl#PlanoDeSaude
  </process:parameterType>
  <process:parameterValue rdf:parseType="Literal">PASA
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">PHS
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">SAMP
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">SMS
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">UNIMED
  </process:parameterValue>
</process:Input>
<process:Input rdf:ID="entrada2">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#anyURI">
    http://www.owl-ontologies.com/unnamed.owl#Cidade
  </process:parameterType>
  <process:parameterValue rdf:parseType="Literal">Vitoria
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">Vila_Velha
```

```
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">Serra
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">Cariacica
  </process:parameterValue>
  <process:parameterValue rdf:parseType="Literal">Viana
  </process:parameterValue>
</process:Input>
<profile:ServiceParameter rdf:ID="horas24">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
                    2001/XMLSchema#boolean">
    true</profile:serviceParameterName>
</profile:ServiceParameter>
<profile:ServiceParameter rdf:ID="publico">
  <profile:serviceParameterName rdf:datatype="http://www.w3.org/
                    2001/XMLSchema#boolean">
    true</profile:serviceParameterName>
</profile:ServiceParameter>
<service:Service rdf:ID="Requisita_Consulta">
  <service:presents rdf:resource="#Requisita_Consulta_Profile"/>
</service:Service>
<profile:Profile rdf:ID="Requisita_Consulta_Profile">
  <service:presentedBy rdf:resource="#Requisita_Consulta"/>
  <profile:serviceCategory rdf:resource="#categoria"/>
  <profile:hasOutput rdf:resource="#saida"/>
  <profile:hasInput rdf:resource="#entrada1"/>
  <profile:hasInput rdf:resource="#entrada2"/>
  <profile:serviceParameter rdf:resource="#horas24"/>
  <profile:serviceParameter rdf:resource="#publico"/>
</profile:Profile>
```

Esse arquivo descreve o serviço desejado da seguinte forma:

- Tipo de serviço: Serviço de Diagnostico;
- Saída: saída requerida pelo usuário: Exame Laboratorial;
- Entrada requisitada:
  - Plano de Saúde; e
  - Cidade.
- Informações contextuais:
  - Disponível 24 horas; e
  - Atendimento público.

Esta descrição é enviada ao *proxy* que a utiliza para selecionar o serviço solicitado.

7.2.2 Seleção de Serviços

Conforme visto em 5.2.5, o processo de descoberta faz uso de um algoritmo de comparação que utiliza o tipo de serviço, a entrada requisitada pelo serviço, a saída fornecida por ele e atributos (informações contextuais) do serviço para selecionar os serviços que atendem às necessidades do cliente. Para a implementação desse algoritmo foi utilizada a biblioteca Jena-2.3 [Programme 2006], que provê um conjunto de classes para a manipulação de classes e propriedades OWL. O algoritmo de comparação descrito a seguir:

1. Consultar S (simulado por um banco de dados) usando o tipo de serviço T_r e verificar o *leasing* do anúncio. O resultado é um conjunto de serviços S' do tipo T_r.

   Esta é a primeira etapa do algoritmo de comparação. Nesta etapa são selecionados os serviços cujo o tipo é semelhante ao que foi solicitado pelo cliente.

2. Para cada serviço em S', verificar se a saída fornecida, é a mesma saída S_r requisitada pelo usuário.

   (a) Se a saída for igual, consultar R em busca das entradas necessárias pelo serviço $s' \in S'$. O resultado são dois conjuntos E (serviços exatos) e I (serviços inexatos).

      i. Se todas as entradas são fornecidas por R, o serviço é classificado como exato e adicionado no conjunto E.

      ii. Se uma entrada não é fornecida por R, procure-a nos provedores de contexto.

         A. Se a entrada que estava faltando foi encontrada, o serviço é classificado como exato e inserido no conjunto E.

         B. Se a entrada não foi encontrada, o serviço é aproximado e incluído no conjunto I.

   (b) Se a saída não for igual, o serviço é descartado.

   Esta é a segunda etapa do algoritmo, quando os serviços cujas saídas fornecidas, ou seja o resultado do processamento do serviço, são selecionados. O resultado dessa verificação é um conjunto de serviços cujos tipos de serviço e as saídas fornecidas são as requisitadas pelo cliente.

3. Avaliar os atributos dos serviços.

   Nesta etapa são verificados os atributos, ou seja informações contextuais, dos serviços.

   (a) Avaliar os atributos dos serviços em E. Se existe algum serviço em E.

      i. Adicionar todos os atributos A_r da requisição na tabela de atributos.

      ii. Adicionar todos os serviços em E como linhas na tabela de atributos.

      iii. Para cada a ($a \in A_r$) determinar se o atributo é oferecido pelo serviço.

         A. Se o serviço oferece o atributo a, adiciona a relação.

      iv. Ordenar o conjunto E segundo a quantidade de relações. O resultado é uma lista ordenada E' de serviços.

Neste momento são selecionadas as descrições de serviços cuja comparação é exata e é feita uma ordenação decrescente dessas descrições.

(b) Avaliar os atributos dos serviços em *I*. Se existe algum serviço em *I*.

i. Adicionar todos os atributos A_r da requisição na matriz de atributos.

ii. Adicionar todos os serviços em I como linhas na tabela de atributos.

iii. Para cada a ($a \in A_r$) determinar se o atributo é oferecido pelo serviço.

A. Se o serviço oferece o atributo a, adiciona a relação.

iv. Ordenar o conjunto I segundo a quantidade de relações. O resultado é uma lista ordenada *I'* de serviços.

Neste momento são selecionadas as descrições de serviços cuja comparação é aproximada e é feita uma ordenação decrescente dessas descrições.

4. Retornar o resultado como uma lista ao cliente. Primeiro *E'* e depois *I'*.

Nesta etapa são retornadas duas listas ordenadas, uma com as descrições cuja comparação é exata e outra com as aproximadas ao cliente.

Durante a fase de testes esse algoritmo atendeu às expectativas, selecionando as descrições de serviços com base na solicitação feita pelo usuário, como pode ser observado na seção 7.3. Entretanto, algumas melhorias podem ser feitas, entre elas levar em consideração a preferência do usuário por um serviço, utilizando uma ontologia do perfil dele e utilizar ontologias mais complexas, que descrevam com mais precisão o contexto do ambiente. Além disso, é necessário verificar a escalabilidade e o desempenho do algoritmo em ambientes onde há um número grande e constante de requisições de descoberta.

7.2.3 Serviço de Diretório

Não foi encontrado um serviço de diretório que suportasse descrições feitas com OWL e por isso, para simulá-lo foi utilizado o SGBD relacional *PostgreSQL*, versão 8.1.2-404 [PostgreSql 2005], para armazenar as descrições de serviços anunciadas pelos provedores.

Utilizar um SGBD relacional em vez de um serviço de diretório não é uma opção adequada, tendo em vista que um serviço de diretório possui funcionalidades específicas e adequadas para a manipulação das informações, como:

- Organização dos dados de modo hierárquico;
- Armazenagem distribuída e replicada dos dados em diferentes servidores de diretório;
- Gerenciamento de identidades e dos relacionamentos entre elas;
- Pesquisas avançadas sobre atributos diferentes associados com os objetos no diretório; e
- Autenticação, que determina quem pode acessar os recursos e com que grau de acessibilidade.

O banco de dados criado para fins de testes do protocolo conta com tabelas utilizadas para armazenar as informações, conta e senha de acesso dos provedores e as descrições de serviço. Não é feita classificação alguma das descrições, muito menos é definida qualquer regra que restrinja o acesso a determinados grupos. As tabelas utilizadas foram:

- Provedores de serviço: utilizada para armazenar as descrições anunciadas, tipo de serviço, senha do provedor, localização, endereços IP e MAC do dispositivo, intervalo entre anúncios e a chave pública do provedor;
- Provedores de serviço removidos: utilizada para armazenar informações dos provedores de serviço cujo registro foi cancelado. Esta tabela é idêntica a de provedores de serviço e é mantida para fins de estatística do *proxy*;
- Usuários: utilizada para armazenar conta e senha de administradores e usuários do protocolo. Esta tabela serviu para simular o processo de autenticação para registro, cancelamento de registro e anúncio de provedores, bem como a autenticação do cliente do protocolo durante a requisição de descoberta de serviço; e
- Informação contextual: utilizada para simular provedores de informação contextual. Foi usada apenas a informação contextual de localização no processo.

O acesso ao banco de dados é feito pelo SCaSDP por meio de rotinas que utilizam a API Java do *PostgreSQL*. Além disso,o administrador pode acessar as informações através da interface de gerenciamento *PhPPgAdmin* [King-Lynne 2005]. Ao receber uma solicitação de registro de um provedor de serviço, o *proxy* estabelece uma conexão com o SGBD e persiste os dados do provedor no banco de dados através de rotinas desenvolvidas especificamente para atender a esta necessidade. A comunicação entre o *proxy* e o SGBD é feita sobre o SSL, o que permite proteger as informações dos serviços de acessos não autorizados.

Com um serviço de diretório as descrições semânticas de serviço e informações de provedores de serviço, aplicações e usuários podem ser organizadas e gerenciadas segundo uma estrutura hierárquica, que pode ser empregada para tornar o processo de seleção mais eficiente. Além disso, os mecanismos de segurança providos por um serviço de diretório possibilitam um maior controle de acesso aos dados das entidades.

7.2.4 Segurança e Autenticação

O SCaSDP é um protocolo seguro pois toda comunicação entre as entidades é cifrada. Isto é feito através de métodos de criptografia implementados pela linguagem Java. É também utilizado o SSL para permitir comunicação segura via TCP. Certificados X.509, criptografia de chave pública, contas e senhas de acesso são outros itens utilizados para garantir a segurança e a autenticação.

Os certificados, que devem ser providos pelos administradores do sistema, são utilizados para que clientes e provedores de serviço consigam criar um canal de comunicação seguro via SSL com o *proxy*. O certificado usado pelo protótipo nos testes foi gerado com o comando:

```
prompt:~scasdp> java -Djavax.net.ssl.trustStore=truststore
-Djavax.net.ssl.trustStorePassword=123456 SCaSDPCert.crt
```

Tanto o *proxy*, quanto provedores de serviço, possuem chaves pública e privada de criptografia. As chaves do *proxy* são empregadas para a autenticação e a cifragem de mensagens trocadas com outros *proxies* durante o processo de descoberta. Essa funcionalidade, conforme mencionado antes, não foi prevista no protótipo. A chave privada

de criptografia dos provedores é utilizada para cifrar seus anúncios feitos ao *proxy*, que por sua vez decifra esses anúncios com a chave pública do provedor, recebida durante o registro. Essas chaves foram criadas com o método *KeyPairGenerator*, provido pela biblioteca de segurança da linguagem Java, utilizando o método DSA. O registro do provedor é realizado sobre uma conexão SSL. Desta forma é garantido que a chave pública não é capturada com ataques, como *man-in-the-middle* e análise de tráfego, durante sua transferência das mensagens.

As contas e senhas são empregadas para autenticar provedores e clientes junto ao *proxy*. Por exemplo, no registro de um provedor o *proxy* verifica se o registro pode ser feito comparando a conta e a senha informada pelo provedor com um conjunto de contas e senhas existentes na tabela de usuários do banco de dados. Se a comparação for efetuada com sucesso, o provedor é registrado.

Os anúncios feitos por provedores são autenticados através da verificação da senha do provedor (que é diferente da senha utilizada para autenticar o registro) cadastrada durante o registro do provedor. Além disso, os anúncios são cifrados e para decifrá-los o *proxy* utiliza a chave pública do provedor que fez o anúncio. Para melhorar a criptografia dos anúncios e aumentar a possibilidade de serem feitos via UDP é utilizado um método de compactação provido pela linguagem Java. Foi verificado, entretanto, que a maioria dos anúncios de provedores quase não foram feitos utilizando UDP. Isto é justificado devido ao uso de OWL, que requer muitas informações para descrever o serviço.

O *leasing* é também utilizado para garantir que nenhum anúncio antigo seja aceito. Isso é feito através da verificação da hora de envio contida na mensagem de anúncio e o emprego da inequação definida em 5.2.4. Para que o cálculo pudesse ser feito sem inconsistências, foi usado um serviço de sincronização de relógios de computadores, muito utilizado na *Internet*, o *Simple Network Time Protocol (NTP)* [Mills 1996]. O emprego do NTP, entretanto, não é uma abordagem que deva ser empregada como padrão, pois esse serviço pode não estar disponível em redes isoladas. Para contornar isso, os relógios dos provedores de serviço podem ser sincronizados com o do *proxy* no momento do registro.

7.3 Cenário de Testes

O protocolo foi testado utilizando os sistemas operacionais *Windows XP* e *Linux SuSE 9.1*, o gerenciador de banco de dados *PostgreSql 8.1.2* [PostgreSql 2005] e a ferramenta *Ethereal* [Combs 2005] para capturar os pacotes do protocolo. A infra-estrutura computacional utilizada foi a do Laboratório de Pesquisas em Redes e Multimídia do DI/UFES, que possui uma rede *FastEthernet* e uma conexão de 10 *Mbps* com o *campus* da UFES. Foram utilizados seis computadores, sendo um hospedando o *proxy*, um outro o SGBD, três os provedores de serviço e um outro o Cliente, conforme é mostrado na Figura 7.4.

Uma aplicação foi desenvolvida para a realização de testes. Nessa aplicação é utilizado um cenário da área da saúde, elaborado para exemplificar uma situação em que um paciente deseja encontrar serviços de consulta e de diagnóstico. Na tela inicial da aplicação, ilustrada pela Figura 7.5, o usuário seleciona o tipo de serviço desejado: de consulta, de diagnóstico, ou de medição climática.

Inicialmente os testes de campo seriam feitos a partir de informações sobre os sensores utilizados nos sítios de pesquisa do projeto LBA, Porém, devido a dificuldade ao acesso a esses sensores e por não possuírem interfaces de conexão de rede, foi utilizado um cenário hospitalar. Este cenário possui semelhanças com o dos sítios de pesquisa no

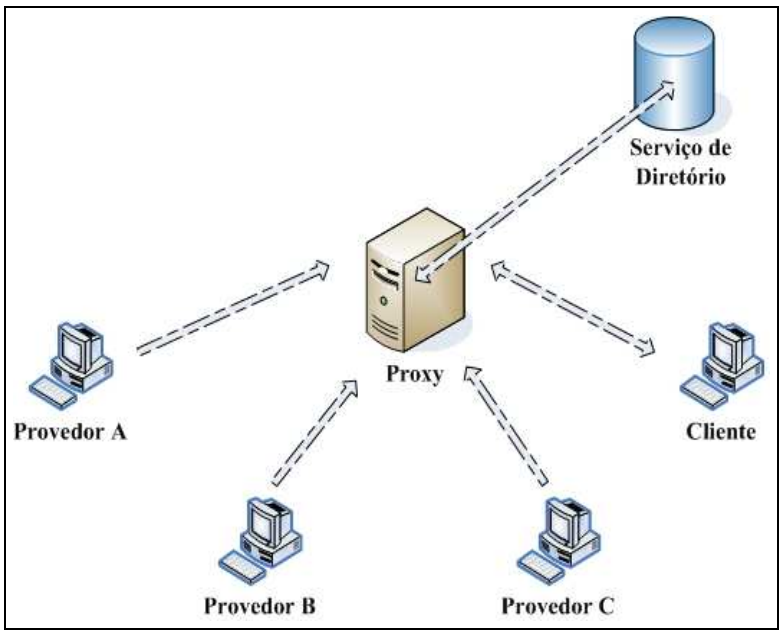

Figura 7.4: Cenário de testes.

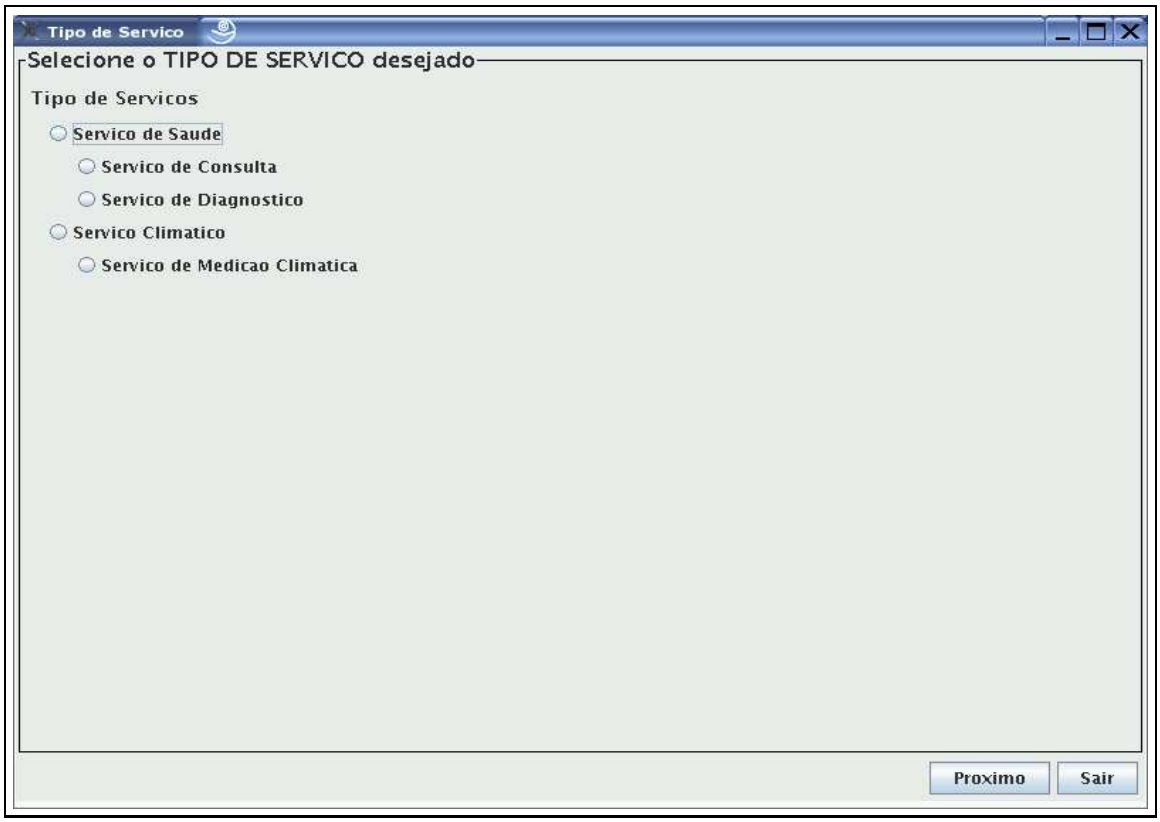

Figura 7.5: Tela inicial da aplicação cliente.

que diz respeito à característica dinâmica das informações tratadas e à mobilidade dos dispositivos.

Um dos testes realizados foi a solicitação de descoberta de um serviço de diagnóstico por ultrassonografia. A Figura 7.6 ilustra a seleção da saída do serviço e das entradas requisitadas (plano de saúde e cidade do atendimento).

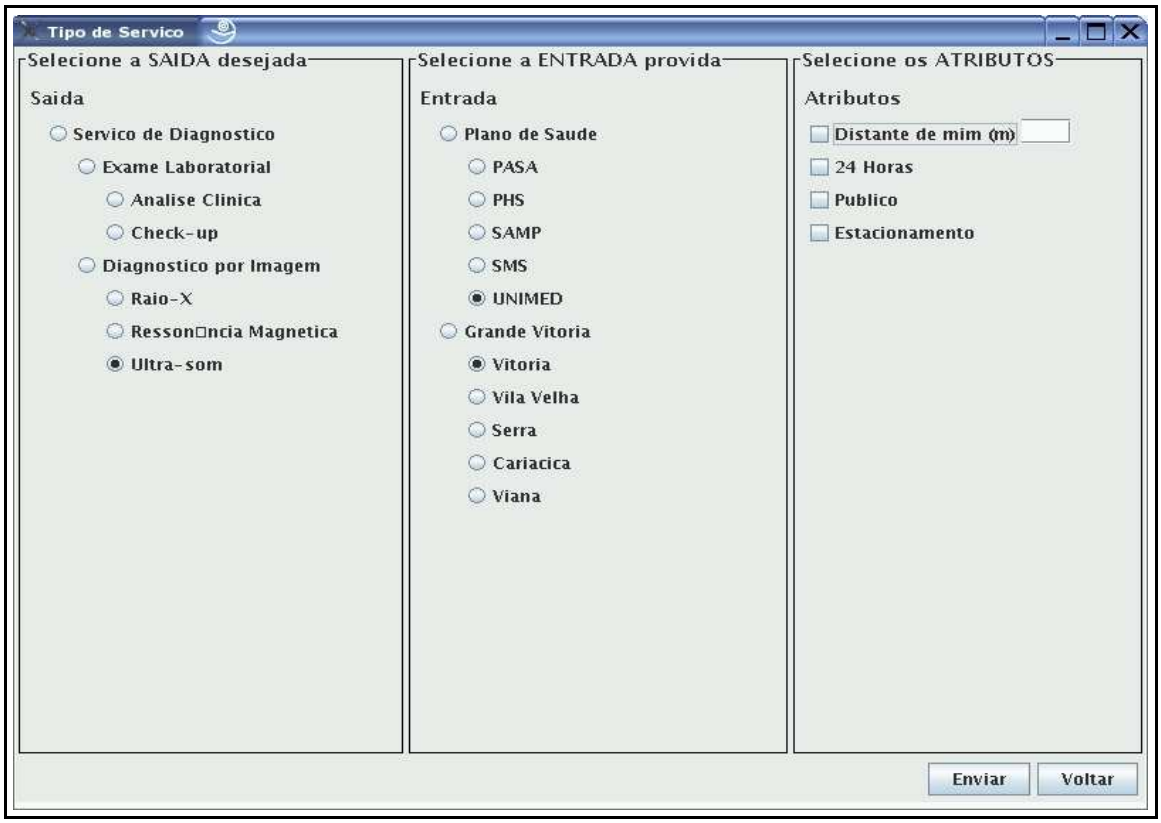

Figura 7.6: Requisição de descoberta de Serviço de Diagnóstico.

O resultado da seleção feita pelo *proxy*, ilustra pela Figura 7.7, não encontrou serviços exatos, ou seja, cujo plano de saúde fosse da UNIMED e que o atendimento fosse em Vitória.

Outro teste feito foi a solicitação de descoberta de serviços de exame laboratorial, dos planos de saúde conhecidos, com atendimento público e de 24 horas na Grande Vitória, conforme é mostrado na Figura 7.8. O resultado da descoberta retornada pelo *proxy*, ilustrada pela Figura 7.9, listou vários serviços exatos encontrados.

Para testar o protocolo foram utilizadas um conjunto de descrições de serviço definidas a partir de informações obtidas de uma lista telefônica da Grande Vitória. Os exemplos construídos foram de ambulatórios, consultórios, hospitais, clínicas e laboratórios.

Foi verificado que, na primeira requisição de descoberta processada pelo *proxy*, há uma demora no acesso a outras ontologias espalhadas pela *Internet* e que são utilizadas na seleção. Entretanto, esse tempo diminui para as requisições posteriores porque, após a primeira requisição de descoberta, essas ontologias são localmente armazenadas.

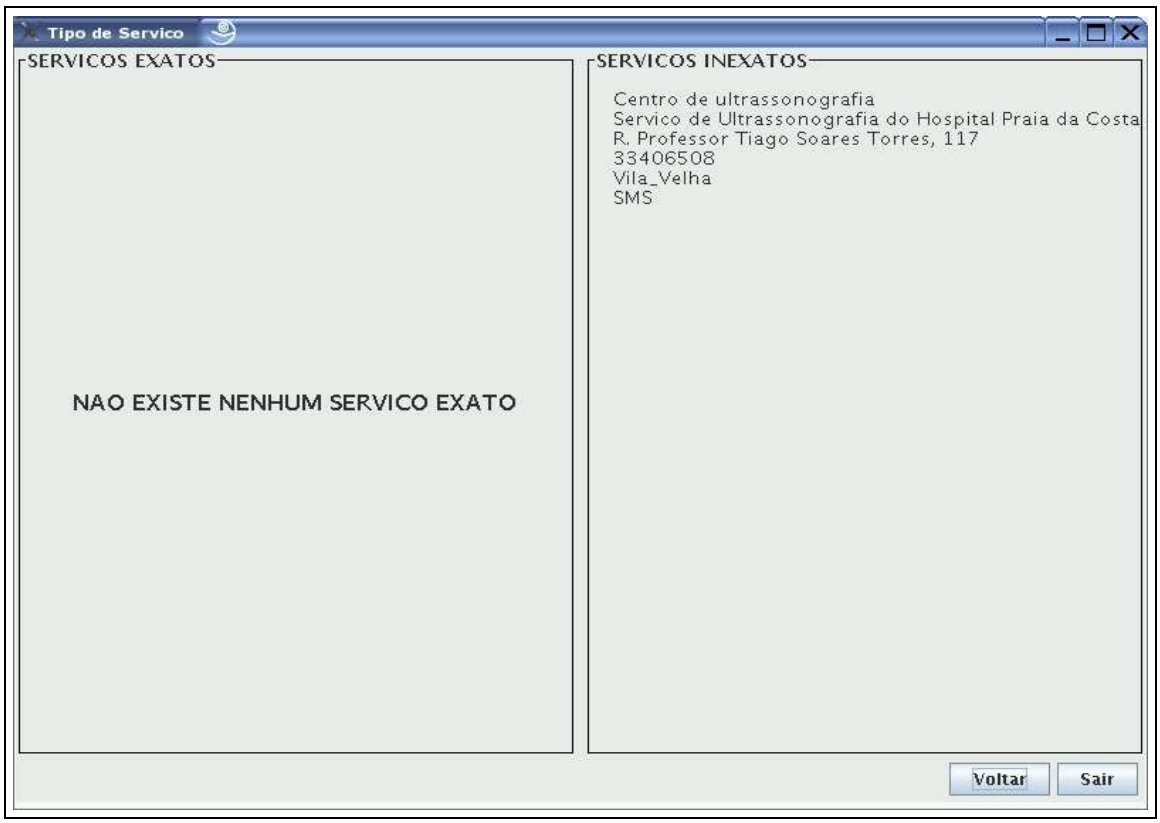

Figura 7.7: Resultado da descoberta de um Serviço de Diagnóstico.

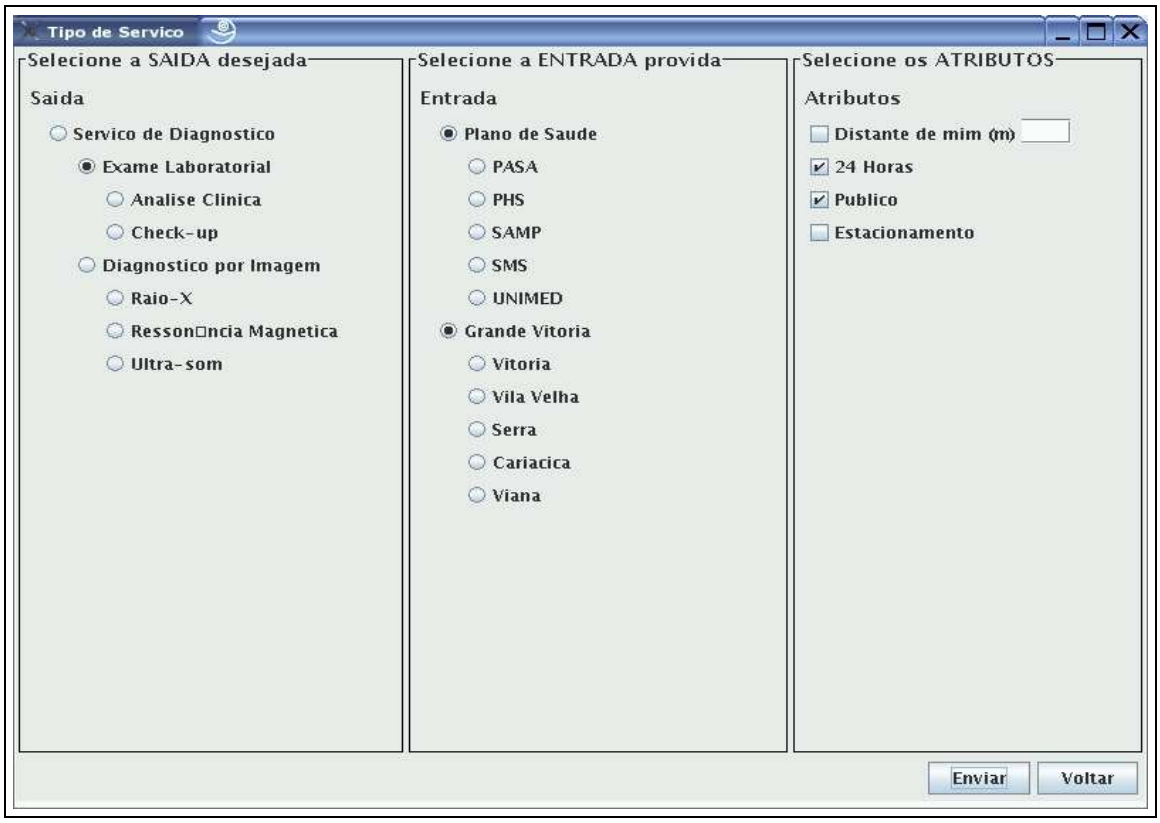

Figura 7.8: Requisição de descoberta de serviço de Exame Laboratorial.

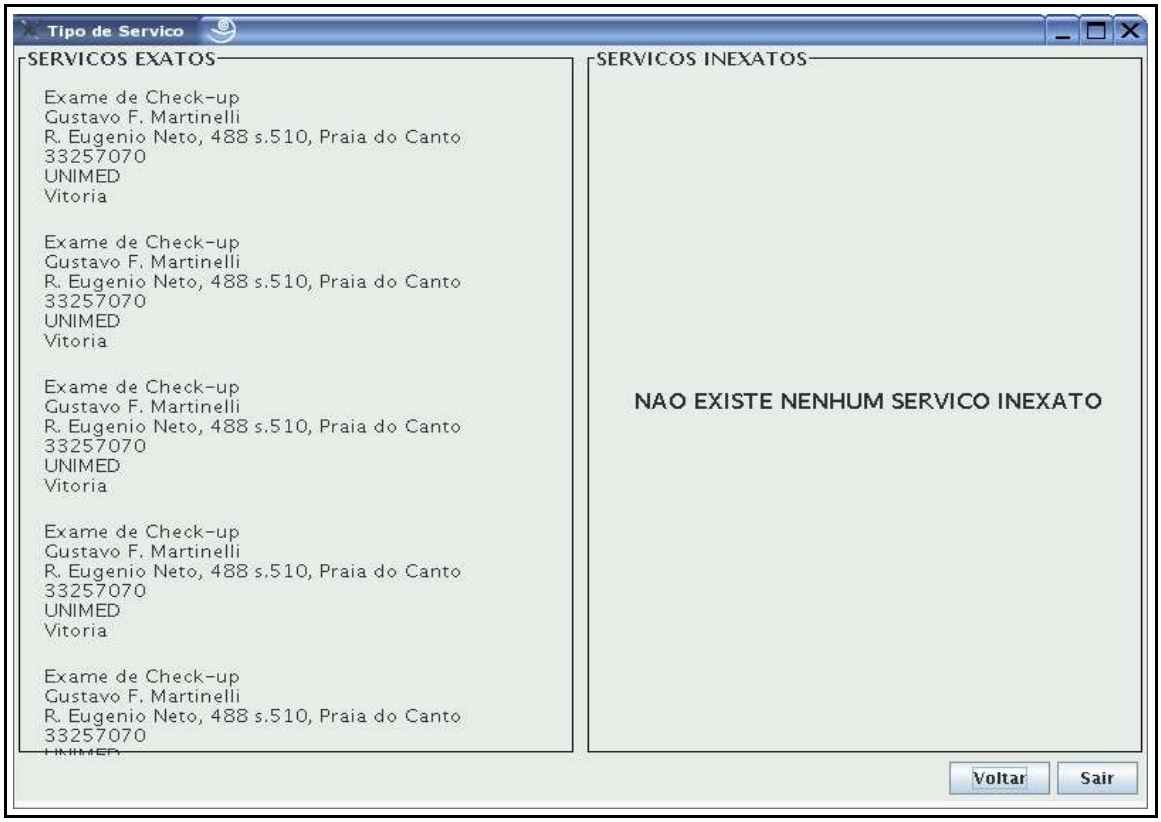

Figura 7.9: Resultado da descoberta de serviço de Exame Laboratorial.

7.4 Conclusão

Na implementação do protótipo do SCaSDP foram comtemplados todos os requisitos, exceto a comunicação entre vários *proxies*. As descrições de serviços utilizadas foram construídas a partir de informações sobre hospitais, ambulatórios, clínicas, laboratórios e consultórios médicos.

A utilização do algoritmo de comparação permitiu aumentar a precisão da descoberta de serviço devido às inferências semânticas realizadas, apenas possível devido a utilização da OWL para descrever tais serviços.

As ontologias FIPA de dispositivos e de informações contextuais básicas possibilitaram fazer do SCaSDP um protocolo sensível ao contexto, pois todas as mensagens enviadas pelos provedores de serviço, pelo *proxy* e pelos clientes continham informações referentes ao contexto no qual essas entidades estavam inseridas.

Os testes realizados indicaram que para a primeira requisição de descoberta houve uma demora para acessar as várias ontologias utilizadas e que não estavam armazenadas localmente. Para as requisições posteriores a resposta do *proxy* aos clientes foi bem mais rápida.

Foi verificado também que as abordagens utilizadas, como *leasing*, compactação das descrições e chaves de criptografia, permitiram garantir a integridade, confidencialidade e não-repúdio das informações trocadas entre os dispositivos, fazendo assim do SCaSDP um protocolo seguro.

Capítulo 8

Conclusão

8.1 Considerações Finais

A Computação Ubíqua vem ganhando espaço em relação ao paradigma computacional tradicional. Esse novo paradigma traz consigo a possibilidade de explorar o ambiente criado pelo crescente uso de dispositivos móveis multi-funcionais, como PDAs e aparelhos de telefonia celular. Nesse contexto, um novo conjunto de aplicações, definidas como *Aplicações Sensíveis ao Contexto*, são identificadas e permitem utilizar no processamento de tarefas não apenas informações providas por usuários, mas também as informações contextuais, obtidas do ambiente, de maneira a influenciar na tomada de decisões dessas aplicações.

O desenvolvimento de aplicações sensíveis ao contexto requer uma infra-estrutura de suporte preparada para lidar com a característica dinâmica da informação contextual. A *Infraware*, em desenvolvimento no Laboratório de Pesquisas em Redes e Multimídia da UFES, é uma proposta de suporte às aplicações sensíveis ao contexto. Essa plataforma se utiliza de conceitos e tecnologias relacionados com a Web Semântica e se propõe a tratar os problemas inerentes à integração de dados contextuais heterogêneos e distribuídos, à resolução de conflitos e coordenação entre aplicações, à privacidade, à segurança e à gerência integrada de serviços semânticos.

Em um ambiente onde a sensibilidade ao contexto é considerada, a gerência integrada de serviços traz consigo alguns desafios que são agravados pela natureza contextual da informação processada por esses serviços. Uma dessas questões refere-se a descoberta de serviços, requisito fundamental para o gerenciamento dinâmico dos serviços disponíveis providos pelos diversos dispositivos computacionais presentes.

Na literatura há várias propostas que tratam a descoberta de serviços para os sistemas tradicionais. Entretanto, para a computação sensível ao contexto essas propostas deixam a desejar, principalmente no que se refere à natureza da informação contextual e à utilização de descrições de serviços mais ricas em semântica, como descrições feitas com base em ontologias e com uma linguagem como OWL. Outras questões, que conseqüentemente necessitam ser melhor tratadas, são as relacionadas a organização, a classificação e a seleção de serviços sensíveis ao contexto. O SCaSDP, desenvolvido neste trabalho, se propõe a tratar essas questões.

O SCaSDP considera a informação contextual e sua natureza, permitindo uma descoberta dinâmica de serviços. Utiliza ontologias de tarefas para descrever semanticamente os serviços providos pelos dispositivos computacionais. Além disso, garante a segurança

desses dispositivos e serviços através da utilização de mecanismos de criptografia e autenticação.

O protótipo desenvolvido permitiu verificar as propriedades do SCaSDP. As descrições de serviços foram feitas com base na ontologia de serviços OWL-S, na de dispositivos FIPA e nas informações contextuais básicas. A linguagem de marcação OWL, aliada a um algoritmo de comparação semântica, permitiu dar maior precisão a descoberta. Além disso, a utilização do *leasing*, da compactação de mensagens do protocolo e de chaves de criptografia permitiram dar ao protocolo respectivamente, dinamismo, rapidez e segurança.

As informações de localização e identificação, presentes na mensagem do protocolo, merecem destaque, pois permitem que sistemas de localização de dispositivos façam uso das funcionalidades do protocolo. Um exemplo disso são os sistemas de rastreamento que consideram apenas essas informações.

Um grande problema encontrado durante o desenvolvimento foi a falta de um serviço de diretório que suportasse descrições semânticas, principalmente as feitas com OWL. Para contornar isso foi utilizado o SGBD relacional *PostgreSQL*, que atendeu ao propósito do protótipo, mas que não é a solução adequada.

Uma dificuldade verificada foi a impossibilidade de testar o protocolo em um ambiente real, visto que não existia uma infra-estrutura adequada, com vários dispositivos móveis, sensores e serviços reais. Os testes realizados utilizaram descrições de serviços tiradas de uma lista telefônica, particularmente foram usados os serviços prestados por hospitais, ambulatórios, clínicas, laboratórios e consultórios médicos.

O SCaSDP é uma proposta genérica e pode ser utilizado nos mais diferentes domínios como em universidades, no comércio, em sítios de pesquisa, como os mantidos pelo LBA na região amazônica.

8.2 Trabalhos Futuros

Por ter sido desenvolvido utilizando a linguagem Java, o protótipo é portável para outros sistemas. Entretanto, é possível que seja necessário fazer alterações para que possa ser utilizado em dispositivos com recursos mais restritos, como sensores, aparelhos de telefonia celular e alguns PDAs. Isto se deve ao fato da versão da linguagem Java para esses dispositivos, a *J2ME*, possuir restrições no que diz respeito aos métodos e classes suportados pela versão utilizada neste trabalho.

Por ser uma linguagem interpretada, Java tem a característica de consumir bastante dos recursos computacionais. São necessários mais testes, principalmente de carga, para avaliar o desempenho e verificar a viabilidade do protótipo implementado com essa linguagem em uma situação real.

O protocolo, para ser utilizado em um ambiente real, necessita de uma infra-estrutura composta de dispositivos móveis, servidores dedicados para abrigarem o *proxy*, um meio de comunicação sem fio e conexão com a *Internet*. Uma infra-estrutura desse porte é completamente viável atualmente, com um custo relativamente baixo, tendo em vista que os dispositivos móveis e os pontos de acesso sem fio estão cada vez mais baratos.

A versão atual suporta criptografia de chave pública, mas é desejável que a criptografia de chave privada seja também suportada. Esta expansão permitirá um estudo mais detalhado sobre as vantagens e as desvantagens de cada método em diferentes cenários.

A autenticação suportada utiliza apenas conta, senha e chave pública de criptografia. Uma abordagem mais complexa como a definição de grupos de usuários, perfis, ou mesmo de restrições de acesso a determinados grupos contribuirão para um melhor controle de acesso às informações e serviços disponíveis.

No que diz respeito aos certificados, é necessário definir uma política de distribuição e geração automática para prover ao protocolo uma menor dependência da interação humana.

O protótipo atual permite que provedores de serviço anunciem apenas um único serviço, o que é limitante. O suporte ao anúncio de vários serviços permitirá um melhor aproveitamento dos dispositivos que os provêem.

Outra questão para estudo é a utilização de um serviço de diretório que suporte descrições semânticas. Os existentes dão suporte no máximo a descrições feitas em XML, o que não garante semântica na descrição. É interessante pesquisar a utilização de soluções de serviços de diretório existentes para verificar a possibilidade da utilização deles em conjunto com OWL.

O algoritmo de comparação utilizado atendeu às expectativas no que se refere aos testes realizados em laboratório. Entretanto, algumas melhorias, como atribuição de pesos aos atributos dos serviços e a elaboração de ontologias mais complexas permitirão, respectivamente, identificar a prioridade dos serviços para os usuários e descrever com mais precisão o contexto do ambiente.

Todas essas melhorias favorecerão em muito a utilização do SCaSDP em ambientes diferentes do apresentado neste trabalho, além do que, contribuirão bastante para o desenvolvimento de novos estudos.

Referências Bibliográficas

[Abowd e Mynatt 2000] ABOWD, G. D.; MYNATT, E. D. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computing Human Interaction*, v. 7, n. 1, p. 29–58, 2000. Disponível em: <http://doi.acm.org/10.1145/344949.344988>.

[Agents 2002] AGENTS, F. for I. P. *FIPA Device Ontology Specification*. [S.l.], Maio 2002. Acessado em 18.11.2005. Disponível em: <http://www.fipa.org/>.

[Albitz e Liu 1998] ALBITZ, P.; LIU, C. *DNS and Bind*. 3^a. ed. [S.l.]: O'Reilly, 1998.

[Almenárez e Campo 2003] ALMENÁREZ, F.; CAMPO, C. SPDP: A secure service discovery protocol for ad-hoc networks. In: DEPT. TELEMATIC ENGINEERING, UNIVERSITY CARLOS III OF MADRID. *9^{th} EUNICE - Open European Summer School and IFIP Workshop on Next Generation Networks*. Avda. Universidad 30, 28911 Leganés (Madrid), Spain, 2003. Disponível em: <http://www.it.uc3m.es/celeste/papers/eunice2003_spdp.pdf>.

[Barbosa, Porto e Melo 2002] BARBOSA, A. C. P.; PORTO, F. A.; MELO, R. N. Configurable data integration middleware system. *Journal of the Brazilian Computer Society*, v. 8, n. 2, p. 12–19, 2002. Disponível em: <http://codims.lprm.inf.ufes.br/publicacoes.html>.

[Bloom et al. 2004] BLOOM, J. D. et al. *Platform Independent Petri-Net Editor 2*. [S.l.], 2004. Acessado em 05.12.2005. Disponível em: <http://pipe2.sourceforge.net/>.

[Brewington et al. 1999] BREWINGTON, B. et al. Mobile agents in distributed information retrieval. In: KLUSCH, M. (Ed.). *Intelligent Information Agents*. Hanover, New Hampshire 03755, 1999. cap. 15, p. 355–395. Disponível em: <http://www.cs.dartmouth.edu/~dfk/papers/brewindton:ir.pdf>.

[Broens 2004] BROENS, T. *Context-aware, Ontology based, Semantic Service Discovery*. Dissertação (Mestrado) — University of Twente, July 2004.

[Buccella, Penabad e Rodriguez 2004] BUCCELLA, A.; PENABAD, M. R.; RODRIGUEZ, F. J. From relational databases to owl ontologies. In: INSTITUTE OF MATHEMATICAL PROBLEMS OF BIOLOGY RAS. *Proceedings of Sixth National Russian Research Conference*. Rússia, 2004. Disponível em: <http://www.impb.ru/ rcdl2004/cgi/get_paper_pdf.cgi?pid=12>.

[Bulusu] BULUSU, N. Operating system support for context-aware computing. In: DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF SOUTHERN CALIFORNIA. Acesso em 06.03.2005. Disponível em: <http://citeseer.ist.psu.edu/461332.html>.

[Cardoso e Valette 1997] CARDOSO, J.; VALETTE, R. *Redes de Petri*. 1ª. ed. [S.l.]: Ed. da UFSC, 1997.

[Carrara e Guarino 1999] CARRARA, M.; GUARINO, N. Formal ontology and conceptual analysis: a structured bibliography. In: LADSEB-CNR (Ed.). Padova, Itália: [s.n.], 1999.

[Cha et al. 2003] CHA, H.-W. et al. Implementation of service location protocol and remote device control for IPv6 based home networking. In: PROTOCOL ENGINEERING CENTER. *ICACT 2003*. ETRI, 161 Gajong-Don, Yusong-Gu, Daejeon, Korea, 2003.

[Chen e Kotz 2000] CHEN, G.; KOTZ, D. *A Survey of Context-Aware Mobile Computing Research*. [S.l.], Novembro 2000. Disponível em: <http://citeseer.ist.psu.edu/chen00survey.html>.

[Chen e Kotz 2002] CHEN, G.; KOTZ, D. *SOLAR: A Pervasive-Computing Infrastructure for Context-Aware Mobile Applications*. [S.l.], 2002. Technical Report TR2002-421.

[Chen e Kotz 2002] CHEN, G.; KOTZ, D. Solar: An open platform for context-aware mobile applications. In: *First Internatinal Conference on Pervasive Computing*. Hanover, NH, USA 03755: [s.n.], 2002. p. 41–47. Disponível em: <http://www.cs.dartmouth.edu/˜dfk/ papers/chen:pervasive.pdf>.

[Chen 2004] CHEN, H. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Tese (Doutorado) — University of Maryland, USA, 2004.

[Combs 2005] COMBS, G. Ethereal - network protocol analyzer. Version 0.10.12. 2005. Disponível em: <http://www.ethereal.com/>.

[Costa 2005] COSTA, A. C. M. *AlComp - Um Algoritmo de Comparação para Descobera de Serviço*. [S.l.], Dezembro 2005. Monografia de Graduação - UFES.

[Costa 2003] COSTA, P. D. *Towards a Service Platform for Context-Aware Applications*. Dissertação (Mestrado) — University of Twente, Agosto 2003. Disponível em: <http://www.cs.utwente.nl/˜dockhorn/files/thesis_dockhorn.pdf>.

[Czerwinski et al. 1999] CZERWINSKI, S. E. et al. An architecture for a secure service discovery service. In: COMPUTER SCIENCE DIVISION, UNIVERSITY OF CALIFORNIA, BERKELEY. *ACM/IEEE Fifth Annual International Conference on Mobile Computing and Networking*. ACM Press, 1999. p. 24–35. ISBN:1-58113-142-9. Disponível em: <http://citeseer.ist.psu.edu/czerwinski99architecture.html>.

[Dey 2000] DEY, A. K. *Providing Architectural Support for Building Context-Aware Applications*. Tese (Doutorado) — Georgia Institute of Technology, 2000.

[Dey. e Abowd 1999] DEY., A. K.; ABOWD, G. D. Towards a better understanding of context and context-awareness. In: GRAPHICS, VISUALIZATION AND USABILITY CENTER AND COLLEGE OF COMPUTING, GEORGIA INSTITUTE OF TECHNOLOGY. *1st International Symposium on Handheld and Ubiquitous Computing (HUC'99)*. Atlanta, GA, USA 30332-0280, 1999. Technical Report GIT-GVU-99-22.

[Filho et al. 2005] FILHO, J. G. P. et al. *Infraware: Uma Plataforma para Desenvolvimento de Aplicações Móveis e Sensíveis ao Contexto*. [S.l.], 2005. Projeto FAPES.

[Filho et al. 2005] FILHO, J. G. P. et al. [S.l.], 2005. Projeto CNPq.

[Forum 2003] FORUM, U. *UPnP^{TM} Device Architecture 1.0 - Versão 1.0.1*. [S.l.], Dezembro 2003. Disponível em: <http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf>.

[Friedrich, Shimkat e Küchlin 2002] FRIEDRICH, M.; SHIMKAT, R.-D.; KüCHLIN, W. Information retrieval in distributed environment based on context-aware, proactive document. In: WILHELM-SCHICKARD-INSTITUTE FOR COMPUTER SCIENCE, UNIVERSITY OF TüBINGEN. Sand 13, D-72076 Tübingen, Germany, 2002.

[Goland et al. 1999] GOLAND, Y. Y. et al. *Simple Service Discovery Protocol/1.0. Operating without an Arbiter*. [S.l.], Outubro 1999. Disponível em: <http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt>.

[Gruber 1993] GRUBER, T. R. A translation approach to portable ontology specifications. In: *Knowledge Acquisition*. [S.l.: s.n.], 1993. v. 5, n. 2, p. 199–220.

[Gruber 1995] GRUBER, T. R. Towards principles for the design of ontologies used for knowledge sharing. In: GUARINO, N.; POLI, R. (Ed.). *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Deventer, The Netherlands: Kluwer Academic Publishers, 1995. Disponível em: <citeseer.ist.psu.edu/gruber93toward.html>.

[Gu et al. 2003] GU, T. et al. An architecture for flexible service discovery in OCTOPUS. In: CENTER FOR INTERNET RESEARCH, DEPARTMENT OF COMPUTER SCIENCE NATIONAL UNIVERSITY OF SINGAPORE. *12th International Conference on Computer Communications and Networks*. [S.l.], 2003. p. 291–296.

[Gu et al. 2004] GU, T. et al. A middleware for building context-aware mobile services. *IEEE Vehicular Technology Conference*, Maio 2004.

[Guarino 1998] GUARINO, N. Formal ontology and information systems. In: NATIONAL RESEARCH COUNCIL, LADSEB-CNR. [S.l.], 1998. (FOIS), p. 3–15.

[Guha 1991] GUHA, R. *Contexts: A Formalization and Some Applications*. Tese (Doutorado) — Stanford University, 1991.

[Guizzardi 2000] GUIZZARDI, G. *Desenvolvimento para e com reuso: Um estudo de caso no domínio de Vídeo sob Demanda*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2000.

[Guttman 1999] GUTTMAN, E. Service location protocol: Automatic discovery of IP network services. *IEEE Internet Computing*, IEEE Educational Activities Department, v. 3, n. 4, p. 71–80, Julho - Agosto 1999.

[Guttman et al. 1999] GUTTMAN, E. et al. *Service Location Protocol. Version 2*. [S.l.], June 1999. Disponível em: <http://www.openslp.org/doc/rfc/rfc2608.txt>.

[Hao e Selker 2000] HAO, Y.; SELKER, T. Context-aware office assistant. In: *Inteligent User Interfaces*. 20 Ames St., E15-320N - Cambridge, MA 02139 USA: [s.n.], 2000. p. 276–279.

[Henricksen, Indulska e Rakotonirainy 2002] HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Modeling context information in pervasive computing systems. *Springer-Verlag Berlin Heidelberg*, p. 167–180, 2002.

[Holzmann 1991] HOLZMANN, G. *Design and validation of computer protocols*. [S.l.]: Prentice Hall, 1991.

[Huang 2002] HUANG, Q. Supporting context-aware computing in ad hoc mobile environment. In: . Saint Louis, MO 63130: Department of Computer Science and Engineering, Washington University, 2002.

[Indulska et al. 2001] INDULSKA, J. et al. An open archicteture for pervasive systems. In: CRC FOR DISTRIBUTED SYSTEMS TECHNOLOGY/UNIVERSITY OF QUEENSLAND, SCHOOL OF COMPUTER SCIENCE AND ELECTRICAL ENGINEERING - MONASH UNIVERSITY. *Proceedings of the 3rd. International Working Conference on Distributed Applications an Interoperable Systems (DAIS)*. Kraków, Poland, 2001. p. 175–188.

[Informatics 2005] INFORMATICS, S. M. 2005. Version 3.1.1, build 216. Disponível em: <http://protege.stanford.edu/>.

[Informatics 2005] INFORMATICS, S. M. 2005. Acessado em 20.01.2006. Disponível em: <http://protege.stanford.edu/plugins/owl/>.

[INPA 2006] INPA. Instituto nacional de pesquisas da amazônia. 2006. Disponível em: <http://www.inpa.gov.br/>.

[Jini 2003] JINI. *Jini Introduction*. [S.l.], 01 de Setembro 2003. Fakultet for informasjonsteknologi, matematikk og elekroteknikk. TTM47AC Laboratory in construction of self-configuring systems. Disponível em: <http://www.item.ntnu.no/fag/ttm47ac/Information/Jini_Introduction.pdf>.

[Johner et al. 1999] JOHNER, H. et al. *LDAP Implementation Cookbook*. Dept. JN9B, Building 003 Internal Zip 2834, 11400 Burnet Road - Austin, Texas, Junho 1999. Disponível em: <http://borg.ucsb.edu/aka/Ucdir/sg245110.pdf>.

[K.Dey et al. 2001] K.DEY, A. et al. Situated interaction and context-aware computing. *Personal and Ubiquitous Computing*, v. 5, p. 1–3, 2001.

[King-Lynne 2005] KING-LYNNE, C. phppgadmin: Web based postgresql administratin tool. Versão 4.0.1. Novembro 2005. Disponível em: <http://phppgadmin.sourceforge.net/>.

[LBA 2006] LBA. Experimento de grande escala da biosfera-atmosfera da amazônia. 2006. Disponível em: <http://lba.inpa.gov.br/>.

[Lei et al. 2002] LEI, B. et al. The design and applications of a context service. In: IBM T. J. WATSON RESEARCH CENTER. *Mobile Computing and Communications Review*. Hawthorne, NY 10532, USA, 2002. v. 6, n. 4, p. 45–55.

[Li e Horroks 2003] LI, L.; HORROKS, I. A software framework for matchmaking based on semantic web technology. In: DEPARTMENT OF COMPUTER SCIENCE - UNIVERSITY OF MANCHESTER. *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*. Manchester, United Kingdom, 2003. p. 331–339.

[Livingstone 2003] LIVINGSTONE, S. R. *Service Discovery in Pervasive System*. Tese (Doutorado) — The School of Information Technology and Electrical Engineering - The University of Queensland, Outubro 2003.

[Marti e Krishnan 2002] MARTI, S.; KRISHNAN, V. *Carmen: A Dynamic Service Discovery Architecture*. Mobile and Media Systems Laboratory, HP Laboratories Palo Alto, 16 de Setembro 2002. Disponível em: <http://www.hpl.hp.com/techreports/2002/HPL-2002-257.pdf>.

[Martin et al. 2005] MARTIN, D. et al. *OWL-S: Semantic Markup for Web Services*. 2005. Disponível em: <http://www.daml.org/services/owl-s/1.1/overview/>.

[McCarthy e Buvac 1994] MCCARTHY, J.; BUVAC, S. *Formalizing Context (Extended Notes)*. [S.l.], 1994. CS-TN-94-13. Disponível em: <http://portal.acm.org/citation.cfm?id=891892&dl=GUIDE&coll=GUIDE>.

[McGrath 2000] MCGRATH, R. E. *Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing*. Champaign, IL, USA, 2000. Disponível em: <http://historical.ncstrl.org/tr/fulltext/tr/uiuc_cs/UIUCDCS-R-2000-2154.txt>.

[Microsystems 2006] MICROSYSTEMS, S. *NetBeans*. 2006. Disponível em: <http://www.netbeans.org/>.

[Miller 1999] MILLER, B. *Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer*. [S.l.], 01 de Julho 1999. IBM. Disponível em: <http://www.waterwood.com.cn/technology/bluetooth-documents/API2SDP.PDF>.

[Mills 1996] MILLS, D. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI - RFC 2030*. [S.l.], Outubro 1996. Acessado em 04.11.2005. Disponível em: <http://www.faqs.org/rfcs/rfc2030.html>.

[Murata 1989] MURATA, T. Petri nets: Properties, analysis and applications. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1989. v. 77, n. 4, p. 541–580.

[Parmar 2005] PARMAR, S. K. *An Introduction to Security*. 6060 Canada Ave., Duncan, BC 250-748-5522, 2005. Information Resouce Guide: Computer, Internet and Network Systems Security. Disponível em: <http://all.net/books/document/secman/index.html>.

[Pascoe 1999] PASCOE, B. *Salutation Architectures and the newly defined service discovery protocols from Microsoft and Sun: How does the Salutation Architecture stack up*. [S.l.], 6 de Junho 1999.

[Pascoe 1998] PASCOE, R. A. *Salutation Architecture: Enabling Applications and Services*. [S.l.], 19 de Agosto 1998. The Salutation Consortium. Senior Technical Staff Consulting.

[PostgreSql 2005] POSTGRESQL. Version 8.1.2. 2005. Disponível em: <http://www.postgresql.com/>.

[Programme 2006] PROGRAMME, H. L. S. W. Jena - a semantic web framework for java. 2006. Disponível em: <http://jena.sourceforge.net>.

[Rusnak 1999] RUSNAK, J. *Discovering Devices and Services In Home Networks*. [S.l.], 1999. IBM.

[Santos 2004] SANTOS, L. O. B. da S. *Semantic Services Support for Context-aware Platforms*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2004.

[Schilit, Adams e Want 1994] SCHILIT, B. N.; ADAMS, N.; WANT, R. Context-aware computing applications. In: *IEEE Workshop on Mobile Computing Systems and Applications*. Santa Cruz, California: IEEE Computer Society Press., 1994. p. 85–90.

[Schilit e Theimer 1994] SCHILIT, B. N.; THEIMER, M. M. Disseminating active map information to mobile hosts. *IEEE Network*, v. 8(5), p. 22–32, 1994.

[Stallings 1995] STALLINGS, W. *Network and Internetwork Security: principles and practice*. United States: Prentice-Hall, Inc., 1995.

[SuSE Linux 2006] SUSE Linux. 2006. Disponível em: <http://www.suse.com/>.

[Systems 2005] SYSTEMS, I. C. *An Introduction to the Service Location Protocol (SLP)*. [S.l.], 2005. Disponível em: <http://www.openslp.org/doc/html/IntroductionToSLP/index.html>.

[UDDI 2000] UDDI. *UDDI Technical White Paper*. [S.l.], 06 de Setembro 2000. Accentrure, Arabia Inc and Commerce One, Inc. and Fujitsu Limited and Hewllet-Packard Company and i2 Technologies, Inc. and Intel Corporation and International Business Machines Corporation and Microsoft Corporation and Oracle Corporation and SAP AG and Sun Microsystems, Inc. amd VeriSign, Inc.

[W3C 2004] W3C. *Extensible Markup Language (XML)*. 2004. Disponível em: <http://www.w3.org/XML/>.

[W3C 2004] W3C. *RDF Vocabulary Description Language 1.0: RDF Schema*. Fevereiro 2004. Disponível em: <http://www.w3.org/TR/rdf-schema/>.

[WASP Project 2003] WASP Project. 2003. Disponível em: <http://www.freeband.nl/projecten/wasp/ENindex.html>.

[Weiser 1991] WEISER, M. The computer for the 21^{st} century. *Scientific American*, n. 265, p. 94–104, September 1991.

[Wikipedia 2005] WIKIPEDIA. Directory service. Acessado em 01.10.2005. Outubro 2005. Disponível em: <http://en.wikipedia.org/wiki/Directory_service>.

[Zhu, Mutka e Ni 2002] ZHU, F.; MUTKA, M.; NI, L. *Classification of Service Discovery in Pervasive Computing Environments*. [S.l.], 2002. Disponível em: <http://www.cse.msu.edu/~zhufeng/ServiceDiscoverySurvey.pdf>.

[Zhu, Mutka e Ni 2003] ZHU, F.; MUTKA, M. W.; NI, L. M. Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services. In: *Proceedings of IEEE International Conference on Pervasive Computing and Communications - PerCom*. [s.n.], 2003. p. 235–242. Disponível em: <http://www.cse.msu.edu/~zhufeng/splendor.pdf>.

[Zwicky, Cooper e Chapman 2000] ZWICKY, E. D.; COOPER, S.; CHAPMAN, D. B. *Building Internet Firewalls*. 2^a. ed. [S.l.]: O'Reilly & Associates, 2000.

Apêndice A

Experimento LBA

O Experimento de Grande Escala da Biosfera-Atmosfera da Amazônia [LBA 2006] é uma iniciativa de pesquisa internacional liderada pelo Brasil. O LBA está projetado para gerar novos conhecimentos para entender o funcionamento climatológico, ecológico, biogeoquímico e hidrológico da Amazônia, o impacto das mudanças no uso da terra nesses funcionamentos e as interações entre a Amazônia e o sistema biogeofísico global da terra.

O projeto atua em vários *campi* espalhados por toda a região amazônica e alguns outros estados do Brasil, como mostra a figura A.1. Esses *campi* possuem experimentos que se utilizam de sensores para fazer medições e coletas de dados relevantes às áreas de pesquisa do LBA.

Figura A.1: Localização dos *campi* do LBA.

Um dos *campi* de pesquisa é a base experimental conhecida como K34, localizada na Reserva Biológica do Cuieiras ($02^o36'32.666''S; 60^o12'33.482'W; 122.98m$), município de

Manaus, no estado do Amazonas. Esta Unidade de Conservação, localizada a 80 Km ao norte da cidade de Manaus, possui 22.735 *ha* de área de floresta tropical úmida, com uma altura de dossel entre 35 e 40 metros. O acesso é dado no Km 50 da BR-174 (Rodovia Manaus-Boa Vista) através da vicinal ZF2, chegando ao Km 34, conforme é ilustrado pela figura A.2. Esta reserva é administrada pelo Instituto Nacional de Pesquisas da Amazônia (INPA) [INPA 2006], do Ministério de Ciência e Tecnologia (MCT).

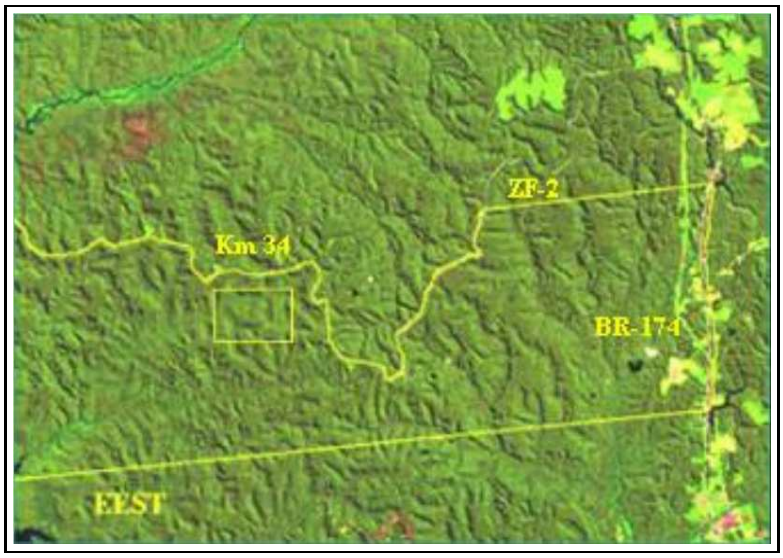

Figura A.2: Localização da base experimental K34.

Nesta reserva, por exemplo, existem vários grupos de pesquisa ligados ao LBA com experimentos referentes à fuga de carbono na atmosfera, à hidrologia e ao solo que se utilizam de sensores. Cada um desses experimentos utiliza sensores que podem ser vistos como serviços, ou seja, recursos que podem ser acessados por alguma aplicação computacional. Por exemplo, um sensor que mede temperatura na copa das árvores é visto como um serviço que informa a temperatura na copa das árvores; um outro que mede a umidade relativa do ar, é visto como um serviço que informa a umidade relativa do ar.

Os dados coletados pelos sensores são armazenados localmente em mídias de armazenagem, como cartões PCMCIA. É necessário que um técnico se dirija ao local onde o sensor está instalado para coletar esses dados e, só então, possa armazená-los em local seguro para que serem utilizados pelos pesquisadores.

Uma vez conectados a uma rede de computadores, esses sensores podem anunciar suas descrições. Assim, o SCaSDP pode ser utilizado para descobrir os sensores que são importantes para algum pesquisador. Além disso, os dados coletados podem ser imediatamente armazenados em uma base de dados para imediata utilização. As situações descritas a seguir são exemplos de como o SCaSDP pode ser utilizado no ambiente descrito acima.

Um pesquisador, localizado em Manaus, está pesquisando sobre as mudanças climáticas da região amazônica. Para isso necessita de informações sobre temperatura, umidade relativa do ar, velocidade do vento e fluxo de calor da região. O pesquisador definiu que sua pesquisa irá se concentrar no raio de abrangência da base experimental K34. O sistema que utiliza para monitorar as mudanças climáticas requer que os dados sejam freqüentemente atualizados. Para isso, os sensores são freqüentemente acessados.

Na K34 há vários sensores e nem todos são conhecidos pelo pesquisador, pois outros

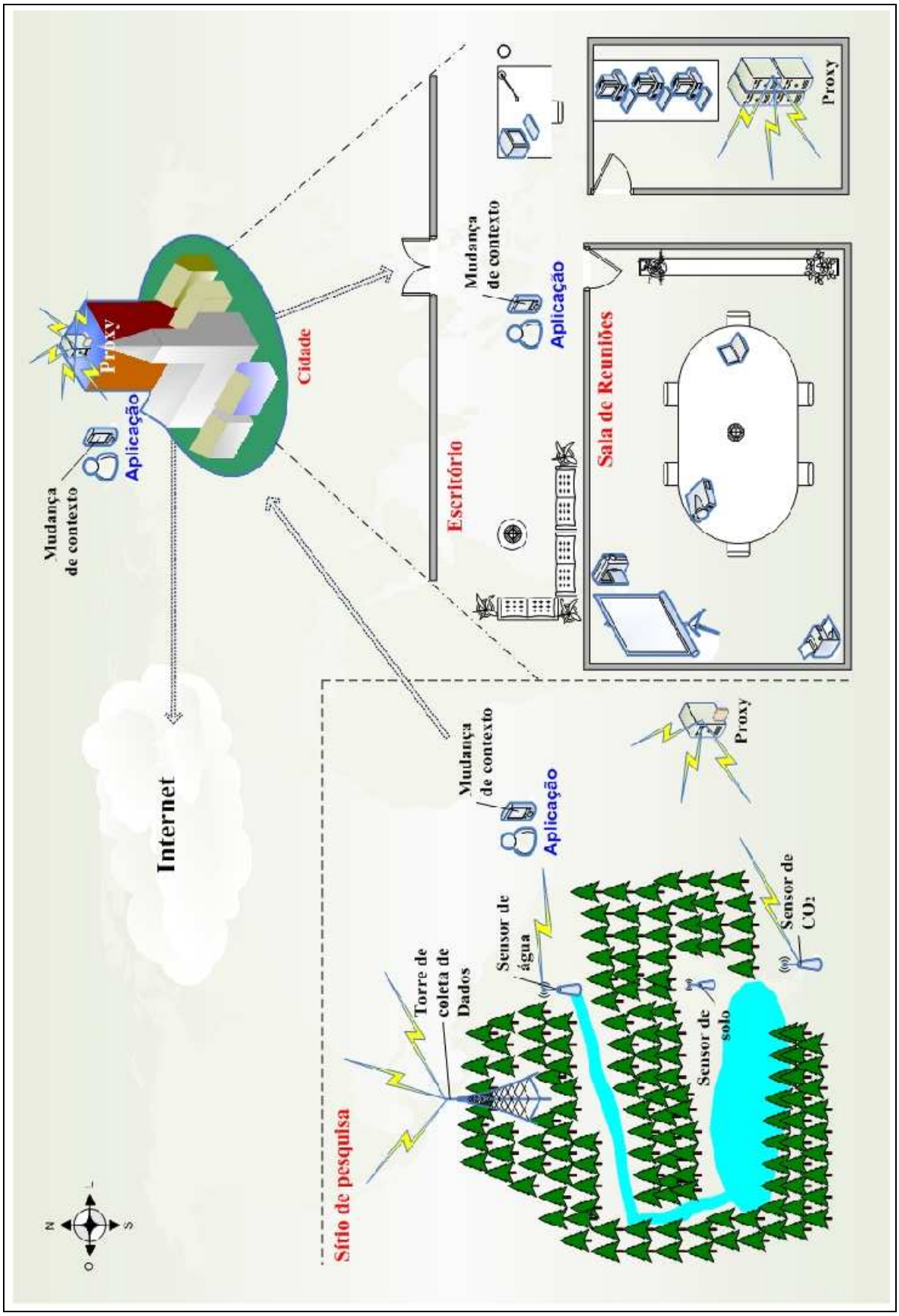

Figura A.3: Exemplo de utilização do SCaSDP no sítio do LBA.

grupos de pesquisa podem tê-los instalado. O pesquisador, portanto, necessita de um mecanismo para descobrir os sensores necessários para fazer a monitoração. Informando apenas o tipo de sensor, o tipo de informação gerada por ele, sua localização, e outras informações relevantes ele poderá utilizar o SCaSDP e, desde que tenha permissão de acesso, obterá os sensores ativos desejados e poderá dar prosseguimento à sua pesquisa.

Para utilizar os sensores o pesquisador não necessariamente precisa estar na base experimental K34, pois, com uma conexão interligando essa base à *Internet*, tais sensores podem ser descobertos e remotamente o sistema pode obter as informações desejadas. O pesquisador pode, por exemplo, estar em seu escritório, ou mesmo em sua residência e de um desses lugares acompanhar o andamento da pesquisa. Uma vantagem que claramente pode ser observada é a redução nos custos de deslocamento de pessoal até os sítios de pesquisa, pois não seria necessário que o pesquisador se deslocasse para coletar os dados uma vez que estes dados estariam disponíveis para seu uso em local acessível a partir da *Internet*.

O SCaSDP pode ser utilizado também por algum sistema de monitoração para automaticamente encontrar serviços que disparem ações pelo pesquisador. Um exemplo disso é a situação em que um sistema que verifica constantemente a taxa de carbono na atmosfera. Se a quantidade de carbono ultrapassar um valor esperado, o sistema deve avisar os integrantes da equipe de pesquisa interessados e que estejam próximos do local onde o fenômeno está acontecendo. Utilizando o protocolo o sistema pode solicitar descoberta dos provedores de serviço dos integrantes da equipe de pesquisa (neste caso os provedores de serviço dos integrantes da equipe de pesquisa são dispositivos móveis que identificam esses integrantes através de descrições semânticas) e avisá-los, ou mesmo enviar para eles um resumo dos dados coletados do evento.

Uma outra tarefa do sistema de monitoração da taxa de carbono na atmosfera é encontrar sensores que possam ter suas medições afetadas pela elevação da taxa de carbono, o que também pode ser feito com a utilização do SCaSDP, e fazer uma compensação na medição desses sensores, ou apenas coletar os dados armazenados neles para posterior análise.

Para garantir uma infra-estrutura computacional em um ambiente inóspito, como o da região amazônica, que interligue sensores, servidores (atuando como *proxies*) e outros dispositivos é necessária uma infra-estrutura elétrica e física que garanta o funcionamento de todos os dispositivos computacionais. Apesar de inicialmente ser necessário um investimento para construção de estruturas físicas, instalação elétrica e aquisição de equipamentos de interconexão, como roteadores, *switches* cabeados e sem fio, a infra-estrutura computacional montada e em operação favorecerá alcançar alguns benefícios, como a redução de custo para o deslocamento de pessoal, agilidade na obtenção e análise de dados coletados, além de permitir acompanhar e monitorar o estado dos sensores instalados no sítio de pesquisa.

Apêndice B

Casos de Uso

Os protocolos de comunicao seguem a abordagem cliente-servidor, por isso as entidades aparecem como atores de alguns casos de uso.

B.1 Cliente SCaSDP

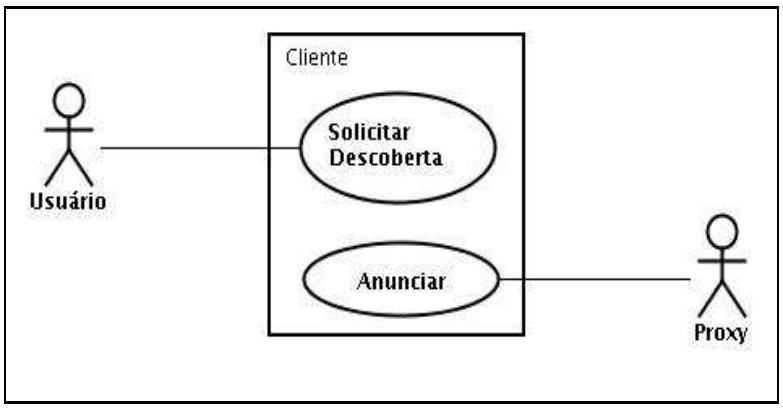

Figura B.1: Caso de uso Cliente.

Dois atores atuam na parte "Cliente", **Ator Usurio** e **Ator Proxy**, que representam respectivamente o papel desempenhado pelo usurio e a parte *Proxy* do protocolo. Abaixo segue uma descrio dos casos de uso.

Projeto: SCaSDP
Sub-Sistema: Cliente SCaSDP
Nome do Caso de Uso: Solicitar Descoberta
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pela solicitação de descoberta de serviço, sendo necessário informar conta e senha do cliente e os parâmetros contextuais para a descoberta.
Curso Normal:

•*Informar conta, senha e parâmetros contextuais*

O usuário informa conta, senha de acesso e parâmetros contextuais.

Caso conta e senha sejam válidas é retornada uma mensagem contendo a resposta referente a descoberta solicitada.

Cursos Alternativos:

•*Informar conta, senha e parâmetros contextuais*

Conta e/ou senha inválidas: uma mensagem de erro é retornada informando que conta e/ou senha são inválidas.

Classes: Cliente.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Cliente SCaSDP
Nome do Caso de Uso: Anunciar
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pelo recebimento de mensagens de anúncio do *proxy* da rede.
Curso Normal:

•*Receber mensagens de anúncio*

O *proxy* envia mensagem UDP contendo seu endereço IP e um campo indicando que é o *proxy* da rede.

Cursos Alternativos:
Classes: Cliente.
Restrições de Integridade:

B.2 Provedor de Servio

Dois atores atuam na parte "Provedor de Servio", **Ator Proxy** e **Ator Usurio**, que representam respectivamente o papel desempenhado pela parte *Proxy* e o administrador responsvel por configurar o protocolo no provedor. Abaixo segue uma descrio mais detalhada dos casos de uso.

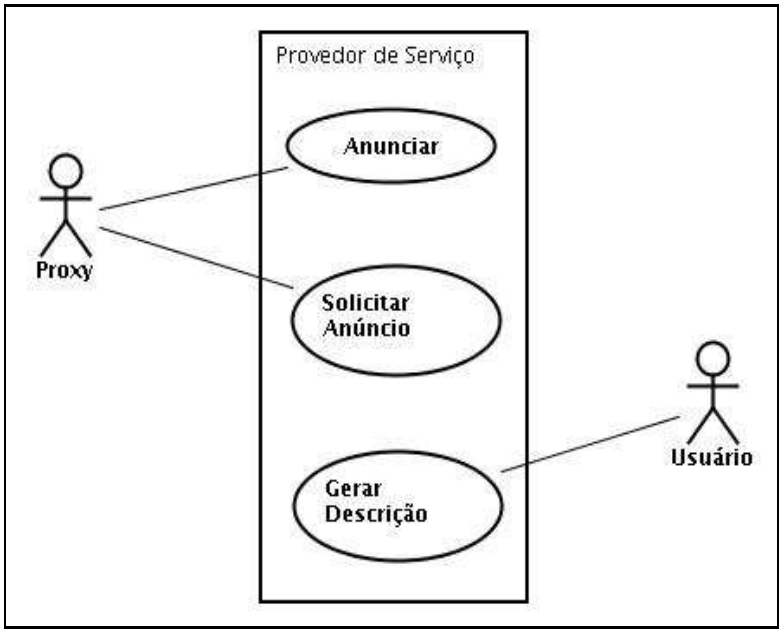

Figura B.2: Caso de uso Provedor de Servio.

Projeto: SCaSDP
Sub-Sistema: Provedor de Serviço SCaSDP
Nome do Caso de Uso: Anunciar
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pelo recebimento de mensagens de anúncio do *proxy* da rede.
Curso Normal:

- *Receber mensagens de anúncio*

  O *proxy* envia mensagem UDP contendo seu endereço IP e um campo indicando qual é o *proxy* da rede.

Cursos Alternativos:
Classes: Provedor de Serviço.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Provedor de Serviço SCaSDP
Nome do Caso de Uso: Solicitar Anúncio
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pela solicitação de anúncio requisitada pelo *proxy* da rede.
Curso Normal:

- *Solicitar anúncio*

  O *proxy* envia mensagem UDP requisitando anúncio do provedor de serviço.

Cursos Alternativos:
Classes: Provedor de Serviço.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Provedor de Serviço SCaSDP
Nome do Caso de Uso: Gerar Descrição
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pela geração da descrição do serviço.
Curso Normal:

- *Gerar Descrição*

  O *usuário* executa o comando de geração de descrição e segue a seqüência de preenchimento.

Cursos Alternativos:
Classes: Provedor de Serviço.
Restrições de Integridade:

B.3 Proxy

Dois atores atuam na parte "Proxy", **Ator Cliente** e **Ator Provedor de Servio**, que respectivamente representam os papis desempenhados pelo Cliente e Provedor de Servio do protocolo. Abaixo segue uma descrio mais detalhada dos casos de uso.

Projeto: SCaSDP
Sub-Sistema: Proxy SCaSDP
Nome do Caso de Uso: Descobrir Serviço
Data: 8 de novembro de 2005

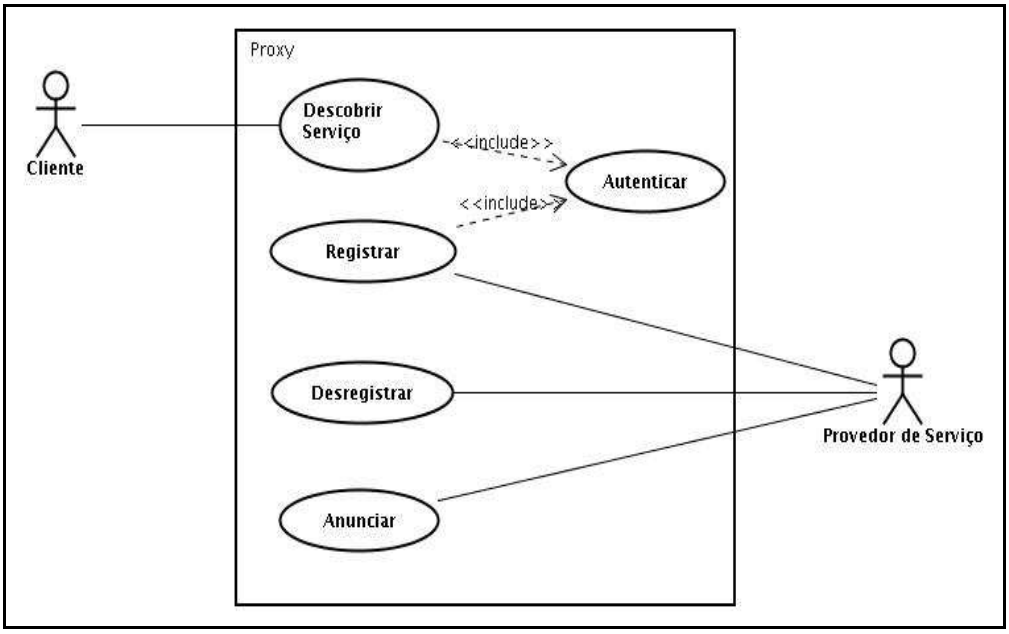

Figura B.3: Caso de uso *Proxy*.

Descrição: Este caso de uso é responsável pela descoberta de serviço, sendo necessário informar conta, senha e parâmetros contextuais para a descoberta.
Curso Normal:

- *Informar conta, senha e parâmetros contextuais*

  O cliente informa conta, senha de acesso e parâmetros contextuais.

  Caso conta e senha sejam válidas é retornada uma mensagem contendo a resposta referente a descoberta solicitada.

Cursos Alternativos:

- *Informar conta, senha e parâmetros contextuais*

  Conta e/ou senha inválidas: uma mensagem de erro é retornada informando que conta e/ou senha são inválidas.

Classes: Proxy.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Proxy SCaSDP
Nome do Caso de Uso: Registrar
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pela registro de um novo provedor de serviço, abrangendo autenticação de conta e senha.
Curso Normal:

•É realizado o caso de uso *Autenticar*.

•*Incluir Novo Provedor de Serviço*

O provedor de serviço informa sua chave pública e a descrição de serviço.

Cursos Alternativos:
Classes: Proxy.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Proxy SCaSDP
Nome do Caso de Uso: Cancelar Registro
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pelo cancelamento de registro do provedor de serviço, abrangendo autenticação de conta e senha.
Curso Normal:

•É realizado o caso de uso *Autenticar*.

•*Cancelar registro do Provedor de Serviço*

O provedor de serviço solicita o cancelamento do se registro.

Cursos Alternativos:
Classes: Proxy.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Proxy SCaSDP
Nome do Caso de Uso: Anunciar
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pelo recebimento de mensagens de anúncios dos provedores de serviço.
Curso Normal:

•*Receber mensagens de anúncio*

Os provedores de serviço enviam mensagens TCP ou UDP contendo informações contextuais e a descrição do serviço.

Cursos Alternativos:
Classes: Provedor de Serviço.
Restrições de Integridade:

Projeto: SCaSDP
Sub-Sistema: Proxy SCaSDP
Nome do Caso de Uso: Autenticar
Data: 8 de novembro de 2005

Descrição: Este caso de uso é responsável pela autenticação de conta e senha.
Curso Normal:

- *Autenticar conta e senha*

  O provedor de serviço ou o cliente informam conta e senha para autenticação.

  Caso conta e senha sejam válidas é retornada uma mensagem contendo a chave pública do *proxy*.

Cursos Alternativos:

- *Autenticar conta e senha*

  Conta e/ou senha não são válidas: é retornada uma mensagem negando o acesso.

Classes: Provedor de Serviço.
Restrições de Integridade:

Apêndice C

Análise Orientada a Objetos do SDCaDP

C.1 Diagramas de Classes

O diagrama de classes do protocolo foi dividido de acordo com as entidades para facilitar a visualização e pode ser visto nas Figuras C.1, C.2 e C.3. As Figuras C.4 e C.5 ilustram o módulo de tratamento dos pacotes e a classe utilizada para manipular os dados do protocolo, respectivamente.

C.1.1 Dicionário de Dados

- **Client:** representa a parte cliente do protocolo. Através dela aplicações de usuário poderão solicitar descoberta de serviços.
  - *latitude:* latitude.
  - *longitude:* longitude.
- **Provider:** representa a parte do protocolo responsável por prover informação sobre serviços. Através dela dispositivos computacionais anunciam descrições de serviços ao *proxy*.
- **ThreadedProviderAdv:** processo executado pelo *Provider* para anunciar descrição de serviço.
  - *allDone:* controle de iniciação e parada do processo.
  - *configpath:* caminho do arquivo de configuração.
  - *privateKey:* chave privada do provedor.
- **Proxy:** representa a parte do protocolo responsável por registrar, desregistrar, autenticar e descobrir serviços. Os provedores de serviço se anunciam ao *proxy* para terem suas descrições disponíveis, enquanto os clientes solicitam descoberta de serviços a ele.
  - *configpath:* caminho do arquivo de configuração.

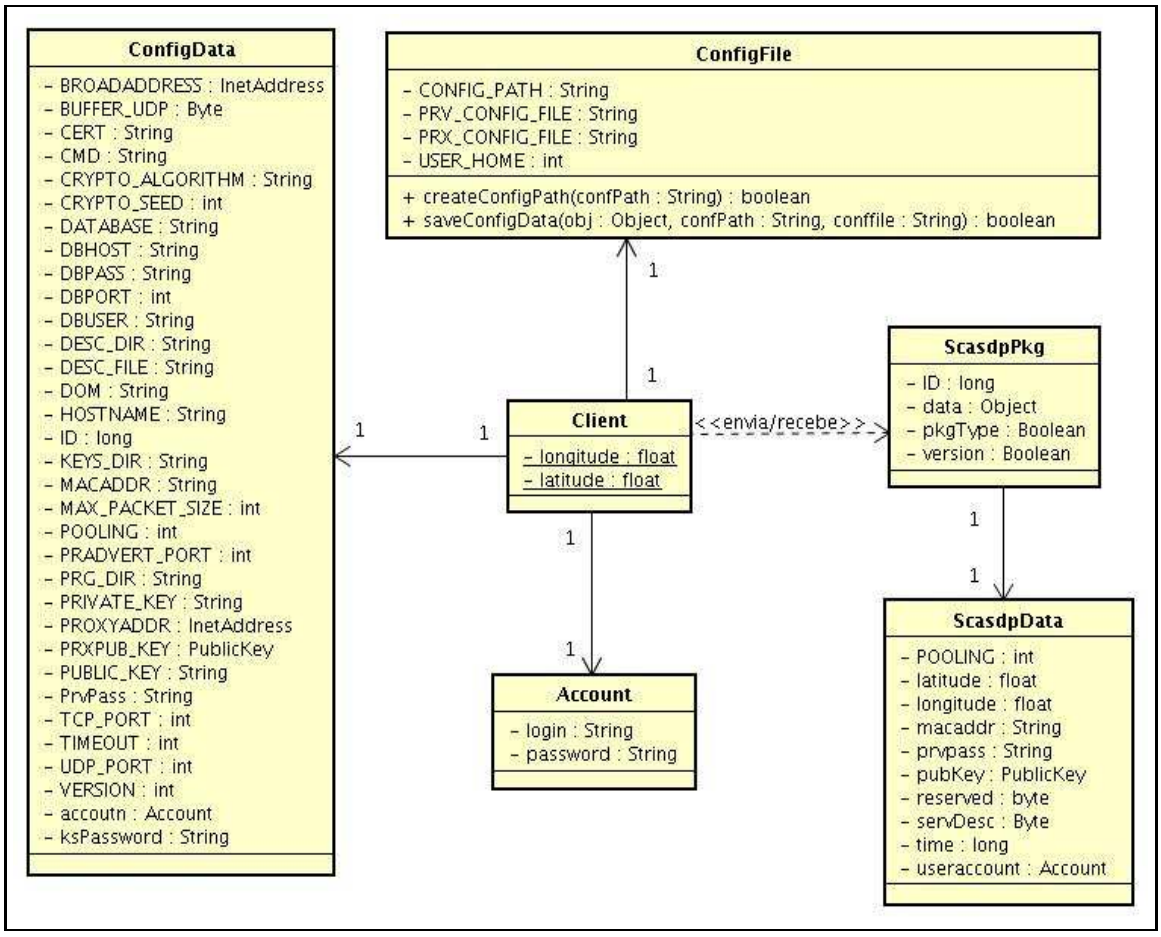

Figura C.1: Diagrama de Classes da entidade Cliente do SCaSDP.

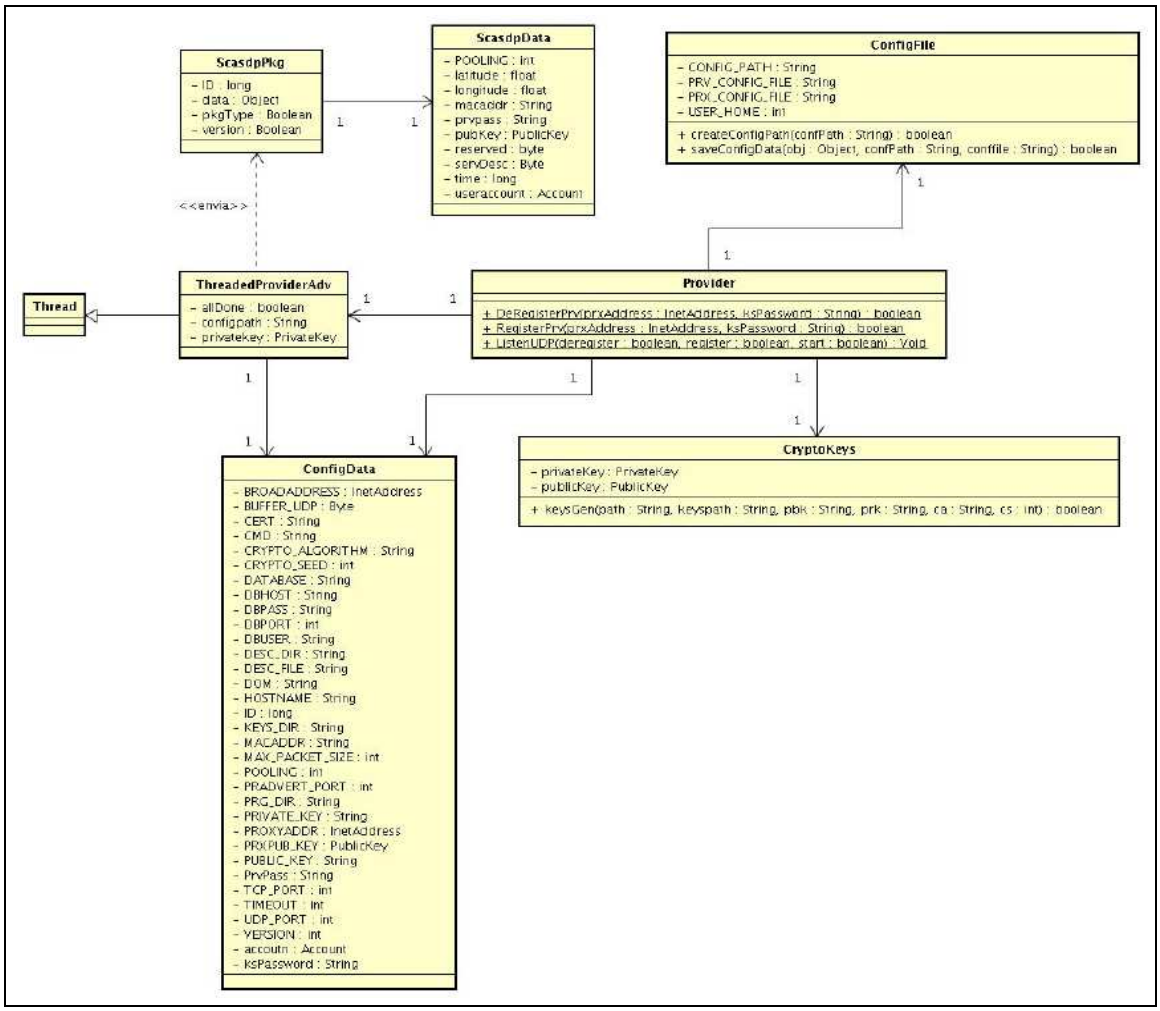

Figura C.2: Diagrama de Classes da entidade Provedor do SCaSDP.

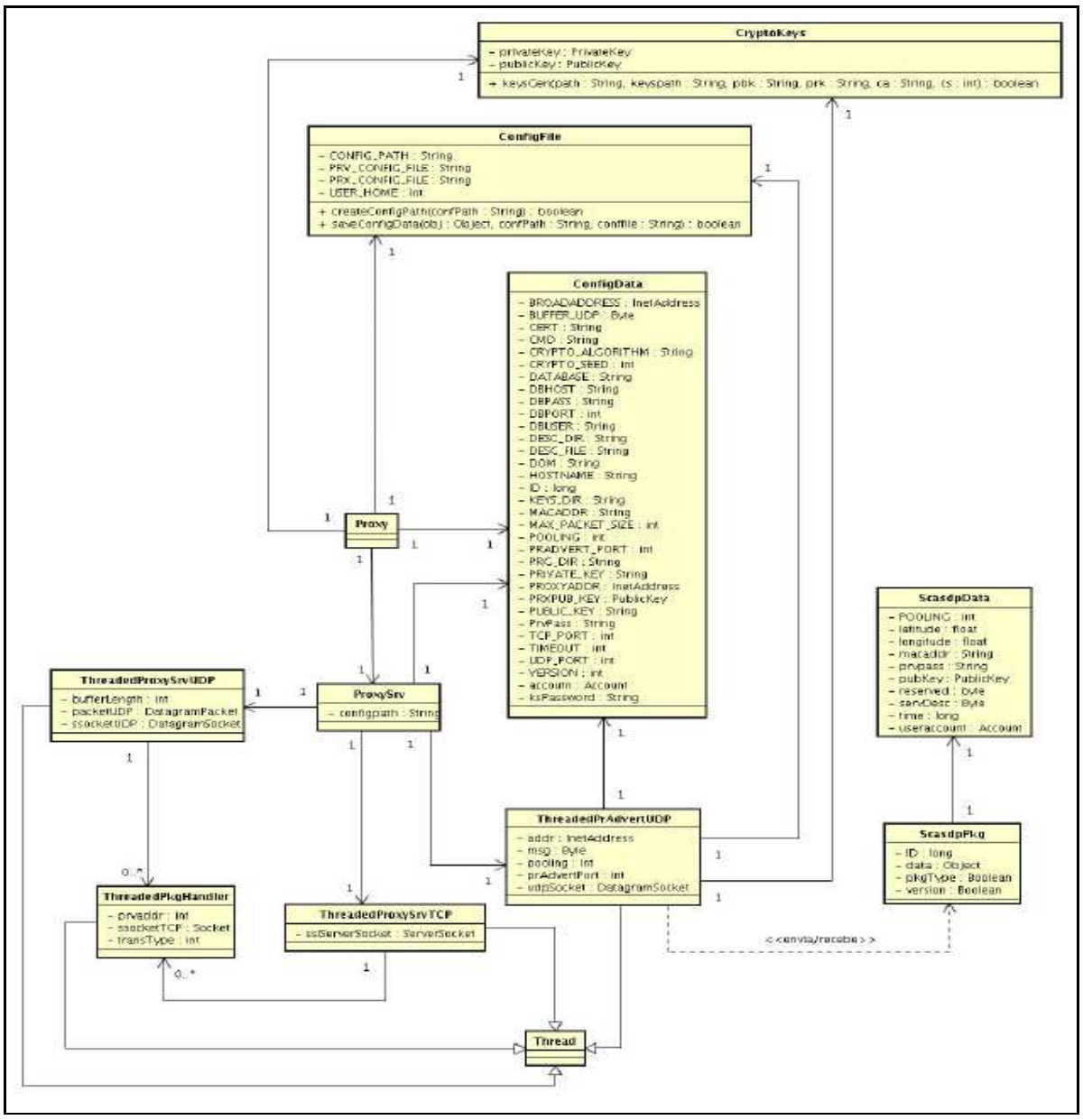

Figura C.3: Diagrama de Classes da entidade *Proxy* do SCaSDP.

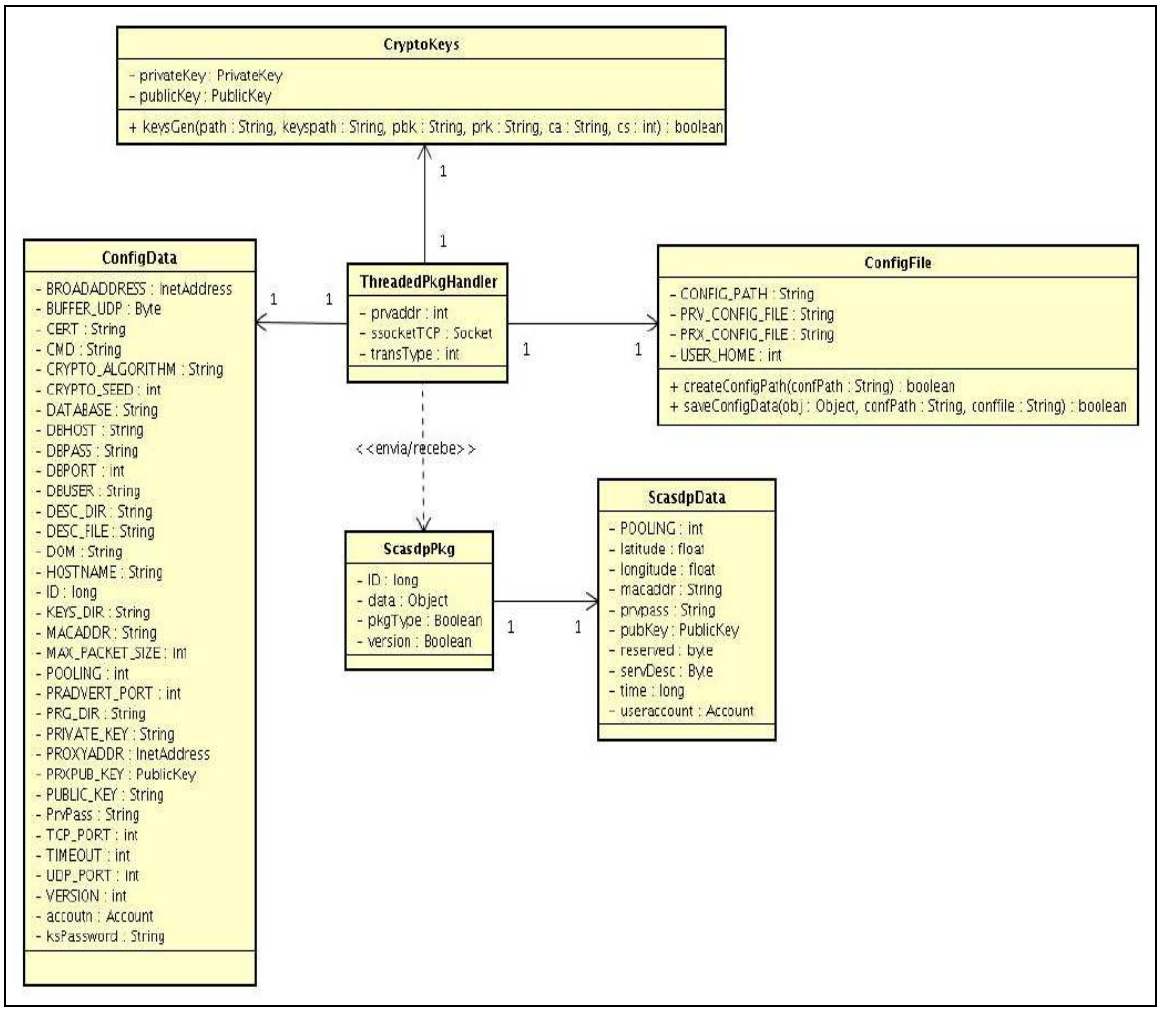

Figura C.4: Classe *ThreadedPkgHandler*.

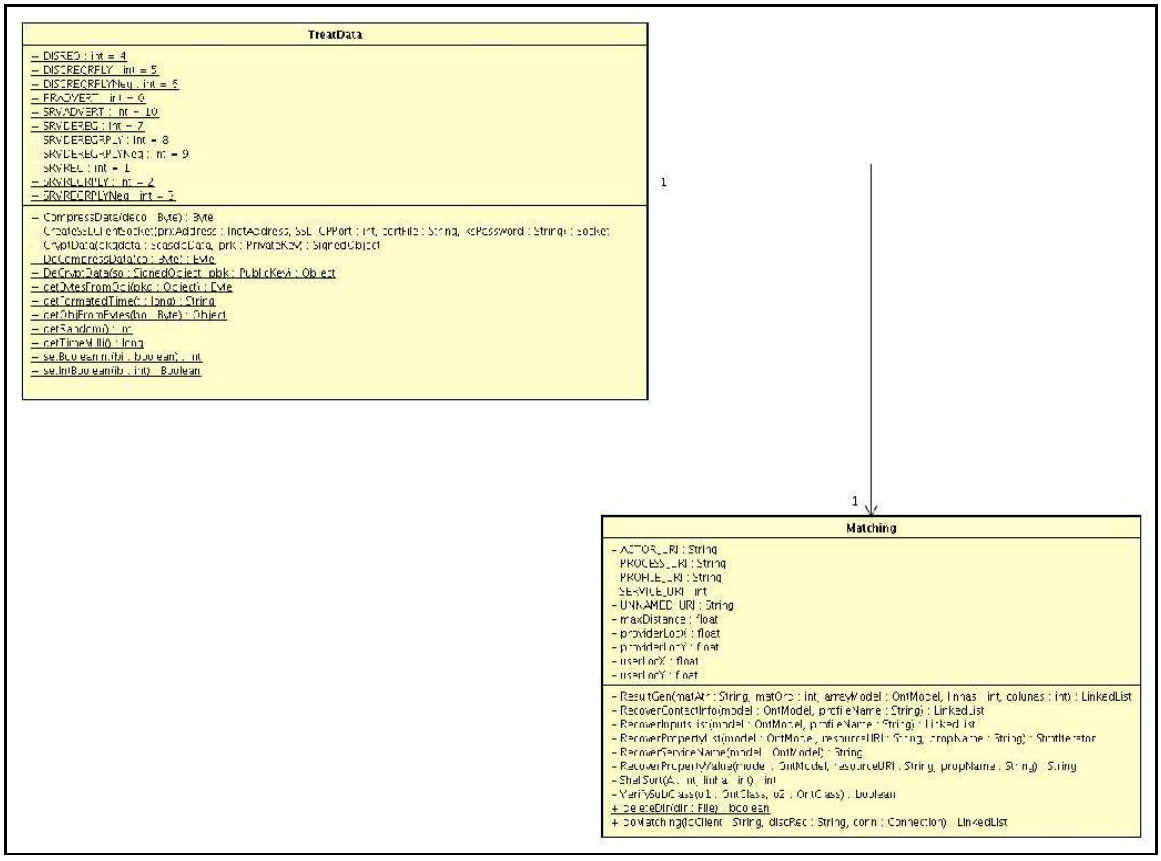

Figura C.5: Classes *TreatData* e *Matching*.

•**ProxySrv:** representa a iniciação e configuração do *Proxy*.

−*configpath:* caminho do arquivo de configuração.

•**ThreadedProxySrvUDP:** representa o processo de anúncio do *Proxy* via UDP.

−*bufferLength:* tamanho do pacote.

−*packetUDP:* pacote UDP.

−*ssocketUDP: socket* UDP.

•**ThreadedProxySrvTCP:** representa o processo de anúncio do *Proxy* via TCP.

−*sslServerSocket:* servidor *socket* seguro.

•**ThreadedPrAdvertUDP:** representa o processo de anúncio do *Proxy* via UDP.

−*addr:* endereço para enviar anúncio.

−*msg:* mensagem a ser enviada.

−*pooling:* intervalo entre anúncios.

−*prAdvertPort:* porta de anúncio.

−*udpSocket: socket* UDP.

•**ThreadedPkgHandler:** representa o processo de tratamento dos pacotes recebidos.

−*prvaddr:* endereço do provedor.

−*ssocketTCP: socket*.

−*transType:* tipo de pacote.

•**ScasdpPkg:** representa o pacote que contém informações relevantes às três partes do protocolo.

−*ID:* identificação da entidade que envia o pacote.

−*data:* dados do pacote.

−*pkgType:* tipo do pacote.

−*version:* versão do protocolo.

•**ScasdpData:** representa os dados de ScasdpPkg.

−*POOLING:* intervalo de tempo entre anúncios.

−*latitude:* latitude.

−*longitude:* longitude.

−*macaddr:* endereço MAC da entidade.

−*prvpass:* senha do provedor.

−*pubKey:* chave pública da entidade.

−*reserved:* reservado para futuras atualizações do protocolo.

−*servDesc:* descrição de serviço.

−*time:* hora em secundos.

−*useraccount:* conta e senha de usuário.

•**Account:** representa a conta de acesso do cliente ou do provedor de serviço utilizada para autenticação no *proxy*.

−*login:* conta de acesso do cliente ou do provedor de serviço (Opcional).

−*password:* senha de acesso do cliente ou do provedor de serviço (Opcional).

•**CryptoKeys:** representa o par de chaves de criptografia utilizadas para a cifragem e decifragem de ScasdpData.

−*publicKey:* chave pública de criptografia.

−*privateKey:* chave privada de criptografia.

•**ConfigFile:** representa os arquivos de configuração da entidade.

−*CONFIG_PATH:* caminho do arquivo de configuração.

–*PRV_CONFIG_FILE:* arquivo de configuração do *Proxy*.

–*PRX_CONFIG_FILE:* arquivo de configuração do *Provider*.

–*USER_HOME:* caminho onde se encontram os arquivos de usuário.

•**ConfigData:** representa os dados de configuração da entidade.

–*BROADADDRESS:* endereço de *broadcast*.

–*BUFFER_UDP:* tamanho do *buffer* do pacote UDP.

–*CERT:* certificado de segurança.

–*CMD:* programa a ser executado.

–*CRYPTO_ALGORITHM:* algoritmo de criptografia.

–*CRYPTO_SEED:* semente para cifragem de dados.

–*DATABASE:* banco de dados.

–*DBHOST:* servidor de banco de dados.

–*DBUSER:* conta de usuári para acesso ao banco de dados.

–*DESC_DIR:* diretório onde descrição de serviço está disponível.

–*DESC_FILE:* arquivo de descrição de serviço.

–*DOM:* domínio *Internet*.

–*HOSTNAME:* home do *host*.

–*ID:* identificação da entidade.

–*KEYS_DIR:* diretório onde as chaves de criptografia estão disponíveis.

–*MACADDR:* endereço *MAC* da entidade.

–*MAX_PACKET_SIZE:* tamanho máximo do pacote UDP.

–*POOLING:* intervalo entre anúncio.s

–*PRADVERT_PORT:* porta de anúncio do *Proxy*.

–*PRG_DIR:* diretório onde o *CMD* está disponível.

–*PRIVATE_KEY:* chave privada.

–*PROXYADDR:* endereço do *Proxy*.

–*PRXPUB_KEY:* chave pública do *Proxy*.

–*PUBLIC_KEY:* chave pública.

–*PrvPass:* senha do provedor.

–*TCP_PORT:* porta TCP do *Proxy* para recebimento de mensagens.

–*TIMEOUT:* tempo limite de espera.

–*UDP_PORT:* porta UDP do *Proxy* para recebimento de mensagens.

–*VERSION:* versão do protocolo.

–*account:* conta/senha de usuário.

–*ksPassword:* senha do certificade de criptografia.

- **TreatData:** classe para tratamento de dados.
  - *DISREQ:* mensagem de requisição de descoberta de serviço.
  - *DISCREQRPLY:* mensagem de resposta positiva de requisição de descoberta de serviço.
  - *DISCREQRPYNeg:* mensagem de resposta negativa de requisição de descoberta de serviço.
  - *PRADVERT:* mensagem de anúnio do *Proxy*.
  - *SRVADVERT:* mensagem de anúnio de descrição de serviço.
  - *SRVDEREG:* mensagem de requisição de cancelamento de registro de provedor de serviço.
  - *SRVDEREGRPLY:* mensagem de resposta positiva de cancelamento de registro de provedor de serviço.
  - *SRVDEREGRPLYNeg:* mensagem de resposta negativa de cancelamento de registro de provedor de serviço.
  - *SRVREG:* mensagem de requisição de registro de serviço.
  - *SRVREGRPLY:* mensagem de resposta positiva de registro de serviço.
  - *SRVREGRPLYNeg:* mensagem de resposta negativa de registro de serviço.

C.2 Diagrama de Estados - Classe Pacote

A Figura C.6 mostra o diagrama de estados da classe **PacoteMensagem**. Os eventos mostrados dizem respeito à realização dos casos de uso ou ações dos mesmos, tal como **Solicitar Descoberta**.

C.3 Diagrama de Seqüência

As Figuras C.7 e C.8 mostram, respectivamente, os diagramas de seqüência do caso de uso **Solicitar descoberta** feita pela primeira vez por um cliente e **Registrar** um provedor de serviço no *proxy*.

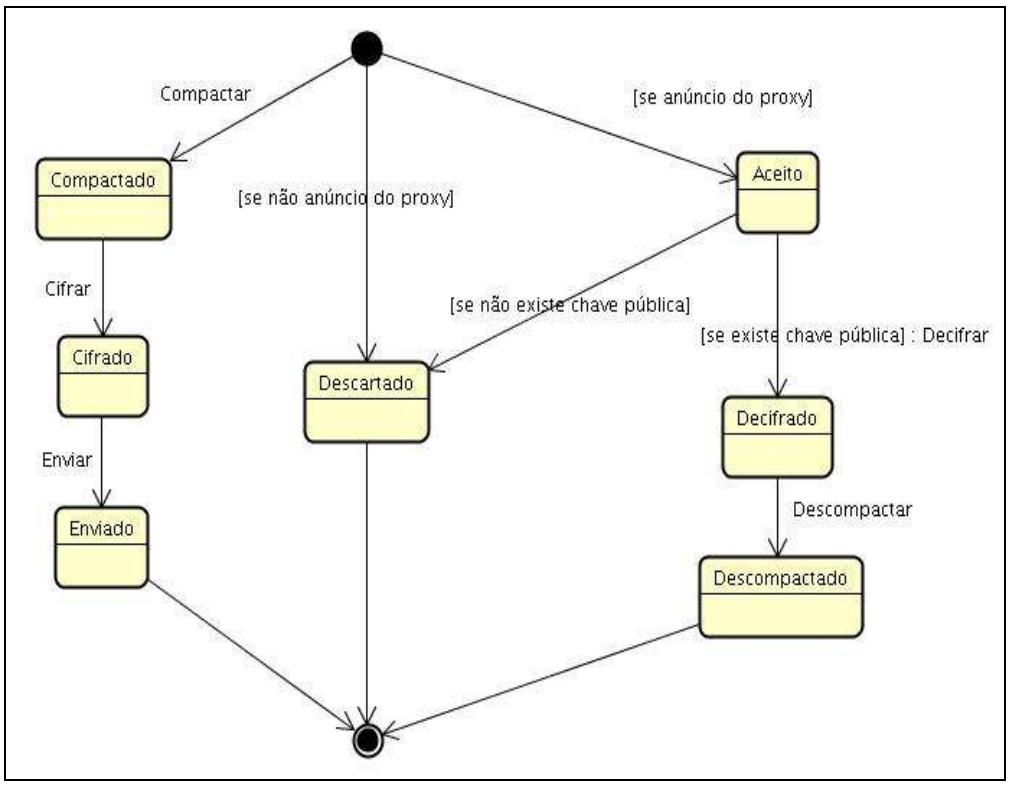

Figura C.6: Diagrama de estado da classe Pacote do SDCaSP.

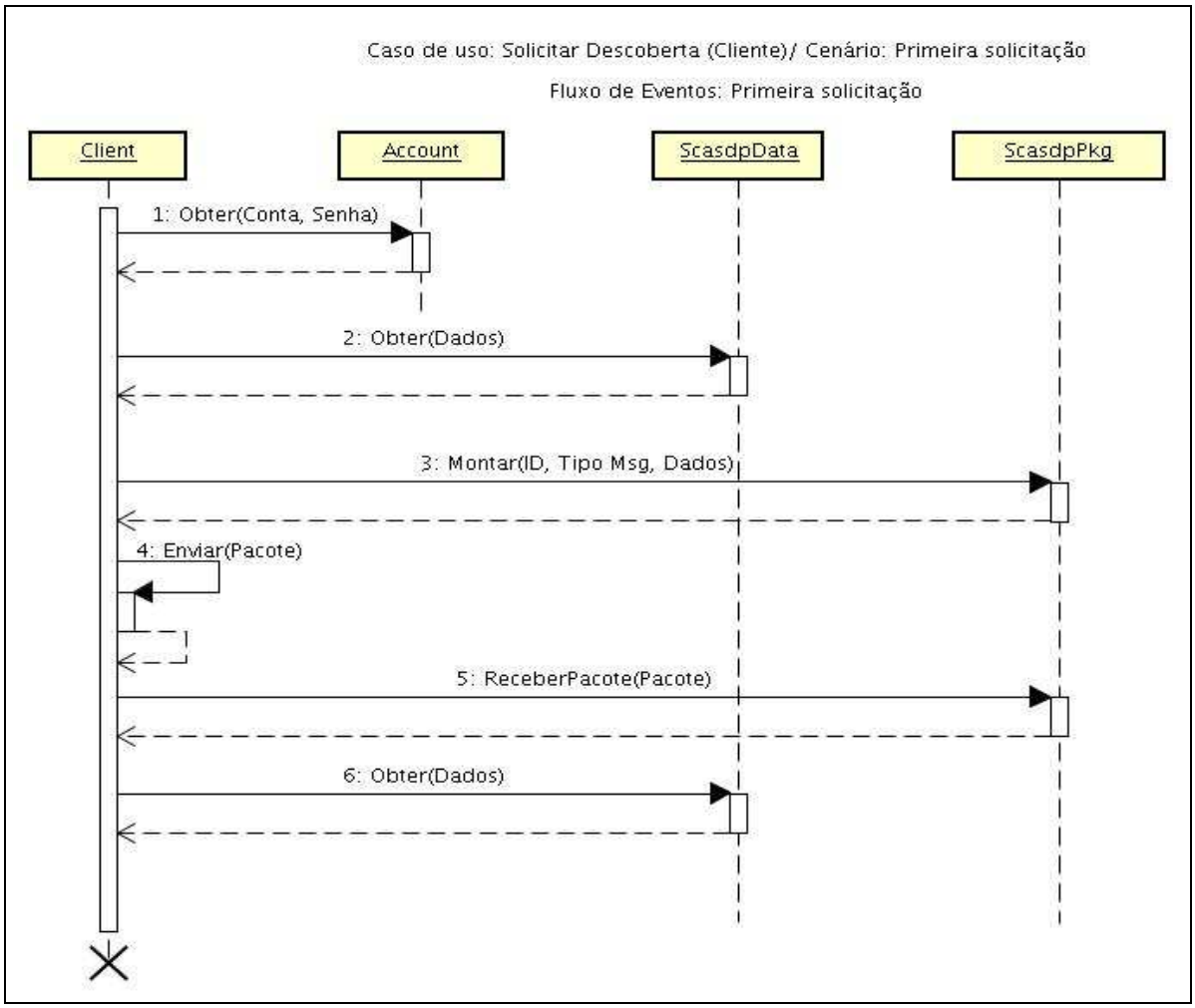

Figura C.7: Primeira solicitação de descoberta de um cliente.

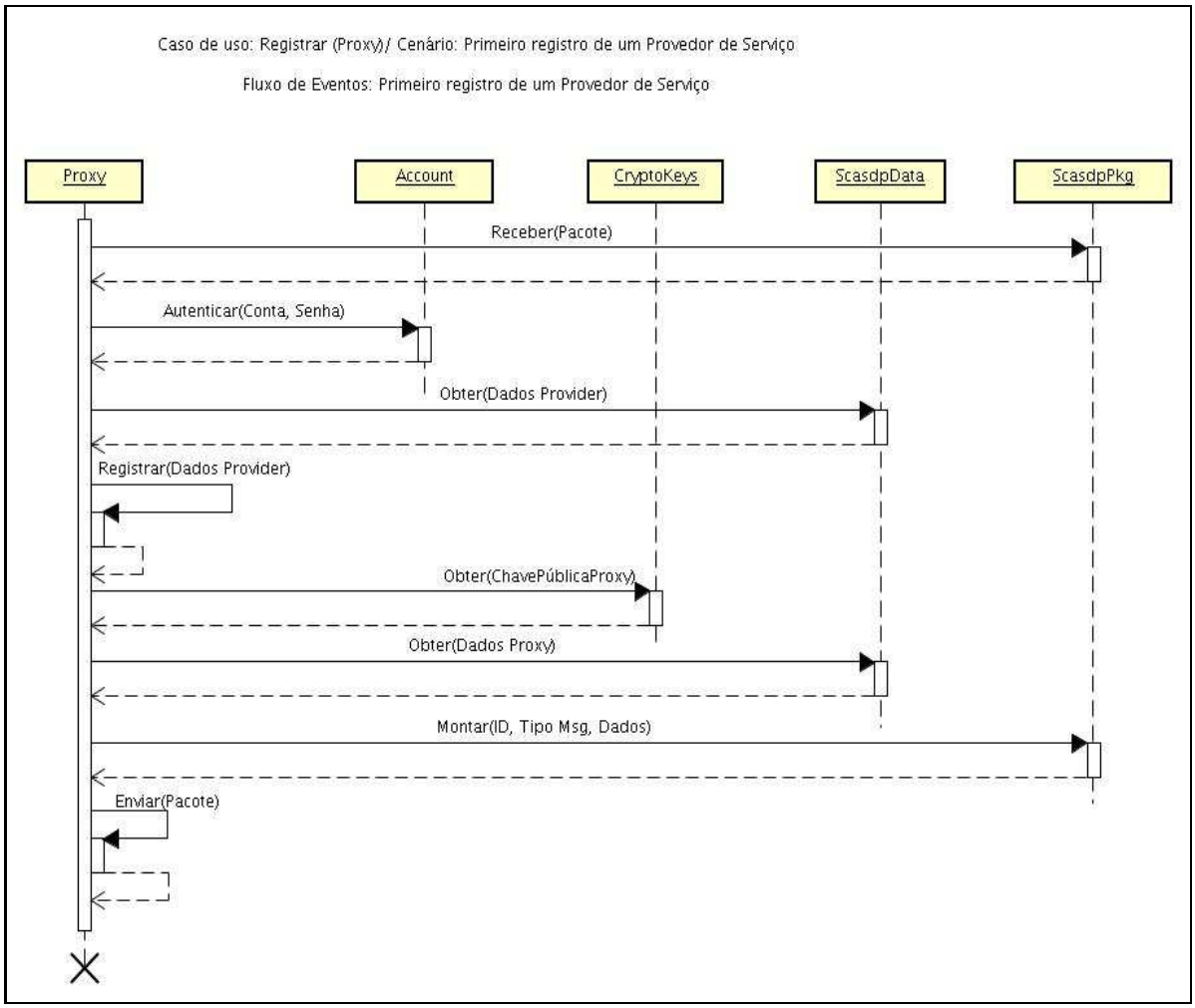

Figura C.8: Registro de provedor de serviço no proxy.