Itamar de Rezende

Uma Arquitetura para Compartilhamento de Informações no Formato *XML* em Redes *Peer-to-Peer*

Florianópolis, Fevereiro de 2008.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Itamar de Rezende

Uma Arquitetura para Compartilhamento de Informações no Formato *XML* em Redes *Peer-to-Peer*

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Frank Augusto Siqueira

Florianópolis, Fevereiro de 2008.

Uma Arquitetura para Compartilhamento de Informações no Formato *XML* em Redes *Peer-to-Peer*

Itamar de Rezende

Essa Tese foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, na área de concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Mario Antonio Ribeiro Dantas, Dr.

Banca Examinadora

_____

Frank Augusto Siqueira, Ph.D. (orientador)

_____

Michelle Silva Wangham, Dra.

_____

Ronaldo dos Santos Mello, Dr.

_____

Lau Cheuk Lung, Dr.

Sem estudo a alma adoece.
(Sêneca)

Aos meus pais e irmãos que sempre me apoiaram.
Em especial a minha querida esposa,
que esteve ao meu lado desde o início
e que foi muito importante
na realização deste sonho.

Agradeço a Deus
e a todos que de alguma forma contribuíram
para a realização deste trabalho.
À Universidade Federal de Santa Catarina
e aos professores do mestrado pelos ensinamentos.
E, em especial, ao meu orientador pelo apoio e atenção dispensados.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

RESUMO

Este trabalho propõe uma arquitetura para disponibilizar acesso a dados em formato *XML* armazenados de forma distribuída em dispositivos interligados através de uma rede *peer-to-peer (P2P)*. Por meio desta arquitetura é possível consultar e disponibilizar dados de maneira amigável e flexível a partir de diferentes dispositivos autônomos distribuídos pela rede, abrangendo desde servidores de conteúdo até dispositivos móveis, como telefones celulares e *PDAs*. Dispositivos com pouca capacidade de armazenamento e sujeitos a desconexão podem utilizar os demais *peers* da rede *P2P* para armazenamento de dados, o que proporciona uma maior disponibilidade dos dados compartilhados na rede. Uma linguagem de definição de ontologias, a *OWL*, é utilizada para descrever a semântica dos dados, a fim de tornar mais eficiente a consulta e o acesso às informações. A arquitetura proposta teve seu funcionamento comprovado através da implementação de um protótipo.

ABSTRACT

This work proposes an architecture for sharing data in XML format stored in a distributed way by devices interconnected through a peer-to-peer (P2P) network. This architecture allows users to share and retrieve data in a friendly and flexible way from different independent devices distributed through the network, ranging from content servers to mobile devices, such as cell phones and PDAs. Devices with low storage capacity and subject to disconnection may use other peers of the P2P network for storing data, providing higher availability to the data shared through the network. An ontology definition language, the OWL, is employed to describe data semantics, resulting in more efficient search results. In order to evaluate the proposed architecture, a working prototype has been implemented.

1 INTRODUÇÃO

A tecnologia *peer-to-peer (P2P)* popularizou-se tendo em vista a transferência de arquivos de áudio e vídeo utilizados amplamente pela comunidade de usuários da Internet. Sistemas de compartilhamento populares como Napster, Gnutella e Kazaa têm atraído a atenção para a tecnologia *P2P*. No entanto, esta tecnologia também permite muitas outras aplicações. A utilização de novas formas de busca e distribuição de informações é uma das maneiras possíveis de utilizar esta tecnologia que ainda não foi suficientemente explorada (ANDROUTSELLIS-THEOTOKIS & SPINELLIS, 2004).

Na *World Wide Web*, que adota o modelo cliente/servidor, as informações são centralizadas em servidores, que as disponibilizam para acesso aos clientes. Esse modelo necessita uma grande estrutura de controle de acesso e armazenamento, resultando em investimentos elevados em infra-estrutura computacional e na administração do sistema, e permite o controle da informação por governos e grandes empresas. Adicionalmente, a *Web* não apresenta uma organização clara dos conteúdos, não contextualizando os dados de forma a permitir maior eficiência no retorno de informações pesquisadas. Tal fato exigiu que fosse criada toda uma infra-estrutura de pesquisa de conteúdo da *Web*, que apresenta alto custo e eficiência questionável, tanto no uso desnecessário de largura de banda quanto em relação à limitada precisão dos resultados obtidos (HALEVY, 2003), (BERNERS-LEE, 2001).

A presente dissertação apresenta o projeto ***XOP – XML data Objects in P2P networks***, ou objetos *XML* em redes *P2P* – que consiste em uma arquitetura para disponibilizar acesso a informações armazenadas de forma distribuída em dispositivos interligados através de uma rede *P2P*. Para tanto, os dados são descritos através de uma ontologia que permite a criação de objetos descritos em formato *XML* (BRAY et al, 2006). Essa estrutura permite o armazenamento, compartilhamento e pesquisa de informações distribuídas pela rede. Uma interface de consulta é disponibilizada, permitindo a busca de forma amigável e intuitiva ao usuário em diferentes dispositivos. Toda essa solução é amparada por uma rede *P2P* usando uma arquitetura descentralizada.

A arquitetura *XOP* é baseada nos conceitos de orientação a objetos. A orientação a objetos está centrada no princípio de que é possível simular o mundo real através de

sistemas de computação, criando um mundo povoado de objetos de software que representam os objetos do mundo real. Nesse contexto, as pessoas têm uma compreensão do mundo ao seu redor na forma de objetos, ou seja, através de modelos físicos com informações descritivas do objeto. Quando se procura alguma coisa, as pessoas fazem mentalmente uma associação a uma forma física do objeto em questão com base nas informações que nos foram fornecidas sobre este objeto. Evidentemente que uma única característica não é o bastante para identificar um objeto, sendo necessário um conjunto de características para que se possa criar um modelo do objeto. Uma vez criado o modelo do objeto, este pode ser utilizado como filtro na busca de informações. Disponibilizar os mais variados conteúdos sobre uma diversidade de assuntos é uma tarefa complexa. Sendo assim, a representação dos conteúdos através de objetos, inserido nos conceitos de orientação a objeto, visa facilitar a definição desses conteúdos.

Para tanto é necessário o uso de um formalismo que possa representar esses conteúdos. Assim, este trabalho emprega uma ontologia – mais especificamente o padrão recomendado pelo *W3C*, a linguagem *OWL* (OWL, 2006) como forma de descrever os conteúdos. É de fundamental importância a definição de uma semântica através de uma linguagem apropriada, que permita descrever os conteúdos, para que se estabeleça um padrão para que os mais variados tipos de conteúdos possam ser criados. Disponibilizar informação de forma organizada, onde os conteúdos retornados aos usuários sejam claros e coerentes com o contexto de uma pesquisa, é um processo complexo. O uso de uma ontologia visa organizar os conceitos de forma que qualquer conteúdo possa ser publicado ou acessado.

1.1 Objetivos

Este trabalho tem como principal objetivo a criação de uma arquitetura para disponibilizar acesso a informações armazenadas de forma distribuída em dispositivos interligados através de uma rede *P2P*.

O trabalho possui os seguintes objetivos específicos:

- criar uma arquitetura de banco de dados *XML* para armazenamento dos dados em redes P2P;

- propor uma arquitetura de rede *P2P* descentralizada;
- propor uma maneira para descrever os dados armazenados a fim de permitir a consulta e o acesso geral às informações;
- criar uma interface de consulta e carga dos dados armazenados de maneira amigável e flexível para diferentes usuários e dispositivos.

1.2 Justificativa

A tecnologia *P2P* foi amplamente utilizada para compartilhamento de arquivos e distribuição de músicas na Web. Porém, o mesmo não ocorre quanto ao compartilhamento de dados, sobretudo para geração de conhecimento. Assim, alguns poucos trabalhos, principalmente acadêmicos, foram apresentados com esse propósito, e mesmo estes são voltados para segmentos específicos e com públicos restritos como, por exemplo, compartilhar conhecimento sobre pesquisas acadêmicas ou o acervo de livros em bibliotecas e obras de arte em museus. (ANDROUTSELLIS-THEOTOKIS & SPINELLIS, 2004)

Assim, na tentativa de apresentar uma arquitetura que permita disponibilizar os mais diversos tipos de dados, e para os mais variados públicos, a proposta dessa dissertação é apresentar uma forma de compartilhar os dados de forma que estes possam ser utilizados em diversas áreas, como informática, telecomunicações, medicina, ou em áreas de negócio como bolsa de valores e comércio exterior. O seu público alvo é, portanto, amplo e heterogêneo, envolvendo desde um usuário com um celular até empresas com grandes servidores. Sua forma de utilização é bastante diversificada, podendo ser, por exemplo: um turista a procura de um hotel em uma praia com características específicas; uma empresa a procura da melhor cotação para um produto entre diversos fornecedores; um passageiro em busca dos horários e rotas dos meios de transporte públicos; entre muitas outras.

Deste modo, a arquitetura foi projetada para ser de fácil compreensão e para ter uma interface amigável. Adicionalmente, a arquitetura foi desenvolvida com o intuito de permitir o gerenciamento de informações envolvendo acesso e distribuição através dos mais diversos dispositivos, levando em conta as limitações de processamento e armazenamento de dispositivos móveis, como telefones celulares, *PDAs*, etc.

1.3 Metodologia

A metodologia utilizada para o desenvolvimento deste trabalho envolveu alguns passos fundamentais, sendo eles:

- Extensa pesquisa bibliográfica sobre linguagem *XML*, ontologias e redes *P2P*;
- Proposta e detalhamento de uma arquitetura para disponibilizar acesso a informações armazenadas de forma distribuída em dispositivos interligados através de uma rede *P2P*;
- Desenvolvimento da arquitetura, envolvendo:
    - Definição das tecnologias a serem utilizadas;
    - Desenvolvimento da ferramenta para suporte da arquitetura proposta *(XOP)*;
    - Testes de validação em um ambiente de rede;
    - Avaliação dos resultados e desempenho;
- Elaboração do trabalho escrito.

1.4 Organização do Texto

Este trabalho está organizado em oito capítulos:

- **Introdução**: contextualização do trabalho e descrição dos objetivos geral e específicos, justificativa e metodologia utilizada;
- **Linguagens de Marcação**: pesquisa bibliográfica sobre *XML (Extensible Markup Language)* envolvendo linguagens de marcação *(SGML – Structured Generalized Markup Language e HTML – HyperText Markup Language)*, aplicações, vantagens e desvantagens, comparativo *HTML* e *XML*, documentos *XML*, definição de esquemas de dados em *XML (DTD – Document Type Definition e XSD – XML Schema Definition)*, interfaces de programação de aplicações *(DOM – Document Object Model e SAX – Simple API for XML)*, bancos de dados *XML* e linguagens de pesquisa *(XPath - XML Path, XQuery – XML Query)*;

- **Ontologias**: pesquisa bibliográfica sobre ontologia, envolvendo conceitos, categorias, vantagens, tipos, passos e linguagens para construção, descrição da linguagem *OWL (Ontology Web Language)* e ferramentas para construção de ontologias;
- **Redes *Peer-to-Peer (P2P)***: pesquisa bibliográfica sobre redes *peer-to-peer (P2P)* envolvendo conceitos, características, aspectos de gerenciamento, vantagens e desvantagens, modelos de arquitetura e aplicações;
- **Compartilhamento de dados em redes P2P**: apresenta um estudo com outros trabalhos que abordam o mesmo assunto, considerando sua adequação para compartilhamento de dados no contexto do presente trabalho;
- **Arquitetura Proposta**: descrição detalhada da arquitetura proposta, envolvendo a sua infra-estrutura, bem como os serviços necessários para o seu funcionamento;
- **Implementação e Resultados**: apresentação de um protótipo da arquitetura *XOP* e descrição dos resultados obtidos em uma avaliação de desempenho;
- **Conclusões e Trabalhos Futuros**: apresentação das conclusões obtidas com a realização do presente trabalho, levando-se em consideração os objetivos geral e específicos propostos, e sugestões para prosseguimento e melhorias futuras da arquitetura *XOP*.

2 Linguagens de Marcação

A sociedade vive atualmente a revolução do conhecimento. A chamada "Era da Informação" trouxe as redes de computadores e os avanços nas áreas de microeletrônica, nanotecnologia e telecomunicações. A popularização das redes de computadores apresentou um novo significado para a palavra "informação", e estruturas sociais foram, e continuando sendo, transformadas por esta conectividade mundial (TIS, 2006).

Esta avalanche contínua de informações gerou também importantes questionamentos: como acessar, controlar e efetivamente utilizar toda esta informação e conhecimento? Como organizar e disponibilizar de forma eficiente todo este conteúdo na rede? Como escrever programas para pesquisar e agrupar estas informações? Como efetuar referências cruzadas e redefinir a finalidade de cada documento para aplicações diferentes?

Sendo assim, para solucionar estes e vários outros problemas relacionados à estrutura dos documentos, e não a sua apresentação em si, criou-se uma codificação genérica, utilizando marcadores descritivos em vez de códigos de formatação. Estes marcadores descritivos, por sua vez, deram origem às linguagens de marcação.

Segundo (RAY, 2001), uma linguagem de marcação é um conjunto de símbolos que pode ser colocado no texto de um documento para demarcar e rotular as partes deste documento. Sendo assim, marcações (ou "*tags*") visam identificar as partes de um documento, mostrando como estas se relacionam umas com as outras.

A primeira linguagem de marcação moderna surgiu em 1969, a *GML (General Markup Language)*, uma linguagem própria para a formulação de estruturas de um conjunto arbitrário de dados e que teve o propósito era ser uma meta-linguagem, isto é, uma linguagem que pudesse ser usada para descrever outras línguas, suas gramáticas e seus vocabulários. Mais tarde, a *GML* tornou-se a *SGML (Standard Generalized Markup Language)*. Em seguida, vieram a *HTML*, amplamente utilizada na *Web*, e a *XML*, usada para representar dados de modo independente de plataforma ou aplicação, e que vem a ser utilizada nesse trabalho. Tais linguagens são descritas e comparadas nesse trabalho, dando ênfase à linguagem *XML* e a sua utilização para representação de informação.

2.1 *SGML (Structured Generalized Markup Language)*

A *SGML* foi originalmente desenvolvida pela *ISO (International Organization for Standardization)* e trata-se de uma linguagem de marcação extremamente poderosa (e bastante complexa) que "marca" um documento de modo a fornecer instruções de formatação, *links* de hipertexto e definições para os componentes do documento (SAVOLA, 1996), (ANDERSON, 2001).

Órgãos governamentais e seus contratados, grandes fabricantes e editores de informação técnica, adotaram a *SGML* como uma forma de padronização de seus documentos técnicos, como livros, relatórios e manuais.

A *SGML* é utilizada para descrever diferentes tipos de documentos em muitas áreas da atividade humana, desde transcrições de antigos manuscritos irlandeses até documentação técnica para aviões de guerra; de registros de pacientes em unidades médicas até notação musical (ALMEIDA, 2002).

2.2 *HTML (HyperText Markup Language)*

A linguagem *HTML* foi desenvolvida em 1992, sendo uma versão simplificada da *SGML* e, portanto, também uma linguagem de marcação. No início, esta linguagem definia apenas a estrutura lógica de um documento, e não a sua aparência física. No entanto, suas versões posteriores foram adquirindo progressivamente o controle da aparência do documento. Sendo assim, pode-se dizer que documentos *HTML* são feitos para prover estrutura lógica da informação destinada à apresentação de páginas da rede mundial de computadores.

Com os avanços no desenvolvimento da *Web*, surgiu a necessidade de adotar uma nova abordagem para compartilhamento de dados, tendo como base uma linguagem de marcação que pudesse combinar todo o potencial da *SGML* com a ampla aceitação do *HTML*.

2.3 *XML (eXtensible Markup Language)*

XML (eXtensible Markup Language) trata-se de uma linguagem de marcação de fácil utilização, direcionada principalmente para a Internet. Alguns de seus objetivos são: atender uma ampla variedade de aplicações, contemplar um mínimo de características opcionais (idealmente zero) e executar o projeto rapidamente, de maneira formal e concisa (XML, 2006).

A *XML* provê, portanto, um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações; facilitar declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas; e estimular, desta forma, o surgimento de uma nova geração de aplicações de manipulação e visualização de dados via internet (FURTADO, 2006).

A linguagem *XML* foi desenvolvida e padronizada pelo *W3C (World Wide Web Consortium)* e promete ser o futuro da Internet na representação de dados estruturados. Essa linguagem propõe um formato universal para descrever dados estruturados, assegurando que os mesmos sejam uniformes e independentes de aplicações e fornecedores (ZACHARY & LU, 2000).

Oracle e IBM tratam a *XML* como um protocolo, utilizando-a para exportar informações de seus sistemas de banco de dados relacional para esse formato padrão. Outras empresas a utilizam como base para seus modelos de dados, beneficiando-se do armazenamento, integração e processamento dos dados dessa linguagem (ZACHARY & LU, 2000).

Contemplando as mais variadas áreas de negócio e pesquisa, como companhias aéreas, instituições financeiras, universidades, telefonia móvel, astronomia, biologia, a linguagem *XML* está presente em uma infinidade de aplicações (OASIS, 2006), como por exemplo:

- Previsão do Tempo *(Weather Observation Markup Format – OMF)*;
- Voz Interativa *(Extensible Phone Markup Language – XPML)*;
- Matemática *(Math Markup Language – MathML)*;
- Biblioteca *(BiblioML – XML for UNIMARC Bibliographic Records)*;
- Comércio Eletrônico *(Electronic Commerce Modeling Language – ECML)*;

- Logística/Transporte *(TransportationXML – tXML)*;
- Robótica *(Robotic Markup Language – RoboML)*;
- Liturgia/Textos Históricos *(Liturgical Markup Language – LitML)*;
- Indústria Petroquímica *(PetroXML Initiative)*.

2.4 Vantagens e Desvantagens da Linguagem *XML*

Dentre as inúmeras **vantagens** da *XML* em relação a outras formas de representação de informação, pode-se citar (W3C, 2004), (W3C, 1999), (FURTADO, 2006), (IDRIS, 2006), (MELLO, 2006), (RAY, 2001), (SPENCER, 1999):

- **padrão aberto**: é um padrão aberto, livre de licenças, independente de plataforma e bem suportada. Seus documentos independem da aplicação e sistemas operacionais, permitindo a troca de dados de forma transparente;
- **diversas formas de visualização dos dados**: dados representados em um documento *XML* podem ser visualizados de diversas formas, uma vez que a linguagem *XML* define somente o conteúdo do documento e não a sua aparência. A apresentação dos dados, portanto, pode acontecer de várias maneiras diferentes, de acordo com o contexto no qual os mesmos são utilizados;
- **desenvolvimento de aplicações flexíveis para a *Web***: o desenvolvimento de aplicações *Web* em três camadas (*three-tier*) é altamente factível com a *XML*. Os dados *XML* podem ser distribuídos para as aplicações, objetos ou servidores intermediários para processamento. Esses mesmos dados também podem ser distribuídos para o *desktop* (*PCs* e similares) e serem visualizados em qualquer navegador (*browser*);
- **natureza autodescritiva**: os documentos *XML* possuem dados autodescritivos, isto é, dados que descrevem a si mesmos e, portanto, são relativamente legíveis e fáceis de interpretar (SPENCER, 1999), (ANDERSON, 2001). Apesar de sua simplicidade, a *XML* permite criar estruturas bastante complexas (árvores ou grafos de profundidade arbitrária e, eventualmente, cíclicos e recursivos);

- **desenvolvimento de padrões específicos**: uma vez que a *XML* trata-se de uma meta-linguagem, isto é, uma estrutura que propicia a definição de outras linguagens, desenvolvedores podem utilizá-la para a criação de padrões específicos, conforme suas necessidades. A MathML, linguagem para descrever notações matemáticas, serve como exemplo de uma nova linguagem baseada na *XML*;
- **buscas mais eficientes**: os dados em *XML* podem ser unicamente "etiquetados", o que permite, por exemplo, que uma busca por livros seja feita em função do nome do autor. Atualmente, uma busca pelo nome do autor poderia levar a qualquer *site* que tivesse referência a tal nome, não importando se fosse o autor do livro ou simplesmente um livro sobre o autor. Sem a *XML* é necessário para a aplicação de procura saber como é esquematizado e construído cada banco de dados que armazena os dados de interesse, o que é impossível. A *XML* permitiria definir livros por autor, título, assunto, etc., o que facilitaria enormemente a busca;
- **integração de dados de fontes diferentes**: atualmente, é praticamente impossível a procura em múltiplos bancos de dados por serem incompatíveis, visto que cada fornecedor cria a sua própria arquitetura. A *XML* permite que tais dados possam ser facilmente combinados. Essa combinação seria feita via software em um servidor intermediário, estando os bancos de dados na extremidade da rede;
- **fácil distribuição na *Web***: assim como o *HTML*, a *XML* possui um formato baseado em texto aberto, podendo ser distribuído via *HTTP* sem necessidade de modificações nas redes existentes;
- **escalabilidade**: devido ao fato dos documentos *XML* separarem completamente os dados da forma como os mesmos são visualizados, autores de aplicações de visualização de dados podem torná-las muito poderosas e interativas, permitindo ao usuário visualizar os dados da forma que lhe agrade. Dessa forma, a interatividade, em termos, não dependeria tanto da comunicação cliente-servidor, mas seria feita "*offline*", reduzindo o tráfego do *link* com o servidor;

- **compressão**: a compressão de documentos *XML* é fácil devido à natureza repetitiva das *tags* usadas para definir a estrutura dos dados. A necessidade de compressão é dependente da aplicação e da quantidade de dados a serem movidos entre clientes e servidores. Os padrões de compressão do HTTP 1.1 podem ser usados para a *XML*;
- **computação e manipulação local**: os dados *XML* recebidos por um cliente são analisados e podem ser editados e manipulados de acordo com o interesse do usuário. Ao contrário de somente visualizar os dados, os usuários podem manipulá-los de várias formas. *APIs* como *DOM (Document Object Model)* e *SAX (Simple API for XML)* permitem às aplicações manipularem facilmente o conteúdo de um documento *XML*, o que torna possível atingir níveis de automação bastante elevados;
- **atualizações granulares dos documentos**: os dados podem ser atualizados de forma granular, evitando que uma alteração em um conjunto de dados implique novamente na busca de todo o documento. Dessa forma, somente os elementos alterados são enviados do servidor para o cliente. Além disso, a *XML* também permite que novos dados sejam adicionados aos já existentes, sem a necessidade de reconstrução da página.

Por outro lado, são apontadas na literatura as seguintes **desvantagens** da *XML (RAY, 2001), (SPENCER, 1999), (MELLO, 2006)*:

- **não é equivalente à tecnologia SGBD**: *XML* não possui soluções para todos os aspectos de gerenciamento de dados, como controle de integridade, gerenciamento de transações, indexação e consultas a múltiplos documentos;
- **integração com bancos de dados relacionais**: o formato de um dado *XML* é altamente irregular e, muitas vezes, combina texto e linguagem natural com informações estruturadas. Sendo assim, bancos de dados relacionais são incapazes de armazenar diretamente ocorrências de dados *XML*.

2.5 Comparação entre *HTML* e *XML*

Tanto o *HTML*, quanto a *XML* utilizam marcadores (palavras envoltas pelos sinais '<' e '>') e atributos (na forma nome="valor"). Mas enquanto *HTML* especifica o que

cada marcador e atributo significam, além de como seu conteúdo será mostrado no navegador, *XML* utiliza os marcadores apenas para delimitar os trechos de dados, deixando que sua interpretação seja realizada pela aplicação (W3C, 2004), (FURTADO, 2006).

Resumidamente, o *HTML* define a aparência de um documento na Internet, enquanto o *XML* descreve o conteúdo do documento. No Quadro 2.1 é apresentado um exemplo dos dois arquivos: enquanto em um documento *HTML* o marcador <H1> significa um cabeçalho em uma página do navegador, no *XML* o marcador <nome> indica o conteúdo nome de uma pessoa. Outras diferenças são: a ordem de fechamento dos marcadores, que é observada em *XML* e não é necessária em *HTML*; a necessidade de fechar todos os marcadores abertos no *XML*; o documento *XML* precisa ser bem-formado, enquanto *HTML* não faz essa verificação.

Quadro 2.1 – Exemplo comparativo entre arquivos *HTML* e *XML*.

Arquivo *HTML*	**Arquivo *XML***
<H1>Viviane Guimarães</H1> <H2>vivi@ufsc.com.br</H2> <P> <B>11</B> <I>9999 4321</I> </P>	<nome> Viviane Guimarães </nome> <email>vivi@ufsc.com.br</email> <telefone> <ddd>11</ddd> <numero>9999 4321</numero> </telefone>

2.6 Estrutura do Documento *XML*

Todo documento *XML* possui duas estruturas, uma física e outra lógica. Fisicamente, o documento é composto de unidades chamadas entidades. Logicamente, o documento é composto de declarações, elementos, comentários, referências e instruções de processamento, todos indicados no documento por um marcador explícito (W3C, 2004).

A organização ou esquema dos elementos em um documento *XML* pode ser definida basicamente de duas maneiras: através de um *DTD (Document Type Definition)* ou de um *XSD (XML Schema Definition)* (SCHEMA, 2006).

Esquemas definem a estrutura, o conteúdo e as semânticas dos documentos *XML* (W3C, 2004). Estabelecem formalmente quais elementos e quais combinações possíveis são permitidas em um documento *XML*. Sendo assim, um documento *XML* é dito válido se estiver de acordo com um esquema *DTD* ou *XSD*.

2.6.1 *DTD (Document Type Definition)*

DTDs são linguagens de esquema extensíveis, isto é, desenvolvedores podem adicionar novas definições, tornando-as ainda mais poderosas. Como premissa básica, *DTDs* devem facilitar a produção e a publicação de documentos, tendo em vista as mais variadas formas (*HTML*, *RTF*, *PostScript*, etc) em que estes são disponibilizados na rede. Além disso, devem prover uma interface intuitiva e eficiente para o processo de estruturação de documentos, não sendo, portanto, muito amplas ou complicadas (W3C, 2004).

Como a linguagem *XML* provê uma forma independente de compartilhar dados, diferentes grupos de pessoas podem utilizar um esquema comum para esta troca de dados. Dessa forma, uma aplicação pode usar um padrão *DTD* para verificar se os dados recebidos de outra aplicação são válidos (REFSNES, 2006).

As *DTDs* são opcionais e os dados enviados com uma *DTD* são conhecidos como dados *XML* válidos. Um analisador de documentos pode verificar os dados e se as regras contidas no *DTD* foram atendidas, isto é, se os dados foram estruturados corretamente (W3C, 2004), (FURTADO, 2006).

2.6.2 *XSD (XML Schema Definition)*

A *XSD* foi criada a fim de oferecer um maior controle sobre os tipos e os padrões de dados, tornando-se uma linguagem mais atraente e, contemplando, por exemplo, conceitos de orientação a objetos, como herança e polimorfismo (MELLO, 2006).

A *XSD* permite registrar as restrições impostas sobre os dados, levando-se em consideração o domínio do problema. Além disso, torna possível o registro de limites numéricos, conjuntos e ordens de listas (ANDERSON, 2001).

A *XSD* oferece, portanto, uma alternativa interessante para a descrição formal dos elementos em um documento *XML*, mas não deve ser considerada uma substituta da *DTD*, que continua possuindo muitos pontos fortes, como tamanho compacto, sintaxe conhecida e simplicidade (RAY, 2001).

DTDs ainda são amplamente utilizadas e a grande maioria das ferramentas (*parsers*) que realizam a validação de documentos *XML* continuam operando sobre *DTDs* (MELLO, 2006).

2.7 *APIs* para Processamento da *XML*

Desde o surgimento da *XML*, inúmeros produtos para *XML* foram criados, como validadores (*parsers*), editores, navegadores (*browsers*), etc. Todos estes produtos compartilham características comuns: trabalham com arquivos, analisam *XML* e utilizam marcadores. Sendo assim, tornou-se necessária também a criação de interfaces de programação de aplicações (*APIs*) para o processamento de documentos *XML*.

Uma *API* é uma base para o desenvolvimento de programas, que trata do material de baixo nível, para que o desenvolvedor possa concentrar-se no conteúdo real do programa. Uma *API XML* preocupa-se com a leitura dos arquivos, a análise e o redirecionamento de dados para manipuladores de eventos, enquanto o desenvolvedor limita-se a apenas escrever as rotinas de tratamento destes eventos (RAY, 2001).

2.7.1 *DOM (Document Object Model)*

DOM é uma plataforma e uma linguagem de interface neutra que permite que programas e *scripts* acessem e atualizem dinamicamente o conteúdo, a estrutura e o estilo dos documentos. O documento pode ser ainda processado e o resultado deste processamento pode ser incorporado ao documento atual (DOM, 2006).

É uma tecnologia que disponibiliza uma *API (Application Programming Interface)* com métodos de acesso e manipulação de documentos *XML*. Diversos

validadores de documentos *XML* suportam a *API DOM* e alguns *browsers* são capazes de processar instruções *DOM* definidas em *scripts HTML* (MELLO, 2006).

As interfaces são definidas para os diferentes objetos *DOM*, permitindo que o desenvolvedor navegue na estrutura do documento e faça leitura, pesquisa, alteração, inclusão e exclusão de seus elementos. Quando o *DOM* é utilizado em um arquivo *XML*, este analisa o arquivo, "quebrando-o" em elementos individuais, atributos, comentários, instruções de processamento, e assim por diante. Em memória, é criada uma representação do arquivo *XML* e o desenvolvedor pode então, através desta árvore de nós, acessar o conteúdo do documento e executar as operações que achar necessário (ANDERSON, 2001).

2.7.2 *SAX (Simple API for XML)*

SAX foi originalmente desenvolvida para Java. Trata-se da primeira *API* amplamente adotada para processamento de documentos *XML*, tornando-se um padrão de fato. É uma interface comum implementada em diferentes validadores *XML*, assim como, comparativamente, o *JDBC (Java Database Conectivity)* é uma interface comum implementada para diferentes tipos de bancos de dados relacionais (SAX, 2007).

SAX é baseado em um modelo controlado por eventos, usados para tratar do processamento. Não existe, desta forma, uma representação em árvore, de modo que o processamento ocorre em uma única passada pelo documento. Como o programa não pode "pular" para lugares aleatórios no documento, o desenvolvedor perde a flexibilidade ao trabalhar em uma representação persistente de memória. No entanto, por outro lado, este processo acaba utilizando pouca memória e sendo executado em grande velocidade, tornando-o ideal para o processamento da *XML* no servidor.

Um programa controlado por eventos pode procurar palavras-chaves em um documento, imprimir conteúdo com formatação, alterar um documento *XML* e fazer a leitura de dados a fim de construir uma estrutura de dados mais complexa. Algumas limitações deste tipo de programa envolvem a reordenação de elementos em um documento, o tratamento de referências cruzadas e a própria validação de um documento *XML*, tendo em vista que a utilização de pouca memória acaba ignorando certos eventos (RAY, 2001).

2.8 Banco de Dados *XML*

Quando dados *XML* necessitam ser mantidos em um banco de dados, existe a necessidade de se gerenciar adequadamente estes dados. Novos desafios surgem em decorrência desta necessidade, uma vez que a tecnologia de banco de dados precisa ser adaptada para garantir o armazenamento e a manipulação de dados *XML* de forma eficiente (MELLO, 2006).

Um documento *XML* pode ser considerado uma coleção de dados, da mesma forma que um banco de dados. Porém, um dado *XML* possui uma natureza diferente de um dado convencional de um banco de dados, uma vez que um dado armazenado em um banco de dados é totalmente estruturado, enquanto que um dado representado em *XML* é um dado semi-estruturado. Em um dado tipicamente semi-estruturado, apenas parte de sua definição possui alguma estruturação e cada ocorrência de um mesmo dado possui um esquema particular e auto-descritivo (MELLO, 2006).

O fato de um dado *XML* ter uma organização mais ou menos semi-estruturada define duas categorias de documentos *XML* (BOURRET, 2006):

- **Documento *XML* Orientado a Dados (DOD)**: documento fracamente semi-estruturado, ou seja, possui uma estrutura mais regular e repetitiva. Documentos desta categoria são tipicamente usados para transferência de dados convencionais entre aplicações. Notas fiscais e relatórios de clientes e produtos são alguns exemplos;
- **Documento *XML* Orientado a Documentos (DODoc)**: documento fortemente semi-estruturado, ou seja, possui uma estrutura mais irregular e particular. Documentos desta categoria são tipicamente usados para o relato da linguagem natural, com destaque para alguns dados delimitados dentro do seu conteúdo. Livros e anúncios classificados são alguns exemplos.

O modo como os dados *XML* são gerenciados está intimamente relacionado a estas duas categorias de documentos. Aplicações que lidam com DODs geralmente utilizam banco de dados relacionais, uma vez que os dados *XML* são fortemente estruturados. Já aplicações que lidam com DODocs utilizam banco de dados específicos para o gerenciamento de dados *XML*, chamados banco de dados *XML* nativos (ou

simplesmente banco de dados *XML*). Estes bancos de dados utilizam mais efetivamente a tecnologia *XML* para a representação e o acesso a dados (BOURRET, 2006).

Um banco de dados XML envolve um conjunto de documentos *XML* e suas partes, mantido por um sistema com capacidade para gerenciar e controlar este conjunto em si e a informação representada por ele. Sendo assim, um banco de dados *XML* é muito mais que um repositório de documentos estruturados ou dados semi-estruturados, pois o gerenciamento de dados *XML* persistentes requer capacidade para tratar independência, integração, direitos de acesso, versões, visualizações, integridade, redundância, consistência, recuperação e padrões (SALMINEN & TOMPA, 2001).

Bancos de dados relacionais vêm sendo bastante utilizados para o armazenamento de dados *XML* quando o foco da aplicação está nos dados propriamente ditos e não no formato *XML* de organização hierárquica e ordenada de dados. Este é o caso, por exemplo, de aplicações que fazem intercâmbio de dados estruturados (dados geralmente mantidos em arquivos ou banco de dados) através da *Web* utilizando *XML*. Esta abordagem de gerenciamento tem a vantagem de utilizar a tecnologia de banco de dados, que apresenta bom desempenho de acesso a dados e utiliza linguagens de consulta declarativas, para a manipulação de dados *XML*. Por outro lado, os bancos de dados relacionais devem se preocupar com a forma como dados de documentos *XML* devem ser armazenados e com a definição de métodos de acesso específicos para estes dados (MELLO, 2006).

Um banco de dados *XML* define um modelo lógico específico para documentos *XML*, armazenando e recuperando dados de acordo com este modelo. Um modelo mínimo para dados *XML* deve conter a definição de elementos, atributos e conteúdo textual (PCDATA); e manter a ordem dos dados no documento. Um modelo completo deve ainda oferecer suporte à representação de conteúdo misto e dados altamente semi-estruturados (com estruturas alternativas e repetitivas). Um banco de dados *XML* é recomendado para aplicações que lidam com DODocs (sua natureza fortemente semi-estruturada torna complexo o mapeamento para um banco de dados relacional), ou aplicações que lidam somente com dados no formato *XML* (MELLO, 2006).

Um problema da aplicação de tecnologias tradicionais de banco de dados para o gerenciamento de documentos *XML* persistentes consiste nas características especiais do dado, não comumente encontradas em banco de dados tradicionais. Documentos

estruturados são geralmente unidades de informação complexas, consistindo de linguagens natural e formal, e possivelmente incluindo entidades de multimídia (SALMINEN & TOMPA, 2001).

As unidades como um todo podem ser importantes registros legais ou históricos. A produção e o processamento de documentos estruturados em uma organização podem criar um complicado conjunto de documentos e seus componentes, versões e variações, envolvendo tanto dados básicos quanto metadados. Assim, para armazenar documentos estruturados e dar suporte a aplicações típicas é necessário um repositório de documentos que possua as seguintes características:

- criação dinâmica de traduções;
- transformações automáticas;
- controle de acesso no nível de elementos;
- acesso a elementos (versionamento de componentes);
- versionamento intencional;
- descrições de mudanças legíveis;
- capacidades de pesquisa entendidas;
- documentos baseados em *workflow* (SALMINEN & TOMPA, 2001).

Além disso, a própria *XML* possui algumas exigências:

- as especificações relatadas pelo *W3C* que tratam as capacidades do *XML* devem se estender também ao desenvolvimento de soluções em banco de dados *XML*;
- a pretensão da *XML* é que esta seja utilizada especialmente na Internet. Referências em documentos *XML* destinam-se aos recursos da Internet e sistemas de banco de dados *XML* devem também incluir o gerenciamento desses recursos;
- um documento *SGML* sempre está associado a um *DTD*, e pode ser usado de maneiras diferentes para dar suporte ao gerenciamento de dados. Documentos *XML* nem sempre estão associados a um *DTD* (SALMINEN & TOMPA, 2001).

2.9 Linguagens de Consulta

O poder de um banco de dados relacional não está exatamente em sua capacidade de armazenamento ou indexação, mas em sua habilidade para utilizar uma linguagem de pesquisa estruturada *(Strutured Query Language – SQL)* a fim de retornar, a qualquer tempo, toda e qualquer informação. Utilizando-se esse mesmo exemplo, a *XML* seria simplesmente um formato para armazenamento de dados caso não possuísse uma maneira padrão para a recuperação de informações (EVANS, 2002).

Sendo assim, as linguagens XPath e XQuery foram criadas com o intuito de pesquisar dados em documentos *XML*. Ambas, portanto, podem percorrer um documento *XML*, definir predicados (expressões booleanas utilizadas como filtro) e retornar, como resultado, dados no formato *XML* (MELLO, 2006).

2.9.1 XPath

XPath é, em termos gerais, uma linguagem de pesquisa para documentos *XML*. Através de suas regras, pode-se recuperar subconjuntos complexos de dados em documentos *XML* (EVANS, 2002). Sua proposta é localizar partes de um documento *XML*, provendo facilidades básicas de manipulação de *strings*, números e dados booleanos.

XPath modela um documento *XML* como uma árvore de nós. Há diferentes tipos de nós, incluindo elementos, atributos e texto. XPath define um modo para computar uma *string* de valor para cada tipo de nó (XPATH, 2006).

Apesar de satisfazer os mais variados tipos de consultas, a linguagem XPath apresenta algumas limitações. Expressões XPath retornam somente posições de um documento *XML*, sendo incapazes de produzir resultados de consultas com uma estrutura diferente daquela existente no documento. Além disso, não é capaz de realizar agrupamentos e junções entre dados *XML* (MELLO, 2006).

2.9.2 XQuery

XQuery é uma linguagem declarativa para dados *XML*. Afirma-se que é uma linguagem mais evoluída, já que se baseia na XPath e corrige algumas de suas limitações. A estrutura básica para um comando XQuery é composta por um bloco com as instruções "*for-let-where-return*" (MELLO, 2006). Trabalha também com construtores de elementos, chamadas de funções e todo tipo de expressões: lógicas, aritméticas, condicionais, seqüenciais, etc (CHAMBERLIN, 2002).

No Quadro 2.2, pode-se verificar um exemplo de consultas em XPath e XQuery, permitindo a comparação entre a sintaxe e a expressividade de ambas.

Quadro 2.2 – Exemplo comparativo entre consultas em XPATH e XQuery.

Consulta em XPath	**Consulta em XQuery**

/publicacoes/artigo/titulo
//artigo/autor[nome = 'Maria']/titulo
//artigo[@versao > 1]/*/@nome
	for $art in /publicações/artigo
	let $v := $art/@versao *10
	where $art/autor/nome = "Maria"
	return {$art/titulo, $v}

XQuery é atualmente uma candidata a recomendação do *W3C*. Isto significa que, para torná-la oficial, desenvolvedores devem:

- basear-se na lista dos requisitos, descrição formal e do modelo de dados que fundamenta a linguagem, lista de funções e operadores, casos de uso que ilustram a sua aplicação, entre outras informações disponibilizadas pelo *W3C*;
- utilizar um conjunto de métricas, chamado *XQTS (XML Query Test Suite)*, que visa identificar os possíveis problemas referentes à utilização de seus softwares e esta documentação;
- retornar suas opiniões, confirmando se esta linguagem de pesquisa possui a interoperabilidade necessária a fim de tornar-se realmente um padrão viável para a pesquisa de dados em todo o mundo (QUERY, 2006), (CHAMBERLIN, 2002).

3 ONTOLOGIAS

Para que seja possível a comunicação entre duas pessoas, é necessário que haja uma linguagem comum a fim de que as partes possam trocar informações oralmente ou na forma escrita. Sendo assim, duas entidades (pessoas ou agentes de software) podem se comunicar somente se concordarem sobre o significado dos termos utilizados (ROCHE, 2003).

A ontologia, compreendida como um vocabulário de termos e significados comuns por um grupo de pessoas, é a solução para este problema. A utilização de ontologias deve-se em grande parte ao que estas prometem: "uma compreensão compartilhada e comum sobre algum domínio que pode ser comunicado entre pessoas e computadores" (ROCHE, 2003).

É um tanto curioso dizer que o principal objetivo da ontologia é normalizar o significado dos termos, sendo que o próprio termo ontologia não possui um entendimento exato de seu significado. De acordo com (NOY & MCGUINNESS, 2001), uma ontologia é uma descrição formal explícita de conceitos em um domínio do conhecimento, com propriedades de cada conceito descrevendo características, atributos e regras de restrições. Uma ontologia formada por um conjunto de instâncias individuais de classes constitui uma base de conhecimento, havendo uma linha muito tênue entre o fim da ontologia e o começo da base de conhecimento.

Classes descrevem conceitos no domínio e são o foco principal da maioria das ontologias. Por exemplo, uma classe "vinho" representa todos os vinhos. Vinhos específicos são instâncias desta classe. Uma classe pode ter subclasses que representam conceitos mais específicos do que a superclasse. Por exemplo, podemos dividir a classe "vinho" em tinto, branco e rosê. Ou podemos dividir esta classe em seco e suave, espumante e não-espumante (NOY & MCGUINNESS, 2001).

Propriedades descrevem as características das classes e instâncias. Por exemplo, as instâncias da classe "vinho" possuem propriedades que descrevem aroma, corpo, nível de açúcar, fabricante, e assim por diante. Assim, temos duas propriedades ao afirmar que o vinho "Château Lafite Rothschild Pauillac" é "encorpado" e produzido pela vinícola "Château Lafite Rothschild" (NOY & MCGUINNESS, 2001).

As ontologias encontram grande aplicabilidade nos mais variados domínios de aplicação do conhecimento e da engenharia de software (GRUBER, 1992). Com isso, é possível classificá-las em diferentes categorias de utilização:

- comunicação entre pessoas e organizações;
- interoperabilidade (comunicação) entre sistemas;
- engenharia de sistemas, para especificação, integridade e reusabilidade;
- gerenciamento do conhecimento, envolvendo recuperação de informação, gerenciamento de documentos, e sistemas de bases de conhecimento;
- tratamento de linguagem natural para análise semântica e léxica;
- aplicações relacionadas à Internet, comércio eletrônico e *Web* semântica (ROCHE, 2003).

Ontologias são comumente utilizadas na *Web* e podem servir para várias aplicações como, por exemplo, categorizar páginas na *Web* (ex.: *site* de busca Yahoo!) ou categorizar produtos (ex.: *site* de compras Amazon.com). Além disso, diversas áreas de conhecimento (ex.: medicina) têm empregado este recurso para produzir, padronizar e estruturar seu vocabulário na rede (NOY & MCGUINNESS, 2001).

As ontologias têm-se apresentado como fator determinante para garantir a interoperabilidade entre sistemas heterogêneos e aplicações de semântica para *Web* (CHOI et al, 2006).

A utilização de ontologias ocorre por diferentes razões:

- **compartilhar o conhecimento comum da estrutura de informação entre as pessoas ou agentes de software**: diferentes *sites* compartilham e publicam ontologias dos termos específicos ao seu contexto (ex.: medicina, engenharia, artes, etc). Desta forma, os agentes de computador podem extrair e agregar informações destes diferentes *sites* utilizando-as, por exemplo, para responder pesquisas de usuários;
- **reutilizar o conhecimento do domínio**: se um grupo de desenvolvedores cria uma ontologia em detalhes, outros podem simplesmente reutilizá-la em seus domínios. Além disso, várias ontologias existentes podem ser integradas a fim de serem utilizadas em um grande domínio, ou uma ontologia geral pode ser estendida com o intuito de descrever um domínio de interesse mais específico;

- **formalizar assuntos de domínio**: a formalização do conhecimento de domínio permite torná-lo público, facilitando desta forma, o seu acesso, utilização e divulgação entre os interessados em um determinado contexto;
- **analisar o conhecimento do domínio**: a disponibilização de um dicionário comum de termos torna possível a análise do conhecimento do domínio. Esta análise formal de termos é extremamente importante quando se deseja reutilizar ontologias e estendê-las (NOY & MCGUINNESS, 2001).

O desenvolvimento de uma ontologia é similar à definição de um conjunto de dados e suas estruturas, de forma que outros programas possam utilizá-los. Por exemplo, uma ontologia desenvolvida para descrição de vinhos, refeições e combinações de ambos pode ser utilizada por aplicações de gerenciamento de restaurantes para criar sugestões de vinho para o menu do dia, responder perguntas dos garçons e clientes, etc (NOY & MCGUINNESS, 2001).

Em termos práticos, o desenvolvimento de uma ontologia envolve:

- definição das classes da ontologia;
- organização das classes em uma hierarquia de subclasses e superclasses;
- definição das propriedades e descrição dos valores permitidos para estas propriedades;
- obtenção dos valores para as propriedades das instâncias (NOY & MCGUINNESS, 2001).

Estas representações ou modelos estruturados estão sendo empregados por um grande número de aplicações, uma vez que consideram as distinções de semântica utilizadas pelas pessoas e tornam possível a manipulação de informações variadas e complexas (DENNY, 2002).

A estruturação semântica proposta pelas ontologias difere da composição e formação da informação, por exemplo, utilizada em bases de dados relacionais ou em *XML*. Nestas bases de dados, todo o conteúdo semântico tem de ser capturado pela aplicação lógica. Ontologias, entretanto, utilizam uma especificação objetiva do domínio da informação, através de uma representação consensual dos conceitos e relacionamentos que caracterizam o modo como o conhecimento neste domínio é expresso (DENNY, 2002).

3.1 Características de Ontologias

Não somente os conteúdos das ontologias podem variar, como também sua estrutura e implementação. Tais diferenças entre ontologias estão diretamente relacionadas às seguintes características das ontologias (DENNY, 2002):

- **nível de Descrição:** ontologias podem ser descritas em diferentes níveis, possibilitando a construção de simples dicionários de termos à dicionários organizados em categorias, dicionários onde os termos são relacionados hierarquicamente ou dicionários completos, onde conceitos estão relacionados e remetem a outros termos;
- **escopo Conceitual:** ontologias também diferem quanto ao escopo e o propósito de seus conteúdos. A distinção mais evidente está entre ontologias que descrevem o domínio de uma área específica (ex.: medicina), e ontologias de nível superior que descrevem conceitos básicos e relacionamentos que são invocados quando uma informação sobre um domínio é expressa em linguagem natural;
- **instância:** todas as ontologias possuem uma parte que é chamada de terminologia. De modo geral, isto é análogo ao esquema de uma base de dados relacional ou documento *XML*. A terminologia define os termos e a estrutura da área de interesse da ontologia. A outra parte, o conteúdo, provê a ontologia com instâncias ou indivíduos que manifestam esta terminologia. Esta extensão pode ser separada da implementação da ontologia e mantida como uma base de conhecimento. A decisão específica de tratar um conteúdo como um conceito ou tratá-lo como um indivíduo é usualmente tomada pela própria ontologia;
- **linguagem de Especificação:** ontologias são construídas de diferentes formas e utilizando-se variadas linguagens, como linguagens de programação de lógica geral (ex.: Prolog). No entanto, é mais comum a utilização de linguagens que têm sido desenvolvidas especificamente para a construção de ontologias, como *OWL (Web Ontology Language)*, *OIL (Ontology Inference Layer)* e *RDF (Resource Description Framework)*.

Uma linguagem precisa ser suficientemente rica e expressiva no sentido de conseguir representar a variação e a complexidade do conhecimento proposto pela ontologia.

A crescente disponibilização de informação na *Web* tem impulsionado a utilização de ontologias e de linguagens específicas para construí-las. Linguagens baseadas nas tecnologias do *W3C* como *RDF (Resource Description Framework)* e *XML (eXtensible Markup Language)* têm sido, portanto, muito requisitadas (DENNY, 2002).

3.2 Construção de Ontologias

Uma ontologia é comumente construída da seguinte forma (DENNY, 2002):

- **definição do domínio do conhecimento:** levantamento dos recursos de informação necessários que definirão, de forma consensual e consistente, os termos utilizados formalmente para descrever os conteúdos do domínio. Estas definições devem ser coletadas a fim de que possam ser expressas através de uma linguagem comum utilizada pela ontologia;
- **organização:** definição de uma estrutura conceitual global do domínio, envolvendo a identificação dos conceitos principais do domínio e suas propriedades, identificação dos relacionamentos entre os conceitos, criação de conceitos abstratos através da organização de características, referências ou inclusão suporte a ontologias, distinguindo quais conceitos possuem instâncias, e aplicando outras diretrizes para a metodologia escolhida;
- **inicialização:** adicionar conceitos, relacionamentos e indivíduos para o nível de detalhe necessário, a fim de satisfazer os propósitos da ontologia;
- **verificação:** verificar as inconsistências sintáticas, lógicas e semânticas dos elementos da ontologia. Esta verificação pode resultar na definição de novos conceitos baseados em propriedades individuais ou relacionamentos de classe;
- **confirmação:** a ontologia deve ser disponibilizada no ambiente pretendido, a fim de ser verificada pelos especialistas e envolvidos com o domínio proposto.

3.3 Linguagens para Construção de Ontologias

Há uma grande variedade de linguagens para a definição de ontologias, desde linguagens naturais informais a linguagens rigorosamente formais. Alguns exemplos de linguagem são: Ontolingua, *Open Knowledge Base Connectivity (OKBC), Open Configuration and Management Layer (OCML), Simple HTML Ontology Extensions (SHOE), Resource Description Framework (RDF), XML-Based Ontology Exchange Language (XOL), Ontology Inference Layer (OIL), DARPA Agent Markup Language (DAML), Web Ontology Language (OWL),* etc. (ROCHE, 2003).

As linguagens lógicas são bastante adequadas aos objetivos da ontologia. Estas linguagens contam com sintaxe clara e semântica formal e provêem mecanismos de inferência de som. Assim, favorecem a criação de bases de conhecimento consensuais, coerentes, precisas e compartilháveis (ROCHE, 2003).

3.3.1 Linguagem *OWL*

A *XML (eXtensible Markup Language)* foi inicialmente projetada para enfrentar os desafios envolvendo o grande volume de informações disponibilizadas na *Web*. Aos poucos seu papel ganhou considerável importância ao tornar possível a troca de uma ampla variedade de dados entre diferentes computadores na *Web* (OWL, 2006).

A linguagem *XML* fornece uma sintaxe para documentos estruturados, entretanto, não possui qualquer regra de semântica para os significados dos documentos. Sendo assim, a partir da idéia da ontologia, o *W3C* desenvolveu o *RDF (Resource Definition Framework)*, um modelo de dados para objetos e seus relacionamentos. Este modelo utiliza a sintaxe *XML* e provê uma semântica simples capaz de integrar uma variedade de aplicações e possibilitar a troca de conhecimento na *Web* (RDF, 2006).

Em seguida, o *RDFS (Resource Definition Framework Schema)* surge como um esquema para descrever as propriedades e as classes dos recursos *RDF*, através de uma semântica para criação das hierarquias de cada um dos objetos e classes. *RDFS* é considerada uma linguagem de ontologia que abrange classes, propriedades e conceitos

de limite e domínio. Além disso, possui a habilidade para descrever subclasses e superclasses (AREF & ZHOU, 2005).

Apesar de todos estes atributos, *RDFS* não é totalmente satisfatória quando são necessárias características para descrever os recursos em um grande nível de detalhes. *RDFS* não é totalmente adequada, pois lhe faltam importantes relacionamentos entre as classes, como equivalências e redundâncias, bem como cardinalidade e características das propriedades (AREF & ZHOU, 2005).

Com o propósito de resolver estes problemas, duas linguagens baseadas na *RDFS* foram desenvolvidas quase que simultaneamente: *OIL (Ontology Inference Layer)* na Europa e *DAML (DARPA Agent Markup Language)* nos Estados Unidos. E, finalmente, após a submissão destas duas linguagens *(DAML+OIL)* ao *W3C*, surgiu a *OWL (Ontology Web Language)* (AREF & ZHOU, 2005).

A *OWL* utiliza a sintaxe da *RDFS* para expressar conceitos primitivos de ontologia como classes, relacionamentos e subclasses. Além disso, acrescenta novos vocabulários para descrever as propriedades e classes, tais como: as relações entre classes (por exemplo, desconexão), cardinalidade (por exemplo, "exatamente um"), igualdade, tipos ricos de propriedades, características de propriedades (por exemplo, simetria) e enumeração de classes. Sendo assim, a *OWL* expressa com mais facilidade os significados e as semânticas do que a *XML* e a *RDF*, além de representar habilmente os conteúdos na *Web* (OWL, 2006).

A linguagem *OWL* pretende ser um padrão de linguagem de ontologias para *Web*, unificando três importantes aspectos:

- modelagem de conhecimento;
- semânticas formais e suporte de raciocínio eficiente;
- sintaxe compatível com os padrões *Web* (ROCHE, 2003).

Os propósitos principais da *OWL* podem ser resumidos da seguinte forma:

- formalizar um domínio através da definição de classes e propriedades destas classes;
- definir indivíduos e propriedades declarativas sobre estas classes;
- racionalizar sobre estas classes e indivíduos até o nível permitido pelas semânticas formais da linguagem (AREF & ZHOU, 2005).

No Quadro 3.1, tem-se o exemplo de uma ontologia criada na linguagem OWL. A classe no exemplo representa um conceito denominado Conteúdo, que é caracterizado pelas propriedades descrição, palavras chave, tópico.

Quadro 3.1 – Exemplo de uma ontologia criada com *OWL*.

```
<owl:Class rdf:ID=”Conteudo”>
 <rdfs:subClassof>
  <owl:Restriction>
   <owl:onProperty>
    <owl:objectProperty rdf:ID=”estaContido”/>
     <owl:minCardinality rdf:datatype=”http://www.w3.org/2001/XMLSchema#int”>1
      <owl:equivalentClass>
       <owl:Class>
        <owl:one of rdf:parseType=”Collection”>
         <owl:Class rdf:about=”#material”/>
         <rdfs:subClassof rdf:resource=”html://www.w3.org/2002/07/owl#Thing”/>
         <rdfs:comment rdf:datatype=”http://www.w3.org/2001/XMLSchema#string">
          Essa classe representa o conceito Conteúdo. É caracterizada pelas
          propriedades: Descrição, palavrasChave, tópico e
          estacontido.</rdfs:comment>
        </owl:oneOf>
       </owl:Class>
      </owl:equivalentClass>
     </owl:minCardinality>
   </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
```

Ontologias já vêm sendo utilizadas para definir o domínio de conhecimentos há algum tempo e exemplos como o Protégé mostram a viabilidade dessa linguagem (NOY & MCGUINNESS, 2001). A busca e o compartilhamento de informações tornam-se cada vez mais importantes no universo da *Web*. Com o uso de ontologias é possível

fornecer um mecanismo formal que viabiliza o processamento semântico da informação, permitindo que o entendimento compartilhado de termos possa ser utilizado por diversas pessoas para o intercâmbio de informações. As ontologias também funcionam como uma ferramenta de consulta para o usuário, fornecendo mecanismos para restrições e relacionamentos entre os domínios. Neste contexto, a ontologia vem ao encontro do objetivo deste trabalho, que busca formas de compartilhar e permitir a fácil localização de informações mantidas por dispositivos computacionais.

4 REDES *PEER-TO-PEER (P2P)*

A Internet, por se tratar de uma rede pública, permite a utilização irrestrita de seus recursos a qualquer momento e por qualquer pessoa. Entretanto, as principais aplicações construídas sobre a Internet baseiam-se na arquitetura cliente/servidor, onde as informações são centralizadas em servidores e acessadas por clientes através da rede. A tecnologia *P2P* surgiu como uma alternativa a esta forma de utilização da Internet, apresentando como grande vantagem em relação à arquitetura cliente/servidor, o fato de ser descentralizada, permitindo a colaboração direta dos usuários, sendo estes os consumidores de recursos e também os responsáveis por disponibilizá-los na rede (ROCHA et al, 2004).

A construção de ambientes colaborativos utilizando a tecnologia *P2P* iniciou-se nos anos 90, com a popularização das aplicações voltadas para o compartilhamento de arquivos. Os equipamentos conectados neste ambiente formam uma rede virtual sobre a rede de dados subjacente (na maior parte dos casos, a Internet). Sistemas *P2P* trazem conectividade para as bordas da rede, permitindo que qualquer equipamento conectado se comunique e colabore com os demais. Dentre suas inúmeras aplicações, pode-se citar novas formas de distribuição e entrega de conteúdo, mensagens instantâneas, compartilhamento da capacidade de armazenamento e de processamento disponível em máquinas ociosas, entre outras (SADOK, 2003).

Várias razões tornam o modelo *P2P* uma alternativa realmente viável: oferece meios para agregar e utilizar recursos geograficamente distribuídos; minimiza a carga de trabalho dos servidores; maximiza a utilização das máquinas clientes, melhorando a performance geral da rede; o custo para a criação dos ambientes colaborativos é relativamente baixo; o investimento em hardware de alto desempenho é pequeno; a natureza descentralizada desses sistemas torna-os inerentemente robustos a falhas ou ataques intencionais; a alta escalabilidade permite tratar o crescimento do número de usuários e de equipamentos que se juntam à rede; entre outras (RIGHI et al, 2006), (SADOK, 2003), (LOO, 2003), (BALAKRISHNAN et al, 2003).

P2P refere-se a uma forma de computação distribuída que envolve um grande número de nós computacionais autônomos – os pares (*peers*) – que cooperam no compartilhamento de recursos e serviços. Estes pares formam redes lógicas,

estabelecendo ligações entre os diversos outros pares conhecidos ou descobertos na rede. Um usuário no sistema *P2P* efetua consultas que descrevem dados do seu interesse. As consultas são propagadas através da rede a fim de encontrarem os pares que fornecem os dados relevantes, retornando os resultados ao usuário (KOLONIARI & PITOURA, 2005).

4.1 Características das Redes *P2P*

As redes *P2P* são aquelas que possuem basicamente três características: auto-organização, comunicação simétrica e controle distribuído. Uma rede *P2P* auto-organizada adapta-se automaticamente à chegada, saída e falha dos nós. A comunicação é simétrica quando os pares agem tanto como clientes quanto como servidores. Já o controle distribuído ocorre devido a não existência de um ponto central de controle em toda a rede (RISSON & MOORS, 2006). Esta ausência de elementos centralizados faz com que os sistemas *P2P* tendam a ser imunes a censura, monopólios, regulamentos e outros exercícios atribuídos às autoridades centralizadoras (RIGHI et al, 2006).

As redes *P2P* apresentam uma grande escalabilidade, já que não existem elementos centralizadores que impeçam o crescimento da rede. Além disso, a arquitetura *P2P* não requer grandes investimentos em servidores de alto desempenho, visto que todos os dispositivos conectados à rede podem ser utilizados para processamento e armazenamento de informações.

Redes *P2P* apresentam-se como alternativa aos modelos de computação centralizados, onde há um único ou pequeno grupo de servidores e vários clientes. O modelo *P2P* não possui qualquer conceito de servidor, uma vez que todos os participantes são iguais, e cada nó possui capacidade de cliente e servidor (KOLONIARI & PITOURA, 2005), (DEMEURE, 2006).

O gerenciamento dos dados em uma rede *P2P* envolve quatro aspectos (SUNG et al, 2005):

- **localização dos Dados**: pares devem consultar e localizar dados armazenados em outros pares;

- **processamento da Pesquisa**: o sistema deve encontrar os pares que contribuem com dados relevantes e, desta forma, executar eficientemente a pesquisa;
- **integração de Dados**: quando os dados a serem compartilhados no sistema seguem diferentes esquemas, mesmo assim os *peers* devem ser capazes de acessar estes dados utilizando preferencialmente a mesma representação modelada na origem dos dados;
- **consistência de Dados**: se um dado está replicado ou armazenado temporariamente (*cached*) no sistema, é necessário manter a consistência desta replicação.

Os sistemas *P2P* são atrativos por diversas razões (BALAKRISHNAN et al, 2003):

- as barreiras para o início e crescimento de tais sistemas são baixas, pois geralmente não requerem arranjos administrativos ou financeiros especiais, ao contrário dos modelos de redes centralizados;
- os sistemas *P2P* oferecem uma maneira de agregar e empregar os recursos de computação e armazenamento em computadores através do Internet;
- a natureza descentralizada e distribuída de sistemas *P2P* dá-lhes o potencial de serem robustos a falhas ou a ataques intencionais, tornando-os ideais para o armazenamento a longo prazo e também para grandes processamentos;
- a entrada e a saída contínua de recursos do sistema confere às redes *P2P* seu aspecto altamente dinâmico (DEMEURE, 2006);
- como as atividades são executadas localmente, usuários podem impedir que informações sobre si mesmos sejam disponibilizadas, garantindo-lhes assim certa privacidade (DEMEURE, 2006).

Dentre os problemas relatados com este tipo de rede, pode-se listar (DEMEURE, 2006):

- não há garantia de desempenho das redes *P2P*, pois alguns nós podem ter, por exemplo, uma conexão de baixa qualidade;
- falta de controle sobre o conteúdo dos arquivos, tendo em vista a ausência de uma administração centralizada;

- deve haver uma grande preocupação com a segurança, já que qualquer nó pode conectar-se à rede sem uma validação prévia;
- dificuldade de regulamentação, já que qualquer usuário, independente de suas intenções, pode conectar-se à rede.

Ainda quanto ao problema referente à segurança, pode-se afirmar que algumas falhas de segurança podem ser evidenciadas em situações, como: exclusão de arquivos e diretórios no computador; leitura e gravação de arquivos no computador; execução de comandos ou de programas, chamadas de longa distância através de um modem ou linha telefônica; conexão a outros computadores e execução de operações ilegais, entre outras (LOO, 2003).

4.2 Modelos de Arquitetura

Segundo (SHENKER, 2002), (BENEVENUTO, 2005) em redes P2P, pode-se destacar três tipos de arquiteturas:

- **Centralizado**: tem um diretório hospedado em servidores centrais que é atualizado constantemente. Os nós na rede *P2P* lançam consultas ao diretório do servidor central para localizar os outros nós, que mantêm a informação desejada. Abordagens centralizadas não são escaláveis e apresentam pontos únicos de falha. Exemplo: Napster;
- **Descentralizado Estruturado**: estes sistemas não têm nenhum servidor central de diretório, mas têm uma quantidade significativa de estrutura. Estrutura significa que a topologia da rede *P2P* (isto é, o conjunto das conexões entre membros do *P2P*) é fortemente controlada e que os recursos compartilhados na rede estão colocados não em nós aleatórios, mas em posições específicas que permitirão consultas mais fáceis de serem efetuadas. Em sistemas "fracamente estruturados", esta localização dos recursos é baseada em sugestões; a rede *P2P* Freenet é um exemplo de tais sistemas (SHENKER, 2002). Em sistemas "altamente estruturados", a topologia da rede *P2P* e a localização dos recursos compartilhados são determinadas precisamente; esta estrutura fortemente controlada permite ao sistema satisfazer mais eficientemente as consultas;

- **Descentralizado Não-Estruturado**: estes são os sistemas em que não há nenhum diretório centralizado nem qualquer controle preciso sobre a topologia da rede ou localização dos arquivos. A rede é formada pelos nós que se juntam a esta com base em algumas regras fracas. A topologia resultante tem determinadas propriedades, mas a localização dos arquivos não está baseada em nenhum conhecimento da topologia. Para encontrar um arquivo, um nó consulta os seus vizinhos. O método mais típico de consulta é por inundação (*flooding*), no geral a consulta é propagada a todos os vizinhos dentro de um determinado raio. Estas redes não-estruturadas são extremamente flexíveis em relação aos nós que entram e que saem do sistema. Entretanto, os mecanismos atuais de busca geram grandes cargas nos participantes da rede. A rede Gnutella é um exemplo desse tipo de rede (SHENKER, 2002).

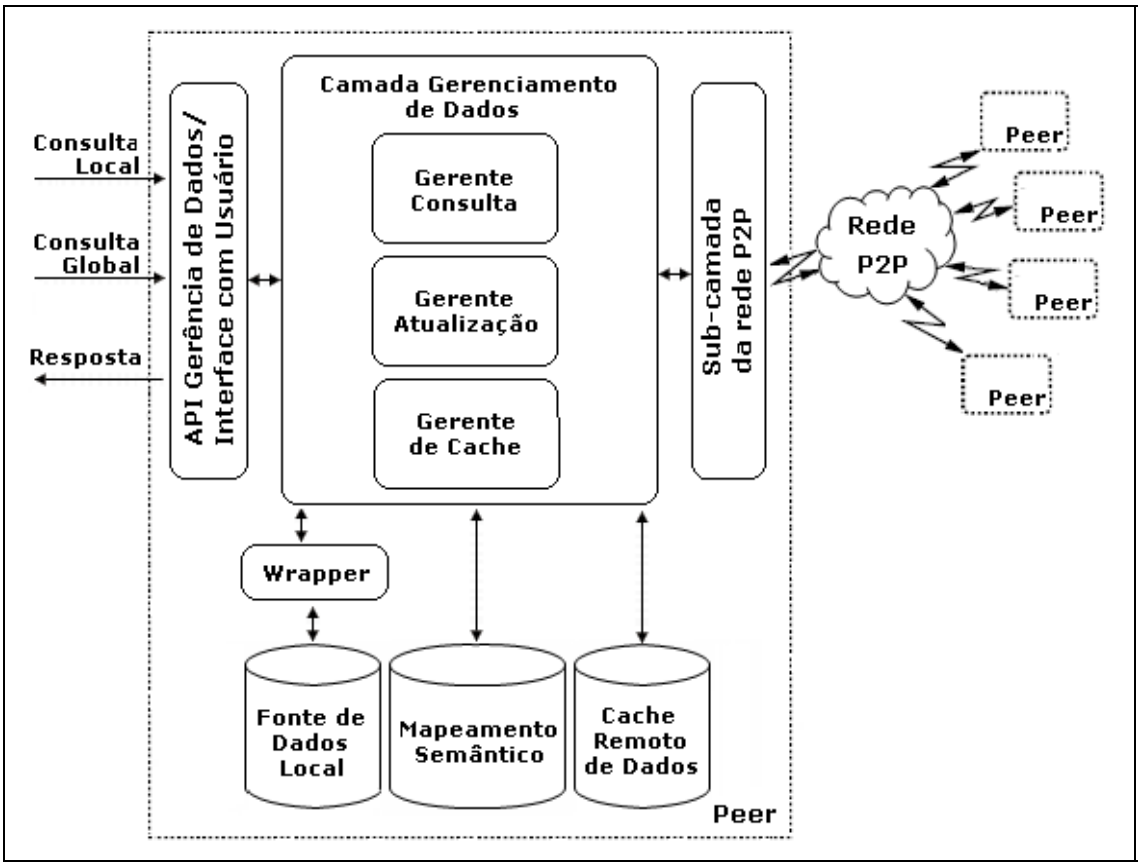

Figura 4.1 – Arquitetura *P2P* (SUNG et al, 2005).

A Figura 4.1 (SUNG et al, 2005) apresenta uma representação para uma arquitetura de redes *P2P* genérica. Dependendo da funcionalidade do sistema *P2P*, um ou mais componentes na arquitetura podem não existir, podem ser agrupados ou implementados por *peers* específicos. O aspecto essencial desta arquitetura é a separação das funcionalidades em uma interface usada para apresentar as consultas, uma camada para gerenciamento de dados que processa as consultas e metadados de informações, bem como uma subcamada de gerenciamento de conectividade à rede *P2P*.

4.3 Tipos de Pesquisa

Em (ROCHA et al, 2004), são apresentados três tipos de arquitetura baseados no sistema de buscas da rede: busca centralizada, busca por inundação e busca por tabela *hash* distribuída (*DHT*):

- **busca centralizada**: rede com um ponto central (possivelmente espelhado para outros pontos, dando a impressão de serem vários) de busca e nós que consultam o ponto central para trocar informações diretamente entre os nós;
- **busca por inundação**: rede com nós totalmente independentes, onde normalmente a busca é limitada à vizinhança mais próxima do nó que fez a busca (assim, a busca é escalável, mas não é completa);
- **busca por tabela *hash* distribuída (*DHT*)**: rede onde os nós têm autonomia e usam uma tabela *hash* para separar o espaço de busca entre eles.

4.4 Aplicações

Desenvolvedores têm ressaltado seis diferentes áreas de aplicação para a utilização de redes *P2P* (MELVILLE et al, 2002):

- **mensagem instantânea**: estas aplicações permitem conversa instantânea entre diversos usuários, além de compartilhamento básico de arquivos e mecanismos para trabalho colaborativo. Cada aplicação utiliza um índice central para armazenar a localização dos nós conhecidos, detalhes dos clientes e suas ações, assim como *backups* das rotas de comunicação para tratamento

de possíveis falhas. O cliente obtém o endereço no servidor, utilizando-o para comunicar-se diretamente com o nó destino (ex.: ICQ, MSN Messenger, AOL Instant Messenger, Yahoo! Messenger);

- **pesquisa distribuída e compartilhamento de arquivos**: permite o gerenciamento de conteúdo distribuído. O conteúdo não está centralizado em um lugar, e permanece onde foi criado. Como em aplicações de mensagem instantânea, um índice central é usado e mantém um catálogo dos arquivos que estão disponíveis e onde estes estão localizados. Nós clientes acessam este índice para pesquisar um arquivo desejado e, uma vez encontrado, obtêm o endereço deste nó. Em seguida, uma conexão direta é feita entre os dois nós e os arquivos são baixados (ex.:Napster, KaZaA, Mopheus);
- **publicação de conteúdo na *Web***: aplicações *P2P* tornam a rede mais "publicável", permitindo que qualquer pessoa hospede *sites* e áreas de discussão e escreva suas próprias páginas *web* (ex.: Browseup, Blogger);
- **trabalho colaborativo e comunidades na rede**: também conhecido como "*Groupware*", conta com uma rede de computadores com canais abertos de comunicação entre pessoas e compartilhamento de dados. Um software colaborativo permite melhorar a produtividade de usuários com objetivos ou interesses comuns, como: envio de e-mails, execução de projetos e aplicações, compartilhamento de arquivos e monitoramento de transações (ex.: Groove);
- **computação distribuída**: *P2P* provê uma infra-estrutura ideal para aplicações de computação distribuída, onde em vez da computação ser executada por um nó, esta é propagada para múltiplos nós na rede. Assim uma aplicação contraria a idéia de serviços sendo executados em um servidor central e o cliente fazendo uma requisição destes serviços. Aqui o serviço é desassociado de um servidor principal e propagado para um conjunto de nós clientes participantes (ex.: Seti@home);
- **jogos**: uma infra-estrutura *P2P* provê uma base para a comunidade de jogos *on-line* que não são controlados centralmente (ex.: CenterSpan).

Outras aplicações em potencial para redes *P2P* incluem: *multicasting*, mecanismos de pesquisa, sistemas baseados em agentes, sistemas de aviso e sistemas de espelhamento (MELVILLE et al, 2002):

- ***multicasting***: os dados existentes em uma única origem podem ser replicados em outros nós da rede. Cada nó que contém o dado replicado comporta-se eficientemente como uma origem secundária. Ao disponibilizar o dado pela rede, reduz-se a possibilidade de congestionamento de um ponto em particular e, por sua vez, aumenta-se a qualidade do serviço;
- **mecanismos de pesquisa**: quando um usuário executa uma pesquisa, um nó pode pesquisar a si mesmo e então enviar um pedido para todos os outros nós conhecidos da rede. Cada nó pode repetir este processo, e a pesquisa expandir-se exponencialmente;
- **sistemas baseados em agentes**: o paradigma *P2P* é similar aos usados em alguns sistemas de agentes distribuídos, com um software agente distribuído nas máquinas dos usuários. Um *framework* desenvolvido para *P2P* poderia potencialmente prover mais suporte aos desenvolvedores destes sistemas;
- **sistemas de aviso**: com um software *peer* em cada nó, uma possível utilização de um sistema *P2P* é avisar o que outros nós ou usuários estão fazendo. Um exemplo simples desta utilização é o ICQ, que informa a você quando um usuário está conectado ou não;
- **sistemas de espelhamento**: uma vez que o sistema execute a partir de um único nó, múltiplas instâncias do sistema são criadas e executadas em outros nós. Caso aconteça uma falha neste nó, então é possível trocar para outro nó, garantindo a operação contínua do sistema.

Além das aplicações citadas, vislumbra-se a possibilidade, que começa a ser explorada no meio científico, que é o compartilhamento de dados/informações em redes *P2P*. Explorar essa possibilidade é um dos objetivos deste trabalho.

4.5 *JXTA*

JXTA é um projeto *open source*, desenvolvido pela *Sun Microsystems* em abril de 2001, de protocolos *P2P* baseados em mensagens *XML* para o desenvolvimento de aplicativos distribuídos. *JXTA* permite que qualquer dispositivo conectado a uma rede, independente de sua plataforma, natureza, ou protocolo de rede, possa interagir,

compartilhar recursos, e formar uma rede distribuída, descentralizada e cooperativa. (GRADECKI, 2002).

O nome vem de *juxtapose*, uma referência ao modo como aplicações *P2P* são construídas sobre uma rede pré-existente, confrontando-se com os modelos tradicionais, baseados no paradigma cliente/servidor. *JXTA* não é uma linguagem de programação, mas uma especificação que tem implementações em diversas linguagens como C e Java.

O objetivo do *JXTA* é facilitar o desenvolvimento de aplicativos *P2P*, encapsulando funcionalidades e serviços comuns, escondendo a complexidade das implementações para o desenvolvedor de aplicativos, podendo esse se preocupar somente com a lógica e os detalhes pertinentes a sua aplicação (GRADECKI, 2002).

A idéia básica do *JXTA* é formar uma camada de rede virtual, conforme Figura 4.2 (GRADECKI, 2002), independente da rede real utilizada em cada dispositivo, provendo transparência da rede utilizada pelos dispositivos e permitindo que mesmo dispositivos conectados indiretamente à internet através de um *proxy*, ou por um *gateway* que realiza o *NAT (Network Address Translation)*, ou até mesmo com restrições de um *firewall*, possam participar da rede oferecendo serviços. Nessa situação, todo o roteamento necessário para a comunicação de *peers* entre redes distintas é realizado de forma transparente, utilizando-se de *peers* intermediários que possam encaminhar as requisições entre redes. A especificação *JXTA* foi concebida para ser independente de uma linguagem específica de programação (por exemplo, C ou Java), plataformas e sistemas operacionais (por exemplo, Windows e LINUX), e mesmo dos protocolos de rede utilizados (como TCP/IP ou *Bluetooth*). A forma básica de comunicação entre os participantes dessa rede é através de mensagens *XML* padronizadas, chamadas *advertisements*, respeitando os protocolos definidos nas especificações definidas no projeto (GRADECKI, 2002).

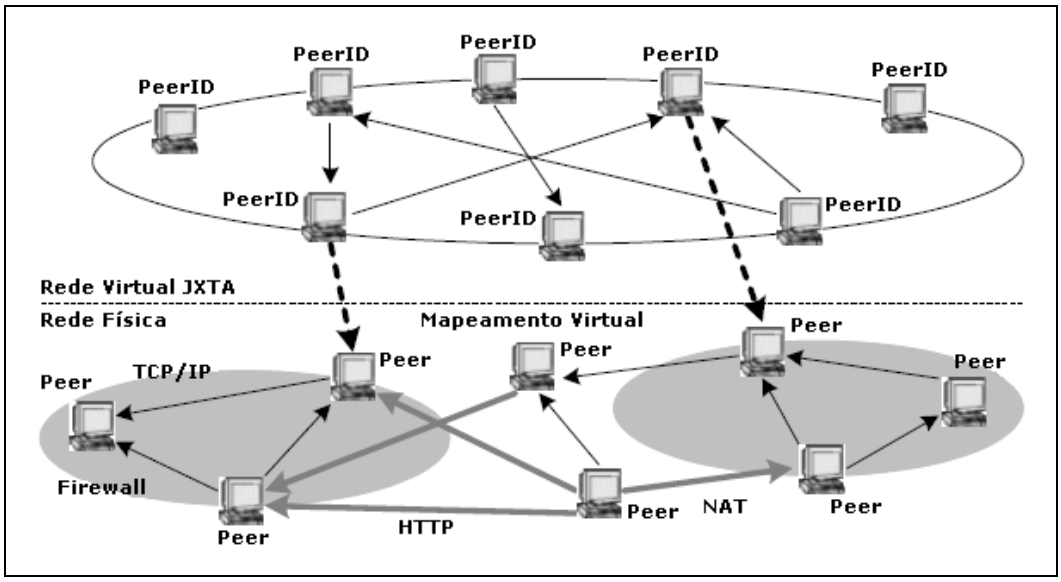

Figura 4.2 – Rede Virtual do JXTA.

A existência de uma grande diversidade de dispositivos tecnológicos, como por exemplo, celulares, *PDAs* e *desktops*, permitem o compartilhamento de muitas informações, mas quase sempre de uma forma indireta, com intermediários, através de servidores e dependendo de uma rede comum, como a Internet ou uma rede local. Utilizando *JXTA* podemos ter esses diversos dispositivos conectados sem a necessidade de uma estrutura centralizada de rede. O *JXTA* também permite que a topologia de rede mais adequada para uma aplicação seja definida por seus criadores, possibilitando o surgimento de novos serviços e funcionalidades. Cada aplicação possui requisitos de rede que nem sempre se encaixam nos modelos e arquiteturas tradicionais existentes. Em muitos casos precisamos utilizar mais de uma arquitetura simultaneamente para suprir os requisitos de uma aplicação, como por exemplo, utilizar uma comunicação típica cliente/servidor e, em alguns casos específicos poder comunicar diretamente dois *peers* sem a necessidade de utilizar um servidor como intermediário.

4.5.1 Rede *JXTA*

O *JXTA* define diversos protocolos, sendo cada um deles responsável por executar uma tarefa específica na rede *P2P* como, por exemplo, realizar o roteamento de mensagens e possibilitar a comunicação entre *peers*. Cada protocolo é definido por uma ou mais mensagens em formato *XML*, trocadas entre os *peers* participantes da rede. *JXTA* utiliza seis protocolos distintos, cujas funcionalidades são descritas na Figura 4.3:

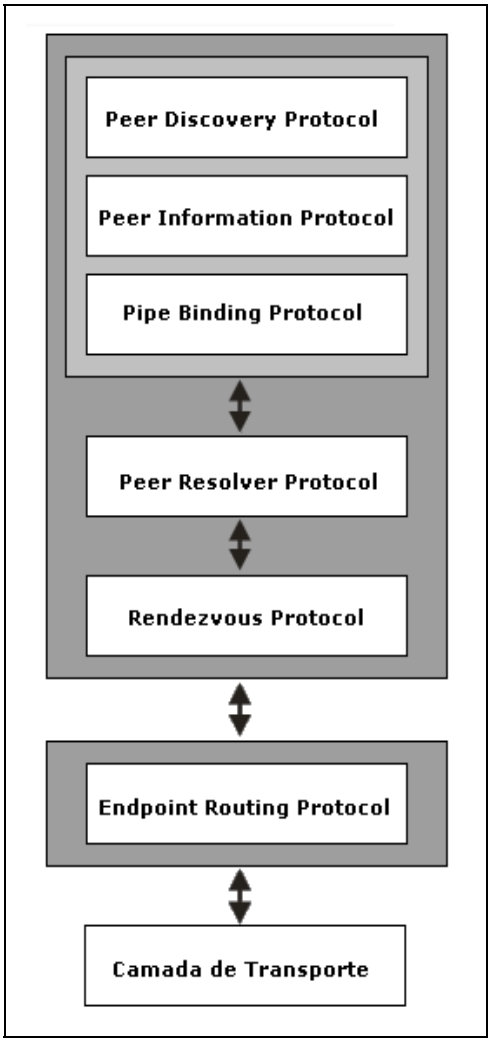

Figura 4.3 – Protocolo da Rede *JXTA*.

1. ***Peer Discovery Protocol (PDP)***: habilita um *peer* a procurar *advertisements* do tipo *peers*, *peers groups*, *pipes* e conteúdo, além de enviar os *advertisements* publicados pelo próprio *peer* para o restante da rede. Este é o principal canal de comunicação do *peer* com o restante da rede. A descoberta de novos *peers* na rede pode ser feita explicitamente, especificando o endereço de *ID* do *peer* ou do *peer group* desejado, neste último caso são retornados todos os *peers* pertencentes ao grupo;
2. ***Peer Information Protocol (PIP)***: responsável por obter informações de outros *peers*, como seu estado atual, tempo de resposta, utilização de recursos e tráfego atual. Permite enviar uma mensagem de *ping* para saber se um *peer*

ainda continua na rede ou não. Utiliza o *Peer Resolver Protocol* para enviar e propagar as requisições de informação;

3. ***Pipe Binding Protocol (PBP)***: responsável por realizar a conexão de um *peer* com um ou mais *peers*. Para cada *peer* destino com o qual uma conexão virtual é estabelecida, um *pipe* de entrada e saída é aberto (isto é, é realizado um "*bind*"), permitindo assim comunicação assíncrona entre os participantes. Também utiliza o *Peer Resolver Protocol* para enviar e propagar a requisição de *bind* dos *pipes*;
4. ***Peer Resolver Protocol (PRP)***: habilita um *peer* a enviar e receber pesquisa (*query*) de um ou mais *peers* e também a enviar informações sobre o próprio *peer* para a rede. Cada *query* é associada a um *handler*, responsável por processar a mensagem e aguardar as respostas resultantes da *query* a serem enviadas pelos *peers*. *Peers* que provêem compartilhamento de arquivos podem oferecer uma busca avançada de seu repositório de dados para os outros *peers;*
5. ***Rendez-Vous Protocol (RVP)***: são *peers* responsáveis por indexar informações sobre outros *peers* conhecidos. Este *peer* auxilia na busca por um *peer* específico ou no roteamento de mensagens propagadas na rede. Caso não haja nenhuma informação disponível em seu índice, a requisição é passada adiante para outro *rendezvous peer*, e assim por diante. *Rendezvous peers* diminuem o tráfego de mensagens enviadas e otimizam o processo de busca de recursos, evitando que a busca propague-se por toda a rede;
6. ***Endpoint Routing Protocol (ERP)***: responsável por informar a rota para comunicação entre dois *peers* distintos quando uma conexão direta não é possível entre eles. A comunicação direta não será possível quando dois *peers* utilizarem redes de protocolos diferentes ou quando entre estes há um *firewall* ou a conexão é realizada por *NAT*. Nesse caso, o *Endpoint Routing Protocol* informa a rota necessária, indicando os *gateways* necessários para realizar a conexão. Caso a rota entre dois *peers* mude, ou a topologia da rede mude por qualquer motivo, esse protocolo será utilizado para indicar a nova rota de comunicação entre eles. Qualquer *peer* pode ser um roteador desse tipo, implementando o *Endpoint Routing Protocol*.

4.6 Exemplos de Aplicações de Redes *P2P*

4.6.1 Napster

O Napster é um exemplo de rede *P2P* que permite aos seus usuários se conectar diretamente aos computadores de outros usuários a fim de compartilharem arquivos de música.

Usuários do Napster baixam um software do próprio *site* Napster e instalam em seus computadores. O computador central (servidor) do Napster, conforme Figura 4.4, mantém diretórios de arquivos dos usuários que estão conectados na rede. Estes diretórios são automaticamente atualizados quando um usuário se conecta e desconecta da rede. Quando o usuário pesquisa um arquivo (passo 1 da figura 4.4), o servidor providencia a informação para que este possa estabelecer uma conexão direta com o computador do outro usuário que possui o arquivo solicitado (passo 2). O *download* do arquivo acontece entre os computadores dos usuários (passo 3), desviando-se do servidor e estabelecendo assim a conexão *P2P* (LOO, 2003), (RIGHI et al, 2006).

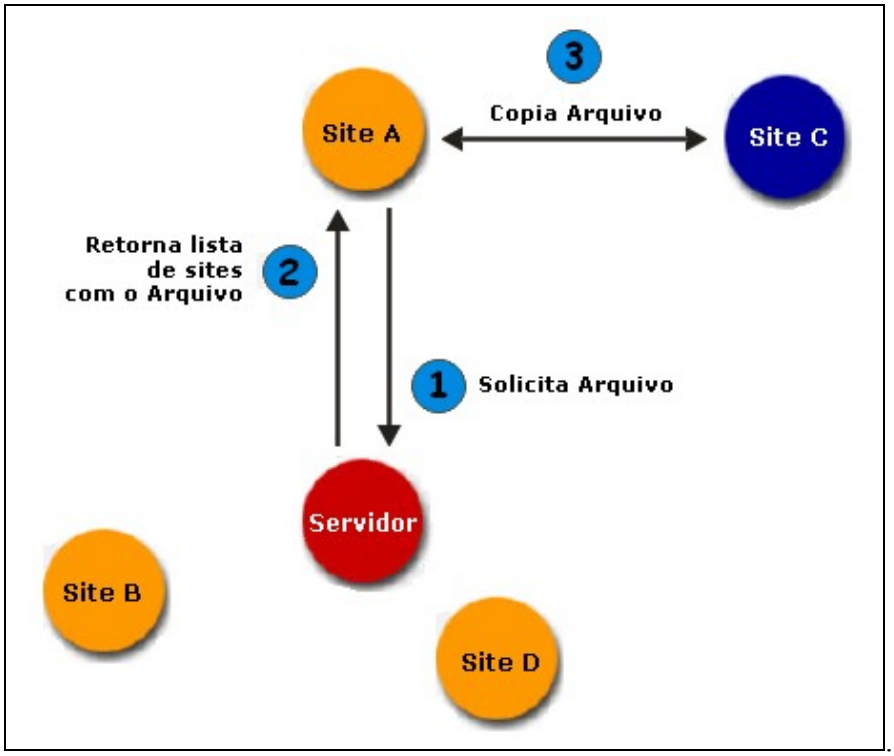

Figura 4.4 – Busca na rede Napster.

A força do Napster, e de aplicações similares, está em permitir o compartilhamento de um grande volume de informações dispersas por toda a rede sem a necessidade de armazená-las em um servidor de arquivos central. A limitação desse modelo está em permitir que seus usuários compartilhem apenas arquivos de música, e nenhum outro tipo de recurso (LOO, 2003), (RIGHI et al, 2006), (NAPSTER, 2006).

Esse modelo requer uma infra-estrutura de gerenciamento (o servidor de diretórios), que armazena informações sobre todos os participantes da comunidade. Tal aspecto pode gerar limites de escalabilidade ao modelo, uma vez que requer servidores maiores para atender a demanda de requisições, e mais espaço para armazenamento à medida que a quantidade de usuários cresce. Contudo, a experiência do Napster mostrou que esse modelo era bastante robusto e eficiente.

Preocupadas com a violação de direitos autorais que a troca de arquivos proporcionava, diversas entidades resolveram processar a Napster Inc., entre estas a Associação de Gravadoras dos EUA (RIAA). Após longas batalhas judiciais, um acordo foi feito e a Napster bloqueou o acesso a músicas cujo direito autoral estaria protegido pelas associações que o processaram. Por fim, a Napster acabou sendo comprada por um grupo de gravadoras.

Existe ainda uma variedade de aplicações *P2P* semelhantes ao Napster, como: Dmusic, Audiogalaxy, MyNapster, Wippit, entre outras (ROCHA et al, 2004).

4.6.2 Gnutella

O Gnutella é considerado a primeira solução puramente *P2P*. O objetivo inicial do projeto era desenvolver um aplicativo de compartilhamento de arquivos totalmente descentralizado e que sobrevivesse a qualquer tipo de intervenção. Para que isso fosse possível, a rede formada pelos usuários do Gnutella não foi projetada com um servidor central. Dessa forma, a busca é realizada em cada *peer* participante da rede, utilizando mecanismos de inundação (*flooding*), isto é, enviando a consulta para todos os nós conhecidos de um *peer*; esses nós reenviam a mesma consulta para todos os seus nós conhecidos, com objetivo de propagar a consulta para o maior número de *peers* conhecidos.

Conforme a Figura 4.5, cada nó nessa rede é responsável por manter a lista de nós conhecidos, buscar novos *peers* na rede e encaminhar cada requisição para os *peers* conhecidos, até que um deles responda avisando que possui o conteúdo desejado. Nesse caso, as respostas percorrem o caminho de volta, identificando o *peer* que possui o conteúdo. Toda e qualquer requisição de consulta é armazenada em *cache*, isto é, armazenada localmente pelos *peers* para uma futura consulta, evitando que as informações sejam desperdiçadas e que requisições duplicadas sejam reenviadas. Além disso, somente a identidade do último *peer* que encaminhou a requisição é conhecida, garantindo a privacidade na busca de conteúdo. Somente o conteúdo que fez a requisição sabe que uma determinada consulta pertence a este. Para os participantes da rede, não há como distinguir se o *peer* que encaminha uma consulta é realmente o que originou essa consulta ou simplesmente mais um *peer* encaminhando a consulta, dando assim a continuidade do processo de pesquisa do conteúdo requerido. Na figura 4.5, o *site* A envia uma solicitação ao *site* B (passo 1), que registra a solicitação (passo 2) e retransmite aos sites conhecidos (passo 3), o retorno dos das informações são recebidas (passo 4) e consultadas na lista de solicitações (passo 5) para identificar a quem deve ser feito o retorno da informação (passo 6).

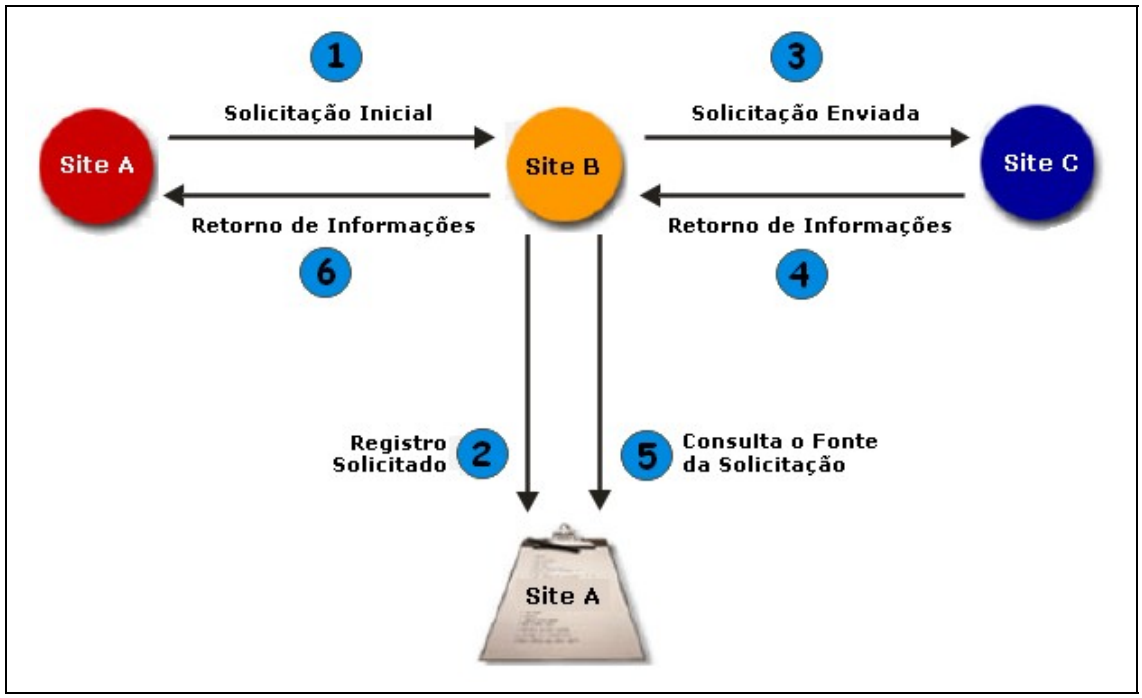

Figura 4.5 – Propagação de busca na rede Gnutella (JÚNIOR, 2003).

As redes Gnutella apresentam um problema de consumo de banda e demora na realização de buscas. Isso acontece devido à total descentralização da rede, o que obriga a repetição das mensagens de busca enviadas. Cada mensagem de busca contém um *TTL (Time To Live)*, que caracteriza o número de saltos máximos entre *peers* que a mensagem deve ser replicada na rede. O padrão utilizado no Gnutella é de 256 saltos para cada busca realizada. A cada *peer* em que a busca é encaminhada, esse valor é decrementado, evitando que uma mensagem seja propagada infinitamente pela rede (SUNG et al, 2005). Apesar do consumo de banda excessivo e da demora na obtenção de resultados, as buscas são sempre efetivas e o conteúdo, caso disponível na rede, quase sempre encontrado. Caso o conteúdo procurado seja encontrado, toda e qualquer transferência de dados é realizada diretamente entre o *peer* que contém o conteúdo e o que iniciou a busca, pois ambos já conhecem os endereços de rede utilizados, caracterizando uma comunicação totalmente *P2P*.

A grande vantagem dessa arquitetura totalmente descentralizada, em relação à abordagem cliente/servidor, é a impossibilidade de se interromper o funcionamento da rede caso um dos *peers* participantes saia da rede por algum motivo. A censura de um determinado conteúdo torna-se praticamente impossível dessa forma, e o anonimato e a privacidade dos publicadores e compartilhadores de conteúdo são garantidos. No entanto, na comparação da eficiência e a rapidez da busca com a arquitetura cliente/servidor utilizada pelo Napster, conclui-se que esta última leva uma grande vantagem (ROCHA et al, 2004).

4.6.3 SETI@HOME

O grande poder de processamento dos atuais computadores pessoais e a ociosidade da maioria desses computadores, possibilitam o surgimento de aplicações com um processamento realmente distribuído, espalhadas pela Internet. O projeto SETI@HOME (*Search for Extraterrestrial Intelligence*), com objetivo de identificar vida inteligente fora do planeta Terra, é um exemplo bem sucedido deste tipo de aplicação. Ao instalar a proteção de tela do projeto, um usuário passa a contribuir com parte dos cálculos para análise de sinais de rádios, capturados diariamente por potentes

telescópios. Depois de realizada a análise dos dados obtidos, quando o computador está ocioso, os resultados são enviados para os servidores localizados na Califórnia (EUA), que mantêm centralizados todos os resultados obtidos. Para realizar a análise desses dados em um único computador, uma capacidade de processamento muito grande seria necessária por um longo tempo e o custo para realização dessa tarefa poderia inviabilizar o projeto. A utilização de recursos ociosos espalhados pela Internet, de forma gratuita, era talvez uma das poucas soluções disponíveis que viabilizariam o projeto (*site* SETI@home). Os diversos resultados obtidos das análises realizadas, e a evolução do projeto, podem ser conferidos no *site* http://setiathome.ssl.berkeley.edu.

O projeto SETI@HOME foi um dos pioneiros na distribuição de tarefas para processamento distribuído pela Internet, e a quantidade de resultados obtidos contribui para a análise de vida extraterrestre, além de possibilitar que diversos usuários anônimos pela Internet possam dar sua contribuição em termos de recursos para o projeto.

4.7 Considerações Finais

Redes *P2P* apresentam uma comprovada aplicabilidade no compartilhamento de arquivos e na realização de computação distribuída. Os exemplos de aplicações descritos na seção 4.6 demonstram formas de utilização das redes *P2P* nesse contexto. No entanto, redes *P2P* também podem ser utilizadas em outro contexto: no compartilhamento de dados mantidos de forma distribuída, por exemplo. O próximo capítulo investiga a utilização de redes *P2P* com esse propósito, objetivando obter um ambiente no qual dados possam ser compartilhados em uma rede composta por dispositivos heterogêneos, incluindo dispositivos móveis com limitada capacidade de armazenamento, baixo poder de processamento e com acesso limitado e intermitente à rede.

5 COMPARTILHAMENTO DE DADOS EM REDES *P2P*

As redes *P2P* vêm sendo utilizadas principalmente para compartilhamento de arquivos e para efetuar computação distribuída, mas suas potenciais aplicações são muitas. Uma importante forma de utilização dessas redes, ainda pouco investigada, é a sua utilização para compartilhamento de dados.

O presente capítulo apresenta um estudo analítico de quatro trabalhos descritos na literatura, que propõem diferentes arquiteturas de compartilhamento de dados baseadas em redes *P2P*. Tal estudo foi realizado com o intuito de identificar as características desejáveis e as limitações desses trabalhos, tendo em vista os objetivos definidos nessa dissertação.

5.1 DBGlobe

O DBGlobe (PITOURA et al, 2003) é um sistema *P2P* de gerenciamento global para descrever, manter e consultar dados. DBGlobe permite a conexão de dispositivos anônimos e móveis, chamados de *Primary Mobile Objects (PMO)*, que armazenam informações sobre um contexto e conectam-se a um *Cell Administration Server (CAS)*, um servidor que disponibiliza conexão e provê serviços básicos aos *PMOs* para publicar e descobrir essas informações. O DBGlobe cria uma rede *P2P* descentralizada estruturada (conforme a classificação apresentação na seção 4.2) e utiliza ontologias para descrição dos dados compartilhados na rede. Os *PMOs*, conforme sua capacidade, podem ser utilizados para disponibilizar, mover, atualizar ou simplesmente transferir dados. As informações são trocadas entre os *PMOs* utilizando *web services*. Os serviços disponibilizados usam o paradigma de *Active XML* (PITOURA et al, 2003) para invocar e executar serviços e trocas de dados.

Apesar de permitir a conexão de dispositivos móveis à rede *P2P*, não existe no DBGlobe a possibilidade de ter acesso aos dados publicados por um *peer* enquanto este se encontra desconectado da rede.

5.2 Bricks

Bricks (RISSE et al, 2005) é um projeto com o objetivo de desenvolver e manter uma infra-estrutura para permitir o compartilhamento e uso de recursos de conhecimento cultural, por exemplo, bibliotecas digitais, sobre uma rede *P2P* descentralizada estruturada.

Na topologia da rede, cada nó, denominado *workstation*, representa uma instituição membro, e tem instalado um software para permitir o acesso ao Bricks. Alguns desses nós, chamados BNodes, armazenam os conteúdos e gerenciam os metadados. Cada BNode conhece apenas um subconjunto de outros BNodes. Assim, se um BNode quiser alcançar outros membros não conhecidos, este deve solicitar a um membro conhecido que propague a requisição ao destino final ou para outro membro.

Uma base de dados *XML*, localizada nos BNodes, permite armazenar metadados que identificam os documentos *XML* utilizando uma tabela de *hash* distribuída (*DHT*). Os documentos são divididos em pequenas partes e são distribuídos para armazenamento na comunidade. Os documentos são criados e mantidos pela comunidade e podem ser acessados por qualquer *peer*, sem preocupar-se com a localização dos dados. O acesso aos dados é feito através de *web services*.

5.3 Piazza

Piazza (HALEVY, 2003) propõe uma infra-estrutura de rede *P2P* para construção de aplicações em *Semantic Web*. Ele oferece uma linguagem para mediação entre fonte de dados e *Semantic Web*, que mapeia tanto domínios como estrutura de documentos.

A arquitetura de Piazza consiste de vários nós, onde cada nó pode fornecer dados mapeados em um esquema, prover apenas um esquema (ou ontologia), ou ambos. Um simples nó pode apenas fornecer dados (possivelmente vindos de um banco de dados relacional); por outro lado, um simples nó também pode prover um esquema ou ontologia que outros nós podem mapear.

Piazza também permite a interoperação de dados *XML* com dados *RDF* associados a uma linguagem de ontologia, como a *OWL*. O mapeamento na arquitetura

Piazza é feito entre pequenos conjuntos de nós, e o algoritmo de processamento da consulta é construído para localizar e obter dados relevantes distribuídos pela rede Piazza.

5.4 KEEx

KEEx (BONIFÁCIO et al, 2004) é um sistema *P2P* que permite uma coleção de nós de conhecimento *(Knowledge Nodes – KN)* para pesquisa. Este fornece documentos em uma semântica básica sem pressupor previamente um acordo em que o documento deve ser classificado, ou sobre uma linguagem comum para representar informações semânticas dentro de um sistema.

No KEEx, *peers* são organizados em comunidades de conhecimento. A arquitetura KEEx está organizada em diferentes *peers*: *PKM peer (Personal Knowledge Manager)* e *Source peer* que representam o conhecimento. O primeiro permite a um usuário gerenciar o seu conhecimento local, ao mesmo tempo em que o compartilha com outros *peers* na rede. Cada *PKM peer* pode exercer dois papéis principais: *provider*, que é responsável pela publicação no sistema do corpo do conhecimento; e *seeker*, que pesquisa informações sobre a rede. O segundo é responsável por integrar os dados disponíveis na organização, classificando-os com base em uma taxonomia – *sites* de internet, ferramentas de gerenciamento de conteúdo, mecanismos de pesquisa, etc. Este difere do *PKM peer* por ser a maneira como a rede compartilha o conhecimento, que consiste apenas em fornecer regras. KEEx também fornece três tipos de S*ervice peer*. Um *Normalization peer* suporta a troca de conhecimentos entre *peers*, permitindo adicionar informações semânticas para categorizar o conteúdo e para descobrir relacionamentos semânticos entre diferentes conteúdos. Um *Super peer* fornece configurações e funcionalidades para administrar a rede. Por fim, um *Rendez-Vous peer* é responsável por permitir a comunicação entre *peers* em redes físicas diferentes ou separadas por um *firewall*.

Tabela 5.1 – Comparação entre arquiteturas de rede *P2P*.

Características	DBGlobe	Bricks	KEEx	Piazza
Topologia da Rede *P2P*	Descentralizada Estruturada	Descentralizada estruturada	Descentralizada estruturada	Descentralizada estruturada
Infra-Estrutura de Comunicação	*Web Services*	*Web Services*	*JXTA*	*Web Services*
Semântica dos Dados	Sim	Não	Sim	Sim
Dispositivos Móveis	Sim	Não	Não	Não
Desconexão de *Peers*	Não	Não	Não	Não
Campo de Aplicação	Genérico	Restrito	Genérico	Genérico

5.5 Considerações Finais

Redes *P2P* disponibilizam uma infra-estrutura que permite a construção de diversas aplicações distribuídas. Essa infra-estrutura tem sido utilizada principalmente para compartilhamento de mídias e para computação distribuída. O presente trabalho considera a possibilidade de utilizá-las como meio para compartilhamento de dados semi-estruturados, em formato *XML*, distribuídos em um ambiente composto por dispositivos heterogêneos, que possam possuir capacidade de armazenamento e poder de processamento limitado e que apresentem a possibilidade de mobilidade.

Os trabalhos encontrados na literatura, cujas principais características são sumarizadas na Tabela 5.1, não atendem aos objetivos traçados nesta dissertação. O projeto Bricks não emprega ontologias para descrever a semântica dos dados, o que poderia potencialmente prover uma maior facilidade de indexação e busca dos dados que a estratégia empregada, baseada em metadados armazenados em uma tabela *hash* distribuída. Adicionalmente, Bricks limita a sua utilização a um campo de aplicação específico, restringindo os dados que podem ser compartilhados na rede. Por sua vez, KEEx e Piazza não provêem suporte para conexão de dispositivos móveis à rede P2P,

limitando a sua utilização. Já DBGlobe, apesar de suportar a conexão de dispositivos móveis, não é capaz de lidar com a freqüente desconexão dos *peers*, o que pode resultar na indisponibilidade dos dados compartilhados.

No próximo capítulo, será apresentada uma arquitetura para compartilhamento de dados sobre uma rede *P2P* que reúne as principais características desejáveis para que dados sejam facilmente localizados e trocados entre dispositivos heterogêneos e móveis. Dessa forma, busca-se resolver as limitações encontradas nos trabalhos relacionados na literatura.

6 ARQUITETURA PROPOSTA

A arquitetura *XOP* (*XML data Objects in P2P*) apresenta uma proposta para formalizar conteúdos e conceitos relacionados a um domínio, e em seguida disponibiliza um mecanismo de armazenamento e pesquisa sobre uma rede *peer-to-peer (P2P)* descentralizada estruturada (SHENKER, 2002), (BENEVENUTO, 2005), a Figura 6.1 apresenta um exemplo da arquitetura da rede XOP, compostas por quatro servidores, chamados de SAI, onde cada um mantém conexão com seus nós, conhecidos como SI e entre eles. Nesse sentido, o primeiro passo é criar um mecanismo para que os usuários possam definir conceitos sobre seus conteúdos. Para tanto, a arquitetura fornece uma ferramenta que possibilita ao usuário construir uma ontologia que descreve, mantém e disponibiliza esses conteúdos. Além disso, a arquitetura prevê facilidades para permitir que *peers* móveis, tais como *PDAs* e celulares, possam compartilhar, pesquisar e recuperar dados, apesar de terem uma limitada conectividade à rede, além de pouca capacidade de armazenamento e de reduzido poder de processamento.

O projeto *XOP* é constituído por um conjunto de dispositivos que compõem a sua infra-estrutura, chamados Serviço de Administração de Informações (SAI), responsáveis por conectar nós anônimos, chamados Serviço de Informações (SI). O SAI é um super-nó na rede *P2P* e por isso deve possuir um maior poder de processamento e capacidade de armazenamento. O SAI também deve ser um dispositivo de alta disponibilidade, pois uma vez fora da rede, as informações nele armazenadas ficam indisponíveis. Os SAIs comunicam-se com seus SIs atendendo as solicitações de consulta ou armazenamento de dados e propagando as consultas para outros SAIs. O SAI tem três funções principais:

- funcionar como um super-nó da rede *P2P* a fim de agrupar e interligar um conjunto de *peers* e comunicar-se com outros super-nós;
- disponibilizar um serviço de armazenamento de conteúdos e mecanismos para localização de conteúdos;
- disponibilizar uma base de dados para armazenamento de informações, que será usada quando os SIs optarem por manter suas informações no SAI.

O SI é um *peer* na rede, ou seja, um dispositivo com capacidade de conectividade que realiza consultas ou disponibiliza informações na rede. Diferentemente do SAI, o SI

não necessita ter uma grande capacidade de processamento e armazenamento, podendo ser um simples *PDA* ou um telefone celular conectado à rede para efetuar uma consulta, sem nenhuma função de armazenamento de dados, ou ser um computador de um usuário doméstico disponibilizando informações para a rede. Também não é necessário que o SI tenha alta disponibilidade, podendo ser um dispositivo móvel que se conecta a rede apenas para efetuar uma consulta. A seguinte regra é utilizada para a comunicação entre SIs: um SI 'A' somente comunica-se com um SI 'B' após receber do SAI o endereço de 'B', que pode inclusive estar conectado a outro SAI. Um SI pode optar por servir de local de armazenamento da informação, caso tenha capacidade e deseje ter um maior controle sobre a informação, ou pode armazená-la diretamente no SAI ao qual está conectado. Um SI pode se conectar a qualquer momento na rede, não necessitando de nenhuma permissão especial, bastando que este tenha o serviço instalado localmente.

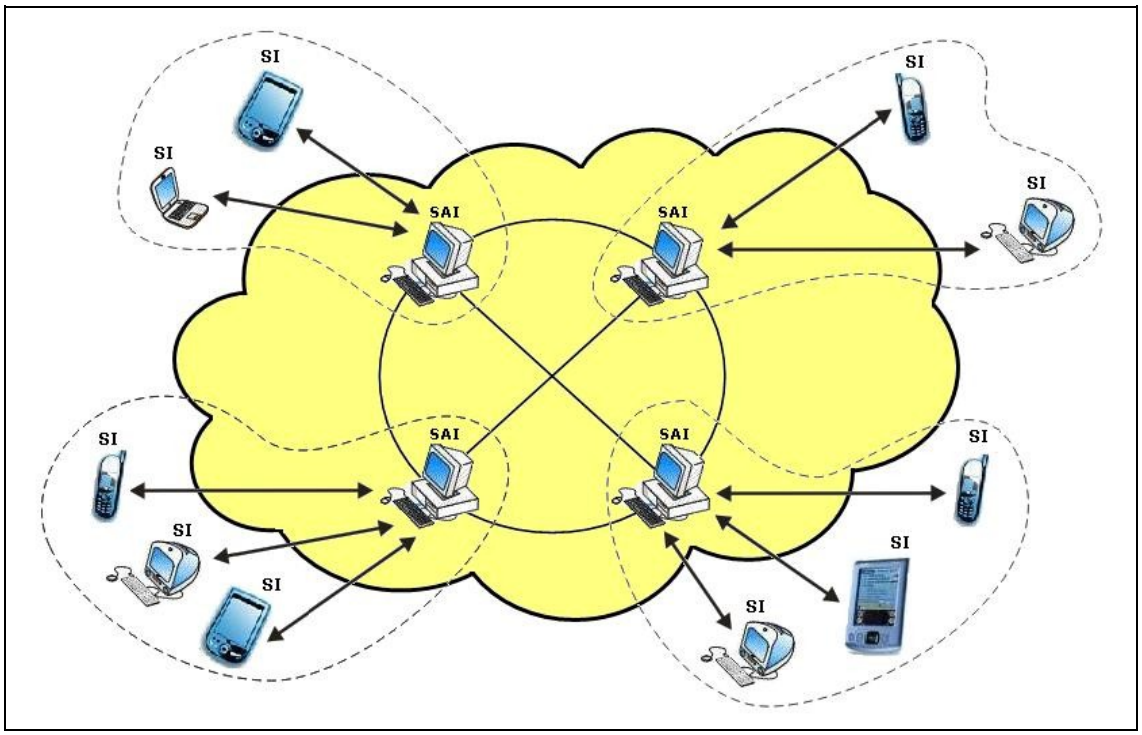

Figura 6.1 – Arquitetura *XOP.*

6.1 Serviço de Administração de Informações (SAI)

O **Serviço de Administração de Informações (SAI)**, apresentado na Figura 6.2, funciona como um super-nó de uma rede *P2P*, possuindo os seguintes componentes:

- SPC – Serviço de Processamento de Consulta;
- SLP – Serviço Local de Pesquisa;
- *XML-DB* – Banco de Dados *XML*;
- SPP – Serviço de Propagação de Pesquisa;
- CES – Catálogo de Endereços de Super-Nós;
- BO – Base de Ontologia.

O SAI é composto por serviços responsáveis por gerenciar as informações na rede, recebendo dos SIs as consultas, armazenando dados na base *XML* ou ainda propagando as consultas para outros SAIs.

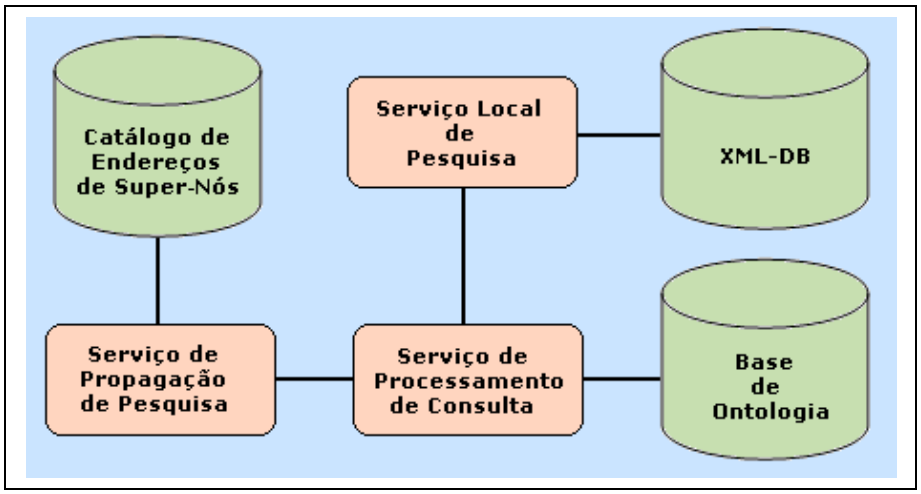

Figura 6.2 – Serviço de Administração de Informações (SAI).

Ao ser executado, o SAI inicializa a rede *JXTA* (GRADECKI, 2002) e, em seguida, o **Serviço de Processamento de Consulta (SPC)**, que é responsável por receber e processar as solicitações de consulta enviadas pelos vários SIs. O SPC armazena as solicitações de consulta de conteúdo em uma fila de estruturas de dados, representadas no Quadro 6.1, que contém a identificação do SI que a originou, o nome ou características para pesquisa do conteúdo e os endereços dos *peers* que satisfazem a consulta, que são preenchidos pelo SPC após a pesquisa.

Essa estrutura contém a variável idCliente, do tipo *String*, que armazena o *peer* solicitante. O nome do conteúdo pesquisado, quando informado, é armazenado na variável nomeObjetoPesquisa, do tipo *String*. As características do conteúdo são

armazenadas no *array* atributo, e os endereços dos *peers* que satisfazem a consulta são armazenados no *array* objetoRetorno.

Quadro 6.1 – Estrutura de dados para armazenamento da consulta solicitada no SI

```
idCliente : String;
nomeObjetoPesquisa : String;
atributo : String [];
objetoRetorno : String [];
```

O SPC lê uma consulta da fila de consultas e efetua a pesquisa na **Base de Ontologia (BO)** os possíveis conteúdos que atendam as solicitações do SI. O BO é o dicionário de dados da arquitetura, contendo os conteúdos e suas respectivas características. Trata-se de um diretório no servidor do SAI que contém dois tipos de arquivos *XML*. Esses arquivos foram criados para atender ao mecanismo de pesquisa, que permite ao usuário consultar pelo nome de um conteúdo ou por uma ou mais de suas características. Assim, o primeiro arquivo tem uma visão do conteúdo, ou seja, permite encontrar um conteúdo pelo seu nome e ter acesso direto às suas características, conforme pode ser observado no Quadro 6.2. O SPC pesquisa esse arquivo quando recebe de um SI uma solicitação para consultar pelo nome de um conteúdo. O segundo arquivo, apresentado no Quadro 6.3, tem uma visão da característica, ou seja, ele permite encontrar uma característica pelo seu nome e com isso saber quais objetos a possuem. Seu conteúdo é pesquisado pelo SPC quando o mesmo recebe do SI uma solicitação para consultar por uma ou mais características. Dessa forma, temos duas visões para a pesquisa, onde é possível pesquisar por um objeto e listar suas características ou pesquisar por uma característica e listar os objetos que a possuem. A estrutura do conteúdo é definida pelo Serviço de Publicação de Ontologia, descrito na próxima seção. Por fim, os conteúdos encontrados na pesquisa são enviados ao SI solicitante.

Quadro 6.2 – Arquivo *XML* com a visão dos conteúdos e suas características.

```
<Objeto>
  <nome>Book</nome>
  <arquivo>book.xml</arquivo>
  <atributo>
    <string>name</string>
    <string>author</string>
    <string>publisher</string>
    <string>year</string>
  </atributo>
  <endereco>
 <string>urn:jxta:uuid-59616261646167A7874610325033B40
          C0D73F7A74048A90 AD5C07966916603</string>
  <string>urn:jxta:uuid-52FD469D90D476ACDBA0EA
          0AD548D22AB29C14936BFC7E26D4A319</string>
  </endereco>
</Objeto>
```

O Quadro 6.2 mostra a estrutura de um conteúdo gravado no formato *XML*. Nesse exemplo temos um conteúdo de nome *Book*, identificado pela *tag* <nome>, que é gravado em um arquivo "book.xml", na tag <arquivo>. As características que identificam esse conteúdo estão na *tag* <atributo>, que contém *name, author, publisher* e *year*. Esse documento encontra-se armazenado em dois *peers* endereçados em urn:jxta:uuid-59616261646167A7874610325033B40C0D73F7A74048A90AD5C07966 916603 e urn:jxta:uuid-52FD469D90D476ACDBA0EA0AD548D22AB29C14936BFC 7E26D4A319, identificados pela *tag* <endereco>.

Quadro 6.3 – Arquivo *XML* com a visão das características e seus conteúdos.

```
<Atributo>
  <nome>name</nome>
  <objeto>
    <string>book</string>
  </objeto>
</Atributo>
<Atributo>
  <nome>nome</nome>
  <objeto>
    <string>pessoa</string>
    <string>cidade</string>
  </objeto>
</Atributo>
```

O Quadro 6.3 mostra o arquivo *XML* que armazena as características e em que conteúdos estas existem. Este quadro apresenta uma característica chamada *name*, na *tag* <nome>, que se encontra no objeto *book*, na *tag* <objeto>. Também é apresentada a característica chamada "nome" que se encontra nos objetos "pessoa", "cidade" e "festa".

O SPC envia a consulta ao **Serviço de Propagação de Pesquisa (SPP)**, que recebe a consulta e a armazena em uma fila própria, semelhante à fila do SPC, porém com uma informação adicional: o *timestamp* de chegada da consulta na fila. Em seguida, o SPP propaga a consulta para outros SAIs. Cada consulta da lista do SPP, aguarda um certo tempo pelo retorno da pesquisa, que conta a partir do *timestamp*, ao término do qual a consulta é eliminada da fila para que não fique indefinidamente aguardando um retorno. Os resultados obtidos no serviço SPP são enviados ao SPC, que por sua vez retorna ao SI o resultado da pesquisa.

Nesse primeiro momento, a pesquisa foi feita apenas no SAI ao qual o SI está ligado e nos SAIs por este conhecido, isto é, que estão armazenados no Catálogo de Endereços de Super-Nós (CES). Assim, nenhum acesso a qualquer SI foi feito. Isso ocorre pois o objetivo é identificar possíveis conteúdos do domínio de interesse da consulta do usuário. Como o SI pode ser móvel e se desconectar durante uma pesquisa, ou simplesmente não ficar aguardando uma resposta, após a tentativa e falha do SAI de se comunicar com o SI, o resultado deve ser armazenado em uma lista, com estrutura de dados igual a utilizada pelo SPP, aguardando a próxima conexão do SI para então repassar os resultados encontrados. Desta forma, a cada conexão do SI, este deve verificar se existem solicitações pendentes. Essas solicitações permanecem armazenadas no SAI por um certo período, após o qual são excluídas.

O SAI também possui um **Serviço Local de Pesquisa (SLP)** que é responsável por fazer as consultas em conteúdos armazenados localmente. Para isso, o SAI possui uma base de dados chamada *XML-DB*. A *XML-DB* trata-se de um conjunto de arquivos *XML* armazenados em um diretório próprio criado na arquitetura. Esses arquivos possuem os conteúdos recebidos para armazenamento no SAI e gerados pelos SIs através do Serviço de Publicação de Ontologia. Cada arquivo da base contém informações sobre um conteúdo, sendo possível ter no mesmo arquivo diversos

registros. Como forma de redução do tamanho dos arquivos, os mesmos armazenam apenas as *tags* e os dados, sendo a validação da estrutura e dos valores para os dados de responsabilidade do serviço gerador da ontologia. O Quadro 6.4 mostra um arquivo da base de dados *XML-DB*.

Quadro 6.4 – Arquivo de dados *XML-DB* com conteúdos sobre livros.

```
<Objeto>
  <Book>
    <name>Big Java</name>
    <author>FURMANKIEWICZ, Edson</author>
    <publisher>Bookman</publisher>
    <year>2004</year>
  </Book>

  <Book>
    <name>Client/server programming with Java and Corba</name>
    <author>HARKEY, Dan</author>
    <publisher>John Wiley E Sons</publisher>
    <year>1998</year>
  </Book>

  <Book>
    <name>Como programar em JavaBeans</name>
    <author>COFFEE, Peter</author>
    <publisher>Makron Books</publisher>
    <year>1999</year>
  </Book>
</Objeto>
```

Enquanto o SPC é responsável por fazer pesquisa nas estruturas dos conteúdos, o SLP faz a pesquisa por dados nos conteúdos. O SLP é iniciado pelo SAI e recebe dos SIs as solicitações de consulta por dados de um conteúdo. Essas solicitações são armazenadas em uma fila, cuja estrutura de dados da solicitação é mostrada no Quadro 6.5. O serviço lê uma solicitação da fila e busca na base *XML-DB* o arquivo correspondente ao nome do conteúdo. Em seguida, é efetuada a leitura do arquivo XML, e para cada *tag* de característica encontrada é feita uma comparação com a característica do conteúdo que se deseja pesquisar. Se as características forem iguais, então, o valor da *tag* da característica do arquivo é comparado ao valor recebido. Ao

término da leitura de um registro do arquivo de conteúdo, se este satisfizer a consulta, então seu nome é armazenado para ser retornado como resposta da consulta.

Quadro 6.5 – Estrutura de dados do SLP que armazena solicitação de consulta.

```
idCliente : String;
nomeObjetoPesquisa : String;
atributo : String[];
valor : String[];
objetoRetorno : String[];
```

Essa estrutura contém a variável idCliente do tipo *String* que armazena o *peer* solicitante. O nome do conteúdo pesquisado é armazenado na variável nomeObjetoPesquisa, do tipo *String*. As características do conteúdo que se deseja consultar são armazenadas em um *array* de S*tring*. Os valores a serem pesquisados estão no *array* de *String* valor, e os registros que satisfazem a consulta são armazenados no *array* de *String* objetoRetorno.

6.2 Serviço de Informações (SI)

O **Serviço de Informações (SI)**, apresentado na Figura 6.3, é um *peer* na rede formado pelos seguintes componentes:

- IC – Interface de Consulta;
- SLP – Serviço Local de Pesquisa;
- SPO – Serviço de Publicação de Ontologia;
- Documento *XML*.

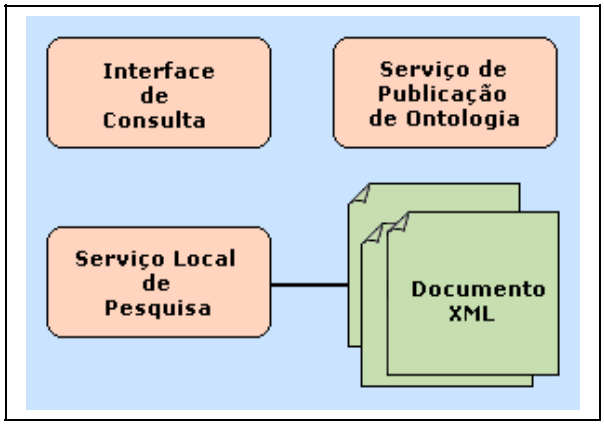

Figura 6.3 – Serviço de Informações (SI).

A **Interface de Consulta (IC)** do SI é a aplicação responsável pela criação da consulta que o usuário deseja fazer. Essa aplicação conduz o usuário de forma amigável através das etapas que devem ser percorridas até chegar ao resultado da consulta. O usuário inicialmente precisa definir na IC o nome do conteúdo, ou especificar quais características identificam o conteúdo que ele deseja consultar. Uma vez definidas tais restrições, a consulta é submetida ao SPC no SAI, que a processa e retorna uma lista, representada no Quadro 6.6, com os conteúdos que satisfazem a consulta efetuada pelo usuário. A lista retornada contém as características completas do conteúdo e seus endereços na rede, que podem reportar a qualquer outro *peer* (SI ou SAI) conectado à rede. O usuário pode, então, selecionar um conteúdo para conhecer detalhes de sua estrutura. Até este ponto o usuário não introduz nenhum dado, mas apenas identifica características e conteúdos que deseja consultar. Também não é estabelecida nenhuma conexão com outros SIs, mas apenas entre um SI e o SAI ao qual se encontra conectado. Uma vez que o conteúdo foi definido, é possível informar dados a serem consultados sobre as características do mesmo. Dessa vez, a consulta é enviada diretamente a um ou mais *peers* que tenham o conteúdo armazenado, conforme foi identificada na primeira consulta feita ao SAI. Como resultado da consulta, é retornado ao SI um objeto com o conteúdo consultado, conforme Quadro 6.6. O usuário então pode visualizar as informações ou mesmo criar uma cópia local das mesmas.

Quadro 6.6 – Arquivo da consulta retornada para o SI.

```
<Objeto>
  <Book>
    <name>Big Java</name>
    <author>FURMANKIEWICZ, Edson</author>
    <publisher>Bookman</publisher>
    <year>2004</year>
  </Book>
  <endereco>
 <string> urn:jxta:uuid-59616261646167A7874610325033B40
          C0D73F7A74048A90 AD5C07966916603</string>
  </endereco>
</Objeto>
```

O Quadro 6.6 demonstra uma consulta retornada para o SI. O arquivo *XML* representa um conteúdo com suas características e dados, além do endereço do *peer* onde o mesmo está armazenado.

O SI pode armazenar documentos *XML,* caso tenha capacidade para tanto, ou então enviá-los para serem armazenados diretamente no SAI. Caso opte por armazenar localmente, o **Serviço Local de Pesquisa** (SLP) será responsável por consultar nos documentos *XML* locais os conteúdos solicitados por outros SIs. O SLP recebe uma consulta via rede, em formato *XML,* e pesquisa entre seus documentos locais quais satisfazem a consulta solicitada. Os resultados encontrados são enviados ao SI solicitante. O mecanismo de consulta do SLP no SI é o mesmo já descrito no SLP para o SAI.

De modo semelhante ao SAI, que possui uma base de dados XML, o SI também possui um conjunto de arquivos *XML* armazenados em um diretório próprio. Esses arquivos são os dados dos conteúdos armazenados no SI e que foram gerados pelo Serviço de Publicação de Ontologia.

Uma versão mais simples do SI, contendo apenas a Interface de Consulta, pode ser executada em dispositivos que desejem apenas efetuar consultas na rede, sem compartilharem dados.

6.3 Serviço de Publicação de Ontologia (SPO)

Diversas soluções foram propostas nos últimos anos na busca de um mecanismo para formalizar e definir conteúdos. O uso de ontologias tem se apresentado como um mecanismo formal capaz de viabilizar o processamento semântico da informação através de um agente computacional. O uso de uma ontologia permite que o entendimento compartilhado de termos possam ser utilizados por homens e programas para ajudar no intercâmbio de informações. O SI disponibiliza um mecanismo denominado **Serviço de Publicação de Ontologia (SPO)**, seguindo um modelo proposto por (NOY & MCGUINNESS, 2001), que permite a criação de estruturas para representar os conteúdos de informações que serão disponibilizados para consulta posteriormente. Esse serviço permite a formalização, o compartilhamento e a definição de conceitos, restrições, relacionamentos e axiomas de um domínio de conhecimento. No seu desenvolvimento, utilizou-se uma linguagem apropriada para a definição semântica de dados, a *OWL* (SMITH et al, 2004) por ser padronizada pelo *W3C* e por oferecer recursos compreensíveis às máquinas, atendendo assim às necessidades deste trabalho.

A arquitetura da Ontologia do SPO permite ao usuário descrever os conteúdos de seu domínio como um cadastro comum de dados, sem se preocupar com regras da linguagem OWL. Os dados necessários para criação da ontologia são:

- **Classe**: nome que identifica um conjunto comum de características de um conteúdo, equivalente ao conceito de objeto na orientação a objetos. É um domínio de interesse comum;
- **Descrição**: texto livre com o objetivo de identificar a classe;
- **Propriedades**: características da classe. Além de um nome, também possui:
    - **Tipo**: pode ser *ObjectProperty*, que indica que a propriedade tem o papel de relacionar uma classe a outra classe. Além disso, pode ser um *DataType*, que define qual o tipo de dado, podendo ser caractere, inteiro, ponto flutuante, lógico, data, entre outros;

- **Cardinalidade**: identificação sobre a quantidade de valores aceita pela propriedade, podendo ser um único valor (simples) ou mais que um valor (múltiplo);
- **Comentário**: observação sobre a propriedade permitindo ao usuário identificar e justificar a sua utilização;
- **Dicionário de Termos**: permite associar palavras que podem ser usadas como sinônimos a fim de facilitar as consultas sobre o objeto;
- **Regras (axiomas)**: a partir delas torna-se possível estabelecer as restrições para um domínio.

Uma vez definida toda a estrutura da ontologia que descreve um conteúdo, ela é salva como um arquivo *XML* no formato da linguagem *OWL*. O Quadro 6.7 mostra um arquivo gravado pelo SPO.

Quadro 6.7 – Arquivo de descrição de dados compartilhados no *XOP*.

```
<owl:class rdf:ID="Vinho">
<owl:DatatypeProperty rdf:ID="ano">
  <rdfs:range  rdf:resource="&xsd:integer%"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="nome">
  <rdfs:range  rdf:resource="&xsd:string%"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="vinícola">
  <rdfs:range  rdf:resource="&xsd:tipo indefinido%"/>
</owl:DatatypeProperty>
```

O Quadro 6.7 apresenta uma classe denominada “Vinho” com as propriedades “ano” do tipo inteiro, “nome” do tipo caractere e “vinícola” que é um relacionamento com outra classe.

A interface do SPO permite que o usuário visualize em uma árvore todas as classes já criadas no SAI ao qual o usuário está conectado. Uma vez criada a estrutura esta é armazenada na **Base de Ontologias (BO)**. O BO é um diretório na arquitetura do

SAI responsável por gravar os documentos *OWL*. É importante ressaltar que essa base não contém os dados, seu objetivo é servir de repositório para que os SIs possam acessar apenas as estruturas que identificam conteúdos e determinar a localização dos dados, ou seja, trata-se de um dicionário de dados. Cada SAI é responsável por manter a ontologia que descreve a semântica dos dados compartilhados pelos SIs que estão conectados diretamente a este. Uma vez armazenada, essa ontologia fica disponível para que outros usuários tenham acesso para compartilhar dados descritos por essa mesma ontologia ou para estender sua estrutura.

O SPO também funciona como uma interface para manter as instâncias de conteúdos. Uma instância é uma ocorrência concreta sobre um domínio de conteúdo que pode ser armazenada. O usuário seleciona um dos conteúdos previamente cadastrados e digita os dados para o mesmo. O SPO efetua a validação dos dados informados, baseado na ontologia associada ao conteúdo. Essas instâncias podem ser armazenadas no SI, em um documento no formato *XML*, ou no banco de dados *XML-DB* do SAI, caso o SI não tenha capacidade de armazenamento e funcione apenas como uma fonte de entrada da informação.

Após a definição das classes, subclasses, propriedades e axiomas, criam-se as instâncias para a base da ontologia. As instâncias representam indivíduos específicos de uma determinada classe.

6.4 Processamento da Consulta

No momento da consulta, uma vez que as informações estão distribuídas em uma rede *peer-to-peer*, os usuários não têm idéia de onde a informação está localizada, porém possuem alguns dados que podem ajudar a encontrá-la. Assim, o processo de pesquisa é orientado pelo sistema a percorrer duas fases, conforme Figura 6.4.

Na primeira fase, o usuário identifica o conteúdo que deseja consultar, sem se preocupar com os dados. Nesse momento, o usuário deve tentar localizar conteúdos na rede que tenham características para satisfazer a consulta, digitando na IC (passo 1 na Figura 6.4) parâmetros de consulta que identifiquem parte de um conteúdo. A consulta é enviada ao SPC (passo 2), que faz uma pesquisa no dicionário de dados da ontologia (passo 3). O SAI propaga a pesquisa a outros SAIs através do SPP (passo 4). Para cada

pesquisa, o SPP guarda por um certo tempo os resultados encontrados, ao término do qual as respostas recebidas são ignoradas. Isto é necessário para evitar que o serviço fique aguardando uma resposta por tempo indefinido. Os conteúdos encontrados nos SAIs são retornados ao SI solicitante (passo 5), contendo além da estrutura completa do conteúdo pesquisado, uma lista de endereços de *peers* – SI ou SAI – nos quais o conteúdo está armazenado. Nessa primeira fase, a pesquisa é processada apenas nos SAIs, sem acessar os SIs, isso porque o objetivo dessa fase é identificar estruturas que sejam semanticamente parecidas com as necessidades da pesquisa do usuário.

Na segunda fase da pesquisa, o usuário informa filtros sobre as características do conteúdo na IC (passo 1) e submete a consulta a um ou mais *peers* identificados na primeira fase da consulta. Caso algum SAI tenha respondido como tendo o conteúdo, o SLP (passo 7) receberá a consulta e fará uma pesquisa na base *XML-DB* (passo 8). Cada SI também recebe a consulta no SLP (passo 7') e acessa seus documentos *XML* (passo 8') para encontrar os documentos que satisfazem a solicitação. Os SIs que satisfizerem a consulta, respondem positivamente para o SI solicitante (passo 9), que por sua vez apresenta os resultados da pesquisa como uma lista ao usuário, permitindo que esse visualize, ou salve localmente o conteúdo.

Devido à natureza dinâmica das redes *P2P*, os *peers* podem desconectar-se durante a pesquisa. Nesse caso, as consultas feitas nos SAIs são armazenadas, aguardando assim a próxima conexão do *peer*. Assim, a cada nova conexão do *peer* na rede, o serviço SPC deve verificar se existem consultas pendentes aos mesmos. Estes resultados armazenados são excluídos de tempos em tempos pelo SAI.

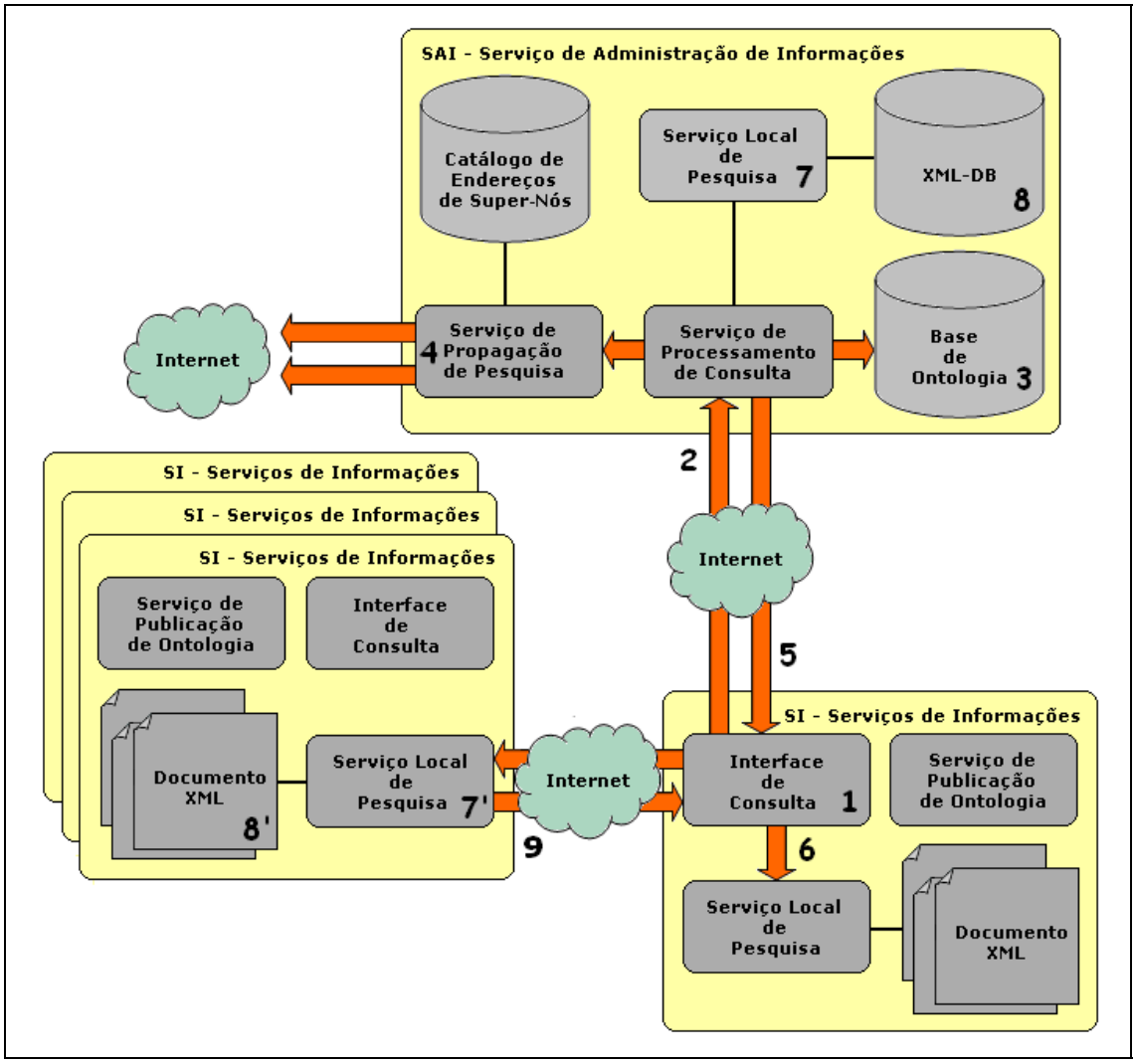

Figura 6.4 – Processamento da consulta.

O diagrama de seqüência apresentado na Figura 6.5 representa a interação entre os serviços envolvidos em uma consulta feita na arquitetura *XOP*. A consulta inicia-se quando o usuário abre no SI, o serviço IC (passo 1 na Figura 6.5) e entra com uma característica ou conteúdo que será pesquisado no SAI, por meio do serviço SPC (passo 2). O SPC acessa a BO para verificar se existem conteúdos que atendam a solicitação do usuário (passo 3). O SPC retorna uma lista de conteúdos ao SI (passo 4) e o usuário seleciona uma ou mais características, informa os valores desejados, e submete a nova pesquisa ao SLP dos *peers* que possuem o conteúdo consultado (passo 5). Cada *peer* acessa a sua base de dados (passo 6) e retorna o resultado da consulta ao solicitante (passo 7).

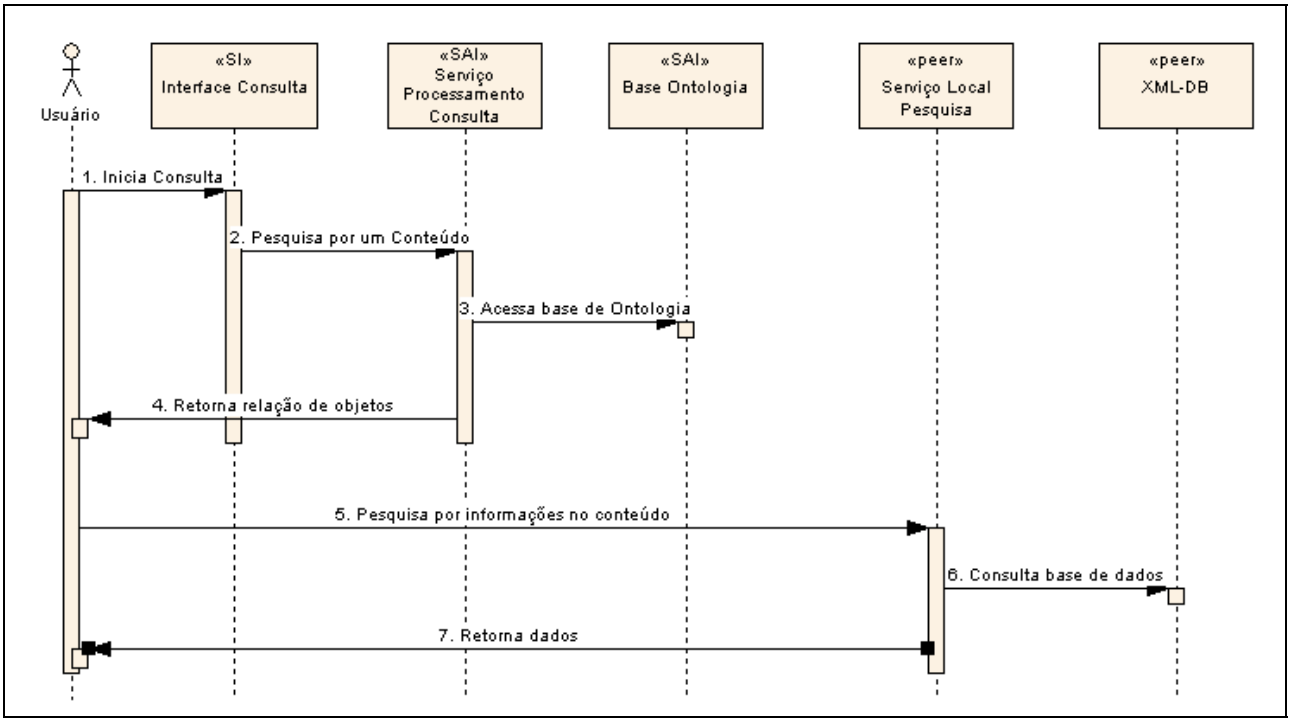

Figura 6.5 – Diagrama de seqüência de uma consulta de conteúdo.

6.5 Considerações Finais

O presente capítulo descreveu a arquitetura XOP, proposta nessa dissertação, que visa permitir o compartilhamento e a busca de dados em um ambiente distribuído, composto por dispositivos heterogêneos e móveis.

Apesar de existirem propostas semelhantes na literatura, as mesmas não atendem a todos os requisitos que foram definidos nessa dissertação, conforme discutido no capítulo 5. A tabela 6.1 mostra as limitações das demais arquiteturas e permite a comparação com as características da arquitetura XOP. Nota-se na tabela que a arquitetura XOP é a única a lidar com a mobilidade e a desconexão de peers e a possuir mecanismos de descrição e busca de dados com base em sua semântica. Além disso, a arquitetura proposta pode ser utilizada nos mais diversos campos de aplicação, já que as descrições semânticas dos dados podem ser facilmente efetuadas pelos usuários, que precisam apenas criar uma nova ontologia ou estender uma ontologia já existente. Esses resultados são obtidos utilizando uma infra-estrutura de rede P2P amplamente difundida – a JXTA – e adotando uma arquitetura descentralizada estruturada.

Tabela 6.1 – Comparação entre os trabalhos relacionados e a arquitetura XOP.

Características	DBGlobe	Bricks	KEEx	Piazza	XOP
Topologia da Rede *P2P*	Descentralizada Estruturada	Descentralizada estruturada	Descentralizada estruturada	Descentralizada estruturada	Descentralizada estruturada
Infra-Estrutura de Comunicação	*Web Services*	*Web Services*	*JXTA*	*Web Services*	*JXTA*
Semântica dos Dados	Sim	Não	Sim	Sim	Sim
Dispositivos Móveis	Sim	Não	Não	Não	Sim
Desconexão de *Peers*	Não	Não	Não	Não	Sim
Campo de Aplicação	Genérico	Restrito	Genérico	Genérico	Genérico

O próximo capítulo apresenta um protótipo de implementação da arquitetura XOP e analisa o seu desempenho, com base em resultados de testes efetuados com o protótipo desenvolvido.

7 IMPLEMENTAÇÃO E RESULTADOS

Os serviços do projeto *XOP* foram desenvolvidos com a linguagem Java versão 1.5 e a rede *P2P* foi construída utilizando-se dos protocolos de rede *JXTA* versão 2.3 (GRADECKI, 2002). Para fazer a leitura dos arquivos *XML* é utilizada a biblioteca *SAX* versão 2.0 (SAX, 2007).

Como um dos objetivos do trabalho é criar uma estrutura independente e que possa ser disponibilizada para diversos dispositivos, optou-se por não utilizar um banco de dados *XML* existente e sim criar uma estrutura própria de manipulação e armazenamento dos conteúdos em documentos *XML*.

7.1 Interfaceamento com os Serviços da Arquitetura

A Figura 7.1 apresenta um exemplo de consulta através da Interface de Consulta do SI. Na aba inicial, de nome “Pesquisa”, uma característica “autor” é informada para ser consultada. Após pressionar o botão “Pesquisar”, o SI comunica-se com o SAI, que responde com três domínios de conteúdo com a característica pesquisada: Jornal, Artigo e Livro. O usuário pode selecionar um dos conteúdos na lista de nome “Objetos” – no exemplo foi selecionado o conteúdo “Livro”.

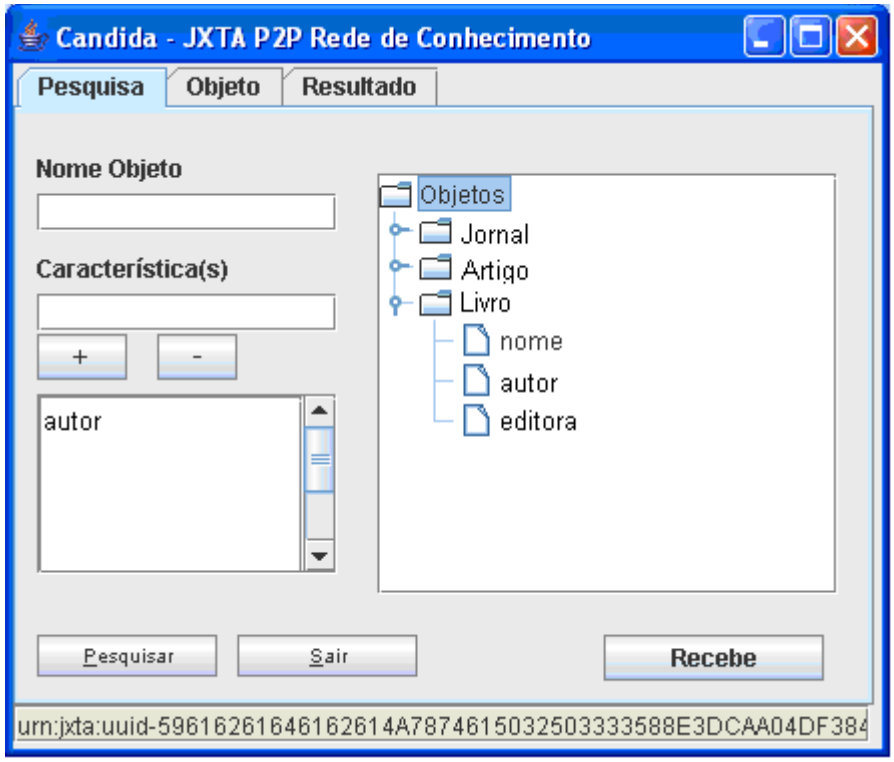

Figura 7.1 – Parametrização de uma consulta na Interface de Consulta.

Na aba de nome "Objeto", ilustrada na Figura 7.2, são apresentadas as características do conteúdo Livro: nome, autor e editora. O usuário informa um valor para pesquisar em uma das características – no exemplo foi informado autor igual a "Antoine de Saint-Exupéry" – e submete a pesquisa a todos os *peers* onde o conteúdo "Livro" esteja armazenado. Os *peers* nos quais a pesquisa informada for satisfeita irão responder retornando o documento *XML* correspondente ao conteúdo pesquisado. Três *peers* identificados como "Biblioteca do Congresso", "Biblioteca Nacional" e "Biblioteca On-Line" responderam positivamente à pesquisa. Por fim, o usuário pode acessar na aba "Resultado" um dos conteúdos retornados para visualizar os dados, conforme Figura 7.3.

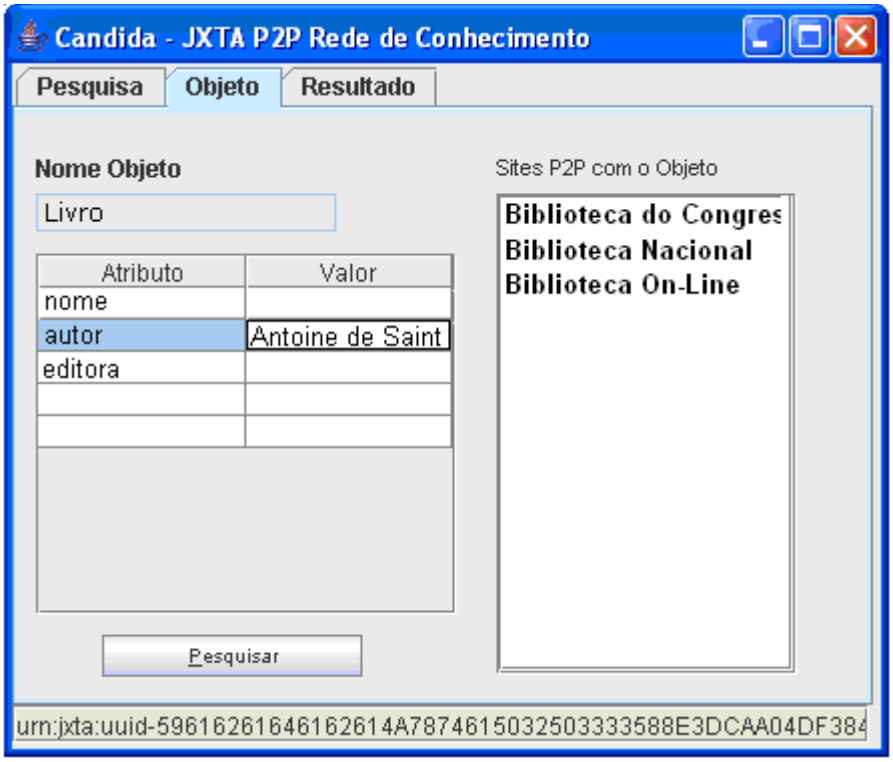

Figura 7.2 – SIs que satisfazem a consulta.

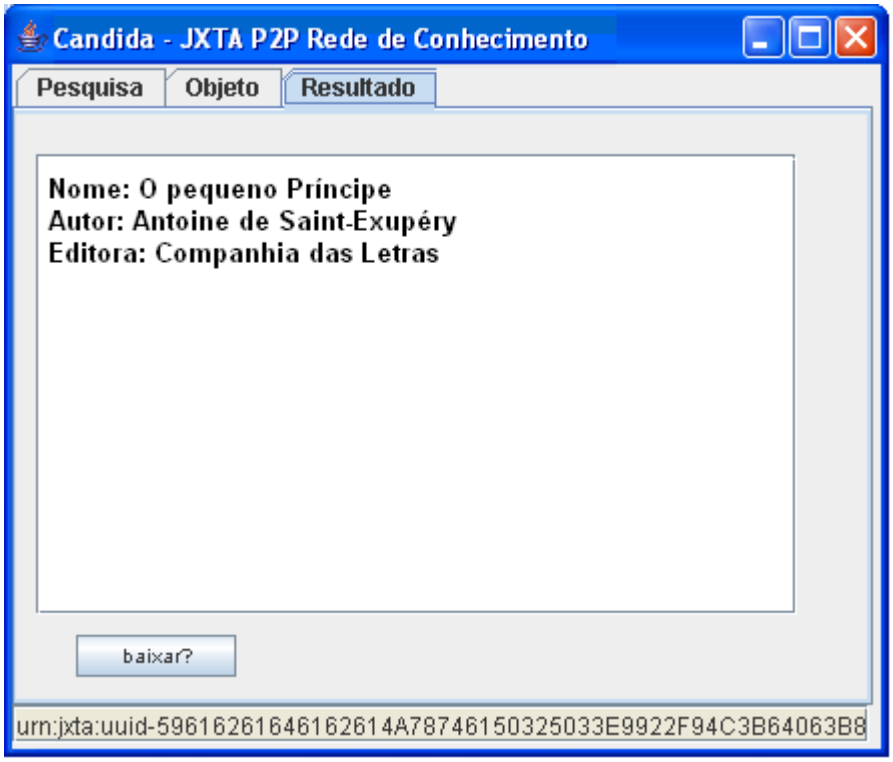

Figura 7.3 – Dados de um conteúdo retornado por um SI.

Para facilitar a criação das estruturas de conteúdos, a arquitetura *XOP* oferece uma interface onde o usuário pode criar a ontologia sem se preocupar em conhecê-la em maiores detalhes. Para isso, basta efetuar o cadastramento dos conteúdos na tela apresentada na Figura 7.4. Na lista da esquerda são apresentadas as classes já criadas e que se encontram armazenadas no SAI. Para editá-las basta dar um duplo clique em qualquer um dos nomes de conteúdos apresentados. À direita da tela estão os campos que devem ser preenchidos pelo usuário para o cadastramento do conteúdo. Cada campo representa uma *tag* da linguagem *OWL*. O usuário identifica o conteúdo com um nome de classe e uma breve descrição com o propósito do conteúdo. As características são informadas no campo propriedade e classificadas com um tipo, indicando se a propriedade é um relacionamento ou um dado simples. Caso seja um tipo de dado simples, deve-se definir adicionalmente o valor como numérico, alfanumérico, data, etc. Após pressionar o botão “>>”, cada propriedade informada é armazenada na lista à direita. Os botões “<<” e “-” permitem editar e excluir uma propriedade, respectivamente.

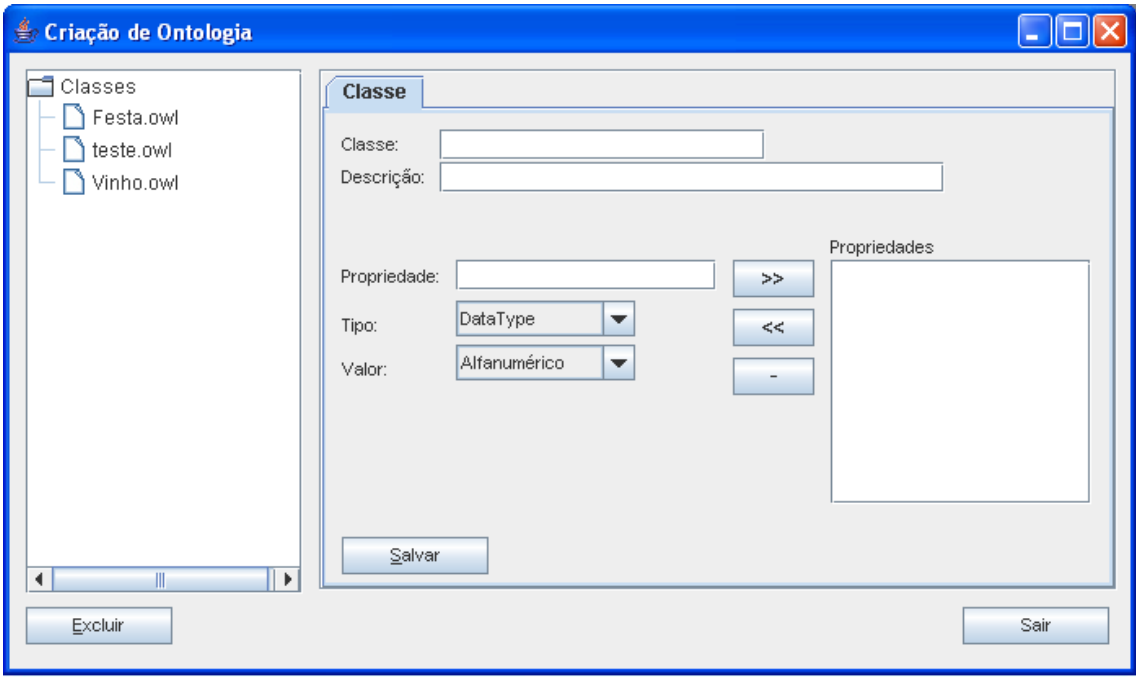

Figura 7.4 – Interface do Serviço de Publicação de Ontologia (SPO).

7.2 Avaliação de Desempenho

A Figura 7.5 apresenta os dois cenários de teste criados para execução da medição de desempenho da arquitetura XOP. O cenário A possui um servidor SAI e dois computadores representando os SIs, e o cenário B possui dois servidores SAI, um deles conectado a dois SIs e o outro com três *peers* SIs.

Os testes foram efetuados em computadores com processador Intel Celeron 2,8 GHz, 512 MB de memória RAM, executando o sistema operacional Windows XP e conectados por uma rede local.

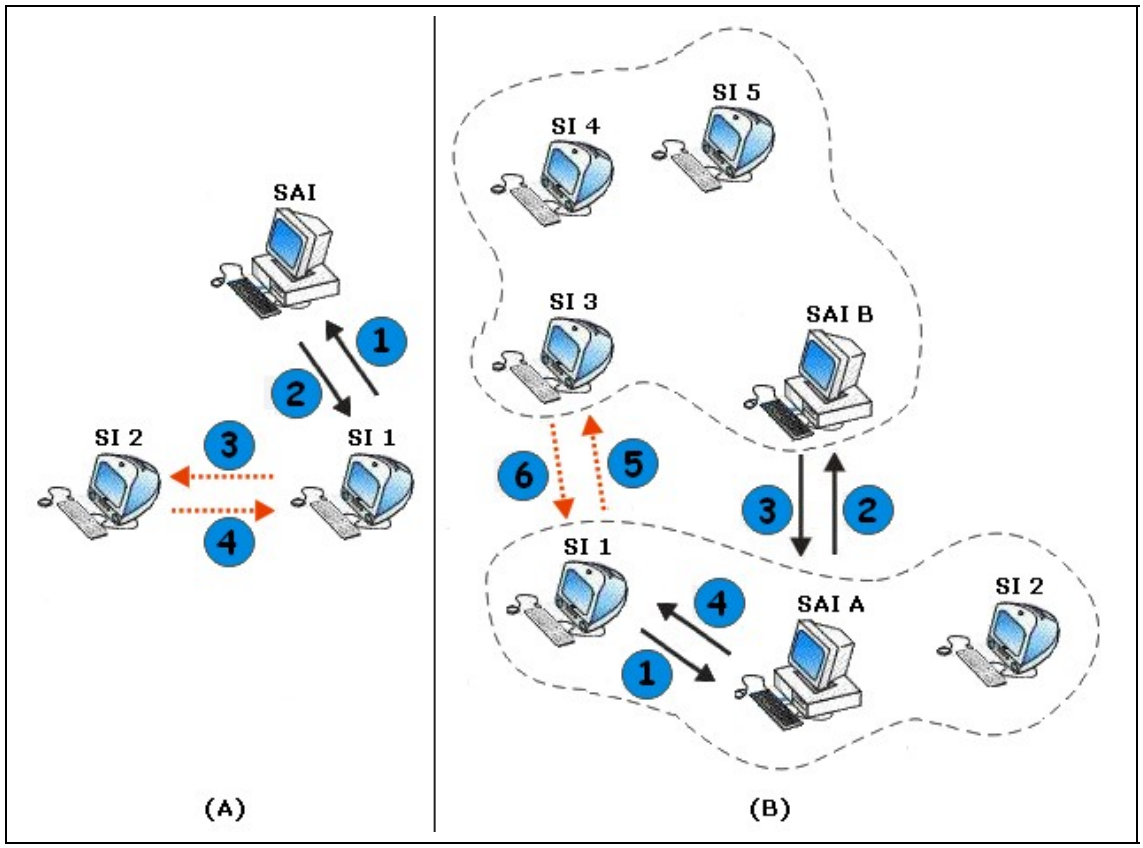

Figura 7.5 – Redes criadas para validar a arquitetura *XOP*.

No **cenário A** da Figura 7.5, a medição do tempo refere-se ao:

- tempo de resposta entre o envio da consulta pelo SI 1 (passo 1) para encontrar os domínios de conteúdo e o retorno da resposta pelo SAI (passo 2);

- tempo de resposta entre a solicitação da informação sobre um domínio feita pelo SI 1 ao SI 2 (passo 3) e o retorno do objeto da consulta pelo SI 2 (passo 4).

Dessa forma, o tempo total de resposta para uma busca pela informação é a soma dos passos 1 a 4.

No **cenário B** da Figura 7.5, a medição do tempo refere-se ao:

- tempo de resposta entre o envio da consulta pelo SI 1 (passo 1) para encontrar os domínios de conteúdo;
- tempo de resposta entre o Serviço de Propagação do SAI A (passo 2) e o retorno da resposta (passo 3) pelo SAI B;
- tempo de retorno da resposta do SAI A ao SI 1 (passo 4);
- tempo de resposta entre a solicitação da informação sobre um domínio feita pelo SI 1 ao SI 3 (passo 5) e o retorno do objeto da consulta pelo SI 3 (passo 6).

Dessa forma, o tempo total de resposta para uma busca pela informação é a soma dos passos 1 a 6.

Durante a fase de testes, o tamanho da base de dados de cada super-nó foi incrementada para avaliar a escalabilidade da arquitetura. A Figura 7.6 mostra o tempo de resposta avaliado para as duas fases da execução da consulta: busca na base de ontologia do SAI e, então, obtenção do objeto pesquisado no SI onde os dados são iguais ao pesquisado. O gráfico mostra que a arquitetura é escalável mesmo com o crescimento do tamanho do banco de dados do SAI. É importante frisar que os objetos armazenados no SAI descrevem diferentes conteúdos, assim é improvável encontrar muitos objetos na base de dados de um único SAI.

O tempo de resposta na primeira fase da pesquisa foi praticamente igual nos dois cenários de teste, apresentando uma variação mínima que pode ser desconsiderada. Apesar de, no cenário B, a pesquisa enviada pelo SI ter sido replicada do SAI A para o SAI B, a execução da pesquisa ocorreu em paralelo nas duas máquinas. Como os *peers* encontravam-se na mesma rede local, o tempo de comunicação na rede não afetou o desempenho das consultas, dado que o SAI A, ao terminar de efetuar a pesquisa em sua base de dados, já tinha a resposta do SAI B à sua disposição. O tempo de resposta da segunda fase da pesquisa, que compreende a comunicação entre os SIs, tanto no cenário

A quanto no cenário B, não apresenta diferenças, pois na segunda fase da pesquisa o comportamento da arquitetura é o mesmo nas duas configurações, ou seja, a comunicação é realizada entre o SI solicitante e o SI que contém o objeto da pesquisa. O gráfico da Figura 7.6 apresenta os tempos medidos no cenário B dos testes, por se tratar de uma configuração mais completa.

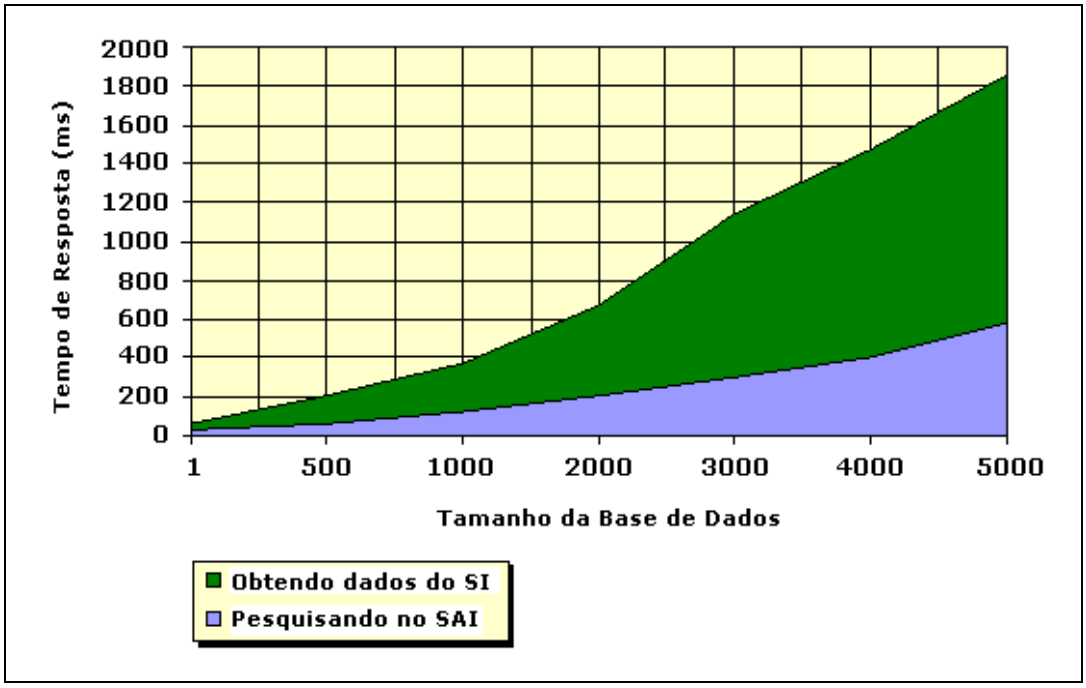

Figura 7.6 – Tempo total de resposta da pesquisa.

8 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho propôs uma arquitetura que permite descrever, compartilhar e descobrir conteúdos sobre diversos assuntos representados na forma de dados semi-estruturados. Para tanto, os dados são distribuídos sobre uma rede *P2P* e descritos através de uma ontologia, que permite representar e elaborar estruturas que identificam e padronizam os diversos tipos de conteúdos. Para permitir a descoberta de conteúdos, qualquer dispositivo com capacidade de conectividade pode servir como local de geração, armazenamento ou simplesmente de consulta de conteúdos. Um servidor com conexão permanente à rede pode servir como local de armazenamento e consulta, ao passo que um dispositivo de baixa conectividade e pouco poder de processamento, como um *PDA*, pode apenas acessar a rede para consulta.

A arquitetura *XOP* tem uma série de vantagens quando comparada com as outras propostas descritas neste trabalho. O projeto Bricks (RISSE et al, 2005) e a arquitetura *XOP* têm características semelhantes, mas enquanto no projeto Bricks documentos são distribuídos entre os seus *peers*, na arquitetura *XOP* o documento permanece localizado em um único *peer*. Além disso, *XOP* usa ontologias para descrever os dados, proporcionando uma maior indexação e facilidades de pesquisa de dados do que os metadados baseados na estratégia utilizada no projeto Bricks.

DBGlobe (PITOURA et al, 2003), KEEx (BONIFÁCIO et al, 2004) e Piazza (HALEVY, 2006) possuem características similares à arquitetura *XOP* ao procurar descrever o conteúdo a ser compartilhado usando ontologias. Entretanto, essas arquiteturas não apresentam um mecanismo que permita a utilização de dispositivos móveis e, assim, não conseguem tratar as freqüentes desconexões durante o processo de consulta.

O protótipo implementado demonstrou a viabilidade da arquitetura apresentada e, portanto, atendeu os objetivos propostos no início deste trabalho. No entanto, torna-se importante ressaltar que, a criação da base de ontologias caracteriza-se como um processo ainda bastante complexo. Isto porque, apesar da linguagem de descrição *OWL* apresentar-se como uma ferramenta bastante completa, as ontologias que necessariamente devem compreender um vocabulário de termos e significados comuns são definidas pelas pessoas que, por sua vez, pertencem a culturas diferentes, possuem

níveis de conhecimento variados e detêm percepções divergentes sobre um mesmo conceito e /ou assunto.

8.1 Publicações

O artigo intitulado *XOP: Sharing XML Data Objects through Peer-to-Peer Networks* foi aceito para publicação no *IEEE 22nd International Conference on Advanced Information Networking and Applications* (AINA 2008), classificado pela CAPES como Qualis A Internacional, e que será realizado de 25 a 28 de março de 2008.

8.2 Trabalhos Futuros

Como sugestão para trabalhos futuros, destaca-se:

- utilização de um mecanismo para melhorar o desempenho das consultas. Isto porque durante os testes ficou evidente que o maior tempo gasto no retorno das informações foi decorrente dos acessos ao sistema de arquivos do *XML-DB*, pois ao ser gerado pela interface da *BO* não foi criado nenhum algoritmo de indexação. Assim, a sugestão seria usar o Apache Lucene (APACHE, 2007), recurso que oferece serviços de indexação e pesquisa de texto em documentos;
- realização de um estudo sobre o tempo de resposta das consultas em larga escala para avaliar o desempenho do sistema sob condições de sobrecarga, visto que nesse trabalho foi montada uma infra-estrutura de rede mínima para demonstrar principalmente a viabilidade da troca de informações entre diferentes computadores;
- implementação do SI usando o JXME – uma versão do JXTA para dispositivos móveis com suporte ao Java Micro Edition (JME) [JXME 2006]. Nesse caso, o serviço do SI rodando em dispositivos móveis seria limitado apenas a enviar consultas e receber respostas.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Maurício B. Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares. Universidade Federal de Minas Gerais. Ciência da Informação, 2002. Vol. 2, 5-13.

ANDERSON, Richard. Professional XML. Ciência Moderna, Rio de Janeiro, 2001.

ANDROUTSELLIS-THEOTOKIS, Stephanos; SPINELLIS, Diomidis. A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys, 2004. Vol. 36, 335–371.

APACHE. Lucene. 2007. Disponível em: http://lucene.apache.org. Último acesso em Novembro de 2007.

AREF, Mostafa M.; ZHOU, Zhengbo. The Ontology Web Language (OWL) for a Multi-Agent Understating System. International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS 2005. Boston, EUA. Abril 2005.

MELVILLE, L., WALKERDINE, J., SOMMERVILLE, I. Ensuring dependability of P2P applications at architectural level. Relatório Técnico. Universidade de Atenas. Atenas, Grécia, 2002.

BALAKRISHNAN, Hari; KAASHOEK, M. Frans; KARGER, David et al. Looking up Data in P2P Systems. Communications of the ACM, 2003. Vol. 46, No. 2, 43-48.

BENEVENUTO, Fabrício, JÚNIOR, José Ismael, ALMEIDA, Jussara. Avaliação de mecanismos avançados de recuperação de conteúdo em sistemas P2P. 23º Simpósio Brasileiro de Redes de Computadores, SBRC 2005. Fortaleza, Brasil. Maio 2005.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The semantic web. Scientic American Magazine, 2001. Disponível em:

http://www.sciam.com/article.cfm?id=00048144-10D2-1C70-84A9809EC588EF21&page=1. Último acesso em Maio de 2006.

BONIFÁCIO, Matteo; BOUQUET, Paolo; DANIELI, Alberto et al. KEEx: A Peer-to-Peer Solution for Distributed Knowledge Management. International Conference on Knowledge Management, I-KNOW'04. Graz, Áustria. 2004.

BOURRET, Ronald. XML and Databases. 2006. Disponível em: http://www.rpbourret.com/xml/XMLAndDatabases.htm. Último acesso em Março de 2006.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. et al. Extensible Markup Language (XML) 1.0 (Fourth Edition), 2006. Disponível em: http://www.w3.org/TR/REC-xml. Último acesso em Janeiro de 2007.

CHAMBERLIN, Dom. XQuery: An XML query language. IBM Systems Journal 2002. Vol. 41, 597-615.

CHOI, Namyoun; SONG, Il-Yeol; HAN, Hyoil. A Survey on Ontology Mapping. ACM Sigmod Record, 2006. Vol. 35, No. 3, 34-41.

DEMEURE, Isabelle. An overview of P2P computing. ANWIRE winter school on middleware. Nicosia, Chipre. Janeiro 2006.

DENNY, Michael. Ontology Building: A Survey of Editing Tools. 2002. Disponível em: http://www.xml.com/pub/a/2002/11/06/ontologies.html. Último acesso em Maio de 2006.

DOM. Document Object Model (DOM). 2006. Disponível em: http://www.w3.org/DOM. Último acesso em Abril de 2006.

EVANS, Kirk Allen. Understanding Xpath. 2002. Disponível em: http://www.informit.com/articles/article.aspx?p=26851. Último acesso em Junho de 2006.

FURTADO, Miguel. XML - Extensible Markup Language. UFRJ, Rio de Janeiro, 2006. Disponível em: http://www.gta.ufrj.br/grad/00_1/miguel. Último acesso em Março de 2006.

GRUBER, Thomas R. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, 1995. Vol. 43, 907-928.

HALEVY, Alon Y.; IVES, Zachary G.; MORK, Peter. et al. Piazza: Data Management Infrastructure for Semantic Web Applications. 20º International World Wide Web Conference, WWW2003. Budapeste, Hungria, Maio 2006.

IDRIS, Nazmul. Benefits of using XML. 2006. Disponível em: http://developerlife.com/xmlbenefits/default.htm. Último acesso em Abril de 2006.

JÚNIOR, João B. R. Arquitetura de Aplicações Peer-to-Peer. Relatório Técnico. Grupo de Trabalho em Computação Colaborativa da UFPE. 2003.

GRADECKI, Joseph D.; GRADECKI, Joe. Mastering JXTA: Building Java Peer-to-Peer Aplications. Wiley Publishing, Inc., Indianápolis, 2002, 1ª Edição.

JXME. JXTA for Java Micro Edition. 2007. Disponível em: https://jxta-jxme.dev.java.net. Último acesso em Março de 2007.

KOLONIARI, Georgia; PITOURA, Evaggelia. Peer to peer management of XML data: issues and research challenges. ACM Sigmod Record, 2005. Vol. 34, 06-17.

LOO, Alfred W. The future of peer-to-peer computing. Communications of the ACM, 2003. Vol. 46, No. 9, 57-61.

MELLO, Ronaldo S. Gerência de dados XML em Bancos de Dados. Escola Regional de Banco de Dados, Passo Fundo, RS, 6-8 abril, 2006.

NAPSTER. Company Information – About Naspter. 2006. Disponível em: http://www.napster.com/about_napster.html. Último acesso em Março de 2006.

NOY, Natalya F.; MCGUINNESS, Deborah L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University, 2001. Disponível em: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html. Último acesso em Janeiro de 2007.

OASIS. XML: Proposed Applications and Industry Initiatives. 2006. Disponível em: http://www.oasis-open.org/cover/xml.html#applications. Último acesso em Junho de 2006.

OWL. OWL – Web Ontology Language – Overview. 2006. Disponível em: http://www.w3.org/TR/owl-features. Último acesso em Junho de 2006.

PITOURA, Evagellia; ABITEBOUL, Serge; PFOSER, Dieter. et al. DBGlobe: A Service-oriented P2P System for Global Computing. ACM Sigmod Record, 2005. Vol. 32, No 3, 77-82.

QUERY. XML Query (XQuery). 2006. Disponível em: http://www.w3.org/XML/Query. Último acesso em Dezembro de 2006.

RAY, Erik T. Aprendendo XML. Campus, Rio de Janeiro, 2001, 5ª edição.

RDF. Resource Description Framework (RDF). 2006. Disponível em: http://www.w3.org/RDF. Último acesso em Dezembro de 2006.

REFSNES, Jan Egil. Introduction to DTD. 2006. Disponível em: http://www.xmlfiles.com/dtd/dtd_intro.asp. Último acesso em Março de 2006.

RIGHI, Rafael da R.; PELLISSARI, Felipe R.; WESTPHALL, Carla M. P2P-Role: Uma Arquitetura de Controle de Acesso Baseada em Papéis para Sistemas Colaborativos Peer-to-Peer. SBRC 2004 – IV Workshop em Segurança de Sistemas Computacionais. Gramado, Brasil, 2006.

RISSE, Thomas; KNEZEVIC, Predrag. et al. The Bricks Infrastructure – An Overview. The International Conference. Moscou, Rússia, 2005.

RISSON, J.; MOORS, J. Survey of research towards robust peer-to-peer networks: search methods. Relatório Técnico. Universidade de New South Wales. Sydney, Austrália, 2004.

ROCHA, João; DOMINGUES, Marco; CALLADO, Arthur. et al. Peer-to-Peer: Computação Colaborativa na Internet. 22º Simpósio Brasileiro de Redes de Computadores, SBRC 2004. Gramado, Brasil. 2004.

ROCHE, Christophe. Ontology: a survey. 8th Symposium on Automated Systems Based on Human Skill and Knowledge. Gothenbug, Suecia. 2003.

SADOK, Djamel; LOUREIRO, Antonio A.; KAMIENSKI, Carlos A. et al. Documento de Especificação do Projeto Piloto: GT – Computação Colaborativa (P2P). Relatório Técnico. Grupo de Trabalho da Rede Nacional de Pesquisa, 2003.

SALMINEN, Airi; TOMPA, Frank Wm. Requirements for XML Document Database Systems. ACM Symposium on Document Engineering. Atlanta, EUA, 2001.

SAVOLA, Tom. Usando HTML – O Guia de Referência mais Completo. Campus, Rio de Janeiro, 1996.

SAX. About SAX. 2007. Disponível em: http://www.saxproject.org. Último acesso em Janeiro de 2007.

SCHEMA. XML Schema. 2006. Disponível em: http://www.w3.org/XML/Schema. Último acesso em Março de 2006.

SHENKER, Scott; LV, Qin; CAO, Pei. et al. Search and replication in unstructured peer-to-peer networks. International Conference on Supercomputing, ICS 2002. Nova Iorque, EUA. Junho 2002.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L. OWL – Web Ontology Language Guide. 2004. Disponível em: http://www.w3.org/TR/owl-guide. Último acesso em Janeiro de 2007.

SPENCER, Paul. Professional XML Design and Implementation. Wrox Press, 1999, 1ª Edição.

SUNG, L. G. Alex; AHMED, Nabeel; BLANCO, Rolando. et al. A survey of data management in peer-to-peer systems. Relatório Técnico. School of Computer Science, University of Waterloo, EUA, 2005.

TIS. The Information Society – An International Journal. 2006. Disponível em: http://www.indiana.edu/~tisj. Último acesso em Abril de 2006.

W3C. XML in 10 points. 1999. Disponível em: http://www.w3.org/XML/1999/XML-in-10-points.html. Último acesso em Março de 2006.

W3C. World Wide Web Consortium - Extensible Markup Language (XML) 1.0 (Third Edition), 2004. Disponível em: http://www.w3.org/TR/2004/REC-xml-20040204. Último acesso em Dezembro de 2006.

XML. Extensible Markup Language (XML). 2006. Disponível em: http://www.w3.org/XML. Último acesso em Dezembro de 2006.

XPATH. XML Path Language (XPath). 2006. Disponível em: http://www.w3.org/TR/xpath.html. Último acesso em Dezembro de 2006.

ZACHARY, G. Ives; LU, Ying. XML Query Languages in Practice. First International Conference on Web-Age Information Management. Londres, Inglaterra. Junho 2000.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo