ALEXANDRE TAKAHASHI ALBERT

UMA PROPOSTA PARA A DESCRIÇÃO E BUSCA POR RECURSOS UTILIZANDO METADADOS XML/RDF EM REDES PEER-TO-PEER

FLORIANÓPOLIS - SC
2002

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

Alexandre Takahashi Albert

UMA PROPOSTA PARA A DESCRIÇÃO E BUSCA POR RECURSOS UTILIZANDO METADADOS XML/RDF 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.

Vitório Bruno Mazzola

Florianópolis, junho de 2002

UMA PROPOSTA PARA A DESCRIÇÃO E BUSCA POR RECURSOS UTILIZANDO METADADOS XML/RDF EM REDES PEER-TO-PEER

Alexandre Takahashi Albert

Esta Dissertação 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.

_____

Professor Vitório Bruno Mazzola, Dr.

Banca Examinadora

_____

Professor Vitório Bruno Mazzola, Dr. (Orientador)

_____

Professor João Bosco Mangueira Sobral, Dr.

_____

Professor Rosvelter Coelho da Costa, Dr.

A Deus que, com sua luz, nos fortalece e protege.

Aos meus pais Eberhard Josef Albert e
Rosa Takahashi Albert que possibilitaram a chegada
e superação de mais esta etapa em minha vida

À minha esposa Consuelo que com sua compreensão
nas ausências e força nos momentos difíceis, tornou
possível minha inteira dedicação a este trabalho.

Agradeço à equipe da Kugel pelo apoio oferecido em minhas ausências na empresa e ao meu orientador,Prof. Vitório, que sempre esteve pronto a me atender. Agradeço também ao Prof. Claudiomir pelo grande apoio em minha trajetória acadêmica e profissional.

ÍNDICE

RESUMO..... **IX**

ABSTRACT..... **X**

LISTA DE FIGURAS..... **XI**

CAPÍTULO 1 – Introdução..... **12**

CAPÍTULO 2 – História da Internet..... **16**

2.1. Internet..... **17**

2.2. A World Wide Web – WWW..... **19**

2.3. Conclusão..... **22**

CAPÍTULO 3 – Conceitos Básicos..... **23**

3.1. A camada de transporte..... **23**

3.1.1. O protocolo TCP (Transmission Control Protocol)..... **24**

3.2. O protocolo HTTP e URLs..... **25**

3.3. DNS..... **27**

3.4. Conclusão..... **29**

CAPÍTULO 4 – Arquiteturas de Comunicação..... **30**

4.1. Arquitetura Cliente/Servidor..... **30**

4.2. Arquitetura de duas camadas (2-tier)..... **32**

4.3. Arquitetura de três camadas (3-tier)..... **33**

4.4. Arquitetura Peer-to-Peer (P2P)..... **34**

4.4.1. O funcionamento das redes P2P..... **36**

4.4.2. Protocolos P2P..... **38**

4.4.2.1. Mecanismos atuais de busca em protocolos P2P..... **39**

4.5. Conclusão..... **41**

CAPÍTULO 5 – O Protocolo Gnutella..... **42**

5.1. Conceitos básicos..... **43**

5.1.1. Mensagens em nível de aplicativo..... **43**
5.1.2. Difusão por TCP..... **44**
5.1.3. Roteamento dinâmico..... **44**
5.2. Comunicação Gnutella..... **46**
5.3. Especificação do protocolo Gnutella..... **49**
5.3.1. Pacote PING..... **52**
5.3.2. Pacote PONG..... **52**
5.3.3. Pacote QUERY..... **53**
5.3.4. Pacote QUERY_HIT..... **53**
5.3.5. Pacote PUSH..... **54**
5.4. Conclusão..... **55**
CAPÍTULO 6 – Metadados..... **56**
6.1. Definição de metadados..... **56**
6.2. A metalinguagem XML..... **57**
6.2.1. Notação..... **60**
6.2.1.1. Namespaces..... **63**
6.3. Iniciativa Dublin Core (DC)..... **64**
6.3.1. Qualificadores DC..... **69**
6.4. RDF (Resource Description Framework)..... **71**
6.4.1. O modelo de dados RDF..... **72**
6.4.2. Sintaxe RDF..... **74**
6.5. Conclusão..... **76**
CAPÍTULO 7 – Pesquisa Avançada........ **77**
7.1. Proposta..... **78**
7.2. Descrição dos recursos compartilhados: metadados..... **80**
7.2.1. Criação dos descritores..... **83**
7.3. Adendos ao protocolo Gnutella..... **84**
7.3.1. Adendo ao pacote de pesquisa: QUERY..... **85**

7.3.2. Adendo ao pacote de resposta: QUERY_HIT..... **89**

7.4. Processamento da pesquisa..... **93**

7.5. Discussão: desenvolvedores, usuários e aplicações..... **96**

8. CONCLUSÃO..... **99**

9. REFERÊNCIAS BIBLIOGRÁFICAS..... **101**

RESUMO

O propósito desta Dissertação de Mestrado é o de realizar um estudo sobre as redes peer-to-peer (P2P), os protocolos associados e as principais aplicações existentes. Nestas redes, o principal objetivo é a localização de recursos compartilhados e tal busca por recursos, atualmente, ocorre de forma relativamente limitada. Pretende-se também propor um mecanismo de pesquisas mais poderoso e flexível, utilizando-se as tecnologias estudadas: XML/RDF para a descrição de recursos e a arquitetura *Gnutella* como infra-estrutura peer-to-peer.

ABSTRACT

The purpose of this work is to realize a research about the peer-to-peer (P2P) networks, the associated protocols and about the peer-to-peer applications. On these networks, the main objective is the search for shared resources and such search, nowadays, occurs in a limitated way. Another purpose of this work is to propose a search mechanism more powerful and flexible, using the XML/RDF to describe the shared resources and the Gnutella architecture as the peer-to-peer infrastructure.

LISTA DE FIGURAS E TABELAS

Figura 1: Uma rede peer-to-peer local..... 12
Figura 2: Uma rede cliente/servidor..... 13
Figura 3: Mensagens na arquitetura cliente/servidor..... 30
Figura 4: Arquitetura cliente/servidor de duas camadas..... 33
Figura 5: Arquitetura cliente/servidor de três camadas..... 34
Figura 6: Uma rede P2P descentralizada..... 40
Figura 7: Uma rede P2P centralizada..... 41
Figura 8: Nó envia pesquisa em uma rede Gnutella..... 47
Figura 9: Nós repassam uma pesquisa em uma rede Gnutella..... 47
Figura 10: Nó com o arquivo solicitado responde à pesquisa..... 48
Figura 11: O nó que efetuou a busca faz o download do arquivo..... 49
Figura 12: Árvore de elementos XML..... 63
Figura 13: Representação da descrição de recursos em RDF..... 74
Figura 14: Exemplo de Tela de verificação/criação de metadados..... 84
Figura 15: Pacote QUERY tradicional..... 86
Figura 16: Pacote QUERY com adendo XML/RDF..... 87
Figura 17: Adendo de descritores no pacote QUERY_HIT..... 92
Figura 18: Nó recebe uma pesquisa tradicional..... 94
Figura 19: Nó recebe uma pesquisa avançada..... 95
Figura 20: Exemplo de um pacote de pesquisa QUERY (com adendo XML/RDF)..... 96
Tabela 1: Uma lista de produtos em um banco de dados relacional..... 59

Capítulo 1 – Introdução

> *"A Internet é um recurso compartilhado, uma rede cooperativa composta por milhões de hosts por todo o mundo." [ORAM, 2001]*

O conceito peer-to-peer (P2P) na caracterização de uma rede, não foi um conceito inventado para a Internet e, muito menos, é algo novo. Inicialmente, as redes peer-to-peer se limitavam a redes locais (LANs) onde se compartilhavam arquivos, impressoras e aplicações entre as estações. Este compartilhamento de recursos disponíveis nos nós de uma rede tornou popular alguns sistemas operacionais como, por exemplo, o Microsoft Windows. A própria Internet, conforme concebida originalmente no final da década de 60, era um sistema P2P. O objetivo inicial da Internet era compartilhar recursos de computação pelos Estados Unidos de forma que, se um dos pontos viesse a se desconectar, a rede continuasse operando normalmente. Nas redes P2P a principal característica é a ausência de uma autoridade central, isto é, uma máquina com papéis administrativos e possuidora de um nível maior na hierarquia em relação às demais máquinas, em outras palavras, um servidor. Em um escritório com uma rede P2P, por exemplo, um usuário em uma estação (X) pode imprimir um arquivo localizado no disco de uma estação (Y) em uma impressora remota conectada em uma terceira estação (Z), sem que para isso, deva existir um servidor central para realizar os compartilhamentos e as requisições dos serviços.

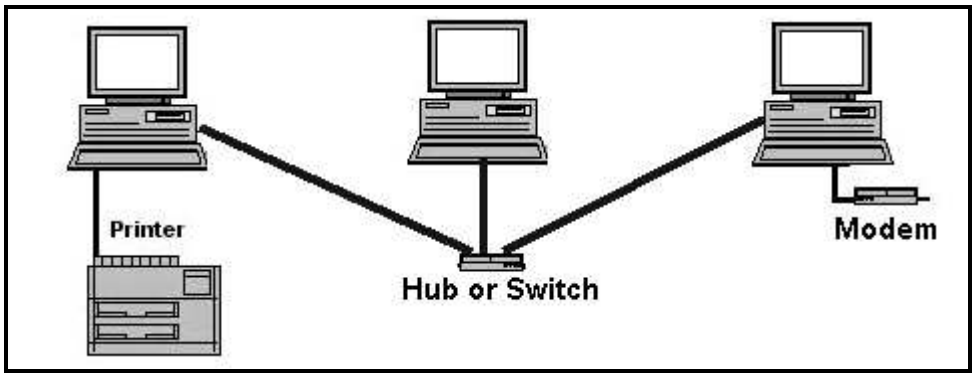

Figura 1: Uma rede peer-to-peer local

As próprias estações efetuam as requisições e as atendem. Já em uma arquitetura cliente/servidor, os recursos estão centralizados em uma máquina geralmente com maior

poder de processamento (servidor), máquina esta, responsável por executar/atender os pedidos das outras máquinas (estações). Ainda na arquitetura cliente/servidor, caso uma destas máquinas com papéis administrativos se desconecte por qualquer motivo, toda a rede é afetada, uma vez que os recursos e tarefas de administração (login, direitos, etc.) estão centralizados nelas.

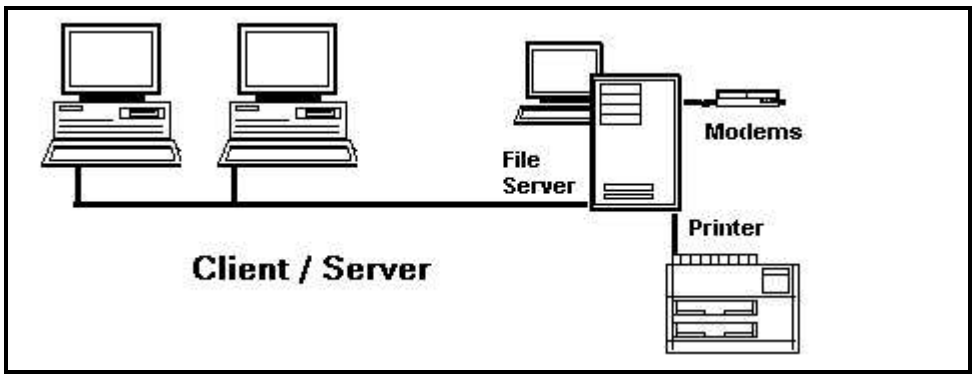

Figura 2: Uma rede cliente / servidor

De forma diferente, se em uma rede P2P um dos nós participantes se desconecte, somente os recursos que estavam compartilhados no nó deixam de ser utilizados, existindo ainda a possibilidade destes recursos estarem presentes em um outro nó. No final de 1999 e início de 2000, estas redes P2P saíram dos limites das redes locais e tomaram a Internet como um novo meio.

É comum encontrar definições precipitadas ou limitadas do termo *peer-to-peer*, colocando-o como sinônimo de aplicações que "explodiram" na *Web* nos últimos dois anos (*Napster* por exemplo). *Peer-to-peer* é algo mais abrangente, um conceito que engloba inúmeras outras tecnologias, entre elas a mais popular atualmente, a de compartilhamentos de arquivos na Internet na qual se enquadra o *Napster*. Justamente pela disseminação meteórica de aplicações como o *Napster* (as chamadas *killer applications)*, o termo *peer-to-peer* passou algum tempo na sombra deste tipo de tecnologia. Por outro lado, graças a essas aplicações é que as redes *peer-to-peer* começaram a ter seu verdadeiro poder revelado: o poder da utilização dos recursos distribuídos. O aproveitamento dos recursos das estações conectadas às margens da Rede (Internet), possibilitado pela teconologia P2P, disponibilizou milhares de *terabytes* de capacidade de armazenamento e distribuiu o processamento. Junte a isto, o fato de se

poder obter arquivos (músicas no caso do *Napster*) sem ter que se pagar por eles, e tem-se a explosão das redes P2P. Como já foi dito, o *Napster* é somente um dos tipos de tecnologia existentes dentro do termo P2P, sendo que, redes P2P totalmente descentralizadas (o *Napster* é uma aplicação P2P híbrida pois necessita de um servidor central para autenticação) são as mais populares atualmente. Ainda nos dias de hoje, inúmeras aplicações que possibilitam o compartilhamento e busca de recursos estão surgindo e, cada vez mais, um protocolo é utilizado: o **Gnutella**. Este protocolo, detalhado no capítulo 5, possibilita que um nó se conecte a um outro nó arbitrário já *online* e, neste mesmo instante, passe a fazer parte de uma rede composta por milhares de pontos, cada um compartilhando vários *megabytes* de recursos ou, *megahertz* de processamento. Nestas redes virtuais, nenhum nó possui um papel especial perante aos demais, todos são servidores e todos são clientes, buscando e fornecendo recursos.

Nas redes P2P, como o principal objetivo é o compartilhamento de recursos, um dos aspectos mais importantes, se não o mais importante, é a pesquisa por tais recursos. Veremos neste trabalho que os mecanismos de busca atuais fornecidos pelos protocolos e aplicações P2P são relativamente limitados. O número de recursos compartilhados é cada vez maior bem como a diversidade de tipos destes recursos. Até pouco tempo o compartilhamento nas aplicações P2P na Internet era de arquivos de áudio (MP3), vídeo e imagens, não mais do que isso. Com a descoberta do real poder destas aplicações, recursos dos mais diversos tipos estão sendo disponibilizados: documentos, livros, receitas culinárias, artigos científicos, fórmulas, etc. Enfim, qualquer tipo de informação pode ser disponibilizada para o compartilhamento mas, com os métodos atuais, sua localização ainda é uma árdua tarefa.

O objetivo geral deste trabalho é o de desenvolver um estudo referente às redes *peer-to-peer* (P2P), suas aplicações e os principais conceitos envolvidos. Para isso, são apresentados o funcionamento e os mecanismos das atuais redes e protocolos P2P, com o objetivo de expor as dificuldades e principais necessidades no que diz respeito à descrição e obtenção dos recursos.

No capítulo 2 é exposta a história da Internet, sua evolução, motivações e problemas até os dias atuais. Neste capítulo busca-se dar uma fundamentação necessária para o

desenvolvimento dos capítulos posteriores, contextualizando o surgimento das redes P2P.

O capítulo 3 expõe os conceitos básicos das redes de computadores com ênfase nas definições técnicas mais importantes no contexto da Internet e *World Wide Web*.

As arquiteturas de comunicação Cliente/Servidor e P2P são apresentadas no capítulo 4, que tem como objetivo mostrar as principais características de cada arquitetura para uma comparação entre elas. Outro objetivo do capítulo 4 é o de fornecer os conceitos básicos das redes P2P, para o desenvolvimento do capítulo referente ao protocolo Gnutella.

O Gnutella, protocolo mais utilizado nas redes P2P atuais, é detalhado no capítulo 5, onde são apresentados os principais aspectos do protocolo bem como sua última especificação. O objetivo deste capítulo é o de fornecer as informações necessárias sobre o Gnutella para que, no momento do desenvolvimento de um mecanismo de busca, verifique-se a necessidade ou não de sua alteração.

Metadados e a tecnologia RDF/XML são expostos no capítulo 6. Os conceitos colocados neste capítulo são imprescindíveis para o desenvolvimento e entendimento da proposta.

Tal proposta é apresentada no último capítulo deste trabalho, o capítulo 7. São apresentados os objetivos do desenvolvimento da Dissertação de Mestrado, expondo-se a linha de raciocínio bem como o mecanismo para buscas avançadas.

Capítulo 2 – História da Internet

A Internet conforme concebida originalmente no final da década de 60, era um sistema ponto-a-ponto [ORAM, 2001], com o objetivo de compartilhar recursos computacionais pelos Estados Unidos. A ARPANET (nome original da Internet) não interconectava os nós da rede em uma relação cliente/servidor, mas sim como níveis iguais em uma hierarquia. A Internet antiga também era muito mais aberta e livre do que a rede que conhecemos atualmente. Os *firewalls* eram desconhecidos até o final da década de 80 e, basicamente, quaisquer duas máquinas na Internet podiam enviar pacotes entre si.

Nos anos subseqüentes, a Internet foi se tornando cada vez mais restrita a aplicativos do tipo cliente/servidor. Tal mudança, em grande parte, foi baseada nos padrões de uso comuns naquele momento, a maioria dos quais, envolvendo *download* de dados. Os navegadores que surgiram no início da explosão comercial da Internet, baseavam-se (e são assim até hoje) em um protocolo simples: o cliente inicia uma conexão com um servidor, transfere dados por *download* e desconecta. Tal processo funciona para tudo, desde navegação por páginas na *Web* até a visualização de seqüências de vídeo.

A troca de informações através de uma forma eficiente e confiável, sempre foi um dos principais objetivos e um grande desafio tanto para desenvolvedores como usuários, seja no início da ARPANET, seja no ambiente cliente/servidor surgido anos mais tarde. Com a descentralização dos ambientes computacionais, ou seja, a troca das redes com grandes servidores acessados por clientes com pouco poder de processamento (arquitetura *mainframe*), por redes onde a capacidade de processamento e armazenamento das estações passou a ser maior , os dados e recursos passaram a estar distribuídos em muitos mais pontos, ou nós, destas redes.

Esta descentralização dos dados e a capacidade de um maior número de nós da rede processá-los e compartilhá-los, está fazendo com que a Internet volte a ser utilizada como concebida originalmente: como um meio para a comunicação entre máquinas que compartilham recursos em uma **relação de igualdade**. Esta arquitetura, que permite este compartilhamento de recursos diretamente entre nós de uma rede, é chamada de

arquitetura P2P (*peer-to-peer*). Tal arquitetura vem trazendo grandes promessas quanto às possibilidades de sua utilização em diversas áreas e, ao mesmo tempo, trazendo uma série de problemas a serem resolvidos e deficiências a serem supridas. Uma das principais deficiências diz respeito à localização de recursos em redes deste tipo e como já dito, neste trabalho busca-se propor um mecanismo para descrição dos recursos compartilhados através da utilização de metadados para uma localização mais precisa e flexível.

2.1. A Internet

Sobre a Internet, o artigo [ACM a, 1997] diz o seguinte:

> *"A Internet tem revolucionado o mundo da computação e das comunicações como nada antes visto. O telégrafo, telefone, rádio e o computador prepararam o palco para a integração, sem precedentes, de possibilidades na Internet. A Internet é ao mesmo tempo um meio de transmissão mundial, um mecanismo para disseminação da informação e uma forma para colaboração e interação entre indivíduos e seus computadores sem preocupação com localização geográfica."*

A Internet, antes de tudo, representa um dos mais bem sucedidos exemplos de investimento e comprometimento com a pesquisa e desenvolvimento de uma infra-estrutura de informação, como diz [ACM a, 1997]. Iniciando com a pesquisa da comutação de pacotes, o governo, a indústria e as universidades têm sido parceiros na evolução e liberação desta tecnologia.

O primeiro registro de interações sociais sobre uma rede, segundo [ACM a, 1997] foi uma série de memorandos escritos em 1962 no MIT (*Massachussets Institute of Technology*), discutindo conceitos da "*Galatic Network*" (Rede Galáctica). Ainda segundo [ACM a, 1997], tal rede, objeto de estudo de J.C.R. Licklider, seria uma série computadores interconectados globalmente que possibilitaria o acesso a dados em qualquer ponto. Em sua essência, este conceito era algo muito próximo da Internet que

temos hoje. Licklider conseguiu convencer alguns pesquisadores do DARPA (*Advanced Research Projects Agency*, que mais tarde voltou a se chamar ARPA) sobre a importância deste conceito de rede global ou galáctica. Em pouco tempo, a busca por uma técnica efetiva para a interconexão de redes, se tornaria o principal objetivo do DARPA [ACM d, 1988].

O ARPA surgiu no auge da guerra fria, em 1957, quando a então União Soviética lançou o *Sputnik*. As tradicionais redes telefônicas de comutação de circuitos eram consideradas vulneráveis, pois a perda de uma linha causaria o fim da conversação na rede. Imagine no caso de uma guerra nuclear ou ataque pesado. Esta era a principal preocupação na época. Em 1966, Lawrence G. Roberts, um pesquisador do MIT foi para o ARPA para desenvolver um conceito de rede de computadores e rapidamente, com uma equipe de pesquisadores, começou a pesquisa para a **ARPANET**. Foram desenvolvidos *switches*, na verdade minicomputadores, chamados IMPs (*Interface Message Processors*). Esses *switches* eram conectados por linhas de transmissão e cada IMP seria conectados a dois outros IMPs. Em dezembro de 1970 o protocolo *peer-to-peer* da ARPANET, chamado de NCP (*Network Control Protocol*), estava pronto. Neste ponto começaram a surgir inúmeras aplicações para a ARPANET, principalmente sistemas para correspondência eletrônica.

Segundo [ACM a, 1997], a ARPANET original cresceu para a Internet baseada na idéia que existiriam inúmeras redes independentes de desenho arbitrário. Em outras palavras, existiriam várias rotas para a transmissão dos dados. Após o desenvolvimento do NCP, a ARPANET logo cresceu para incluir redes de pacotes de satélite, redes de pacote via ondas de rádio e outros tipos de redes. Hoje em dia a Internet incorpora uma idéia chave: rede de arquitetura aberta. Neste conceito, redes de diferentes arquiteturas podem utilizar um metanível para interagir umas com as outras.

[ACM a, 1997] diz que a idéia da rede de arquitetura aberta foi introduzida por Kahn no final de 1972, logo após sua chegada ao DARPA e que esta idéia seguia quatro regras:

1. Cada rede distinta tinha que se manter por si própria, e mudanças internas não seriam requeridas para conectá-las à Internet;

2. Comunicação seria baseada na técnica do melhor esforço. Se um pacote não atingisse seu destino , ele poderia ser rapidamente retransmitido da origem;
3. Caixas preta (mais tarde chamadas de roteadores e *gateways*) deveriam ser utilizadas para conectar as redes. Nenhuma informação deveria ser retida por estes *gateways*;
4. **Não existiria um controle global das operações.**

Segundo [ACM a, 1997] logo após a chegada de Kahn ao DARPA, ele solicitou que um dos desenvolvedores do protocolo NCP, Vinton Cerf, fosse para a sua equipe. A experiência de Cerf com protocolos aliada ao conhecimento de Kahn em arquiteturas de comunicação, fez surgir o que se tornaria o TCP / IP (*Transmission Control Protocol / Internet Protocol*). O TCP/IP é exposto no capítulo 3 deste trabalho.

Com a adoção do protocolo TCP/IP como protocolo oficial da ARPANET, o número de redes, máquinas e usuários conectados cresceu rapidamente [TANENBAUM, 1997,p. 59]. Em meados da década de 80, este conjunto de redes passou a ser considerado como uma inter-rede ou a **Internet**. O crescimento passou a ser exponencial e segundo [TANENBAUM, 1997, p.59] em 1990, a Internet já conectava 3000 redes e 200 mil computadores. Em 1992, o milionésimo nó foi conectado e em 1995 o número de nós passava da casa dos milhões. A conexão de redes já existentes em todo o mundo, contribuiu para este rápido crescimento.

Como os elementos básicos que formam a Internet são o modelo de referência e a pilha de protocolos TCP/IP, um computador estar conectado à Internet significa que ele possui um endereço IP, executa a pilha de protocolos e pode enviar pacotes a qualquer outro computador na Rede. Até o início da década de 90, segundo [TANENBAUM, 1997, p. 61], a Internet era um verdadeiro reduto de pesquisadores ligados à universidades, até que a WWW (*World Wide Web*) atraiu milhares de novos usuários.

2.2. A World Wide Web – WWW

Geneticistas trocam dados sobre o genoma com colegas. Fãs dos Beatles falam de seus discos. Milhares de programadores compartilham código para criar sistemas gratuitos e livres. Esta possibilidade fez da Internet uma revolução no mundo das comunicações. A arquitetura que permite o acesso a documentos/recursos espalhados por milhares de máquinas e vinculados a milhões de páginas na Internet é a World Wide Web (WWW). Segundo [SOARES, 1995, p. 416], a WWW foi desenvolvida para permitir o acesso à informações organizadas na forma de hipertexto que, basicamente, são documentos com links a outros documentos (hiperlinks).

A Web, ou WWW, começou em 1989 no CERN (Centro Europeu Para Pesquisa Nuclear). Como os projetos do CERN eram complexos e desenvolvidos por pesquisadores de diferentes países, surgiu a necessidade de permitir que eles colaborassem uns com os outros através da troca de relatórios, plantas, desenhos, fotos e outros tipos de recursos. Segundo [TANENBAUM, 1997, p.778], em 1991 na conferência de San Antônio, Hypertext 91, ocorreu a primeira demonstração pública de uma "teia" de documentos vinculados em modo texto. O desenvolvimento deste protótipo prosseguiu e, em 1993, surge a primeira interface gráfica para a WWW: o Mosaic. Conforme [TANENBAUM, 1997, p.778], o Mosaic tornou-se tão popular que seu criador, Marc Andreessen, deixou o *National Center for Supercomputing Applications*, onde o Mosaic foi desenvolvido, para fundar sua própria companhia: a Netscape. Em 1994, o CERN e o M.I.T. assinaram um acordo criando o *World Wide Web Consortium* (W3C), uma organização voltada para o desenvolvimento da Web e para a padronização de protocolos.

A requisição de recursos na WWW opera na forma cliente/servidor: um programa cliente WWW requisita, pelo nome, um documento localizado em algum nó servidor na Internet. Estes recursos disponíveis na WWW, que podem ser menus, imagens, documentos, etc, são endereçados através identificadores únicos chamados URLs, que serão detalhados no próximo capítulo. Como base para recuperação da informação os projetistas da WWW utilizaram o conceito do hipertexto, o qual permite interfaces do tipo "aponte e clique", como os browsers conhecidos e amplamente difundidos atualmente. Do ponto de vista do usuário, a Web é uma vasta coleção de documentos, normalmente chamados de páginas [TANEMBAUM, 1997, p. 778]. Cada documento

desses pode conter ligações (links) com outros documentos localizados em qualquer lugar do mundo, formando a Web. Esta ligação entre documentos é possível porque eles são escritos em hipertexto.

As páginas são visualizadas em softwares denominados browsers. Hoje os mais utilizados são o Internet Explorer (Microsoft) e o Netscape (Netscape Co.). No browser, o usuário digita o endereço de um determinado recurso (página, arquivo, imagem, etc.) e aguarda sua exibição. A localização geográfica da página é transparente ao usuário e a busca e exibição fica por conta do browser. Ou seja, ele é quem deve buscar, "resolver" o endereço digitado e localizar o documento. Como as páginas podem conter links a outras páginas, a maioria dos browsers oferece controles de navegação como voltar à página anterior, avançar, ir para a pagina inicial do usuário e marcar páginas já visitadas.

Os sites na WWW que disponibilizam páginas aos usuários têm processos servidores que ficam aguardando as suas requisições. Depois de estabelecida a conexão de um cliente, este envia a solicitação de uma página e o servidor envia uma resposta. Esta reposta pode ser a página solicitada ou alguma mensagem de erro. O protocolo que define as requisições e as respostas válidas é chamado de HTTP e será detalhado no próximo capítulo. O processo de requisição e obtenção de uma página ou qualquer outro recurso na Web ocorre da seguinte forma:

1. O browser determina o identificador (endereço) digitado pelo usuário;
2. O browser pergunta ao DNS (servidor responsável por informar a localização de máquinas a partir de seus nome (detalhado no capítulo 3) qual o endereço IP do identificador digitado;
3. O browser estabelece uma conexão TCP com o nó apontado pelo endereço IP retornado pelo DNS;
4. Em seguida, o browser envia uma requisição para obter a página desejada;
5. O servidor envia a página e a conexão é liberada;
6. Com a página em mãos, o browser só precisa mostrá-la ao usuário.

Uma prova da eficiência da WWW é a importância e a atenção que o comércio mundial vem dando a ela. Inúmeras transações como comércio eletrônico, internet banking,

pesquisas, colaboração entre empresas, são realizadas na Web. Tudo isto em cima de um conceito simples porém muito eficiente, possibilitado por interfaces "aponte e clique", documentos disponibilizados em servidores e pelo protocolo HTTP.

2.3. Conclusão

Desde o surgimento da ARPANET até a Internet que conhecemos hoje, o compartilhamento de recursos/informações é o foco principal das pesquisas e trabalhos desenvolvidos. Com o surgimento da Internet, máquinas com qualquer localização geográfica passaram a ser comunicar com quaisquer outras máquinas. Mais tarde a *World Wide Web* possibilitou uma certa estruturação de recursos e uma localização mais eficaz das informações disponibilizadas na Internet. No capítulo seguinte, veremos os principais conceitos envolvidos quando se trata da comunicação e localização de recursos na Internet.

Capítulo 3 – Conceitos Básicos

A estruturação da rede em camadas consiste em um conjunto de camadas hierárquicas, cada uma sendo construída utilizando as funções e serviços oferecidos pelas camadas inferiores [SOARES, 1995]. Cada camada pode ser entendida como um processo que se comunica com processos correspondentes, ou seja, de mesma camada em outras máquinas.

Um ou mais protocolos podem ser especificados em um determinado nível *N*. Neste caso diz-se que o protocolo é um protocolo de nível *N*. Projetar protocolos em níveis foi a saída encontrada para que uma alteração em um determinador nível, não causasse um impacto muito grande, desde que as interfaces entre os níveis estejam bem definidas. O padrão definido pela ISO (*International Organization for Standardization*) foi o modelo denominado OSI (*Open Systems Interconnection*). O modelo OSI propõe uma estrutura com sete camadas como referência para a arquitetura dos protocolos de redes: aplicação, apresentação, sessão, transporte, rede, enlace e física.

Na próxima seção deste trabalho trataremos com mais detalhes a camada de transporte. Isto porque a camada de transporte é a camada responsável pela comunicação fim-a-fim ou ponto-a-ponto. Nela estão definidos os protocolos que garantirão que um pacote chegue ao seu destino ou, que seja tratado corretamente em caso negativo.

3.1. A Camada de Transporte

O nível de rede não garante necessariamente que um pacote chegue a seu destino, e pacotes podem ser perdidos ou mesmo chegar fora da seqüência original de transmissão [SOARES, 1995]. A camada de transporte realiza uma comunicação fim-a-fim, logo, é o nível que contém os protocolos necessários para uma comunicação confiável entre máquinas em uma rede.

Protocolos da camada de transporte possibilitam a comunicação ponto-a-ponto entre dois ou mais nós. No modelo de referência de sete camadas OSI, a camada de transporte

é a de nível mais baixo que opera na comunicação fim-a-fim entre nós [ACM b, 1999]. Esta camada se localiza na fronteira entre estes nós e os dispositivos como roteadores, bridges e links de comunicação, que movem a informação. Um bom serviço de transporte permite às aplicações utilizarem um conjunto padrão de primitivas e serem executadas em diferentes tipos de redes, sem se preocupar com as interfaces. Essencialmente, a camada de transporte isola as aplicações da tecnologia e projeto da rede. Dúzias de protocolos de transporte têm sido desenvolvidos nas duas últimas décadas [ACM b,1999]. Da perspectiva de um programador de aplicações, a camada de transporte permite a comunicação entre processos quase sempre sendo executados em diferentes nós.

3.1.1. O protocolo TCP (Transmission Control Protocol)

Nas últimas duas décadas o protocolo da Internet (TCP/IP) tornou-se o protocolo mais utilizado. Pode-se destacar alguns outros protocolos também utilizados como o User Datagram Protocol (UDP), o SNA da IBM e o DECnet da Digital. Devido à sua ampla utilização, principalmente por parte da maioria das aplicações P2P na Internet, trataremos com mais detalhes do TCP.

O TCP é um protocolo orientado à conexão que fornece um serviço confiável de transferência de dados ponto-a-ponto [SOARES, 1995]. Como dito anteriormente, o TCP isola as aplicações das redes nas quais elas serão executadas, ou seja, o TCP está localizado entre os processos das aplicações e os processos da arquitetura da rede (protocolo IP). Pode se fazer uma analogia da interface entre a aplicação e o TCP com um conjunto de chamadas que os sistemas operacionais fornecem aos processos para a manipulação de arquivos. Por exemplo, no TCP existem chamadas para abrir e fechar conexões e para enviar e receber dados (pacotes).

O TCP não exige um serviço de rede confiável para operar, logo ele se responsabiliza pela entrega de forma correta dos pacotes. Por exemplo, ele deve garantir que o pacote chegue ao seu destino. Para isso, quando uma entidade TCP transmite um pacote, ela coloca uma cópia em uma fila de retransmissão e dispara um temporizador. Se um

reconhecimento da recepção correta do pacote chegar, sua cópia é retirada da fila de retransmissão. Caso o pacote não chegue ao destino antes do temporizador expirar (timeout), a cópia em fila é retransmitida. O TCP envia juntamente com os pacotes números de seqüência, que são utilizados pelo receptor para ordená-los caso tenham chegado fora de ordem. Isto porque cada pacote pode ter seguido uma rota diferente até o destino.

Para permitir que vários processos em um único nó possam simultaneamente transmitir cadeias de dados (pacotes), o TCP utiliza o conceito de porta [TANENBAUM, 1997]. Cada uma das aplicações ou processos de aplicações atendidos pelo TCP, são identificados por uma porta distinta. No TCP todas as conexões são full-duplex, isto é, o tráfego dos pacotes pode ser feito em ambos os sentidos ao mesmo tempo. Quando dois processos desejam se comunicar, as instâncias do TCP nas extremidades devem estabelecer uma conexão. As aplicações transmitem seus dados fazendo chamadas ao TCP, passando como parâmetros os buffers onde estão os dados. O TCP empacota os dados destes buffers em segmentos e os envia ao destino determinado pelo endereço IP. O TCP receptor desempacota os segmentos, coloca os dados em buffers e notifica o processo da aplicação do evento.

3.2. O protocolo HTTP e URL's

O protocolo de transferência padrão da Web é o HTTP (HyperText Transfer Protocol). Como visto anteriormente neste trabalho, os dados transferidos via HTTP podem ser textos não estruturados, hipertextos, imagens ou qualquer outro recurso. Podemos dividir o protocolo HTTP basicamente em dois itens: um conjunto de solicitações dos browsers (clientes) aos servidores e um conjunto de respostas que retornam destes servidores.

Segundo [TANENBAUM, 1997, p. 787], as novas versões do HTTP aceitam dois tipos de solicitações: simples e completas. As solicitações simples consistem apenas de uma linha GET que identifica a página/recurso desejado, sem a versão do protocolo. Já a

reposta é formada apenas pela página/recurso, sem cabeçalhos. Por exemplo, para se obter a página Receita.Html após uma conexão já ter sido estabelecida:

GET /documentos/WW/receitas/Receita.Html

Hoje em dia, as solicitações simples são utilizadas somente para compatibilidade com versões mais antigas do HTTP [TANEMBAUM, 1997, p. 787]. As solicitações completas são indicadas pela presença da versão do protocolo na linha de solicitação GET. Este tipo de solicitação pode conter várias linhas, terminando em uma linha em branco. Os nomes fazem distinção entre maiúsculas e minúsculas. Por exemplo, GET é um método válido, mas get não. Assim como o GET, existem outros métodos disponíveis no HTTP:

- GET – solicita a leitura de uma página da Web;
- HEAD – solicita a leitura de um cabeçalho de página Web, sem a página propriamente dita. Esse método é utilizado para obtenção de informações que constam no cabeçalho como data de modificação, informações para índices ou para testar a validade de um endereço (URL);
- PUT – solicita o armazenamento de uma página. Esse método possibilita a criação de páginas em servidores remotos.;
- POST – acrescenta um recurso. Semelhante ao PUT só que, em vez de substituir dados existentes, novos dados são acrescentados;
- DELETE – remove a página da Web;
- LINK – conecta dois recursos existentes;
- UNLINK – desconecta dois recursos.

Um servidor só enviará uma página mediante a solicitação GET e, se os dados dela tiverem sido alterados desde a data fornecida pelo browser (a partir da página em seu cache). Esse mecanismo evita a transferência desnecessária de páginas. Os padrões HTTP descrevem os cabeçalhos e corpos das mensagens, tanto as de solicitação como as de resposta.

Em diversas partes deste capítulo foi mencionada a palavra endereço para os identificadores das páginas na Web. Estes identificadores, digitados pelos usuários nos browsers ou presentes em links entre documentos, são chamados de URLs (Uniform Resource Locators). Já na concepção da Web, as páginas precisaram ser identificadas para que sua localização fosse possível. Mesmo que cada página na Web tivesse um nome único o problema da localização não seria resolvido. Como saber em qual servidor está a página Receita.Html?

A solução encontrada possibilitou a resposta a três questões: qual a página desejada?; onde se localiza?; como ela pode ser acessada? A cada página/recurso é atribuído um URL que funciona como o nome universal da página. Por isto, a sigla URL muitas vezes é traduzida como Universal Resource Locator. Os URLs possuem três partes: o protocolo, o nome DNS da máquina onde o recurso está disponível e um nome para o recurso. Por exemplo, o URL para uma receita poderia ser:

http://www.delicia.com.br/Receita.Html

Facilmente identificamos as três partes: o protocolo (http), o nome da máquina onde a receita está armazenada (www.delicia.com.br) e o nome do arquivo da receita (Receita.Html). Quando um URL é digitado em um browser, ele verifica o protocolo, localiza a máquina perguntando ao servidor DNS o seu endereço IP, estabelece uma conexão e faz a solicitação através do método GET do HTTP.

3.3. DNS

As aplicações existentes na Internet, geralmente, fazem referência a nós, caixas postais e outros recursos, utilizando strings ou nomes ao invés dos endereços IP. Por exemplo, em um browser, ao invés do usuário digitar http://200.247.10.1 (nó fictício) ele digita http://www.mesusite.com.br. [TANENBAUM, 1997, p. 709] diz:

> *"Todavia, a rede em si só compreende endereços binários; portanto, é necessário algum tipo de mecanismo para converter os strings de nomes em endereços de rede..."*

Na ARPANET, havia simplesmente um arquivo, hosts.txt, que listava todos os nós e seus endereços IP. Todas as noites, todos o acessavam no site onde ele era mantido. A partir do momento que milhares de estações de trabalho foram conectadas à rede, viu-se que esta estratégia deveria ser abandonada. A saída encontrada foi o DNS (Domain Name System).

Em essência, o DNS consiste em um esquema de atribuição de nomes hierárquico. Para mapear um nome em um endereço IP, um programa aplicativo chama um processo denominado resolvedor passando o nome do nó a ser resolvido (ter seu endereço IP retornado). O resolvedor envia um pacote para um servidor DNS, que procura o nome e retorna o endereço IP. Com o endereço IP em mãos, o aplicativo pode estabelecer a conexão TCP desejada.

O gerenciamento do grande número de nomes que está constantemente mudando, é feito de forma hierárquica. Essa hierarquia é composta, na Internet, por domínios. Analisando-se o endereço www.meusite.com.br podemos identificar os tipos de domínios: os domínios primários (.br e .com), identificam o país e classe (tipo) do site respectivamente (.br de Brasil e .com de site comercial); o domínio do próximo nível (.meusite) identifica o site em si, que possui sua localização já restringida no país Brasil e em sites comerciais. Em outras palavras, a resolução de um nome de domínio ocorre de trás para frente (ou da direita para a esquerda). A atribuição de nomes leva em conta as fronteiras organizacionais, e não as redes físicas.

Caso um servidor DNS não possua informação sobre um domínio (nome) solicitado, ele repassa a pesquisa para outro servidor DNS em primeiro nível. Este processo se repete até que um servidor responda retornando o endereço IP do nome solicitado ou, até que o temporizador do cliente que solicitou o endereço expire.

3.4. Conclusão

A pilha de protocolos TCP/IP juntamente com o protocolo de transferência de documentos HTTP, possibilita a obtenção de qualquer documento disponibilizado na Internet. Foi visto que cada documento/recurso possui um localizador único: a URL. Através de uma URL, um browser localiza recursos na Internet solicitando a servidores DNS o endereço IP correspondente à URL digitada. Este mecanismo de solicitação e resposta para obtenção de endereços e de documentos, é denominado de cliente / servidor. No capítulo 4 são apresentados os fundamentos da arquitetura cliente/servidor e a arquitetura peer-to-peer para possibilitar uma comparação entre elas.

Capítulo 4 – Arquiteturas de Comunicação

As redes originais de PC's eram baseadas em arquiteturas de compartilhamento de arquivos (file sharing architecture). Esta arquitetura funciona perfeitamente enquanto o uso de compartilhamentos é baixo, a taxa de atualização é baixa e também o volume de dados a ser transferido é baixo. No final da década de 80 e princípio da década de 90, as redes locais de PC's (PC LAN) começaram a se adaptar a uma nova arquitetura: Client / Server [ACM c, 1992].

4.1. Arquitetura Cliente/Servidor

O termo Client/Server é utilizado para descrever a comunicação entre processos computacionais que são classificados como consumidores de serviços (clientes) e provedores de serviço (servidores). Clientes e servidores são módulos funcionais com interfaces bem definidas, e, estes módulos podem ser compostos por software, hardware ou pela combinação dos dois elementos. Cada relacionamento cliente/servidor é estabelecido quando o módulo cliente inicia um serviço de requisição e o módulo servidor trata de responder à requisição executando ou não a tarefa solicitada. Requisições a um sistema gerenciador de banco de dados (SGBD) pode-se colocar como o exemplo mais conhecido.

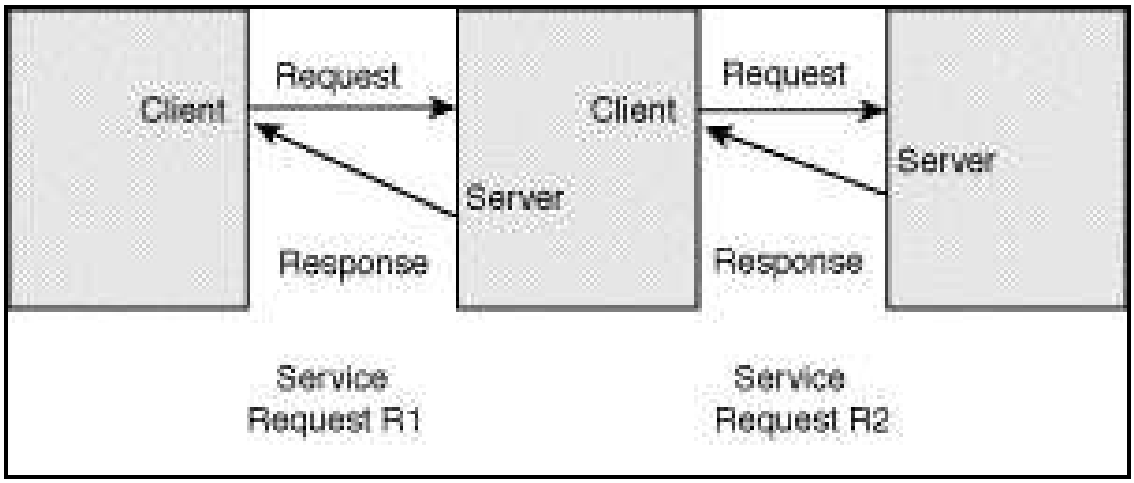

Figura 3: Mensagens na arquitetura Client/Server

A troca de informações entre os módulos cliente e servidor ocorre estritamente via mensagens (figura 3), ou seja, nenhuma informação é trocada por meio de variáveis (regiões de memória) comuns a ambos. A requisição do cliente juntamente com informações adicionais (endereço do cliente, usuário, senha, etc.), são enviadas ao servidor. Este, analisa a mensagem, executa a tarefa solicitada e por intermédio de outra mensagem envia a reposta. Esta troca de mensagens é a característica crucial da arquitetura cliente/servidor e as mensagens trocadas são normalmente interativas. Em outras palavras, não há processamento offline (com poucas exceções), onde o cliente envia a requisição, desconecta e busca o resultado em um momento posterior. Este modelo de requisição-resposta, reduz o tráfego na rede justamente pelo fato de somente a requisição do cliente e a resposta do servidor serem transmitidos pela rede, enquanto na arquitetura file-sharing, todos os arquivos utilizados trafegavam pela rede.

Como exemplos para a arquitetura cliente/servidor podemos citar: os servidores de arquivos que disponibilizam para as estações serviços de armazenamento; servidores de impressão que, mediante as requisições dos clientes gerenciam os trabalhos de impressão; servidores de banco de dados que recebem requisições na linguagem SQL (Structured Query Language) e enviam os registros solicitados da base de dados mantida por eles. Além de uma infinidade de outros tipos de servidores, cada um dos quais responsáveis por executar determinados serviços para seus clientes.

Algumas características de um ambiente cliente/servidor:

- Um servidor provê serviços a clientes. Estes serviços podem requer muito poder de processamento do servidor (servidor de banco de dados, servidor de imagens) ou pouco processamento (servidor de impressão, servidor de arquivos);
- Um servidor somente responde às requisições dos clientes. A conversação é, quase sempre, iniciada pelo cliente;
- Para o cliente fica transparente a forma como o servidor executará a tarefa solicitada ou até mesmo onde se localiza o servidor. O cliente simplesmente recebe a reposta à requisição feita;

- A arquitetura cliente/servidor divide uma aplicação em dois processos (cliente e servidor) distintos que podem ser executados em diferentes máquinas conectadas pela rede.

Podemos classificar a arquitetura client/server por camadas:

4.2. Arquitetura de duas camadas (2-tier):

Na arquitetura client/server de duas camadas o cliente se localiza em uma estação da rede e o provedor de serviços (sistema gerenciador de banco de dados por exemplo) se localiza em uma máquina mais potente da rede, geralmente chamada de servidor (figura 4). Mais potente porque deve conseguir atender as requisições de N clientes [SCHUSS]. A arquitetura de duas camadas começa a apresentar problemas com o aumento no número de clientes e, duas limitações principais causam queda de performance e escalabilidade: a primeira é que, como o servidor começa a ficar sobrecarregado com as requisições, este deve manter a conexão com os clientes que estão na fila através de mensagens "keep-alive" para que estes não se desconectem por timeout, ou seja, clientes que ficam por longos períodos aguardando uma resposta do servidor, desistem da requisição. A segunda limitação, a de escalabilidade, é que a solução adotada para prover os serviços utiliza um processamento proprietário. Desta forma, por exemplo, uma aplicação (cliente) pode ficar presa a um sistema gerenciador de banco de dados (servidor).

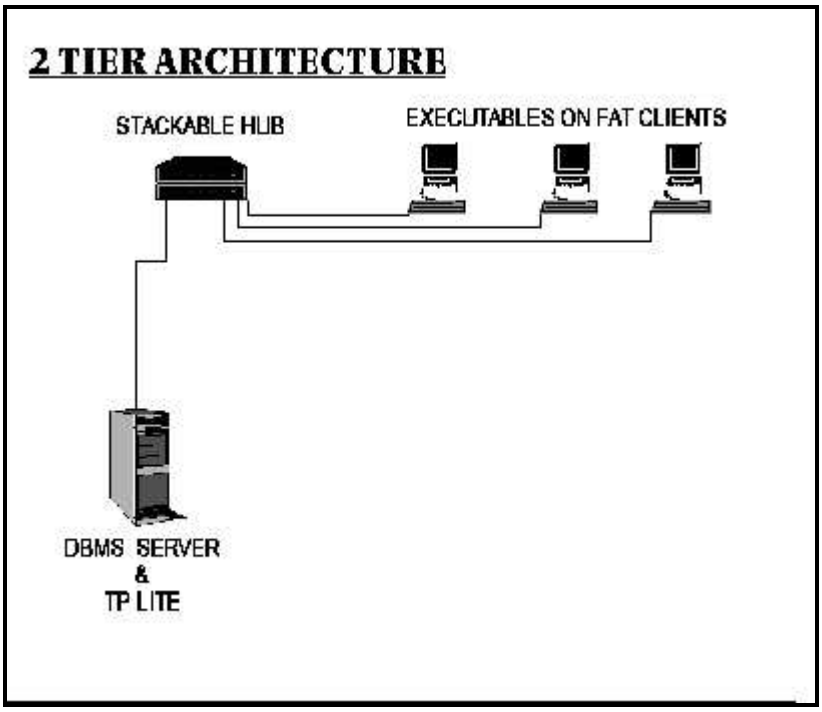

Figura 4: Arquitetura Client/Server de duas camadas

4.3. Arquitetura de três camadas (3-tier):

Devido às limitações da arquitetura de duas camadas uma terceira camada ou camada do meio (middle tier) foi adicionada entre o cliente e o servidor (figura 5). Esta terceira camada pode possuir várias atribuições: pesquisa, execução de aplicações, controle do banco de dados, entre outras. A utilização da tecnologia cliente/servidor com uma terceira camada possibilita um ganho considerável de performance e flexibilidade. Com três camadas a camada intermediária fica responsável pelas pesquisas no servidor, possibilitando desta forma que um processo antes síncrono, na abordagem de duas camadas, se torne assíncrono. Isto porque o cliente pode enviar sua requisição à esta camada intermediária e continuar com outras tarefas tendo a certeza que uma resposta apropriada será retornada [SCHUSS]. A camada intermediária, então, fica com a responsabilidade de efetuar as solicitações dos cliente ao servidor.

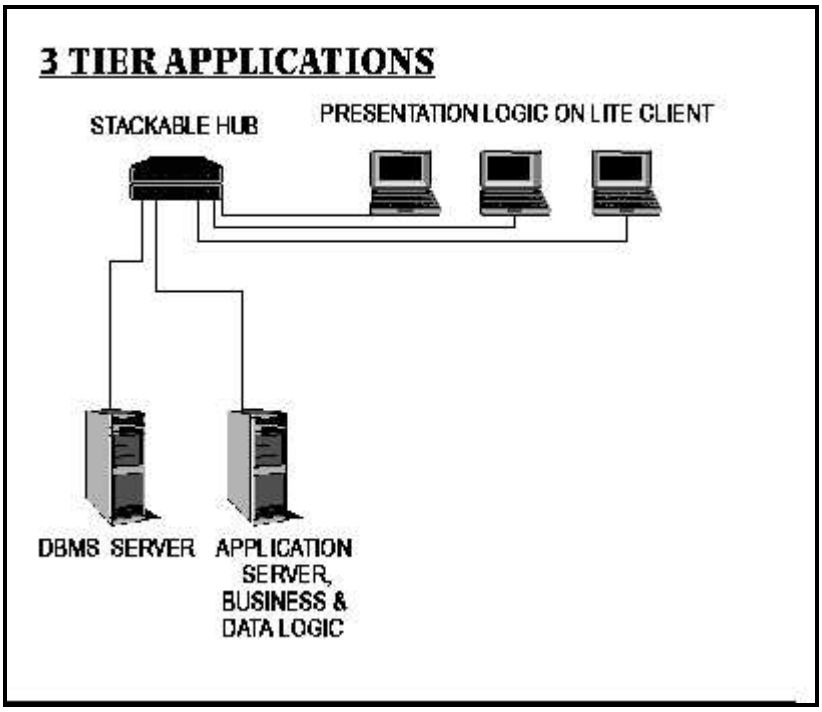

Figura 5: Arquitetura Client/Server de três camadas

Os diversos tipo de arquiteturas (2-tier, 3-tier, n-tier) cliente/servidor podem ser combinados para atender à necessidades específicas. Na Internet as aplicações cliente/servidor estão por toda parte, quase tudo, senão tudo, o que fazemos na Internet é através de aplicações clientes que solicitam serviços a servidores. Seja o download de um arquivo, o envio de um e-mail e até a navegação pelas páginas HTML. Para muitos tipos de aplicações na Internet, a arquitetura cliente/servidor continuará sendo por muito tempo a melhor opção, enquanto para outras, uma arquitetura que promete mudar a forma como utilizamos a Rede vem trazendo soluções, recursos, discussões e, principalmente, descentralização dos recursos. A arquitetura Peer-to-Peer.

4.4. Arquitetura Peer-To-Peer (P2P)

Sobre peer-to-peer, Gene Kan diz o seguinte em [ORAM, 2001]:

> *"A Internet do futuro vai parecer diferente do que é hoje. A Web não passará de um pontinho na cronologia do desenvolvimento tecnológico. Foi só algo bastante real nos últimos seis anos. Pense na Web como o equivalente na Internet ao telégrafo: é muito útil, mas ainda muito bruta. As tecnologias peer-*

> *to-peer e a experiência adquirida com a Gnutella, a Freenet, o Napster e as mensagens instantâneas vão reformatar a Internet drasticamente."*

A evolução dos recursos computacionais, principalmente dos nós conectados à Rede, e o crescimento da Internet, aliados à eterna necessidade da obtenção do maior número de informações com o menor custo possível, fez uma nova tecnologia mostrar-se promissora neste contexto: *as redes peer-to-peer (P2P).* Tais redes possibilitam que seus nós exerçam papel tanto de cliente como o de servidor, trocando mensagens e compartilhando recursos diretamente com outros nós. Redes P2P de pequena escala não são novidade. Elas têm sido utilizadas por anos em ambientes de trabalho locais, tornando popular sistemas operacionais como o da Microsoft e Apple, para o compartilhamento de recursos como impressoras, discos e modems na LANs.

Nos últimos três anos (1999-2001) muito têm-se falado das redes peer-to-peer e, muitas vezes, utilizando este termo de forma incorreta limitando seu real significado. Muitos sistemas existentes são colocados como sinônimos de peer-to-peer quando na verdade são elementos que trouxeram ,e ainda trazem, grandes contribuições para a formação desta tecnologia. Exemplos destes sistemas: Napster (compartilhamento de arquivos de áudio), SETI@home (projeto que busca vida extra-terrestre utilizando a colaboração de nós na Internet) e Groove Networks (sistema de colaboração). O Napster, em particular, devido ao seu grande sucesso, ainda hoje é confundido com a tecnologia peer-to-peer. O Napster possui grandes méritos, mas a tecnologia peer-to-peer é muito mais do que o compartilhamento de arquivos, é uma mudança na forma de se utilizar a grande Rede.

A alteração que vem ocorrendo no uso da Internet é um fenômeno que não pode ser atribuído ao simples fato do surgimento da arquitetura peer-to-peer. A arquitetura P2P vem é resolver problemas e suprir necessidades surgidos com esta nova forma de se utilizar a Rede. O que mudou foram os nós que compõe os sistemas (ou redes) P2P: PC's conectados à Internet, que antes eram relegados a simples clientes, localizados às margens da Internet e excluídos do DNS (Domain Name System) por não terem endereços fixos. A tecnologia P2P vem então dar poder a esses nós e ao mesmo tempo tirar proveito deste poder (armazenamento e processamento) para atingir seus objetivos. Permitir que os nós da Rede tenham autonomia para compartilhar recursos entre si,

significa operar em um ambiente de conectividade instável e endereços IP imprevisíveis. Em outras palavras, num determinado momento um nó pode estar conectado à rede com um determinado endereço IP e já em outro não estar mais presente ou estar com outro endereço.

Os sistemas peer-to-peer andam de mãos dadas com o termo descentralização. Em um sistema descentralizado, não só cada nó é um participante igual, como não há nós com recursos especiais ou papéis administrativos. O P2P é uma maneira de descentralizar não apenas recursos, mas também custos e administração [ORAM, 2001].

4.4.1. O funcionamento das redes P2P

Até 1994, a Internet tinha um modelo básico de conectividade. Pressupunha-se que as máquinas estivessem sempre ligadas, conectadas e com endereços IP permanentes. O DNS foi projetado para esse ambiente, no qual uma alteração no endereço IP seria considerada anormal. Por alguns anos, tratar os PC's como clientes "burros", porém caros, funcionou bem. Entretanto, à medida que o hardware e o software eram aprimorados, os recursos não utilizados que existiam por trás destes dispositivos passaram a se mostrar algo de qual se valesse a pena tirar proveito. Em 1996 o ICQ (na pronúncia inglesa, "*I Seek You*"), precursor dos sistemas de mensagens instantâneas, marcou a primeira vez na qual dois PC's trocaram mensagens e foram utilizados diretamente por usuários casuais. Para este feito, o ICQ simplesmente ignorou o DNS e criou seu próprio diretório de endereçamento atualizado em tempo real. Toda vez que um nó se conectava à rede, este diretório era atualizado com o seu endereço IP momentâneo. Acredita-se que, quando novos endereços IP's forem criados, o antigo regime de um endereço para cada dispositivo poderá ser restaurado.

Não se pode explicar o funcionamento das redes P2P sem se falar no sistema de maior crescimento em número de usuários nos últimos anos. Sistema este que acabou rotulando a própria arquitetura P2P: o Napster. O Napster é peer-to-peer porque os endereços de seus nós ignoram o DNS e porque, uma vez que o Napster define os endereços IP's dos PC's que hospedam uma determinada música, ele passa o controle da

transferência dos arquivos para os nós. O resultado é que, com a capacidade dos nós de hospedar arquivos de áudio e gerenciar as transferências, os usuários destes nós têm acesso a vários terabytes de armazenamento sem custo adicional (imagine armazenar e gerenciar está quantidade de arquivos em um servidor central). Um bom exemplo da autonomia e independência dos endereços estáticos está na tecnologia do ICQ. No ICQ o endereço viaja com o usuário [ORAM, 2001]. Se em outro nó o usuário se logar com seu "endereço", automaticamente os demais nós passarão a "enxergá-lo" neste novo local de forma transparente. O endereço, neste caso, nada mais é que um identificador único do usuário.

O entusiasmo nos últimos dois anos pelo P2P levou a várias declarações irrefletidas sobre sua superioridade para a maioria das classes de aplicativos em rede. Atualmente, o P2P é distintamente ruim para muitos tipos de aplicativos, principalmente os que envolvem mecanismos de pesquisa em grandes bases de dados. A maioria das pesquisas funciona melhor quando efetuadas em um banco de dados central. O Naspter, assim como outros aplicativos P2P, mistura centralização com descentralização. Centralização pois, como um mecanismo de busca, mantém em um servidor central uma lista de músicas e quais nós da rede hospedam essas músicas. Descentralização pois, uma vez identificado o nó onde se encontra uma música desejada por outro nó, a transferência é realizada diretamente entre os nós. Desta forma encontrou-se uma solução para viabilizar a propagação dos aplicativos P2P: unir o poder de um banco de dados central para pesquisa de conteúdo com o poder do armazenamento distribuído.

No modelo da Internet atual, devido à forma como a Internet cresceu, o *upload* (carga de arquivos para servidores na Web) continua sendo uma árdua tarefa. Já o *download* é executado de forma relativamente mais fácil [ORAM, 2001]. A maioria dos aplicativos P2P leva isto em conta, fazendo com que o conteúdo em si (imagens, músicas, textos e arquivos em geral) permaneça nos nós e com que seja enviado para servidores centrais somente informações sobre a localização deste conteúdo. Desta forma o aplicativo P2P simplesmente faz a intermediação das solicitações entre os nós, delegando a responsabilidade da transferência a eles. Outros aplicativos e protocolos P2P, não necessitam da figura do servidor central. Todo o controle das buscas e o gerenciamento

das transferências entre nós, são responsabilidade dos próprios nós. Expõe-se com mais detalhes adiante.

4.4.2. Protocolos P2P

Redes P2P atuais fazem uso de software especializado e protocolos de comunicação que permitem que cada computador (nó) trabalhe como cliente e também como servidor. Um dos protocolos mais utilizados atualmente é o Gnutella, protocolo utilizado em diversos softwares peer-to-peer, que possibilita o compartilhamento de recursos através da Internet.

Para utilização de um protocolo P2P, um software denominado servent (assim chamado por executar tarefas de cliente e servidor :server+client) é executado em cada nó da rede. Este software permitirá que os usuários executem consultas, vejam os resultados, façam download de arquivos dos nós em melhores condições de banda e interajam com outros usuários na rede (bate-papo por exemplo). Podemos classificar os protocolos P2P da seguinte forma:

- ***Protocolos de busca centralizada:*** neste tipo de protocolo, a busca por recursos (arquivos de áudio, vídeo, software, documentos, etc.), ocorre em um servidor central que mantém uma lista dos arquivos presentes em cada nó da rede. Toda vez que um nó se conecta à rede, o servidor central atualiza em sua lista o endereço IP atual do nó;
- ***Protocolos de busca direta:*** a busca ocorre diretamente entre os nós da rede. Quando um nó se conecta ele anuncia sua presença a alguns outros nós, que fazem o mesmo até que todos os nós a rede reconheçam o novo nó conectado. Através de um protocolo, as buscas por recursos são enviadas em mensagens aos nós que fazem parte da rede.

Um protocolo P2P pode ser definido como um conjunto de regras que determinam como pode ocorrer uma conversa direta entre dois computadores. Num determinado

momento em um sistema P2P, alguns computadores estão "falando" enquanto outros estão "escutando". Para coordenar esta conversação os pacotes são marcados com descritores especiais para que desta forma, cada nó, ao recebê-los, saiba como reagir. O protocolo Gnutella, por exemplo, possui descritores como: Ping, Pong, Query, QueryHit e Push que serão vistos no capítulo 5.

4.4.2.1. Mecanismos atuais de busca em protocolos P2P

Busca em protocolos P2P descentralizados:

Antes de poder iniciar as pesquisas em uma rede P2P descentralizada, um nó deve se anunciar para os demais nós já conectados (figura 6). Um pacote anuncia a presença de um nó na rede. Quando outro nó recebe este pacote ele irá enviar uma confirmação do recebimento do anúncio. Este mesmo nó, que recebeu o anúncio do novo nó, irá repassar este anúncio para outros computadores que farão o mesmo. Cada um desses pacotes contém um identificador único (UI) do nó na rede, além de informações como endereço IP e número da porta.

Uma vez anunciada a presença de um nó na rede e outros nós ativos tenham respondido a este anúncio, o usuário pode iniciar as pesquisas. Outro tipo de pacotes, são aqueles que permitem a pesquisa em outros computadores perguntando o que estes computadores estão compartilhando. Neste ponto, o protocolo deve levar em consideração o estado da conexão dos usuários, trazendo como resultado de uma consulta os nós com melhores condições para a transferência.

A pesquisa em protocolos P2P descentralizados ocorre de maneira simples: primeiro um pacote com uma string de consulta (algum conteúdo com a palavra "P2P", por exemplo) é enviada a todos os computadores que responderam ao anúncio no momento da conexão. Os computadores que recebem este pacote de pesquisa verificam se há algum conteúdo (geralmente nomes de arquivos) com a string de busca e também repassam o pacote para todos os nós aos quais esteja conectado. Este processo continua até terminar os nós a serem pesquisados ou o pacote de pesquisa começar ficar obsoleto (time out). Este é um aspecto muito importante, deve estar bem definido um Time To Live (TTL)

para o pacote para que este não fique circulando por muito tempo. Outro ponto a observar é que um pacote de pesquisa deve passar somente uma vez por cada nó da rede para evitar os "loopings". Isto é garantido pelo identificador único do nó que realizou a busca (UI). Os nós que possuam o conteúdo pesquisado retornarão, pela mesma rota da pesquisa, um pacote indicando uma busca efetuada com sucesso. Ao receber esta confirmação e o usuário indicar a intenção de fazer o download de um determinado nó, na maioria dos protocolos P2P, uma conexão HTTP é iniciada para a transferência (método GET).

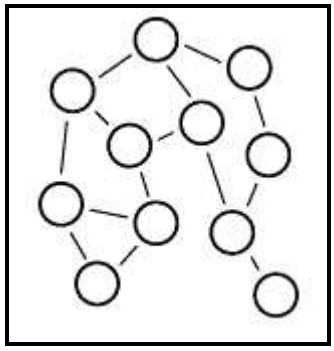

Figura 6: Uma rede P2P descentralizada

Busca em protocolos P2P centralizados:

Já a pesquisa em protocolos P2P centralizados ocorre da seguinte forma: uma string de consulta é enviada a um servidor central que mantém uma lista atualizada, em tempo real, dos nós conectados e o conteúdo em cada um dos nós. Este servidor central (figura 7) realiza uma busca em seu banco de dados, gerando um resultado com os endereços dos nós que possuam o conteúdo pesquisado. Este resultado, geralmente, é ordenado pelos nós com conexões mais rápidas. Com a lista dos endereços IP dos nós com o conteúdo pesquisado, a transferência é realizada diretamente entre os nós, sem a intermediação do servidor central. Um exemplo de uma aplicação que funciona desta forma é o Napster.

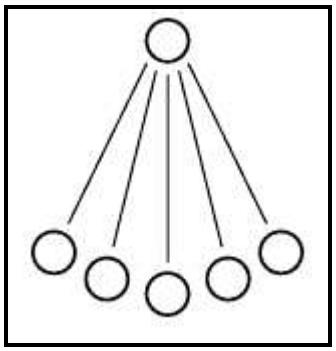

Figura 7: Uma rede P2P centralizada

Em ambos os tipos de protocolos peer-to-peer, as possibilidades nas buscas geralmente são bem limitadas. As pesquisas bem sucedidas, basicamente, são aquelas nas quais determinados nós possuam algum arquivo cujo nome contenha a string de pesquisa. Caso um usuário, em um dos nós, compartilhe um recurso com um nome que não condiz com o seu conteúdo, este conteúdo ainda será retornado para pesquisas que busquem por este nome.

4.5. Conclusão

A principal diferença entre a arquitetura cliente/servidor e a arquitetura P2P está no papel que os nós conectados à rede exercem. Na arquitetura cliente/servidor, alguns nós fazem somente solicitações e outros são responsáveis por administrar e atender tais solicitações. Em outras palavras, existe diferença na hierarquia dos nós. Já na arquitetura P2P todos os nós estão no mesmo nível, isto é, todo nó é ao mesmo tempo cliente e servidor. Cada nó pode realizar solicitações e decidir atender a solicitações que chegam. Ainda na arquitetura P2P, se um ou mais nós se desconectarem a rede continua operando, diferentemente de uma rede na arquitetura cliente/servidor onde, caso um nó servidor se desconecte, inúmeros nós clientes ficarão inoperantes. No capítulo seguinte será apresentado o protocolo P2P mais utilizado atualmente e cuja utilização em aplicações de compartilhamento de recursos vem crescendo a cada dia: o protocolo Gnutella.

Capítulo 5 - O Protocolo Gnutella

Sobre o protocolo Gnutella [BORD] diz o seguinte:

> *"A maioria dos serviços da Internet são implementados no modelo de processamento cliente / servidor. Ao contrário deste esquema centralizado, Gnutella define uma rede em nível de aplicação, com o conceito de rede peer-to-peer, onde um nó trabalha como cliente e servidor ao mesmo tempo."*

A arquitetura Gnutella consiste em um conjunto dinâmico de nós conectados entre si, através do protocolo TCP/IP, utilizando o protocolo também denominado Gnutella. Cada nó desta rede dinâmica atua como cliente (efetuando pesquisas e obtendo recursos) e também como servidor (disponibilizando recursos e respondendo à pesquisas de outros nós). Cada nó, para estar conectado à rede Gnutella, executa uma instância de uma aplicação Gnutella.

A arquitetura Gnutella nasceu em março de 2000. Justin Frankel e Tom Pepper, trabalhando em uma empresa de tecnologia chamada Gnullsoft, são os inventores do Gnutella. Um pouco antes, em 1999, ambos já haviam lançado um produto que obteve grande sucesso: o Winamp (aplicação utilizada para se ouvir arquivos de áudio como MP3). De acordo com Pepper, o Gnutella foi desenvolvido inicialmente para o compartilhamento de receitas. A empresa de Frankel e Pepper foi, então, comprada pela AOL em 1999. O Gnutella foi desenvolvido rapidamente e lançado como experiência [ORAM, 2001]. Os executivos da AOL, entretanto, não foram receptivos à idéia proposta para o compartilhamento, e encerraram o projeto.

Neste ponto, entraram em cena os desenvolvedores de sistemas de código aberto. Segundo [ORAM, 2001], um programador chamado Bryan Mayland fez um engenharia reversa da linguagem de comunicação Gnutella (protocolo Gnutella), e publicou suas descobertas em um site na Web: *gnutella.nerdherd.net*. A partir deste momento começou a se formar uma comunidade interessada na continuidade do desenvolvimento do protocolo. Canais de bate-papo foram criados, assim como inúmeros sites para que os

desenvolvedores pudessem produzir com eficiência. O protocolo Gnutella vem se tornando muito popular pelo fato de ser simples e aberto. A principal testemunha disto é a grande quantidade de softwares que "falam" o protocolo Gnutella disponíveis na Internet. Devido à sua simplicidade, até mesmo programadores iniciantes conseguem compreender e utilizar o protocolo Gnutella.

Em um sistema Gnutella não é necessário se digitar um endereço para se obter acesso a determinado recurso ou informação. A estrutura subjacente de Internet na qual "vive" o Gnutella é quase toda escondida do usuário [ORAM, 2001]. A busca pelos recursos ocorre simplesmente com a digitação de palavras-chave. Pode-se dizer que a rede formada pelo protocolo Gnutella é uma rede virtual, pois está a todo instante em modificação. Nós se conectam enquanto outros se desconectam, recursos disponíveis em um único nó , em poucos instantes, podem ser encontrados milhares de outros nós. Uma estrutura virtual sobre uma infra-estrutura física é formada.

O que torna o protocolo Gnutella diferente dos demais protocolos P2P, é que ele não depende de uma autoridade central para organizar a rede ou intermediar as transações. No Gnutella um nó, executando um programa cliente, só precisa se conectar a um outro nó arbitrário. A partir daí sua presença na rede começa a ser anunciada. Diferentemente do Napster, por exemplo, onde os nós se conectam a servidores centrais para efetuarem as pesquisas, e sem esses servidores centrais a rede se torna inoperante.

5.1. Conceitos básicos

5.1.1. Mensagens em nível de aplicativo

As redes tradicionais em nível de aplicativo são baseadas em circuitos, enquanto a Gnutella é baseada em pacotes (mensagens). Não há uma conexão ou circuito persistente entre dois nós arbitrários na rede Gnutella [ORAM, 2001]. Ambos estão na rede mas, não estão conectados um ao outro e nem indiretamente conectados de qualquer modo previsível ou estável. Em vez de forçar a maneira na qual as redes baseadas e roteamento funcionam, as mensagens são entregues por uma "brigada de apagar incêndio", daquelas onde os baldes passam de mão em mão. Na rede Gnutella, cada balde é um pacote e cada

voluntário da brigada é um nó. As mensagens são passadas de nó em nó, quer queiram ou não, dando à rede uma exclusiva e singular topologia de interconexões redundantes.

5.1.2. Difusão por TCP

Outra abordagem não convencional que a Gnutella utiliza é um modelo de difusão de comunicação por meio de TCP. O mecanismo de difusão é extremamente próximo à mecanismos presentes em situações do mundo real. Em seu livro [ORAM, 2001] compara a difusão de uma mensagem na rede Gnutella com a solicitação de informação em um ponto de ônibus: "...uma pessoa pergunta a outra em um ponto quando passa o próximo ônibus. A pessoa pode não saber mas, perguntará a alguém próximo a ela ou uma outra pessoa que escutou a pergunta pode se intrometer na conversa...".

Na rede Gnutella, cada mensagem se difunde pela rede sendo retransmitida pelos nós, passadas adiante. Cada mensagem originada recebe um identificador único (UUID) que é memorizado pelo nó enquanto a mensagem passa por ele. Caso um nó receba uma mensagem cujo identificador já esteja em sua lista, isto é sinal de um *loop* e a mensagem não é retransmitida. Isto evita o desperdício de recursos da rede, enviando consultas para nós que já a receberam. Outra idéia interessante da Gnutella é o mecanismo de expiração. Cada mensagem recebe um contador TTL (*Time to Live*) que determina sua validade. A cada nó que a mensagem passa, o TTL é decrementado. Quando o TTL chega a zero, a solicitação não é mais retransmitida.

5.1.2. Roteamento dinâmico

A difusão de mensagens, na rede Gnutella, é utilizada para as consultas, entretanto, para as respostas utiliza-se o roteamento. O mecanismo de difusão permite que as consultas na rede Gnutella alcancem um grande número de nós. Quando um determinado nó possui recurso sendo pesquisado e responde, envia uma mensagem com o mesmo identificador (UUID) da mensagem de consulta ao nó adjacente, de quem recebeu a consulta repassada. Este nó adjacente recebe a mensagem de resposta, olha o identificador e verifica qual nó enviou um pacote com aquele identificador. Então envia a mensagem ao

seu nó adjacente e, assim por diante, até que se encontre o originador da pesquisa, que recebe a resposta e a mostra ao usuário. É como a busca por um causador de boatos para processá-lo. Você pergunta a alguém: "De quem você ouviu essa informação?". E ela responde: "Do Fulano!". Então você pergunta ao Fulano de quem ele ouviu e prossegue até encontrar a origem.

Uma mensagem na rede Gnutella só é identificada pelo seu UUID. Não está associada ao endereço IP de seu emissor ou nada deste tipo. Portanto, sem as rotas baseadas em UUID, não há como uma resposta ser entregue ao nó que fez a solicitação. Este tipo de roteamento é que possibilita à rede Gnutella não possuir qualquer tipo de infra-estrutura fixa. Com o roteamento dinâmico, a infra-estrutura vem junto com os nós que ligam a rede, em tempo real. Um nó traz consigo alguma capacidade de rede que é instantaneamente integrada à trama de roteamento da rede como um todo. Quando um nó deixa a rede, ele não a deixa em ruínas, como é comum na Internet [ORAM, 2001]. Os nós conectados ao nó de saída simplesmente limpam suas memórias de UUID para "esquecê-lo".

Uma das formas com a qual os *softwares* Gnutella lidam com a mudança constante da infra-estrutura é a criação de um *backbone ad hoc*. Existe uma grande disparidade entre as velocidades de conexão dos diversos nós. Alguns estão conectados a 56 Kbps enquanto outros, por exemplo, possuem linhas T3. O objetivo é que, com o tempo, os nós conectados a velocidades maiores migrem para o centro da rede e passem a suportar o tráfego mais pesado. Enquanto isso, os nós mais lentos se deslocam para o exterior da rede. Em termos de redes, o posicionamento de um nó na rede Gnutella, não é determinado geograficamente, mas sim, em relação à topologia das conexões que os nós estabelecem. Quando um nó se conecta à rede, é como se ele caísse de pára-quedas. Cai onde cair. A rapidez com que ele consegue se tornar um membro produtivo na sociedade Gnutella, é determinada pelo seu (aplicativo Gnutella) algoritmo de análise dos nós já presentes. Em outras palavras, verifica com quais nós é interessante manter contato e quais podem ser descartados.

5.2. Comunicação Gnutella

Existem cinco tipos de pacotes no protocolo Gnutella: ***Query, QueryHit, Ping, Pong*** e ***Push***. Através destes pacotes ocorre toda a conversação entre os nós da rede. Para um melhor entendimento podemos dividir a conversação em duas partes: o anúncio de um novo nó na rede e a pesquisa por recursos.

O anúncio ocorre da seguinte forma:

1. Um novo nó (A) na rede, executando uma aplicação Gnutella, se conecta a outro nó já presente na rede e envia um pacote do tipo PING a um outro nó (B) já presente;
2. O nó B responde com um pacote do tipo PONG (neste momento o nó A já faz parte da rede) e repassa o pacote PING enviado por A para seus "vizinhos" imediatos;
3. Cada um desses nós vizinhos de B fazem a mesma coisa: respondem para A com um pacote PONG e repassam para os seus vizinhos;
4. Este processo se repete até que todos os nós da rede recebam o anúncio (pacote PING de A), ou até este pacote expirar (timeout);
5. O novo nó A está apto a efetuar pesquisas.

Já a pesquisa por recursos, após um nó se conectar à rede, envolve os seguintes passos:

1. Primeiramente, um pacote QUERY é enviado contendo uma string de pesquisa (figura 8). Esta string, por exemplo, pode conter nomes de arquivos desejados (MP3, imagens, software, etc.). Cada nó que recebe o pacote utiliza esta string para determinar se possui algum dos conteúdos sendo pesquisados e também repassa a pesquisa a outros nós (figura 9). O protocolo Gnutella não define um formato padrão ou semânticas para esta string [HEIMB, 2001]. Desta forma, a interpretação fica a cargo de cada nó que recebe a string, sendo que na prática ela é interpretada como substring que será comparada com os nomes de arquivos armazenados localmente por cada nó;

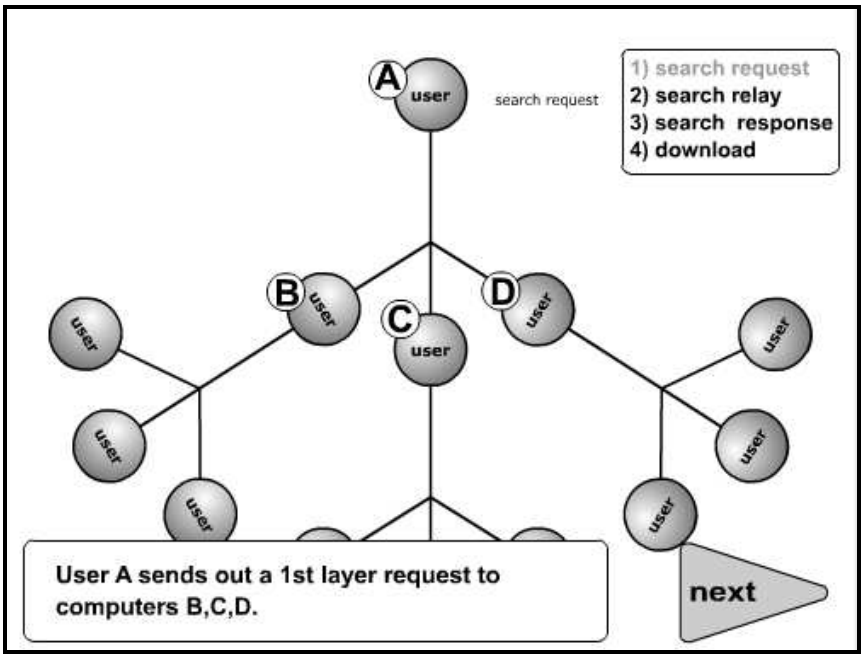

Figura 8: Nó envia pesquisa em uma rede Gnutella

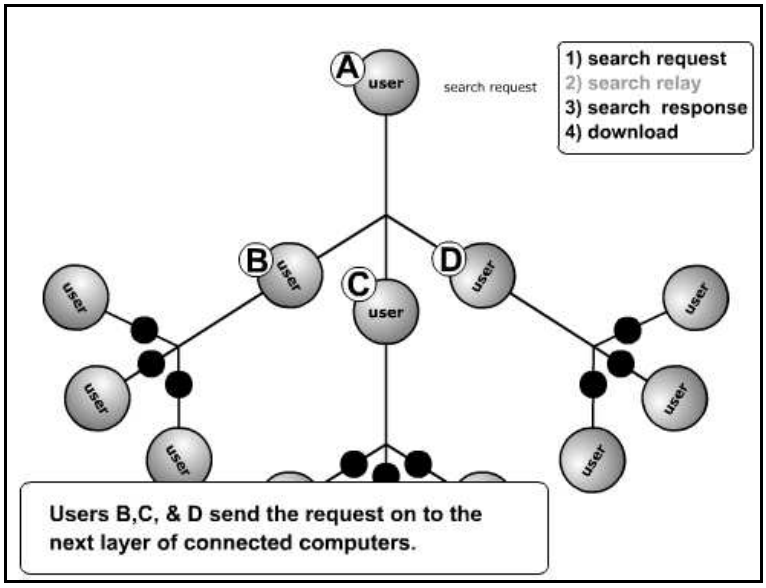

Figura 9: Nós repassam uma pesquisa em uma rede Gnutella

2. No segundo passo, um nó que possua o(s) arquivo(s) sendo pesquisado(s) (pela string do pacote QUERY), gera um mais pacotes QUERYHIT com informações sobre a obtenção deste conteúdo (figura 10). Novamente, não há um formato padrão, exceto pelo fato de ser um conjunto de strings finalizadas com nulo. Como uma regra, estas strings determinam URL's (Uniform Resource Locators) para cada arquivo. Pelo fato do pacote

QUERYHIT possuir o endereço IP do nó, ocorre de certa forma a quebra do anonimato deste nó;

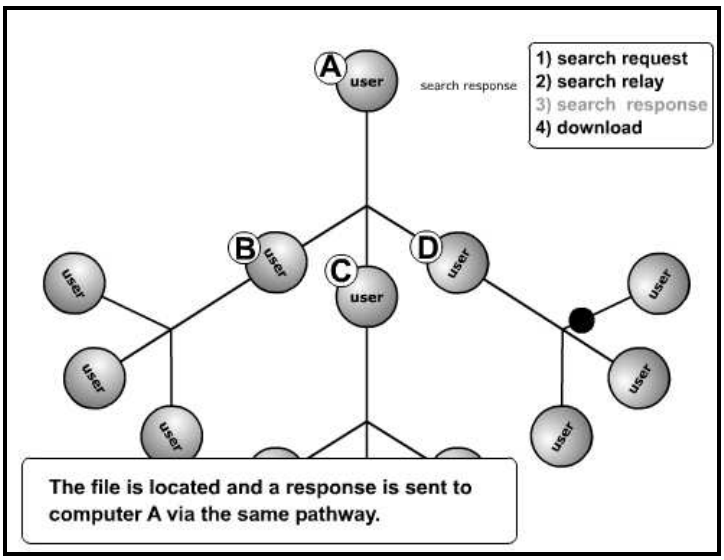

Figura 10: Nó com o arquivo solicitado responde à pesquisa

3. Por fim, o nó que efetuou a pesquisa enviando o pacote QUERY com a string, recebe como resultado os nós que possuem o conteúdo pesquisado. Fica a critério do software Gnutella a forma de exibição deste resultado ao usuário, sendo que, geralmente a listagem é colocada em ordem dos nós com melhores condições para a transferência. Então, o nó "pesquisador" se conecta diretamente (visto que o pacote QUERYHIT possui o endereço IP) a um nó que tenha respondido à pesquisa e utiliza uma versão simplificada do protocolo HTTP para obter os recursos apontados pelas URL's (figura 11);

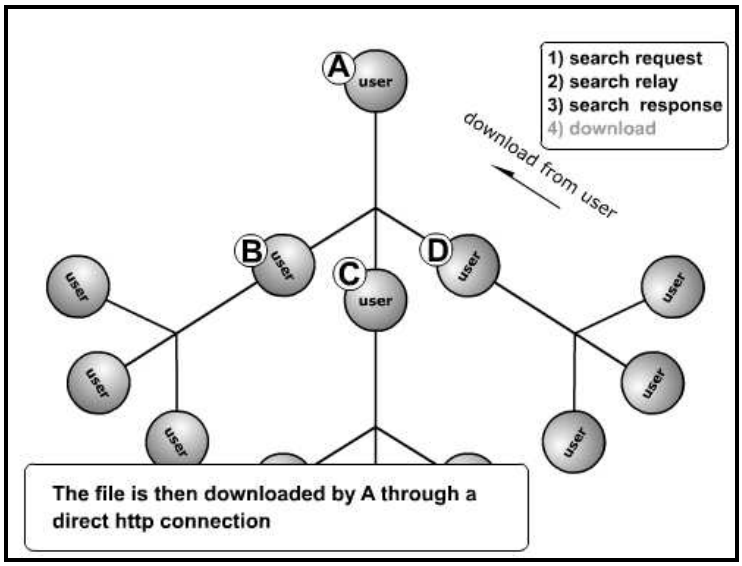

Figura 11: O nó que efetuou a pesquisa faz o download do arquivo

No protocolo Gnutella, sempre que um nó recebe uma mensagem, ele a repassa para nós que com os quais ele esteja conectado. Este fato pode gerar um grande volume de tráfego redundante. Por este motivo, o Gnutella utiliza dois mecanismos para redução de tráfego: primeiro, cada mensagem possui um contador TTL (time-to-live) que é decrementado em cada transmissão. Segundo, cada nó armazena em um cache informações sobre mensagens já recebidas e, se alguma mensagem duplicada for recebida, eles podem não repassá-la. Mesmo com estes mecanismos, o Gnutella ainda gera uma grande quantidade de mensagens.

No protocolo, pacotes de resposta (PONG e QUERYHIT) não são amplamente disseminados. Ao invés disso eles retornam por um caminho específico até o nó que gerou os pacotes PING ou QUERY originais. Isto requer que os nós intermediários, localizados entre o nó que faz a pesquisa e o que responde, armazenem informações sobre o caminho dos pacotes. Alguns softwares Gnutella não armazenam este tipo de informação, causando uma ampla disseminação de pacotes de resposta.

5.3. Especificação do protocolo Gnutella

Apesar do protocolo Gnutella suportar o tradicional modelo client/server, sua principal distinção é sua característica descentralizada peer-to-peer. Como visto anteriormente,

cada cliente é um servidor e vice versa e são chamados, por este motivo, de servents (server + clients). Devido à sua natureza distribuída, uma rede composta por estes servents que utilizam o protocolo Gnutella é altamente tolerante à falhas. Isto é, a operação da rede não será interrompida se um subconjunto destes servents estiverem offline.

A especificação do protocolo Gnutella está atualmente na versão 0.4. Um servent Gnutella se conecta à rede estabelecendo uma conexão com outro servent já presente. A obtenção do endereço de um nó já presente na rede varia de acordo com a aplicação servent. A técnica mais utilizada atualmente são os caches de nós, que nada mais são, do que listas disponíveis na Web com diversos nós atualizadas de tempos em tempos. Uma vez que o endereço de outro servent na rede é obtido, uma conexão TCP/IP é criada através da requisição:

GNUTELLA CONNECT/<protocol version string>\n\n

Onde <protocol version string> é definida como uma string contendo a versão atual do protocolo Gnutella (atualmente ‘0.4”).

O outro servent, para aceitar a conexão, precisa responder com:

GNUTELLA OK\n\n

Qualquer outra resposta indica que o servent não aceitará a conexão. Um servent pode rejeitar a conexão por uma variedade de razões: os slots de conexões podem estar esgotados ou ele pode não suportar a mesma versão do protocolo do servent que fez a requisição.

Após a conexão bem sucedida à rede, um nó pode se comunicar com outros servents enviando e recebendo pacotes do protocolo Gnutella, chamados de descritores [GNU]. Cada descritor é precedido por um cabeçalho com a estrutura de bytes abaixo:

ID do descritor	Descritor da	TTL	Hops	Comprimento

	carga			da carga
0...15 (16 bytes)	...16...(1 byte)	...17...(1 byte)	...18...(1 byte)	19...22 (4 bytes)

- **ID do descritor:** uma string de 16 bytes identificando unicamente o descritor na rede;
- **Descritor da carga:** tipo do pacote sendo transmitido (0x00 = ping, 0x01 = pong, 0x40 = push, 0x80 = query, 0x81=queryhit);
- **TTL:** do inglês Time To Live (tempo de vida). Indica o número de vezes que o descritor será repassado pelos servents Gnutella antes de ser descartado na rede.
- **Hops:** número de vezes que o descritor foi repassado. Como um descritor é passado de servent para servent, os campos TTL e Hops precisam satisfazer a seguinte condição: $TTL(0) = TTL(i) + Hops(i)$, onde $TTL(0)$ é to TTL original e $TTL(i)$ e $Hops(i)$ são os valores destes campos no nó i da rede, para $i > 0$
- **Comprimento da carga:** comprimento do descritor sendo enviado logo após o cabeçalho. O próximo descritor está localizado exatamente N bytes a partir do fim deste cabeçalho, onde N é o valor do campo *Comprimento da Carga*. Isto é, não há espaços vazios nos "streams" de dados Gnutella

Imediatamente após o cabeçalho, está o pacote ou a carga propriamente dita que consiste em um dos seguintes descritores :

5.3.1. Pacote PING (0x00)

Descritores deste tipo não possuem dados associados e possuem comprimento zero. Um PING é simplesmente representado por um cabeçalho cujo campo *payload_descriptor*, visto anteriormente, é 0x00 e cujo campo *payload_length* é 0x00000000. Um servent utiliza o PING para se anunciar na rede a outros servents. Quando um servent ativo recebe um PING, ele pode decidir responder com um descritor PONG, que contém o seu endereço e a quantidade de dados ele está compartilhando na rede. Na especificação do protocolo Gnutella não há recomendações relativas à freqüência na qual um servent deve

enviar descritores. Contudo, as aplicações servents atuais utilizam mecanismos para reduzir o tráfego na rede.

5.3.2. Pacote PONG (0x01)

Porta	Endereço IP	Número de arquivos compartilhados	Número de Kilobytes compartilhados
0...1 (2 bytes)	2...5 (4 bytes)	6...9 (4 bytes)	10...13 (4 bytes)

- **Porta:** número da porta na qual o nó pode receber solicitações de conexão;
- **Endereço IP:** o endereço IP do nó;
- **Número de arquivos compartilhados:** número de arquivos que o servent do endereço IP e porta fornecidos, está compartilhando na rede;
- **Número de kilobytes:** quantidade em (Kbytes) de dados sendo compartilhado pelo nó.

Descritores PONG são somente enviados em resposta para um descritor PING. Ele é válido o envio de mais de um descritor PONG como resposta a um PING. Existem na Web, servidores que possuem um cache de endereços de nós. Ao receber um PING, estes servidores respondem com N PONGS contendo endereços de nós.

5.3.3. Pacote QUERY (0x80)

Velocidade mínima	Critério de pesquisa
0...1 (2 bytes)	**2... (n bytes)**

- **Velocidade mínima:** a velocidade mínima (em kb/seg) dos servents que devem responder a esta mensagem. Um servent recebendo um descritor QUERY com o campo velocidade mínima com valor igual a n kb/seg, só deve responder com um descritor QUERYHIT se estiver apto a se comunicar a uma velocidade maior ou igual a n kb/seg;

- **Critério de pesquisa:** Uma *string* de pesquisa **terminada por nulo**. O comprimento máximo é limitado pelo valor do campo *Payload_Length* do cabeçalho.

5.3.4. Pacote QUERYHIT (0x81

No. De Hits	Porta	Endereço IP	Velocidade	Resultados	ID do servent
0 (1byte)	1..2 (2 bytes)	3..6 (4 bytes)	7..10(4bytes)	11..n	n + 16

- **Número de Hits:** a quantidade de hits (acertos) neste resultado;
- **Porta:** o número da porta na qual o nó que está respondendo pode aceitar conexões;
- **Endereço IP:** endereço IP do nó que está respondendo;
- **Velocidade:** a velocidade (em kb/seg) do nó que responde;
- **Resultados:** um conjunto de respostas para a consulta correspondente. O número de elementos neste conjunto é indicado pelo valor do campo *No. De Hits* e, cada elemento possui a seguinte estrutura:

Índice do arquivo	Tamanho do arquivo	Nome do arquivo
0..3 (4 bytes)	4..7 (4 bytes)	8...

  - **Índice do arquivo:** um número, determinado pelo nó que responde, que é utilizado para identificar unicamente o arquivo resultado da consulta;
  - **Tamanho do arquivo:** tamanho (em bytes) do arquivo cujo índice é o campo *índice do arquivo;*
  - **Nome do arquivo:** nome do arquivo apontado pelo índice indicado no primeiro campo deste descritor.

- **ID do servent:** uma string de 16 bytes que identifica de forma única o nó que está respondendo à consulta na rede. Geralmente o valor deste campo é

resultado de funções aplicadas ao endereço de rede do nó. Este ID é instrumento para a operação do descritor PUSH.

Descritores QUERYHIT são somente enviados em resposta a descritores de pesquisa QUERY. Um servent deve responder à uma consulta com o QUERYHIT, somente se ele contém dados que fecham com o critério de pesquisa enviado. O campo *ID do Descritor* no cabeçalho de um descritor QUERYHIT, deve conter o mesmo valor do cabeçalho associado ao descritor QUERY.

5.3.5. Pacote PUSH (0x40)

ID do Servent	Índice do arquivo	Endereço IP	Porta
0..15 (16 bytes)	16..19 (4 bytes)	20..23 (4 bytes)	25..25 (2 bytes)

- **ID do servent:** uma string de 16 bytes que identifica unicamente o nó na rede, de qual está sendo solicitado o arquivo indicado pelo índice. O servent que está solicitando o arquivo (push), seta este campo com o identificador retornado pelo pacote QUERY_HIT;
- **Índice do arquivo:** índice que identifica unicamente o arquivo sendo solicitado no nó alvo. O servent que solicita o arquivo seta este campo com um dos índices retornados no pacote QUERY_HIT;
- **Endereço IP:** endereço do nó na rede de onde o arquivo deve ser obtido;
- **Porta:** a porta na qual a comunicação (transmissão do arquivo) ocorrerá.

Pode ocorrer de um servent enviar um descritor PUSH para um servent do qual ele tenha recebido o descritor QUERY_HIT e esse servent não suportar a chegada de requisições. Isto ocorre quando o servent que enviou o descritor QUERY_HIT está atrás de um firewall. Quando um servent recebe um pacote PUSH, ele só pode tomar alguma atitude se, e somente se, o campo ID do Servent conter o valor de seu identificador.

5.4. Conclusão

A principal característica e ponto forte do protocolo Gnutella é a descentralização. Nós em uma rede Gnutella não dependem de uma autoridade central para se conectarem e efetuarem suas pesquisas. Outra característica importante no protocolo Gnutella é a propagação "viral" das mensagens, isto é, um nó ao receber uma mensagem, seja uma solicitação de conexão ou uma pesquisa, a repassa a todos os nós aos quais ele está conectado, como um vírus o faz. Por ser um protocolo simples, com basicamente 5 tipos de mensagens (ping, pong, query, query_hit e push), o Gnutella está sendo difundido muito rapidamente.

As pesquisas, entretanto, baseiam-se em simples comparações das strings de pesquisa (as palavras-chave) com os nomes dos arquivos compartilhados. Como a proposta deste trabalho é o desenvolvimento de um mecanismo mais flexível e poderoso, o capítulo 6 mostrará os conceitos básicos de metadados, bem como uma introdução à metalinguagem XML. Tal metalinguagem, será utilizada para a descrição dos recursos disponibilizados por uma aplicação P2P e o motivo da adoção da XML é que ela vem se tornando o padrão mundial para troca de informações na Web.

Capítulo 6 - Metadados

> *"Temos hoje uma tremenda confusão. Spiders (aranhas), robôs, screen-scraping (raspa-tela) e buscas ao longo de texto simples são práticas padrão que indicam uma desesperada tentativa de estabelecer diferenças arbitrárias entre a agulha e o feno." [ORAM, 2001]*

A arquitetura peer-to-peer transforma cada simples estação de trabalho em pontos (peers) que conversam uns com os outros, varrendo os depósitos de dados que estejam compartilhados. Como cada um destes ponto disponibiliza uma variedade de recursos compartilhados, o problema da localização destes recursos se mostra preocupante. Um outro agravante é que este amontoado de dados díspares são transitórios, isto é, aparecem e desaparecem à medida que os nós entram e saem da Internet. À medida que o conteúdo compartilhado pelas aplicações P2P se tornam heterogêneos, de alguma forma têm que se resolver os problemas associados à busca de informações específicas. Através de metadados é possível realizar a descrição destes recursos e possibilitar que sua localização não dependa da força bruta.

6.1. Definição de metadados

Metadados estão presentes desde que os primeiros bibliotecários fizeram uma lista do acervo de doumentos manuscritos [DUBLIN]. O termo "meta" tem origem grega e denota "com, após, próximo, que acompanha". Utilizações mais recentes no latim emprega o termo meta para denotar algo transcedental, além da natureza. O conceito de metadados precede a Internet e a Web. O interesse mundial por padrões de metadados explodiu com o aumento da publicação eletrônica e bibliotecas digitais. Qualquer um que tentou encontrar informações *online* utilizando qualquer um dos mecanismos de busca atuais, passou pela frustração de receber centenas, ou até milhares, de respostas devido à limitada habilidade para se refinar pesquisas.

Metadados podem ser definidos, de maneira simples, como dados sobre dados [RDFREC]. As propriedades de um recurso que podem ser preenchidas por valores, com

o objetivo de descrevê-lo ou caracterizá-lo, são chamadas de metadados. Por exemplo, para um livro pode-se definir como metadados o *nome, autor, assunto, editora, data de publicação*, etc. Já para um filme teríamos o *nome, gênero, produtora, atores, ano, idioma original*, etc.

Como não existe uma entidade que determine os metadados para cada tipo de recurso, atualmente eles são criados conforme a necessidade. Por exemplo, uma vídeo locadora pode definir como metadados de um filme os campos NOME e TIPO. Já outra locadora pode ter utilizado os campos TITULO e GÊNERO para representar a mesma coisa. A Web é como se fosse um grande repositório de recursos. Nelas encontramos estes recursos através de suas URLs, se as conhecermos. Entretanto, milhões de recursos como livros, documentos, vídeos, produtos, entre outros, estão disponíveis na Web. Temos a mesma situação, não existe um órgão responsável por definir os metadados para os recursos existentes. Encontrar qualquer recurso, muitas vezes, depende de técnicas de força bruta como grande parte dos mecanismos de busca (Altavista e Infoseek por exemplo). Até recentemente, praticamente, os únicos metadados consistentes em um documento HTML é o elemento <title>, que fornece não mais que uma dica, se tanto, sobre o conteúdo da página.

Como conseguir descrever recursos tão diferentes como "um artigo periódico sobre orquídeas no formato PDF", "um discurso do presidente da república no formato MP3" ou "um episódeo dos Simpsons em MPEG" ? Muitas iniciativas e esforços estão sendo realizados para identificar as operações e características comuns aos recursos e, através de metadados, permitir que sejam compartilhados e localizados de maneira eficiente na Internet. Podemos citar dentre estas iniciativas o RDF (Resource Description Framework) da W3C e o trabalho da Dublin Core Metadata Iniciative. Não se pode falar em metadados na Internet se se falar na metalinguagem XML, pela qual são criadas as descrições dos recursos.

6.2. A metalinguagem XML

A *Extensible Markup Language* (XML) é uma metalinguagem desenvolvida pelo W3C (*World Wide Web Consortium*) e nasceu com o objetivo de solucionar as limitações do HTML (*Hypertext Markup Language*). Originalmente a Web foi a solução para a

publicação de documentos científicos e, hoje, se tornou um meio para tudo. Para acomodar novas aplicações e os requerimentos dessas novas aplicações, a HTML foi estendida com o passar dos anos, novas *tags* (*tags* são as marcações da linguagem que aparecem entre sinais de menor e maior (<>)) foram inseridas e outras, modificadas. Segundo [XML, 2000], a primeira versão da HTML possuía um dúzia de tags e a mais recente está perto de cem tags.

Apesar do grande número de tags da HTML, outras são necessárias. As aplicações de comércio eletrônico precisam de tags para referências de produto, preços, nomes, endereços, etc. Outras aplicações necessitam de tags para controle de imagem e som, palavras-chave e descrição (no caso de mecanismos de consulta), enfim, cada aplicação necessita de tags específicas para a troca de dados. Neste ponto entra a XML.

A XML existe porque a HTML teve sucesso e, portanto, incorpora muitos recursos bem-sucedidos da HTML Como exemplos de utilização da XML podemos citar:

- Manutenção de um *web site*;
- Troca de informações entre organizações;
- Carga e descarga de bancos de dados;
- Publicação de conteúdo;
- Descrição de recursos.

As aplicações XML podem ser classificadas como pertencentes a um destes dois tipos:

- Aplicações de documentos que manipulam informações voltadas principalmente para o consumo humano;
- Aplicações de dados que manipulam informações voltadas principalmente para o consumo do *software.*

No primeiro tipo, aplicações de documentos, a XML se concentra na estrutura do documento e o torna independente do meio de distribuição. Por exemplo, uma receita de bolo escrita em XML poderia ser convertida para HTML e apresentada num *browser*,

convertida para PostScript e impressa ou convertida para o formato RTF e ser aberta em um editor de textos qualquer. Pelas rápidas mudanças que ocorrem na Web, muitas empresas estão optando por manter suas informações em XML e converter, quando necessário, para o formato desejado. No segundo tipo, aplicações de dados, a XML é utilizada por *softwares* diversos para o gerenciamento de dados. Por exemplo, tabelas e relacionamento de um banco de dados podem ser representados em XML. Considere a tabela abaixo:

Identificador	Nome	Preço
P1	Editor XML	R$ 499,00
P2	Editor de textos	R$ 100,00
P3	Livro sobre XML	R$ 99,00

Tabela 1: Uma lista de produtos em um banco de dados relacional

Em XML pode-se representar os dados contidos na tabela da seguinte forma:

Listagem 1: Lista de produtos em XML

```
<?xml version=”1.0”?>
<produtos>
     <produto identificador=”P1”>
          <nome>Editor XML</nome>
          <preço>499,00</preco>
     </produto>
     <produto identificador=”P2”>
          <nome>Editor de textos</nome>
          <preço>100,00</preco>
     </produto>
     <produto identificador=”P3”>
          <nome>Livro sobre XML</nome>
          <preço>99,00</preco>
     </produto>
</produtos>
```

Neste tipo de utilização, a XML é principalmente utilizada para trocar informações entre organizações. Através de suas extranets, muitas empresas trocam informações, via XML, com seus parceiros.

6.2.1. Notação

A principal característica da metalinguagem XML é que não existem tags predefinidas sendo que, o programados/autor é quem cria suas próprias tags de acordo com a necessidade. Por isso é chamada de *extensible (X)*, o programador pode estender a linguagem através da criação de novas tags. A XML refere-se à estrutura do documento, isto é, enquanto a HTML se preocupa em como será mostrada uma informação, XML se preocupa em dizer **o que é** tal informação.

O que diferencia um documento escrito em XML do mesmo documento na forma de texto puro é a marcação. As listagens abaixo exemplificam esta diferença:

Listagem 2: Um catálogo de endereços em texto puro

```
João dos Santos
Rua Santos, 34
São Paulo, SP
222-5555
Jaime da Silva
Av. Tiradentes, 987
Rio de Janeiro, RJ
333-7654
```

Listagem 3: Catálogo de endereços em XML

```
<enderecos>
     <contato>
          <nome>Joao dos Santos</nome>
          <rua>Rua Santos, 34</rua>
          <cidade>São Paulo</cidade>
          <estado>SP</estado>
          <fone>222-5555</fone>
```

```
        </contato>
        <contato>
              <nome>Jaime da Silva</nome>
              <rua>Av. Tiradentes 987</rua>
              <cidade>Rio de Janeiro</cidade>
              <estado>RJ</estado>
              <fone>333-7654</fone?
        </contato>
</enderecos>
```

Para pessoas, a listagem dois é um código mais legível. No entanto, para um *software*, a listagem 3 é a ideal. Um programa de computador precisa saber quem é quem, isto é, o que é cada dado (o que é o nome, o que é endereço, etc.). Para isso server a marcação: dividir o documento em seus constituintes para que um sistema possa processá-lo. A cada marcação damos o nome de *tag*. Na listagem 3, por exemplo, temos uma tag para nome, outra para rua, outra para cidade, estado e uma outra para o fone. Além das *tags* de grupo **contatos** e **contato**. O bloco de montagem da XML é o ***elemento*** e, cada elemento possui um nome e um conteúdo:

```
<telefone>222-0055</telefone>
```

O conteúdo de um elemento é delimitado por marcações especiais, conhecidas como *tags de início* e *tags de fim*. O mecanismos das *tags* é semelhante ao usado em HTML. A *tag* de início é o nome do elemento (telefone no exemplo acima) entre sinais de menor e maior (<>). Já a *tag* de fim inclui uma barra (/) antes do nome do elemento. As *tags* de início e fim são obrigatórias e é importante enfatizar que a XML não define elementos [XML, 2000]. Os nomes dos elementos devem seguir certas regras como: os nomes precisam começar com uma letra ou com um caracter sublinhado (_). O restante do nome é constituído de letras, dígitos, o sublinhado, ponto (.) ou hífen (-). Os espaços não são permitidos nos nomes; os nomes não podem começar com a string "xml", que é reservada para a própria especificação XML; letras maiúsculas e minúsculas são diferenciadas.

É possível anexar informações aos elementos na forma de *atributos.* Cada atributo possui um nome e um valor, sendo que os nomes seguem as mesmas regras dos nomes de elementos. Cada elemento, por sua vez, pode possuir um ou mais atributos na *tag* de início e o nome é separado do valor pelo caracter de igualdade (=). Já o valor deve aparecer entre aspas:

```
<telefone comercial="sim">333-8765</telefone>
```

Um determinado elemento pode não possuir conteúdo e, quando isto ocorre, estes elementos são denominados *elementos vazios*. Para os elementos vazios existe uma notação abreviada: as *tags* de início e fim se reúnem e a barra da *tag* de fim é incluída no final. Os dois elementos abaixo são válidos e representam a mesma coisa:

```
<e-mail link="albert@kugel.com.br"/>
<e-mail link="albert@kugel.com.br"></email>
```

O conteúdo de um elemento não está limitado a apenas texto. Os elementos podem conter outros elementos que, por sua vez, podem conter texto ou ainda outros elementos. Em outras palavras, um documento XML é uma árvore de elementos, não existindo limite para a profundidade desta árvore.

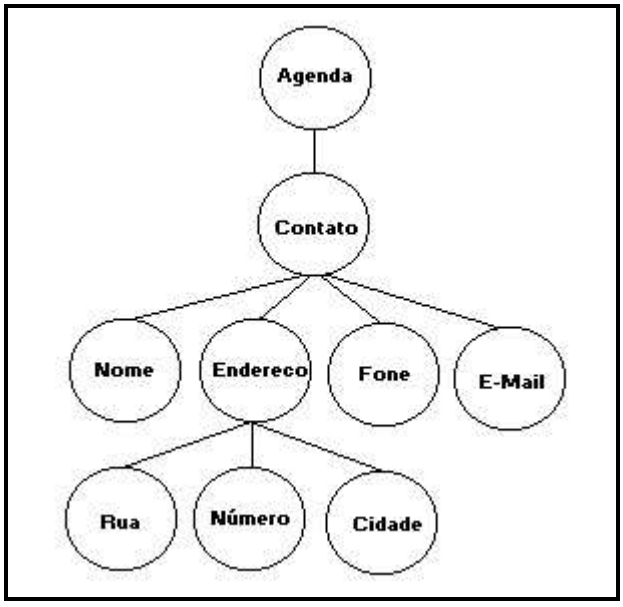

Figura 12: Árvore de elementos XML.

Um elemento que esteja embutido em outro elemento é chamado de filho e, conseqüentemente, o elemento no qual ele está contido é chamado de pai. Em um documento XML deve sempre existir um elemento raiz, isto é, todos os elementos de um documento precisam ser filhos de um único elemento. Na listagem 3 o element raiz é *endereços*.

6.2.1.1. Namespaces

Pelo fato da XML ser extensível com a possibilidade da criação de qualquer *tag*, conflitos entre estas *tags* podem ocorrer. Num mesmo documento podem existir dois elementos com mesmo nome, mas que possuam significados diferentes. Ou ainda, como não existe um órgão global responsável pela especificação de elementos XML (isto tiraria a flexibilidade da linguagem), um programador pode criar uma *tag* para armazenar determindada informação e, outro programador pode criar uma outra *tag* para guardar o mesmo tipo de informação. Por exemplo, um dono de uma vídeo locadora cria a *tag* GÊNERO para indicar se o filme é de comédia, terror, ação ou drama. Já um outro dono de locadora, cria a *tag* TIPO para o mesmo objetivo. Se as duas empresas forem trocar informações, ocorrerá um conflito ou não entendimento dos dados contidos no documento XML.

Para esta problemática foi criada uma pequena extensão para XML: *namespaces* [XML, 2000]. *Namespaces* não limita a facilidade de extensão da XML, mas insere um mecanismo para gerenciá-la. Conforme [ORAM, 2001], com os *namespaces* pode-se misturar elementos descritivos definidos por comunidades independentes, sem medo de cometer discrepâncias na nomeação, uma vez que cada segmento de dados está vinculado a um URI (*Universal Resource Indicator*) que fornece um contexto e uma definição para tal segmento.

Listagem 4: Utilização de namespaces em XML

```
<?xml version="1.0"?>
     <referencias
xmlns:dc="http://dublincore.org/documents/">
```

```
        <dc:description>Artigo sobre P2P</dc:description>
        <dc:creator>Alexandre T. Albert</dc:creator>
        <dc:date>15/04/2002</dc:date>
    </referencias>
```

Na listagem acima foram utilizados elementos definidos pela iniciativa *Dublin Core*, a qual será detalhada mais adiante neste capítulo. Ainda na listagem acima, foi utilizado o *namespace* "dc", que , por sua vez, está associado a um URI. Neste URI é que estão definidos o que significa cada elemento (*creator, date, description, etc.*). Com o recurso do *namespace*, é permitido que cada autor elabore semânticas adicionais requeridas pelos tipos particulares de recursos ou por uma área de atuação específica.

6.3. Iniciativa Dublin Core (DC)

A *Dublin Core Metadata Initiative (DCMI)* começou em 1994 durante a segunda conferência da W3C, em Chicago. Seu nome vem de *Dublin*, uma cidade do estado de *Ohio*, onde se localiza o *Online Computer Library Center (OCLC)*, um grupo que trabalha na área de bibliotecas digitais [DUBLINW]. A missão original da DCMI era a de "turbinar" a descoberta de recursos na Web, estabelecendo um conjunto mínimo de metadados. Novas necessidades e implementações são definidas em encontros anuais, sendo que em 2002 este encontro ocorrerá em Florença, Itália.

A proposta da iniciativa *Dublin Core (DC)* é disponibilizar um padrão para metadados, através de um conjunto de elementos para a descrição de uma alta gama de recursos compartilhados. O padrão DC consiste em 15 elementos cuja semântica de cada um tem sido estabelecida pelo consenso de um grupo internacional e interdisciplinar, formado por profissionais das áreas de Ciência da Computação, codificação de textos, museus, bibliotecas e outras campos da educação. Uma outra forma de olhar para a iniciativa DC é como "uma pequena linguagem para desenvolver uma classe particular de sentenças sobre recursos" [DUBLIN]. Nesta linguagem, existem duas classes de termos: elementos (nomes) e qualificadores (adjetivos).

Cada elemento é opcional, pode ser repetido e possui um conjunto limitado de qualificadores (atributos que podem ser utilizados para refinar o significados do elemento). Apesar do padrão DC favorecer objetos do tipo "documentos" (porque recursos texto são mais bem entendidos), ele pode ser aplicado a outros recursos também. Segundo[DUBLIN], a iniciativa DC possui como principais objetivos e características, os seguintes:

- **Simplicidade de criação e manutenção:** o conjunto de elementos DC tem sido mantido o mais simples possível para permitir a pessoas não especialistas, criar registros descritivos simples para recursos diversos;
- **Semânticas comumente inteligíveis:** a descoberta de informações na Internet tem como principal obstáculo as diferenças na terminologia e práticas de descrição de um campo de conhecimento para outro. O padrão DC procura ajudar um "turista digital" a encontrar seu caminho, através da utilização de um conjunto de elementos com semântica universalmente inteligível e suportada. Por exemplo, cientistas concentrados em localizar artigos de um autor em particular e estudantes de arte interessados em obras de um determinado pintor, podem concordar sobre a importância de um elemento "creator". Esta convergência em um conjunto comum de elementos aumenta a visibilidade e acessibilidade de todos os recursos;
- **Escopo internacional:** o conjunto de elemento DC foi, originalmente, desenvolvido em inglês, mas versões estão sendo desenvolvidas em outros idiomas como Norueguês, Japonês, Francês, Português, Alemão, Grego e Espanhol. Um dos objetivos da iniciativa DC é coordenar esforços para ligar estas versões em um registro distribuído, através da tecnologia RDF (*Resource Description Framework*) da W3C;
- **Extensibilidade:** ao mesmo tempo que a DC se preocupa com a necessidade de simplicidade na descrição de recursos digitais para uma precisa localização, ela também reconhece a importância de um mecanismo que possibilite a extensão do conjunto de elementos. Tal preocupação visa dar a possibilidade de, diferentes áreas de atuação, possam estender os elementos DC para suas necessidades específicas.

Os elementos DC podem ser utilizados tanto em documentos HTML como em documentos XML. Neste trabalho nos concentraremos na tecnologia XML/RDF visto que, será a tecnologia utilizada na proposta para a descrição dos recursos compartilhados em redes P2P. A tecnologia RDF, que será detalhada na próxima seção deste capítulo, permite que vários esquemas de metadados (DC é um destes esquemas) sejam lidos por humanos ou interpretados por máquinas. Ela utiliza a metalinguagem XML para expressar sua estrutura e definir a semântica. Esta abordagem descentralizada reconhece que não existe um esquema único para todas as situações [DUBLIN]. Desta forma pode-se, por exemplo, utilizar um elemento da iniciativa DC (por exemplo, "creator") e um outro elemento de outra iniciativa no mesmo descritor.

Listagem 5: Exemplo de elementos Dublin Core em XML/RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description
rdf:about="http://exemplo.com/artigos/p2p.txt">
            <dc:creator>Alexandre T. ALbert</dc:creator>
            <dc:title>Redes P2P</dc:title>
            <dc:description>Descreve o funcionamento das redes
            P2P</dc:description>
            <dc:date>2001-01-20</dc:date>
      </rdf:Description>
</rdf:RDF>
```

Como já foi dito, cada elemento DC é opcional, pode ser repetido e os elementos podem também aparecer em qualquer ordem. Os elementos DC são divididos de acordo com o tipo de informação que eles representam: elementos de conteúdo, propriedade intelectual e instanciação.

Elementos de conteúdo:

- ***Coverage:*** define o escopo do recurso sendo descrito. Geralmente indica localização espacial (nome de um lugar, coordenadas geográficas,etc.), um período temporal, ou jurisdição;

- ***Description:*** uma descrição do conteúdo do recurso. Este elemento pode conter mas não está limitado a: um resumo, tabela de conteúdos ou referência a um documento que descreva o recurso;
- ***Type:*** a natureza ou gênero do conteúdo do recurso (música, vídeo, artigo, receita, etc.). Recomenda-se, para melhor utilização, selecionar tipos já existentes em vocabulários controlados como o da Dublin Core [DCT1];
- ***Relation:*** indica uma referência a um outro recurso. Tal referência é feita através de um código que identifique unicamente um recurso;
- ***Source:*** referência a um recurso do qual o recurso sendo descrito é derivado. Esta referência também é feita através de um identificados único;
- ***Subject:*** descreve o assunto tratado no recurso descrito. Geralmente, o assunto é expressado por palavras-chave, frases ou códigos de classificação. A utilização de códigos de um vocabulário controlado também é recomendada neste elemento;
- ***Title:*** o nome dados ao recurso descrito. Geralmente, o nome pelo qual o recurso é conhecido.

Elementos de propriedade intelectual:

- ***Contributor:*** entidades que contribuíram na criação do recurso;
- ***Creator:*** principal entidade responsável pela criação do conteúdo do recurso. No caso de uma música, por exemplo, será o compositor ou cantor que a interpreta;
- ***Publisher:*** entidade responsável por tornar o recurso disponível. Esta entidade pode ser uma pessoa, organização ou serviço;
- ***Rights:*** informação sobre os direitos do recurso descrito. Geralmente, contém informações sobre propriedade intelectual , *copyright* e outros direitos. Caso este elemento não esteja presente no descritor do recurso, nenhum pressuposto será feito sobre seus direitos.

Elementos de instanciação:

- ***Date:*** uma data associada a algum evento no ciclo de vida do recurso. Normalmente, este elemento estará associado à criação ou disponibilidade do recurso;
- ***Format:*** indica a manifestação física ou digital do recurso. Geralmente, indica o formato digital no qual o recurso está disponível: MP3, BMP, MPEG, DOC, PDF, etc.
- ***Identifier:*** uma referência não ambígua ao recurso dentro de um determinado contexto. Geralmente, a referência é uma URI (*Uniform Resource Indicator*) como uma URL por exemplo (http://www.teste.com/documento.htm);
- ***Language:*** idioma original do recurso sendo descrito. Recomenda-se, para melhor utilização, os códigos definidos pela RFC 1766 [RFC1766].

Com os elementos descritos pela iniciativa Dublin Core para metadados, pode-se descrever praticamente qualquer recurso de qualquer tipo. Entretanto somente quando combinados com uma tecnologia ou linguagem de descrição que permita representar estes elementos e seus valores, é que eles se tornam realmente úteis e poderosos.

6.3.1. Qualificadores DC

Em julho de 2000, a *Dublin Core Metadata Initiative* disponibilizou sua lista de *Qualificadores Dublin Core* recomendados. Qualificadores são valores discretos definidos para os elementos DC. Por exemplo, um usuário pode preencher o elemento DC *"Language"* com o valor *"Inglês"* enquanto outro coloca *"English"* e um terceiro preenche com *"ENG"*. Todos querendo dizer a mesma coisa, que o recurso em questão tem como idioma original o inglês, mas utilizaram qualificadores diferentes. Isto se torna um problema quando tais qualificadores são processados por máquinas que os utilizarão como critério de pesquisa e comparação. Por isso, além dos elementos definidos pela DC existem também os qualificadores recomendados. Os principais definidos por [DUBLINQ] são:

Qualificadores para o elemento *description:*

- ***TableOfContents:*** uma lista de tópicos para o recurso;

- ***Abstract:*** um resumo do conteúdo do recurso;
- **Exemplo:**

```
<dc:description.tableOfContents>redes, P2P,
Gnutella</dc:description.tableOfContents>
```

Qualificadores para o elemento *title:*

- ***alternative:*** indica um título alternativo para o recurso (abreviado, traduzido, etc.);
- **Exemplo:**

```
<dc:title.alternative>Guerra nas
Estrelas</dc:title.alternative>
```

Qualificadores para o elemento *date:*

- ***created:*** indica que o valor significa a data de criação do recurso;
- ***valid:*** indica a data de validade do recurso;
- ***available:*** indica data na qual o recurso estará disponível;
- ***issued:*** data na qual o recurso foi disponibilizado;
- ***modified:*** data de modificação do recurso
- **Exemplo:**

```
<dc:date.created>10-10-2001</dc:date.created>
```

Qualificadores para o elemento *format:*

- ***extent:*** indica tamanho ou duração do recurso;
- ***medium:*** meio físico que contém o recurso;
- **Exemplo:**

```
<dc:format.extent>256K</dc:format.extent>
```

Qualificadores para o elemento *relation:*

- ***isVersionOf:*** indica que o recurso é uma variação do recurso relacionado;
- ***hasVersion:*** indica que o recurso possui uma versão que é o recurso relacionado;
- ***isReplacedBy:*** indica que o recurso pode ser substituído pelo recurso relacionado;
- ***replaces:*** pode substituir o recurso relacionado;
- ***requires:*** requer o recurso relacionado para seu correto funcionamento;
- ***isPartOf:*** o recurso descrito é parte do recurso relacionado;

- ***hasPart:*** o recurso descrito possui como uma de suas partes o recurso relacionado;
- ***references:*** o recurso descrito referencia, cita o recurso relacionado;
- ***isFormatOf:*** possui o mesmo conteúdo do recurso relacionado, só que em outro formato;
- ***hasFormat:*** o recurso descrito deu origem ao recurso relacionado em outro formato;
- **Exemplo:**

```
<dc:relation.references>Tanenbaum</dc:relation.referenc
es>
```

Além dos qualificadores acima, utilizados para refinar um elemento DC, temos os qualificadores que são os *vocabulários controlados*. O elemento DC *type*, por exemplo, possui um vocabulário controlado, também criado pela iniciativa Dublin Core, que define os valores válidos recomendados para o elemento. Os dois elementos mais importantes dentre os que possuem vocabulários controlados são *type* e *language*:

Vocabulário para o elemento *type:*

- ***Collection:*** indica uma coleção de outros recursos;
- ***Dataset:*** indica conteúdo de um elemento de banco de dados (tabela, lista, etc.);
- ***Event:*** indica um recurso não persistente. Divulgação de um acontecimento por exemplo (palestra, show, curso,etc.);
- ***Image:*** indica que o recurso é um arquivo de imagem (bmp, gif, jpg, etc.);
- ***Interactive resource:*** o recurso é interativo. Um formulário na Web, por exemplo;
- ***Service:*** indica que o recurso oferece algum tipo de serviço ao usuário;
- ***Software:*** o recurso sendo descrito é um *software*;
- ***Sound:*** o recurso sendo descrito é um recurso de áudio (mp3, wav, etc.);
- ***Text:*** o recurso é um documento texto (livros, artigos, dissertações, cartas, receitas, etc.);
- **Exemplo:**

```
<dc:type>image</dc:type>
```

Vocabulário para o elemento *language:*

- Siglas para este vocabulário estão definidas na recomendação ISSO 639-2 – *Codes for representation of Names of Languages* [ISO639]*;*
- Exemplos: POR para português, ENG para inglês, ARA para árabe, etc.
- **Exemplo:**

```
<dc:language>POR</dc:language>
```

6.4. RDF (Resource Description Framework)

Em 1995, no W3C, estava sendo desenvolvido um mecanismo para seleção de conteúdo na Internet: o PICS (*Platform for Internet Content Selection*). Era uma tecnologia para comunicação de resumos de páginas da Web entre servidores e clientes. Estes resumos continham informações sobre o conteúdo das páginas . Ao invés de criar uma série fixa de critérios, o PICS introduziu um mecanismo genérico para a criação destes resumos de conteúdo. Diferentes organizações podiam criar resumos de conteúdos em suas páginas e, os usuários podiam configurar seus *browsers* para filtrar estas páginas baseando-se nestes resumos. Um pai, por exemplo, podia configurar o *browser* para que seu filho não conseguir acessar páginas que contenham assuntos relacionados à sexo, nudez, violência e drogas. Através de uma série de reuniões com a comunidade responsável por bibliotecas digitais, as limitações no PICS foram identificadas e, novos requerimentos e funcionalidades foram acrescentadas. Como resultados da reuniões, o W3C formou um novo grupo de trabalho: o PICS-NG (*Platform for Internet Content Selection Next Generation*).

Logo após a criação do PICS-NG, ficou claro que a infra-estrutura desenvolvida nos documentos mais recentes era aplicável em diversas outras situações. Sendo assim, o W3C consolidou estas aplicações com o W3C RDF (*Resource Description Framework*). RDF é o resultado de diversas comunidades de metadados buscando, juntos, encontrar uma solução robusta e flexível para suas necessidades. RDF não foi inventada por um indivíduo ou organização [MILLER], foi sim altamente inspirado em diversos trabalhos anteriores (PICS, por exemplo).

6.4.1. O modelo de dados RDF

É muito difícil automatizar qualquer coisa na Web[RDFREC] e devido ao grande volume de informações que ela contém, não é possível gerenciar seu conteúdo manualmente. A solução proposta na tecnologia RDF é a de utilizar metadados para descrever os recursos contidos na Web. O RDF é o fundamento para o processamento de metadados. Ele provê interoperabilidade entre aplicações que trocam informações na Web e pode ser utilizado em diversas áreas: em descrição de recursos para permitir buscas mais poderosas, na catalogação de páginas da Web, bibliotecas digitais e inúmeras outras.

A sintaxe no RDF utiliza a metalinguagem XML e um dos objetivos do RDF é o de tornar possível especificar semânticas para dados de uma forma padrão. Para facilitar a definição dos metadados, RDF faz uso de um sistema de classes e propriedades como muitos modelos orientados a objetos. A esse sistema de *classes* e *propriedades* dá-se o nome de *esquemas* (schemas) ou vocabulários. Desta forma existe a possibilidade de se utilizar vocabulários já existentes ao invés de criar outro similar (conceito de reuso).

O RDF consiste basicamente em um modelo para representação de propriedades e seus valores. Propriedades no RDF, podem ser entendidas como atributos dos recursos e podem representar também relacionamentos entre recursos. O modelo de dados do RDF é uma forma neutra de representar expressões RDF. Diz-se que, duas expressões RDF são equivalentes se, e somente se, a representação de seus modelos de dados forem as mesmas. Esta definição de equivalência permite algumas variações de sintaxe na expressão, sem alterar o significado. O modelo do RDF possui três tipos de objetos:

- **Recursos:** tudo aquilo que está sendo definido por expressões RDF. Um recurso pode ser uma página da Web, uma parte de uma página da Web, coleções de páginas, um arquivo MP3 ou algo que não esteja diretamente acessível como um livro impresso. Recursos são sempre identificados de forma não ambígua por URIs;
- **Propriedades:** uma propriedade é um aspecto específico, característica, atributo ou relação utilizada para descrever um recurso. Cada propriedade tem um significado específico, define os valores válidos e os tipos de recursos que ela pode descrever;

- **Sentenças:** um recurso específico juntamente com uma propriedade e seu valor, é uma sentença RDF. Estas três partes individuais da sentença são chamadas respectivamente de *assunto*, *predicado* e *objeto*. O objeto (valor de uma propriedade por exemplo) pode ser outro recurso ou uma literal (tipo de dado primitivo por exemplo).

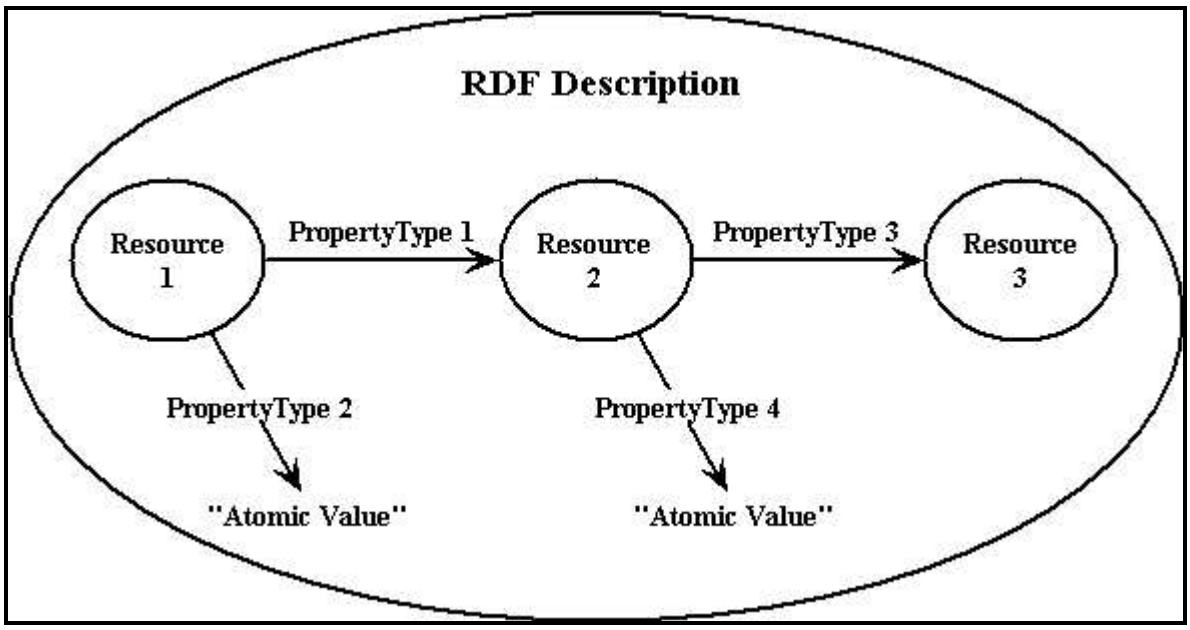

Figura 13: Representação da descrição de recursos em RDF.

Recursos são identificados por um identificador que consiste de uma URI (Unique Resource Indicator). Por exemplo, considere as sentenças: *"Paulo Coelho é o autor do livro Diário de um Mago"* e *"O autor do livro Diário de um Mago é Paulo Coelho"* . Para nós humanos ambas as sentenças possuem o mesmo significado. Para uma máquina, entretanto, são duas sentenças (ou *strings*) completamente diferentes. Utilizando uma tríade de *recursos*, *propriedades* e *valores,* RDF procura prover uma forma não ambígua para se expressar semânticas em um código que possa ser processado por uma máquina [MILLER].

6.4.2. Sintaxe RDF

O modelo de dados no RDF provê uma estrutura abstrata e conceitual para definição e utilização de metadados. Uma sintaxe concreta é também necessária para criação e troca de metadados. RDF utiliza XML para atribuição de valores à propriedades, de

propriedades à recursos e com XML permite à diferentes comunidades definir semânticas. É importante, entretanto, evitar a ambiguidade destas semânticas entre as comunidades. O elemento/propriedade "*creator*", por exemplo, pode ter diferentes significados de acordo com as diferentes necessidades das comunidades. Para evitar este problema, RDF identifica unicamente os elementos/propriedades utilizados, através da utilização do mecanismo de *namespaces* da metalinguagem XML. Por exemplo, para se indicar o "autor" de um determinado recurso, pode-se utilizar o elemento "*creator*" definido pela iniciativa Dublin Core. Para isso, identifica-se unicamente o esquema da Dublin Core acrescentando-se um *namespace* XML.

Listagem 6: Descrição de um recurso em RDF/XML

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-
ns#"xmlns:dc="http://purl.org/dc/elements/1.1/">
     <rdf:Description
about="c:\PastaCompartilhada\DM.txt">
          <dc:title>Diário de um Mago</dc:title>
          <dc:creator>Paulo Coelho</dc:creator>
     </rdf:Description>
</rdf:RDF>
```

No exemplo da listagem 6, RDF e o esquema da Dublin Core são abreviados como RDF e DC respectivamente. Declara-se sempre o esquema antes se utilizá-lo e, tal declaração é feita através da associação de um URI, indicando a localização do esquema. A linha *Description about* indica o recurso sendo descrido pelo documento RDF. O valor de uma propriedade pode ainda ser preenchido com um outro recurso:

Listagem 7: Valor de uma propriedade pode ser outro recurso

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF"
?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<?xml:namespace ns = "http://person.org/BusinessCard/" prefix =
"CARD" ?>
<rdf:RDF>
     <rdf:Description
about="c:\PastaCompartilhada\DM.txt">
```

```
            <dc:title>Diário de um Mago</dc:title>
            <dc:creator RDF:HREF = "autor 001"/>
        </rdf:Description>
        <rdf:Description ID="autor 001">
            <card:name>Paulo Coelho</card:name>
            <card:email>pcoelho@exemplo.com</card:email>
        </rdf:Description>
    </rdf:RDF>
```

RDF é uma aplicação XML que impõe limites estruturais necessários para prover métodos não ambíguos, para uma consistente representação e troca de metadados seja entre humanos, máquinas ou organizações [MILLER].

6.5. Conclusão

A World Wide Web forneceu um acesso sem precedentes à informação distribuída. Através de metadados pode-se melhorar este acesso. RDF é uma das proposta e iniciativas existentes para fazer com que a Web trabalhe com metadados. O uso efetivo de metadados entre aplicações requer convenções sobre semântica, sintaxe e estrutura. O principal objetivo da aplicação de metadados em aplicações na Web é o de transformá-la em um recurso ainda mais útil e poderoso. Como RDF ,combinado com iniciativas como a da Dublin Core, possibilita a descrição de praticamente qualquer recurso, pretende-se utilizar metadados em redes P2P. Desta forma, a localização de recursos nestas redes pode se tornar uma tarefa mais suave do que é hoje.

Capítulo 7 – Pesquisa avançada

Diversas vezes, em aplicações peer-to-peer utilizadas para o compartilhamento de recursos, faz-se *downloads* de arquivos na crença de que possuam determinado conteúdo devido aos seus nomes mas, ao abrí-los, constata-se que não. Pode ocorrer ainda, e isto é muito comum, do usuário baixar um arquivo de outro computador na rede e renomeá-lo, acidentalmente ou não. Neste caso, a maioria dos mecanismos atuais de pesquisa em redes peer-to-peer não funcionariam, trariam resultados incorretos, visto que a maioria dos *softwares* clientes utilizados, pesquisam por arquivos que tenham em seus nomes a palavra-chave sendo pesquisada. Em outros casos, deseja-se ter mais poder de pesquisa e maior flexibilidade para se ter a possibilidade de filtrar os resultados. Imagine que, por exemplo, determinado usuário deseja obter na rede através de um *software* de compartilhamento de recursos (P2P) uma receita, não uma receita comum (como um bolo de fubá, brigadeiro, etc..) que muito provavelmente teria sua descrição no nome do arquivo, mas uma receita com características bem particulares: um "*bolo de morangos silvestres com cobertura de calda de ameixas pretas"*. Nos mecanismos atuais de pesquisa e descrição, seria muito difícil pesquisar por tal recurso, visto que o nome do arquivo que contém a receita não necessariamente deve conter uma descrição detalhada.

Quando realiza-se pesquisas através dos mecanismos oferecidos pelos clientes peer-to-peer/Gnutella atuais, consegue-se limitar a busca a no máximo alguns tipos de recursos: áudio (MP3), vídeo, imagens e *software*. Estas categorias genéricas não bastam quando se deseja realizar buscas por recursos mais específicos ou de tipos pouco comuns. Por exemplo, poder obter imagens da "cidade de Joinville tiradas em 1945", arquivos MP3 de "música clássica" ou artigos sobre "a vida dos tubarões". Este é um outro grande problema a ser resolvido nas redes peer-to-peer atuais: consultas mais especializadas necessitam que os recursos compartilhados estejam descritos de forma mais detalhada. Os elementos de metadados "*artista*" e "*título*" podem funcionar muito bem, por exemplo, no âmbito do Napster, mas com certeza se provarão insuficientes para ambientes P2P mais heterogêneos compostos por um idefinido número de tipos de recursos [ORAM, 2001].

Com base nos estudos realizados nos capítulos anteriores, buscou-se uma solução para a problemática da descrição e pesquisa por recursos, que fosse aplicável a novos clientes Gnutella sem, por outro lado, afetar os clientes já existentes. Uma alteração somente em nível do protocolo Gnutella mostrou-se inviável devido ao fato de que as pesquisas em si, isto é, a comparação do critério de pesquisa com os recursos compartilhados, são feitas pelas aplicações Gnutella. Desta forma, qualquer alteração no protocolo referente à busca por recursos, irá requerer uma adaptação, mesmo que sutil, dos clientes P2P Gnutella. Da mesma forma, uma alteração somente nas aplicações P2P sem uma modificação do protocolo, não se mostrou uma solução viável. Todas as pesquisas atuais são feitas baseadas em palavras-chave carregadas em pacotes QUERY do protocolo Gnutella. Estes pacotes QUERY possuem um espaço de carga para transportar estas palavras-chave e, sem uma modificação deste pacote, não haveria como ele comportar, ao mesmo tempo, as buscas tradicionais (palavras-chave) e as avançadas (descritores XML/RDF).

A proposta desenvolvida neste capítulo consiste em uma solução híbrida, isto é, um alteração no protocolo Gnutella que não afete as aplicações já existentes combinada com a implementação de uma nova funcionalidade para a criação de metadados nos clientes Gnutella.

7.1. Proposta

Atualmente estão disponíveis para *download* dezenas de clientes Gnutella, dentre os quais podemos citar: *Morpheus*, *Bearshare*, *Limewire*, *Minitasking* e *Gnucleus*. Todos estes clientes foram desenvolvidos baseados na especificação 0.4 do protocolo e hoje, juntos, contam com uma base instalada que passa da casa dos milhões. Qualquer nova funcionalidade, principalmente quando se refere às pesquisas, deve levar em conta este grande número de usuários já existentes e possibilitar que as aplicações clientes possam ter a opção de utilizá-la ou não. Toda alteração que não cause grandes traumas ou revoluções nas aplicações legadas tem maiores chances de serem aceitas, disseminadas e, uma proposta para a melhoria das pesquisas em uma rede P2P deve levar isto em conta. Suponha que se encontre uma metodologia perfeita para pesquisas avançadas em redes P2P, a qual resolveria todos os problemas pertinentes à localização de recursos. Suponha

também, que tal metodologia requeira uma alteração drástica do protocolo Gnutella de forma que as aplicações clientes em execução tenham que ser reescritas para operarem neste novo contexto. Com certeza esta metodologia não teria aceitação por parte dos desenvolvedores e também por parte dos usuários, devido ao fato que passariam a existir redes Gnutella paralelas: uma com os clientes Gnutella "antigos" e suas buscas baseadas em palavras-chave e outra com a nova especificação do protocolo e com os novos clientes.

A abordagem deste trabalho para uma proposta de um mecanismo de pesquisa avançada e descrição de recursos através de XML/RDF, compreenderá as seguintes partes:

- **Descrição dos recursos compartilhados:** nesta parte será proposta uma metodologia para criação dos descritores para cada um dos recursos compartilhados. Estes descritores descreverão, através de metadados, as características de cada compartilhamento e , tais características serão utilizadas nas pesquisas avançadas;
- **Adendos ao protocolo Gnutella:** propõe-se adendos aos pacotes do protocolo Gnutella com o intuito de fazê-lo suportar as pesquisas avançadas, transportando critérios de busca escritos em XML/RDF. Tais adendos, entretanto, não poderão afetar o funcionamento de aplicações já existentes, isto é, criadas na especificação do protocolo que não os previa. Caso uma aplicação Gnutella não preparada para buscas avançadas receba um pacote contendo adendos, de alguma forma deverá simplesmente ignorá-los e realizar a busca da maneira tradicional;
- **Processamento da pesquisa:** Nesta parte da proposta serão expostos detalhes do processamento de pesquisas avançadas. Tal processo terá como objetivo principal ser uma metodologia que possa ser implementada em qualquer cliente Gnutella existente..

7.2. Descrição dos recursos compartilhados: metadados

Qualquer aplicação de compartilhamento de recursos/arquivos possui um diretório compartilhado. Todo arquivo colocado neste diretório se torna disponível a outros

usuários, isto é, toda pesquisa por recursos ocorrerá somente sobre os recursos existentes neste diretório. Normalmente, as aplicações de compartilhamento denominam este diretório de *My Shared Folder* (Minha Pasta de Compartilhamento), a qual será representada neste trabalho pela sigla *MSF*. Por exemplo, se uma determinada imagem for copiada para o MSF, automaticamente estará na base de pesquisa da rede P2P. O nome e localização deste repositório varia de aplicação para aplicação.

Para possibilitar pesquisas avançadas, os recursos compartilhados no MSF também devem estar descritos de forma avançada. Estas descrições devem estar escritas em um formato que possa ser processado por mecanismos de busca implementados nos clientes Gnutella. Ao mesmo tempo, esta descrição deve ser legível para qualquer pessoa, facilitando assim sua criação. Como visto no capítulo 6, a tecnologia XML/RDF atende a estes requisitos e pode ser utilizada para descrever qualquer tipo de recurso. Nesta proposta irá se utilizar o conjunto de elementos e qualificadores descritos pela *Dublin Core* [Cap. 6]. A utilização de elementos definidos por uma iniciativa bastante aceita hoje na Internet, evita que metadados sejam criados de diversas formas querendo representar a mesma informação. Sem elementos pré-definidos, para uma música por exemplo, um determinado usuário pode querer criar o elemento de metadado "artista" enquanto outro "inventa" o elemento "autor". Neste exemplo, utilizando-se o conjunto para metadados da *Dublin Core*, ambos os usuários utilizariam o elemento "creator" para representar a informação. Quando este dois usuários trocarem arquivos e seus descritores, ambos entenderão o que está sendo representado em "creator".

A idéia principal é a de disponibilizar no MSF um descritor para cada arquivo lá existente. Para efeito de demonstração desta proposta utilizaremos um estudo de caso com as seguintes características:

- Um usuário (U) possui em sua estação (E), um cliente Gnutella fictício denominado *GnuX* com os seguintes arquivos compartilhados no MSF: a música "*Imagine*" dos Beatles (TheBeatles_imagine.mp3), uma foto da *queda do muro de Berlim* (berlim.jpg) e um artigo sobre a *segunda guerra mundial* (warii.doc);

- A estação do usuário está sempre conectada à Internet e disponível à pesquisas de outros usuários;
- Os nomes dos arquivos compartilhados por (U) não dão boas pistas sobre seus conteúdos;
- Cada arquivo compartilhado necessita de um descritor XML/RDF.

Utilizando a tecnologia XML/RDF e o conjunto de elementos DC, pode-se criar os seguintes descritores para o caso descrito acima:

Listagem 8: Descritor para música "Imagine"

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="c:\GnuX\My Shared
      Folder\TheBeatles_Imagine.mp3 ">
            <dc:title>Imagine</dc:title>
            <dc:creator>The Beatles</dc:creator>
            <dc:type>sound</dc:type>
            <dc:date>10-10-1965</dc:date>
            <dc:language>ENG</dc:language>
            <dc:description>Música gravada em apresentação ao
            vivo</dc:description>
      </rdf:Description>
</rdf:RDF>
```

Listagem 9: Descritor para a foto da queda do muro de Berlim

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="c:\GnuX\My Shared
      Folder\berlim.jpg ">
            <dc:title>Queda do muro de Berlim</dc:title>
            <dc:creator>Alexandre T. Albert</dc:creator>
            <dc:type>image</dc:type>
            <dc:date>10-12-1990</dc:date>
```

```
            <dc:description>Foto tirada durante férias na
            Alemanha</dc:description>
      </rdf:Description>
</rdf:RDF>
```

Listagem 10: Descritor para o artigo sobre a segunda guerra

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="c:\GnuX\My Shared Folder\warii.doc
      ">
            <dc:title>Segunda Guerra Mundial</dc:title>
            <dc:creator>John Smith</dc:creator>
            <dc:type>text</dc:type>
            <dc:date>10/10/1995</dc:date>
            <dc:description>Artigo que descreve as principais
            batalhas da II guerra e suas
            motivações</dc:description>
      </rdf:Description>
</rdf:RDF>
```

Como pode-se constatar nos descritores XML/RDF acima, vê-se que são legíveis tanto para o usuário que o cria como para o *software* que irá processá-lo. Legível para o usuário pelo fato de ser escrito em formato texto puro, o que possibilita sua abertura em qualquer editor de textos. Legível para um *software* por serem escritos em XML/RDF, isto é, serem descritores bem tipados e com marcações bem definidas. Na proposta deste trabalho cada descritor será definido em um arquivo de extensão RDF e com o nome igual ao recurso ao qual ele se refere. Estes arquivos serão criados no MSF. No estudo de caso os descritores terão os seguintes nomes:

- *TheBeatles_imagine.mp3* => ***TheBeatles_imagine.rdf****;*
- *berlim.jpg* => ***berlim.rdf****;*
- *warii.doc* => ***warii.rdf****.*

7.2.1. Criação dos descritores

Em dois momentos distintos ocorrerá a criação de um descritor para um recurso: quando um arquivo for copiado/movido para o MSF ou quando for efetuado um *download* de um recurso compartilhado por outro nó. No primeiro caso o descritor deverá ser criado e preenchido pelo usuário que está disponibilizando o arquivo para compartilhamento. Já no segundo caso, quando se efetua um *download*, faz-se a criação do descritor para o recurso no MSF do nó destino. Tal criação utilizará os metadados recebidos na resposta à pesquisa (pacote QUERY_HIT), que será detalhada adiante neste capítulo. Caso o recurso sendo baixado não possua descritor no nó de origem, sua criação ficará a critério do usuário.

A criação dos descritores pode ser feita através de qualquer editor de textos, entretanto, esta prática oferece uma grande margem de erro, seja de sintaxe ou semântica. A prática recomendada nesta proposta é a utilização de uma nova aplicação ou funcionalidade, a ser implementada nos clientes Gnutella, que permita, através de uma interface gráfica, a criação dos descritores RDF para cada arquivo existente no MSF. Esta aplicação ou funcionalidade fará uma verificação ,sempre que o *software* cliente for aberto, da existência dos descritores. Caso seja encontrado um recurso no MSF que não possua um descritor, o cliente Gnutella abrirá uma tela solicitando o preenchimento das informações para o arquivo em questão. Então, o usuário preenche os campos (elementos DC) que julgar pertinentes ao recurso.

Visto que a criação dos descritores dependerá da boa vontade dos usuário de aplicações Gnutella, como incentivo à tarefa da criação dos descritores, pode-se atrelar a possibilidade de se efetuar pesquisas avançadas à verificação da existência e criação dos descritores. Em outras palavras, um usuário somente poderá efetuar pesquisas através de metadados se, em sua estação, o cliente Gnutella estiver com a opção de verificação e criação de descritores ligada. Como a possibilidade de um recurso sem descritor em um determinado nó, já ter sido descrito por outro usuário em outro nó, a aplicação Gnutella pode, de tempos em tempos, efetuar uma busca automática de descritores. Esta busca automática será efetuada enviando-se um pacote QUERY com o nome do recurso em questão. As repostas recebidas são filtradas para que somente arquivos com extensão

RDF sejam mostrados e, o usuário possa selecionar o descritor correspondente ao recurso.

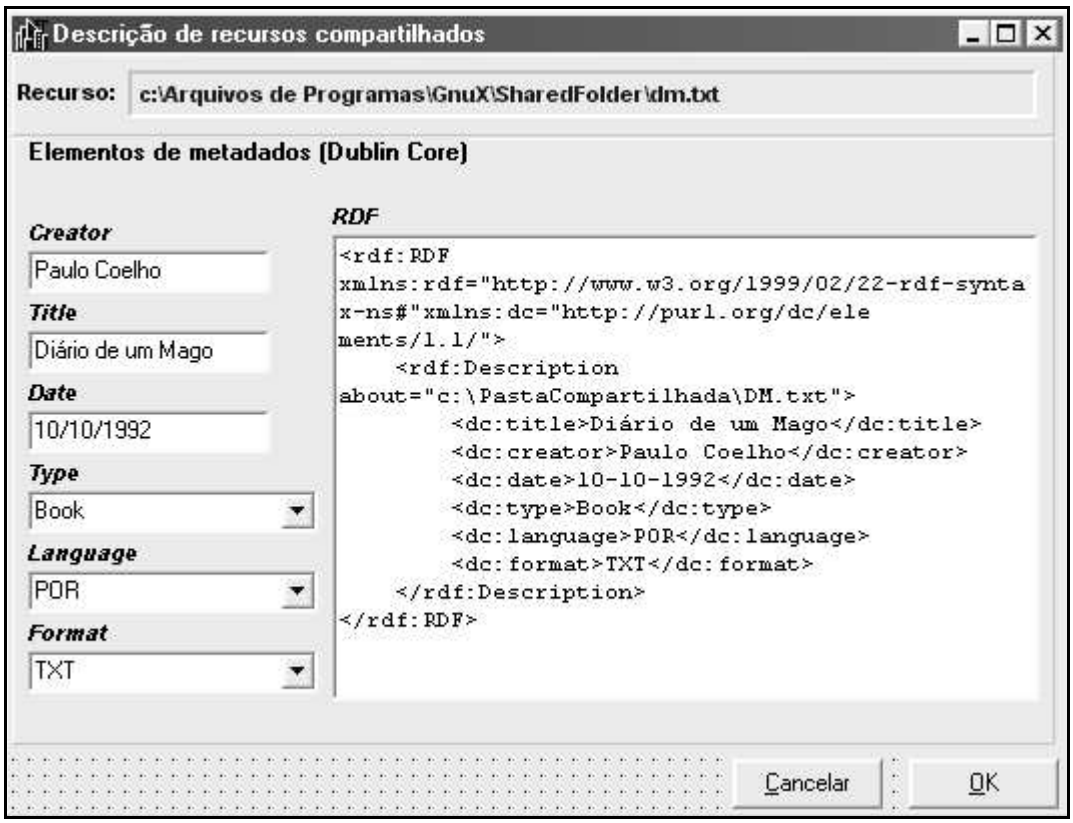

Figura 14 : Exemplo de uma tela de verificação e criação de metadados

7.3. Adendos ao protocolo Gnutella

Para que clientes que já utilizam a especificação 0.4 do protocolo Gnutella, não deixem de funcionar devido a uma alteração do mesmo e, mais importante, que aplicações novas possam "conversar" com clientes "antigos" (clientes existentes antes da alteração), ela deve se dar na forma de *adendos* aos pacotes. Em outras palavras, as aplicações atuais pressupõe que os pacotes serão recebidos com a estrutura definida na especificação do protocolo e ignorarão qualquer informação adicional. Entretanto, nada impede que sejam enviadas informações adicionais e que novas aplicações possam utilizá-las.

Alguns dos pacotes existentes no protocolo Gnutella (QUERY, QUERY_HIT,...) carregam informações que não possuem um tamanho pré-definido, como o *critério de*

pesquisa por exemplo. Um critério de pesquisa pode ser uma palavra-chave com três caracteres ou uma frase com dez palavras. Então, as aplicações lêem estes pacotes até encontrar um caracter nulo (NULL). Quando encontram este caracter, consideram que é o fim do dado sendo transportado pelo pacote. A proposta deste trabalho explora este fato para criar adendos aos pacotes Gnutella que permitam às aplicações enviarem informações de pesquisa e resultados avançados.

7.3.1. Adendo ao pacote de pesquisa: QUERY

Dos pacotes existentes no protocolo Gnutella, os dois relacionados à localização de recurso são: QUERY e QUERY_HIT. Enquanto o pacote QUERY é difundido, isto é, repassado entre os nós da rede, o pacote QUERY_HIT (resposta) é roteado, em outras palavras, segue um rota até o nó que efetuou a pesquisa.

Na especificação atual do protocolo Gnutella, a 0.4, o pacote QUERY possui o seguinte formato:

Velocidade mínima	Critério de pesquisa
0...1 (2 bytes)	**2... (n bytes – até NULL)**

O campo velocidade mínima, como visto no capítulo 3, transporta a velocidade (taxa de transferência) mínima dos nós que devem responder à pesquisa. Caso um determinado nó possua um arquivo que feche com o critério de pesquisa utilizado, mas possua uma taxa de transferência menor que o campo *velocidade mínima*, ele não responderá à pesquisa com um pacote QUERY_HIT. Já o campo *critério de pesquisa* carrega a palavra chave digitada pelo usuário para a busca. Este campo consiste de uma *string* de caracteres cujo o final é indicado por um caracter nulo [Cap. 5]. É aí que está o ponto mais importante para a alteração, através da criação de um **adendo XML** para o pacote QUERY.

Como o *critério de pesquisa* termina com um **caracter nulo**, tudo o que vier após este nulo será ignorado pelos clientes Gnutella atuais. O adendo XML proposto, será criado após este caracter nulo. Desta forma, aplicações já existentes simplesmente ignorarão o adendo e farão a pesquisa baseadas no tradicional *critério de pesquisa* passado. Já novas

aplicações farão uma verificação após este caracter nulo e, se encontrarem o adendo XML, o utilizarão para uma pesquisa avançada nos descritores XML/RDF presentes no MSF.

Por exemplo, em uma pesquisa tradicional pela música "*Imagine*" dos Beatles o pacote QUERY gerado por um nó Gnutella teria a seguinte forma:

Figura 15: Pacote QUERY tradicional.

Este pacote será recebido pelos diversos nós da rede que utilizarão o critério "*imagine*" contido no pacote para fazer a comparação com os arquivos do MSF. Um nó com um arquivo chamado *PauloRicardo_imagine.mp3* responderia à consulta, mas suponha-se que o usuário está buscando pela música "Imagine" dos *Beatles*. Os clientes Gnutella atuais não entendem, e nem verificam, após o caracter nulo no critério de pesquisa. Entretanto, novas versões destes clientes podem ser preparadas para analisar o conteúdo após o caracter nulo. A mesma consulta, só que efetuada com recursos avançados de pesquisa gerará o seguinte pacote:

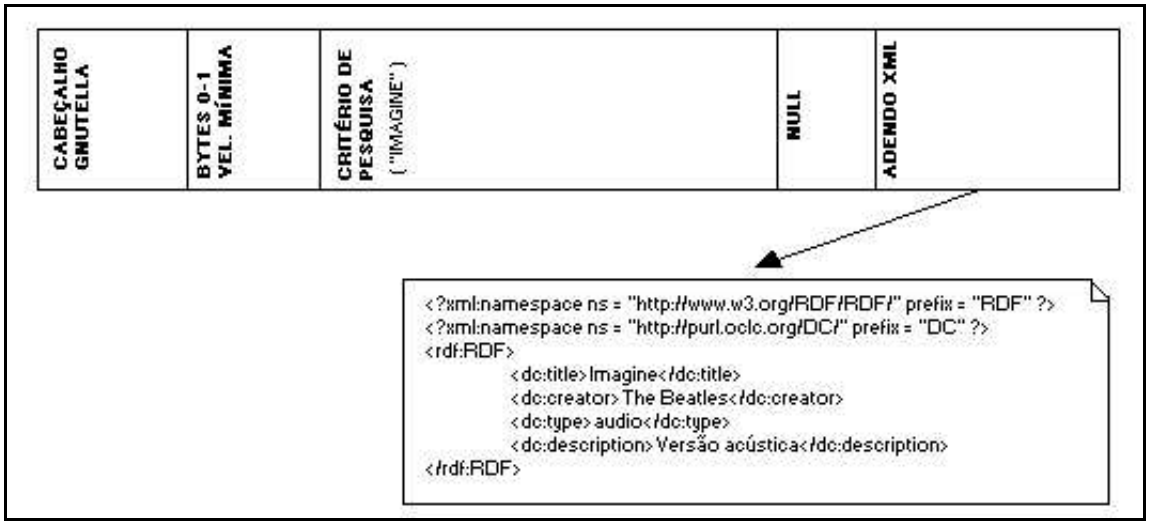

Figura 16: Pacote QUERY com adendo RDF/XML.

Os nós da rede Gnutella que receberem o pacote acima poderão utilizar tanto o critério de pesquisa tradicional (palavra "ímagine" no exemplo) como o adendo XML/RDF para efetuar uma busca avançada. Suponha-se que um dos nós receba o pacote da figura 15 e possua no MSF os arquivos "*TheBeatles_imagine.mp3", "FrankSinatra_imagine.mp3*" e os descritores "*TheBeatles_imagine.RDF" e "FrankSinatra_imagine.RDF*" das listagens 11 e 12. Se tal nó realizar a pesquisa em sua base compartilhada (MSF) da forma tradicional, ou seja, comparando o critério de busca recebido com os nomes dos arquivos, ambos os arquivos seriam retornados para o nó que efetuou a pesquisa. De forma diferente, se o nó optar por realizar a busca baseada no adendo XML recebido, somente o arquivo *TheBeatles_imagine.mp3* seria retornado como resultado válido. Isto porque os elementos XML do adendo seriam comparados um a um com os elementos do descritor do recurso.

Listagem 11: Descritor *TheBeatles_imagine.RDF*

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="c:\GnuX\My Shared
      Folder\TheBeatles_Imagine.mp3 ">
            <dc:title>Imagine</dc:title>
            <dc:creator>The Beatles</dc:creator>
            <dc:type>audio</dc:type>
            <dc:language>ENG</dc:language>
            <dc:description>Última interpretação antes da morte
            de John Lennon</dc:description>
      </rdf:Description>
</rdf:RDF>
```

Listagem 12: Descritor *FrankSinatra_imagine.RDF*

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
```

```
      <rdf:Description about="c:\GnuX\My Shared
      Folder\FrankSinatra_Imagine.mp3 ">
            <dc:title>Imagine</dc:title>
            <dc:creator>Frank Sinatra</dc:creator>
            <dc:type>audio</dc:type>
            <dc:language>ENG</dc:language>
            <dc:description>Gravada no show em
      N.Y.</dc:description>
      </rdf:Description>
</rdf:RDF>
```

Desta forma o pacote QUERY proposto para pesquisas avançadas fica com a seguinte estrutura. O campo *velocidade mínima* permanece inalterado, assim como o campo *critério de pesquisa* que vai até o primeiro caracter NULO. Foi acrescentado o campo *adendo* que vai do primeiro caracter NULO até o próximo NULO :

Velocidade mínima	Critério de pesquisa	Adendo XML/RDF
0...1 (2 bytes)	**2...n (até NULL)**	**NULL ... m (até NULL)**

Um importante aspecto, que não pode ser esquecido, é o fato de que se um nó envia um pacote QUERY com o adendo XML, é de se esperar que ele deseje somente receber respostas de nós que entendam tal adendo ou, pelo menos, dar prioridade a elas. Em outras palavras, nós que não percebam a existência do adendo de pesquisa não poderiam responder a estes pacotes para evitar um amontoado de repostas desnecessárias (muito comum hoje). Uma saída é: sempre que uma pesquisa avançada for efetuada, ou seja, sempre que existir o adendo XML, o campo *critério de pesquisa* deve conter somente NULO (caracter NULL). Assim, nós que não estejam preparados para o adendo irão encontrar NULL no critério e não efetuarão busca nenhuma, conseqüentemente não gerarão pacotes QUERY_HIT de respostas. Já os nós com clientes que verifiquem a exsitência do adendo, esses sim, efetuarão suas pesquisas e retornarão respostas QUERY_HIT. Isto evita também um tráfego desnecessário na rede Gnutella. Uma prática recomendada neste trabalho, é que tal saída seja uma opção para o usuário do cliente Gnutella. Isto é, o usuário poderá configurar sua aplicação Gnutella para que somente nós com recurso de pesquisa avançada ou todos nós devam responder suas pesquisas.

7.3.2. Adendo ao pacote de resposta: QUERY_HIT

Pesquisas que utilizam metadados devem receber respostas que também contenham um adendo com metadados, descrevendo o(s) recurso(s) encontrado(s). No protocolo Gnutella, o pacote responsável por tranportar as respostas às consultas é o pacote QUERY_HIT [Cap. 5]. Na especificação 0.4 do protocolo o pacote QUERY_HIT possui a seguinte estrutura:

No. De Hits	Porta	Endereço IP	Velocidade	Resultados	ID do servent
0 (1byte)	**1..2 (2 bytes)**	**3..6 (4 bytes)**	**7..10(4bytes)**	**11..n**	**n + 16**

O campo *no. de hits* indica a quantidade de respostas (arquivos encontrados) que o pacote carrega. Os campos *porta* e *endereço IP* indicam o número da porta na qual o nó que enviou a resposta está operando e o seu endereço IP respectivamente. A velocidade, ou seja, a taxa de transferência do nó que enviou o QUERY_HIT, é indicada pelo campo

velocidade. Os resultados vêm armazenados no campo *resultado*, que guarda as informações sobre os arquivos resultantes. O tamanho do vetor de resultados é indicado pelo campo *no. de hits*. Cada um destes resultados possui a estrutura:

Índice do arquivo	Tamanho do arquivo	Nome do arquivo
0..3 (4 bytes)	**4..7 (4 bytes)**	**8...(terminado por null)**

Como no pacote QUERY, onde o adendo foi colocado após o caracter nulo que indica o fim do pacote, coloca-se no QUERY_HIT o adendo XML/RDF após o *ID do nó*. Entretanto, um pacote QUERY_HIT pode conter mais de um resultado, isto é, mais de um arquivo que satisfizeram a pesquisa foram encontrados em um determinado nó. Desta forma, o adendo deve trazer um descritor para cada arquivo resultado, ou seja, um vetor de descritores.

Para exemplificar, suponha-se a seguinte situação:
1- um nó **(A)** efetuou uma busca pela música de título "*imagine"* em formato MP3:

Listagem 13: Adendo XML/RDF enviado na pesquisa efetuada por **(A)**.

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <dc:title>Imagine</dc:title>
      <dc:format>MP3</dc:format>
</rdf:RDF>
```

2- então, um pacote QUERY com o adendo XML descrito na listagem 13, é difundido na rede. Um segundo nó **(B)** recebe o pacote, verifica o adendo, faz a pesquisa no MSF e retorna um pacote QUERY_HIT com os adendos da listagem 14, indicando quais arquivos possui e quais suas características.

Listagem 14: Adendos XML/RDF enviados na resposta de **(B)** à pesquisa de **(A)**.

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="TB_Imagine.mp3 ">
      <dc:title>Imagine</dc:title>
      <dc:format>MP3</dc:format>
      <dc:creator>The Beatles</dc:creator>
</rdf:RDF>

<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix = "RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<rdf:RDF>
      <rdf:Description about="F_Imagine.mp3 ">
      <dc:title>Imagine</dc:title>
      <dc:format>MP3</dc:format>
      <dc:creator>FrankSinatra</dc:creator>
      <dc:date>10-10-1970</dc:date>
</rdf:RDF>
```

3- o cliente Gnutella em **(A)**, ao receber o pacote QUERY_HIT, verificará os adendos, os decodificará e utilizará as informações neles contidas para permitir ao usuário decidir qual arquivo baixar.

A utilização dos descritores também no pacote QUERY_HIT, possibilita um auxílio para a tomada de decisão, evitando que o usuário baixe arquivos desnecessário ou não desejados. Repare no exemplo anterior, que o nome dos arquivos resultado não dizem muito sobre seu conteúdo. Desta forma, nas pesquisas tradicionais dos atuais clientes Gnutella, seria mais complicada a decisão do *download*.

O pacote QUERY_HIT proposto para pesquisas avançadas fica com a seguinte estrutura:

No. de hits	Porta	End. IP	Velocidade	Resultados	Id. Do nó	Descritores
0 (1byte)	**1..2 (2 bytes)**	**3..6(4 bytes)**	**7..10(4bytes)**	**11..n**	**n + 16**	**n+16..m**

A diferença para o pacote QUERY_HIT tradicional está no campo *descritores,* que possibilita ao pacote transportar as características de cada resultado obtido. Assim como o campo *resultados* contém os resultados na quantidade indicada pelo campo *no. de hits*, o campo *descritores* irá conter ***N*** descritores, onde ***N*** também é o valor indicado por *no. de hits.* Cada descritor neste vetor será terminado por um caracter nulo. Assim o cliente Gnutella, a partir da informação contida em *no. de hits*, pode extrair os resultados, seus descritores e apresentar ao usuário as características de cada um.

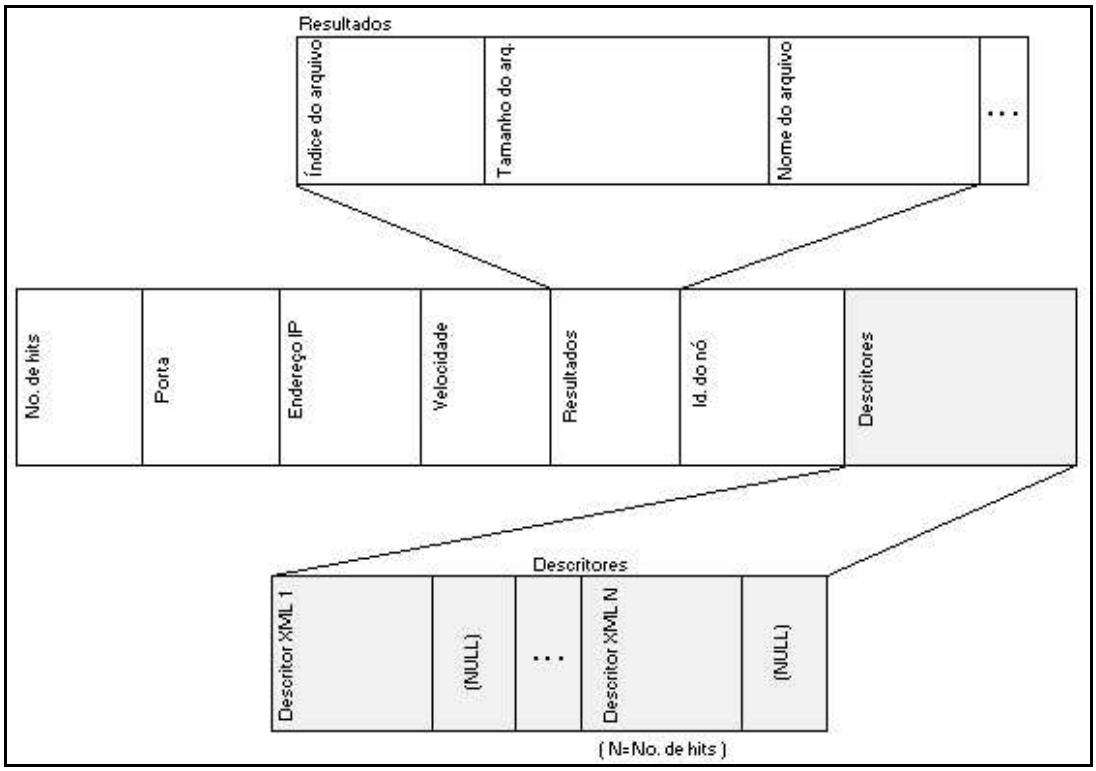

Figura 17: Adendo de descritores no pacote QUERY_HIT.

Este adendo de descritores só será enviado em resposta à pesquisas avançadas, isto é, pesquisas que tenham utilizado o adendo no pacote QUERY. Toda pesquisa tradicional, ou seja, que contenha somente uma palavra-chave, será respondida com o pacote QUERY_HIT tradicional (sem o adendo). Desta forma, a interoperabilidade entre clientes já preparados para consultas XML/RDF e clientes tradicionais é mantida.

7.4. Processamento da pesquisa

A maioria dos clientes *Gnutella* atuais não suportam metadados para a descrição e busca por recursos. Para que os nós que possuam estes clientes *Gnutella* instalados possam fazer uso de pesquisas avançadas e metadados, os desenvolvedores destas aplicações terão que disponibilizar novas versões. Já os novos clientes que surgirão, já poderão implementar a funcionalidade de pesquisas avançadas e descrição de recursos. Neste trabalho propõe-se um mecanismo de busca avançada e descrição de recursos, baseado nos adendos e descritores *XML/RDF* já apresentados neste capítulo.

O mecanismo consiste em se analisar todos os pacotes *QUERY* que chegam à estação, verificando a existência de adendos de pesquisa *XML*. O cliente Gnutella deverá saber como agir quando receber pacotes com e sem adendos XML. Caso estes adendos estejam presentes em um pacote de pesquisa, as informações nele contidas deverão ser utilizadas para efetuar a busca avançada contra os descritores XML/RDF contidos no MSF. Entretanto, se após a verificação da existência destes adendos, for constatado que não há adendo algum presente, a pesquisa tradicional deverá ser efetuada, ou seja, a palavra-chave contida no campo *critério de pesquisa* do pacote será comparada com os nomes dos arquivos. Este fato, o de um nó com cliente Gnutella já preparado para buscas avançadas receber pacotes sem adendo, será bastante comum por um tempo, enquanto existirem nós com clientes Gnutella tradicionais, sem suporte a metadados.

Para o entendimento do mecanismo proposto para buscas avançadas, considere a seguinte notação:

- ***(P),*** um pacote de pesquisa tradicional (sem adendos XML/RDF);
- ***(P'),*** pacote de pesquisa avançada (com adendos XML/RDF);
- ***(A),*** nó, com suporte à pesquisas avançadas, que recebe uma pesquisa por um recurso;
- ***(B),*** nó que envia uma pesquisa por um recurso;
- ***(G),*** cliente Gnutella utilizado por *B;*
- ***(S),*** processo responsável por verificar a presença de adendos XML/RDF nos pacotes QUERY*;*
- ***(R),*** reposta à um pacote de pesquisa.

Nas figuras 18 e 19 são apresentadas duas situações:

Situação 1:

O nó ***(A)*** recebe uma pesquisa tradicional ***(P)***. Neste caso o processo ***(S)*** intercepta o pacote QUERY mas, como constata a inexistência de adendos, ignora-o. A pesquisa tradicional será efetuada pelo *software* ***(G)*** no MSF e um pacote QUERY_HIT (resposta (R)) sem adendo XML será enviado para ***(B)*** :

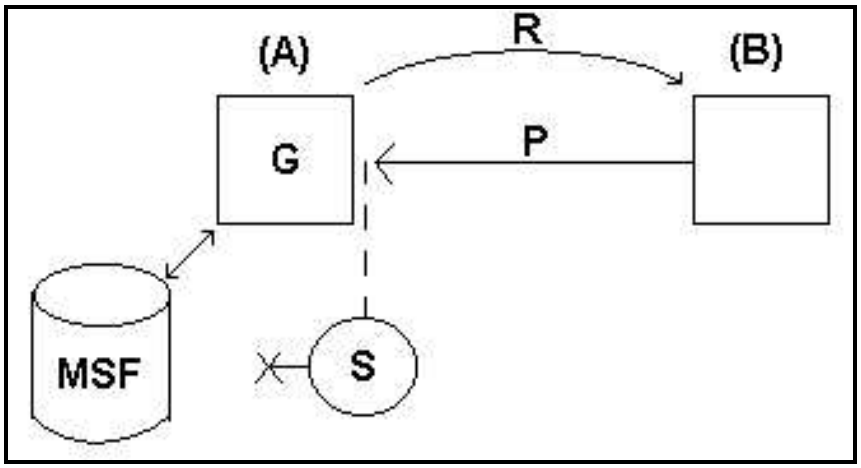

Figura 18: Nó recebe uma pesquisa tradicional.

Situação 2:

O nó ***(A)*** recebe uma pesquisa avançada ***(P')***. Neste caso o processo ***(S)*** intercepta o pacote QUERY , constata a existência de adendos, e efetua a busca avançada no *MSF*. Já a busca tradicional não é realizada pois o campo critério de pesquisa, que transporta a palavra-chave, está com valor NULO. Então, o nó *(A)* envia a *(B)* pacote QUERY_HIT com o resultado da pesquisa. Este pacote QUERY_HIT conterá adendo pois é uma resposta a uma pesquisa avançada:

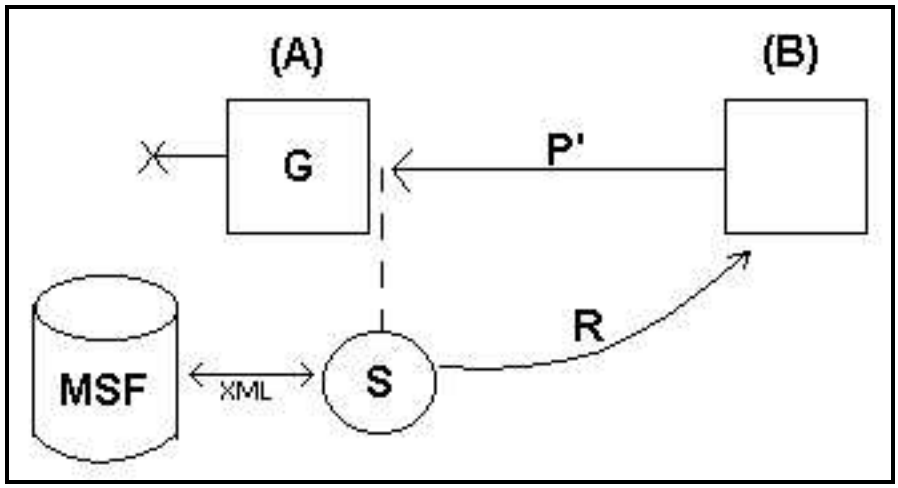

Figura 19: Nó recebe uma pesquisa avançada (XML/RDF).

Em ambas as situações expostas, o processo *(S)* é o processo responsável por efetuar a verificação da existência de adendos nos pacotes QUERY e efetuar a busca avançada nos descritores RDF/XML contidos no *MSF*. Mas como se dá essa busca avançada? Quando um pacote QUERY chega à estação contendo um adendo *XML/RDF*, o processo *(S)* deverá obter cada um dos elementos e seus valores, codificados em *XML*. Com as duplas (elemento, valor) em mãos, o processo *(S)* abre cada descritor no *MSF* da estação e compara estes elementos e valores. Somente o recurso que possui descritor que contenha elementos e valores que batam com os que estão sendo pesquisados, será considerado um recurso resultado. Por exemplo, considere o pacote QUERY da figura 20:

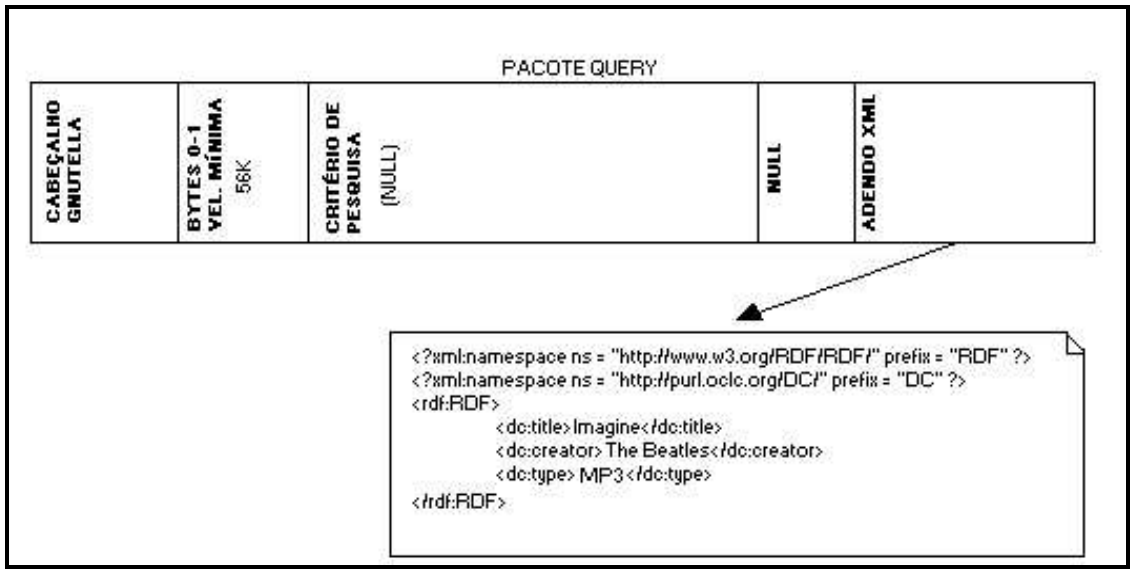

Figura 20: Exemplo de um pacote de pesquisa (QUERY).

1. o processo *(S)*, após analisar o adendo, terá um vetor com as seguintes duplas: **title=”imagine”**, **format=”mp3”** e **creator=”the beatles”**;
2. no MSF, então, o processo (S) busca pelos descritores que contenham os elementos **title**, **format** e **creator**;
3. com os descritores candidatos a serem resultados em mãos, o processo ***(S)*** extrai os valores dos elementos **title**, **format** e **creator** de cada um deles e compara com os valores do adendo do pacote QUERY (‘imagine”, ‘mp3” e ‘the beatles”);
4. feita a comparação, se algum descritor possuir os valores desejados, um pacote QUERY_HIT é enviado como resposta contendo, entre as informações do pacote QUERY_HIT, um adendo contendo os metadados do recurso.

Quanto mais elementos um adendo de pesquisa possuir, mais refinada é a pesquisa e menor é o número de resultados. Nada impede que o usuário envie uma pesquisa com somente um elemento (**<dc:creator>TheBeatles</dc:creator> ,** por exemplo). Quando uma estação receber a resposta de sua pesquisa avançada e decidir fazer o *download* para o seu MSF, de um dos arquivos resultado, um descritor (XML/RDF) também deverá ser criado no MSF com as informações sobre o arquivo sendo baixado. Desta forma faze-se com que o descritor de um recurso o acompanhe em sua distribuição pela rede Gnutella. Entretanto, nada impede que o usuário edite tal descritor e o altere ou o enriqueça com informações que julgar importantes.

7.5. Discussão: desenvolvedores, usuários e aplicações

A difusão de um mecanismo para pesquisas avançadas por recursos nas redes Gnutella, para se tornar uma realidade, deverá contar com o apoio e a conscientização dos desenvolvedores e usuários das aplicações P2P. Iniciativas na área de metadados e de pesquisas avançadas, são os primeiros passos para um consenso entre os que desenvolvem aplicações Gnutella. Principalmente no tocante à criação dos elementos de metadados, estas iniciativas devem ser levadas em conta quando da elaboração de qualquer solução para a problemática exposta neste trabalho. Se cada desenvolvedor elaborar seu próprio mecanismo para a descrição e localização de recursos, e não levar em conta que uma rede Gnutella é composta por milhões de nós, uns com sua aplicação

e outros com aplicações de desenvolvedores diferentes, a rede Gnutella correrá o risco de se dividir em diversas redes paralelas compentindo por usuários.

Os usuários, por sua vez, têm um papel fundamental para o sucesso da tecnologia Gnutella e também para a incorporação qualquer novo recurso à ela. Hoje, por exemplo, milhões de usuários deixam de colaborar com a arquitetura Gnutella quando somente obtém recursos de outros nós sem, por outro lado, disponibilizar qualquer recurso à rede. Imagine se todos resolvessem cancelar os *uploads* em suas estações e passassem a executar somente *downloads.* Em muito pouco tempo a rede Gnutella ruiria por escassez de recursos. Da mesma forma, a conscientização dos usuários de aplicações Gnutella para a importância da descrição dos recursos compartilhados por eles é um aspecto fundamental. As aplicações que surgirão irão disponibilizar funcionalidades que possibilitarão a descrição dos recursos de forma fácil e os usuários, por sua vez, deverão ser incentivados a utilizar estas funcionalidades, principalmente na fase inicial, onde a maioria dos recursos compartilhados já existentes ainda não estarão descritos. Os usuários deverão ser conscientizados, através das aplicações Gnutella, que assim como a Gnutella possibilita uma capacidade ilimitada de armazenamento de recursos devido ao fato de cada nó colaborar com um pouco dos seu espaço em disco, as pesquisas avançadas só se tornarão uma realidade se cada nó (usuário / aplicação) também colaborar descrevendo os recursos que diponibilizam em seus nós.

As novas versões dos principais clientes Gnutella existentes hoje e as novas aplicações que surgirão, irão ditar a velocidade na qual a rede irá crescer e na qual a localização avançada será implantada. Nos primórdios da rede Gnutella, quando um usuário desejava se conectar à rede, ele ia a diversos *sites* de bate-papo perguntando se existia algum usuário conectado com um cliente Gnutella e qual era seu endereço IP. Com este endereço em mãos, ele abria o seu cliente Gnutella e conectava-se ao nó correspondente e, só então, passava a pesquisar e disponibilizar recursos. Mais tarde, os desenvolvedores de aplicações Gnutella criaram o recurso dos *hosts caches,* que são nós sempre conectados que mantém uma lista atualizada de endereços de nós Gnutella *online* no momento. Assim, as aplicações atuais pesquisam nestes caches e , automaticamente, se conectam aos nós. O usuário simplesmente abre o cliente e começa a utilizá-lo. Assim como o surgimento dos *hosts caches* possibilitou a explosão da utilização de aplicações

P2P/Gnutella, recursos que facilitem criação de descritores de recursos e a execução de pesquisas, podem fazer com que a rede Gnutella evolua para um outro nível: o dos recursos descritos por metadados. Entre estes recursos pode-se citar a busca automática por descritores e um assistente para montagem de pesquisas. A busca automática por recursos consistiria em um processo que, automaticamente, procuraria em outros nós Gnutella descritores já existentes para cada recurso compartilhado no MSF (*My Shared Folder*) que ainda não possua um. A idéia básica é que, se o usuário possui um recurso compartilhado, é muito provável que este recurso esteja sendo compartilhado em outros nós e que já exista um descritor XML/RDF criado por alguém. Após a busca, os descritores são baixados no MSF e, a partir deste momento, o usuário poderia alterá-los ou enriquecê-los. Um assistente para pesquisa também será um grande incentivo. Uma sequência de telas, dos estilo próximo/anterior, que guie o usuário no momento da busca por um recurso. Este assistente, baseado , por exemplo, na iniciativa Dublin Core exposta neste trabalho, solicitaria ao usuário as informações/valores para os elementos de metadados, que caracterizem o seu interesse. Se o usuário selecionou que que uma imagem, por exemplo, o assistente somente solicitaria o preenchimento dos elementos DC aplicáveis a uma imagem (*creator, date, format,description*).

O fato da tecnologia peer-to-peer fazer com que a Internet remonte seus momentos iniciais, isto é, momentos onde ocorria a interação direta entre os nós em um nível de igualdade, faz com que, mais do que nunca, a palavra **colaboração** ganhe importância. O verbo **colaborar** ganha um significado além do de simplesmente disponibilizar arquivos a outros nós, o significado de possibilitar o crescimento da rede Gnutella através de atitudes coletivas, isto é, atitudes tomadas lembrando que cada usuário é a rede Gnutella e porque não dizer, que a rede Gnutella é cada usuário.

8. CONCLUSÃO

Kelly Truelove diz em seu artigo [OPEN] que o protocolo Gnutella restaurou a simetria original da Web, isto é, até mesmo estações transientes podem participar como servidores de recursos. A rede Gnutella está começando a ser efetivamente utilizada para compartilhar informações além dos tradicionais (e polêmicos) arquivos de música MP3 e está apenas em fase embrionária. Um ambiente distribuído, com recursos compartilhados cada vez mais heterogêneos tomou lugar na Internet, dominada por aplicações do tipo cliente/servidor. Com o crescimento da quantidade de recursos compartilhados de diferentes tipos e de áreas de interesse diferentes, não se pode deixar de buscar mecanismos e tecnologias para a descrição e localização destes recursos na rede.

A evolução dos recursos computacionais, principalmente daqueles localizados às margens da Internet, as nossas estações de trabalho, permitiu que fossem utilizados para cooperarem entre si, compartilhando recursos em condições de igualdade. Por isso, as duas palavras que melhor representam a arquitetura P2P são: descentralização e distribuição. Um protocolo aberto e simples, o Gnutella, possibilitou o surgimento de inúmeras aplicações P2P para o compartilhamento de recursos e, possibilitou também, sua popularização. Milhões de estações de trabalho ao redor do mundo compartilham recursos entre si, criando uma gigantesca base de dados com capacidade praticamente ilimitada. Alterações no protocolo Gnutella, mais especificamente adendos aos pacotes, que não comprometam o funcionamento das aplicações existentes, aliadas à utilização de metadados XML/RDF para a descrição dos recursos, mostra-se como a melhor solução para a problemática das pesquisas nas redes P2P. Solução esta totalmente viável e, principalmente, necessária para o progresso desta tecnologia.

Os recursos disponíveis nos nós de uma rede P2P precisam ser descritos e, tal descrição, precisa ser entendida por pessoas e também ser processada por máquinas. A tecnologia RDF une o poder dos metadados (escritos em XML) e uma notação clara e bem definida (marcações), possibilitando nas redes P2P, o compartilhamento também de descritores de recursos. Ao se descrever um objeto/recurso, criam-se elementos de metadados que o caracterizam (nome, tamanho, tipo, etc..). Esta criação, se deixada a critério de cada usuário, causará o surgimento de elementos com nomes diferentes mas querendo

representar o mesmo tipo de informação. Por este motivo, utilizou-se neste trabalho, iniciativas de metadados já existentes, como a DCMI (*Dublin Core Metadata Initiative*). As pesquisas tradicionais carregam somente uma palavra-chave como critério de busca. Os adendos RDF/XML propostos neste trabalho permitem que as pesquisas, transportadas nos pacotes QUERY, carreguem informações mais detalhadas que descrevam melhor o interesse do usuário. Desta forma, com buscas mais refinadas, têm-se também resultados mais específicos. A localização eficaz de recursos nas redes Gnutella não tem como objetivo somente disponibilizar mais uma facilidade aos seus usuários, é também um aspecto crucial para a própria sobrevivência da tecnologia peer-to-peer.

Como sugestões para trabalhos futuros, pode-se citar, (i) a implementação de um *sniffer* para a porta utilizada por aplicações Gnutella (6346), que capture os pacotes de pesquisa, faça a verificação dos adendos e as pesquisas avançadas. Esta implementação deverá possibilitar, como um *plugin,* a qualquer cliente Gnutella o suporte a pesquisas avançadas; e (ii) um estudo para permitir a compactação dos pacotes Gnutella, principalmente dos adendos XML/RDF de pesquisas avançadas. Como mais detalhes serão enviados em consultas, os tamanhos dos pacotes QUERY e QUERY_HIT aumentarão consideravelmente. Um processo para compactação e descompactação dos pacotes de pesquisa será, também, necessário para otimização do tráfego em redes Gnutella.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[ACM a, 1997] – Communications of the ACM .The Past and Future History of the Internet. Vol. 40 No.2, 1997.

[ACM b, 1999] – ACM Computing Surveys. The Tranport Layer: Tutorial and Survey. Vol. 31, No. 4, 1999.

[ACM c, 1992] – Communications of the ACM. Client Server Computing. Vol. 15, No. 7, 1992.

[ACM d, 1988] – Communications of the ACM. The Design Philosophy of the Darpa Internet Protocols. ACM –089791-279-9/88/008/0106, 1988.

[BORD] – Bordignon, Fernando R. A. ; Tolosa, Gabriel H.. Gnutella: Distributed System for Information Storage and Searching Model Description. Disponível online. http://www.gnutella.co.uk/library/pdf/paper_final_gnutella_english.pdf . 2001.

[DCT1] – Dublin Core.Dublin Core Types. Disponível online. http:/dublincore.org/documents/1999/07/02/dces/#dct1

[DUBLIN] – Dublin Core.Using Dublin Core. Disponível online. : http://dublincore.org/documents/2001/04/12/usageguide/

[DUBLINW] – XML.COM .Dublin Core in the Wild. Disponível online. : http://www.xml.com/pub/a/2000/10/25/dublincore/dc8.html . 2000.

[GNU] – Clip2.The Gnutella Protocol Specification v.0.4. Revisão 1.2. Disponível online. http://www.clip2.com

[HEIMB, 2001] – Heimbigner, Dennis. Adapting Publish/Subscribe Middleware to Achieve Gnutella-like Functionality. Disponível online. http://www.cs.colorado.edu/serl/papers/CU-CS-909-00.pdf. 2001.

[ORAM, 2001] - Oram, Andrew. Peer-to-Peer, O Poder Transformador das Redes Ponto-a-Ponto.
Primeira Ed. São Paulo: Berkeley, 2001. 447 p.

[RDFREC] – W3C Recomendation. Resource Description Framework (RDF) Model and Syntax Specification. Disponível online. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222. 1999.

[RFC1766] – Dublin Core. Languages Identifiers. Disponível online. http://dublincore.org/documents/1999/07/02/#rfc1766

[SCHUSS] – Schussel, George. Client/Server: Past, Present and Future. Disponível online. http://www.dciexpo.com/geos/dbsejava.htm..

[SOARES, 1995] - Soares, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sergio. Redes de Computadores: das LANS, MANS e WANS as redes ATM. Segunda Ed. Rio de Janeiro: Campus, 1995. 705 p.

[TANENBAUM, 1997] - TANENBAUM, Andrew S. Redes de Computadores. Terc. Ed. Rio de Janeiro: Campus, 1997. 923 p.

[XML, 2001] - Marchal, Benoit. XML – Conceitos e aplicações. São Paulo: Berkeley, 2000. 548 p.

[MILLER] – Miller, Eric. An Introduction to the Resource Description Framework. D-Lib Magazine 05/1998.

[OPEN] – W3C Recomendation. Resource Description Framework (RDF) Model and Syntax Specification. Disponível online: http://www.openp2p.com/pub/a/p2p/2001/03/22/truelove.html?page=1.