MARCELO HENRIQUE VITHOFT

UM SERVIÇO DE GERENCIAMENTO SEGURO DE CONTEÚDOS P2P COMPARTILHADOS EM MANET

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná**,** como requisito parcial, para obtenção do título de Mestre em Informática.

CURITIBA

2009

MARCELO HENRIQUE VITHOFT

UM SERVIÇO DE GERENCIAMENTO SEGURO DE CONTEÚDOS P2P COMPARTILHADOS EM MANET

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial, para obtenção do título de Mestre em Informática.

Área de Concentração: *Ciência da Computação*

Orientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2009

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

V844s 2009

Vithoft, Marcelo Henrique
Um serviço de gerenciamento seguro de conteúdos P2P compartilhados em Manet / Marcelo Henrique Vithoft ; orientador, Altair Olivo Santin. – 2009.
xvi, 94 p. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2009
Bibliografia: p. 91-94

1. Arquitetura não-hierárquica (Rede de computador). 2. Redes de computação – Medidas de segurança. 3. Sistemas de comunicação móvel. I. Santin, Altair Olivo. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 06/2009

Aos 02 dias do mês de março de 2009 realizou-se a sessão pública de Defesa da Dissertação **"Um serviço de Gerenciamento Seguro de Conteúdos P2P Compartilhados em Manet"**, apresentada pelo aluno **Marcelo Henrique Vithoft** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Altair Olivo Santin PUCPR (Orientador)	*(assinatura)*	Aprov. *(aprov/reprov.)*
Prof. Dr. Luiz Augusto de Paula Lima Junior PUCPR		Aprov.
Prof. Dr. Rafael Rodrigues Obelheiro UDESC	Rafael Obelheiro	APROVADO

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Mauro Sérgio Pereira Fonseca
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 · Fax: (41) 3271 2121 www.ppgia.pucpr.br

Dedico a todos que acreditam que a Educação e Cultura são fundamentais à formação do homem.

Agradecimentos

Agradeço, primeiramente a Deus, por ter me capacitado a entrar e prosseguir no programa de mestrado.

Agradeço ao Prof. Altair Olivo Santin que durante todo o mestrado sempre esteve à disposição, tendo excepcional dedicação e paciência para auxiliar-me em todas as fases.

À minha família que sempre me ajudou e apoiou em todos os meus projetos de vida.

À minha esposa e filha que além do apoio incondicional, foram pacientes em todas as horas e não deixaram que eu esmorecesse.

À empresa Siemens por viabilizar e dar condições de cursar o programa de Mestrado, e a todos que, de uma forma ou de outra, contribuíram para que este trabalho fosse realizado.

Resumo

Redes *Ad hoc* móveis (MANet) são muito flexíveis, mas aos mesmo tempo, são muito instáveis, devido à mobilidade dos dispositivos interconectados. Redes de sobreposição oferecem a abstração apropriada para construir um ambiente distribuído independentemente da rede subjacente. Redes *peer-to-peer* (P2P) são infra-estruturas flexíveis que permitem conexões fim-a-fim, sem a necessidade de servidores. Aplicações usando P2P como rede de sobreposição em MANet (PoM) combinam as principais características necessárias para construir sistemas dinâmicos flexíveis e auto-organizados. No entanto, o ambiente PoM é muito instável em termos de interconexão de MANet, porque *peers* entram e saem de redes P2P com muita freqüência - *churn*. Aplicações P2P baseadas em MANet normalmente exigem considerável *overhead* de uma aplicação P2P consumidora para controlar a comunicação fim-a-fim com o *peer* que provê o conteúdo compartilhado. Além disso, redes P2P tradicionais oferecem frequentemente conteúdos compartilhados poluídos, o que resulta no desperdício de escassos recursos deste tipo de dispositivo. Este trabalho propõe um serviço que minimiza a instabilidade de ambientes PoM para dispositivos móveis e reduz a probabilidade de descarga de conteúdos corrompidos. O protótipo desenvolvido mostrou a viabilidade da proposta e que a mesma pode ser facilmente integrada com infra-estruturas tradicionais.

Palavras-Chave: MANet; *Peer-to-peer*; Compartilhamento de conteúdos P2P; Segurança.

Abstract

Mobile Ad hoc Networks (MANets) are very flexible, but at the same time they are unstable due to the mobility of the interconnected devices. Overlay networks offer an appropriate abstraction to build a distributed environment independently of the underlying network. Peer-to-peer (P2P) networks are flexible infrastructures that enable end-to-end connections without the need of servers. Applications using P2P as an Overlay network in MANets (PoM) combine the main characteristics needed to build dynamic, flexible and self-organizing systems. The PoM environment, however, is very unstable in terms of MANet interconnections because peers join and leave the P2P network very frequently - churn. MANet-based P2P applications usually produce considerable overhead due to the control of the end-to-end communication with the resource provider peer. Moreover, traditional P2P networks offer polluted contents wasting the device's scarce resources. This work proposes a service that minimizes the PoM environment instabilities and reduces the probability of downloading corrupt contents. The prototype developed showed the viability of the proposal and that it can be easily integrated into the traditional P2P infrastructure.

Keywords: MANet*; Peer-to-peer; P2P content sharing; Security*.

Sumário

Lista de Figuras e Tabelas

Lista de Símbolos

$\boldsymbol{N}$	Número de dispositivos conectados à rede *ad hoc*
$\boldsymbol{t}$	Tempo para uma mensagem trafegar entre dois nós da rede
$\boldsymbol{dt_M}$	tempo total para um *peer* consumidor fazer a descarga de um conteúdo diretamente de um *peer* provedor na MANet
$\boldsymbol{dt_c}$	tempo total para um *peer* consumidor fazer a descarga de um conteúdo que se encontra na *cache* do VSP
$\boldsymbol{dt_p}$	Tempo total para descarga de um conteúdo do *peer* provedor até o VSP e posteriormente para o *peer* consumidor
A_{http}	Atraso proveniente da busca de um fragmento de conteúdo no *peer* provedor via http socket (VSP <-> *peer* provedor)
A_{JXME}	Atraso proveniente da busca de um fragmento de conteúdo no *peer* provedor via mesnagens JXME (VSP <-> *peer* provedor)
t_{http}	tempo de transmissão de uma mensagem via http socket (VSP <-> *peer* provedor)
t_{JXME}	tempo de transmissão de uma mensagem via mensagens JXME (VSP <-> *peer* provedor)

Lista de Abreviaturas

ATM	*Asynchronous Transfer Mode*
CLDC	Configuração de Dispositivo Conectado Limitado
DHT	*Distributed Hash Table*
DIS	*DHT Index Service*
DSP	*DHT Service Provider*
HTTP	*HyperText Transfer Protocol*
IP	*Internet Protocol*
JXTA	Acrônimo para a Palavra *Juxtapose* (em Português, Justapor)
J2ME	*Java to Micro Edition*
JXME	*Java to Micro Edition* com suporte a JXTA

LDAP	*Lightweight Directory Access Protocol*
MANet	*mobile ad hoc networks*
MD5	*Message-Digest algorithm 5*
MIDP	Perfil de Dispositivo de Informação Móvel
MIT	*Massachusetts Institute Technology*
NAT	*Network Address Translation*
NFS	*Network File System*
OEM	*Original Equipment Manufacturer*
OS	*Operating System*
P2P	*Peer-to-Peer*
PAM	*Pluggable Authentication Model*
PBP	*Peer Binding Protocol*
PC	*Personal Computer*
PDA	*Personal digital assistants*
PDP	*Peer Discovery Protocol*
PEP	*Peer Endpoint Protocol*
PIP	*Peer Information Protocol*
PMP	*Peer Membership Protocol*
PRP	*Peer Resolver Protocol*
PSTN	*Public Switched Telephone Network*
RPV	*Rendezvous Peer View*
SHA-1	*Secure Hash Algorithm 1*
SST	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
VoIP	*Voz Over Internet Protocol*
VSP	*Virtual Service Peer*
XML	*Extensible Markup Language*

Capítulo 1

Introdução

Redes *peer-to-peer* (P2P) são caracterizadas pela dinâmica com que os *peers* entram e saem da rede *churn*, assim como pela variedade de tipos de conteúdos compartilhados que oferecem [SCHOLL2001]**.** Redes *Ad hoc* móveis (MANets) oferecem grande flexibilidade de configuração, mas são muito instáveis em termo de manutenção da conectividade [PERK2001].

Acompanhando a evolução tecnológica e social, os dispositivos móveis (aparelhos celulares, smartphones, etc.) têm sido parte de nosso dia-a-dia. Naturalmente, a maioria destes dispositivos são capazes de se comunicar por ondas de rádio (conexão sem fio) em MANets, apesar de suas limitações em termos da capacidade de processamento, de armazenamento e de taxa de transferência, quando comparados aos computadores pessoais.

Aplicações que usam P2P como rede de sobreposição (*overlay*) em ambientes MANet (PoM) devem ter boa tolerância ao *churn* e instabilidade da conexão [PERK2001] para funcionar corretamente. Ou seja, dois importantes desafios no desenvolvimento de aplicações em PoM são: suportar a conectividade *ad hoc* com os dispositivos móveis vizinhos e manter a disponibilidade dos *peers* em suas conexões fim-a-fim.

Atualmente, grande parte dos conteúdos compartilhados nas redes P2P tradicionais (e.g., *emule*, *gnutella*, etc.) são poluídos, falsificados ou corrompidos (“lixo P2P”) [KUMAR2006]. Consequentemente, os dispositivos móveis que consomem recursos da rede P2P podem gastar uma parte importante de seus recursos, estabelecendo e mantendo conexões com *peer* provedores que podem fornecer lixo P2P. Além disto, estes provedores podem tornar-se ausentes durante o fornecimento dos conteúdos ou a conectividade da MANet pode ser perdida.

Quando as redes P2P foram propostas, não foi prevista uma infra-estrutura para publicar e encontrar conteúdos (repositório de índices); a publicação/busca por conteúdos compartilhados era feita através do envio de mensagens em *broadcast*. A *Distributed Hash Table* (DHT) assumiu o papel repositório de índices de conteúdos compartilhados para as redes P2P [STOICA2001]. O principal benefício alcançado com o uso de DHTs em redes P2P é consequentemente a indexação de conteúdos. Além disto, a DHT é distribuída, tolerante a faltas e visa manter-se sempre disponível.

Baseado em uma rede de sobreposição, nossa proposta define um serviço que libera o *peer* móvel (que geralmente tem o papel de consumidor na rede P2P) da tarefa de controlar conexões fim-a-fim com o *peer* que lhe fornece conteúdos compartilhados. Este esquema permite minimizar o efeito negativo do *churn* da rede P2P para o *peer* consumidor, fazendo com que o conteúdo compartilhado seja ininterruptamente fornecido ao *peer* consumidor de forma transparente.

O serviço proposto também visa minimizar o impacto da instabilidade da conectividade de MANet mantendo, em nível de aplicação, o estado conexão e permitindo transferências parciais de conteúdos. O serviço opera de modo seguro – garantindo a integridade, autenticidade e disponibilidade dos conteúdos. Neste cenário, também foi considerada a migração semi-automática de conteúdos compartilhados a partir de redes P2P tradicionais para a rede orientada a serviço em PoM.

O restante desta dissertação está organizado como apresentado a seguir. O capítulo 2 faz uma breve apresentação de aspectos relativos às redes de sobreposição, MANet e P2P. A apresentação de trabalhos relacionados com o tema proposto é feita no capítulo 3. A proposta, juntamente com o protótipo, são apresentados no capítulo 4. No capítulo 5, o cenário criado para realizar a avaliação da proposta, bem como a avaliação dos resultados, são apresentados. E finalmente, conclusões e trabalhos futuros são abordados no capítulo 6.

Capítulo 2

Redes *Overlay*, *Peer-to-peer* e *MANet*

Este capítulo é destinado à fundamentação teórica dos conceitos utilizados nesta dissertação de mestrado.

Esta base teórica é importante para nivelar o conhecimento e permitir o acompanhamento do raciocínio que será desenvolvido no capítulo destinado à elaboração da proposta do tema a ser desenvolvido.

Este conhecimento será útil também durante a apresentação do estado da arte, ou seja, casos de usos que foram estudados para nortear e motivar este trabalho.

Por tratar-se de um trabalho baseado em sistemas *P2P* (P2P), este deveria ser o ponto de partida para a fundamentação, no entanto, o conceito de *overlay* é extremamente importante para que se entenda P2P. Desta forma, o tópico a seguir, tratará este tema.

2.1. *Overlay*

Uma rede *overlay* é uma rede "virtual" criada sobre uma rede física existente, entre as quais podem ser citadas: rede IP. A rede *overlay* apresenta um nível de abstração mais alto para esta rede. Pode apresentar uma proposta de arquitetura exclusivamente para este nível de abstração que permita solucionar possíveis problemas de difícil solução, encontrados no nível físico, tais como de roteamento. A internet é um exemplo de *overlay*, quando faz uso do protocolo IP como solução de *interworking* sobre tecnologias de redes diversas como ATM, *Frame Relay*, PSTN, LANs e outros. [ROCH2004]

Com base no conceito, acima descrito, pode-se considerar que grande maioria das

redes P2P implementadas podem ser consideradas como redes *overlay*, uma vez que funciona como uma rede virtual, formada pela interconexão dos nós (*peers*), executando sobre a infra-estrutura de uma rede física, como ilustra a Figura 2-1. No entanto, é importante ressaltar que *overlay* e P2P não são sinônimos, ou seja, nem toda rede *overlay* é P2P e da mesma forma, nem toda rede P2P é *overlay*.

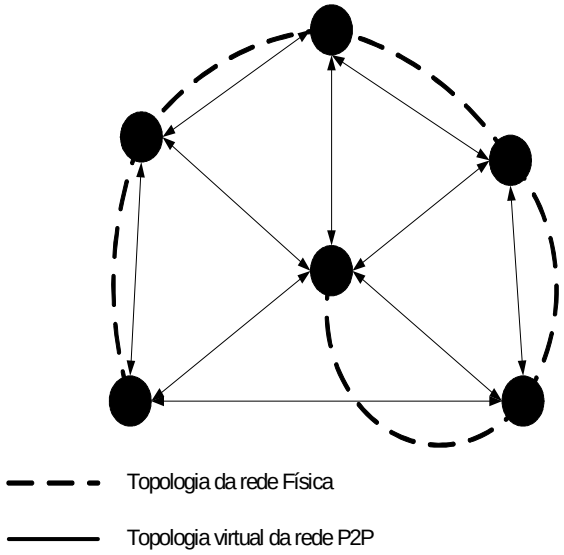

Figura 2-1: Rede *Overlay* [ROCH2004]

2.1.1. DHT

As tabelas de *hash* distribuídas (DHT) são utilizadas para o mapeamento de chaves em valores (obtidos através de uma função de cálculo de *hash*) assim como as tabelas tradicionais de *hash*, que, no entanto, destinguem-se da tradicional pela distribuição de partes desta tabela entre os nodos/peers da rede *overlay*.

Cada *peer* fornece o mapeamento para um grupo de chaves e que são distribuídas na DHT de acordo com o identificador dos *peers*. O identificador único (ID) de cada *peer* é obtido através da mesma função *hash* que dá origem às chaves da DHT. Além das informações da tabela de pares <chave, valor>, os *peers* possuem também uma tabela de roteamento, com informações sobre outros *peers*, que é utilizada para a localização dos nós conhecidos, também chamados de nós adjacentes ou vizinhos.

Para garantir o desempenho de uma rede *P2P* diante do *churn* dos *peers* participantes desta e, consequentemente, da DHT, o conjunto de entradas da DHT (pares <chave, valor>) também é replicado nos nodos adjacentes ao nodo, pois estes nodos adjacentes garantirão a integridade da DHT com o armazenamento dos pares, em caso de saída do nodo em questão.

Este conceito ficará claro quando forem abordadas as várias implementações de DHT.

Os fatores abaixo devem ser considerados por ocasião da criação de um algoritmo para as DHTs:

- **Balanceamento de Carga**: a identificação das chaves e dos nodos (*peers*) geralmente é feita por um número único de **n** bits (ID). A distribuição das chaves entre os *peers* da rede deve ser feita de acordo com a proximidade do identificador dos *peers*,
- **Encaminhamento de Mensagens ao Nó Adequado:** quando um *peer* receber uma consulta por uma determinada chave, este *peer* precisa enviar esta mensagem ao *peer* mais próximo possível do *peer* responsável por armazenar o conjunto de chaves, ao qual a chave procurada pertence,
- **Construção Dinâmica de Tabelas de Roteamento:** para permitir a transmissão de mensagens e a localização dos *peers* e dos conteúdos, os *peers* necessitam de um mecanismo que lhes permitam conhecer outros *peers* presentes na rede *overay*. As tabelas de roteamento armazenadas em cada um dos *peers* fazem parte deste mecanismo. Estas tabelas exigem alterações constantes para refletir o estado dos *peers* que entram e saem da rede de forma dinâmica**,**
- **Definição da Função Distância**: define a métrica que será usada para determinar a distância que uma chave está de um *peer*, para que seja armazenada no *peer* responsável pelo grupo de chaves adequado. Esta métrica norteará o particionamento da tabela *hash* distribuída. Cada implementação define a sua própria Função Distância.

Existem inúmeras implementações de DHT no mercado, entre as quais, destancam-se: Chord [ROCH2004], Pastry [ROCH2004], Tapestry [ROCH2004] e CAN [ROCH2004].

2.2. *Peer-to-Peer*

Por definição, as redes *P2P* são caracterizadas pela comunicação ponto-a-ponto, entre duas entidades computacionais (PC, *palm tops*, celulares e muitos outros dispositivos), sem a existência de servidores, diferentemente do que, tradicionalmente, ocorria nas redes cliente/servidor.

Na literatura, é possível encontrar algumas definições para esta tecnologia e como ilustração, segue a definição encontrada em [BARC2006]:

> "Redes *Peer-to-Peer* (P2P) são sistemas distribuídos, consistindo de nodos interconectados, capazes de se auto-organizar em topologias de overlay, com o objetivo de compartilhar recursos, tais como: conteúdo (música, vídeos, documentos, etc.), ciclos de CPU, armazenamento e largura de banda capazes de se adaptar a populações transientes de nodos enquanto mantendo conectividade aceitável e desempenho, sem necessitar da intermediação ou apoio de uma entidade central."

Neste tipo de rede (*P2P*), as entidades computacionais chamadas de *peers*, respondem pelos papéis de cliente e servidor (ainda referindo-se às arquiteturas tradicionais cliente/servidor) ao mesmo tempo. Ao longo deste trabalho, serão apresentadas redes *P2P* com topologias, onde a existência de *peers* centrais é permitida. No entanto, com o papel de registro de conteúdo, grupos de *peers* e até mesmo para fim de segurança (autenticação, autorização, certificação).

Inicialmente, as redes P2P eram destinadas ao uso doméstico e, com o passar do tempo e consequente evolução desta tecnologia, têm conquistado maior espaço nos cenários acadêmico e corporativo.

As redes P2P são fortemente utilizadas para compartilhamento de recursos porém, não só para este fim. Entre outras, destacam-se aplicações como:

- *Instant messaging (.NET Messenger Service, AOL Instant Messenger, Excite/Pal, Gadu-Gadu, Google Talk, iChat, ICQ, Jabber, Qnext, QQ, Meetro, Skype, Trillian, Yahoo! Messenger and Rediff Bol Instant Messenger),*
- *Grid Computing*;

As características desta modalidade de redes serão descritas a seguir.

2.2.1. Características

Neste tópico, serão abordadas as principais características de redes *P2P*, entre os quais se destaca o dinamismo de seus elementos, ou seja, a alta freqüência com que os *peers* entram e saem da rede. Tipicamente, em uma rede P2P não existem regras que regulamentem o ato de entrar e sair de uma rede. Esta característica acaba resultando em outras, como a necessidade de auto-organização das redes P2P, além de gerar a necessidade de tratamento adequado a

esta prática, sem a ocorrência de danos à rede como um todo.

A seguir, um breve comentário das características mais marcantes de um sistema *P2P*:

- **Descentralização**: como não poderia ser diferente por tratar-se de uma rede distribuída, a descentralização dos recursos torna evidente, pois todos os *peers* são colaboradores na provisão de recursos,
- **Auto-Organização**: as redes P2P são caracterizadas pelo *churn* e isto exige desta modalidade de rede, a capacidade de auto-organizar-se. Com o entra e sai dos *peers* de uma rede, é fundamental que esta seja capaz de manter atualizadas as tabelas de índices dos conteúdos e registros de *peers*, em redes estruturadas ou híbridas, ou então, em redes não estruturadas, criar mecanismos que permitam tratar a saída de um *peer* durante o *download* de um conteúdo. Em resumo, é a capacidade que a rede deve ter para tratar *churn* de modo a não causar grandes distúrbios,
- **Escalabilidade**: com a união das características anteriormente citadas: dinamismo, descentralização e auto-organização, não é difícil entender que a entrada de novos *peers* não seja difícil. Então, a expansão de capacidade de sistemas *P2P* é uma atividade relativamente fácil, o que caracteriza a escalabilidade do sistema, além do que, estes sistemas são caracterizados pelo baixo acoplamento entre os seus *peers* e quanto menor for o acoplamento entre os elementos, mais escalável é o sistema.
- **Anonimidade:** tipicamente, os usuários de uma rede *P2P* podem fazer uso de conteúdos protegidos (como por exemplo: músicas, filmes, jogos, software que têm seus direitos autorais protegidos por lei) sem preocupações legais, pois a censura de conteúdo digital é difícil. No entanto, cada vez mais, tem-se usado o recurso de assinatura digital, nos conteúdos disponíveis em redes desta modalidade e, desta forma, esta propriedade deixa de ser verdadeira, principalmente em sistema corporativos e aplicações comerciais,
- **Técnicas para aumento desempenho/ disponibilidade**: entre elas, destacam-se: Replicação - Cópias dos conteúdos perto dos *peers* que fazem as requisições, porém, as mudanças nestes conteúdos devem ser propagadas entre as réplicas; *Cache* – assim como a técnica anterior, visa à redução do caminho de busca e que requer o mesmo cuidado para propagação de mudanças de conteúdos; Roteamento inteligente – busca a ligação dinâmica inteligente entre os *peers*, a aproximação de *peers* com interesses semelhantes e diminuir tráfego de mensagens na rede,

2.2.2. Aplicações P2P

Tipicamente, as redes *P2P* são utilizadas com a finalidade de prover as funcionalidades/aplicações que serão destacadas a seguir, conforme ilustra muito bem a publicação "Segurança em Redes P2P: Princípios, Tecnologias e Desafios" [BARC2006].

- **Compartilhamento de arquivos** (*file sharing*): a mais popular das aplicações para redes *P2P* é o compartilhamento de arquivos, ou ainda, compartilhamento de conteúdo. É importante ressaltar que nesta modalidade de aplicação, os arquivos não devem ter seus conteúdos alterados, ou seja, não será permitida a alteração de seu conteúdo pelos usuários do sistema. Neste tipo de aplicação, é permitido a qualquer usuário publicar conteúdos (neste caso, arquivos), que posteriormente poderão ser acessados por demais usuários de forma geograficamente distribuída. Tipicamente, qualquer usuário pode publicar ou usar estes arquivos disponíveis na rede. No entanto, algumas redes podem e implementam regras de uso. Entre estas regras, é comum encontrarmos sistemas que premiam os usuários que publicam conteúdos, com privilégio de acesso a outros, ou seja, quanto maior o número de publicações, maior será o privilégio de acesso,
- **Sistema de armazenamento de arquivos em rede** (*network storage*): esta modalidade de aplicação é similar a anterior (compartilhamento de arquivos), porém, caracteriza-se por permitir que os arquivos compartilhados sejam alterados. Em redes P2P com replicação de informação, é importante que as aplicações sejam capazes de atualizar os conteúdos das réplicas. Tipicamente, nestas redes, regras que restrinjam a escrita e leitura devem ser definidas para o bom funcionamento, através da implementação de mecanismos de controle de acesso,
- **Transmissão de dados ou *overlay multicast*:** *overlay* forma uma infra-estrutura de comunicação que visa suprir a ausência de suporte *multicast* nativo na rede. Esta facilidade permite que um mesmo conteúdo seja transmitido por um *peer* e entregue a um número potencialmente grande de *peers* (usuários) geograficamente dispersos na rede. Esta aplicação tem sido usada para transmissão de eventos ao vivo,
- **Computação distribuída**: basicamente, esta aplicação explora a ociosidade de processamento em *peers,* da rede (neste caso, mais adequadamente chamada de grade

– *grid computing*), garantindo o processamento intensivo. Esta modalidade de aplicação ganha, cada vez mais, destaque devido à crescente necessidade de capacidade computacional exigida. Em outras palavras, o sistema distribui processamento entre os *peers* membros da rede, balanceando e ganhando capacidade de processamento. Associado a esta modalidade de aplicação, surge a necessidade de controle de acesso e a gerência de reputação, mecanismos fundamentais para o bom funcionamento destes sistemas,

- **Colaboração e comunicação entre usuários**: com a crescente popularização de computadores pessoais (PC) e a globalização dos relacionamentos pessoais, este tipo de aplicação tem obtido mais força. Estas aplicações permitem que usuários se comuniquem através de voz (VoIP), mensagens de texto, imagens gráficas e arquivos, em geral de forma direta, sem passar por um servidor. Alguns exemplos de sistemas que implementam esta modalidade de aplicação que merecem destaque, são: aplicações gerais de *Instant Messaging* como ICQ, MSN *Messenger*, *Yahoo Messenger* e, em particular, o *Skype*;

É importante deixar claro que não são apenas as aplicações imediatamente acima categorizadas que compõem o mundo dos sistemas *P2P*. Existem outras aplicações que empregam conceitos de P2P, como por exemplo, “sistema gerente de bases de dados distribuída”.

2.2.3. Classificação das redes P2P

A organização de redes P2P acontece em função de sua organização e topologia.

Como citado anteriormente, as redes P2P são provedores de infraestrutura para o compartilhamento de recurso, através da participação/ colaboração de inúmeros nodos, os quais são denominados *peers*.

Por definição, um sistema *P2P* diferencia-se de uma arquitetura cliente/servidor pela inexistência de servidor centralizado, e onde todos os elementos (*peers*) desempenham tanto a função de cliente, quanto a função de servidor. A existência de *peers* centrais nos sistemas *P2P* é permitida, no entanto com o papel de registro de conteúdo, grupos de *peers* e até mesmo para fim de segurança (autenticação, autorização, certificação).

I. Topologias

A classificação das redes P2P em função da topologia, é feita como:

- **Centralizada**: faz uso de uma entidade central utilizada pelo sistema para o registro dos *peers* e respectivos conteúdos, que é, na maioria das vezes, chamado de *super-peer*,
- **Distribuída**: esta, a mais pura das topologias de redes *P2P*, caracterizada, exatamente, pela inexistência de *peers* centrais e cuja técnica de busca de conteúdo se dá, na maioria das vezes, por inundação. (representante: Gnutella),
- **Hierárquica**: a base de localização de *peers* e conteúdos devem ser distribuídos entre os chamados *super-peers* (*peers* que acumulam funções de organização do sistema e o provisionamento de conteúdo). Os *super-peers* estão distribuídos dentro da rede P2Ps e que se caracterizam também por dar acesso a um novo nível hierárquico de uma rede P2P;

As topologias centralizadas e hierárquicas são classificadas como híbridas, pela existência de *peers* centralizados, com a função de registro de índices e grupos, além de dar auxílio aos métodos de segurança (autenticação, autorização e certificação). A topologia distribuída é classificada como pura, por não possuir entidades centralizadas e ter todas as funcionalidades distribuídas entre os *peers*.

Nos tópicos a seguir, estas topologias serão apresentadas com mais detalhes.

A. Topologia de rede P2P Centralizada (Híbrida)

Mesmo com a existência de um *peer* centralizado, como ilustra a Figura 2-2, esta topologia não descaracteriza um sistema P2P por ser funcionalmente fiel às definições desta tecnologia. Esta topologia faz uso de um nodo central para prover serviços de índices (registro de recurso) e registros de grupos, com o intuito de melhorar o desempenho da rede, principalmente durante a busca/ procura de um conteúdo para *download*.

O *Napster* é uma rede que implementa esta topologia de rede *P2P*.

Na Figura 2-2, é apresentado um exemplo onde o PeerId#01 registra um recurso no *Peer* Centralizado em um primeiro momento. Posteriormente, o PeerId#02 faz uma consulta pelo recurso A, que fora anteriormente registrado. O *Peer* Centralizado responde à requisição do PeerId#02, informando a localização do recurso (A). Por fim, o PeerId#02 acessa o provedor do recurso e baixa o conteúdo A, por ele requisitado.

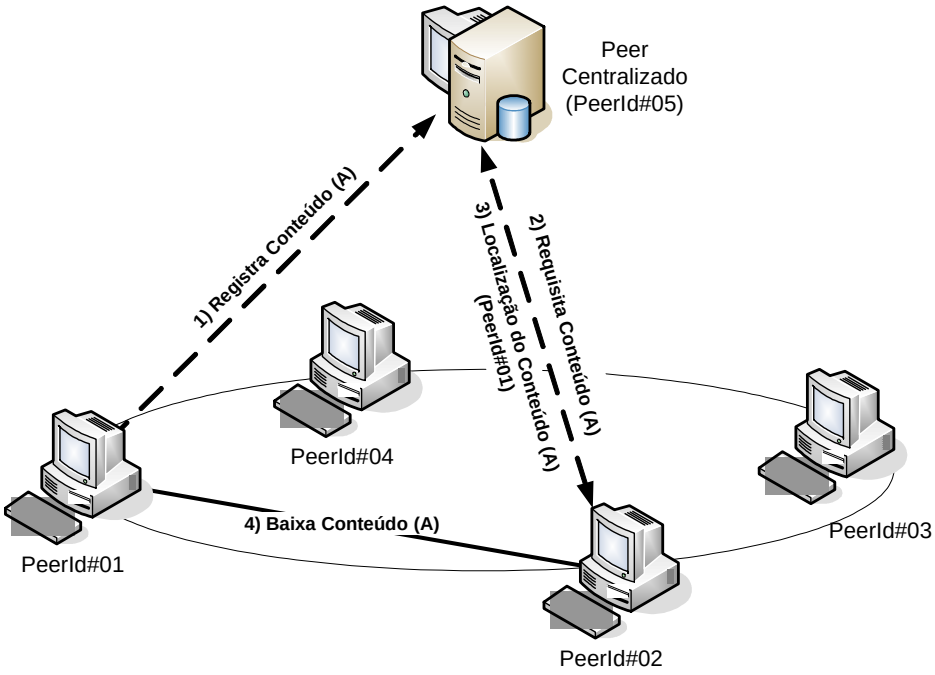

Figura 2-2: Topologia P2P Centralizada (Híbrida) [OLIV2005]

B. Topologia de rede P2P Hierárquica (Híbrida)

Esta topologia é caracterizada pela existência de *super-peers* do primeiro nível hierárquico capazes de atuar como *peer* servidor(centralizado) para os *peers* do segundo nível hierárquico, conforme ilustra a Figura 2-3.

O sistema deverá ser dinâmico, para que, com a saída de um destes *super-peers* da rede, o sistema deve ser capaz de auto-organizar-se, de modo a garantir o funcionamento, sem causar perda de funcionalidade.

Sobre a Figura 2-3 é importante ainda comentar que, em um primeiro momento, os *peers* da camada hierárquica inferior registram seus conteúdos/recursos em seus respectivos *Peers* Centralizados. Os passos denominados por x, y, z, w e T (na referida figura) são dependentes de implementação, ou seja, o registro automático de recursos entre *peers* centralizados (passos x, y e z) pode ou não ocorrer. Os passos denominados por w e T, por sua vez, serão necessários apenas, se os passos x, y e z não ocorrerem espontaneamente.

Como ilustração, o *framework* **JXTA** implementa esta topologia para prover a infraestrutura para a aplicação *P2P* que será implementada sobre ele e com a propagação automática/ espontânea dos registros de recursos entre *peer* centralizados, nele chamados de *super-peers rendezvous*.

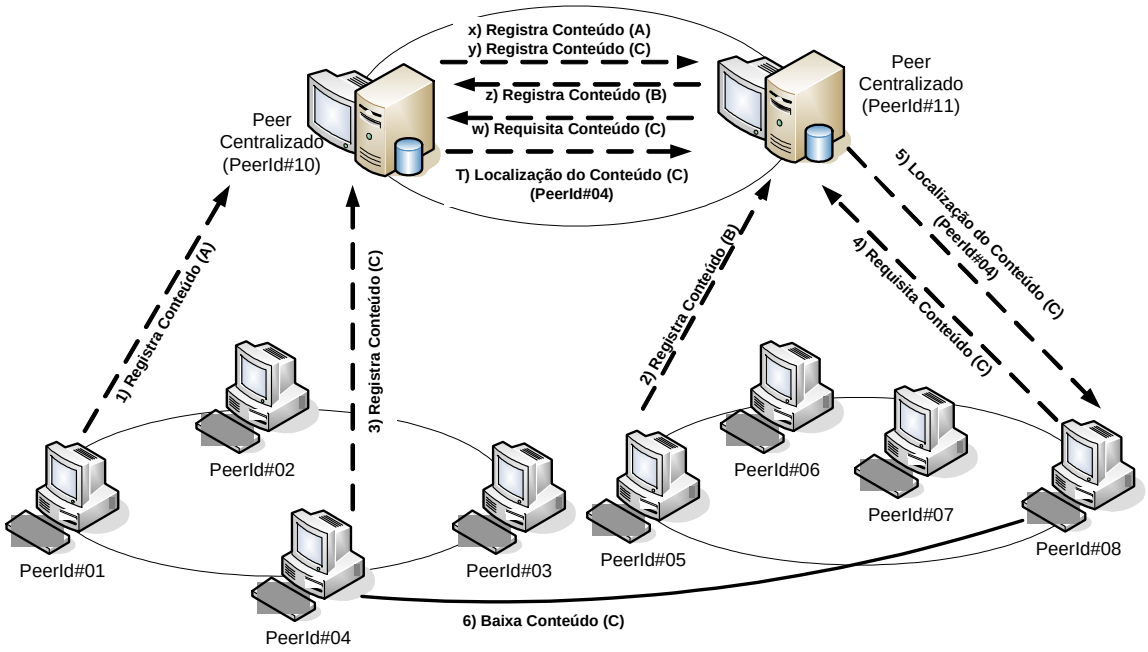

Figura 2-3: Topologia P2P Hierárquica (Híbrida) [TRAV2003]

C. Topologia de rede P2P Distribuída (Pura)

Esta topologia de rede é caracterizada por *peers* que desempenham, tanto a função de cliente, quanto a de servidor (*servent*), ou seja, todos os *peers* têm as mesmas funções. Não há qualquer tipo de controle de conexão (*peers* com funções especiais para controle da rede). Os sistemas Gnutella 0.4 e Freenet são exemplos típicos deste de redes ***P2P,*** com estas características.

A Figura 2-4 ilustra este tipo de topologia de redes *P2P*, mostrando que existe o relacionamento cruzado entre todos os *peers* da rede, ou seja, as buscas de conteúdos são realizadas por inundação.

Ainda sobre a Figura 2-4, é fundamental que os passos 3, 4 e 5 (que ocorrem entre os *peers* com Id 01, 03 e 04) sejam dependentes de implementação, pois uma alternativa para a apresentada na figura referenciada neste parágrafo, seria:

- com o recebimento no peerId#04, da confirmação da localização do conteúdo (A) no peerId#03, o peerId#04 poderia buscar este conteúdo no peerId#03,
- dando seqüência a esta alternativa de implementação, o peerId#04 responderia a peerI#01 que o conteúdo encontra-se disponível nele, por sua vez, o peerId#01 buscaria no peerId#04 o conteúdo (A) desejado e não mais no peerId#03 como

ilustrado na Figura 2-4;

E esta é apenas uma das possíveis alternativas para a topologia abordada neste item.

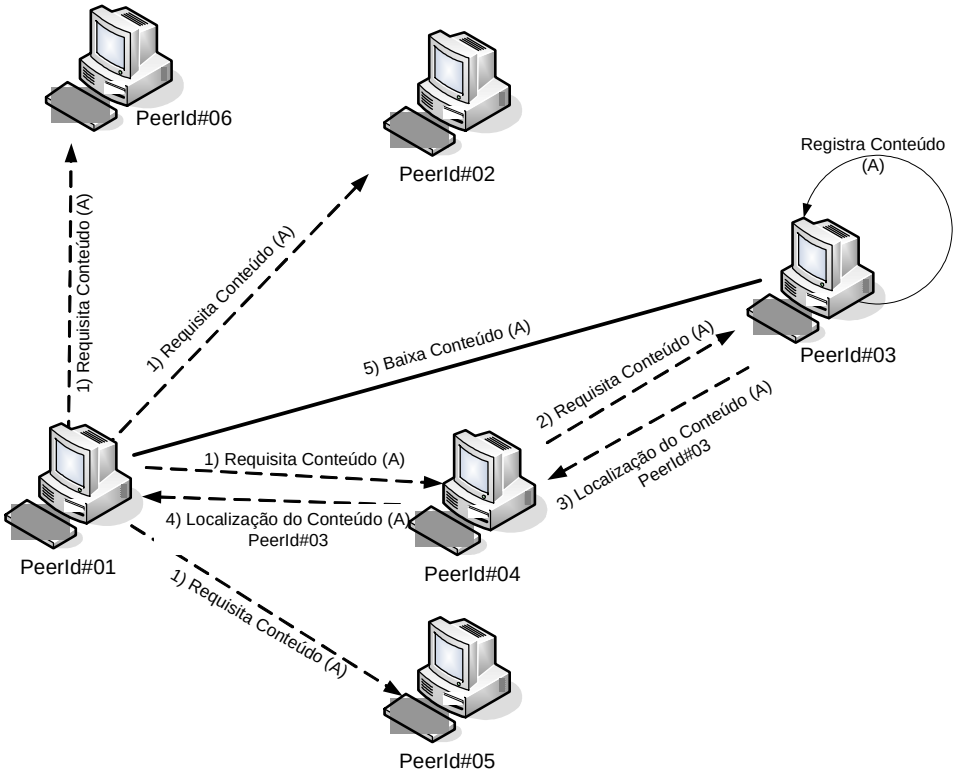

Figura 2-4: Topologia P2P Distribuída (Pura) [ROCH2004]

O que deve ficar claro é que, para cada uma das topologias apresentadas nesta seção, podem ser encontradas alternativas diferentes de implementação, o que resulta em representações diferentes, para uma mesma topologia abordada.

II. Organização

A organização de uma rede tradicional P2P é subdividida em pura estruturada e não-estruturada, cuja caracterização se dá pela capacidade de recuperar ou não, toda a informação disponível em um sistema P2P.

- **Redes P2P Não-Estruturadas**: este tipo de rede P2P é representada pelo *Gnutella*, que faz uso de mensagens em broadcast (por inundação) para descobrir a localização de uma informação desejada. Neste tipo, nem sempre a informação disponível na rede, é encontrada, por fatores como *timeout* ou mesmo perda de conectividade com *peers* intermediários. A Figura 2-4 ilustra a busca, com sucesso, por um conteúdo e evidencia a quantidade de mensagens geradas para esta operação neste tipo de rede

P2P.

- **Redes P2P Estruturadas:** este tipo de rede P2P caracteriza-se pelo uso de métodos deterministas para busca de informação que permite a recuperação de todas as informações disponíveis na rede. Uns dos exemplos típicos de sistema estruturados são os baseados em tabelas *hash* distribuídas (DHT – *Distributed Hash Table*). Alguns sistemas baseados em DHT: *Chord* [ROCH2004], *Pastry* [ROCH2004], *Tapestry* [ROCH2004] e *CAN* [ROCH2004].
- É importante deixar claro que rede *P2P* estruturada não deve ser sinônimo de DHT, embora a DHT seja amplamente usada em redes estruturadas, mas não é obrigatório este relacionamento.
- O tipo de busca por recursos dentro de redes P2P tem sido usado como forma de classificar redes P2P, das quais se destacam:
- **Busca centralizada**: rede com um ponto central (possivelmente espelhado para outros pontos, dando a impressão de serem vários) de busca e nós que consultam o ponto central para trocar informações diretamente entre os *peers*,
- **Busca por inundação**: rede com nós totalmente independentes, onde normalmente a busca é limitada à vizinhança mais próxima do nó que fez a busca (assim, a busca é escalável, mas não é completa),
- **Busca por tabela *hash* distribuída (DHT)**: rede onde os nós têm autonomia e usam uma tabela *hash* para separar o espaço de busca entre eles;

2.2.4. JXTA

A tecnologia JXTA é uma plataforma de desenvolvimento computacional e de redes que foi projetada para resolver problemas na computação distribuída, mais especialmente para sistemas P2P.

O projeto de JXTA tinha como objetivo a criação de um *framework* independente de linguagens de programação e protocolos de transporte, que possa ser implementado em qualquer dispositivo conectado à rede IP ou não.

O JXTA oferece uma infraestrutura P2P completa que é capaz de prover as funções básicas de um sistema deste tipo: busca e compartilhamento de recursos entre elementos participantes (*peers*).

O compartilhamento de informações nas redes JXTA é possível por intermédio de

diversos protocolos de mensagens entre os *peers*.

I. Arquitetura do JXTA

A arquitetura do JXTA é composta de três camadas, conforme mostra a Figura 2-5 e que serão detalhadas a seguir:

- **JXTA Core**: encapsula as primitivas mínimas e essenciais que são comuns a todas as aplicações P2P, que inclui os blocos componentes dos mecanismos chaves de aplicações P2P, incluindo descoberta de recursos, transporte, criação de *peers* e grupos de *peers*, e as primitivas de segurança,
- **JXTA *Services***: suporta funções de alto-nível, como: buscas, compartilhamento de arquivos e recursos, indexação e *caching*,
- **JXTA *Application***: faz uso das camadas anteriores para suportar aplicações P2P mais complexas, como: gerenciamento de conteúdo busca compartilhada, computação distribuída e mensagens instantâneas;

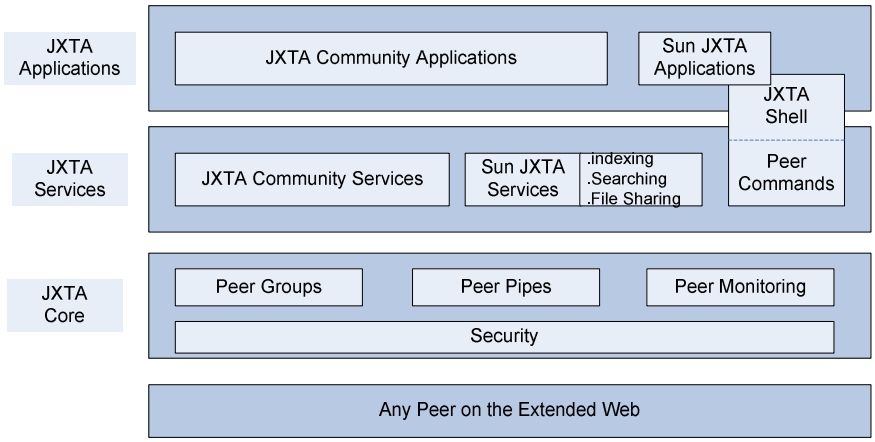

Figura 2-5: Arquitetura do *framework* JXTA [GONG2002]

II. Protocolos do JXTA

O conjunto de protocolos do JXTA é composto por 6 protocolos: *Peer Endpoint Protocol* (PEP), *Peer Resolver Protocol* (PRP), *Peer Discovery Protocol* (PDP), *Peer Binding Protocol* (PBP), *Peer Information Protocol* (PIP) e *Peer Membership Protocol* (PMP), que garantem: descoberta, organização, monitoramento e comunicação entre *peers* que são dispostos em forma de pilha, assim como no OSI da ISO

Cada um destes protocolos, anteriormente mencionados, são independentes uns dos outros. Para que um *peer* seja considerado um *peer* JXTA, não é exigido dele que seja implementado todos estes protocolos do JXTA ou seja, os *peers* podem implementar apenas os protocolos de que necessite fazer uso. No entanto, todos os *peers* devem implementar protocolos para que seja endereçável como um *peer*: PRP e ERP. A Figura 2-6 ilustra a obrigatoriedade ou não, de implementação dos protocolos do JXTA.

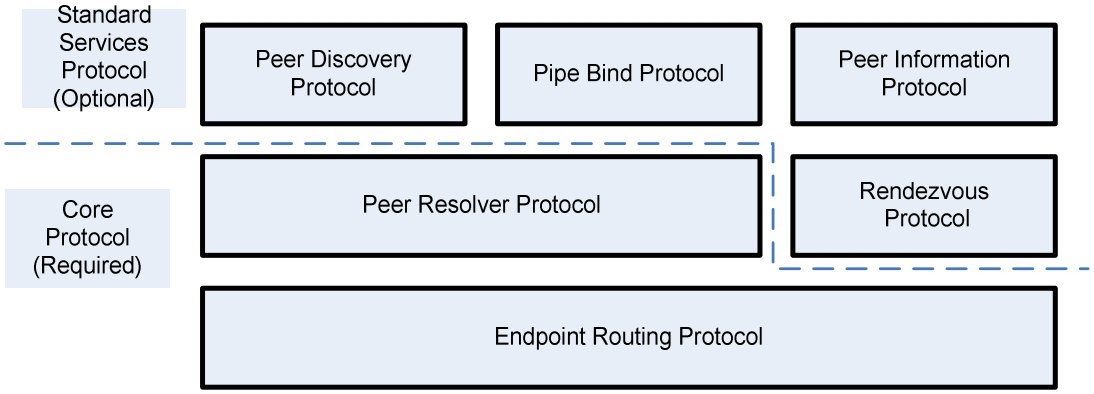

Figura 2-6: Protocolos do JXTA: Representação em Níveis [TRAV2002]

III. JXTA para sistemas embutidos (embedded systems)

JXTA é uma especificação de um *framework* para aplicações P2P que não depende de linguagens de programação ou protocolos de transporte, o que permite aos desenvolvedores, a possibilidade de optar pela linguagem de programação que mais lhe agrade e o mesmo princípio é válido para o protocolo de transporte.

A comunidade JXTA tem usado a arquitetura do JAVA Micro *Edition* (J2ME), solução JAVA para dispositivos de pequeno porte e que necessitam de programação embutida, entre os quais, destacam-se os aparelhos celulares e os PDAs .

É difícil para um dispositivo embutido de menor porte, como alguns sensores inteligentes, implementar os protocolos de JXTA, mas pode-se utilizar um aplicativo *desktop*, como *gateway* de uma rede de sensores com menor poder computacional ou de comunicação. Dessa forma, é possível interligar diversas redes de sensores e realizar a descoberta automática de recursos, comunicação entre sensores e outras funcionalidades que o JXTA permite.

Como complemento a este tema, uma breve descrição da plataforma J2ME do JAVA será apresentada a seguir.

2.2.5. Project JXTA 2.0 *Super-Peer Virtual Network*

O projeto JXTA 2.0 [TRAV2003] define uma rede virtual sobre a rede física (*overlay*) IP ou não, com a finalidade de obter uma rede onde os *peers* sejam capazes de interagir diretamente entre si, além de serem capazes de se auto-organizar, independentemente de suas redes de conectividade física.

Além das características descritas na seção 2.2.5 e de funcionalidades que visam dar maior escalabilidade e desempenho à rede JXTA, nesta seção será dada ênfase à topologia P2P hierárquica híbrida (baseada em *Super-Peer*). O JXTA está baseado em 2 *Super-Peers*, *Relay* e *Rendezvous*.

I. Rendezvous

O JXTA faz uso de um mecanismo universal de *binding* para recursos, chamado de *resolver* que permite executar as operações de busca em sistemas distribuídos. As principais funções do resolver são:

- Resolver nome de *peers* para endereço IP,
- "Associar" um *socket* a uma porta,
- Localizar um serviço via LDAP,
- Procurar um conteúdo em um sistema de arquivos distribuído (NFS)

Os *rendezvous* (Figura 2-7) são *super-peers* em uma rede JXTA responsáveis por prover um serviço de índices de recursos. Ou seja, propiciar a localização de *advertisements* de *peers*, *peergroups*, *pipes*, e serviços.

Durante a inicialização de um *peer* em uma rede P2P desenvolvida sobre o *framework* do JXTA, por exemplo, este *peer* poderá utilizar-se dos serviços do *super-peer rendezvous* para registrar seus recursos e para localizar recursos.

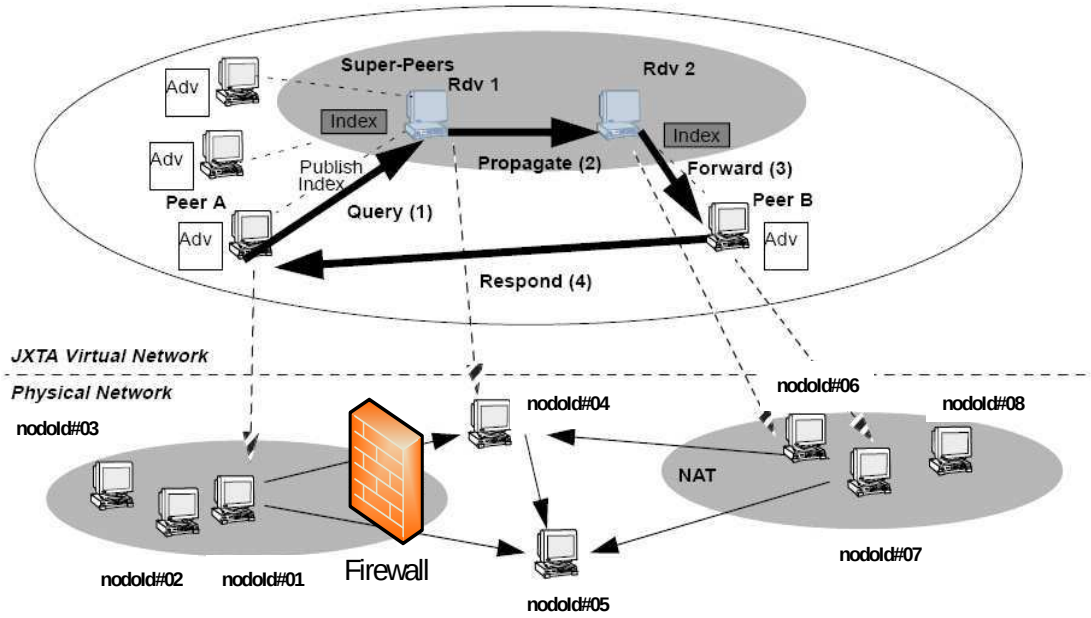

Figura 2-7: JXTA 2.0 - *Super-Peer Rendezvous(Rdv#n)* [TRAV2003]

Os *super-peers rendezvous* fazem a propagação (*propagate*, Figura 2-7) de seus índices de recursos para os demais *rendezvous* da rede através de algoritmos que podem usar passos de + 1 ou – 1, como mostra a Figura 2-8 e que será detalhado na seção 2.2.6. .

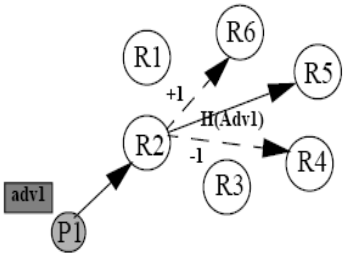

Figura 2-8: Publicação de um recurso (*advertisements*) [TRAV2003]

A propagação dos índices de recursos é realizada para facilitar a procura de um índice de recurso (otimizando o procedimento de localização recursos), além de replicar as informações (tabela de chaves de recursos) armazenadas em um *rendezvous*, garantindo assim, que com a queda de um *super-peer*, as suas informações sejam protegidas através de replicação.

II. Relay

O papel básico do *relay* () é prover o mapeamento entre *peers* localizados em redes distintas (mapeamento de portas inter-domínios), tornando transparente a conectividade entre

estes *peers* para o solicitante da conexão.

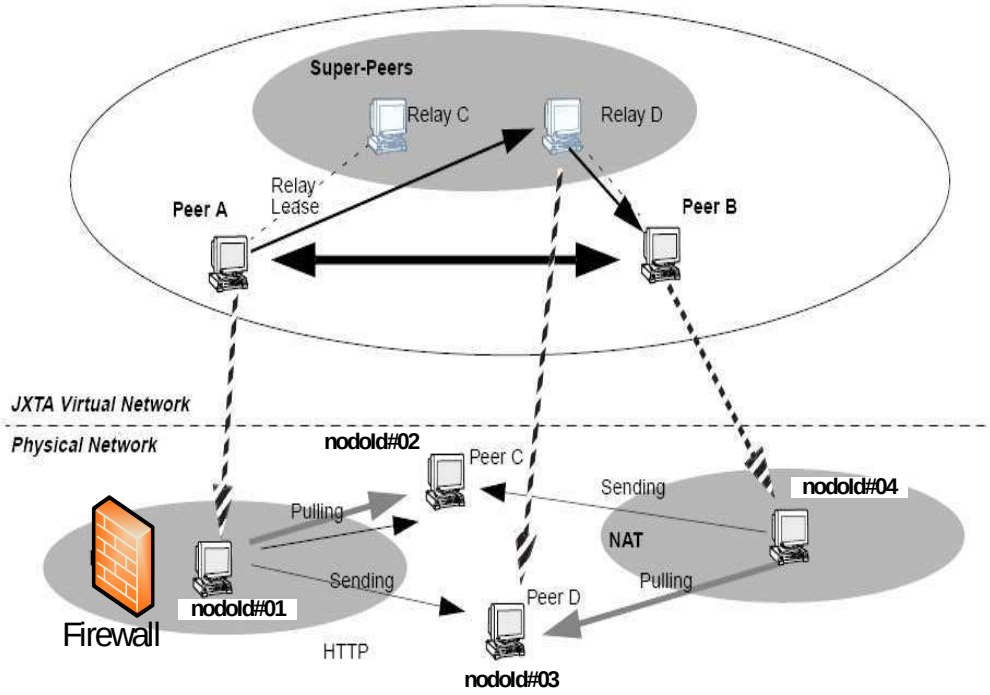

Figura 2-9: JXTA 2.0 - *Super-Peer Relay* [TRAV2003]

Entre estes domínios distintos pode existir um *Firewall* ou mecanismo de NAT (*Network Address Translation*).

A ilustra a busca de um recurso requisitado pelo *Peer* A e localizado no *Peer* B, através do *Relay* D. A Figura 2-10 destaca o mapeamento que o *Relay* D realiza na rede *overlay* fornecida pelo JXTA para que o *Peer* A se conecte ao *Peer* B de modo transparente.

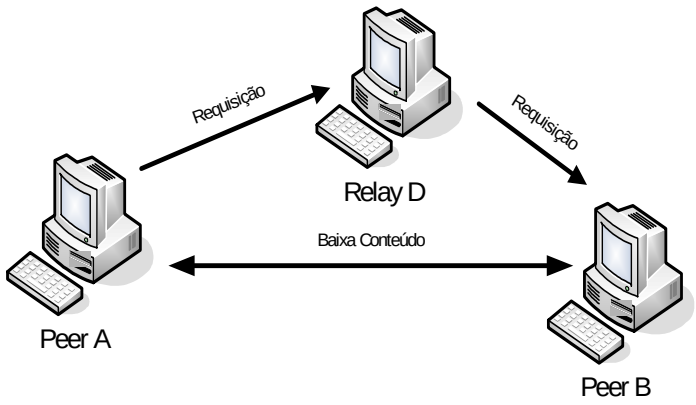

Figura 2-10: Busca de Recurso através do *Super-Peer Relay* [TRAV2003]

2.2.6. Project JXTA: *A Loosely-Consistent DHT Rendezvous Walker*

Como visto na seção 2.2.5, o projeto JXTA 2.0 adiciona o *super-peer rendezvous* para prover o armazenamento, propagação e, consequentemente, replicação de índices de recursos, para auxiliar os *peers* na publicação e localização dos recursos disponíveis na rede P2P. Em [TRAVWALK] é descrita uma técnica de replicação e busca de índices no projeto JXTA, que combina uma abordagem DHT com um *limited range Walker*.

Antes de apresentar a descrição do funcionamento de um *Loosely-Consistent DHT Rendezvous Walker*, é importante apresentar duas nomenclaturas:

- ***Rendezvous Peer View* (RPV)**: é uma lista dos *rendezvous* conhecidos deste *rendezvous* ordenado por seus peerId mantida por cada *rendezvous peer*;
- ***Peer-Group***: prove um escopo para propagação de mensagens em um agrupamento lógico de *peers*. No JXTA, todo *peer* é membro de um grupo *default* (*NetPeerGroup*), mas um mesmo *peer* pode pertencer a vários *peer-groups* ao mesmo tempo.

A abordagem DHT proporciona o mecanismo mais eficiente para acesso a índice de recurso e, consequentemente, leva mais rapidamente ao próprio recurso, devido aos algoritmos de acesso a DHT que reduzem o número de *peers* a serem visitados para encontrar o recurso desejado. A ausência da DHT resulta na necessidade de implementação de mecanismos equivalentes para otimização do processo de localização de recursos ou então a utilização da técnica de requisição por "inundação" (*flood request*), por exemplo, que se destaca pelo grande fluxo de mensagens, devido a propagação das requisições entre os *peers* de uma rede P2P.

Em [TRAVWALK] é proposta uma solução híbrida combinando o uso da abordagem *loosely-consistent* DHT com *rendezvous walker* de alcance limitado (*limited range walker*).

Se a *Rendezvous Peer View* se mantiver sincronizada, o *loosely-consistent* DHT será sincronizado também, resultando em um bom desempenho nas buscas.

Um *rendezvous* pode ter temporariamente ou permanentemente um RPV inconsistente e ainda pode não saber da existência de outros *rendezvous* na rede. No entanto, existem algoritmos destinados à sincronização de *rendezvous*.

Na Figura 2-11 (1) está ilustrada a publicação de um *advertisement* pelo *peer* (P1) no *rendezvous* R2. Através da função H(Adv1) é calculado e identificado o *rendezvous* onde deverá ocorrer a replicação da publicação (R5). Para aumentar a probabilidade de sucesso na

busca deste recurso publicado, o *rendezvouz* R2 replicará o índice desta publicação aos *peers* vizinhos de R5 (+1/- 1). Ou seja, os *rendezvous* R4 (-1) e R6 (+1) também receberão a replicação da informação. Com isso, é aumentada a região onde o índice publicado pode ser localizado.

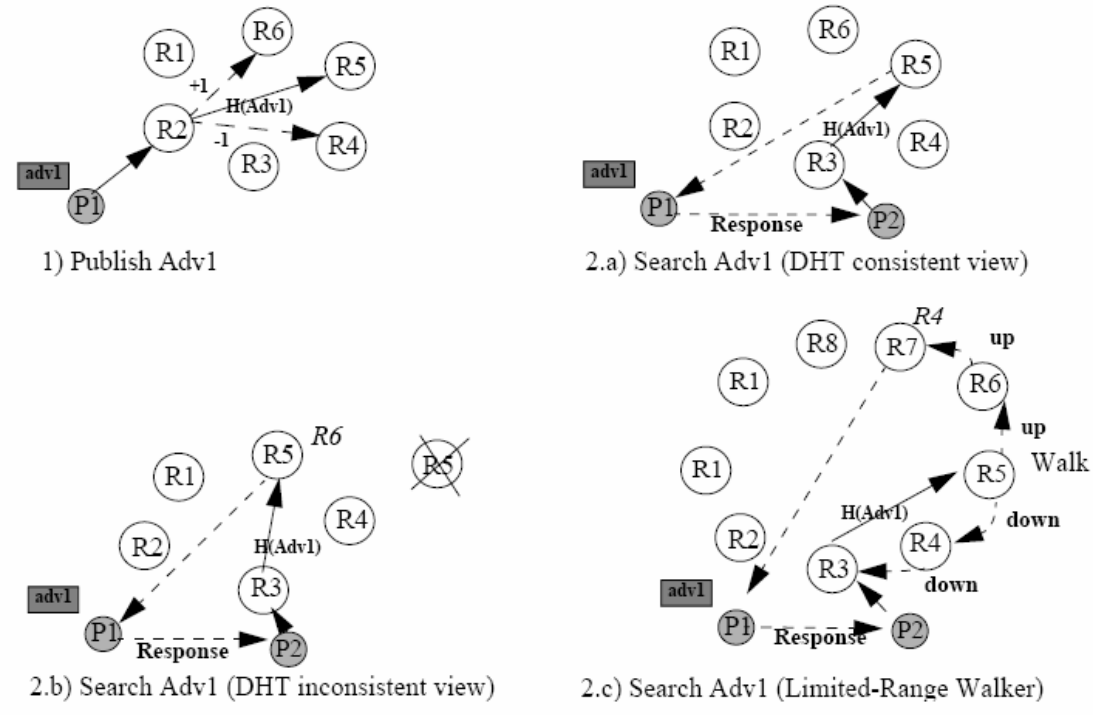

Figura 2-11: *Loosely-consistent* DHT [TRAVWALK]

A Figura 2-11 (2.a) ilustra um processo de busca, onde o *peer* P2 solicita ao *rendezvous* R3 a busca pelo recurso Adv1; através da função de *Hash* H(Adv1) é encontrado o índice do recurso no *rendezvous* R5, indicando que o recurso se encontra em P1. Este é a ilustração de uma busca com uma DHT consistente.

A Figura 2-11 (2.b) mostra uma busca com DHT inconsistente, *rendezvous* R5 não presente na rede. Ao realizar a busca, o *peer* P2 acessa o *rendezvous* R3, a partir da função de *Hash* H(Adv1) é calculado o identificador do *rendevous* que contém o índice do recurso publicado, neste estudo de caso R5(que está fora da rede). Como é sabido pelo sistema que a replicação da informação ocorre com um passo de +1/- 1, então o *rendezvous* R3 consegue chegar a R6, recuperando a informação desejada.

Na Figura 2-11 (2. c) é mostrada uma busca com DHT e *limited-range walker*. O RPV do *rendezvous* R3 é composto de 8 *rendezouz* (R1 a R8). Quando o *peer* P2 busca no R3 um dado recurso, a partir da função de Hash H(Adv1) o *rendezvous* R5 é indicado como

repositório de índice. Devido a uma reorganização proveniente de um balanceamento de carga, por exemplo, a informação não está mais lá, neste caso um mecanismo alternativo será usado para "andar" (*walk*) entre os *rendezvous* do RPV. Um *default limited-range walker* é usado para andar entre os *rendezvous* do RPV. Este algoritmo executa uma seqüência de movimento UP e DOWN consecutivos, a partir do identificador do *rendezvous*, resultante da operação H(Adv1), até que o recurso seja encontrado. Como ilustra a Figura 2-11 (2. c), o índice do recurso está armazenado no *rendezvous* R7.

2.3. MANet

Como conseqüência dos avanços significativos na computação móvel e na tecnologia de comunicação sem fio, os dispositivos móveis ganharam maior capacidade de comunicação, computação, e recursos de memória para serem interconectados.

Por definição, MANets (*mobile ad hoc networks*) diferenciam-se das redes existentes, pelo fato de não serem baseadas em uma infra-estrutura fixa [MERW2007], onde as redes não possuem estação base, pontos de acessos, servidores remotos. Todas as funções da rede são executadas pelos nós que compõem esta rede. Cada nó de uma MANet executa a funcionalidade de *host* e roteador, retransmitindo dados para estabelecer a conectividade entre os nós fonte e destino que não se encontram diretamente dentro de uma mesma área de transmissão, ou seja, estes nós intermediários vão retransmitindo a informação, até que ela encontre o seu destino ,conforme ilustra a Figura 2-12. Esta característica faz com que redes *ad hoc* sejam financeiras viáveis, pois não necessitam de qualquer custo relacionado com a criação ou manutenção de uma arquitetura de rede fixa (infra-estrutura fixa).

MANets são redes autônomas, multi-pontos interconectadas através de ligações sem fio. A palavra *ad hoc* (traduzido do Latim) significa que a rede é formada de maneira espontânea para atender uma demanda imediata e um objetivo específico. MANets tem uma topologia de rede dinâmica. Desde que os nós nas redes são móveis, com movimento irrestrito, a configuração da topologia de rede pode mudar muito rapidamente. A posição dos nós, relativos a outros, podem mudar de forma aleatória, o que resulta na imprevisibilidade das vizinhanças nesta modalidade de redes. A falta da infra-estrutura, topologia de rede dinâmica, e conectividade sem fio resulta nas freqüentes rupturas de "conexão" (conexão lógica), implicando na conectividade esporádica. Com isso, os protocolos para MANets devem ser capazes de minimizar estes efeitos em ambientes completamente distribuídos e

através da auto-organização.

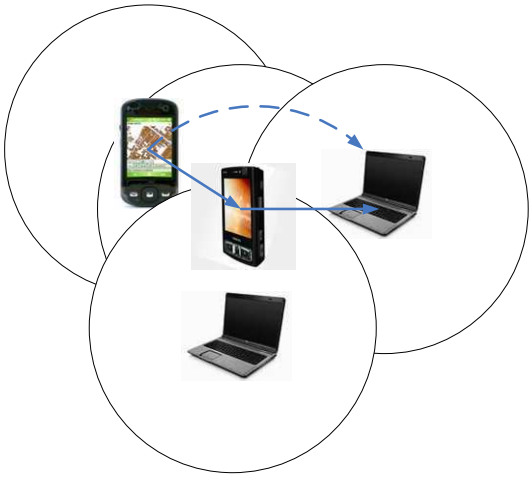

Figura 2-12: Comunicação em redes *Ad Hoc*.

Do ponto de vista de segurança, considerando as características originais de redes *ad hoc*, espera-se que os mecanismos propostos para garantir a segurança de redes convencionais do cabo não são necessariamente apropriados ou adaptáveis a MANets. Os mecanismos e os protocolos devem ser especialmente projetados para redes *ad hoc*.

2.3.1. Histórico

O conceito de uma rede *ad hoc* data do início da década de 70, quando a *U.S DARPA (United States Defense Advanced Research Projects Agency)* iniciou o projeto *PRNET (Packet Radio Network)*, para explorar o uso de redes de pacote de rádio, num ambiente tático, para comunicação de dados. Mais tarde, em 1983, a DARPA lançou o programa *SURAN (Survivable Adaptive Network)* para expandir a tecnologia desenvolvida no projeto PRNET a fim de suportar grandes redes e para desenvolver protocolos de rede adaptativos os quais pudessem se adaptar às rápidas mudanças de condições em um ambiente tático. O último da série dos programas iniciados pela DARPA para satisfazer os requisitos de defesa para sistemas de informações robustos e rapidamente expansíveis, foi o *GloMo (Global Mobile Information Systems)*, que teve início em 1994. Enquanto as comunicações táticas militares permaneciam a principal aplicação das redes MANets, a comunidade científica passou a utilizar esta tecnologia para fins não militares.

2.3.2. Áreas de aplicação das redes MANet

Além da área militar que deu origem a esta tecnologia, devido a facilidade e baixo custo para a criação de redes baseadas no modelo MANet, estas redes podem ser empregadas, praticamente, em qualquer área. As redes de sensores constituem redes MANet, basicamente utilizadas para mapeamento de ambientes, onde cada nó é responsável pelo mapeamento de uma determinada área. Depois de coletados, os dados são retransmitidos até uma unidade central de processamento, onde são realizadas análises, a fim de obter os resultados mais precisos, em busca da melhor solução para cada situação. Algumas aplicações onde este tipo de rede pode ser empregada: vigilância, monitoração de ambientes, análises geográficas, entre outras. Como rede pessoal, a concepção deste tipo de rede é simples. O seu principal objetivo é fazer com que diferentes dispositivos como notebooks, celulares, palmtops, e todos os demais que disponham de uma interface para a conexão, possam comunicar-se, sem depender de um meio físico para a conexão entre eles.

2.3.3. Características de Redes *Ad Hoc* Móveis (MANets)

É importante conhecer algumas propriedades que caracterizam uma MANet, de modo a entender esta tecnologia e permitir que seja usada mais eficientemente. Entre as características, destacam-se:

I. **Infra-estrutura de rede:** não existe uma infra-estrutura fixa ou pré-existente; todos os nós de uma rede *ad hoc* executam as funções típicas de rede: roteamento de pacotes, segurança, gerenciamento de rede e outros. Como a área de transmissão de dados de um nó é limitado, é fundamental que os nós sejam capazes de retransmitir estes pacotes, de modo que eles alcancem o seu destino. A inexistência de uma infraestrutura transforma cada um dos nós em um ponto de ataque, desta forma, cada um destes nós deve ser capaz de defender-se de ataque, através da implementação de mecanismos de segurança;

II. **Topologia de rede:** Os nós em redes *ad hoc* podem ser móveis, tendo por resultado uma topologia dinâmica, fracamente conectada. Já que a mobilidade do nó é irrestrita, a topologia será imprevisível. A topologia é fracamente conectada, devido a conectividade transiente, sujeita a erros, sem fio. Os usuários podem, com freqüência, se deparar com indisponibilidade de serviços essenciais, providos por esta rede. A mobilidade do nó e a

conectividade sem fio permitem que os nós, espontaneamente, se juntem e saiam da rede, caracterizando o *churn* da rede;

III. **Auto-organização:** devido a mobilidade dos nós e dinamicidade com que eles saem e juntan-se a redes MANet, estas redes devem ser capazes de auto-organizar-se, de modo que os serviços oferecidos pela mesma, não sofram grandes impactos e que a indisponibilidade dos serviços sejam as menores possíveis (imperceptíveis para os usuários). Neste momento, a rede fica mais suscetível ao ataque de nós maliciosos;

IV. **Recursos limitados:** esta característica já teve mais impacto em MANet, devido a baixa capacidade de processamento, armazenamento e energia dos dispositivos móvies do passado, mas que no entanto, com as crescentes evoluções da tecnologia móvel, estes limites estão sendo cada vez menos impactantes, pois a capacidade de armazenamento e processamento tem aumentado significativamente. Assim,, estudos têm feito com que dispositivos móveis tenham cada vez mais autonomia de uso, devido a fontes de energia cada vez mais duradouras;

V. **Segurança física pobre:** Os nós são móveis e, consequentemente, não podem ser guardados em quarto ou em um armário seguro. Estes dispositivos móveis são pequenos e devido a isto, podem ser facilmente perdidos ou roubados.Por isso, é muito provável que um adversário pode fisicamente comprometer um ou vários nós e executar todo o número de testes e de análise. O adversário pode igualmente usar os nós para atacar serviços de rede distribuída. A captura de dispositivos móveis (por perda ou roubo) não é a única forma de invadir uma rede *ad hoc*. Devido a comunicação sem fio, estas redes podem ser facilmente ”ouvidas” e decifradas por dispositivos alheios à rede;

VI. **Meio físico compartilhado:** O meio de comunicação sem fio é acessível a toda a entidade, com o equipamento apropriado e os recursos adequados. O acesso ao canal de comunicação não pode ser restrito. Os adversários podem, consequentemente, “ouvir” comunicações e injetar mensagens falsas na rede, sem limitação. O canal de comunicação compartilhado e os nós com segurança física pobre destacam, outra vez, que os mecanismos de segurança devem estar aptos a impedir e detectar estes ataques.

VII. **Sistemas Distribuídos:** Considerando as propriedades acima, os nós em redes *ad hoc* têm um relacionamento simétrico. Isto implica que são todos iguais e devem, consequentemente, distribuir igualmente todas as responsabilidades de fornecer a funcionalidade da rede. Isto não é somente por motivos de segurança, mas para permitir que os recursos e serviços de rede disponíveis tenham sua carga computacional igualmente distribuídos entre os nós participantes da rede.

2.3.4. Gerenciamento de chaves em MANets

Uma das formas mais eficientes de garantir a segurança para redes onde a inexistência de servidores centrais e de uma infraestrutura fixa ou pré-existente é o uso de criptografia, através do gerenciamento de chaves de segurança, dificultando assim, a ação de dispositivos maliciosos.

No entanto, em redes *ad hoc* o gerenciamento de chaves é um grande desafio. Os mecanismos tradicionais de gerenciamento de chaves não são adequados para as redes *ad hoc*, uma vez que necessitam de uma entidade confiável central. O principal problema de qualquer sistema de segurança baseado em chaves-públicas, é fazer com que a chave pública de cada usuário da rede seja disponibilizada para os demais usuários, de forma que sua autenticidade seja verificada. Esse problema é ainda maior nas MANets, uma vez que que não existe o papel de uma autoridade central na rede. Outra característica dos MANets é que eles podem ser particionados devido ao dinamismo de sua topologia.

2.3.5. Sistemas P2P Móveis

Os dispositivos móveis sempre foram considerados limitados, no que tange a transmissão de dados, se comparados aos computadores pessoais (PC) e entre estas limitações, destacavam-se:

- Quantidade de Memória
- Capacidade de Processamento
- Disponibilidade de Largura de Banda
- Autonomia das baterias
- Conectividade

Com o passar dos anos e a evolução tecnológica da área móvel, mais especificamente dos dispositivos celulares e periféricos, algumas destas limitações foram amenizadas e até mesmo eliminadas. Cartões de memória com capacidade de armazenamento cada vez maior e com menor tempo de acesso são disponibilizados no mercado com mais freqüência (por exemplo, cartões de memória de 4 GB são facilmente encontrados). A capacidade de processamento dos dispositivos móveis (PDA's, *pagers*, Celulares) que entre os quais se destaca o aparelho celular, bem como a evolução da tecnologia, no que diz respeito à conectividade e a expansão da cobertura, contribuiu para o aumento da comunicação de dados, via dispositivos celulares, sem se preocupar com as constantes perdas de conexão. A autonomia das baterias vem aumentando e com a chegada dos aparelhos de terceira geração, espera-se acabar de vez, com a limitação de banda.

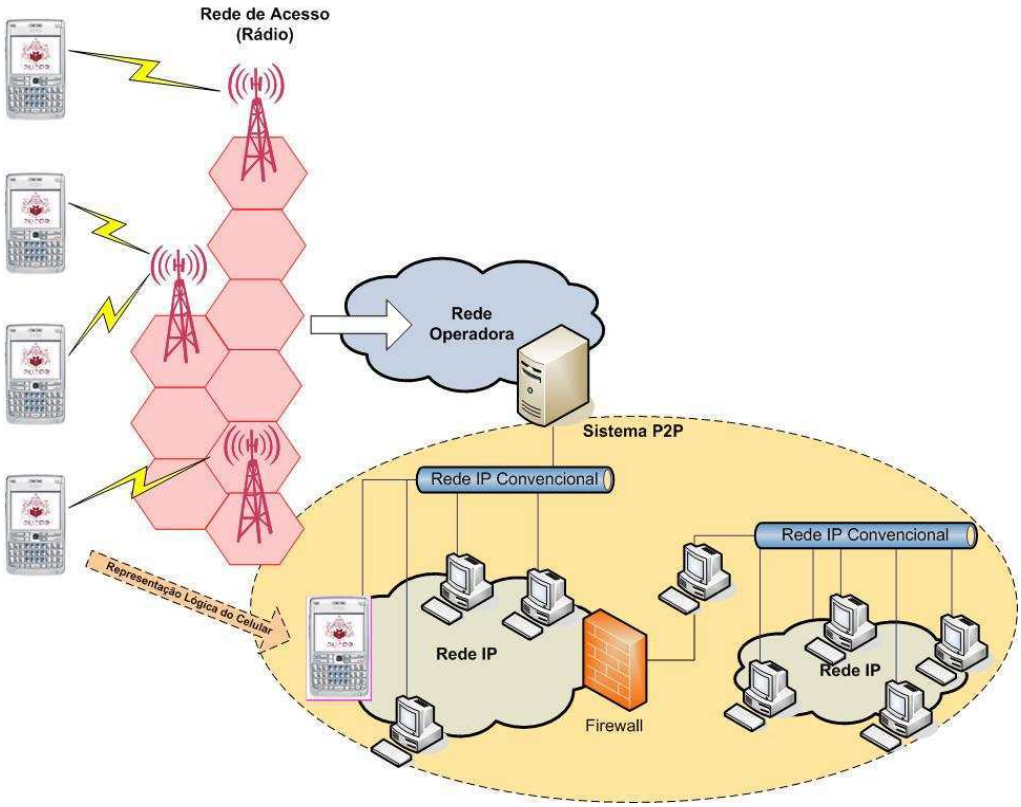

Figura 2-13: Interconexão entre um dispositivo móvel e *Internet* [OLIV2005]

As evoluções, acima citadas, contribuíram significativamente para a migração de aplicações *P2P* para os dispositivos móveis. A seguir, as características anteriormente apresentadas sobre as redes P2P, serão transportadas para o mundo das telecomunicações móveis, mais especificamente celulares, e para tanto, é importante entender como ocorre a

transmissão de dados neste contexto. A ilustra a conexão de um dispositivo celular com a *internet*.

Abstraindo a complexidade de interconexão de um dispositivo móvel com uma rede de computadores (*Internet*, por exemplo), que é provida por uma operadora de telefonia celular, a qual o dispositivo está conectado, temos a representação destacada pela elipse na .

Uma das características destacada durante a fundamentação do P2P, foi o *overlay*, com o qual um sistema *P2P* é capaz de criar uma camada de abstração, na qual todos os *peers* são tratados como entidades com características similares, independente de tecnologia.

Este nível de abstração é obtido em redes *P2P*, através de *frameworks* como o **JXTA,** cuja ilustração desta camada de abstração ou virtual, é mostrada na Figura 2-14.

Fazendo uso desta característica típica de redes *P2P*, o *overlay*, é possível abstrair a origem da tecnologia do elemento da rede, tratando-o apenas como um *peer* da rede.

A seguir, serão apresentadas as implementações de bibliotecas como J2ME e JXME, que aparecem como especializações de soluções da SUN para JAVA e mais especificamente JXTA, respectivamente.

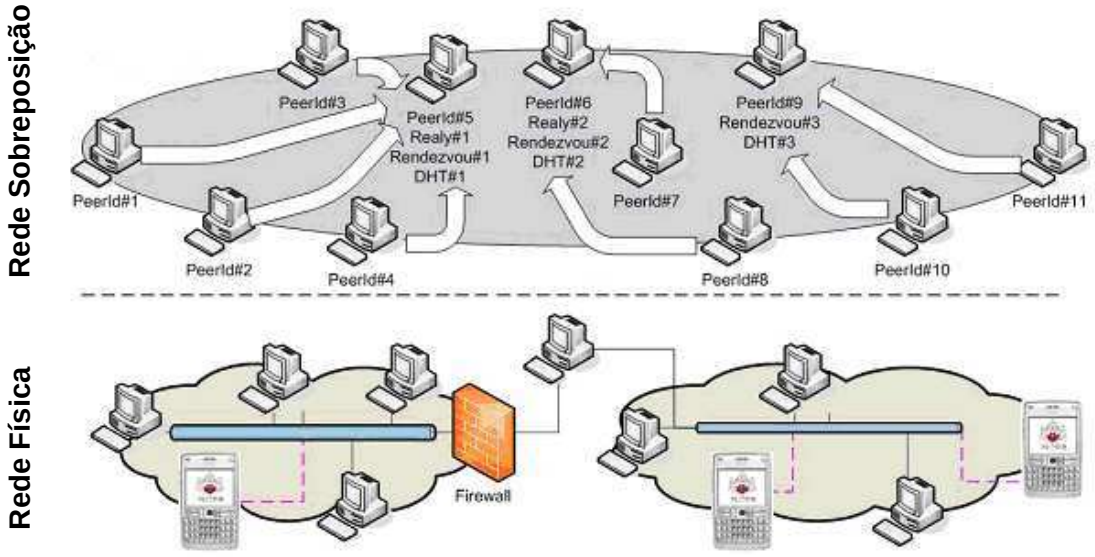

Figura 2-14: Uso do JXTA para sistemas P2P [TRAV2002]

2.3.6. Plataforma J2ME

Dispositivos móveis como celulares, PDA's, *pagers*, dentre outros, têm sido utilizados, comumente, para realizar tarefas que no passado, seriam de difícil execução,

principalmente em decorrência das limitações que estes dispositivos apresentavam.

Com a evolução tecnológica dos últimos anos, as aplicações destinadas a este tipo de dispositivos podem ser implementadas por uma ampla gama de tecnologias e a escolha sobre qual delas utilizar, dependerá, muitas vezes, da finalidade proposta pela aplicação e também de requisitos arquiteturais e temporais que deverão ser respeitados.

A plataforma J2ME (JAVA 2 Micro *Edition*) é uma das possíveis tecnologias que podem ser utilizadas para desenvolvimento de aplicações para vários pequenos dispositivos, como telefones celulares, *pagers*, etc.

Como destacado anteriormente, as aplicações *P2P* estão cada vez mais presentes nesta modalidade de dispositivos, principalmente com as facilidades fornecidas pelas plataformas J2ME e JXTA.

Uma aplicação J2ME deve ser projetada para uma arquitetura predefinida pela própria plataforma J2ME, na qual a camada mais básica da arquitetura (ver Figura 2-15) encontra-se o componente chamado MID (Dispositivo de Informação Móvel). Esse componente representa o hardware propriamente dito, ou seja, o dispositivo onde será implantada a aplicação. Esse dispositivo pode variar de acordo com a finalidade da aplicação que está sendo desenvolvida.

Na camada acima, encontra-se o sistema operacional nativo. Esse sistema operacional vem embutido no dispositivo desde a sua fabricação (*firmware*), mas que pode ser substituído, posteriormente, pelos usuários deste dispositivo. Dentre os mais conhecidos sistemas operacionais para pequenos dispositivos, estão o *Microsoft Windows CE* e o *Symbian OS*.

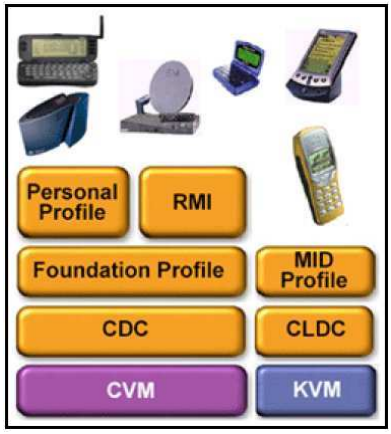

Figura 2-15: Plataformas J2ME

Acima da camada que representa o KVM (Kilo Virtual Machine), ilustrado pela Figura 2-15, tem-se o CLDC (Configuração de Dispositivo Conectado Limitado). O CLDC

inserido no contexto de uma aplicação J2ME, representa uma configuração que classifica determinados dispositivos como pertencentes a uma classe. Como exemplo de configuração, tem-se no CLDC dispositivos que recebem cargas elétricas, através de baterias. Já os dispositivos que pertencem à configuração CDC, possuem, como uma de suas características, conexão de rede.

O MIDP (Perfil de Dispositivo de Informação Móvel) é o componente que, em termos de software, representa as limitações impostas por uma configuração. Em outras palavras, o MIDP é uma API contendo as classes Java que satisfazem os critérios de implementação descritos pela especificação do CLDC.

A Figura 2-15 apresenta a arquitetura da plataforma J2ME para as configurações CDC e CLDC, lado a lado, permitindo que o leitor possa comparar os componentes de ambas as plataforma.

Porém, o J2ME aplicado para P2P, sobre a plataforma JXTA é conhecido como JXME que será mostrado a seguir.

2.3.7. JXME

Resumidamente, JXME é uma implementação do JXTA destinada a dispositivos móveis (J2ME), cujo uso da API **JXTA4J2ME** é fundamental.

Entre os objetivos do JXME, destam-se:

- Interoperabilidade com outras implementações de JXTA,
- Provimento de estrutura P2P para dispositivos móveis,
- Atendimento das restrições de tamanho comum em dispositivos móveis,
- Compatibilidade com o MIDP;

Devido às limitações dos dispositivos móveis e do MIDP, os *peers* com implementação do JXME não podem desempenhar funções sofisticadas, ou seja, não podem, por exemplo, rotear informações para outros *peers* e até mesmo, oferecer serviços a outros membros de um *peer group*. Este tipo de comportamento é denominado de *peer* mínimo (ou *minimal peer*). Com isso, todo o processamento mais pesado deve ser feito em outros *peers* da rede. Estas tarefas, são em sua grande maioria, assumidas pelos *Relays Peers*.

Os *Relays Peers* desempenham o papel de *Proxy* para os dispositivos móveis,

assumindo a maior parte das tarefas fundamentais de uma rede *P2P*, no que tange a interoperabilidade entre os dispositivos móveis, é o resto da rede. As tarefas mais comuns de um *Relay Peer*, são:

- Prover a interoperabilidade com os protocolos do JXTA
- Atuar como *Proxy* na:
    - criação de grupos e *pipes*,
    - descoberta de *peers*,
- Filtrar o tráfego JXTA (pré-processar mensagens XML)
- Otimizar os *advertisements*;

Com todo este relacionamento entre um *peer* móvel e o *Relay*, não existe uma descaracterização da filosofia de uma rede *P2P*, devido a natureza das conexões entre eles. Não existe a necessidade de estabelecer ou manter uma conexão estática, como ocorre no modelo cliente/servidor. O *peer* móvel pode trocar ou usar múltiplos *Relays* dinamicamente.

A plataforma MIDP, cuja representação modular é ilustrada na Figura 2-15, não fornece suporte ao processamento de arquivos XML, no caso da versão para celulares. Além disso, enquanto é processado, um arquivo XML precisa ser mantido na memória. Em dispositivos com limitação de memória, não há como processar arquivos XML diretamente, por isso, o *Relay* precisa filtrar as mensagens XML e as redireciona ao dispositivo móvel num novo formato binário (usando o protocolo HTTP). Com isso, o volume de tráfico na rede é reduzido.

I. API

JXME oferece uma API simples, com o mínimo de serviços, para que um dispositivo móvel possa interagir com o *Relay* e, consequentemente, com uma rede P2P. Esta API possui apenas três classes:

- ***Message*:** métodos para criar e manipular mensagens JXTA,
- ***Element*:** métodos para construir e manipular componentes básicos das mensagens de JXTA,
- ***PeerNetwork*:** mecanismos para interagir com uma rede JXTA;

Apesar de reduzida, a API apresenta um razoável conjunto de operações:

- **Descoberta de *Pipes*:** procurar e manter uma lista limitada de *pipes*,
- **Descoberta de Grupos:** procurar *peer groups* e de se associar a eles,
- **Descoberta de *Peers*:** descobrir outros *peers*,
- **Criação de *Pipes*:** criar *pipes* ponto a ponto e *propagate pipes*,
- **Criação de Grupos:** criar *peer groups*,
- **Comunicação:** trocar mensagens com outros *peers*,

Na Figura 2-16 é possível verificar os componentes da arquitetura JXME baseada em proxy, cuja API e demais informações foram apresentadas acima.

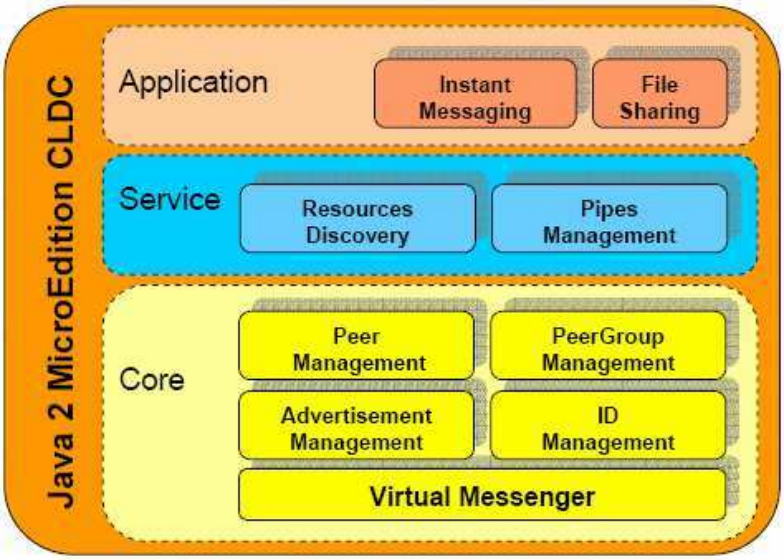

Figura 2-16 : Arquitetura JXTA baseada em proxy

II. JXME PROXYLESS

A versão *proxyless* de JXME (lançada em 2005) está focada em dispositivos móveis, com um poder de processamento maior, como por exemplo, palms e até mesmo celulares. Essa versão provê manipulação das mensagens XML, roteamento de mensagens e uso do protocolo TCP para o transporte da informação. Com esta versão do JXME, como sugere o nome, o papel do *Relay Peer* como *proxy* para os *peers* móveis, tornou-se desnecessário.

A Figura 2-17 mostra uma arquitetura de uma aplicação P2P que faz uso do JXME para a implementação de um *peer* móvel, que é detalhado em [MB102005] e que é, basicamente, dividida em 3 partes: *Application*, *Service* e *Core*, onde o *Core* é subdividido em:

- *Peer Management:* responsável pela identificação e gerenciamento de um *peer*,

- *Peer Group Management:* responsável pela identificação e gerenciamento dos grupos a que um *peer* pertence,
- *Advertisement Management* : gerenciamento das funcionalidades relacionadas aos anúncios de recursos de um *peer*,
- *ID Management*,
- *Virtual Messenger*: compreende os protocolos de transporte e serviço e que é responsável pelo gerenciamento da comunicação do *peer* com o resto da rede;

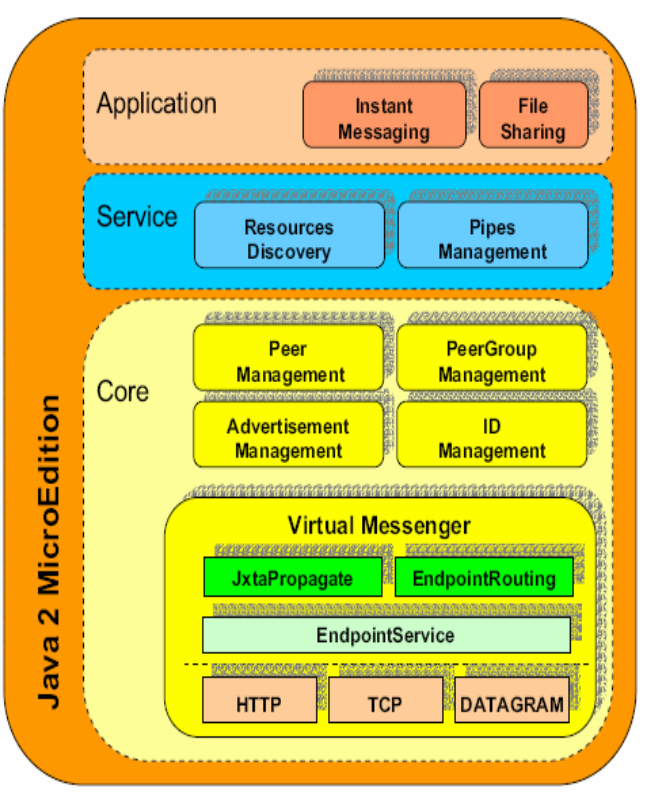

Figura 2-17 : Arquitetura JXTA sobre J2ME [MB102005]

O módulo de serviço é responsável pelas funcionalidades de *discovery* e pelo gerenciamento dos *pipes*, que nada mais são do que os canais de entrada e de saída de um *peer*, através dos quais, os *peers* conectam-se com outros *peers*.

A camada de aplicação, por sua vez, é onde esta proposta atuará, modelando e implementando a aplicação que acessará o serviço de busca com alta disponibilidade e garantia de integridade e autenticidade dos recursos que é o grande objetivo deste projeto.

2.4. Conclusão

Neste capítulo foram apresentados os conceitos fundamentais para o entendimento dos próximos capítulos, ou seja, todos os assuntos abordados nesta seção serão utilizados como base conceitual para a proposta que será apresentada posteriormente em uma seção apropriada. O protótipo, por sua vez, utilizará além dos conceitos, tecnologias como o JXTA e DHT para seu desenvolvimento.

Em suma, as informações apresentadas neste capítulo são de grande importância para a compreensão do trabalho proposto.

Capítulo 3

Redes *Peer-to-peer*, *Overlay* em Ambiente MANet (PoM)

Neste capítulo, são apresentados os trabalhos encontrados na literatura técnica sobre o assunto, que estão fortemente relacionados com o assunto abordado pela proposta. A proposta leva em consideração as experiências positivas e negativas apresentadas por estes trabalhos, permitindo, desta forma, que fracassos do passado não sejam repetidos, mas ao mesmo tempo garantindo que os sucessos sejam usados como norteadores da proposta, ou até mesmo, reaproveitados. Oportunamente, serão destacados estes pontos que foram obtidos através dos trabalhos relacionados e que foram úteis para a composição desta proposta.

3.1. Avaliação de Desempenho do Chord (DHT) em MANet

Em [CRAM2006] é relatada uma experiência de uso do protocolo de busca do Chord sobre uma rede com infra-estrutura móvel *ad hoc* produzida a partir de simuladores.

Segundo [CRAM2006] há um consenso entre os autores de literatura sobre o tema, que P2P estruturado não é eficiente em MANet, porém [CRAM2006] realiza experimentos com o objetivo de obter evidências que contradigam este suposto consenso.

- Na proposta são adotadas as restrições de projeto para a rede *ad hoc*:
- Inexistência de infraestrutura,
- Descentralização,
- Configuração (ou reconfiguração) autônoma e rápida,
- Limitação de banda,

- Canais sem fio não confiáveis,
- Mobilidade dos nodos;

O objetivo principal da simulação é identificar as limitações de desempenho, colocando um protocolo padrão do P2P sobre um protocolo de roteamento MANet. Para este fim, foi implementado e avaliado um Chord em um simulador de MANet (GloMoSim [BAJA1999]). Mais especificamente, o interesse dos autores foi avaliar a habilidade do Chord para resolver corretamente buscas sob uma variedade de condições (por exemplo: redes com tamanhos diferentes, grau de mobilidade dos nodos e uma quantia de tráfego da aplicação).

Todos os nodos usaram o IEEE 802.11 DFWMAC com extensão RTS/CTS e interface de rádio de 2.4 GHz com taxa de transmissão de 2Mbps. A propagação do rádio é bidirecional, com alcance de transmissão de 250m.

O cenário básico do experimento é composto de 50 nodos, uniformemente distribuídos em uma área de uma quadra (100m x 100m). Todos os nodos movem-se em direção aleatória, com velocidade entre 0 m/s e 2 m/s, para representar realistamente a velocidade de pedestres. Cada nodo faz requisições a cada 5 segundos (em média).

Antes de apresentar os resultados obtidos com o experimento, é importante comentar as métricas e critérios adotados. As métricas são as seguintes:

- ***Request success ratio***: representa a relação entre requisições realizadas e respostas recebidas, ou ainda, requisições realizadas por requisições bem sucedidas,
- ***Overlay consistency***: indica o percentual de nodos que selecionaram corretamente seu nodo successor, de acordo com um dado protocolo de roteamento,
- ***End-to-end delay***: denota o tempo entre uma requisição e uma reposta,
- ***Total Network load***: indica o total de loads realizados em uma rede;

Os critérios considerados no experimento são: variação da taxa de requisições, variação da mobilidade dos nodos e a variação do tamanho da rede.

Os resultados dos experimentos foram tabulados e representados em gráficos pelos autores [CRAM2006]. A seguir, serão comentados alguns aspectos relacionados com estes resultados.

Considerando a taxa de sucesso em requisições (buscas), o desempenho da técnica por

inundação mostrou-se mais eficiente em todos os critérios apresentados acima e comparados com as técnicas baseadas em Chord sobre vários protocolos de roteamento (por exemplo, AODV [PERK1999], DSR [John1996] e OLSR [JACQ2001]). Mesmo para o critério de variação do tamanho da rede, onde todos os critérios tiveram um desempenho da taxa de sucesso inversamente proporcional ao crescimento da rede, o desempenho da técnica por inundação foi melhor, conforme mostra a Figura 3-1.

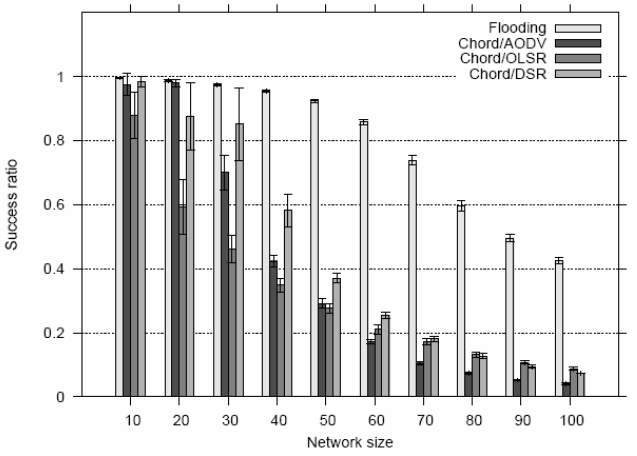

Figura 3-1: Taxa de sucesso de req. com a variação do tamanho da rede [CRAM2006]

Considerando o *delay* de uma requisição fim-a-fim, é importante ilustrar o desempenho das técnicas através de gráficos, pois não houve uma tendência, como no caso anterior. O gráfico da Figura 3-2 mostra que o *delay* entre requisições aumenta exponencialmente para a técnica por inundação, com o crescimento da rede, enquanto que para o Chord sobre os protocolos de roteamento, o crescimento é bem suave, tendendo à estabilização.

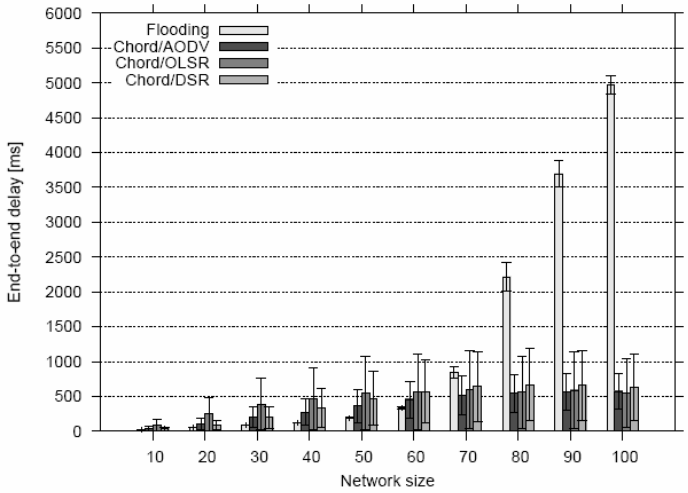

Figura 3-2: *Delay* de req. fim-a-fim com a variação do tamanho da rede [CRAM2006]

O gráfico da Figura 3-3, por sua vez, mostra que o *delay* entre requisições tem um valor altíssimo para a técnica por inundação para o intervalo entre requisições igual a 1s. No entanto, com o aumento do intervalo entre requisições, o *delay* diminui para a técnica por inundação e aumenta para o Chord.

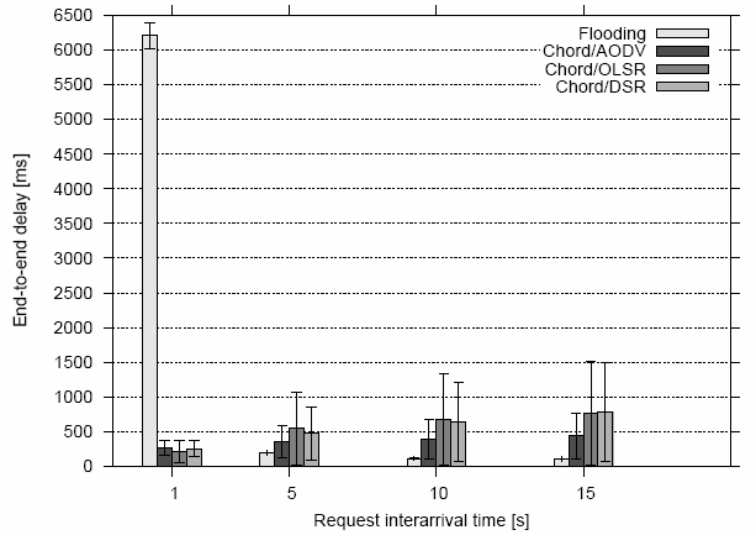

Figura 3-3: *Delay* de req. com a variação do intervalo entre requisições [CRAM2006]

Com relação à mobilidade, o Chord tem um desempenho pior se comparado com a técnica por inundação. Uma exceção é observada para a carga total da rede, conforme ilustra o gráfico da Figura 3-4 que mostra que a carga é maior e constante para a técnica por inundação.

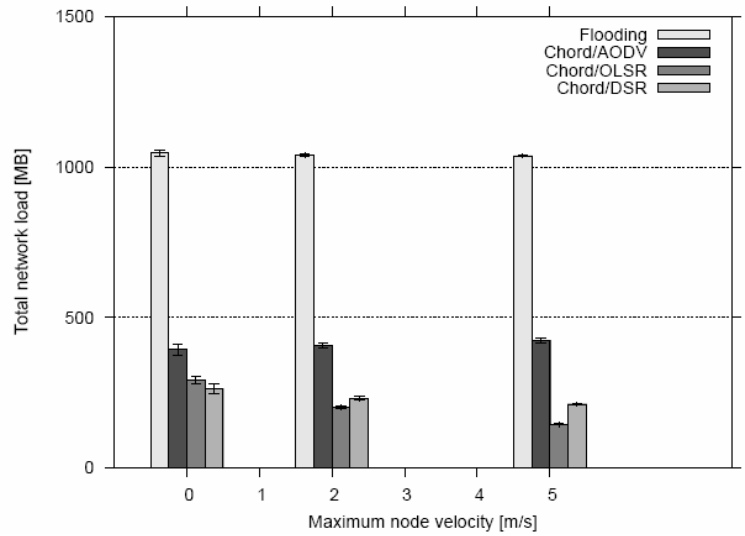

Figura 3-4: Carga total da rede variando com a velocidade dos nodos [CRAM2006]

No entanto, com relação ao grau de consistência, o Chord não possui bom desempenho para todos os critérios apresentados acima.

Na simulação, foi possível constatar que a habilidade do Chord em resolver buscas com consistência, foi prejudicada, devido a estratégia pessimista de tratamento de saída de um nodo da rede, que descarta, imediatamente, buscas por *timeout*. Com isto, buscas são inconsistentemente resolvidas, resultando no comportamento incorreto da aplicação. Na internet, a estratégia de saída de um nodo da rede é usada para detecção imediata de nodos que saíram da rede. Para minimizar os efeitos de inconsistências temporárias, objetos são replicados em múltiplos sucessores.

Com base nestas informações, o autor de [CRAM2006] conclui que ao contrário da opinião que prevalece na literatura, a baixa taxa das buscas com sucesso observado no Chord não é resultado do congestionamento relacionado com perda de pacote, mas sim, pela inconsistência causada pela perda de pacotes, devido à mobilidade ou erro de transmissão.

O crescimento do número de encaminhamentos com falha e, consequentemente, a taxa de invalidação de sucessores é diretamente proporcional ao número de requisições enviadas.

E por fim, inesperadamente, foi observado que a causa do congestionamento da rede não está diretamente relacionado com os problemas de desempenho.

3.2. *JMobiPeer: a middleware for mobile peer-to-peer computing in MANets*

Em [MB252005] é descrita a plataforma JMobiPeer para aplicações P2P considerando dispositivos móveis. JMobiPeer foi desenvolvida para trabalhar em J2ME e ambiente *ad hoc* (MANet).

As metas do projeto da plataforma JMobiPeer serão resumidas nos seguintes aspectos:

- superar os limites e restrições do JXME(*proxied*),
- ser compatível com os protocolos do JXTA,
- estar habilitado a trabalhar em ambiente MANet, mesmo quando desconectado de uma rede JXTA tradicional,
- executar em dispositivos com limites de recursos, como o J2ME está habilitado a fazer, para ambas as configurações: CLDC (*Connected Limited Device Configuration*) e CDC (*Connected Device Configuration*);

Uma visão geral da arquitetura do JMobiPeer é ilustrada através da Figura 3-5. A figura central da plataforma JMobiPeer é o módulo *Virtual Messenger* que pertence ao core

(núcleo) da arquitetura, juntamente com os módulos de gerenciamento de *Peer*, *PeerGroup*, *Advertisement* e *Discovery*. O módulo de *service* (serviços) implementa funcionalidades que são requisitadas pelas aplicações P2P, enquanto que o módulo *Applications* (aplicações) provê as aplicações comuns à rede P2P.

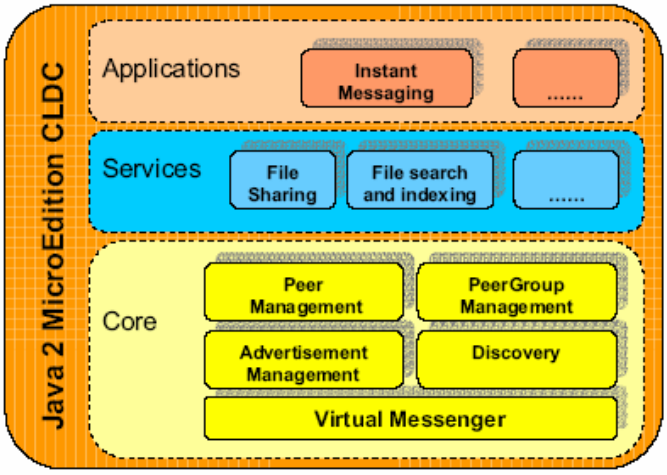

Figura 3-5: Arquitetura do JmobiPeer [MB252005]

Uma das principais funcionalidades da plataforma JMobiPeer é a interoperabilidade com a plataforma JXTA, porém, sem o uso do *Super-Peers* (*Relay*, do projeto JXTA 2.0). O desempenho desta plataforma (JMobiPeer) depende, basicamente, do custo de:

- uso de mensagens XML para manter a desejada interoperabilidade,
- procedimento de descoberta (*discovery*);

O ambiente de teste é composto por:

- 2 computadores pessoais (PC) equipados com um processador Pentium 2.4 GHz e 512 MB de RAM,
- Conectados através de uma interface e*thernet* de 100 MBit,
- Um terceiro PC usado apenas para coletar dados,
- Não foi adicionado tráfego à LAN durante o teste,
- Nos 2 PCs o *test-bed* foram instalados apenas os softwares necessários para a plataforma JMobiPeer e JXME *proxyless*;

Os passos para a fase de descoberta, durante os testes, foram:

1. O PeerA procura por um Adv1 (*advertisement* publicado pelo PeerB) através de uma mensagem de consulta em *broadcasting*,
2. O PeerB recebe a mensagem de consulta e envia uma resposta à consulta contendo o Adv1 para o PeerA,
3. O PeerA recebe a resposta à consulta e armazena o Adv1, eventualmente, fornecendo o Adv1 aos serviços superiores;

O tempo total de descoberta (*Total Discovery Time*) é medido como o tempo gasto desde o envio da mensagem de consulta (passo 1 Figura 3-6) até o recebimento da resposta à consulta (passo 3 Figura 3-6).

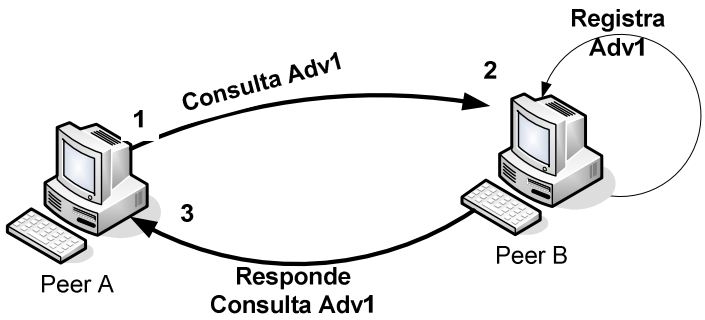

Figura 3-6: Passos para descoberta do Adv1 (*advertisement* 1)

Na Figura 3-7 é apresentado um gráfico com os resultados obtidos na realização dos testes, de acordo com as características citadas do ambiente de teste mencionado anteriormente.

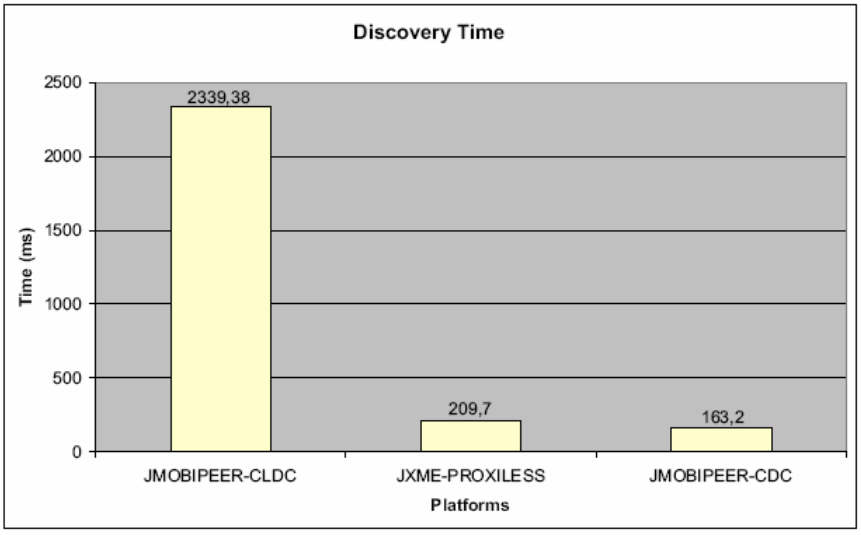

Figura 3-7: Tempo Total de Descoberta (TTD) - [MB252005]

Considerando o desempenho relacionado diretamente com o tempo total de descoberta, que compreende os passos relacionados acima, ficou evidente, no gráfico da Figura 3-7,, que o desempenho do JXME *proxyless* foi muito superior ao da plataforma JMobiPeer CLDC. No entanto, o desempenho da plataforma JMobiPeer CDC foi ligeiramente superior ao do JXME *proxyless*.

Os testes mostram que é possível obter uma plataforma que mantenha a interoperabilidade com os protocolos do JXTA, superando as restrições e limitações do JXME *proxyless* para ambiente de MANet. Além disto, JMobiPeer tem desempenho similar ao JXME *proxyless* .

3.3. *Expeerience: a JXTA middleware for mobile ad hoc networks* [MBIS2003]

O trabalho [BISIG2003] propõe a implementação de uma plataforma cujo objetivo é eliminar a instabilidade de conexão para os desenvolvedores de aplicações P2P. A implementação desta plataforma exigiu mudanças no núcleo de JXTA, conforme ilustra a Figura 3-8, para incluir funcionalidades como a gerência de conexões intermitentes e de múltiplas interfaces físicas. Estas mudanças aumentaram o potencial da descoberta do recurso.

A Figura 3-8 ilustra os pontos do core do JXTA onde foram necessárias as alteações para minimizar os efeitos da mobilidade característica das redes *ad hoc* e obter os ganhos desejados.

No módulo *SUN JXTA Services* foi adicionada a feature *DiscoveryService* para permitir facilitar (potencializar) a atividade de procura (descoberta) de recursos de uma rede P2P sobre *ad hoc*.

Ainda no módulo *SUN JXTA Services* o novo *EndPoint Service* será responsável por suportar múltiplos endereços.

No módulo JXTA Community Services, o MobilyCodeService é disponibilizado através de uma interface específica, através da qual a aplicacação acessa métodos implementados para gerenciar a mobilidade característica das redes *ad hoc*, de modo a minimizar o efeito da mesma, neste tipo de rede.

O novo serviço TcpTransport é responsável por gerenciar conexões intermitentes aos múltiplos endereços.

Com isso, a plataforma ***Expeerience*** minimiza os efeitos da mobilidade em redes P2P.

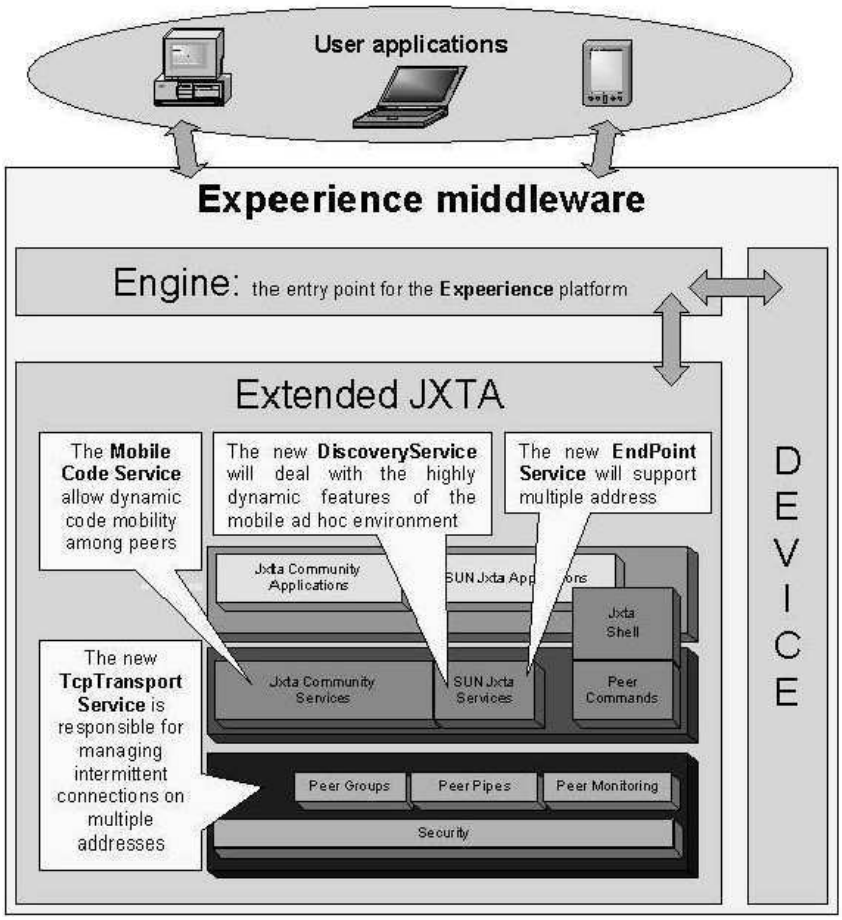

Figura 3-8: Arquitetura do *Expeerience* [BISIG2003]

3.4. *Garantia de Autenticidade e Integridade*

Como citado anteriormente, a grande quantidade de conteúdo poluído nas redes P2P tradicionais foram motivadores desta proposta e com o intuíto de resolver o problema da falta de autenticidade de conteúdo e publicação.

Em redes P2P, autenticidade de conteúdo é tipicamente obtida através de assinatura digital sem o uso de uma autoridade de certificação, e.g. certificados autoassinados, pois devemos lembrar que este tipo de rede não tem o papel de um servidor. Em criptografia e segurança de computador, um certificado autoassinado é um certificado de identidade que é assinado por seu próprio criador. Isto é, a entidade que criou o certificado também o assinou.

Para que este tema seja mais facilmente comprendido, é importante descrever o conceito de Assinatura Digital [SANT2004] . Entende-se por assinatura digital, a técnica usada para representar eletronicamente a assinatura física em papel. O modo usual de implementação de assinatura digital é a geração de um resumo/sumário que é cifrado

juntamente com a chave primária da origem da informação, gerando um dado adicional que é enviado juntamente com a mensagem assinada, conforme ilustra a Figura 3-9.

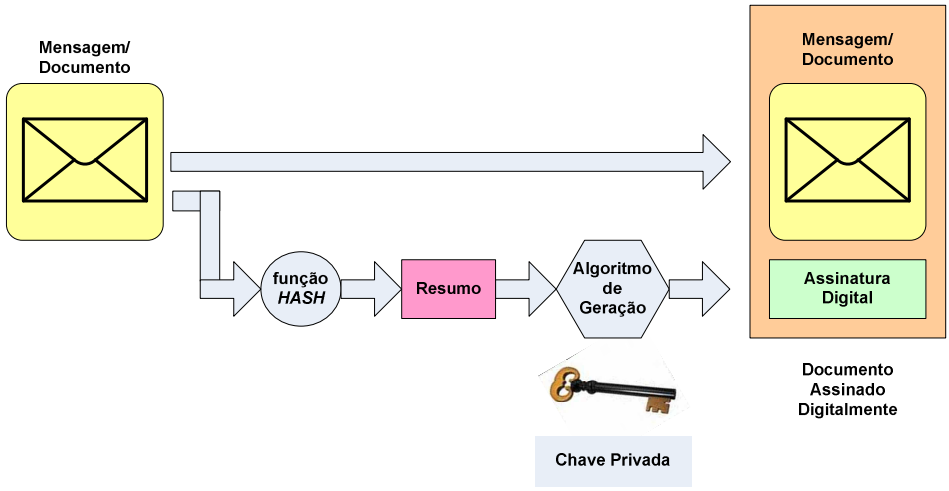

Figura 3-9: Gerando uma assinatura digital

O resumo é gerado a partir de uma função de *hash* H, dada por h = H (m), onde H retorna sempre uma string de tamanho fixo – h (normalmente de 128 ou 160 bits), independente do tamanho da mensagem (m). Como o espaço de conteúdo da mensagem m é bastante amplo é imperativo a escolha de um algoritmo de geração de sumário que empregue uma função de *hash* sem colisões possíveis.

Entre as funções *hash* mais utilizadas em produtos e protocolos criptográficos, cabe destacar:

- **MD5**: É uma função de espalhamento unidirecional inventada por Ron Rivest, do MIT, que também trabalha para a *RSA Data Security*. A sigla MD significa *Message Digest*. Este algoritmo produz um valor hash de 128 bits, para uma mensagem de entrada de tamanho arbitrário. Foi inicialmente proposto em 1991, após alguns ataques de criptoanálise terem sidos descobertos contra a função *Hashing* prévia de *Rivest*: a MD4. O algoritmo foi projetado para ser rápido, simples e seguro. Seus detalhes são públicos, e têm sido analisados pela comunidade de criptografia. Foi descoberta uma fraqueza em parte do MD5, mas até agora ela não afetou a segurança global do algoritmo. Entretanto, o fato dele produzir um valor hash de somente 128 bits é o que causa maior preocupação; é preferível uma função *Hashing* que produza um valor maior,
- **SHA-1**: O *Secure Hash Algorithm*, uma função de espalhamento unidirecional

inventada pela NSA, gera um valor hash de 160 bits, a partir de um tamanho arbitrário de mensagem. O funcionamento interno do SHA-1 é muito parecido com o observado no MD4, indicando que os estudiosos da NSA basearam-se no MD4 e fizeram melhorias em sua segurança. De fato, a fraqueza existente em parte do MD5, citada anteriormente, descoberta após o SHA-1 ter sido proposto, não ocorre no SHA-1. Atualmente, não há nenhum ataque de criptoanálise conhecido contra o SHA-1. Mesmo o ataque da força bruta torna-se impraticável, devido ao seu valor *hash* de 160 bits. Porém, não há provas de que, no futuro, alguém não possa descobrir como quebrar o SHA-1,

A seguir, serão descritos os principais algoritmos usados para a geração de uma assinatura digital:

- **RSA**: O RSA pode ser utilizado para a geração de assinatura digital. Matematicamente falando: há uma chave pública e uma chave privada, e a segurança de um sistema baseia-se na dificuldade da fatoração de números primos grandes.
- **ElGamal**: Como o RSA, o ElGamal também é comutativo, podendo ser utilizado tanto para assinatura digital, quanto para gerenciamento de chaves; assim, ele baseia sua segurança da dificuldade do cálculo de algoritmos discretos em um corpo finito.
- **DAS**: O *Digital Signature Algorithm*, unicamente destinado a assinaturas digitais, foi proposto pelo NIST em agosto de 1991, para utilização no seu padrão DSS (*Digital Signature Standard*). Adotado como padrão final, em dezembro de 1994, trata-se de uma variação dos algoritmos de assinatura ElGamal e Schnorr. Foi inventado pela NSA e patenteado pelo governo americano.

3.5. *Conclusão*

Neste capítulo foram apresentados alguns trabalhos relacionados com a dissertação proposta, dos quais foram extraídas características positivas e negativas que permitiram nortear os caminhos seguidos por esta proposta.

A avaliação de desempenho apresentada em [CRAM2006] é seguramente importante para conscientizar que o uso de DHT em MANet tem problemas que devem ser contornados ou mesmo evitados. Fazendo uso desta informação, a abordagem utilizada pelo serviço proposto neste documento foi distribuída a DHT, na parte fixa da rede (denominada DSP),

evitando assim, o aspecto negativo do uso de DHT em MANet. Em [MB252005] foi apresentada uma arquitetura alternativa ao uso de JXME para aplicações P2P em dispositivos móveis, acompanhada de uma avaliação comparativa de desempenho entre esta plataforma proposta e o JXME *proxyless*. Neste trabalho, a opção foi fazer uso da plataforma JXME, por tratar-se de um ambiente estável e compatível com a plataforma JXTA da SUN. Diferentemente do que foi apresentado em [BISIG2003], esta proposta não altera o *core* do JXTA para minimizar os efeitos da mobilidade em redes P2P.

Adicionalmente, as experiências obtidas através dos trabalhos relacionados, apresentados neste capítulo, conceitos como os *super-peers* [TRAV2003], proveniente do capítulo de fundamentação, foram importantes na definição do VSP (*Virtual Service Peer*) proposto neste documento. A técnica de replicação e busca de índices no projeto JXTA baseado em DHT e apresentado em [TRAVWALK] foi importante na opção pela DHT *Bamboo* para este fim, no projeto proposto.

Após avaliar a literatura disponível sobre integridade e autenticidade, optamos por fazer uso de certificado autoassinado para autenticar os conteúdos disponibilizados pelo serviço proposto.

Capítulo 4

Gerenciamento Seguro de Conteúdo Compartilhado em PoM

Este capítulo é destinado a apresentação do trabalho proposto, tendo como ponto de partida os pontos que motivaram apresentação do mesmo, bem como os objetivos a serem alcançados. O protótipo desenvolvido para validar o trabalho proposto, também será apresentado neste capítulo.

4.1. Motivação

Redes *peer-to-peer* (P2P) são caracterizadas pela dinâmica com que os *peers* entram e saem da rede, churn, assim como pela variedade de tipos de conteúdos compartilhados que oferecem [SCHOLL2001]. Redes *ad hoc* móveis (MANets) oferecem grande flexibilidade de configuração, mas são muito instáveis em termo de manutenção da conectividade [PERK2001] .

Acompanhando a evolução tecnológica e social, os dispositivos móveis (aparelhos celulares, smartphones, etc.) têm sido parte de nosso dia-a-dia. Naturalmente, a maioria destes dispositivos são capazes de se comunicar por ondas de rádio (conexão sem fio) em MANets, apesar de suas limitações em termos da capacidade de processamento, de armazenamento e de taxa de transferência, quando comparados aos computadores pessoais.

Aplicações que usam P2P como rede de sobreposição (overlay) em ambientes MANet (PoM) devem ter boa tolerância ao churn e instabilidade da conexão [PERK2001] para funcionar corretamente. Ou seja, dois importantes desafios no desenvolvimento de aplicações

em PoM são: suportar a conectividade *ad hoc* com os dispositivos móveis vizinhos e manter a disponibilidade dos *peers* em suas conexões fim-a-fim.

Atualmente, grande parte dos conteúdos compartilhados nas redes P2P tradicionais (e.g., *emule*, *gnutella*, etc.) são poluídos, falsificados ou corrompidos ("lixo P2P") [KUMAR2006] . Consequentemente, os dispositivos móveis que consomem recursos da rede P2P podem gastar uma parte importante de seus recursos, estabelecendo e mantendo conexões com *peer* provedores que podem fornecer lixo P2P. Além disto, estes provedores podem tornar-se ausentes, durante o fornecimento dos conteúdos ou a conectividade da MANet pode ser perdida.

Quando as redes P2P foram propostas, infraestrutura para publicar e encontrar conteúdos (repositório de índices) não foram previstas; a publicação/busca por conteúdos compartilhados era feita através do envio de mensagens em broadcast. A Distributed Hash Table (DHT) assumiu o papel repositório de índices de conteúdos compartilhados para as redes P2P [STOICA2001]. O principal benefício alcançado com o uso de DHTs em redes P2P é consequentemente a indexação de conteúdos. Além disto, a DHT é distribuída, tolerante a faltas e visa manter-se sempre disponível.

4.2. Objetivos

O objetivo deste trabalho é propor um modelo de provimento de serviço P2P seguro que permita a alta disponibilidade do conteúdo de forma transparente para os *peers* no papel de "cliente", mesmo quando há desconexão de alguns nodos da rede. Além disto, objetiva-se a autenticidade e integridade dos conteúdos providos.

Assim, baseado em uma rede de sobreposição, nossa proposta é definir um serviço que libera o *peer* móvel (que geralmente tem o papel de consumidor na rede P2P) da tarefa de controlar conexões fim-a-fim com o *peer* que lhe fornece conteúdos compartilhados. Este esquema permite minimizar o efeito negativo do churn da rede P2P para o *peer* consumidor, fazendo com que o conteúdo compartilhado seja ininterruptamente fornecido ao *peer* consumidor de forma transparente.

O serviço proposto também visa minimizar o impacto da instabilidade da conectividade de MANet, mantendo, em nível de aplicação, o estado conexão e permitindo transferências parciais de conteúdos. O serviço opera de modo seguro – garantindo a integridade, autenticidade e disponibilidade dos conteúdos. Neste cenário, também foi

considerada a migração semiautomática de conteúdos compartilhados, a partir de redes P2P tradicionais, para a rede orientada a serviço em PoM.

Para atingir o objetivo geral, foi necessário alcançar os seguintes objetivos específicos:

a) Modelar a inclusão do *peer* de serviço em uma rede P2P, baseado em uma arquitetura tradicional,
b) Propor um mecanismo que garanta autenticidade aos conteúdos oferecidos,
c) Definir um mecanismo para a integridade do conteúdo,
d) Garantir alta disponibilidade e desempenho ao serviço de provimento de conteúdos PoM,
e) Definir um mecanismo para migrar os conteúdos da rede tradicional para a rede orientada a serviço,
f) Prototipar um caso de uso para avaliar a proposta, através de medições em um ambiente real.

4.3. Serviço Proposto

Em nossa proposta, um *peer* de serviço denominado VSP (*Virtual Service Provider*) faz a intermediação da conexão entre o *peer* consumidor e os *peers* que efetivamente proverão os conteúdos requisitados.

A intermediação do *VSP* inclui a localização de provedores para os conteúdos compartilhados, bem como o gerenciamento da conexão fim-a-fim entre o *peer* consumidor e os provedores, mantendo o estado da conexão do consumidor e garantindo segurança (disponibilidade, integridade e autenticidade) ao conteúdo compartilhado.

Na verdade, o *peer* consumidor não deverá perceber que está conectado a um *peer* de serviço (VSP), ou seja, do ponto de vista do *peer* consumidor sua conexão é estabelecida com um *peer* provedor do conteúdo compartilhado requisitado.

O provimento de serviço baseado em DHT (DSP, *DHT-based Service Providing*) representa a rede P2P orientada a serviço como uma rede de sobreposição (*overlay*) sobre MANet (PoM). Cada nó da DHT na DSP é um VSP vizinho na tabela de roteamento da rede de sobreposição.

Os VSP são implementados como nós da rede de sobreposição DHT, dando suporte a DSP. Isto significa que o nó da DHT responsável por armazenar uma entrada, par (chave, valor), que indexa um conteúdo compartilhado instancia o VSP para controlar os aspectos

relativos a esse conteúdo. Assim, todos os conteúdos compartilhados relacionados a um serviço são armazenados no mesmo nó da DHT. Esta estratégia facilita a administração do serviço e permite a customização quando requerida pelo serviço, sem causar perdas ao esquema da DHT. A customização do nó da DHT poderia ser requisitada quando o conteúdo compartilhado é muito solicitado, por exemplo. Neste caso, a combinação de *hardware* e *software* poderia ser melhorada para suportar essa demanda.

Adicionalmente, é proposta a estratégia de migração semiautomática dos conteúdos compartilhados da rede P2P tradicional para a rede P2P orientada a serviço. O principal objetivo do esquema de migração é permitir a aquisição de conteúdo compartilhado de uma rede P2P tradicional, classificá-lo e disponibilizá-lo na rede orientada a serviço, assegurando sua integridade (e.g. conteúdos livres de alterações maliciosas) e sua autenticidade (e.g. garantia de que o conteúdo compartilhado não é falso ou poluído).

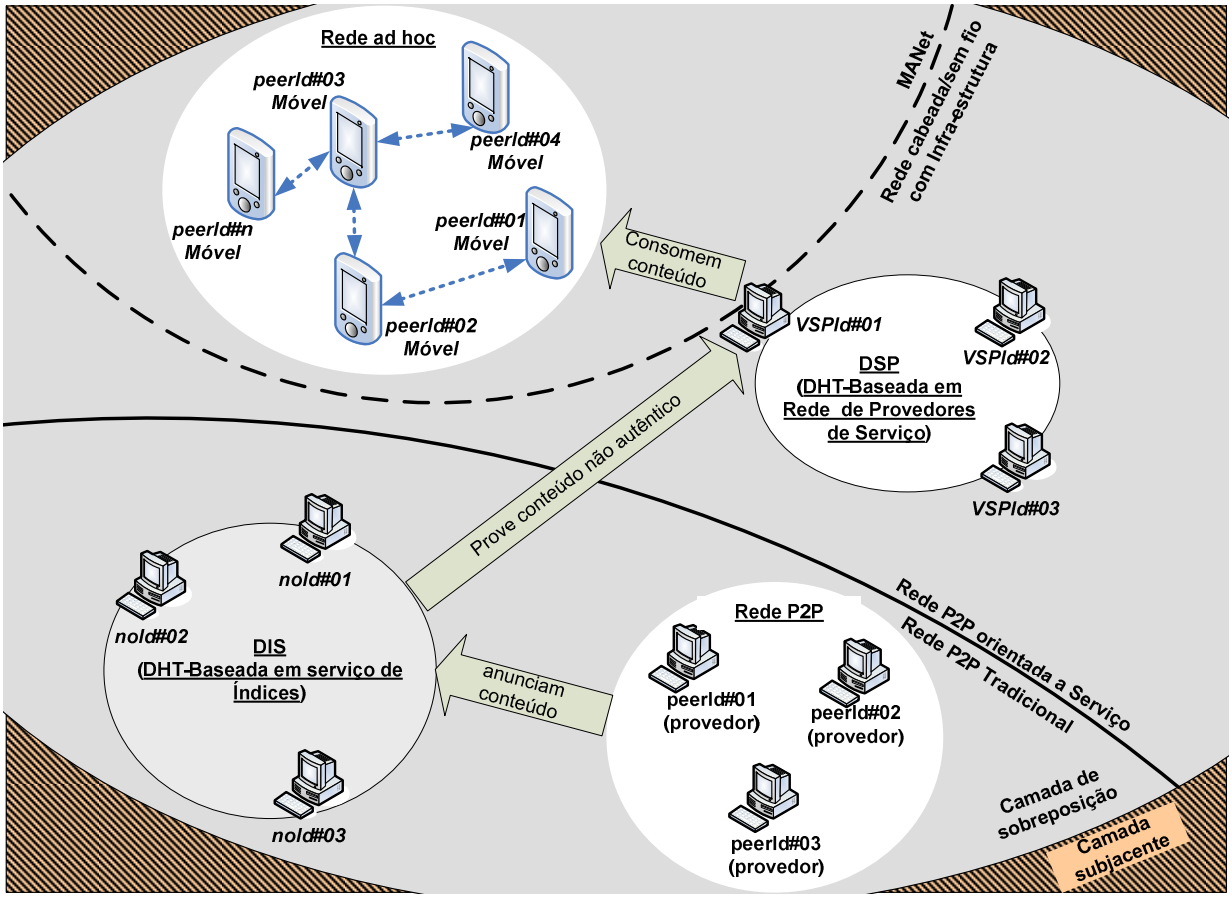

Figura 4-1: Visão geral dos elementos da PoM

A classificação dos conteúdos é baseada em métodos heurísticos que pré-classificam os conteúdos de acordo com o seu grau de similaridade [OLIV2006]. Depois disso, o

resultado da classificação deve ser avaliado e certificado por um humano, o administrador do VSP, que é o provedor do serviço. Só então, o VSP (Figura 4-1) pode publicar na DSP o conteúdo obtido (migrado) a partir da rede P2P tradicional. Depois de disponibilizados na rede de serviço (DSP na Figura 4-1), os conteúdos não precisam mais de classificação e certificação por humanos, pois são íntegros e autênticos. Além disso, os conteúdos compartilhados obtidos a partir de fonte confiável podem ser publicados e disponibilizados diretamente na rede de serviço.

Na Figura 4-1, o *peer* consumidor requisita um conteúdo ao VSP#01, que procura dentre os *peers* da rede tradicional os que poderiam ser fonte para o provimento do conteúdo compartilhado que está sendo requisitado. Nas implementações recentes de redes P2P tradicionais, *peers* anunciam seus conteúdos compartilhados numa DHT (que se comportam como um serviço de índices P2P, DIS – *DHT-based Index Service*) e os *peers* consumidores acessam a DIS procurando pela localização de provedores para os conteúdos desejados. Em nosso caso, após a localização do conjunto de provedores potenciais para o provimento do conteúdo requisitado, o VSP contata aqueles que estão disponíveis a fim de descarregar o conteúdo requisitado. Em seguida, os conteúdos são classificados e podem ter sua autenticidade e integridade certificadas por um humano.

Ainda na Figura 4-1 o VSP#01 anuncia o conteúdo na DSP. Uma vez que o anúncio é registrado na DSP, um serviço assegurará a integridade e autenticidade de tal conteúdo compartilhado, que se torna disponível para os *peers* móveis consumidores.

Um *peer* consumidor pode requisitar um conteúdo compartilhado a um VSP, que de acordo com a designação de nó DHT responsável deveria armazenar tal entrada, mas pode acontecer que o VSP não tenha um serviço para a requisição. Neste caso, o VSP propaga a requisição de um conteúdo a DIS, e aguarda a descarrega e pré-classifica o conteúdo, e responde ao *peer* consumidor com o conteúdo selecionado – possuidor do maior grau de probabilidade de ser autêntico e íntegro. Entretanto, o VSP não tem nenhum recurso para assegurar a autenticidade e a integridade do conteúdo selecionado, uma vez que o mesmo não foi certificado por um humano e previamente anunciado na DSP. Se o *peer* consumidor concordar com estas condições de não provimento de garantias, o VSP não pode ser responsabilizado por uma avaliação imprecisa da classificação automática.

É importante observar que, para a classificação no VSP, o conteúdo deve ser transferido a partir dos provedores e armazenado no *cache* do VSP. Assim, o consumidor

pode obter fragmentos de diferentes *offsets* do mesmo conteúdo, simultaneamente, a partir do *cache* do VSP, com boa probabilidade do conteúdo estar íntegro**.** Depois de transferir o conteúdo compartilhado, um consumidor pode anunciar-se na DSP como provedor do conteúdo que acabou de descarregar.

Quando o VSP anuncia um serviço na DSP, também publica o *hash* do arquivo de conteúdo como um todo e uma lista de fragmentos e respectivos *hashes*. Porém, enquanto os fragmentos não forem descarregados e os *peer* consumidores não se anunciarem na DSP como provedores dos mesmos, esses serão mantidos no *cache* do VSP para garantir que haverá no mínimo uma cópia de um fragmento disponível na rede orientada a serviço.

O administrador do VSP é responsável por configurar o tempo que o conteúdo deverá ser mantido no *cache* do VSP e a quantidade de recurso (memória, HD) destinado a esse fim. Apesar da importância de tais políticas de atualização e das técnicas da manutenção do *cache*, estes assuntos não são objeto deste trabalho.

A tolerância a faltas do VSP é obtida diretamente através da auto-organização da rede de sobreposição da DHT. No caso de falha de um VSP, o nó vizinho na rede de sobreposição da DHT tomará seu lugar no provimento do serviço, sendo que a reconfiguração da tabela de roteamento da DSP implica na perda do estado do serviço. Assim, o nó que assumiu o lugar do nó faltoso deverá tocar mensagens com o *peer* consumidor a fim de recuperar o estado da conexão anterior à ocorrência da falta.

4.4. Protótipo

O protótipo utiliza a infra-estrutura JXTA como *framework* P2P, assim como a implementação da DHT *Bamboo* [PLANET2008], e NetBeans IDE 5.5 com o *toolkit* para comunicação sem fio (*J2ME Wireless Toolkit*) [CLDC2008]. Este *toolkit* emula um dispositivo móvel, comunicando-se via JXME com o JXTA. Todos os blocos identificados como APP correspondem aos módulos que dão suporte à arquitetura proposta. A rede de serviço (DSP) é composta pelo VSP que está integrado a DHT *Bamboo*, como um de seus nós.

A Figura 4-2 apresenta a arquitetura da proposta, destacando os elementos que compoem o sistema proposto (*Consumer*, *Provider*, VSP, DSP e DIS) bem como as facilidades de cada um deles.

O módulo *GUI Administrator* permite o gerenciamento do VSP, incluindo a

configuração de funcionalidades e a manipulação de conteúdos. O histórico (*history*) registra todos os conteúdos autênticos e corrompidos, bem como os conteúdos automaticamente classificados e selecionados pelo VSP como candidatos à certificação, através da intervenção humana. Esta facilidade também auxilia o classificador heurístico (*classifier*) na escolha do conteúdo que possui a maior chance de ser autêntico. O histórico atua como um arquivo de log no *service manager* do VSP; sua atualização é executada pelo *content manager*. O gerenciamento de integridade e de autenticidade (*integrity manager* e *authenticy manager*) é executado através de funções *hash* e assinatura digital, respectivamente. Estes módulos quando invocados permitem que o serviço de gerenciamento (*service manager*) possa assegurar a integridade e a autenticidade dos conteúdos anunciados na rede P2P, orientada a serviço.

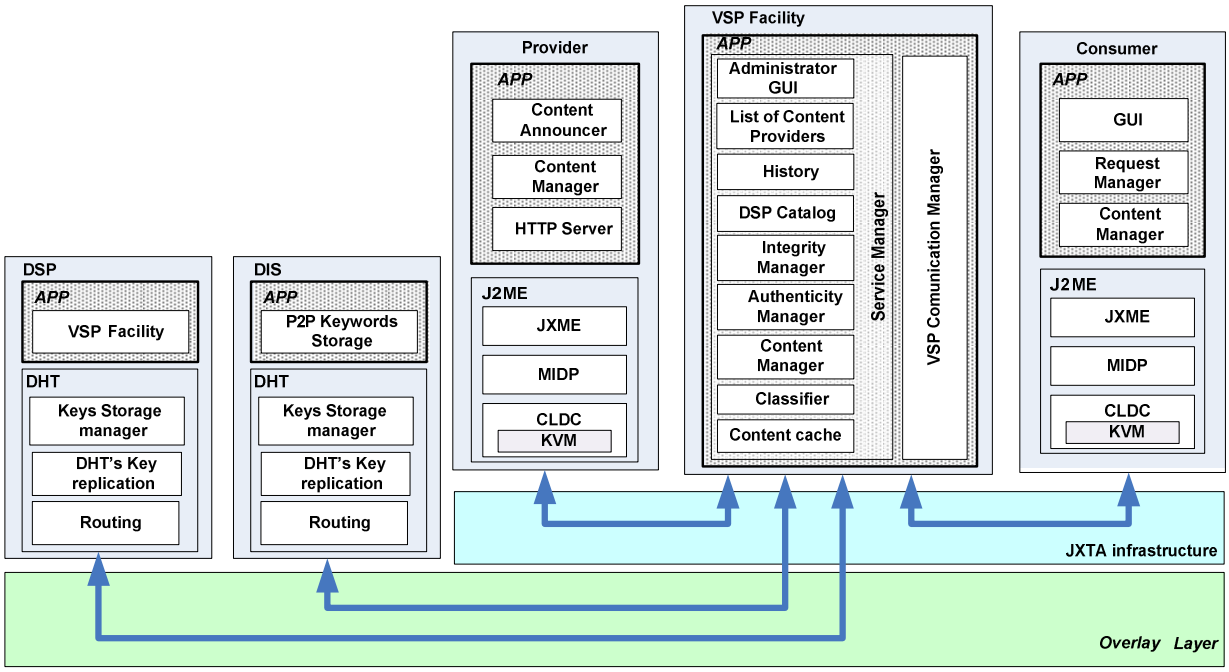

Figura 4-2: Arquitetura da proposta

O Classificador (*classifier*) faz uso da técnica de clusterização para agrupar os conteúdos por similaridade. Então, baseado na semelhança dos arquivos de conteúdos e no número de ocorrências, o conteúdo com maior probabilidade de ser autêntico, é estimado. O módulo gerenciador de conteúdos (*content manager*) é o objeto central do VSP, responsável pelo armazenamento dos arquivos de conteúdos como *cache* do VSP e pela oferta de uma

interface de manipulação dos conteúdos providos como serviços.

O gerenciador de conteúdos procura por provedores de conteúdos, atualizando a lista dos provedores de conteúdo (*list of content providers*), gerenciando a descarga de conteúdos e a *cache* de conteúdos (*content cache*). Os parâmetros de configuração da *cache* de conteúdos são manipulados através do *Administrator GUI*.

O módulo de catálogo da DSP (*DSP catalog*) registra todos os conteúdos autênticos e íntegros disponíveis na DSP. Este módulo usa o histórico como fonte de informação. O gerenciador de serviços usa o catálogo DSP como serviço de anúncio de conteúdos, na rede P2P, orientada a serviço.

O gerenciador de serviço (*service manager*) é a interface de acesso ao VSP. Esta interface possibilita a busca e aquisição de conteúdos, a migração de conteúdos da rede P2P tradicional para a DSP, o gerenciamento do estado do serviço, gerenciamento de descarga de conteúdos, gerenciamento da lista de provedores de conteúdos e a integração entre todos os módulos do VSP

A DSP instancia o módulo de facilidades do VSP (*VSP Facility*) que suporta a rede de VSPs e as funções típicas de DHT como o roteamento, o gerenciamento do armazenamento de entradas (pares (chave, valor)) e a replicação das entradas da DHT.

O acesso à rede P2P tradicional é implementado por uma aplicação *emule*, cliente da *DHT Kademlia* (representando a DIS), utilizada como repositório para índices P2P (*storage* for *P2P keywords)*, através da qual o *peer* consumidor pode fazer buscas pelas palavras chave que identificam os conteúdos.

Embora as entidades consumidor e provedor sejam detalhadas separadamente por razões didáticas, na prática, cada *peer* pode atuar em ambos os papéis de acordo com suas necessidades momentâneas.

O módulo anunciador de conteúdos (*content announcer*) do provedor é utilizado no protótipo para publicar na DSP a disponibilidade de um ou mais fragmentos íntegros e autênticos de alguns arquivos de conteúdos. O módulo gerente de conteúdo do provedor ocupa-se com o armazenamento local dos conteúdos anunciados e oferece uma interface para manusear o conteúdo armazenado localmente; isto inclui recursos para assegurar a integridade e a autenticidade dos conteúdos anunciados. O servidor http (*http server*), implementado no lado P2P servidor, fornece os conteúdos compartilhados a serem transferidos através do JXTA para o *peer* consumidor.

A *Consumer GUI* habilita acesso aos recursos no lado consumidor, enquanto o módulo Gerente de Requisição (*request manager*) oferece uma interface para busca de conteúdos no VSP e através da *Consumer GUI* gerenciar as requisições enviadas ao *peer* provedor. O Gerente de Conteúdos do consumidor executa requisições de descarga, reconstrói arquivos de conteúdos, a partir de fragmentos, após o fim da transfência e verifica a integridade e autenticidade do arquivo de conteúdo.

A inicialização do *peer* consumidor exige a inicialização do CLDC e JXME para que a inicialização do *peer* aconteça através da leitura do arquivo de configuração com informações de rede (*IP Address*, IP/porta do proxy JXTA, IP do *relay* JXTA, etc.). Então, um *peerID* (associado a um *groupID*) e um *pipe* de entrada são criados e anunciados na DSP. Quando o *peer* armazena conteúdos compartilhados, passa a atuar também como provedor, então um *pipe* de saída é criado e todos os conteúdos disponíveis para descarga são anunciados na DSP durante a inicialização do *peer*.

A inicialização do VSP requer a inicialização do nó da DHT *Bamboo*, que por sua vez, invoca a execução da aplicação JXTA que agirá como um provedor de serviço P2P. Além disso, todos os módulos do gerente de serviço são carregados.

Na Figura 4-2 foi assumido que os provedores na rede P2P tradicional haviam anunciado a disponibilidade de alguns conteúdos na DIS. Após a inicialização do *peer* consumidor, o usuário poderá selecionar uma das seguintes opções: procurar um *peer*, enviar um anúncio, procurar um conteúdo e descarregar o conteúdo requerido.

Um *peer* móvel requisita ao gerente de serviço a busca por um conteúdo. O VSP, por sua vez, aciona a DSP que invoca a função *get* da DHT, através do gerente de conteúdos. Isto faz com que a função de *hash* seja aplicada sobre o conteúdo buscado (palavra chave), e a rede de sobreposição faz o roteamento da requisição ao nó responsável por aquele espaço chave da DHT, o VSP.

O VSP endereçado pela rede de sobreposição é então instanciado. VSPs são virtualmente nós da rede de sobreposição da DHT *Bamboo*, dando suporte a DSP. Isto significa que o nó DHT armazena todas as informações relacionadas a um serviço provido pelo VSP. Para conseguir tal comportamento em uma DHT, todas as informações relativas a um conteúdo são armazenadas com a mesma palavra chave, gerando o mesmo *hash* no espaço de *hashes* (entradas) da DHT. Assim, um único nó da DHT armazenará todo o conjunto de informações sobre um conteúdo/serviço, facilitando a administração do mesmo.

Se o conteúdo buscado já foi certificado por um humano (estará publicado na DSP), então a DSP retorna a lista de provedores para aquele conteúdo, para o gerente de serviço. O gerente de serviço conecta-se aos provedores ativos, através do gerente de conteúdos e começa transferir o conteúdo. Cada provedor implementa um servidor HTTP básico para fornecer conteúdos. Informações sobre o servidor HTTP do fornecedor, como: endereço IP, porta TCP e URL (caminho relativo no *peer* provedor) são incluídas no anúncio do conteúdo, que é publicado na DSP durante a inicialização do provedor.

Após a transferência do conteúdo e a verificação de sua integridade e autenticidade, o VSP encaminha o conteúdo ao *peer* consumidor. Se o gerenciador de serviços não encontrar o conteúdo desejado na DSP, uma requisição é encaminhada a DIS, retornando uma lista de possíveis fornecedores para o conteúdo na rede P2P tradicional, então o classificador será ativado.

Esta sequencialidade de ações apresentada acima teve o intuíto de apresentar as facilidades do sistema proposto (módulos funcionais), destacando as responsabilidades de cada um destes módulos. No capítulo 4.6. alguns cenários serão apresentados mais detalhadamente, destacando a sequencialidade de ações que caracterizam cada um deles.

No capítulo a seguir, serão apresentados detalhes relacionados com a implementação do protótipo utilizado para validar a proposta.

4.5. Implementação do protótipo

No protótipo, a autenticidade é garantida pelo mecanismo de assinatura digital, porém, para trabalhar com a assinatura digital são necessárias chaves assimétricas. Como em P2P é difícil trabalhar com entidades certificadoras, optou-se pelo uso de certificado digital autoassinado. Foi utilizada a aplicação *keytool* [KEYTOOL2002] para gerar as chaves RSA de 1024 bits, cuja sintaxe do comando é: *keytool -genkey -keyalg RSA -keysize 1024 -alias alias -keystore .keystore*.

Conforme citado na dissertação (ver Figura 4-2 importada da dissertação para ilustrar o protótipo implementado), o protótipo é composto por mais de uma entidade: Provedor (*Provider)*, Consumidor(*consumer*), VSP, DSP e DIS.

As entidades DIS e DSP estão relacionadas basicamente a DHTs, sendo a primeira uma DHT característica de redes P2P tradicionais e a segunda caracterizada por dar suporte ao serviço proposto por este trabalho, tendo no VSP uma entidade que proverá o acesso e os

dados que serão armazenados na DSP.

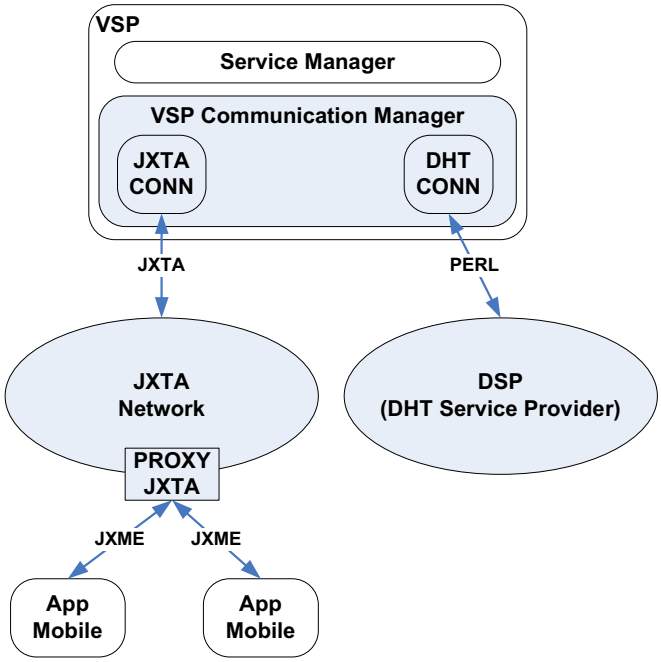

Figura 4-3: Aplicações que compõem o protótipo

Quanto as outras entidades: Consumidor, Provedor e VSP, é importante ressaltar que nestas se concentraram o maior esforço de implementação do protótipo. Estas entidades estão agrupadas em duas aplicações distintas, conforme mostra a Figura 4-3.

Por tratar-se de uma rede P2P, onde um *peer* pode assumir os papéis de provedor e consumidor, no protótipo, a implementação destas duas entidades foi feita em uma única aplicação, denominada "*App Mobile*".

Já a implementação da entidade VSP foi efetuada em uma aplicação chamada "VSP", que é subdividida em dois módulos: *Service Manager* e *VSP Communication Manager*, assim feito devido a funcionalidade de cada um deles. O *VSP Communication Manager* é responmsável pelo gerenciamento da comunicação entre o VSP e demais entidades: DSP e *App Mobile*. O *Service Manager*, por sua vez, é responsável pela implementação das demais funcionalidades desempenhadas pelo VSP que já foram descritas ao longo deste documento.

Com o intuito de detalhar estas duas aplicações, as seções que seguem irão descrever estas aplicações.

4.5.1. App Mobile

Basicamente, esta aplicação concentra os módulos funcionais necessários por executar os papéis de consumidor e provedor de conteúdo P2P, além é claro, das funcionalidades de

gerenciamento de conexão com os VSPs.

O diagrama a seguir (Figura 4-4) ilustrará, de forma sintética, a máquina de estados desta aplicação. O detalhamento dos processos apresentados no diagrama ilustrado por esta figura (de forma macro) permitirá um melhor entendimento desta aplicação.

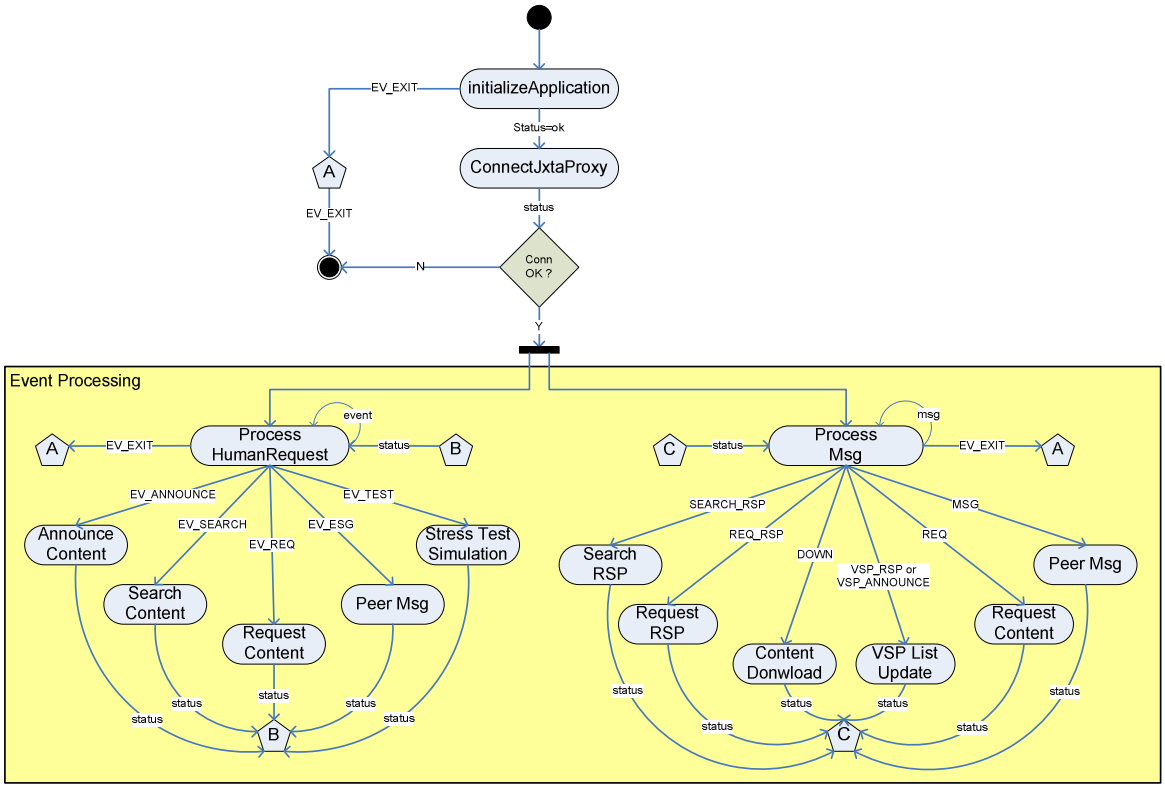

Figura 4-4: Máquina de estados da *App Mobile*

I. Detalhamento dos processos da aplicação *App Mobile*

Duas fases ficam bem claras na *AppMobile*, que imediatamente abaixo, serão descritas:

A. **Inicialização**: durante esta fase, esta aplicação faz, primeiramente, a inicialização da aplicação (*InitializeApplication*), onde acontece a inicialização da aplicação propriamente dita (leitura do arquivo de propriedade e inicialização de respectivas variáveis e configuração de parâmetros baseado em informações fornecidas pelo operador da aplicação). Posteriormente, o *peer* é inicializado (ConnectJXTAProxy), ou seja, a inicialização da aplicação em relação ao ambiente JXTA acontece, onde a entidade *peer* é criada e devidamente registrada no JXTA, além do que, este *peer* é associado a um grupo. Finalmente, um canal de comunicação (*pipe*) pelo qual o *peer* é acessado por demais *peer* da rede P2P é criado e registrado na rede JXTA nesta fase. Ainda nesta fase, a *AppMobile*

se conectará ao VSP (VSP Master) configurado através do arquivo de propriedade desta aplicação, para obter a lista de todos os VSPs ativos na rede, candidatos a “provedores” de conteúdo para esta rede P2P.

B. **Criação dos processos da aplicação**: após a fase de inicialização da aplicação, dois processos são criados com o intuíto de processar dois tipos diferentes de entradas da aplicação: Eventos gerados pelo operador da aplicação (*ProcessHumanRequest*) e outro para tratar as mensagens JXTA (mais propriamente dito, mensagens JXME — ProcessMsg) que chegam à aplicação, quer seja como requisições de outros *peers*, ou como resposta a requisições feitas por este *peer*.

Figura 4-5: Ilustração do HCI (*Human Client Interface*) do *App Mobile*

Entrando mais a fundo no processamento de eventos desta aplicação (Figura 4-4– *Event Processing*), vale detalhar os processos a seguir:

- **Eventos do operador (*ProcessHumanRequest*):**Como previamente descrito, no desenvolvimento deste protótipo, foi usado um *ToolKit* para a implementação da

aplicação móvel CLDC. Através de menus disponibilizados pela biblioteca deste *ToolKit*, o operador pode acessar funcionalidades disponíveis nesta aplicação, conforme ilustra a Figura 4-5. As funcionalidades relevantes apresentadas na Figura 4-4 e mais especificamente no quadro destacado e chamado *Event Processing*, são:

- ***Anúncio de conteúdo*** (*Announcecontent*): A *App Mobile*, no desempenho de seu papel de provedor, deve anunciar seus conteúdos para a DSP, através do VSP. Para tanto, esta aplicação oferece dois mecanismos para este fim. O primeiro deles é através de uma opção de menu, de forma similar ao que foi ilustrado na Figura 4-5, porém destinada a este fim. O operador do sistema deverá indicar o nome do conteúdo que deseja anunciar na DSP e selecionar a opção de menu que comandará a execução do anúncio, que se dará através do envio de uma mensagem JXME, destinada ao VSP em que a aplicação está conectada, através do módulo VSP *Communication Manager* e cujo formato é apresentado pela Mesnagem 1 (Anexo 1) .A Figura 4-6 apresenta o diagrama de seqüência do anúncio/publicação de um conteúdo na DSP; O segundo mecanismo será apresentado no tópico **Simulação e teste de estresse.**
- ***Procura conteúdo*** (*SearchContent*): O operador do *App Mobile* poderá executar a procura por um conteúdo desejado. Esta aplicação disponibiliza uma opção de menu para este fim. Solicitada a procura por um dado conteúdo, o *App Mobile* enviará uma mensagem JXTA para o VSP, ao qual está conectado, requisitando a verificação da existência de algum anúncio daquele conteúdo na DSP, mensagem cujo formato é apresentado em Mesnagem 2 (Anexo 1). O resultado da procura por um conteúdo, ou seja, ele é positivo ou não, é enviado ao *peer* solicitante através de outra mensagem JXTA apresentada na Mesnagem 3 (Anexo 1). De posse da mensagem de procura por um conteúdo, o *Content Manager* submeterá esta consulta ao DSP, de acordo com o diagrama de seqüência, mostrado na Figura 4-7. Se houver a necessidade de encanhinhar a consulta por um conteúdo a DIS, o fluxo desta propagação de consulta segue a ilustração feita pelo diagrama da Figura 4-8. O acesso a DIS será realizado através do *Kademlia*, que propiciará a API necessária para realizar as consultas aos conteúdos registrados nesta DHT. Lembrando sempre que para o requisitante, o provedor do conteúdo é o VSP, independente de

quem realmente é o provedor, seja ele um provedor da DSP ou da DIS. No entanto, se a fonte real deste conteúdo for a DIS, o consumidor será informado que o conteúdo não tem garantia de autenticidade.

- ***Requisita conteúdo*** (*RequestContent*): Encontrado um conteúdo (independente se na DSP ou DIS) o *peer* consumidor enviará uma mensagem para o VSP que está conectado de acordo com o formato apresentado em Mesnagem 4 (Anexo 1). da mesma forma que por ocasião da procura por um conteúdo, o peer consumidor gera um identificador exclusivo para a requisição do conteúdo e ambas as informações são fornecidas para que conteúdo seja mais facilmente encontrado na DSP. Como resposta a esta requisição, o VSP enviará dois formatos de mensagens para o *peer* consumidor. A primeira delas conterá as informações relacionadas ao conteúdo que será transferido (provedor, número total de fragmentos, ...) e um segundo formato está relacionado com a transferência em sí. Conterá os dados que estão sendo transferidos. Os formatos destas mensagens serão apresentados pela Mesnagem 5 e Mesnagem 6 (Anexo 1) respectivamente. Se por qualquer problema, o conteúdo não estiver mais disponível, apenas a Mesnagem 5 (Anexo 1) será enviada informando o resultado negativo da busca. A dinâmica desta operação foi amplamente abordada no capítulo destinado à proposta (Capítulo 4 da dissertação).
- ***Envia mensagem para outro peer*** (*PeerMsg*): Através deste comando, o ususário pode enviar uma mensagem para um outro peer. Esta funcionalidade, assim como as anteriores, faz uso de uma mensagem JXME, cujo formato está descrito na Mesnagem 7 (Anexo 1).
- ***Simulação e teste de estresse*** (*StressTestSimulation*):esta funcionalidade foi implementada basicamente para simular um teste de estresse do sistema proposto. Basicamente o operador poderá executar duas operações através desta funconalidade: a) fazer o anúncio de um conjunto de conteúdos deste *peer*, através de um arquivo texto (*script*) onde todos os conteúdos relacionados neste arquivo são anunciados/publicados na DSP, através da VSP, de modo similar, porém automático, ao que foi descrito no item 1 desta seção; b) fazendo uso de um arquivo texto (*script*), o operador do sistema poderá

iniciar uma seqüência múltipla de procuras e transferências de conteúdos, de forma similar ao que foi descrito nos itens 2 e 3 desta seção, porém de modo automático, gerando carga para o sistema.

- **Entrada de mensagens JXTA(*ProcessMsg*):** Esta tarefa, por sua vez, é responsável pela captura e processamento de mensagens JXME, enviadas para este *peer*. O código fonte abaixo (Código Fonte 4-1) ilustra, em uma visão geral, o algoritmo da função que executa esta tarefa. Esta tarefa fica executando eternamente, enquanto a aplicação estiver executando. Na linha 3, a pilha de mensagens JXME recebida é acessada e a primeira mensagem é retirada para processamento. Na linha 4, o campo MSG_TYPE é verificado para saber se a mesnagem obtida é válida para o sistema. Se a mensagem for conhecida pelo sistema, na linha 5, ela será processada/interpretada e devidamente encaminhada à função que é responsável pelo seu completo processamento. Caso contrário, a mensgem será descartada e então o ciclo repitirá até que esta tarefa seja encerrada. Basicamente, esta aplicação tratará as mensagens JXME citadas no item A, desta seção e descritas através das mensagens especificadas no Anexo I da dissertação.

```
1. ProcessMsg ()
2.          Loop (forever)
3.                    msg = PeerNetwork::poll (timeout)
4.                    IF(verifyMsgType (msg) == VALID)
5.                              parseMsg(msg)
6.                    Else
7.                              discardMsg (msg)
8.                    End IF
9.          End Loop
10.End Function
```

Código Fonte 4-1: Processamento de mensagens JXME

Ainda relacionado com o processamento de mensagens JXME, é importante descrever o mecanismo de *lookup* que é responsável pelo monitoramento da conectividade entre a aplicação consumidora e o VSP. A existência de múltiplos VSPs, na solução proposta, é possível e comum, no entanto, é necessário que o "alguém" monitore a conectividade com os VSPs. Durante a fase de inicialização dos *peers*, o mecanismo de *lookup* é inicializado para a conectividade com o VSP default (configurado via arquivo de propriedades) seja verificada. Ao longo do tempo de vida do sistema, sempre que um novo VSP seja introduzido à rede, ou ainda que um VSP cadastrado saia de serviço, uma mensagem é enviada a todos os *peers* (VSPs e *App Mobiles*) para que a tabela de VSP da rede seja atualizada [Mesnagem 9 (Anexo 1)]. Desta forma, o mecanismo de *lookup* poderá basear-se nesta tabela para escolher um novo

VSP em caso de necessidade. O Código Fonte 4-2 apresenta o algoritmo do mecanismo de *lookup* implementado, onde a linha 4 do Código Fonte 4-2 é responsável por pegar o próximo elemento (VSP) da tabela de VSPs disponíveis na rede. De posse do identificador do próximo VSP, na linha 6, o mecanismo validará este elemento, através de uma chamada ao VSP, ou seja, o mecanismo tentará conectar-se a este VSP, que em caso positivo, nas linhas 7 e 8 do Código Fonte 4-2, as informações serão armazenadas em variáveis locais, para futuro acesso ao novo VSP. Se o VSP recuperado não tiver sua validade atestada, o contador de tentativas será decrementado na linha 11. O critério de parada é baseado no número de tentativas, configurado a partir do arquivo de propriedades da application.

```
1. vspLookup()
2.          Loop (tryOneMoreTime > 0)
3.
4.                    vspId = getNextVspId()
5.
6.                    IF (verifyVspConnectivity(vspId) == VALID)
7.                              activeVspId = vspId
8.                              activeVspPipeId = getVspPipeId(vspId)
9.                              break
10.                   Else
11.                             tryOneMoreTime—
12.                   End IF
13.         End Loop
14.End Function
```

Código Fonte 4-2: Mecanismo de *lookup* de VSPs

4.5.2. VSP

No intuíto de detalhar o VSP, os dois módulos funcionais (VSP Communication Manager e Service Manager) que constituem esta aplicação serão detalhadas nas seções a seguir.

I . VSP Comunication Manager

Com relação à implementação do módulo funcional *VSP Communication Manager*, ele é basicamente responsável pela conectividade do VSP com os App Mobiles (*peers* consumidores e provedores) e DHT (DSP e DIS).

A connectividade do VSP com o DSP se dá através de uma interface PERL com a DHT, conforme ilustra a Figura 4-3. A DIS, por sua vez, é acessada através de um cliente *Kademlia*, que via Internet, connecta-se à rede P2P *emule*, de onde os conteúdos são transferidos para o VSP e, posteriormente, ao *peer* solicitante ou consumidor.

Quanto à conectividade com o *App Mobile*, este módulo funcional, durante a inicialização do VSP, deve conectar-se com o *proxy* JXTA, de modo que o VSP torne-se apto

a trocar mensagens JXME com os demais *peers* da rede. Assim como foi apresentado no Código Fonte 4-1, no *App Mobile*, este módulo implementa um procedimento de captura e processamento de mensagem JXME. Para ilustrar as rotinas de processamento de uma mensagem JXME, o Código Fonte 4-3 mostrará como uma mensagem de busca por um conteúdo é interpretada.

```
1.parseContentRequest(Message msg)
2.
3.        IF (validateMsg (msg) == true)
4.                Loop (numberOfFields
5.                        fieldList [n]= getMsgField (n, msg)
6.                        /*this procedure will repeat according with msg field number*/
7.                End Loop
8.
9.                ContentManager::RequestContent (fieldList) /* ServiceManager */
10.
11.       Else
12.               discardMsg (msg)
13.       End IF
14. End Function
```

Código Fonte 4-3: Processamento da mensagem de anúncio de conteúdo

Na linha 3 do Código Fonte 4-3, a mensagem recebida tem a validade de seus campos verificada e desde que a mensagem seja válida, através de um laço de repetição que ocorre entre as linhas 4 e 7, as informações são extraídas da mensagem e armazenadas em uma variável local, variável esta que será passada como parâmetro na função que executará a requisição de busca por um conteúdo, que no Código Fonte 4-3 é mostrada na linha 9. Cabe ressaltar, neste ponto, que no módulo funcional *VSP Communication Manager*, as mensagens JXME são processadas, mas que as informações extraídas destas mensagens serão repassadas ao módulo funcional adequado do *Service Manager,* onde o serviço será efetivamente provido, que no Código Fonte 4-3, refere-se ao *Content Manager*.

O caminho inverso também é verdadeiro, ou seja, o Service Manager fará uso de funções do *Communication Manager* para comunicar-se com demais *peers*, sejam eles *App Mobiles* ou outros VSPs.

II. Service Manager

Neste tópico, o módulo funcional *Service Manager* será abordado no aspecto implementação. Para tanto, seus principais serviços seráo detalhados.

Entre as principais funcionalidades oferecidas por este módulo funcional, é importante ressaltar: Anúncio de um conteúdo, Procura por um conteúdo e Busca de um conteúdo.

A. **Anúncio de um conteúdo**: como amplamente comentado nos capítulos e seções

anteriores, esta funcionalidade é responsável pela publicação dos conteúdos disponíveis nos *peers* provedores na DSP. No protótipo implementado, a *App Mobile*, através de uma mensagem JXME (Anexo I), anuncia os seus conteúdos ao VSP, ao qual está conectado. A Figura 4-6 ilustra, através de um diagrama de sequências, o fluxo da mensagem e a seqüencialidade das ações realizadas para que o anúncio seja realizado. Apenas relembrando o que já foi explicado e detalhado anteriormente, a porta de entrada/saída de mensagens JXME no VSP é o módulo funcional *VSP Communication Manager*, que é responsável por encaminhar a mesma ao módulo funcional adequado do *Service Manager*, que para esta funcionalidade é o *DSP Catalog*, onde os dados do conteúdo a ser anunciado serão coletados e encaminhados ao DSP, através do *VSP Communication Manager*.

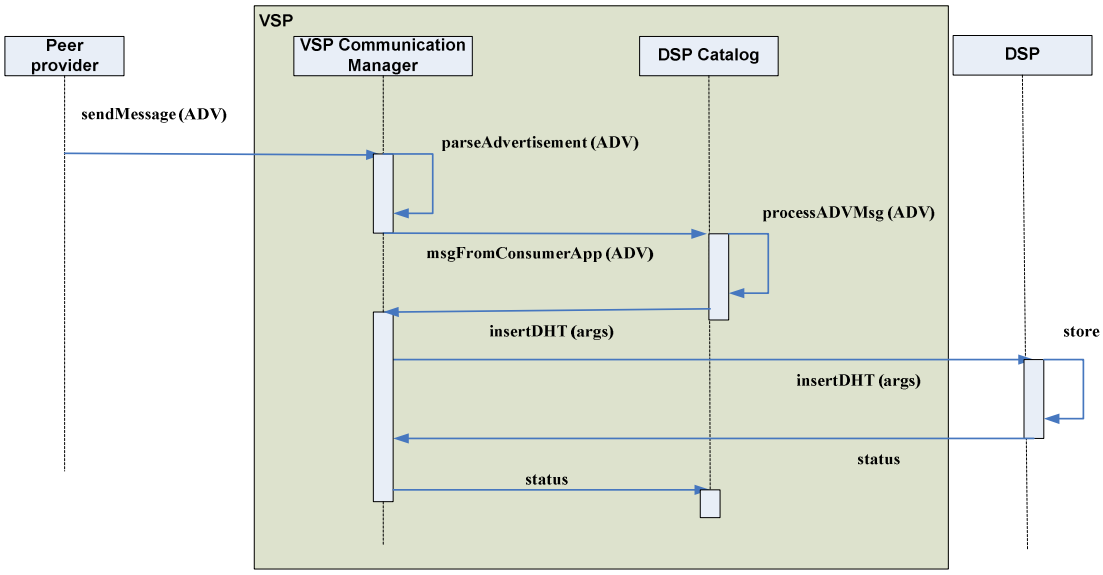

Figura 4-6: Publicação de um conteúdo

Ainda sobre o anúncio de conteúdo, é importante comentar como que as informações relacionadas ao anúncio são armazenadas no DSP. Informações como: nome do *peer*, endereço IP (do *peer* que está anunciando o conteúdo), identificador do *pipe* e o conteúdo (o qual é composto pelo nome do conteúdo, número total de fragmento, número do fragmento e *link* para o fragmento anterior e próximo) são armazenados em um "formulário" no formato XML, de modo a permitir que o provedor do conteúdo seja recuperado por ocasião de uma operação de procura e/ou busca por um conteúdo.

B. **Procura por um conteúdo**: esta facilidade oferecida pelo VSP é de grande importância para o serviço proposto por este trabalho, pois é através desta

funcionalidade que o usuário do sistema iniciará a busca por um conteúdo autêntico e íntegro. Um usuário através do *App Mobile*, requisitará ao VSP ao qual está conectado, a procura por um dado conteúdo. Através de uma mensagem JXME (Anexo I), o VSP será acionado para encontrar um conteúdo na DSP (*DHT Service Provide*), conforme mostra o diagrama de sequência ilustrado pela Figura 4-7 Neste diagrama de sequência, é possível ver que, assim como no anúncio de um conteúdo, a mensagem JXME entra no VSP, através do *VSP Communication Manager*, que é responsável por processar a mensagem e encaminhar ao módulo operacional adequado, que para esta funcionalidade é o *Content Manager*. Este, por sua vez, consultará a DSP, em busca do conteúdo requisitado, por intermédio do *VSP Comunication Manager*.

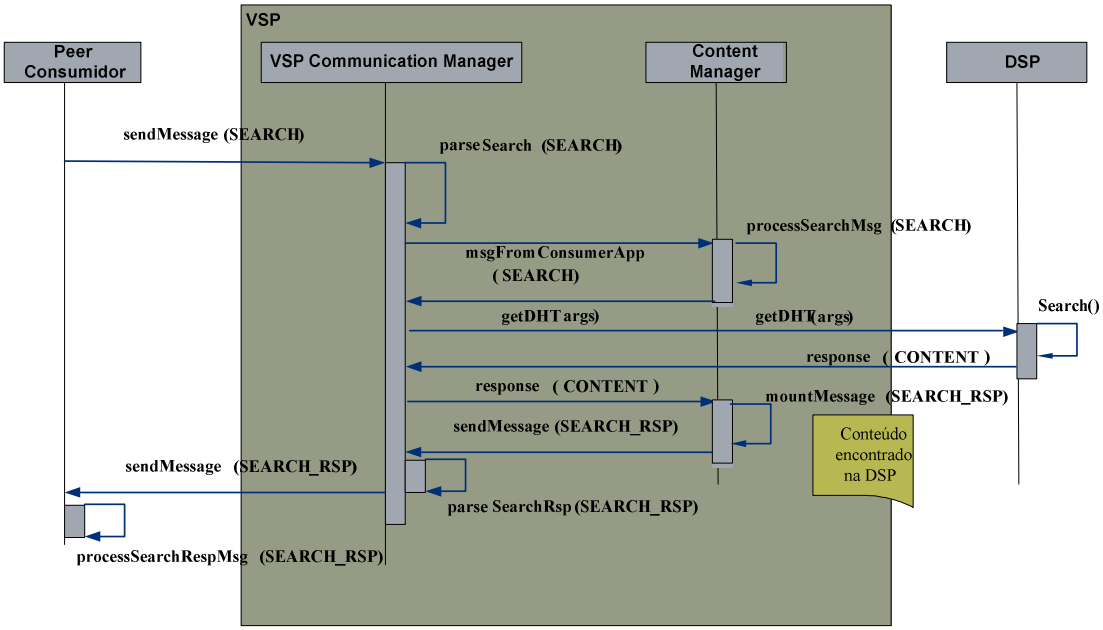

Figura 4-7: Procura por um Conteúdo na DSP

Neste ponto, é importante chamar a atenção do leitor para os resultados possíveis. Na Figura 4-7, está ilustrado o caso positivo, onde o anúncio de um conteúdo é encontrado na DSP e então a resposta à consulta por um conteúdo é enviada ao *peer* solicitante. No entanto, o conteúdo consultado pode não ter sido anunciado no DSP. Neste momento, o VSP deve encaminhar a consulta por um conteúdo ao DIS, conforme mostra o diagrama de seqüência ilustrado pela Figura 4-8. O protótipo faz uso de um cliente da *DHT Kademlia* para acessar a rede P2P tradiconal (DIS) e procurar por conteúdos. Com o resultado da pesquisa na DIS disponível, o *Content Manager* encaminhará esta

informação a outro módulo operacional do *Service Manager* para classificação, o *Classifier*, que faz uso de métodos heurísticos para classificar os conteúdos encontrados na DIS, que serão apresentados como resposta à consulta feita pelo *App Mobile*, onde, através do processo de classificação, o usuário terá um conteúdo com maior chance de ser autêntico e íntegro.

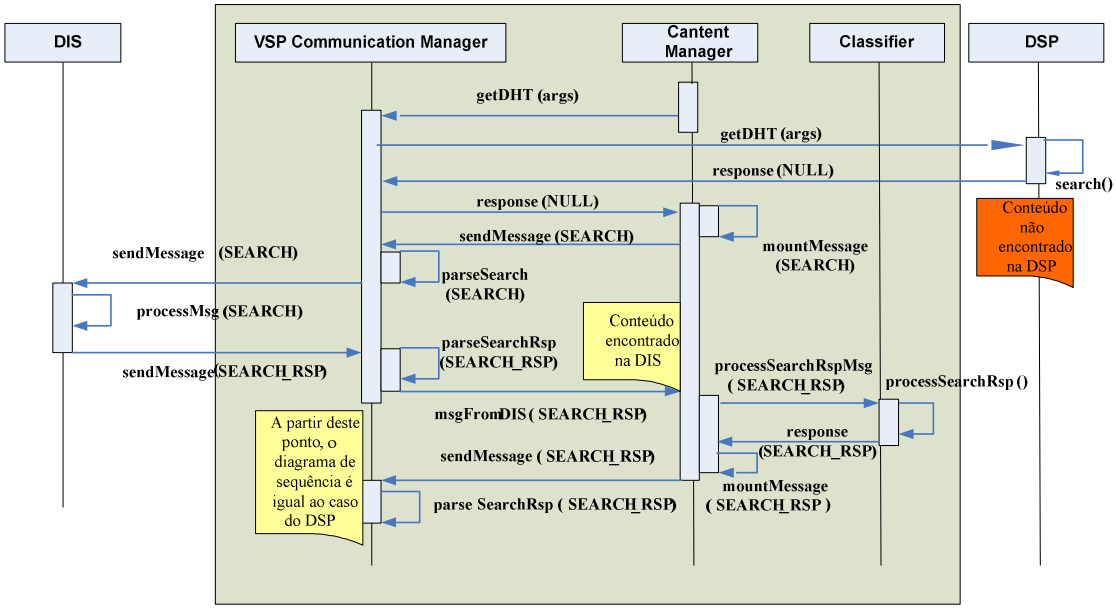

Figura 4-8: Procura por um conteúdo na DIS

Encerrada a procura pelo conteúdo, o VSP encaminha uma resposta a *App Mobile* em forma de mensagem JXME (Anexo I), independentemente do resultado. É importante deixar claro que independente de onde o conteúdo é encontrado, o provedor do conteúdo sempre será o VSP, pois será função dele buscar o conteúdo no provedor real, fazendo desta forma *cache* do conteúdo, fragmentando-o e garantindo assim a alta disponibilidade do mesmo, pois estes fragmentos e informações sobre o processo de busca de um conteúdo serão replicados entre os VSPs vizinhos. Para o *peer* solicitante, este tipo de informação deve ser transparente, ou seja, os malabarismos realizados pelo VSP para garantir conteúdo autêntico e íntegro, não são relevantes para ele.

C. **Busca por um conteúdo**: encontrado o conteúdo, seja na DSP ou na DIS, a App Mobile poderá requisitar a transferência deste conteúdo ao *peer* informado como provedor (VSP ao qual o *peer* está conectado). No capítulo 4 da dissertação, esta operação já foi ilustrada de modo macro, através de diagramas de sequência, os passos

necessários para a busca de um conteúdo, sem o atentado de autenticidade e com atestado de autenticidade respectivamente. Nesta seção, alguns detalhes de implementação serão apresentados para permitir um maior entendimento desta funcionalidade. Como independente de onde o provedor real do conteúdo encontra-se, para o *peer* consumidor, o VSP ao qual está conectado, é o provedor do conteúdo, neste tópico o detalhamento da operação de busca (transferência se concentrará nos mecanismos implementados para transferir o conteúdo do VSP para o *peer* consumidor.

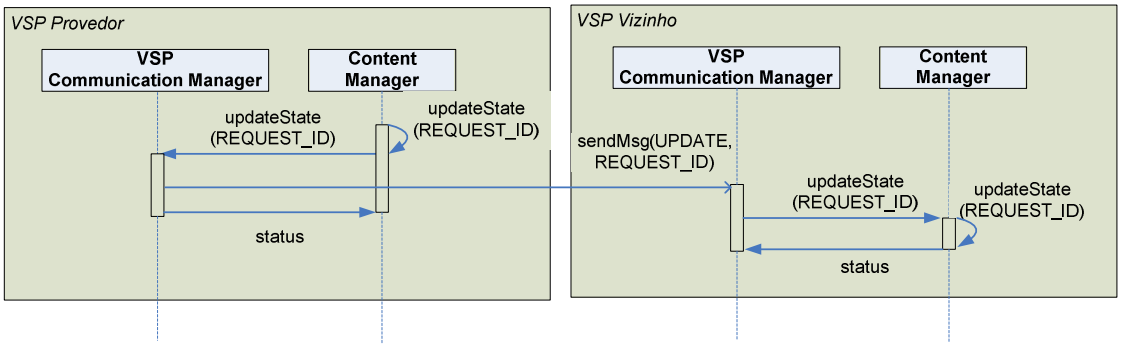

Figura 4-9: Sincronização de estados entre VSP vizinhos

Partindo do princípio de que o módulo *Content Manager* já recebeu a mensagem de requisição de busca de um conteúdo e que o conteúdo já foi devidamente copiado no *cache* do VSP, a Figura 4-10 apresentará um diagrama de sequência das atividades realizadas no VSP para atender a requisição. Porém, antes disto, na Figura 4-9 é apresentado um diagrama de seqüência da operação de sincronização de estados entre um VSP e seu vizinho, pois vale lembrar que além da replicação das tabelas de índices (que é realizada automaticamente como uma funcionalidade nativa da DHT), um VSP replica conteúdo (*cache*) e estados das operações de procura e busca por um conteúdo com o VSP vizinho, o qual é devidamente configurado, através de arquivos de propriedades neste protótipo.

A sincronização de conteúdo (*cache*) acontece de forma similar ao que foi ilustrado para a sincronização de estados.

Estando a requisição devidamente processada no *Content Manager*, e o conteúdo armazenado na *cache* do VSP (mesmo que parcialmente) tem início a transferência dos fragmentos do conteúdo para o *peer* solicitante.

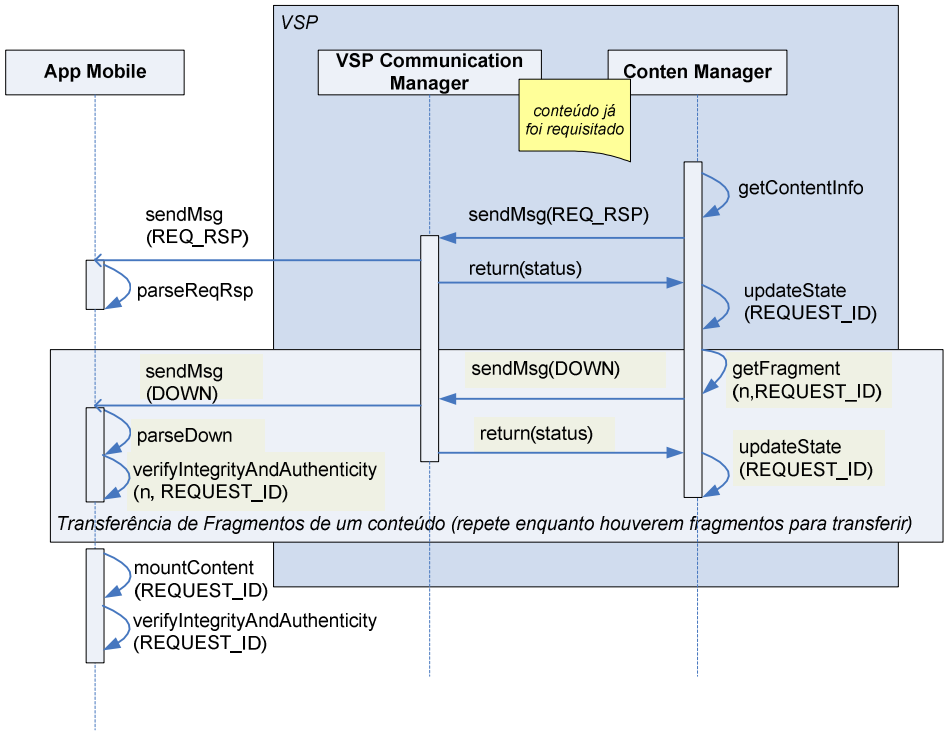

Figura 4-10: Transferência de um conteúdo

Na Figura 4-10,, os passos relevantes da transferência de um conteúdo para o *peer* solicitante são apresentados. O módulo *Content Manager* é responsável pelo gerenciamento desta transferência, determinando os passos a serem executados, bem como a atualização do estado da operação. Inicialmente, o *Contant Manager* obtém as informações relacionadas ao conteúdo que será transferido (*getContentManger*). De posse destas informações, o *Content Manager* envia uma mensagem para o *peer* solicitante (sendMsg (REQ_RSP)), sempre através do módulo *Communication Manager* e identificado pelo atributo REQUEST_ID. Através desta mensagem (Anexo I), o *peer* solicitante é informado de detalhes da transferência do conteúdo solicitado, como por exemplo, o *peer* povedor (que neste caso, sempre será o VSP, ao qual o *peer* solicitante está conectado) e o número de fragmentos em que o conteúdo foi dividido e será transferido. Feito isso, tem início a transferência do conteúdo. vale lembrar que o conteúdo, neste momento, pode estar completamente ou parcialmente armazenado no *cache* do VSP. Se o conteúdo estiver parcialmente armazenado no VSP, significa que em paralelo a esta operação de transferência dos fragmentos do conteúdo para o *peer* solicitante, o VSP estará buscando no(s) *peer(s)* provedor(es) os fragmentos restantes, porém esta operação já foi ilustrada no capítulo 4 da dissertação, e por essa razão, não

será mostrada nesta seção. Voltando ao diagrama ilustrado pela Figura 4-10,, os fragmentos armazenados no *cache* do VSP, que já tiveram sua autenticidade e integridade atestadas (a menos que se tratem de conteúdos migrados da DIS e que não foram verificados através da intervenção humana), serão um-a-um transferidos para o *peer* solicitante (sendMsg(DOWN)). Quando a mensagem JXME (Anexo I) é recebida na *App Mobile*, o fragmento é identificado pelos atributos REQUEST_ID e FRAME_ID. O fragmento recebido tem sua integridade e autenticidade verificadas (verifyIntegrityAnd Authenticity(n, REQUEST_ID)), para evitar que o conteúdo tenha que ser transferido em sua totalidade e só então, o usuário possa fazer está verificação. Este é um dos pontos positivos deste serviço proposto e um diferencial em relação às redes P2P tradicionais. Em caso de necessidade de retransmissão de um fragmento, o *peer* solicitante enviará a mensagem de requisição de um conteúdo (Anexo I) com o campo FRAME_ID preenchido com o fragmento a ser re-enviado. Na Figura 4-10 as operações de sincronização de estados da transferência de um conteúdo são ilustradas (updateState(REQUEST_ID)). Esta sincronização é realizada para que, em caso de problema com o VSP que está realizando a transferência de conteúdos, o VSP vizinho possa assumir suas atividades no ponto exato em que elas foram interrompidas, ou seja, no caso da transferência de um conteúdo, o VSP substituto retomará as atividades, a partir do ponto em que o VSP com problemas parou e o *peer* consumidor não perceberá a mudança de VSP. É claro que, em caso do *peer* consumidor tentar enviar uma mensagem para o VSP, será necessário que o mecanismo de lookup (Código Fonte 4-2) aponte para o novo VSP.

Com o final da transferência dos fragmentos, a *App Mobile* montará o conteúdo, a partir dos fragmentos enviados, e a integridade e autenticidade do conteúdo completo será verificada. Com isso, uma operação de transferência de conteúdos é concluída.

III. Classificação de conteúdo proveniente da DIS

A classificação dos conteúdos encontrados na DIS, nesta prototipação da proposta, é realizada através do comandoaplicação "*diff*" do LINUX que compara os conteúdos indicando as diferenças entre eles e baseado nestas diferenças, os conteúdos vão sendo agrupados, até que através de múltiplas iterações, entenda-se que a classificação seja encerrada. Estes agrupamentos de conteúdos são apresentados como resultado da consulta a um conteúdo.

E assim é encerrada a apresentação detalhada da implementação do protótipo, focando os principais módulos funcionais do sistema proposto.

Em seguida, serão apresentados os principais cenários para o trabalho proposto, através dos quais será possível avaliar mais detalhadamente a sequencialidade dos passos executados durante a busca de um conteúdo no serviço proposto.

4.6. Cenário

Assume-se que um conteúdo pode ser ou não fragmentado em várias partes e armazenado em vários *peers*, para permitir o paralelismo no acesso ao conteúdo e, consequentemente o ganho de desempenho. No registro do conteúdo, há um *hash* do conteúdo como um todo e *hash* das várias partes. Em um momento inicial, os *peers* da rede tradicional fazem a publicação de seus conteúdos na DIS (*DHT Index Service*), que será oportunamente descrita. Sobre este momento inicial, é importante ressaltar que os conteúdos ainda não tiveram suas autenticidades atestadas.

O modelo dinâmico será representado, neste trabalho, através de 3 diagramas de sequência que serão apresentados a seguir. O primeiro deles representa a busca de um conteúdo sem autenticidade atestada, onde a DSP propagará a requisição de busca de um conteúdo à rede tradicional, conforme mostra a Figura 4-11.

A BUSCA - Um *peer* cliente envia uma requisição de busca de um conteúdo desejado a um VSP, que faz uma consulta na DSP e em caso de fracasso na busca nesta DHT para conteúdos com garantia de autenticidade e integridade, a mesma consulta é encaminhada/propagada à DIS. O resultado da busca será apresentado ao cliente, que terá consciência de que o conteúdo é ou não autêntico. Todo conteúdo sem garantia de autenticidade que trafega através destes VSPs será registrado em um arquivo de histórico, para que o administrador da DSP possa ter conhecimento da existência deste conteúdo na rede e que, posteriormente, possa analisar o mesmo e atestar a sua atenticidade. Feito isto, este conteúdo poderá ser catalogado na DSP.

O Download – Apresentado o resultado de uma busca ao *peer* cliente, o VSP sempre identificará um VSP como fonte do conteúdo, exercendo assim, o papel de *proxy* durante a transferência do conteúdo desejado, pois em caso de queda da verdadeira fonte do conteúdo, o VSP, buscará novas fontes, de modo que o *peer* cliente tenha a impressão de que a conectividade não tenha sido perdida, além do que, uma das características de implementação

do VSP será a criação de *cache*, ou seja, a partir do momento que o cliente selecionar o conteúdo, o VSP que está atendendo o pedido, passará a transferir o conteúdo, criando um buffer deste conteúdo para melhorar o desempenho da transferência ou simplesmente, para minimizar o impacto, em caso de queda da fonte de origem do conteúdo. Se por outro lado, for o VSP que seja desconectado, por qualquer que seja a razão, na tentativa de re-estabelecer a conexão com o VSP original, um VSP vizinho assumirá o seu lugar e através de uma mensagem do cliente, indicando o ponto de parada da transferência, o novo VSP será capaz de retomar o serviço no ponto em que houve a interrupção, pois recordando, uma das características de uma DHT é replicar as tabelas de chaves de responsabilidade de um nó, entre os nós vizinhos

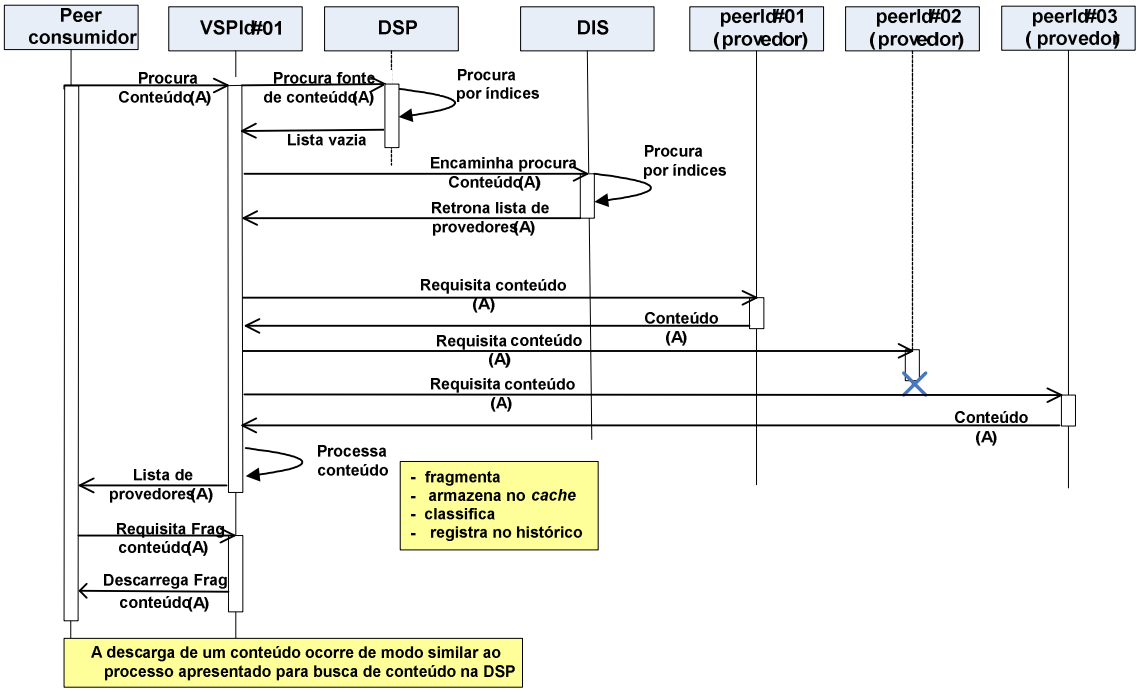

Figura 4-11: Busca de um conteúdo sem atestado de autenticidade

O atestado de Autenticidade – O atestado de autenticidade, neste sistema, será feito de forma manual e efetuado por um ser humano, ou seja, o administrador do serviço avaliará o conteúdo para qualificá-lo e uma assinatura digital será atribuída a este conteúdo que será então cadastrado na DSP, conforme ilustra a Figura 4-12, e descreve o diagrama de seqüência da Figura 4-13.

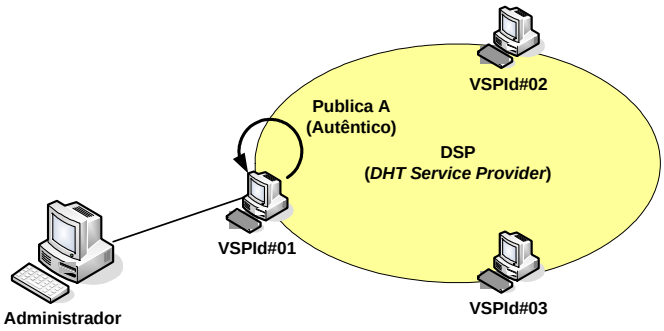

Figura 4-12: Atestado autenticidade de um conteúdo

Os conteúdos que tiverem suas autenticidades atestadas terão, simultaneamente, sua integridade garantida através de funções *hash*, por exemplo, que integrará o conjunto de informações a serem armazenadas, juntamente com a chave na DSP.

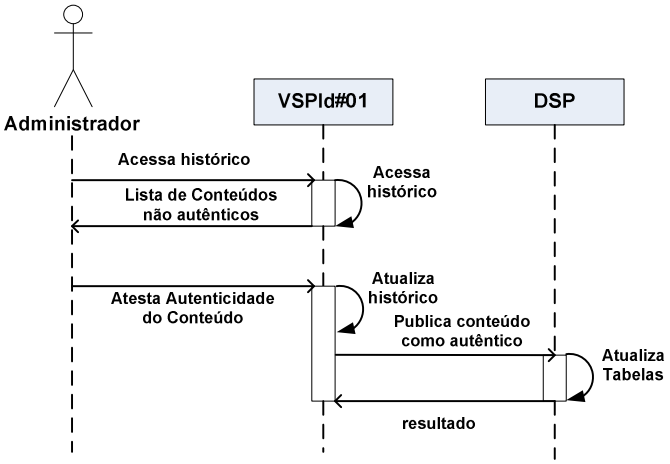

Figura 4-13: Diagrama - Atestando a autenticidade de um conteúdo

A Figura 4-14 mostra o diagrama de sequência para uma busca de conteúdo que tem sua autenticidade atestada e consequentemente, catalogado na DSP. Se comparado com o diagrama da Figura 4-11, é possível verificar que o VSP realiza busca do conteúdo na DSP e apresenta ao *peer* cliente o resultado obtido, tornando desnecessário propagar a busca à DIS. Adicionalmente, a DSP passa a validar a autenticidade atestada ao conteúdo durante o procedimento de transferência do conteúdo para *peer* cliente.

Mecanismo de Classificação de Conteúdos - Este serviço implementará um mecanismo que permitirá a classificação dos conteúdos sem atestado de autenticidade, durante o procedimento de busca, de modo a comparar os conteúdos e/ou seus fragmentos, para que durante a apresentação dos resultados, os mesmos sejam tabelados por grupos de

similaridade e pela ocorrência e então apresentados ao cliente, que poderá ter alguns parâmetros para decidir entre os conteúdos não autênticos apresentados, mesmo sabendo que estes resultados não implicam no atestado de autenticidade dos mesmos, ou mesmo que um grupo de conteúdos que possua maior incidência dentro da rede é mais ou menos íntegro/autêntico que um outro com menor incidência . Estes mecanismos não implementam qualquer espécie de lista de reputação, sejam elas brancas ou cinzas. A única função destes mecanismos é classificar os conteúdos por similaridades/frequência e apresentar ao cliente os resultados tabulados e ao mesmo tempo, estes dados serão armazenados em um arquivo de histórico para futuro uso do administrador do VSP, para a avaliação e atribuição de uma chave que ateste a autenticidade do grupo de conteúdos.

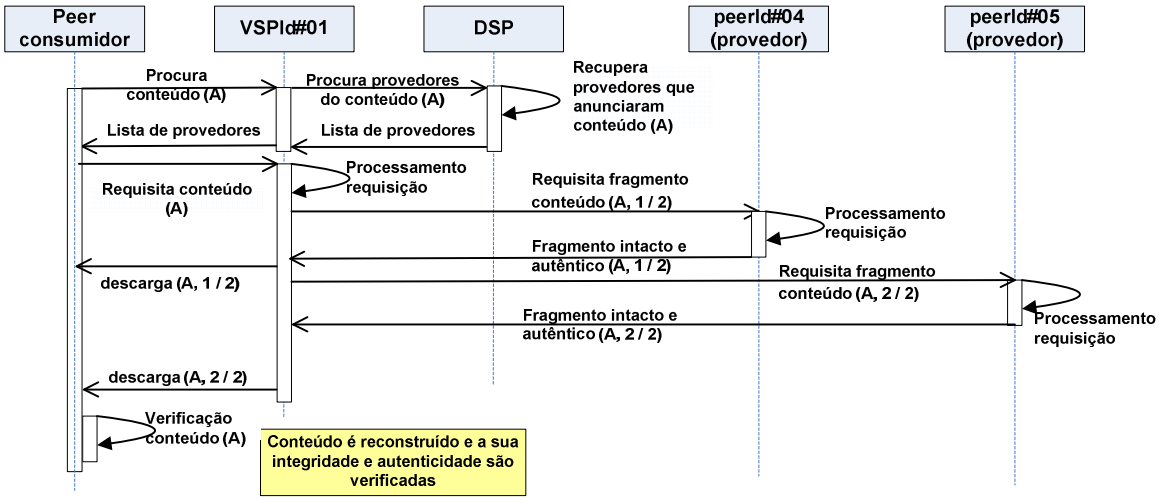

Figura 4-14: Busca de um conteúdo com atestado de autenticidade

A seguir, será apresentada a avaliação da solução proposta por esta dissertação, através dos resultados obtidos com o protótipo.

4.7. *Conclusão*

Neste capítulo foi apresentado o serviço proposto para garantir a integridade e autenticidade dos conteúdos, bem como a alta disponibilidade de conteúdos e *peers* provedores em redes P2P, além de propiciar a migração de conteúdos de redes P2P tradicionais para o serviçõ proposto.

O protótipo implementado para avaliar a solução proposta também foi apresentado neste capítulo e os resultados obtidos, a partir deste protótipo, serão avaliados na próxima seção.

Capítulo 5

Avaliação do Protótipo

5.1. Descrição do Cenário

Conteúdos compartilhados de mídia, como vídeo, áudio e outros, são muito comuns em redes P2P tradicionais. Entretanto, não são fornecidas garantias de integridade e autenticidade dos conteúdos. O objetivo principal desta proposta é prover estas garantias junto com a disponibilidade provida, através de um serviço, o VSP.

Um VSP poderia, por exemplo, prover um serviço para conteúdos compartilhados sobre Beethoven. Quando um *peer* consumidor procurasse por um conteúdo compartilhado através dessa palavra chave, receberia como resposta um conjunto de arquivos de áudio e vídeo, providos por um VSP da DSP. Este exemplo é simples, mas existem muitos outros cenários onde esta abordagem poderia ser aplicada.

A Figura 4-14 ilustra o comportamento do sistema quando os conteúdos compartilhados são fornecidos por uma rede orientada a serviço (DSP). O cenário é baseado na arquitetura apresentada na Figura 4-1. O cenário assume que os conteúdos compartilhados, após serem descarregados pela primeira vez, serão fragmentados e a integridade de cada um desses fragmentos, assim como do arquivo de conteúdo como um todo, serão assegurados por uma função de *hash*. A assinatura digital do serviço provido pelo VSP garante a autenticidade dos fragmentos de conteúdo e do conteúdo do arquivo, como um todo.

Cada fragmento do conteúdo compartilhado pode ser armazenado por diferentes provedores (na Figura 4-14, peerId#04 e peerId#05), desde que cada um seja indexado pela DSP. Quando um consumidor procurar por algum conteúdo compartilhado, o VSP localiza-o

na DSP e busca uma lista de provedores para cada fragmento que compõe o conteúdo desejado. O VSP obtém e provê os fragmentos um a um para o consumidor, que pode verificar individualmente a integridade e autenticidade dos fragmentos. Uma cópia de um fragmento pode ser armazenada em vários provedores diferentes. Depois de receber todos os fragmentos de um determinado conteúdo compartilhado, o consumidor reconstrói o conteúdo original, juntando todos os fragmentos na ordem correta. Então, o consumidor verifica a integridade do conteúdo e a assinatura do VSP, a fim de garantir a autenticidade do mesmo.

Os conteúdos são fragmentados para permitir a alta disponibilidade, adicionalmente se o *peer* consumidor tiver largura de banda suficiente, estará habilitado a fazer a descarga de vários fragmentos simultaneamente. Na Figura 4-14, por exemplo, os fragmentos (A, 1/2) e (A, 2/2) do conteúdo requisitado poderiam ser transferidos em paralelo.

Em uma rede P2P tradicional, mesmo que um consumidor obtenha diferentes fragmentos de diferentes provedores, não existe a garantia de que todos sejam cópias de fragmentos de um mesmo conteúdo compartilhado. Consequentemente, a ocorrência de corrupção de conteúdos é muito comum na descarga de fragmentos em paralelo, em redes P2P tradicionais, pois um mesmo fragmento de um conteúdo requisitado a fontes diferentes, pode ser distinto.

A Figura 4-11 apresenta o comportamento do sistema, no caso da requisição de um conteúdo que não se encontra disponível na rede orientada a serviço, ou seja, o conteúdo desejado está disponível na rede P2P tradicional. Este cenário é baseado na arquitetura apresentada na Figura 4-1. O cenário assume que os conteúdos são fornecidos por provedores obtidos pelo VSP como resultado de uma consulta a DIS (serviço de índice para a rede P2P tradicional). Os conteúdos são descarregados de diferentes provedores, podendo ocorrer cópias diferentes do mesmo conteúdo. Na prática, diferentes arquivos de um mesmo conteúdo são descarregados de diferentes provedores em paralelo. Ao final do processo de descarga, os arquivos com maior grau de similaridade serão disponibilizados no VSP. A integridade de cada arquivo é verificada em relação ao código *hash* publicado na DIS. Então, a classificação heurística seleciona o conteúdo com maior probabilidade de ser íntegro e autêntico. A Figura 4-11 ilustra a migração de conteúdos de uma rede P2P tradicional (DIS, peerID#01, peerID#02 e peerID#03) para a rede P2P orientada a serviço, representada pelo VSPId#01.

Depois da descarrega do conteúdo, o histórico para aquele conteúdo é atualizado de acordo com sua classificação e o arquivo de conteúdo selecionado é oferecido ao consumidor

pelo VSP. O objetivo do histórico é ajudar a heurística na definição de qual conteúdo tem mais probabilidade de ser autêntico. Se o *peer* consumidor concordar com a provisão de um conteúdo sem garantia de integridade e autenticidade, que tenha sido classificado automaticamente e descarregado de uma rede P2P tradicional, então poderá iniciar a descarga do conteúdo com a intermediação do VSP. É importante ressaltar que o *peer* consumidor não acessará o *peer* provedor na DIS para descarregar esse conteúdo. Todo o processo de descarga ocorrerá com a intermediação do VSP, que para o consumidor é o provedor do conteúdo.

Após o registro da descarga do conteúdo a partir da DIS no histórico, em algum momento, a intervenção humana deverá acontecer para que o conteúdo tenha certificação da autenticidade. Somente neste caso, o conteúdo concluirá a sua migração para a rede P2P orientada a serviço, com sua publicação como serviço na DSP. Em outras palavras, quando algum conteúdo for anunciado como um serviço, esse será armazenado na forma fragmentada na *cache* do VSP, para que os demais *peers* consumidores possam fazer a descarga do mesmo, conforme detalhado no cenário anterior. O *peer* consumidor que fizer a descarga deste conteúdo, também poderá anunciar a si próprio como provedor de todo o conteúdo ou de fragmentos deste. Assim, é encerrado o processo de migração de um conteúdo da DIS para a DSP.

Deve-se observar que os conteúdos podem se tornar disponíveis na rede P2P orientada a serviço por livre escolha do VSP ou provocada pela requisição do consumidor. Em ambos os casos, o procedimento é o mesmo que foi apresentado na Figura 4-11. Entretanto, quando o consumidor requisita um conteúdo que não está disponível na DSP, o atraso no tempo de resposta poderá ser transferido ao *peer* consumidor, pois o VSP deverá executar as ações mencionadas na Figura 4-11, antes que o conteúdo requisitado seja disponibilizado ao *peer* consumidor.

A fragmentação também é usada nesta proposta para minimizar o efeito negativo do consumo de largura de banda da abordagem tradicional, onde só é possível descobrir que o conteúdo é falso ou corrompido, quando a descarga completa do mesmo, tenha sido finalizada. Uma vantagem de nossa proposta é que cada um dos fragmentos pode ter sua integridade e autenticidade verificadas com finalização de sua descarga.

5.2. Resultados obtidos com o protótipo

Conforme previamente apresentado no capítulo da dissertação destinado à avaliação

do protótipo, mais especificamente avaliação dos resultados, foi montado em laboratório, um ambiente controlado para validar, quantitativamente ,a solução proposta pela dissertação.

Na Figura 5-1 (também disponível na dissetação), este cenário montado em laboratório, é ilustrado.

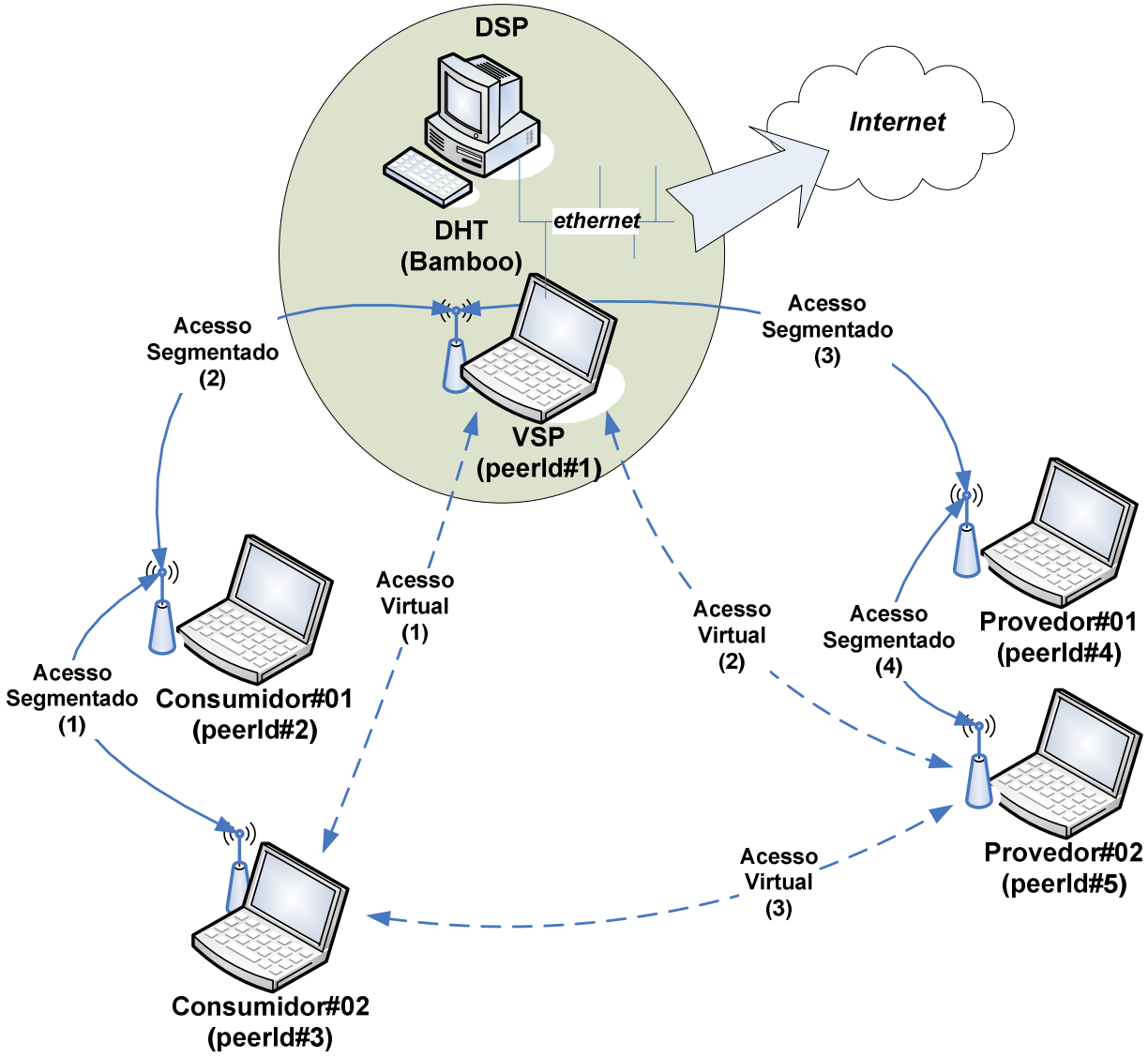

Figura 5-1: Cenário de avaliação da proposta

Este capítulo é destinado a compilar os resultados quantitativos obtidos através de ensaios do cenário ilustrado pela Figura 5-1, servem como embasamento para as fórmulas apresentadas no capítulo avaliação dos resultados.

Inúmeros ensaios foram realizados, mas iniciaremos os estudos, baseados nos

resultados obtidos no experimento onde o *peer* Consumidor#01 solicita à DSP um conteúdo anunciado pelo *peer* Provedor#01, ou seja, a comunicação entre os *peers* consumidor e provedor e o VSP tem a "distância" de um *hop*, ou seja, não existem *peers* intermediários na comunicação do VSP com os *peers* consumidor e provedor.

Na operação de anúncio de um conteúdo, a mensagem JXME de anúncio enviada pelo *peer* Provedor #01 percorre o caminho ilustrado pela seta denominada de Acesso Segmentado (3) para contactar o VSP, o que correspondendo a um consumo de tempo $t+Pt$, onde P_t é tempo de processamento da mesnagem . Para o *peer* Provedor #02 fazer o anúncio de um conteúdo, o tempo gasto seria $(2*t)+ P_t$. Por analogia, se houvessem *N hops* entre o *peer* provedor e o VSP, o tempo consumido seria dado pela equação $(N*t)+ P_t$.

Para a operação de procura por um conteúdo na DSP, para o mesmo cenário descrito na operação anterior, o tempo consumido seria: $(2*t)+ P_t$ ou para múltiplos *hops* $(2*t*N)+ P_t$, pois esta operação é baseada em *request-response*, ou seja, o *peer* consumidor necessita de uma mensagem de resposta para dar como encerrada a operação.

Para a descarga ou busca de um conteúdo, o tempo consumido seria:

- Mensagem de resposta à requisição: $(2*t)+ P_t$ e $(2*t*N)+ P_t$,
- Mensagem de descarga de fragmentos:$[(t+t')+ P_t]*NF$ *e* $[((t+t')*N)+ P_t]*N_F$, onde N_F é o número de fragmentos. O termo: $(t+t')$ deve ser substituído por $(2*t)$ quando for usado um único tipo de mensagem (protocolo) para o envio de fragmentos de conteúdo do provedor para o VSP e do VSP para o consumidor;

Numericamente, os dados imediatamente acima formulados para cada uma das operações, serão tabelados a seguir e cabe lembrar, que os valores mostrados foram obtidos através da média ponderada dos resultados conseguidos em vários ensaios. A diferença nos tempos obtidos para o envio das diversas mensagens JXME se deu, principalmente, devido a diferença no tamanho das mensagens (formato das mensagens foi descrito no Anexo 1), bem como o tráfego de dados na rede.

Na Tabela 5-1 estão registrados os tempos (média ponderada) obtidos durante a operação de anúncio de um conteúdo. Além do tempo de envio de uma mensagem de anúncio, foi computado o tempo de processamento no VSP.

Anúncio de um conteúdo – Mensagem JXME - (ms)	*Tempo de processamento da Mensagem no VSP (ms)*
~ 800	~ 20

Tabela 5-1: Anúncio de um conteúdo

O tempo médio para a realização de uma consulta por um conteúdo está armazenado na Tabela 5-2. para esta tabela, as informações relevantes são o tempo de envio da mensagem de consulta por um conteúdo, o tempo de processamento da informação no VSP e tempo de envio da mensagem de resposta, ou seja, é a contabilização do tempo desde que a mensagem sai do *peer* consumidor, até o momento em que ela volta ao mesmo *peer*.

Consulta por um conteúdo – Mensagem JXME - (ms)	*Tempo de processamento da Mensagem no VSP (ms)*	*Resposta à consulta – Mensagem JXME - (ms)*
~ 600	~ 20	~ 500

Tabela 5-2: Consulta por um conteúdo

Na **Tabela 5-3**, por sua vez, foram registrados os tempos para requisição por um conteúdo, anteriormente consultado. Nesta operação, os tempos relevantes são: a) tempo de envio da mensagem de requisição por um conteúdo, b) o tempo de processamento no VSP e c) tempo de envio da mensagem de resposta. Todas estas mensagens estão detalhadamente descritas no Anexo I.

Requisição de descarga de um conteúdo – Mensagem JXME - (ms)	*Tempo de processamento da Mensagem no VSP (ms)*	*Resposta à consulta – Mensagem JXME - (ms)*
~ 1000	~ 20	~ 500

Tabela 5-3: Requisição de descarga de um conteúdo

Na Tabela 5-4 estão registrados os tempos obtidos em um ensaio na operação de transferência de um conteúdo do *peer* provedor, através do VSP para o *peer* consumidor.

Número do Fragmento	*Tempo de envio do Fragmento do provedor para o VSP Engine – http socket - (ms)*	*Tempo de processamento da Mensagem no VSP (ms)*	*Tempo de envio do Fragmento do VSP para o peer requisitante /consumidor – Mensagem JXME - (ms)*
01	32	15	344
02	38	32	735
03	38	20	1500
04	38	16	656
05	54	31	2000
06	38	23	1047
07	38	16	2765
08	38	16	2047
09	38	16	1110
10	38	19	2500
Tempo médio (média aritimética das medidas)	**39**	**20.4**	**1470.4**

Tabela 5-4: Descarga de um conteúdo

Utilizando o *emule* conectado à rede *kademlia*, foi feita a descaga da aplicação "*arg.exe*" (*freeware*), para verificar o tempo gasto com esta operação. A referida aplicação tem 148 KB (ou 152.064 bytes) de tamanho e o tempo necessário para concluir a descarga foi de aproximadamente 30 segundos. Se aplicarmos a mesma regra que o VSP usa para fragmentar os conteúdos, ou seja, dividir o conteúdo em fragmentos com tamanho máximo de 1KB (1024 bytes), este conteúdo seria dividido em 149 fragmentos. Então, dividindo o tempo gasto para realizar a descarga deste conteúdo (~30s) pelo número de fragmentos de tamanho máximo 1KB (149 fragmentos), chegamos ao tempo gasto (estimado) para baixar cada um dos fragmentos da aplicação "arj.exe", que ficou na ordem de 200ms.

Valor de N	*Anúncio de um conteúdo – Mensagem JXME - (ms) (N*t), t=500*	*P_t = Tempo de processamento da Mensagem no VSP (ms)*
1	500	~ 20
2	1000	~ 20
3	1500	~ 20
4	2000	~ 20

Tabela 5-5: Anúncio de um conteúdo (*N*=1...4)

Baseado nos tempos apresentados nas tabelas 1, 2, 3 e 4 e se considerarmos cenários

com valores diferentes para *N* e t = 500ms para mensagens JXME e t = 40ms para mensagens http socket, e aplicando a formulação anterior teríamos as tabelas abaixo.

Na Tabela 5-5 foram armazenados os tempos relativos a uma operação de anúnico de um conteúdo, considerando diferentes número de *peers* (*N= 1..4*) e tempo *t* igual a 500 ms

Na Tabela 5-6 foram armazenados os tempos relativos a uma operação de consulta por um conteúdo, considerando diferentes números de *peers* (*N= 1..4*) e tempo *t* igual a 500 ms.

Valor de *N*	**Consulta por um conteúdo – Mensagem JXME - (ms)** *2*t**N), t=500*	**P_t = Tempo de processamento da Mensagem no VSP (ms)**
1	1000	~ 20
2	2000	~ 20
3	3000	~ 20
4	4000	~ 20

Tabela 5-6: Consulta por um conteúdo (*N*=1...4)

A operação de requisição de descarga de um conteúdo estão armazenados na Tabela 5-7 onde são considerados diferentes número de *peers* (*N= 1..4*) e tempo *t* igual a 500 ms.

Valor de *N*	**Requisição de descarga de um conteúdo – Mensagem JXME - (ms)** *(2*t**N), t=500*	**P_t = Tempo de processamento da Mensagem no VSP (ms)**
1	1000	~ 20
2	2000	~ 20
3	3000	~ 20
4	4000	~ 20

Tabela 5-7: Requisição de descarga de um conteúdo (*N*=1...4)

Para a Tabela 5-8 é importante ressaltar que existem valores diferentes para t, devido ao uso de diferentes protocolos usados nesta operação (JXME e http, ambos sobre IP).

Valor de *N*	**Tempo de envio de 1 (um) Fragmento (ms)** $(t+t')*N$, $t = 500$ (JXME) e $t' = 40$ (http)	**P_t = Tempo de processamento da Mensagem no VSP (ms)**	**Tempo total de envio de 10 (N_F) fragmentos (ms)** $[((t+t')*N)+ P_t]*N_F$ $t = 500$ (JXME) e $t' = 40$ (http)
1	540	~ 20	5600
2	1080	~ 20	11000
3	1620	~ 20	16400
4	2160	~ 20	21800

Tabela 5-8: Descarga de um conteúdo (*N*=1...4)

Com base na formulação matemática obtida a partir de experimentos realizados em laboratório e nas tabelas acima apresentadas, fica fácil determinar o tempo necessário para que uma descarga de conteúdos seja feita utilizando a solução proposta.

5.3. Avaliação dos Resultados

Para avaliar o cenário descrito na seção 5.1. , o protótipo desenvolvido foi apresentado na seção 4.4. e avaliado em laboratório em um ambiente controlado, conforme ilustra a Figura 5-1.

Este cenário, montado em laboratório, esteve composto por um conjunto de máquinas equipados por uma interface de rede *wireless* (sem fio), que foram configuradas para atuar no modo *ad hoc*, através do qual as características típicas de uma rede *ad hoc* puderam ser reproduzidas, permitindo maior realismo durante os experimentos. Os resultados obtidos durante os experimentos, bem como dados calculados a partir de formulação obtida estão disponíveis na seção anterior.

O objetivo do experimento foi medir o impacto que a introdução do VSP causa no tempo de resposta para o *peer* consumidor. As instabilidades da conexão de rádio de MANet não foram levadas em consideração, por não ser nosso objetivo de avaliação. O número de topologias possíveis que podem produzir resultados conclusivos da avaliação é muito variado. Porém, para este experimento foi adotado um cenário híbrido de estruturação da topologia da rede, composto de uma rede sem fio em modo infra-estruturado para DSP e de uma MANet para os *peers* consumidores e provedores (modo *ad hoc*). A DSP não está em modo *ad hoc*, pois de acordo com [CRAM2006] o comportamento de uma DHT sobre MANet é imprevisível.

A MANet é composta de ***N*** *laptops* (no cenário apresentado pela Figura 5-1 ***N*=5**) conectados uns ao outros em modo *ad hoc*. Na verdade, há $\lceil N/2 \rceil$ laptops agindo como provedores e $\lceil N/2 \rceil$ como consumidores e todos os ***N*** laptops estão interconectados, cada um com diferente nível de sinal de rádio, mas somente há um laptop provedor e um laptop consumidor conectado a DSP.

A rede DSP, baseada em ponto de acesso (modo infra-estrutura), instancia alguns VSPs, todos com acesso à internet. Ou seja, o VSP pode conectar-se à rede *Kademlia* diretamente através do acesso à *Internet*, representado na Figura 5-1. Os conteúdos foram distribuídos entre os provedores, de modo que, em média, há $\lceil N/2 \rceil$ *hops* entre os *peers* consumidor e provedor na MANet e no pior caso *N*-1 *hops* desconsiderando o VSP. Assumimos que o tempo t_t para uma mensagem trafegar entre um nó e seus vizinhos é constante. Assim sendo, o tempo total dt_M para um *peer* consumidor fazer a descarga de um conteúdo diretamente de um *peer* provedor na Manet, é dado por:

$$dt_M = \lceil N/2 \rceil * t_t \qquad \text{(Eq. 1).}$$

Em nossas medidas, observamos que o processamento no VSP introduz um o*verhead* de tempo t_p no tempo de resposta para o *peer* consumidor, para ambas as operações: procura a descarga de conteúdos, se comparado com a conexão direta entre os *peers* consumidor e provedor sobre MANet. Consequentemente, quando um *peer* consumidor requisita um conteúdo para o VSP e este encontra-se armazenado em *cache*, o tempo de descarga é:

$$dt_c = dt_M + t_p \quad \text{(Eq. 2)}$$

No entanto, quando o conteúdo requisitado não se encontra em *cache*, e este deve ser transferido a partir do *peer* provedor até o VSP e posteriormente fornecido ao *peer* consumidor, o atraso é aproximadamente $I_1 * t_t$, onde *I* é um fator que depende do meio onde a mensagem está trafegando. Ou seja:

$$dt_p = dt_M + t_p + I_1 * t_t \text{ (Eq. 3).}$$

Por outro lado, quando um conteúdo não é encontrado no VSP e será necessário buscá-lo na DIS, o *overhead* para disponibilizar o conteúdo na *cache* do VSP é equivalente a $I_2 * t_t$, ou seja:

$$dt_D = dt_M + t_p + I_2 * t_t \text{ (Eq. 4).}$$

Com base nos dados obtidos na seção 5.2. , é possível calcular os valores dos fatores I_1

e I_2. O valor para o fator I_1 pode ser obtido de duas formas diferentes, baseado no protocolo usado para obter os fragmentos no *peer* provedor:

1. troca de mensagens via http socket: se a troca de mensagens entre o VSP e o *peer* provedor se der através de http socket, o valor de I_1 é obtida a partir dos dados tabulados na Tabela 5-4 e destacados imediatamente abaixo:
   A_{http} = atraso via http socket,
   t_{http} = tempo de transmissão de uma mensagem via http socket,
   A_{http} = 2x t_{http} => tempo para envio de uma mensagem de requisição de um fragmento + o tempo de envio da mensagem de resposta à requisição,

   t_{http} = 39ms, A_{http} = 2x t_{http} = 2 x 39ms = 78ms

   t_t = 500ms (tempo obtido no experimento e tabulado na Tabela 5-3)
   $I_1 = A_{http} / t_t$ => I_1 = 78ms/500ms = 0,156 ;

2. troca de mensagens via JXME: se a troca de mensagens entre o VSP e o *peer* provedor se der através de JXME, o valor de I_1 é obtida a partir dos dados tabulados na Tabela 5-3 e destacados imediatamente abaixo:
   A_{JXME} = atraso via mensagens JXME,
   t_{JXME} = tempo de transmissão de uma mensagem via mensagens JXME,
   A_{JXME} = 2x t_{JXME} => tempo para envio de uma mensagem de requisição de um fragmento + o tempo de envio da mensagem de resposta à requisição,

   t_{JXME} = 500ms, A_{JXME} = 2x t_{JXME} = 2 x 500ms = 1000ms

   t_t = 500ms (tempo obtido no experimento e tabulado na Tabela 5-3)
   $I_1 = A_{JXME} / t_t$ => I_1 = 100ms/500ms = 2 ;

O valor para o fator I_2 , por sua vez, é obtido a partir de dados provenientes da operação de busca de conteúdo, utilizando um cliente do *emule* (detalhado na seção 5.2.) será calculado a seguir:

conteúdo (arj.exe) => 148 KB (ou 152.064 bytes) => tempo de download = 30s

- 149 fragmentos (1KB cada fragmento) = 30s
- 1 fragmento = (tempo total / número fragmentos) = (30s / 149 fragmentos)
- 1 fragmento = 200ms/fragmento

A_{emule} = atraso provocado pelo *emule*,

$A_{emule} = (I_2 \text{ x } t_t) \Rightarrow I_2 = (A_{emule} / t_t) \Rightarrow I_2 = (200\text{ms} / 500\text{ms}) \Rightarrow I_2 = 0{,}4$

Os valores para $I_1 = 2$, $I_2 = 0{,}4$ (calculados imediatamente acima,) e o tempo t_t = 500ms, que representa 50% do tempo para conectar o *peer* consumidor ao *peer* provedor através de $\lceil N / 2 \rceil$ *hops* em MANet, dt_M.= 2* t_t =1000ms. Observamos também que $t_t = 25 * t_p$ = 20ms.

A seguir, será comparada a aquisição de um conteúdo, por meio do VSP, com a mesma aquisição, conectando-se o *peer* consumidor ao *peer* provedor em MANet (modo puramente *ad hoc*). Assim, quando o *peer* consumidor obtém um conteúdo a partir da DIS, é imposto um tempo de atraso de 4,4%, obtido através da Eq. 4, conforme mostrado a seguir:

$dt_D = dt_M + t_p + I_2 * t_t$, onde I_2=0,4 ou 2/5 e $t_p = (1/25) * t_t$

$dt_D = dt_M + (1/25) * t_t + (2/5) * t_t$

$dt_D = dt_M + t_t * (1/25 + 2/5) = dt_M + t_t * ((1+10)/25)$

$dt_D = dt_M + (11/25) * t_t$

$dt_D = dt_M + 0{,}44 * t_t$

O atraso para transferência de conteúdos até o *peer* consumidor a partir do *peer* provedor em MANet é de 10,40% ($dt_P = dt_M + 1{,}04 * t_t$) e a partir do *cache* do VSP 0,4% ($dt_C = dt_M + 0{,}04 * t_t$).

No Anexo 2, estão tabulados os resultados numéricos obtidos durante os experimentos em laboratório, bem como os cálculos matemáticos baseados nas equações obtidas, através da avaliação do fluxo de mensagens, para executar as operações de anúncio de conteúdos, procura por um conteúdo e descarga de um conteúdo por intermédio do protótipo implementado, que serviram de subsídio à obtenção dos resultados apresetnados acima.

Se o consumidor possuir disponibilidade de largura de banda, esse poderá fazer a descarga de mais de um fragmento do conteúdo do VSP em paralelo. Neste caso, o tempo de atraso da descarga é reduzido sigmificativamente, de modo proporcional ao número de fragmentos que estão sendo transferidos simultaneamente, porque o tempo de transferência dependerá apenas do roteamento das mensagens.

Todas as medidas mencionadas acima foram obtidas calculando-se a média dos resultados em cem repetições das experiências e o coeficiente de variabilidade observado durante as medidas, estava abaixo de 5%.

As principais vantagens de nossa proposta são qualitativas, ao invés de quantitativas. O serviço proposto garante que os conteúdos transferidos por uma aplicação móvel são íntegros e autênticos, ou seja, a aplicação P2P não desperdiçará tempo com a transferência de conteúdos corrompidos ou indesejados sobre MANet. Consequentemente, a expectativa é que o tráfego de conteúdo P2P indesejável, diminua significativamente.

No que diz respeito a provedores múltiplos, o serviço garante que não serão transferidos fragmentos corrompidos ao *peer* consumidor, independentemente do provedor, ou seja, no fim da operação de transferência, o *peer* consumidor pode concatenar todos os fragmentos íntegros na ordem correta e então, o conteúdo original será reconstruído. A DHT replica os índices P2P, através de suas funcionalidades nativas. Consequentemente, se um VSP se tornar indisponível, um de seus vizinhos assumirá o seu lugar. O VSP também mantém o estado da conexão com um *peer* consumidor, permitindo assim, a recuperação transparente em caso de desconexão momentânea com o *peer* consumidor, devido a instabilidades da MANet.

O VSP abstrairá a origem dos conteúdos, substituindo aqueles provedores que ficarem indisponíveis, de modo transparente, para o *peer* consumidor. O *peer* consumidor não precisa gerenciar a conexão com cada um dos provedores, dado que o VSP faz isto de uma maneira confiável.

Do ponto de vista do *peer* consumidor, o VSP assume transparentemente o lugar do *peer* provedor, dando ao consumidor a impressão de que não houve quaisquer mudanças, ou seja, a proposta foi desenvolvida, levando em conta a compatibilidade com a tecnologia P2P em uso.

O VSP acelera, consideravelmente, as transferências de conteúdos populares, através do uso de seu *cache* e de transferência em paralelo.

5.4. *Conclusão*

Neste capítulo, destinado à avaliação do protótipo, inicialmente foi descrito o cenário montado e utilizado para obtenção de dados necessários para a avaliação da solução proposta.

Estes resultados foram detalhadamente apresentados neste capítulo, juntamente com a avaliação do mesmo, de modo a prover subsídios para uso posterior.

Estas informações serão fundamentais para que, no capítulo a seguir, a proposta seja avaliada de modo a justificar ou não, a sua viabilidade.

Capítulo 6

Resultados, Conclusão e Trabalhos Futuros

Este trabalho apresentou uma proposta para minimizar o impacto da instabilidade e do *churn* no uso de redes P2P sobre MANet em um contexto onde as propriedades de segurança para conteúdos compartilhados, são importantes. O cenário proposto considerou a migração semi-automática dos conteúdos de redes P2P tradicionais para uma rede P2P orientada a serviços.

O VSP, *Virtual Service Provider*, como uma interface que intermedia as conexões entre os *peers* consumidores e provedores, mantém a compatibilidade com os *peers* consumidores atualmente em uso. O VSP permite:

a) buscas e aquisição de conteúdos,
b) anúncios de conteúdos,
c) migração de conteúdos de uma rede P2P tradicional para a DSP,
d) gerência do estado dos serviços,
e) provimento seguro de conteúdos, cuja origem seja a rede de serviço,
f) gerência transparente dos provedores ativos, e
g) transferência paralela sem a corrupção de integridade.

O protótipo desenvolvido mostrou-se viável e de fácil integração com redes P2P tradicionais sobre MANet, devido ao uso de pacotes e aplicações disponíveis no mercado, como JXTA, JXME, *Kademlia*, DHT *Bamboo*, sem a necessidade de alterá-las para atingir os resultados desejados, como vimos em [BISIG2003].

Como resultados da solução proposta, além dos resultados qualitativos relacionados

aos mecanismos propostos para garantia de integridade, autenticidade e alta disponibilidade do serviço apresentado nesta dissertação, podem ser citados o protótipo desenvolvido e a publicação de um artigo no evento ICACT 2009 [ICACT2009]. Quanto aos resultados quantitativos, no Anexo 2 foram apresentados os resultados obtidos através de experimentos realizados em laboratório, em um ambiente controlado, e que serviram de base para a avaliação realizada no capítulo 5.2. , onde as informações foram equacionadas para permitir a avaliação determinística do sistema.

Como trabalhos futuros, consideramos a implementação de algoritmos robustos de classificação e o uso de um mecanismo de reputação para acelerar o processo de certificação de autenticidade de conteúdos, minimizando a intervenção humana no processo.

Referências

[BAJA1999] Bajaj, L.; Takai, M.; Ahuja, R.; Tang, K.; Bagrodia, R.; Gerla, M.; GloMoSim: ***A Scalable Network Simulation Environment***; *Technical Report* 990027; *UCLA Computer Science Department*; 1999;

[BARC2006] Barcellos, Marinho P.; Gaspary, Luciano P.; Segurança em Redes P2P: Princípios, Tecnologias e Desafios, **SBRC 2006 - 24º Simpósio Brasileiro de Redes de Computadores**, MC5; Maio/2006;

[BISIG2003] Bisignano, M.; Calvagna, A.; Di Modica, G.; Tomarchio, O.; *"Expeerience: a JXTA middleware for mobile ad-hoc network"s*, Em Anais da **3ª Conferência Internacional de Computação em *Peer-to-Peer***, IEEE, 2003;

[CRAM2006] Cramer, Curt; Fuhrmann, Thomas; "*Performance Evaluation of Chord in Mobile Ad Hoc Networks"*; ***MobiShare'06 - ACM***; Los Angeles, Califórnia, USA; Setembro/ 2006;

[CLDC2008] *Sun Java Wireless Toolkit for CLDC Home***, http://java.sun.com/products/sjwtoolkit/;** último acesso Novembro/2008**;**

[GONG2002] Gong, li; "*Project JXTA: A Technology Overview"*; Sun Microsystems, Inc; San Antonio Road, Palo Alto, CA; outubro/2002;

[ICACT2009] Vithoft, Marcelo H.; Santin, Altair O.; Lima Jr, Luiz A. de Paula; "*A Management Service for P2P Content Over MANets"*; http://www.icact.org; em ***The 11th International Conference On Advanced Communication Tecnology* (ICACT)**; IEEE; aceitação Novembro/2008;

[JACQ2001] Jacquet, P.; Mühlethaler, P.; Clausen, T.; Laouiti, A.; Qayyum, A.; Viennot, L.; "*Optimized Link State Routing Protocol for Ad Hoc Networ*ks"; In Proc. ***IEEE INMIC***; pages 62–68; Lahore, Pakistan; Dezembro/2001;

[JOHN1996] Johnson, D. B.; Maltz, D. A.; *"Dynamic Source Routing in Ad Hoc Wireless Networks"*; ***Mobile Computing***, 353:153–181; Fevereiro/1996;

[KEYTOOL2002] Sun Microsystems, Inc; *"**keytool - Key and Certificate Management Tool**"*; http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html; último acesso Maio/2009**;**

[KUMAR2006] Kumar, R.; Yao, D.; Bagchi, A.; Ross, K.; Rubenstein, D.; ***"****Fluid Modeling of Pollution Proliferation in P2P Networks"***; Em Anais da *ACM Sigmetrics,*** 2006;

[MBIS2003] Bisignano, M.; Calvagna, A.; Di Modica. G.; Tomarchio, O.; *"Experience: a JXTA middleware for mobile ad-hoc networks"*, **Em Anais da *3º International Conference on Peer-to-Peer Computing***, IEEE, 2003.

[MB102005] Bisignano, Mario; Di Módica, Giuseppe; Tomarchio, Orazio; *"A JXTA compliant framework for mobile handheld devices in ad-hoc networks"*; ***10º IEEE Symposium on Computers and Communications*** (ISCC 2005); 2005;

[MB252005] Bisignano, Mario; Di Modica, Giuseppe; Tomarchio, Orazio; *"JMobiPeer: a middleware for mobile peer-to-peer computing in MANets"*; ***25 º IEEE International Conference on Distributed Computing Systems Workshops*** (ICDCSW'05)***;*** 2005;

[MERW2007] Van der Merwe, Johann; Dawoud, Dawoud; McDonald, Stephen; *"A Survey on Peer-to-Peer Key Management for Mobile Ad Hoc Network"s; **ACM Computing Surveys***, Vol. 39, No. 1, Artigo 1; Abril/2007;

[OLIV2005] Oliveira, Luciana Pereira; "**Os Dispositivos Móveis & As Redes *Peer-to-Peer"***; Centro de Informática – Universidade Federal de Pernambuco, Recife/PE, Agosto de 2005;

[OLIV2006] Oliveira, L. S.; Morita, M; Sabourin, R.; "*Feature Selection for Ensemble using the Multi-objective Optimization Approach"*. Estudos em ***Computational Intelligence*** (SCI); ISSN: 1860-949X; 6, 49-74; 2006;

[PERK1999] Perkins, C. E.; Royer, E. M.; "*Ad hoc On-Demand Distance Vector Routing"*; In Proc. ***2nd IEEE WMCSA***, pages 90–100;, New Orleans, LA, USA; Fevereiro/ 1999;

[PERK2001] Perkins, C. E.; Royer, E. M.; "***Ad Hoc Networking; Addison-Wesley"***, Bostom, MA, EUA, 2001;

[PLANET2008] PlanetLab: Home, **http://www.planet-lab.org/**; último acesso Novembro/2008**;**

[ROCH2004] Rocha, João; Domingues, Marco; Callado, Arthur; Souto, Eduardo; Silvestre, Guthemberg; Kamienski, Carlos; Sadok, Djamel; "*Peer-to-Peer*: Computação Colaborativa na Internet"; **SBRC2004**; capítulo 1; maio/2004;

[SANT2004] Santin, A. O.; "**Teias De Federações: Uma Abordagem Baseada em Cadeias de Confiança para Autenticação, Autorização e Navegação em Sistemas de Larga Escala** "; Tese de Doutorado em Engenharia Elétrica, UFSC; Florianópolis; 2004; pag; 153 e 154;

[SCHOLL2001] Schollmeier, R.; "*A definition of peer-to-peer networking towards a delimitation against classical client-server concepts"*. **Em Anais da *7º EUNICE Open European Summer School and the IFIP Workshop on IP and ATM traffic management***; Paris, França, 2001;

[STOICA2001] Stoica, I.; Morris, R.; Karger, D. R.; Kaashock, M. Frans; Balakrishman, H.; "*Chord: A scalable peer-to-peer lookup protocol for internet applications"*; **Em Anais da *ACM SIGCOMM***, páginas 149-160; San Diego, Califórnia, 2001;

[TRAV2002] Traversa, Bernard; Abdelaziz, Mohamed;Duigou, Mike; Hugly, Jean-Christophe; Pouyoul, Eric; Yeager, Bill; "***Project JXTA Virtual Network"***; Sun Microsystems, Inc, San Antonio Road, Palo Alto, CA; outubro/2002

[TRAV2003] Traversa, Bernard; Abdelaziz, Mohamed; Duigou, Mike; Hugly, Jean-Christophe; Pouyoul, Eric; Yeager, Bill; Arora, Ahkil; Haywood, Carl; "***Project JXTA 2.0 Super-Peer Virtual Network"***; Sun Microsystems, Inc, San Antonio Road, Palo Alto, CA; Março/2003;

[TRAVWALK] Traversa, Bernard; Abdelaziz, Mohamed; Pouyoul, Eric; "***Project JXTA: A Loosely-Consistent DHT Rendezvous Walker"***; Sun Microsystems, Inc, San Antonio Road, Palo Alto, CA;

Anexo 1

Formato de Mensagens

1.1. Formato de Mensagens da aplicação *App Mobile*

Para que o leitor deste trabalho passa ter uma melhor idéia do protótipo implementado, os formatos das principais mensagens serão apresentados nesta seção, mensagens estas que foram separadas em categorias, numeradas de I a V, como pode ser observado a seguir.

I. Anúncio de Conteúdos:

O formato da menagem JXME enviado pelo *peer* para anunciar seus conteúdos é apresentado na Mensagem 1, onde os campos são:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para anúncio de conteúdos é: **ADV;**
- **PEER_NAME**: este campo por sua vez, é usado para indentificar o *peer* que está anunciando o conteúdo;
- **IP_ADDRESS**: para compor a identificação do *peer* que está anunciando um conteúdo, este protótipo faz uso do endereço IP do *peer*, informação esta que é colocada neste campo da mensagem;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que demais *peers* acessem o *peer* que está anunciando o conteúdo;
- **CONTENT**: finalmente, neste campo, o nome do conteúdo que está sendo anunciado, é informado. Todas estas informações serão utilizadas para compor o

registro que será armazenado na DSP, para que qualquer *peer* possa encontrar e recuperar a origem do anúncio e, conseqüentemente, o próprio conteúdo;

{ MSG_TYPE | PEER_NAME | IP_ADDRESS | PIPE_ID | CONTENT }

Mensagem 1: Formato da mensagem de Anúncio de Conteúdo

II. Procura por um Conteúdo:

O formato da menagem JXME enviado pelo *peer* para procurar por um conteúdo é apresentado na Mensagem 2, onde os campos são:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para procura por um conteúdo é: **SEARCH;**
- **PEER_NAME**: este campo, por sua vez, é usado para indentificar o *peer* que está procurando por um conteúdo;
- **PEER_ID**: este campo é usado para informar o ID do *peer* que está procurando por um conteúdo;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que demais *peers* acessem o *peer* que está procurando por um conteúdo;
- **SEARCH_ID**: para cada operação de procura por um conteúdo, o peer solicitante cria um identificador único, com a finalidade de identificar, exclusivamente, a operação. Esta informação também será utilizada na mensagem de resposta para esta finalidade;
- **RESOURCE**: neste campo, o nome do conteúdo que está sendo procurado, é informado;

{ MSG_TYPE |PEER_NAME|PEER_ID|PIPE_ID|SEARCH_ID| RESOURCE }

Mensagem 2: Formato da mensagem de procura por um conteúdo

Após a mensagem de procura por um conteúdo ter sido interpretada e convertida no módulo VSP Comunication Manager, ela é encaminhada para o módulo funcional, cuja responsabilidade é tratar esta requisição e neste caso, é o módulo *Content Manager* que através de uma consulta a DHT, usando a função getDHT(args), que faz uso de um script python (get.py). De posse do resultado da pesquisa por um conteúdo na DSP, o

VSP envia a resposta para o peer solicitante, através de uma mensagem JXME, cujo formato é apresentado pela Mensagem 3, cujos compos serão descritos a seguir:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para resposta, a procura por um conteúdo é: **SEARCH_RSP;**
- **PEER_NAME**: este campo, por sua vez, é usado para indentificar o *peer* que estava procurando por um conteúdo;
- **PEER_ID**: este campo é usado para informar o ID do *peer* que estava procurando por um conteúdo;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que demais *peers* acessem o *peer* que está procurando por um conteúdo;
- **SEARCH_ID**: campo utilizado para identificar a consulta por um conteúdo;
- **RESOURCE_NAME**: neste campo, o nome do conteúdo que estava sendo procurado, é informado;
- **SEARCH_RESULT**: em caso positivo, neste campo está contido o PIPE_ID do *peer* que anunciou o conteúdo. Aqui, vale lembrar, que para a solução proposta, o *peer* solicitante sempre receberá, neste campo, o PIPE_ID do VSP ao qual está conectado e caberá ao VSP prover este conteúdo de forma transparente para o *peer* consumidor.

{ MSG_TYPE |PEER_NAME|PEER_ID|PIPE_ID| [SEARCH_ID| RESOURCE_NAME| SERCH_RESULT }

Mensagem 3: Formato da mensagem de resposta à procura por um conteúdo

III. Busca por um Conteúdo:

Após o peer consumidor ter encontrado o conteúdo desejado, ele enviará uma mensagem JXME para o VSP, cujo formato é apresentado pela Mensagem 4 e onde os campos são:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para busca por um conteúdo é: **REQ;**
- **PEER_NAME**: este campo, por sua vez, é usado para indentificar o *peer* que está buscando um conteúdo;
- **PEER_ID**: este campo é usado para informar o ID do *peer* que está buscando um conteúdo;

- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que demais *peers* acessem o *peer* que está buscando o conteúdo;
- **SEARCH_ID**: Este campo contém o identificador da operação de procura por um conteúdo, onde o conteúdo que agora está sendo requisitado foi encontrado na DSP. Esta informação permitirá que os dados da procura sejam recuperados no VSP facilitando a operação de localização do conteúdo requisitado;
- **REQUEST_ID**: Assim como o campo SEARCH_ID, este campo é preenchido com um identificador exclusivo para esta operação que será utilizado com a finalidade de identificar a requisição de um conteúdo;
- **FRAME_ID**: este campo é usado para requisitar a transferência de um fragmento específico, em caso da necessidade de retransmissão de um fragmento ou algo do gênero. Em caso de requisição de um conteúdo completo, este campo deverá ser preenchido com o valor 0 (zero);
- **REQUEST_NAME**: neste campo, o nome do conteúdo que está sendo requisitado é informado;

{ MSG_TYPE |PEER_NAME|PIPE_ID|SEARCH_ID| REQUEST_ID|FRAME_ID|REQUEST_NAME}

Mensagem 4: Formato da mensagem de busca por um conteúdo

Como resposta à requisição de um conteúdo (indemependente desta resposta ser positiva ou negativa), o VSP enviará, em um primeiro momento, a Mensagem 5, cujos campos adicionais à Mensagem 4, serão apresentados abaixo:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para resposta da busca por um conteúdo é: **REQ_RSP;**
- **PEER_NAME, PEER_ID, PIPE_ID, SEARCH_ID, REQUEST_ID e REQUEST_NAME**: Estes campos carrregam as mesmas informações que foram descritas na **Mensagem 4**;
- **NUMBER_FRAMES:** Este campo informará ao *peer* consumidor o número de fragmentos que serão enviados para compor o conteúdo;
- **PROVIDER_ID**: Neste campo é armazenado o PIPE_ID do *peer* provedor do conteúdo, para que em caso de perda de comunicação com o provedor, o consumidor

possa acessá-lo oportunamente (vale lembrar que neste serviço proposto, o *peer* provedor será o VSP, ao qual o *peer* consumidor está conectado). Em caso de resultado negativo, este campo não trará qualquer informação;

{ MSG_TYPE |PEER_NAME|PIPE_ID|SEARCH_ID| REQUEST_ID |REQUEST_NAME|NUMBER_FRAMES|PROVIDER_ID}

Mensagem 5: Formato da mensagem de resposta à busca por um conteúdo

Em seguida, o peer consumidor começará a receber os fragmentos do conteúdo requisitado e o formato da mensagem é apresentado em Mensagem 6, cujos campos são:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para transferência de conteúdo é: **DOWN;**
- **PEER_NAME, PEER_ID, PIPE_ID, SEARCH_ID, REQUEST_ID e REQUEST_NAME**: Estes campos carrregam as mesmas informações que foram descritas na Mensagem 4;
- **NUMBER_FRAMES e PROVIDER_ID**: Estes campos carrregam as mesmas informações que foram descritas na Mensagem 5;
- **FRAME_ID**: Este campo contém o identificador do fragmento do conteúdo;
- **HASH**: Este campo contém o valor do *hash* calculado para este fragmento de conteúdo que deve ser usado para checagem de integridade;
- **LENGTH**: Este campo contém o tamanho em *bytes* do fragmento enviado;
- **PAYLOAD**: Este campo contém o fragmento de conteúdo propriamente dito;

{ MSG_TYPE |PEER_NAME|PIPE_ID|SEARCH_ID| REQUEST_ID |REQUEST_NAME|NUMBER_FRAMES|PROVIDER_ID|FRAME_ID**|HASH|** LENGTH|**PAYLOAD}**

Mensagem 6: Formato da mensagem de resposta à busca por um conteúdo

IV. Troca de mesnagens entre *peers*:

A troca de mensagens entre peers se dá através do envio de uma mensagem JXME de um peer para outro e cujo formato da mensagem está ilustrado em Mensagem 7 e onde a descrição dos campos é:

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para a troca de mesnsagens entre *peers* é: **MSG;**
- **PEER_NAME**: este campo por sua vez, é usado para indentificar o *peer* que está enviando a mensagem;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que demais *peers* acessem o *peer* que está enviando uma mensagem;
- **MSG**: Este campo contém a mensagem de texto que está sendo enviada pelo *peer*;

{ MSG_TYPE |PEER_NAME|PIPE_ID|MSG }

Mensagem 7: Troca de mensagens entre peers

V. Lista de VSPs:

Sempre que um novo VSP for introduzido na rede ou então seja excluído dela, uma mensagem JXME é enviada a todos os peers, atualizando a lista de VSPs da rede. No momento da inicialização dos *peers*, este pode requisitar ao VSP que está conectado à lista de VSPs disponíveis na rede. As mensagens que serão descritas, a seguir, são destinadas a este fim.

A requisição da lista de VSPs se dá através da Mensagem 8, que será imediatamente descrita.

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo *peer* e cujo valor para a requisição da lista de VSPs é: **VSP_REQ;**
- **PEER_NAME**: este campo é usado para indentificar o *peer* que está enviando a mensagem;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que o VSP possa enviar a mensagem e resposta à requisição;

{ MSG_TYPE |PEER_NAME|PIPE_ID }

Mensagem 8: Requisição da lista de VSPs disponíveis na rede

Tanto a mensagem de resposta a requisição da lista de VSPs disponíveis, quanto a notificação esponstânea da mudança desta lista possuem o formato apresentado na Mensagem 9.

- **MSG_TYPE**: este campo é utilizado para indicar o tipo da mensagem que está sendo enviada pelo/para o *peer* e cujos valores para a resposta a uma requisicão da lista de VSPs e para o anúncio espontâneo da mudança desta lista são respectivamente: **VSP_RSP e VSP_ANNOUNCE;**
- **PEER_NAME**: este campo é usado para indentificar o *peer* que está enviando a mensagem;
- **PIPE_ID**: este campo carrega a informação da identificação do canal criado para que o *peer* possa ser acessado pelos demais *peers*;
- **NUMBER**: este campo informa o número de VSPs que compõem a lista que está sendo enviada nesta mensagem;
- **VSP**: este campo pode conter várias instâncias, pois nele estão armazenadas as informações nome e PIPE_ID do VSP (separados por ";");

{ MSG_TYPE |PEER_NAME|PIPE_ID }

Mensagem 9: Lista de VSPs disponíveis na rede