COPPE/UFRJ

CNOSSOS - MIDDLEWARE DISTRIBUÍDO BASEADO NA PERMUTA MULTILATERAL COMO MODELO DE COMPARTILHAMENTO DE RECURSOS EM REDES P2P

Luiz Gustavo Lourenço Moura

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
Carlo Emmanuel Tolla de Oliveira

Rio de Janeiro
Setembro de 2010

CNOSSOS - MIDDLEWARE DISTRIBUÍDO BASEADO NA PERMUTA MULTILATERAL COMO MODELO DE COMPARTILHAMENTO DE RECURSOS EM REDES P2P

Luiz Gustavo Lourenço Moura

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Examinada por:

_____

Prof. Felipe Maia Galvão França, Ph. D.

_____

Prof. Carlo Emmanuel Tolla de Oliveira, Ph.D.

_____

Prof. Geraldo Bonorino Xexéo D.Sc.

_____

Prof. Alexandre Sztajnberg D.Sc.

_____

Prof. Toacy Cavalcante de Oliveira D.Sc.

_____

Prof. Fabio Protti D.Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2010

Moura, Luiz Gustavo Lourenço

Cnossos - Middleware distribuído baseado na permuta multilateral como modelo de compartilhamento de recursos em redes P2P/ Luiz Gustavo Lourenço Moura - Rio de Janeiro: UFRJ/COPPE, 2010.

XVII, 178 p.: il.; 29,7 cm.

Orientadores: Felipe Maia Galvão França,

Carlo Emmanuel Tolla de Oliveira

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2010.

Referencias Bibliográficas: p. 170-176.

1. Compartilhamento de recursos. 2. Reputação. 3. *Peer-to-Peer.* 4. Permuta Multilateral . I. França, Felipe Maia Galváo et al . II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

"...Bom mesmo é ir a luta com determinação, abraçar a vida com paixão, perder com classe e vencer com ousadia... Pois o triunfo pertence a quem se atreve..." (Charles Chaplin)

AGRADECIMENTOS

Muitas são as pessoas a quem devo esse momento de conclusão desse trabalho. Acredito que Agradecer é uma virtude. E mesmo correndo o risco de esquecer alguém, não posso deixar de citar alguns nomes e expressar minha eterna gratidão.

A Deus, o autor e consumador da minha fé! A Ele realmente seja a glória! Só nós sabemos como foi difícil... Muito obrigado não só pela força para chegar até o fim, pela sabedoria para as escolhas certas, pelo auxílio das pessoas que você colocou no meu caminho para me ajudar, mas principalmente por estar ao meu lado. Isso foi o que mais me motivou a ir até o fim. Obrigado.

A minha mãe e ao meu pai, que foram os grandes auxiliadores desse projeto. Vocês me dão todo suporte emocional e físico necessário para que eu possa atingir meus objetivos. Você é meu exemplo de garra e de que devemos lutar pelo que queremos. Eu te amo e te dedico essa vitória!

Agradeço a minha namorada Danielle, que prestou grande favor em me escutar nos devaneios de pesquisa, nas horas de certa dúvida no êxito, nos finais de semana sem passeio, nas férias que não podemos viajar...enfim Muito Obrigado por ficar do meu lado também nestas horas.

Ao meu orientador Prof. Carlo Emmanoel, que me auxiliou no desenvolvimento desse trabalho. Muito obrigado pela sua amizade, por acreditar em mim, por me ajudar e principalmente por ter paciência nos momento em que precisava de ajuda. Termino com a sensação de que ganhei mais do que um mentor... Ganhei um grande e bom amigo.

Ao meu orientador Prof. Felipe França, pela sua participação fundamental para conclusão desse trabalho. Acredito que sem seu apoio ele não teria se concretizado. Obrigado por suas ideias, pelo apoio e pela paciência comigo. Mais do que um auxilio de professor ganhai o apoio de um amigo.

A Universidade Candido Mendes e ao Instituo Federal Fluminense, por me auxiliar durante esse período. Tenho um orgulho enorme de trabalhar nessas duas empresas.

Enfim, agradeço a todos que de alguma forma contribuíram para que este trabalho fosse finalizado. Espero um dia poder retribuir de alguma maneira toda força e apoio que me ofereceram.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CNOSSOS - MIDDLEWARE DISTRIBUÍDO BASEADO NA PERMUTA MULTILATERAL COMO MODELO DE COMPARTILHAMENTO DE RECURSOS EM REDES P2P

Luiz Gustavo Lourenço Moura

Setembro/ 2010

Orientadores: Felipe Maia Galvão França
Carlo Emmanuel Tolla de Oliveira

Programa: Engenharia de Sistemas e Computação

O compartilhamento de recursos em sistemas distribuídos é uma área com uma constante necessidade de pesquisas, tanto pelas promissoras evoluções previstas nos sistema de Grid e P2P, quanto pelas dificuldades de conciliar o funcionamento de todos os tipos e configurações de *peer* que podem existir nesse tipo de rede. Uma diferença fundamental das aplicações do P2P, comparadas aos sistemas distribuídos tradicionais, é o fato deque as decisões dos *peers* são baseadas em seu próprio interesse e este princípio pode conduzir a uma operação ineficiente de sistema. Assim, incentivos apropriados devem ser dados aos *peers* para que os mesmos possam compartilhar seus recursos com a máxima eficiência. Nesta tese foram identificados e categorizados os mais importantes conceitos relacionados com essa prática, e escolhemos a discussão sobre a questão do fornecimento de conteúdos de compartilhamento em sistemas P2P baseados no modelo econômico de permuta multilateral como o principal foco de estudo. Na permuta, uma das partes promete uma coisa em troca de outra sem a utilização de nenhum tipo de moeda.

Apresentamos a base dos requisitos de sistema para apoiar a funcionalidade necessária para esta abordagem e, assim, formular um modelo econômico adequado. Neste modelo, os benefícios da permuta estão diretamente relacionados com o tempo que um peer é forçado a permanecer on-line e, consequentemente, contribuir para outros. Por último, utilizamos mecanismo de reputação como regra para determinar a confiança dos peers como também à confiança dos seus recursos e serviços.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

CNOSSOS - DISTRIBUTED MIDDLEWARE BASED ON MULTILATERAL EXCHANGE AS A MODEL OF RESOURCE SHARING IN P2P NETWORKS

Luiz Gustavo Lourenço Moura

September/09

Advisors: Felipe Maia Galvão França

Carlo Emmanuel Tolla de Oliveira

Department : Computing and Systems Engineering

The resource sharing in the distributed systems is an area with constant need for researches, not only because of the promising previewed evolutions in the Grid and P2P, but also for the difficulty of conciliating the functioning of all types of peer configures that can exists in this type of net. Although new applications are being proposed and projected to explore the different types of resources that are available, such as the computational power, the storage and the band extension, among others. A remarkable difference of the applications of P2P, compared to the traditional distributed systems, is the fact that the peer decisions are based in its own interest and this principle can lead to an inefficient operational system. The most important concepts related to this practice have been identified and categorized in this thesis. We have chosen the discussion about the concept of sharing in P2P systems based on a model of multilateral economical exchange as the main focus of this study. In the exchange, one part promises something in exchange for another without the use of any monetary system.

We have presented the basis of the system requirements to support the functionality needed for this approach and to formulate an adequate economical model; in this model, the exchange of benefits are directly related to the timing that one peer is forced to remain on line and consequently to contribute to others. At last, reputation mechanisms as a rule to determine the trust on peers have been used as well as the trust on its resources and services.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API	*Application Programming Interface*
CAN	*Content Addressable Network*
CPU	*Central Processing Unit*
DHT	Distributed *Hash* Table
DTO	*Data Transfer Objects*
HTTP	*HiperText Transfer Protocol*
IP	*Internet Protocol*
JEE	*Java Enterprise Edition*
P2P	*Peer-to-Peer*
PDA	*Personal Digital Assintant*
QoS	*Quality of Service*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TTL	*Time-to-Live*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifier*
URN	*Uniform Resource Name*
WCG	*World Community Grid*
WWW	*World Wide Web*

1 INTRODUÇÃO

Neste capítulo, é apresentada, sucintamente, a pesquisa documentada nesta tese, abordando as motivações e justificativas, o problema e a hipótese de pesquisa, os objetivos, contribuições, metodologia utilizada e a organização do texto.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

A revolução ocorrida na última década em relação à tecnologia de informação surge como um dos principais vetores de desenvolvimento socioeconômico em nível mundial. A convergência tecnológica dos setores das Telecomunicações tradicionais e da Informática potenciaram o aparecimento, implantação e crescimento explosivo de novas redes, serviços de telecomunicações e aplicações informáticas associadas aos novos serviços. De um ponto de vista puramente técnico, a evolução tecnológica no setor das Telecomunicações está, atualmente, totalmente dependente das facilidades oferecidas pelas tecnologias digitais.

Com a massificação dos microcomputadores, os sistemas computacionais vêm se tornando cada vez mais presentes no dia-a-dia do ser humano (TANENBAUM, 2002). O aumento da computação móvel, das capacidades de processamento e armazenamento de informações, aliado à miniaturização de computadores e outros dispositivos, fazem com que a computação saia de um ambiente baseado em *desktops* para um ambiente caracterizado pela presença de dispositivos e sistemas computacionais interagindo com usuários, em qualquer lugar, em qualquer instante.

Aliada a esse fator, a necessidade de computadores com alto desempenho não é recente. Desde o princípio da computação há a necessidade de realizar tarefas que exigem uma quantidade considerável de recursos, geralmente de processamento. Essa necessidade levou os pesquisadores e as empresas ligadas à área de computação a desenvolverem sistemas que possibilitassem satisfazer, inicialmente, a necessidade de processamento. Juntamente a essa evolução, deu-se a expansão de redes de comunicação em larga escala, garantindo que os computadores fossem interligados em velocidade cada vez maior, permitindo o desenvolvimento de sistemas que possam realizar a divisão de tarefas entre *peers* dispostos em locais diferentes.

Atualmente, a maior parte das aplicações na Internet é baseada no modelo cliente-servidor (TANENBAUM, 2002). Neste modelo, os recursos, informações e serviços, são centralizados em servidores, que os disponibilizam em resposta às requisições de vários clientes. Os clientes assumem um papel passivo, consumindo

recursos, mas não os disponibilizando. O aumento do número de clientes torna necessário um investimento cada vez maior em servidores mais robustos, enquanto muitas vezes o potencial computacional dos clientes não é aproveitado.

As redes P2P (LEUF, 2002) apresentam uma maneira de disponibilizar recursos diferentes do modelo cliente-servidor (PARAMESWARAN, SUSARLA e WHINSTON, 2001) (PETERSON e DAVIE, 2003). Em uma rede P2P, todos os *peers* podem disponibilizar e consumir recursos, apresentando, simultaneamente, características de cliente e servidor. Para realizar o compartilhamento, os *peers* se conectam diretamente uns aos outros. Os vizinhos de um *peer* são entidades com características similares às do próprio *peer*.

Diversas propostas para o problema geral da composição de redes P2P são baseadas na criação de uma economia fundada em soluções bastante conhecidas da economia de mercado. Nesse modelo econômico, usuários que necessitam de recursos negociam seu uso com os provedores desses, e pagam pelo serviço obtido. Motivar a provisão de recursos e regular o acesso aos mesmos é uma área que está em constante transformação e por isso, foi escolhida como área de investigação da presente proposta.

1.2 PROBLEMA

O grande aumento na capacidade de processamento dos dispositivos computacionais trouxe, para a maioria dos usuários, um poder computacional maior do que o necessário para a execução de suas aplicações. Enquanto máquinas servidoras das empresas trabalham em seu limite máximo de processamento, o restante de seu parque de computadores é utilizado como estações de trabalho de funcionários ou como laboratórios para uso de alunos e pesquisadores.

Em (MUTKA e LIVNY, 1987) (MUTKA e LIVNY, 1991), os autores analisaram a quantidade efetiva de recurso utilizada nas estações e concluíram que a maior parte do tempo dos recursos analisados ficava ociosa. Até mesmo quando as máquinas encontravam-se em uso, na maioria das situações, não havia a utilização plena dos todos os recursos. Encontrar a melhor maneira para se tirar proveito dessa ociosidade, com oportunidade para implementar uma forma de compartilhar os recursos disponíveis, permitindo a disponibilização, a baixo custo, de altos níveis de serviço, torna-se uma tarefa motivadora.

Neste sentido, duas importantes linhas de pesquisa se desenvolveram: grades computacionais (Grid Computer) e redes Peer-to-Peer (P2P) (FOSTER e IAMNITCHI,

2003). A motivação inicial para o desenvolvimento da tecnologia de computação em grade foram os requisitos de comunidades profissionais, para acesso a recursos remotos para processamento em larga escala e bases de dados distribuídas. Assim, foram tratadas, principalmente, questões de localização e distribuição otimizada de recursos, bem como paralelismo de tarefas. Já o desenvolvimento de redes P2P foi impulsionado, inicialmente, pela popularização de serviços de finalidades específicas, como por exemplo, compartilhamento de arquivos de música.

No entanto, segundo (FOSTER e IAMNITCHI, 2003), ambas as tecnologias propõem-se a resolver o mesmo problema: a organização dos recursos compartilhados em comunidades virtuais; e utilizam as mesma abordagem para resolvê-los, porém fazendo uso de técnicas diferentes.

Dessa forma, no contexto dessa tese, pretende-se investigar a seguinte questão: “Como auxiliar usuários a compartilhar seus serviços e a reconhecer indivíduos confiáveis para efetuar esse compartilhamento, dentro de uma rede P2P?”.

1.3 HIPÓTESE

Motivar a provisão e o acesso a recursos de forma organizada também pode ser visto com um problema econômico. Nessa visão, usuários que necessitam de recursos negociam seu uso com os provedores desses, e pagam pelo serviço obtido. Essas soluções visam à criação de rede onde qualquer tipo de aplicação pode ser executado.

É possível, a partir de formas de interação bem definidas nos modelos econômicos, automatizar o processo de negociação e obtenção do acesso aos serviços em uma rede P2P. Além disso, dependendo do modelo econômico utilizado, é possível cobrar dos usuários algum retorno para o provedor do serviço.

Em modelos econômicos, existem duas formas de alocar recursos entre agentes competidores: a economia baseada em preços e a economia baseada em trocas (FERGUSON, NIKOLAOU, *et al.*, 1996). Na abordagem baseada em preços, uma moeda é utilizada para expressar os custos dos recursos. Cada agente possui um padrão financeiro e o utiliza para comprar acesso aos recursos de que necessita de outro agente. Os preços dos recursos são obtidos da negociação entre consumidores e provedores, regulando oferta e demanda. Na segunda abordagem, cada agente detém um conjunto de recursos e faz trocas destes recursos com os de outro agente que lhe interessam.

A maioria dos sistemas construídos com base nesse modelo econômico utiliza a abordagem baseada em preços. A utilização de uma moeda dá flexibilidade ao sistema e possibilita o uso de mecanismos bastante elaborados para a obtenção do equilíbrio no mercado. Entretanto, a utilização de uma moeda tornam necessários mecanismos de controle como cobrança, entre outros. Além disso, prover as garantias necessárias para o funcionamento correto destes mecanismos em um ambiente computacional de participantes distribuídos não confiáveis é uma tarefa muito complexa.

Troca, permuta ou mesmo escambo são sinônimos que não remetem apenas à forma de comércio de mercadorias praticado na antiguidade. Há muitos anos, quando não existia moeda, as negociações por meio de trocas eram frequentes. As pessoas davam o que tinham em excesso e recebiam o que lhes faltava. Esse modelo nunca saiu de moda na economia. Nos dias de hoje essa prática não só ganha novas formas, como também novos nomes: as empresas chamam de permuta. E não se troca apenas por necessidade, mas, principalmente, por desejo. Muitas empresas no mundo estão se organizando e fazendo parte de redes de permuta, criando ambientes de negócios gerenciados, principalmente, por meio da Internet. Vale trocar tudo, desde produtos até serviços, e, geralmente, não é utilizado nenhum tipo de dinheiro real nas negociações.

Também com o intuito de motivar o compartilhamento de serviços/recursos, os conceitos de confiança e reputação são aplicáveis em ambientes de interação virtual por meio dos Sistemas de Reputação (JOSANG, ISMAIL e BOYD, 2006) (RESNICK, ZECKHAUSER, *et al.*, 2000). Esses sistemas coletam, distribuem e agregam informações sobre o comportamento dos participantes nas interações realizadas. Dessa forma, auxiliam os usuários a decidirem em quem confiar, motivam o bom comportamento dos participantes, e procuram controlar a participação daqueles que são considerados desonestos.

Segundo (RESNICK, ZECKHAUSER, *et al.*, 2000), os Sistemas de Reputação representam uma alternativa de auxiliar os próprios usuários a criar relacionamentos confiáveis na Internet, permitindo que eles realizem avaliações sobre a atuação dos indivíduos, e identifiquem as reputações avaliadas perante a opinião de uma comunidade. Nesse sentido, os mecanismos utilizados pelos sistemas de reputação se apresentam como uma alternativa interessante para promover e gerenciar relações de confiança entre os participantes, em redes P2P, com o objetivo de compartilhamento de recursos.

Dessa forma, com base na questão em estudo dessa tese, foi formulada a seguinte hipótese: “Se for aplicado o modelo econômico baseado em trocassem conjunto com mecanismos de reputação em redes P2P, é possível auxiliar os usuários no compartilhamento de seus recursos e na identificação de indivíduos confiáveis para compartilhar tais recursos.”

1.4 OBJETIVOS

Especificar um modelo econômico para tarefas realizadas em um sistema de compartilhamento de recursos é, em geral, uma tarefa muito complexa. A principal razão é que os participantes deverão contribuir com tipos diferentes de recursos (largura de banda, armazenamento, ciclos CPU, conteúdo), com características diferentes.

Essa tese tem como objetivo a construção de um *midlleware* para gerenciamento do compartilhamento de recursos em redes P2P que forneça estrutura para que haja troca desses serviços e que essa troca seja estabelecida tendo como apoio o mecanismo de estabelecimento de relações de confiança, baseado na reputação de cada *peer*. Identificar as dificuldades existentes para se criar um ambiente de rede P2P mostrando os pontos críticos no projeto da infraestrutura e quais fatores viabilizam ou inviabilizam a sua utilização no processo comunicação e troca de serviços computacionais entre os *peers* é um dos objetivos específicos desse trabalho.

Quando falamos em eficiência do modelo econômico, temos questões importantes que deverão ser tomadas para discussão. Além de ser complexo definir um modelo para sistemas P2P, outra questão adicional é a falta de informações sobre as características e preferências dos *peers* presentes no sistema, fatores que influenciam tanto na determinação da quantidade de serviços que serão disponibilizados quanto na definição das características da troca.

Outro importante objetivo especifico é apresentar uma proposta de utilização de um modelo econômico simplificado para o compartilhamento de recursos. Esse modelo econômico é baseado na troca de recursos ociosos. O modelo também visa abrandar a questão de usuários que consomem, mas não compartilham seus recursos: o objetivo é que esse usuário seja estimulado a compartilhá-los. Esta tarefa deverá ser feita mantendo a descentralização completa de todos os serviços e a escalabilidade com relação ao número de *peers* participantes. Para tanto, é necessário

associar diversos conceitos como organização dos *peers* na rede de forma adequada à execução de buscas e o uso de algoritmos de replicação, entre outros.

1.5 CONTRIBUIÇÕES

Como contribuição principal este trabalho agrega a possibilidade de compartilhamento de recursos entre *peers* de uma rede P2P,por meio da permuta de recursos.

O modelo visa apresentar uma opção consistente na rede, desatrelando os recursos dos seus respectivos provedores, mantendo a consistência e a atualidade da informação e criar um modelo que possa ser utilizado para trabalhar com recursos e com localização dos mesmos, promovendo um mercado de permuta de tais recursos em um ambiente P2P.

Finalmente, apresenta a proposta de um mecanismo que adicione funcionalidades na localização de informações referentes a serviços, recursos ou dados. Para tal, o mecanismo propõe seleção de informações levando em consideração critérios referentes ao ambiente onde o recurso se encontra (por exemplo: quantidade de memória disponível pelo peer onde o serviço se encontra, capacidade de armazenamento, preço da informação). Com essas informações é possível balancear a seleção de dados referentes a serviços e recursos, por exemplo. Assim, uma vez que uma busca é feita por um serviço de impressão, por exemplo, é possível a permuta desse com outro servidor ou recurso que o *peer* possa disponibilizar na permuta.

1.6 METODOLOGIA

Esse trabalho foi realizado em etapas seguindo os procedimentos metodológicos que orientam a realização de uma pesquisa científica:

A primeira etapa compreendeu a realização de uma revisão bibliográfica em artigos científicos, teses e dissertações sobre Modelos Econômicos, *Midllewares*, redes P2P e Sistemas de Reputação, seguida de um levantamento em sites e serviços na *Web* que discorrem sobre esses temas. Esse procedimento teve como objetivo aprofundar os conhecimentos teóricos referentes ao assunto, no sentido de obter a fundamentação teórica necessária para a pesquisa, que abrange a definição de um problema e a formulação de uma hipótese de solução.

A segunda etapa compreendeu o desenvolvimento de uma proposta de solução para o problema de pesquisa, visando atender à hipótese formulada. Essa etapa será seguida do desenvolvimento de um protótipo, o qual abrangeu as seguintes etapas de desenvolvimento de um sistema: levantamento dos requisitos, especificação, modelagem, descrição da ferramenta e implementação.

A etapa final foi a realização de um estudo de caso a fim de avaliar e verificar a viabilidade do modelo proposto em atender à hipótese que orienta essa dissertação

1.7 ORGANIZAÇÃO DO TRABALHO

O restante deste trabalho está organizado como segue. Nos capítulos 2, 3, 4, 5 apresentados a revisão teórica contendo os princípios de Sistemas Distribuídos, reputação e economia, respectivamente, aplicados ao longo deste trabalho. No capítulo 6 apresentam-se detalhes da arquitetura proposta. No capítulo 7 apresenta-se particularidades dos mecanismos para o processo de permuta multilateral. No capítulo 8 exibe a infraestrutura computacional necessária para o compartilhamento de recursos em ambiente P2P. No capítulo 9 apresentam-se as simulações e as análises dos resultados obtidos. No capítulo 10 apresenta-se a conclusão.

2 SISTEMAS DISTRIBUÍDOS

2.1 INTRODUÇÃO

O surgimento das redes de computadores, nos anos setenta, e a rápida disseminação dos computadores pessoais, no início dos anos oitenta, possibilitaram um novo paradigma - o de sistemas distribuídos–no qual os recursos compartilhados passaram a ser também distribuídos entre um número de computadores ou dispositivos periféricos geograficamente distribuídos.

Com a distribuição do poder computacional entre componentes remotos, foi possível que programas em máquinas locais (clientes) tivessem acesso a serviços remotos (servidores). Esse benefício trouxe novos desafios, tanto no aspecto de desenvolver e gerenciar *softwares* em ambiente distribuído, quanto nas dificuldades inerentes ao novo ambiente: um conjunto de sistemas autônomos sem acesso a estados globais.

Dificuldades como a impossibilidade de resolver certos problemas básicos, concordar em ações comuns na presença de falhas (FISCHER, LYNCH e PATERSON, 1985),ou mesmo a falta de ferramentas (de desenvolvimento e gerenciamento distribuído) passaram a ser uma barreira para o novo paradigma implantação de sistemas distribuído também denominada Computação Distribuída.

Um sistema distribuído pode ser definido como aquele "formado por componentes de hardware e software, situados em redes de computadores, e que se comunicam e coordenam suas ações apenas por meio de trocas de mensagens" (COULOURIS, DOLLIMORE e KINDBERG, 2006). A separação espacial dos computadores que formam esses sistemas distribuídos pode variar desde poucos metros, em uma rede local, até quilômetros de distância, em uma rede largamente dispersa. Da mesma maneira, podem variar bastante os desafios enfrentados pelos projetistas desses sistemas - frente aos problemas de heterogeneidade, segurança e confiabilidade (tolerância a falhas).

A partir do advento da Internet, a computação distribuída passou a ter relevância definitiva, a ponto de aquela tornar-se a plataforma almejada pela maioria dos desenvolvedores de software. Em outras palavras, os projetos de *Software* passaram a incluir requisitos tais como: acesso via *World Wide Web(*WWW) a sistemas corporativos legados, desenvolvimento de aplicações tipo cliente/servidor sobre o Transmission Control Protocol / Internet Protocol (TCP/IP) que é o protocolo padrão da Internet.

Em comparação ao modelo cliente-servidor, muitas vantagens podem ser observadas no modelo distribuído. Em contrapartida, dificuldades são encontradas, principalmente, no que diz respeito à necessidade de troca de mensagens para garantia de consistência no modelo. Características positivas são vistas no modelo distribuído, tais como: transparência, balanceamento de carga, processamento distribuído, os quais trazem uma enorme vantagem aos sistemas adeptos desse modelo.

Uma abordagem promissora para construir esses sistemas distribuídos de larga escala na Internet são as redes overlay (AMIR e DANILOV, 2003). Em princípio as aplicações distribuídas poderiam gerenciar seus recursos acessando diretamente a rede subjacente, mas uma rede overlay pode oferecer serviços como manutenção da rede, segurança e alta disponibilidade que dificilmente poderiam ser providos no nível de rede. Em redes P2P as redes overlay têm sido utilizadas como forma de tornar o sistema escalável (DOVAL e O'MAHONY, 2003), outras redes overlay destinam-se a melhorar os serviços oferecidos pelos protocolos da Internet, como suportar QoS ou multicast (QBONE, 2005) ou ainda oferecer formas de roteamento alternativas (ANDERSEN, BALAKRISHNAN, *et al.*, 2001).

Nesse sentido, nos últimos anos surgiram novas tecnologias integradoras que possibilitam a organização de recursos computacionais dispersos na Internet de modo a facilitar seu uso por usurários ou software na rede, de forma transparente.. Dentre essas tecnologias, destacam-se os sistemas *Peer-to-Peer*, as grades computacionais (*grid computing*), os aglomerados (*clusters*) e os agentes móveis.

As limitações de sistemas cliente-servidor tornaram-se evidentes num ambiente distribuído da Internet. Os recursos estão concentrados em um pequeno número de peers, que devem aplicar algoritmos sofisticados de carregamento balanceado e tolerância a erros para prover acesso contínuo e confiável. Uma alternativa aos sistemas cliente-servidor tradicional são os sistemas *Peer-to-Peer* (P2P) (LEUF, 2002). Nestes, cada peer (*peer*) do sistema funciona como cliente e servidor ao mesmo tempo e fornece parte da informação geral disponível pelo sistema.

Redes *P2P* são redes virtuais que funcionam na Internet com o objetivo de compartilhar recursos entre os participantes, sendo que, por princípio, não há diferenciação entre os participantes. O grupo de pesquisa sobre redes P2P da IRTF (IRTF, 2006) as define como “o compartilhamento de recursos e serviços computacionais diretamente entre sistemas”. Esses recursos são necessários para prover os serviços e conteúdo oferecidos pela rede que passam a ser acessíveis por todos os pares sendo esses provedores e demandadores desses mesmos recursos .

As razões pelas quais este modelo de computação é atrativo podem ser descritas por três aspectos: redes P2P são *escaláveis*, lidam eficientemente tanto com grupos pequenos quanto com grupos grandes de participantes; é possível uma dependência maior do funcionamento dessas redes, já que não possuem ponto central de falhas e resistem melhor a ataques intencionais como os de negação de serviço; redes P2P oferecem *autonomia* a seus participantes, possibilitando que *entrem* e *saiam* da rede de acordo com seu interesse e disponibilidade, bem como que tomem suas decisões sem depender de entidades externas. Outros fatores que determinam o sucesso dos sistemas P2P são os diversos benefícios, tais como: alta escalabilidade, auto-organização, balanceamento de carga, processamento paralelo e tolerância a falhas por meio de massiva replicação. Além disso, sistemas P2P podem ser muito úteis no contexto de computação móvel ou computação pervasiva. No entanto, existem sistemas que só funcionam para aplicações simples (ex.: compartilhamento de arquivos), suportam funções limitadas (ex.: somente busca por palavra-chave) e usam técnicas simples as quais causam problemas de performance. (MARTINS, PACITTI e VALDURIES, 2007). Muitas pesquisas têm desafiado os problemas referentes aos sistemas P2P e dentre esses podemos destacar o alto grau de compartilhamento de dados com segurança, eficiência e consistência.

2.2 CARACTERÍSTICAS DE SISTEMAS DISTRIBUÍDOS

Um sistema distribuído consiste em uma coleção de componentes, distribuídos entre vários computadores conectados via uma rede. Esses componentes precisam interagir entre si, a fim de trocar dados ou acessar os serviços uns dos outros. Embora essa interação possa ser construída diretamente no topo das primitivas do sistema operacional, isso seria extremamente complexo para muitos desenvolvedores de aplicações. Em vez disso, usa-se o suporte de sistemas de *middleware*, localizado entre componentes do sistema distribuído e componentes do sistema operacional, cuja tarefa é facilitar as interações entre esses componentes.

A definição de sistemas distribuídos apresentada acima se aplica tanto a sistemas fixos quanto aos sistemas móveis. As principais diferenças entre os dois tipos podem ser analisadas segundo três conceitos, que irão também influenciar o tipo de sistema de *middleware* a ser utilizado: o conceito de dispositivo, de conexão de rede e de contexto de execução.

Quanto ao tipo de dispositivo, sistemas distribuídos fixos são compostos por dispositivos fixos, que podem variar de PCs a estações de trabalho Unix e

computadores de grande porte. Sistemas distribuídos móveis são compostos por dispositivos móveis que variam de PDAs a celulares e *smartcards*. Enquanto os primeiros são, em geral, máquinas poderosas, com grande quantidade de memória e processadores rápidos, os últimos têm capacidades limitadas, como baixa velocidade de CPU, pouca memória, baixa potência de bateria, entre outras características.

Quanto ao tipo de conexão de rede, dispositivos fixos são em geral conectados, permanentemente, à rede por meio de enlaces contínuos de grande largura de banda. As desconexões, quando ocorrem, são realizadas explicitamente por razões administrativas ou são causadas por falhas imprevistas. Essas falhas são consideradas esporádicas e, portanto, tratadas como exceção ao comportamento normal do sistema. Já os sistemas móveis sem fio podem obter largura de banda razoável apenas se os dispositivos estiverem dentro do alcance de poucas centenas de metros de sua estação base. Caso o número de dispositivos conectados, simultaneamente, aumente muito, a largura de banda cai rapidamente. Além disso, se o dispositivo se mover para uma área sem cobertura ou com alta interferência, a largura de banda pode cair para zero e a conexão pode ser perdida. A ocorrência de desconexões imprevistas passa a não ser mais exceção, e sim parte do comportamento normal para comunicações sem fio.

Quanto ao tipo de contexto de execução, esse conceito pode ser apresentado como tudo o que influencia o comportamento de uma aplicação, incluindo recursos internos ao dispositivo, como quantidade de memória, energia disponível, e recursos externos, como largura de banda, qualidade da conexão de rede, localização, dispositivos (ou serviços) vizinhos, etc. Em um ambiente distribuído fixo, o contexto é mais ou menos estático: a largura de banda é alta e contínua, a localização quase nunca muda, dispositivos são adicionados ou removidos do sistema com pouca frequência. Os serviços disponíveis podem mudar, mas a descoberta de serviços é facilmente realizada forçando-se os fornecedores de serviços a se registrarem com um serviço de localização bastante conhecido.

Por outro lado, em sistemas móveis o contexto é extremamente dinâmico. Dispositivos podem entrar e sair do sistema com frequência, e os serviços que estão disponíveis, quando ocorre uma conexão na rede, podem não estar mais acessíveis quando ocorrer uma nova conexão. A procura de serviço é muito mais complexa em cenários móveis. A localização do dispositivo não é fixa e, dependendo de onde ele está e se há ou não movimento, a largura de banda e qualidade da conexão podem também variar muito.

De acordo com o tipo de dispositivo, da conexão de rede e do contexto sobre o qual é construído um sistema distribuído, podem-se distinguir três classes de sistemas distribuídos: tradicionais, *ad-hoc* e nômades. Segundo essa classificação, sistemas distribuídos fixos são categorizados como sistemas tradicionais. Os principais requisitos não funcionais para esses sistemas são a escalabilidade, a heterogeneidade dos componentes, a tolerância a falhas e o compartilhamento de recursos. Sistemas distribuídos móveis *ad-hoc* (ou simplesmente *ad-hoc*) consistem em um conjunto de dispositivos móveis, conectados à rede de modo intermitente por meio de enlaces de qualidade altamente variável e executando em ambientes também altamente dinâmicos. Eles se diferenciam dos sistemas tradicionais por não terem uma infraestrutura fixa. Entretanto, os requisitos não funcionais listados para sistemas distribuídos tradicionais permanecem os mesmos alguns problemas tornam-se ainda mais graves.

Já os sistemas distribuídos nômades situam-se entre os sistemas fixos tradicionais e os sistemas *ad-hoc*. São baseados em um núcleo de roteadores, switches e computadores fixos. Porém, na periferia dessa infraestrutura de rede fixa, estações-base com capacidades de comunicação sem fio controlam o tráfego de mensagens de/para configurações dinâmicas de dispositivos móveis.

Como a infraestrutura física é diferente, as soluções de middleware propostas para sistemas tradicionais não podem ser empregadas com sucesso em ambientes *ad-hoc* ou nômades. São necessárias novas soluções, especialmente projetadas para tais ambientes.

2.3 COMPUTAÇÃO CLIENTE-SERVIDOR

O modelo de computação Cliente-Servidor é constituído de duas partes lógicas. Dessa forma, ele descreve o relacionamento entre essas partes, que são processos executados em computadores interligados por meio de conexões de rede. Tais partes lógicas, então, são chamadas de cliente e servidor. O cliente é um processo que requisita os serviços disponíveis na outra parte lógica, o servidor. O servidor espera por requisições feitas pelos vários clientes que o acessam. Por isso, quando o servidor recebe as requisições, ele as trata e, logo após, devolve uma resposta para a origem da requisição. A interação entre o cliente e o servidor conhecida como comportamento requisição-resposta, que é mostrado na Figura 1.

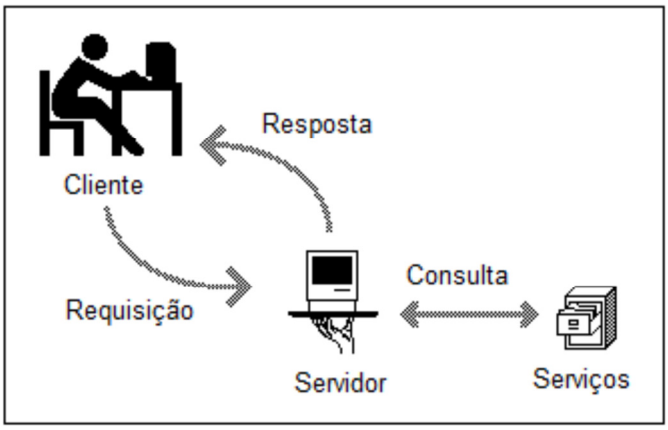

Figura 1: Modelo de Computação Cliente-Servidor

Segundo Tanenbaum e van Steen (TANEMBAUM e VAN STEEN, 2007), as aplicações construídas com base no modelo de computação cliente-servidor, em sua maioria, seguem um modelo arquitetural em camadas, que divide uma aplicação em três camadas. São elas:

1. **Camada de interface de usuário:** contém um conjunto de elementos necessários para garantir a acessibilidade e a usabilidade da aplicação por parte do usuário. Além disso, é nessa camada que a lógica de uma aplicação é desenvolvida, pois o tratamento das requisições do usuário é realizado por meio de controladores, que são acionados quando o usuário interage com os controles da interface gráfica. Apesar disso, a interface gráfica com o usuário pode ser uma tela baseada em caracteres, dispensando assim um conjunto de componentes de interface gráfica, tais como janelas, caixas de diálogos, botões, menus, lista de seleção, entre outros.
2. **Camada de aplicação**: contém a lógica de domínio da aplicação. Nessa camada, são implementados os códigos necessários para que a aplicação atenda aos requisitos impostos por um domínio específico. Além disso, a camada de aplicação é responsável, de maneira geral, por todo o processamento de uma aplicação. Assim, ela é capaz de recuperar qualquer informação da camada abaixo, processá-la de acordo com a necessidade do usuário e, por fim, devolver os dados alterados para a camada abaixo.
3. **Camada de dados:** gerencia os dados sobre os quais está sendo executada alguma ação. Uma importante propriedade dessa camada é

> que os dados costumam ser persistentes, isto é, ainda que nenhuma aplicação esteja em execução, os dados estarão armazenados em algum lugar e em algum formato para a próxima utilização. Em uma forma mais simples, por exemplo, a camada de dados é formada por um sistema de arquivos. No modelo de computação cliente-servidor, acamada de dados é implementada no lado servidor.

Fisicamente, o modelo de computação Cliente-Servidor pode ser dividido conforme a Figura 2. Uma máquina cliente contém apenas o software que implementa a camada de interface gráfica com o usuário. Já a máquina servidor contém as implementações da camada de aplicação e da camada de dados.

Por fim, estruturalmente, um modelo de computação Cliente-Servidor pode seguir uma estrutura plana, onde todos os clientes se comunicam com um único servidor. Além dessa estrutura, o modelo discutido nesta seção pode seguir uma estrutura hierárquica, na qual os servidores de um nível atuam como clientes de servidores do nível acima, com o objetivo de melhorar a escalabilidade.

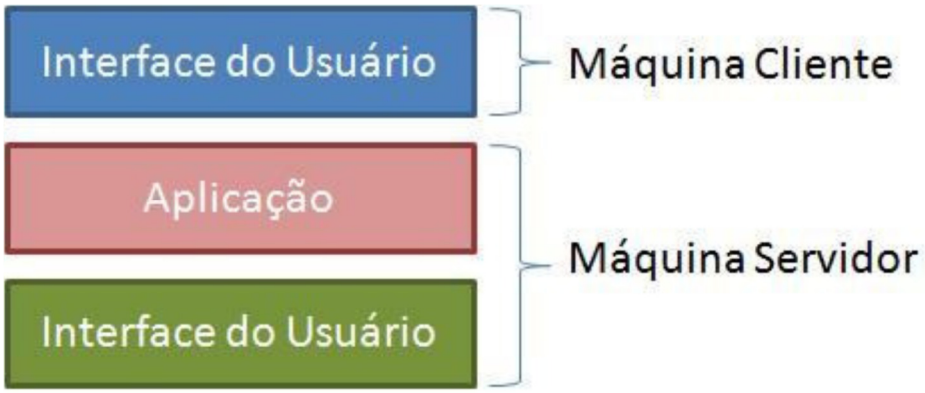

Figura 2:**Arquitetura de software de sistemas cliente-servidor (TANEMBAUM e VAN STEEN, 2007)**

2.4 COMPUTAÇÃO NAS NUVENS

De acordo com (BOSS, MALLADI, *et al.*, 2007), computação nas nuvens é um termo para descrever uma plataforma e tipo de aplicação. Assim, uma plataforma de computação nas nuvens pode ser entendida como uma infraestrutura de software e hardware que dinamicamente conecta e desconecta servidores, tanto físicos como virtuais, a fim de atender uma demanda computacional. Tal plataforma é disponibilizada na infraestrutura da Internet como uma entidade computacional chamada de nuvem. Com isto, aplicações construídas sob o modelo de Computação

nas Nuvens são aplicações *Web* que usam os recursos de uma nuvem como meio de hospedagem física e execução de suas funcionalidades. Essas aplicações têm suas funcionalidades definidas por meio de serviços *Web*.

Atualmente, os serviços de uma nuvem podem ser agrupados em três grandes categorias (CHAPELL, 2008), a saber: software como serviço, serviços anexados plataformas de nuvem. A Figura 3 mostra as categorias citadas e os relacionamentos com seus respectivos usuários diretos.

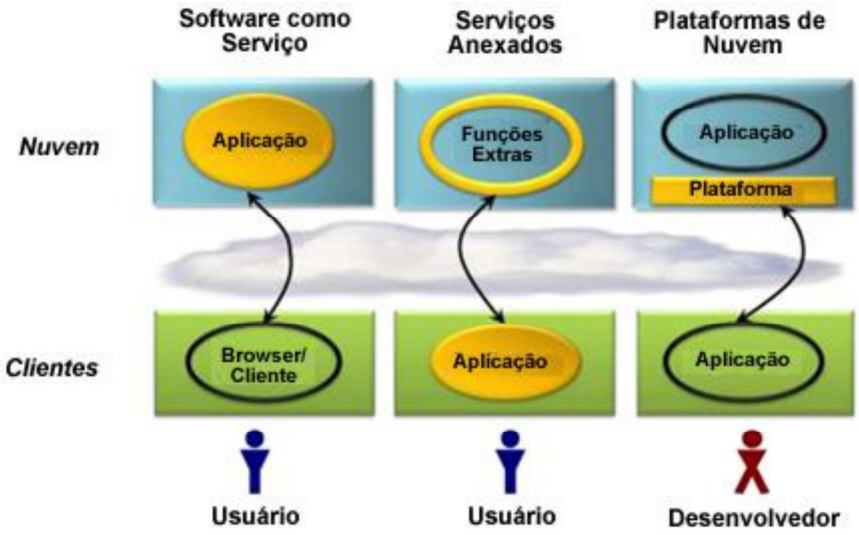

Figura 3 - Categoria de serviços de nuvens (CHAPELL, 2008)

Em computação nas nuvens, uma aplicação de software como serviço executa inteiramente na nuvem. Esta é usada por usuários comuns, por meio de *browsers,* ou algum outro software cliente simples. Os serviços anexados são serviços específicos oferecidos pela infraestrutura da nuvem que podem ser anexados a uma aplicação de usuário. Já as plataformas de nuvens fornecem serviços baseados na nuvem para criação de aplicações. Por exemplo, em vez de desenvolvedores construírem suas próprias arquiteturas, eles poderiam criar aplicações de software como serviço a partir de uma plataforma. A partir desta, então, cada desenvolvedor pode tirar proveito de componentes específicos para acesso às funcionalidades e serviços da nuvem.

Para que um software seja disponibilizado como um serviço em uma nuvem, essa deve oferecer uma plataforma de aplicação. De acordo com (CHAPELL, 2008), uma plataforma de aplicação é compreendida basicamente por três partes, que são: estrutura básica, serviços de infraestrutura e serviços de aplicação. O relacionamento entre essas partes e uma aplicação é apresentado por meio do modelo da Figura 4.

Figura 4- Modelo de plataforma de aplicação (CHAPELL, 2008)

A estrutura básica do modelo, apresentado pela Figura 4, compreende os recursos disponíveis no computador em que a aplicação executa, tais como bibliotecas padrão (.Net*framework* e Java), bancos de dados (Oracle DBMS, MySQL, Microsoft SQL Server e IBMDB2) e um sistema operacional (Windows, Linux e outras versões de Unix). Por meio da estrutura básica, uma aplicação pode acessar os serviços de infraestrutura da nuvem (armazenamento remoto, integração, identificação, entre outros), bem como acessar serviços de outras aplicações, uma vez que essas estão se tornando cada vez mais orientadas a serviços (CHAPELL, 2008).

2.5 COMPUTAÇÃO AUTÔNOMA

Os avanços na tecnologia de computação e redes de computadores, bem como em ferramentas de software têm causado uma explosão no número de aplicações conectadas a redes de computadores e a serviços de informação, presentes em todas as áreas de conhecimento e dos setores produtivos. Neste número, destaca-se um conjunto de aplicações e serviços extremamente complexos, heterogêneos e dinâmicos. Além disto, infraestruturas computacionais agregam grandes números de recursos de hardware, software e dados. Em tais infraestruturas, complexidade, heterogeneidade e dinamismos são características inerentes. Assim, a combinação entre aplicações e serviços, e infraestruturas computacionais complexas, heterogêneas e dinâmicas, tem aumentado consideravelmente a complexidade em atividades de gerenciamento, configuração e desenvolvimento de software. Por isto, a

comunidade acadêmica tem investigado um novo modelo de computação distribuída, conhecido como computação autônoma, para tratar, em ambientes computacionais distribuídos, de problemas de escala, complexidade, heterogeneidade e incerteza em ambientes computacionais (PARASHAR e HARIRI, 2005).

Expostos requisitos acima, pode-se inferir que o modelo de computação autônoma é aquele em que sistemas de software e computadores podem se gerenciar, à medida que mudanças de contexto afetam os seus comportamentos. Entende-se como contexto tudo o que influencia o comportamento de um sistema, incluindo recursos internos ao mesmo, como a quantidade de memória e a energia disponível; e recursos externos, como largura de banda, qualidade da conexão de rede, localização do dispositivo, dispositivos vizinhos, entre outros. Assim, a partir das mudanças de contexto, mecanismos presentes em uma arquitetura de computação autônoma reagem, fazendo com que o sistema ou aplicação se recupere de uma falha e, em seguida, reconfigure-se, de modo a buscar um estado de equilíbrio, baseando-se em condições do ambiente de execução.

Aplicações e sistemas autônomos são compostos por elementos autônomos e são capazes de gerenciar seus comportamentos e relacionamentos com outros sistemas e aplicações, em concordância com as preferências de seus usuários (PARASHAR e HARIRI, 2005). Assim, um elemento autônomo é um elemento arquitetural, podendo ser um software autocontido ou um módulo de sistema, com interfaces de entrada/saída e dependência de contextos específicos (PARASHAR e HARIRI, 2005). Além disto, elementos autônomos também têm mecanismos para seu próprio gerenciamento, com responsabilidades de executar suas funcionalidades; exportar restrições; gerenciar seus comportamentos de acordo com o contexto e regras; e interagir com outros elementos.

Um elemento autônomo, segundo (PARASHAR e HARIRI, 2005)é divido em três partes. São elas:

- **Elemento gerenciado:** é a menor parte funcional da aplicação e contém o código executável. Durante o seu tempo de execução, um elemento gerenciado pode ser afetado de diversas formas, por exemplo, falhas, indisponibilidades de recursos computacionais, baixo desempenho do ambiente de execução, entre outras.
- **Ambiente:** representa todos os fatores que podem impactar no elemento gerenciado. Qualquer mudança no ambiente faz com que uma aplicação ou sistema saia de um estado estável para um estado instável. Isso faz com que o elemento gerenciado reaja às mudanças de

ambiente, de modo que a aplicação ou sistema volte a um estado estável. As mudanças de ambiente podem ser internas, quando refletem o estado da aplicação ou sistema, ou externas, quando refletem o estado do ambiente de execução.

- **Controle:** cada elemento autônomo tem seu próprio gerenciador, que: aceita requisitos especificados pelo usuário (por exemplo: desempenho, tolerância à falha, segurança, entre outros); monitora o estado do elemento; captura o contexto do sistema ou aplicação; determina o estado do ambiente (estável ou instável); usa informações de contexto para controlar a operação do elemento gerenciado.

2.6 COMPUTAÇÃO P2P

Não existe consenso quanto à definição de rede P2P, tampouco quanto à sua classificação, com diversas propostas podendo ser encontradas na literatura. Uma primeira categorização, divide as redes P2P (LIV, CAO, *et al.*, 2002) em três categorias, descritas a seguir: centralizadas, descentralizadas estruturadas, e descentralizadas não estruturadas.

- Centralizada apresenta um ou mais servidores centrais, responsáveis pela coordenação dos *peers* ou pela manutenção de índices de busca.

- Descentralizada estruturada não apresenta servidores centrais, mas tem um alto grau de estruturação, com a atribuição de responsabilidades especiais a certos peers, a topologia sendo controlada, ou os recursos sendo posicionados de maneira estratégica na rede.

- Descentralizada não estruturada não apresenta qualquer diretório centralizado e não mantém um controle rígido sobre a topologia ou o posicionamento dos recursos.

Em algumas aplicações, as redes P2P descentralizadas não estruturadas oferecem vantagens sobre o modelo cliente-servidor. Estas vantagens surgem em razão das características relacionadas à sua arquitetura distribuída e descentralizada, como a robustez, a tolerância a falhas, e a utilização da capacidade de processamento ou armazenamento de *peers* ociosos. Porém, as características relacionadas à descentralização e ao dinamismo da topologia das redes P2P descentralizadas não estruturadas são aquelas de maior importância para este trabalho.

Uma segunda proposta (ATC, 2002) considera dois modelos de rede P2P:

- Descentralizado – uma rede onde não há um ponto central e cada peer tem o mesmo nível. Neste modelo, todos os *peers* são autônomos e responsáveis por troca de recursos e por controle (gerenciamento) de recursos. Os *peers* podem se comunicar de maneira direta ou por meio de vizinhos comuns (melhor escala de informações de controle, pior escala de tráfego de rede);
- Sem centralizado – uma rede onde há diferença de relevância entre os *peers*, havendo um *peer* central para informações de controle (e, possivelmente, para tráfego de dados) ou um conjunto de *superpeers* que assume tais funções (onde a queda de um *super*p*eers* afeta apenas os *peer* inferiores ligados a ele). Os *peers* inferiores podem ter maior ou menor nível de autonomia, mas são equivalentes entre si.

Na terceira categorização (TOWSLEY, 2003), também bastante utilizada, existem três tipos de arquitetura de rede P2P:

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

Para simplificar, consideramos as três classes principais de topologia: não-estruturado, estruturado e *superpeers*. Sistemas não-estruturados e estruturados são chamados sistemas P2P "puros" enquanto que sistemas de *superpeers* são qualificados como "híbridos". Sistemas P2P puros consideram que todos os *peers* são iguais, ou seja, nenhum *peer* tem algum tipo de funcionalidade específica dentro do sistema.

2.6.1 Sistemas Não Estruturados

Em ambientes P2P não estruturados, a rede é criada de uma maneira não determinística (*ad hoc*) e a localização dos dados não tem nenhuma relação com a topologia da rede. Cada *peer* conhece seus vizinhos e se comunica com eles, mas não conhece os recursos que eles têm. O roteamento das consultas é tipicamente feito pela inundação (*flooding*) (GENÇ, 2005) da consulta para os *peers* vizinhos em uma determinada distância de passos *i* que originou a consulta. Mecanismos de consulta baseados em inundação geralmente não consultam toda a rede em razão do grande tráfego que isso acarretaria. Com isso, a possibilidade de uma informação não ser encontrada é alta se o *peer* que contém a informação estiver muito longe do que originou a consulta.

Como todos os *peers* são igualmente capazes de trocar informações, é necessário que a tolerância a falhas seja muito alta. A Figura 5 ilustra um sistema não-estruturado simples onde os *peers* possuem alta autonomia. Cada *peer* suporta o mesmo tipo de comunicação P2P. Deste modo, basta que um *peer* conheça seu vizinho para poder se comunicar com ele, ou consultar seus dados (VALDURIEZ e PACITTI, 2005).

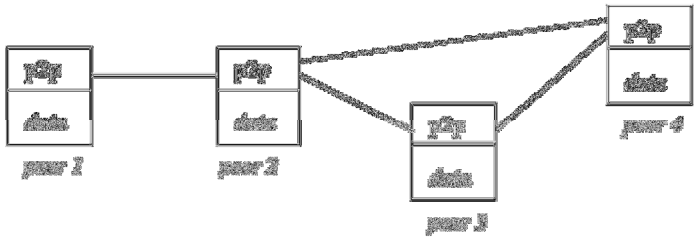

Figura 5– Rede ponto-a-ponto não estruturada. (VALDURIEZ e PACITTI, 2005)

2.6.2 Sistemas Estruturados

Sistemas P2P estruturados têm emergido para resolver o problema de escalabilidade dos sistemas não estruturados. Eles atingem esse objetivo controlando a topologia da rede e a localização dos dados. Dados (ou ponteiros para eles) são encontrados precisamente em uma localização específica e mapeamentos entre

dados e suas localizações são providos na forma de uma tabela de roteamento distribuído (MARTINS, PACITTI e VALDURIES, 2007).

Um dos maiores representantes desse tipo de sistemas são os sistemas estruturados baseados em tabelas *hash* distribuídas (DHT), como por exemplo, CAN (RATNASAMY, FRANCIS, *et al.*, 2001) CHORD (STOICA, MORRIS, *et al.*, 2001). Um sistema DHT provê uma interface para uma tabela *hash* com primitivas *put(chave,valor)* e *get(chave)*, em que *chave* é um identificador único e cada *peer* é responsável por armazenar o valor (conteúdo associado ao identificador). Existe um esquema de consulta sobre a rede que entrega a requisição, por uma chave, para um *peer* responsável por aquela determinada chave. Isto permite desempenho da ordem de *O(log n)* nas consultas, onde *n* é o número de *peers* na rede.

Como um *peer* é responsável por armazenar o valor correspondente para um grupo de chaves, a autonomia é limitada. Além disso, consultas DHT restringem-se a uma comparação exata de chaves, embora alguns pesquisadores estejam trabalhando com o objetivo de estender a capacidade de consultas em DHTs (VALDURIEZ e PACITTI, 2005).

2.6.3 Superpeers

Diante do fato de que as redes P2P são compostas por um número elevado de nós com capacidades (processamento, banda de rede, entre outros) bastante heterogêneas e possuem um comportamento altamente dinâmico (os peers entram e saem da rede constantemente), novos modelos têm sido propostos para lidar com essas características, e neles, nós com melhores recursos desempenham funções especiais. Esses nós são chamados *superpeers* e atuam como servidores para um conjunto de clientes e como um nó igual a outros em uma rede de *superpeers*.

Embora a abordagem de *superpeers* produza uma melhora significativa no desempenho das redes P2P, nos modelos atuais cada *superpeers* assume a responsabilidade sobre os recursos disponibilizados por um conjunto de outros nós. Isso significa que eles respondem às pesquisas endereçadas a esses nós e realizam o roteamento das mensagens circulando na rede, conforme apresentado pela Figura 6.

As principais vantagens das redes com *superpeers* são a eficiência percebida pelo usuário, como por exemplo, o tempo de resposta da consulta, e a qualidade do serviço. O tempo necessário para encontrar uma informação pelo acesso direto ao índice em um *superpeers* é muito menor do que na inundação da rede (*flooding*). Além disso, as diferentes capacidades dos *peers* em relação ao poder da CPU, largura de

banda ou capacidade de armazenamento, são levadas em conta quando se promove um *peer* a *super*p*eers*. Nas redes P2P puras, todos os *peers* são igualmente carregados, sem levar em consideração suas capacidades. Controle de acesso e segurança da informação podem ser mais bem gerenciados em *superpeers*. Entretanto, a autonomia é restrita, visto que os *peers* não podem conectar-se livremente a qualquer *superpeers*. A tolerância a falhas é baixa, pois os *superpeers* são pontos únicos de falha para os seus *sub-peers*. No entanto, a mudança dinâmica de *superpeers* pode reduzir este problema (MARTINS, PACITTI e VALDURIES, 2007).

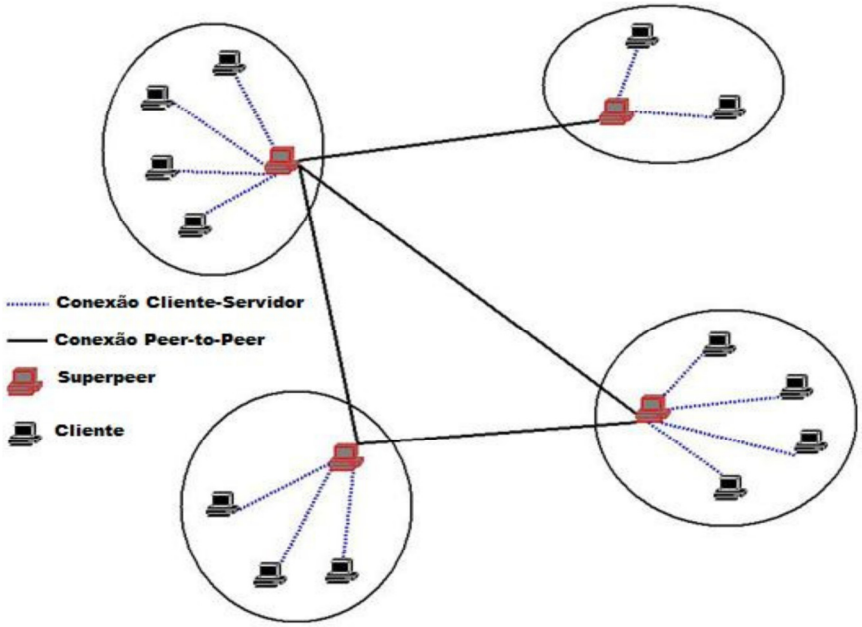

Figura 6- Rede de sobreposição baseada em *superpeers*

2.6.4 Comparativo entre redes P2P

Na Tabela 1 apresentamos a comparação entre os tipos de redes P2P descritos por (MARTINS, PACITTI e VALDURIES, 2007)baseada nos requisitos autonomia, expressividade das consultas, eficiência, qualidade do serviço, tolerância a falhas e segurança, apresentados no início desta seção.

Nota-se que os sistemas de superpeers são os que melhor atendem aos requisitos.

Tabela 1 - Comparativo entre os tipos de rede P2P

Requisitos	Não Estruturado	Estruturado	Superpeers
Autonomia	Alta	Baixa	Moderada
Expressividade das Consultas	Alta	Baixa	Alta
Eficiência	Baixa	Alta	Alta
Qualidade do Serviço	Baixa	Alta	Alta
Tolerância a Falhas	Alta	Alta	Baixa
Segurança	Baixa	Baixa	Alta

2.6.5 Características das redes P2P

Não existe consenso na literatura sobre o que exatamente são sistemas P2P ou quais são as características imprescindíveis de tais sistemas. Originalmente, P2P se refere a um estilo de arquitetura distribuída que contrasta com a cliente/servidor: sistemas distribuídos completamente descentralizados, em que todos os *peers* são equivalentes em termos de funcionalidade e tarefas que executam. Esta definição é purista e exclui diversas aplicações aceitas hoje como P2P.

Em (MILOJICIC, KALOGERAKI, *et al.*, 2003) (ATC, 2002)os autores descrevem as características que apresentam maior impacto na construção de sistemas e aplicações P2P. São elas:

- *Peers* podem estar localizados nas bordas da rede;
- *Peers* com conectividade variável ou temporária e endereços também variáveis;
- A capacidade de lidar com diferentes taxas de transmissão entre *peers*;
- *Peer* com autonomia parcial ou total em relação a um servidor centralizado;
- Assegurar que os *peers* possuem capacidades iguais de fornecer e consumir recursos de seus *peers*;
- A rede deve ser escalável; A capacidade de os *peers* se comunicarem diretamente uns com os outros.

Já em (THEOTOKIS e SPINELLIS, 2004), as duas características chave deP2P são:

- Compartilhamento direto de recursos entre *peers*, sem a intermediação de um servidor centralizado (embora se admita o uso de servidores em tarefas com menor demanda computacional ou de comunicação);
- Capacidade de auto-organização tolerante a falhas, assumindo conectividade variável e população transiente de *peer* como norma, automaticamente adaptando-se a falhas tanto nas conexões de rede como em computadores.

2.6.6 Topologia de Hipercubo em redes P2P

O Hipercubo (*HyperCube*), também conhecido como hipercubo binário , n-cubo binário ou apenas n-cubo, destaca-se por suas diversas características favoráveis; entre elas podemos citar a alta conectividade entre os *peers*, o que possibilita maior tolerância a falhas e fácil expansibilidade.

O hipercubo virtual é uma rede virtual formada por *peers* alocados em computadores na Internet integrados por um ambiente de rede P2P. Estes P2P realizam tarefas de processamento distribuído e trocam mensagens através de enlaces virtuais. Quando todos os *peers* que compõem o sistema estão sem falhas, os enlaces virtuais e os nodos formam um hipercubo. O hipercubo virtual é tolerante a falhas, sendo capaz de realocar automaticamente os nodos das máquinas que se tornam indisponíveis.

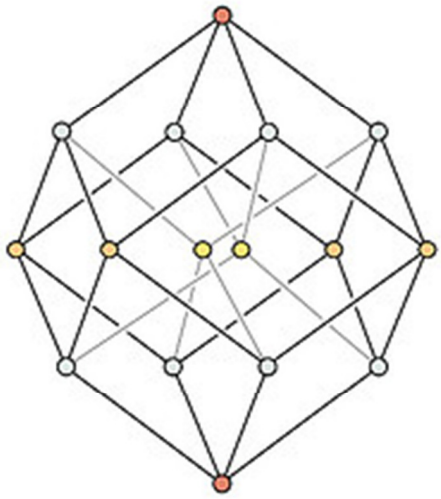

Figura 7 – Hipercubo de dimensão 4

O hipercubo também pode se caracterizado por possuir um único parâmetro chamado de dimensão do hipercubo. Esse parâmetro determina o número de *peer* da rede, o número de vizinhos que cada *peer* poderá ter e a menor distância entre os dois *peer* mais distantes que podemos chamar de diâmetro do hipercubo. Um hipercubo de dimensão n é composto de 2^n peers, cada um dos quais ligados an outros *peers*, um em cada dimensão, totalizando $n2^{n-1}$enlaces em todo o sistema. A Figura 7 apresenta um hipercubo de dimensão quatro.

O hipercubo foi amplamente difundido devido a uma série de vantagens que o oferece. Entretanto podemos citar com uma das mais importantes a sua compatibilidade com os padrões de comunicação da tarefa criados a partir da decomposição natural de muitos algoritmos numéricos. Sua rica estrutura de interconexão comporta em si a simulação de importantes topologias como linear, em anel em grade, entre outras. Essa mesma estrutura de interconexão permite que o diâmetro do hipercubo cresça de forma logarítmica com o número de peer. Outra vantagem está na facilidade de expansão física oferecida pela característica recursiva que esta topologia apresenta. Além do mais, o alto grau de conectividade entre os *peers* torna o hipercubo tolerante a falhas, já que existem vários caminhos de comunicação entre dois *peers*. (NASCIMENTO , 2000).

2.7 CONSIDERAÇÕES FINAIS

O modelo de computação cliente-servidor é o mais conhecido e amplamente utilizado para o desenvolvimento de aplicações distribuídas, tanto que os serviços de internet e aplicações Web são desenvolvidos e disponibilizados com base nesse modelo. No entanto, sistemas cliente-servidor não escalam bem quando a quantidade de usuários desses sistemas cresce. Neste caso, usuários disputam recursos computacionais e serviços ofertados por uma única entidade. Por isso, tais sistemas tendem a perder desempenho ou, até mesmo parar, quando a oferta de recurso computacional é insuficiente para atender à demanda dos usuários. Além disso, sistemas cliente-servidor necessitam de constante gerenciamento, ou seja, é necessário que um administrador possa verificar *logs*, consumo de recursos (percentual de uso do processador, espaço em disco, quantidade de memória livre, tráfego de rede, por exemplo)e também gerenciar o acesso dos usuários. Desta forma, o modelo de computação cliente-servidor torna-se caro e até mesmo difícil de manter ao longo do tempo.

Nesse sentido, o modelo de computação *peer-to-peer*, em relação ao modelo de computação cliente-servidor, permite o aproveitamento dos recursos de todos os participantes do sistema. Enquanto no modelo cliente-servidor, a oferta de recursos parte de um único ponto, o modelo de computação *peer-to-peer* permite que todos os integrantes do sistema compartilhem seus recursos, a fim de formar uma estrutura computacional que sirva para os mais variados propósitos. Assim, redes de sobreposição podem servir não somente para manter sistemas de compartilhamento de arquivos, mas também para manter ambientes distribuídos, onde os *peers* colaboram para a hospedagem de conteúdos e execução de software de forma cooperativa. Para isso acontecer, é necessário que os compartilhamentos de recursos na Internet migrem do centro para a periferia da nuvem, uma vez que na periferia existe um número maior de computadores que no centro. Além disto, os computadores localizados nas bordas da Internet possuem capacidade computacional maior ou igual à dos servidores localizados no centro da rede mundial de computadores. Entretanto, os recursos de hardware ficam ociosos, pois o modelo de computação vigente não permite a criação de uma via de sentido duplo para utilização desse potencial computacional.

Esta tese apropria-se dos conceitos e características estruturais do modelo de computação *peer-to-peer* para compor a camada de comunicação da arquitetura de software proposta por este trabalho. O objetivo dessa camada é compor uma infraestrutura computacional descentralizada, a partir de computadores conectados direta ou indiretamente à Internet. Com isto, recursos computacionais podem ser compartilhados, ofertados e trocados para execução de serviços. Uma vez que recursos tenham sido compartilhados sobre uma rede *peer-to-peer*, estes podem ser agregados a fim de compor aglomerados de recursos, de modo que possam ser utilizados para o armazenamento de conteúdo e a execução de software deforma distribuída. No entanto, aglomerados de recursos compartilhados devem ser vistos de maneira única, semelhante a uma nuvem computacional, ou seja, usuários não devem ter conhecimento sobre os detalhes de baixo nível que envolvem o compartilhamento, localização uso dos recursos.

Finalmente, para formar agregados computacionais, este trabalho apropria-se das dinâmicas que tangem as negociações estudadas pelos modelos econômicos em seus aspectos estruturais e propriedades. Com isto, as relações econômicas entre os usuários de um ambiente altamente distribuído são utilizadas para a construção de redes *peer-to-peer*. Assim, sob a ótica de que pessoas tendem a se comportar de maneira colaborativa quando estas possuem interesses comuns, o compartilhamento

de recursos pode formar um ambiente de computação capaz de prover acesso transparente a hardware, software conteúdos compartilhados. Mas, para manter o comportamento de uma rede *peer-to-peer* é necessário utilizar políticas de computação distribuída que enderecem requisitos de computação autônoma; São eles: autonomia, heterogeneidade, complexidade e incerteza. Por isso, as funcionalidades de uma aplicação distribuída devem ser disponibilizadas na forma de serviços.

O levantamento bibliográfico apresentado neste capítulo auxiliou a compreensão das teorias que estruturam fundamentação deste trabalho. Portanto, todo esse referencial teórico possibilitou a busca por abordagens, tanto arquiteturais quanto algorítmicas, que pudessem compor uma proposta metodológica para o compartilhamento e uso de recursos em ambientes computacionais. No entanto, faz-se necessária a investigação do estado da prática, no que diz respeito ao compartilhamento do modelo proposto de troca de recursos em redes *peer-to-peer*. Por isto, foram estudadas abordagens algorítmicas e arquiteturais que pudessem guiar esta pesquisa para a proposição de uma arquitetura de software e políticas de computação distribuída, ambos para endereçar o compartilhamento de recursos em redes *peer-to-peer*.

3 SISTEMA DE REPUTAÇÃO

Neste capítulo é abordada a importância dos Sistemas de Reputação dentro do contexto das redes P2P. São tratados, também, conceitos, mecanismos e modelos de reputação extraídos de referenciais teóricos e serviços disponíveis na *Web*, que fundamentam a pesquisa nessa área.

3.1 INTRODUÇÃO

A transiência dos usuários que fazem parte dos sistemas P2P torna pouco provável que eles se conheçam uns aos outros. Assim, as interações ocorrem entre entidades desconhecidas ou que não se conhecem bem (ROBINSON, VOGT e WAGEALLA, 2005), tornanda-os vulneráveis a ameaças de usuários mal intencionados. Embora as redes P2P possam oferecer uma maior resistência a determinados tipos de ataques intencionais (BARCELLOS e GASPARY, 2006), esses sistemas não estão imunes aos ataques de usuários maliciosos, que podem agir individualmente ou em acordo. Informações privadas podem ser compartilhadas em certos contextos e circunstâncias e torna-se importante identificar pares em quem confiar para passar tais informações.

O estabelecimento de confiança se torna necessário para permitir que as entidades possam trocar informações de forma segura, mesmo sem a intervenção de uma autoridade centralizadora. Essas interações podem ser realizadas com base no grau de desconfiança de cada entidade. A partir dos mecanismos de confiança, podem-se determinar quantos e quais recursos podem ser disponibilizados ou revelados para outras entidades (ROBINSON, VOGT e WAGEALLA, 2005). O gerenciamento de confiança por meio de mecanismos de reputação é reconhecido como um elemento importante da computação ubíqua para estabelecer confiança e facilitar as interações entre as entidades (LIU e QIU, 2007).

Os mecanismos de reputação agregam opiniões sobre o comportamento passado das entidades com o objetivo de estimar o seu comportamento futuro. Essa estimativa é baseada em valores que exprimem o grau de confiança nas entidades. Diversos mecanismos de reputação têm sido propostos na literatura para redes P2P; cada um deles exprime métricas para aplicações específicas que visam calcular um valor de reputação que será atribuído a cada entidade. Diversos mecanismos foram propostos, mas não existe uma avaliação sobre o seu desempenho e eficácia em situações práticas dentro da computação (ROBINSON, VOGT e WAGEALLA, 2005).

Diante dessa lacuna, há questões abertas sobre como esses modelos de gerenciamento de confiança respondem aos ataques de segurança que as entidades possam sofrer nesses ambientes.

Confiança e reputação representam uma significante tendência no apoio à decisão para a Internet mediante a prestação de serviços. Os termos reputação e confiança estão intimamente ligados. Há variação considerável nas definições de confiança na literatura, não havendo consenso sobre seu significado.

Aparentemente, não existe, hoje, um sistema de reputação que resolva os vários problemas provenientes de *peers* que não querem cooperar (agindo de má fé) ou de *peers* que estão atacando a rede ativamente. Existem, na literatura, muitas propostas tentando endereçar alguns dos problemas de reputação, porém eles apresentam fragilidades e não resolvem o problema de um modo geral.

A confiança de um *peer* em outro é baseada, naturalmente, na visão do *peer* sobre a reputação do outro. *Peers* adquirem boa reputação pelas interações bem sucedidas, que recebem uma avaliação positiva por parte dos *peers* envolvidos; por essa razão, esses sistemas são ditos baseados em *feedback*(retroalimentação).

Segundo (MARTI e GARCIA-MOLINA, 2006), sistemas de reputação têm em comum três partes principais: coleta de informações, determinação de escore e ranqueamento, e ações de resposta. A coleta de informações está relacionada ao esquema de identificação usado, às fontes de informação, à agregação de informações e à política adotada para *peers* que ingressam no sistema sem histórico associado (tais *peers* são definidos como *estranhos*). No extremo de precaução, um *peer* confia como fonte de informação apenas em suas informações locais. Um usuário cauteloso pode aumentar as fontes de informação perguntando a opinião de outros em que confia, com base em uma relação externa prévia, ou um *peer* pode perguntara outros *peers* (vizinhos ou *peers* com quem já interagiu com sucesso) sobre um determinado *peer*. Se insuficiente, um *peer* A pode solicitar a um *peer* B (em que confia), que pergunte aos *peers* em que B confia, sobre um *peer* C, de forma recursiva e transitiva. Aqui há um *trade-off* entre segurança e desempenho: aumentar o número de *peers* consultados (buscando qualificar as informações sobre outros *peers*) pode afetar negativamente o desempenho do sistema.

O *survey* (MARTI e GARCIA-MOLINA, 2006)sobre confiança discute fontes de informação que um *peer* pode buscar sobre outro *peer*, bem como as estratégias que um deles pode usar. As seguintes fontes são elencadas:

- Relação de confiança *a priori* com outro *peer*; fontes externas que sejam de confiança; *peers* que sejam de confiança e que estejam a um *peer* de distância;
- *Peers* que sejam de confiança e que estejam a vários *peers* de distância;
- Um sistema global de reputação.

As informações obtidas podem ser aplicadas por um *peer* com as seguintes estratégias de confiança: de forma otimista, quando assumir que todos os estranhos são de confiança até prova em contrário; de forma pessimista, quando ignorar todos estranhos até que sejam provados com confiança; investigar um estranho perguntando a *peers* de confiança; transitivamente quando propagar a investigação por meio de amigos de amigos; ou quando usar um sistema de reputação centralizado.

O problema de reputação não acontece apenas no mundo digital, mas também no mundo real onde pessoas podem enganar outras, agir de má fé, passar informações errôneas ou caluniosas.

O grande dilema está em quem confiar quando se trata de uma interação com outro *peer*. A única informação realmente segura que se tem é quando existem conhecimento e interação prévia com o *peer*, pois não podemos garantir que outros *peers* enviarão informações confiáveis sobre o *peer* em questão. Mas , e quando se interage com um *peer* pela primeira vez? Que abordagem deve ser tomada?

Sistemas de reputação coletam, distribuem e agregam informações sobre o comportamento anterior dos seus participantes. Estes sistemas ajudam os participantes envolvidos a decidir em quem podem confiar e a evitar a participação daqueles que não são confiáveis, isto é, que são desonestos (RESNICK, ZECKHAUSER, *et al.*, 2000).

Os sistemas de reputação em redes P2P (*Peer-to-Peer)* têm um papel importante no provimento da rede. Esses protocolos são responsáveis para auxiliar os *peers* a encontrar a fonte mais confiável de um determinado recurso que desejem utilizar. Normalmente, os sistemas baseiam-se em experiências anteriores para calcular um valor numérico que represente um valor de confiança com relação a um determinado peer possuidor do recurso ou mesmo ao próprio recurso (PELISSARI, RIGHI e WESTPHALL, 2004).

Um dos problemas fundamentais de sistemas de reputação é garantir a validade das informações prestadas por outros *peers*. Portanto, é natural que, na

determinação do escore de reputação de um estranho, experiências anteriores do próprio *peer* sejam valorizadas em relação à opinião de outros *peers*.

Uma abordagem comum nesse sentido é a aplicação de pesos: a reputação de uma informação dada por um *peer* é proporcional à reputação desse *peer*. Informações coletadas por meio de confiança transitiva podem ser pesadas de acordo com a reputação do *peer* de menor reputação na cadeia de confiança; alternativamente, se os valores de confiança situam-se em [0,1], então o valor resultaria da multiplicação dos escores de reputação de cada um dos *peers*.

Sobre gerência dos escores de reputação, (GUPTA, JUDGE e AMMAR, 2003) apresentam duas abordagens possíveis para mapear ações de *peer s*em um escore não negativo de reputação: uma é baseada em crédito e débito (para *peers* que prestam ou consomem um conteúdo/serviço, respectivamente), e a outra apenas em crédito, mas onde créditos expiram naturalmente com o passar do tempo.

Para seres humanos, é difícil atribuir um valor numérico que represente um valor de confiança com relação a outro ser humano. No mundo real, quando um ser humano deseja saber se outro é confiável, pergunta a um terceiro conhecido de ambos e pode-se obter uma resposta do tipo “ele é amigo”. Nota-se que esses rótulos reproduzem papéis que outros seres humanos representam. Portanto, os protocolos de reputação que seguissem essa ideia estariam mais próximos da realidade dos seres humanos e teriam melhor aceitação em uma rede P2P cujos *peers* representam seres humanos.

Existem muitos casos em que sistemas de reputação são interessantes para garantir alguns requisitos ou a melhor utilização do sistema. Um exemplo disso é utilizar a reputação para beneficiar ou punir um usuário dependendo da sua colaboração com a rede. Assim, um peer que tentasse atuar como um *free-rider* apenas consumindo recursos começaria a ser impedido com o tempo até que contribuísse com os outros *peers*.

Outro exemplo é o de comércio eletrônico como o *eBay* (EBAY, 1995) e o Mercado Livre (LIVRE, 1998) onde os usuários do sistema classificam uns aos outros. Essa classificação ocorre a partir da experiência que um vendedor teve com um comprador. Geralmente, somente usuários com boa qualificação conseguirão vender seus produtos sem problemas ou desconfiança. Nesse modelo, o *eBay* ou Mercado Livre atuam como gerentes de reputação dando uma certa garantia aos outros *peers* sobre as informações que eles distribuem. Mesmo com esse tipo de sistema, ainda existe o problema de traição, isto é, um *peer* tem comportamento correto durante um

período de tempo e, quando ganha bastante confiança, aplica um golpe prejudicando outro usuário.

3.2 DEFINIÇÃO DE CONFIANÇA

A literatura sobre confiança é considerada bastante confusa devido ao fato de o termo ser usado com uma variedade de significados. Segundo (GRANDISON e SLOMAN, 2000), confiança pode ser definida como a "firme crença na competência de uma entidade em agir de forma confiável e segura, em que se possa depender, dentro de um contexto especificado". Ainda segundo Grandison, a confiança é usualmente especificada em termos de uma relação entre uma entidade que confia, o fiador(*trustor)*, e outra que recebe a confiança, o depositário (*trustee*). Confiança é a base para permitir que um depositário use ou manipule recursos de um fiador, ou pode afetar a decisão de um fiador, ou, do contrário, em usar um serviço de um depositário.

Para (JOSANG, ISMAIL e BOYD, 2006) a confiança é uma medida individual e subjetiva que pode combinar experiência pessoal com referências advindas da comunidade (reputação). Também encontramos em Josang duas maneiras de definir confiança. A primeira é baseada na confiabilidade e afirma que confiança é a probabilidade subjetiva pela qual um indivíduo espera que outro indivíduo execute uma determinada ação em seu beneficio. A segunda, tem como base a confiança na tomada de decisão e define a mesma com base na possibilidade de uma das partes estar disposta a depender de algo ou de alguém em uma determinada situação, no sentido de obter uma segurança, embora sejam possíveis consequências negativas.

Segundo (GOLBECK e HENDLER, 2006), os indivíduos necessitam de uma definição para que saibam como descrever sua confiança para com outras pessoas. Os autores também descrevem algumas propriedades da confiança que são familiares no nosso cotidiano social. Essas propriedades nos ajudam a entender como a confiança pode ser propagada em um ambiente computacional, entre pessoas que não estão conectadas diretamente nas redes sociais, ou seja, não se conhecem ou nunca interagiram diretamente.

- *Transitividade:* a confiança não é perfeitamente transitiva no sentido matemático. Numa aplicação computacional, a confiança pode acontecer de duas maneiras: Confiar no indivíduo e confiar nas recomendações que ele faz sobre outras pessoas. Em muitos casos, confiamos no indivíduo, mas não confiamos na pessoa que ele recomendou devido à baixa reputação dela na

comunidade. Em outros casos, a opinião do indivíduo em quem confiamos pesa mais do que a opinião da comunidade.

- *Assimetria:* entre duas pessoas envolvidas em uma relação, a confiança não é, necessariamente, idêntica em ambas as direções porque os indivíduos têm experiências diferentes, envolvendo diferentes históricos de atuação e diferentes currículos. Diante disso, é aceitável e compreensível que as pessoas não confiem umas nas outras da mesma maneira. A assimetria é muito comum em interações nas quais existem diferentes *status* de hierarquia entre as pessoas. Por exemplo, alunos explicitam que confiam nos professores, mais do que os professores explicitam que confiam nos alunos.
- *Personalização*: a confiança é uma opinião pessoal. Isso significa que duas pessoas podem ter opiniões diferentes sobre um mesmo indivíduo, dependendo do resultado de experiências individuais anteriores. Em um ambiente computacional, a personalização baseada em confiança pode ser utilizada para fazer recomendações mais precisas para os usuários de uma comunidade. Nesse caso, o algoritmo de recomendação se baseia nas opiniões de pessoas em quem o usuário explicitamente confia, ou pode dar um peso maior nas opiniões dessas pessoas, do que nas de pessoas com alta reputação na comunidade.

Tanto no mundo presencial quanto no virtual, a confiança se manifesta de diferentes formas, envolvendo dependência na entidade confiada e o risco de se obterem resultados negativos. Portanto, a decisão sobre confiar ou não em alguém se baseia em fatores ou evidências que podem obter maior ou menor peso nessa decisão. Experiências pessoais (confiança direta) tipicamente têm maior peso do que referências de outras pessoas (confiança indireta), mas na ausência de uma experiência pessoal, a confiança é geralmente baseada nas referências externas.

3.3 DEFINIÇÃO DE REPUTAÇÃO

Uma das definições encontradas na literatura é a adotada em (JOSANG, ISMAIL e BOYD, 2006): “reputação é uma medida coletiva ou senso comum, baseada em notas ou referências advindas de membros da comunidade sobre a entidade avaliada”. Para esses autores, a reputação pode estar associada a um indivíduo ou a um grupo. De acordo com o dicionário Oxford, a reputação pode ser boa ou ruim, dependendo da opinião que as pessoas têm sobre alguém ou alguma coisa, com base

em situações ocorridas no passado que tenham levado à construção de uma boa ou má reputação.

Em (SABATER e SIERRA, 2001) os autores definem reputação como sendo uma opinião geral de uma comunidade formada sobre uma entidade e atualizada por meio de interações diretas dos usuários, ou de informações proporcionadas por outros membros da sociedade sobre experiências vividas com a entidade no passado. Para esses autores, a reputação está baseada em três dimensões: individual, social e ontológica. Na visão desses autores, o indivíduo herda a reputação do grupo ao qual ele pertence. Em muitas situações, quando não existem informações diretas por meio de interações realizadas com o indivíduo, a reputação do grupo ao qual ele pertence fornece expectativas iniciais sobre o seu comportamento. Isso acontece porque pertencer a certo grupo implica, *a priori*, que o indivíduo compartilha da mesma forma de pensamento do grupo.

A dimensão individual considera as impressões mais recentes do indivíduo sobre a entidade avaliada. Essa experiência pessoal contribui com a dimensão social, que considera as impressões do grupo ao qual o indivíduo pertence. Além disso, a reputação pode estar relacionada a aspectos específicos da entidade avaliada. A dimensão ontológica possibilita combinar reputações nesses diferentes aspectos. Por exemplo, a boa reputação de um vendedor está relacionada com a sua reputação em oferecer um produto de qualidade, com um bom preço e entrega rápida. As impressões dos outros sobre essas informações, quando combinadas, refletem o valor da experiência e constroem a reputação do vendedor.

Outro aspecto importante observado em (MUI, HALBERSTADT e MOHTASHEMI, 2002)é que a reputação é extremamente dependente do contexto. Sendo assim, o indivíduo pode ter reputações diferentes em cada contexto em que se encontra inserido. Por exemplo, a reputação de um profissional no seu ambiente de trabalho pode ser diferente da sua reputação como pai de família, ou como jogador de futebol dentro do seu grupo de amigos.

Em Sistemas de Reputação, é importante considerar como as informações coletadas sobre o indivíduo devem ser avaliadas na estimativa da sua reputação. Primeiro, porque ter uma “boa reputação” pode ter diferentes significados de grupo para grupo, dependendo das informações relevantes em questão. Segundo, porque dentro de um grupo, a reputação do indivíduo pode ser diferente em áreas de conhecimento específicas. Além disso, a reputação do indivíduo pode aumentar ou diminuir com o tempo, de acordo com a sua participação e colaboração com o grupo.

A reputação assim identificada pode ser utilizada para auxiliar os indivíduos a decidirem entre confiar ou não nos membros de uma comunidade dentro de um ambiente computacional.

3.4 MODELOS DE ARQUITETURAS PARA SISTEMAS DE REPUTAÇÃO

A arquitetura da rede determina como a pontuação é compartilhada entre os participantes em sistemas de reputação. O dois tipos principais são a arquitetura centralizada e a distribuída.

3.4.1 Sistema Centralizado de Reputação

Nos sistemas centralizados de reputação, informações sobre o desempenho de um dado participante é coletado a partir da avaliação de outros membros da comunidade que tenham tido experiência direta com esse participante. A autoridade central (centro reputação), que recolhe todas as classificações, obtém uma pontuação para cada participante e a coloca à disposição do público. Os participantes podem, então, usar a pontuação uns dos outros.

Segundo (JOSANG, ISMAIL e BOYD, 2006), são dois os aspectos fundamentais em um sistema centralizado de reputação:

- *Protocolo de comunicação centralizado* que permite aos participantes fornecerem uma avaliação sobre a transação com seus parceiros à autoridade central, bem como a obtenção de reputação pela pontuação de potenciais parceiros da autoridade central.
- *Mecanismo computacional para o cálculo da reputação* utilizado pela autoridade central para derivar pontuações da reputação de cada participante, com base em classificações recebidas e, possivelmente, também sobre outras informações.

3.4.2 Sistemas Distribuídos de Reputação

Em um sistema distribuído não existe um local central para a apresentação das avaliações de uma transação nem ao menos para obter a pontuação de reputação de outros membros. Em vez disso, pode haver locais de armazenamentos distribuídos onde as avaliações podem ser apresentadas, ou cada participante possui o seu

parecer sobre cada experiência com outros partidos, e fornece essas informações quando requisitadas.

Josang (JOSANG, ISMAIL e BOYD, 2006) também descreve dois aspectos fundamentais em um sistema distribuído de reputação:

- O protocolo de comunicação distribuída permite aos participantes obter a avaliação por meio de outros membros dentro da comunidade.
- O método computacional de reputação usado individualmente, por agente da rede, para derivar pontuações da reputação de cada participante, com base em classificações recebidas e, possivelmente, também sobre outras informações.

3.5 MECANISMOS DE REPUTAÇÃO E CONFIANÇA

Mecanismos de reputação e confiança são meios de, segundo critérios próprios adequados a diferentes situações, certificar que os diferentes membros da comunidade envolvidos no processo têm identidade idônea e histórica de trocas satisfatórias. Esses mecanismos podem ser usados para classificar os participantes de uma P2P em função de seu comportamento para com os demais. Devido à heterogeneidade dos *peers*, alguns podem ser benevolentes em prestar serviços. Porém outros *peers* podem ser maliciosos e não prestarem serviços com a qualidade que eles anunciam. Como em rede P2P não existe um *peer* centralizado, para servir como uma autoridade para fiscalizar e punir os *peers* que se comportam mal, os *peers* maliciosos usam esse fator como um incentivo para prestar serviços de má qualidade, em seu benefício, pois eles sabem que podem ficar impunes.

Algumas técnicas tradicionais de segurança, tais como autenticação requerida pelo provedor de serviço e pelos consumidores, são utilizadas como proteção contra os *peers* que possuem comportamento malicioso. No entanto, elas não podem impedir que os *peer* possam prestar serviços de qualidade variável. Mecanismos de confiança e de reputação podem ser usados para ajudar a distinguir bons e maus *peers*.

Mecanismos de reputação vêm sendo largamente utilizados. Um dos mais conhecidos e estudados sistemas de reputação é o mecanismo de "*feedback*" utilizado pelo *eBay*. Nesse sistema, compradores e vendedores podem trocar informações (que são disponibilizadas publicamente) sobre a sua satisfação com a transação. (RESNICK, ZECKHAUSER, *et al.*, 2000) concluem que os vendedores com as reputações mais elevadas vendem, aproximadamente, 7% mais do que os vendedores com baixa reputação. Reputação também tem sido um parâmetro adotado para

coordenar o comportamento de sistemas de compartilhamento de arquivos P2P. Nesses sistemas, o objetivo é aumentar a quantidade e a qualidade do conteúdo disponível na rede. Sistemas como Gnutella e Kazaa priorizam o *download* de clientes que tenham estabelecido uma boa reputação, provendo *uploads* (MCKNIGHT e HOWISON, 2003).

O principal desafio na construção de sistemas, para formalizar e estender as regras do comportamento por meio de reputação, é solucionar questões relativas a mecanismos de incentivo.

Os mecanismos de confiança são utilizados com o objetivo de auxiliar os usuários a entenderem os benefícios de demonstrarem sua confiança uns nos outros, de forma que essa confiança possa ser propagada na rede social. A propagação da confiança em um serviço mediado por meio da Internet automatiza o processo social de recomendação boca-a-boca (*word of mouth*), onde as pessoas solicitam a amigos, referências sobre profissionais confiáveis para a realização de um serviço.

Alguns mecanismos de reputação e confiança são descritos a seguir.

3.5.1 *Ranking* de pontuação

Esse mecanismo de reputação pode ser baseado em informações implícitas sobre a atuação do usuário no ambiente: frequência de visitas, participações, período como membro da comunidade, escores de desempenho, etc.; ou em informações explícitas recebidas por meio do retorno (*feedback*) de outros usuários.

Qualquer uma das duas formas de construção do *ranking* de pontuação pode ser bem aplicada em ambientes que valorizam a velocidade e a habilidade do usuário no desempenho das suas atividades. Por exemplo, em jogos multiusuários (*Multiplayer Online Games*), o *ranking* serve como um indicativo para os jogadores escolherem seus pares (TANG, LI, *et al.*, 2005);no comércio eletrônico serve como informação complementar sobre o desempenho de vendedores, compradores e revisores de produtos.

(JENSEN, DAVIS e FARNHAM, 2002)apontam, como vantagem desse tipo de mecanismo, a facilidade de computação e de exibição dos dados, e como desvantagem, o fato de essas informações revelarem pouco sobre a personalidade do usuário, indicando, apenas um padrão de comportamento que ele tenha.

3.5.2 *Feedback* da comunidade

Esse mecanismo de reputação possibilita que os usuários avaliem as interações com outros usuários. Essas interações podem ser vistas como a realização de um serviço, onde um usuário é o cliente e o outro é o fornecedor. Geralmente, o Sistema de Reputação requisita avaliações positivas e negativas de um usuário (cliente) sobre sua satisfação com relação ao serviço fornecido por outro usuário (fornecedor). Tais serviços podem ser: negociações de compra/venda de produtos (MercadoLivre, *Ebay*), consulta a avaliações de produtos (*Amazon*, *Epinions*), consultoria a especialistas (*AllExperts*), etc.

O *feedback* da comunidade permite fornecer informações mais precisas sobre o comportamento dos indivíduos associado ao *ranking* de pontuação, tais como: a trajetória do indivíduo ao longo do tempo (histórico da reputação); o total de qualificações positivas e negativas recebidas das contrapartes nas avaliações. Essas informações auxiliam o cliente a decidir entre confiar, ou não, no serviço de um fornecedor.

3.5.3 Redes de Confiança

Alguns sistemas permitem que os usuários construam suas próprias redes de confiança. Essa estratégia é utilizada pelo *Epinions*, no qual o usuário explicita revisores de produtos em que ele confia e bloqueia aqueles em que não confia (*Trust/Block*). Na página de perfil de um revisor do *Epinions,* fica visível a lista de pessoas em quem ele confia e de pessoas que o consideram confiável. A visibilidade da lista de bloqueio não é permitida para evitar conflitos entre os usuários.

Alguns sistemas permitem que o usuário indique níveis de confiança em outros usuários, sem que essa confiança fique visível para os membros da comunidade. Essa estratégia é, geralmente, utilizada por sistemas que permitem a formação de redes sociais, e precisam conhecer o grau de confiabilidade dos usuários em pessoas da sua rede para realizar recomendações precisas de itens.

O sistema *FilmTurst* (GOLBECK e HENDLER, 2006)requisita que usuários informem uma nota (grau) de confiança nos amigos que adicionam à sua rede social. Os usuários são previamente avisados de que devem indicar uma nota (de zero a dez) sobre o quanto confiam em um amigo para a recomendação de filmes.

3.6 TIPOS DE ESTIMATIVA DE REPUTAÇÃO

A partir das informações obtidas por meio dos mecanismos de reputação e confiança, é possível estimar a reputação dos usuários de um ambiente computacional. Existem diversos tipos de estimativa de reputação. Alguns deles são descritos a seguir.

3.6.1 Estimativa baseada na agregação de notas

Na maioria das vezes, a reputação é estimada a partir da média ou somatório das notas obtidas pelo *feedback* da comunidade. Essa estratégia é utilizada pelos sítios MercadoLivre e *Ebay* (somatório das notas), *Amazon* e *Epinions* (média das notas).

(JENSEN, DAVIS e FARNHAM, 2002) apontam como principal vantagem desse tipo de estimativa a facilidade de computação. No entanto, (DELLAROCAS, 2003) afirma que o uso de apenas média ou somatório, na estimativa de reputação, torna o sistema vulnerável a usuários desonestos que usam de artifícios para aumentar suas notas ou diminuir as notas de outras pessoas. Alguns mecanismos de imunização contra avaliadores mal intencionados são descritos mais adiante, nesse trabalho.

A credibilidade da estimativa por meio da agregação de notas também pode ficar prejudicada se a comunidade não fornecer avaliações suficientes. Mais um problema apontado por alguns autores como (DELLAROCAS, 2003) e (RESNICK, ZECKHAUSER, *et al.*, 2000), é que essa abordagem não identifica as diferenças de perfil dos avaliadores. Dois clientes podem avaliar o serviço de um mesmo fornecedor de forma diferente, porque têm preferências também diferentes sobre determinados aspectos do serviço. Por exemplo, um fornecedor de pacotes turísticos de baixo custo pode agradara um cliente que não esteja interessado no conforto da hospedagem e desagradara outro cliente que prefira pagar mais por uma hospedagem confortável e menos pelos passeios.

3.6.2 Estimativa baseada na similaridade de usuários

Em determinados contextos, os usuários tendem a concordar nas suas avaliações com frequência, mas em outros, isso não acontece. Por exemplo, em sítios de leilões, geralmente, ocorre um alto nível de concordância na qualificação negativa

de um vendedor que não cumpre o prazo de entrega de um produto. Já em sítios sobre cinema, os usuários podem avaliar, de forma diferente, as opiniões de um crítico, por não gostarem do mesmo estilo de filme.

O comportamento discriminatório de fornecedores é descrito por (DELLAROCAS, 2003) como a situação em que um fornecedor decide atender bem apenas a um grupo específico de clientes, e atender de forma razoável aos demais, garantindo a sua alta reputação no sistema. Nesse caso, esse tipo de comportamento pode afastar clientes com perfis similares, e aproximar clientes com perfis diferentes que foram beneficiados pelo mesmo fornecedor.

3.6.3 Estimativa baseada na confiança entre usuários

As redes de confiança são bons instrumentos de possíveis interações positivas, mas para isso não basta apenas a ausência de mau comportamento. As partes envolvidas devem tirar proveitos e se sentirem satisfeitas com a interação (JENSEN, DAVIS e FARNHAM, 2002). Na vida real, utilizamos informações sobre reputação de forma diferente do que é normalmente visto em ambientes de interação virtual. Os serviços que escolhemos dependem muito das referências dos nossos amigos.

Recomendações feitas explicitamente pelos “pares” de um usuário, ou inferidas implicitamente por meio da sua rede social, têm uma influência significativa no seu processo de tomada de decisão. Portanto, o contexto social proporcionado pelas recomendações de um amigo, ou de um amigo do amigo, vem sendo cada vez mais investigado em ambientes de interação virtual, como pode ser visto no trabalho de (GOLBECK e HENDLER, 2006).

Reputações baseadas nas redes de confiança podem ser estimadas a partir do *feedback* de pessoas que os usuários escolhem, o que possibilita uma estimativa mais confiável e de alta relevância social para o usuário. Entretanto, as desvantagens requerem alto custo de desenvolvimento com relação à propagação da confiança e lidam com questões relativas à privacidade. Além disso, se os usuários selecionarem pessoas pouco confiáveis para fazerem parte da sua rede, comprometerão os resultados da estimativa de reputação e ficarão expostos ao risco de obterem recomendações indesejáveis.

3.7 PROBLEMAS EM SISTEMAS DE REPUTAÇÃO

Motivar os usuários a utilizarem os mecanismos de reputação de forma adequada nem sempre é fácil. Em alguns contextos, existe o clima de competitividade entre fornecedores de serviços, que pode causar diversos problemas de avaliações desonestas. Nesses casos, convencer um usuário que tenha sido prejudicado, a voltar a utilizar um sítio é muito mais difícil do que atraí-lo para uma primeira visita.

Um conjunto de avaliações desonestas pode ser tão prejudicial para a estimativa confiável da reputação quanto a ausência de avaliações. O ambiente deve ser capaz de detectar usuários com comportamentos indesejados, puni-los, ou pelo menos alertar a comunidade da sua ocorrência.

Para isso, o Sistema de Reputação precisa utilizar alguns mecanismos que motivem os usuários a participarem, de forma honesta e produtiva, e tornem a estimativa de reputação menos vulnerável à ação de possíveis avaliadores desonestos.

De acordo com (DELLAROCAS, 2003) os principais efeitos identificados nas avaliações desonestas são:

– Falsas avaliações positivas "*ballot stuffing*" - são fraudes que visam a aumentar a reputação de usuários por meio de avaliações falsas positivas. Avaliadores desonestos cadastram perfis falsos no ambiente computacional ou combinam com um grupo de amigos, simulam interações e realizam um conjunto de avaliações positivas em benefícios próprios;

– Falsas avaliações negativas "*bad-mouthing*" – são fraudes que geram o efeito de difamação, diminuindo a reputação de usuários injustamente. Avaliadores desonestos criam perfis falsos ou combinam entre si para realizar um conjunto de avaliações negativas sobre um concorrente;

– Discriminação positiva - fornecedores oferecem um excelente serviço apenas para um grupo seleto de clientes e serviço razoável para os demais, de forma a garantir a sua alta reputação por meio da média acumulada por notas altas recebidas pelo grupo seleto e notas razoáveis recebidas pelos demais;

– Discriminação negativa - fornecedores proporcionam um serviço muito bom para a maioria, exceto para um pequeno grupo de clientes, de forma que as notas acumuladas pelas avaliações do grupo discriminado não são suficientes para baixar sua reputação. (DELLAROCAS, 2003) propõe alguns mecanismos de imunização contra esses tipos de comportamento:

– Uso do anonimato controlado – os efeitos de difamação e discriminação negativa são baseados na habilidade dos usuários em selecionar vítimas específicas para proporcionar notas baixas ou serviços de baixa qualidade. Sendo assim, esses efeitos poderiam ser controlados ocultando-se a identidade de fornecedores e clientes por meio de um pseudônimo e modificando essa identidade a cada nova negociação. Cada usuário seria autenticado de forma que apenas o sistema conheceria a sua identidade verdadeira. Os outros usuários fariam suas decisões de negociação apenas com base nas informações de reputação. O aspecto negativo é que esse tipo de solução fica limitado a situações em que seja possível omitir as identidades de clientes e fornecedores.

– Uso da mediana na agregação de notas – o anonimato controlado não resolve o problema de faltas avaliações positivas e discriminação positiva, visto que um grupo de usuários conspiradores pode encontrar formas de criar pseudônimos que permitam uma fácil identificação. Diante disso, a estimativa da reputação por meio do somatório ou média das notas recebidas pela comunidade fica vulnerável a situações em que avaliadores emitem uma grande quantidade de falsas notas positivas. O uso da mediana (valor do meio da distribuição) na estimativa da reputação é proposto como uma forma de dificultar o sucesso de avaliadores desonestos nas tentativas de contaminar as amostras das avaliações em benefício próprio.

– Análise da frequência de avaliações desonestas - a frequência com que os usuários são avaliados pelo mesmo grupo de pessoas pode ser um indicativo de formação de grupos de avaliadores desonestos. Entretanto, em alguns casos, fornecedores honestos mantêm um grupo de avaliadores que podem ser seus clientes fiéis. Portanto, o histórico de reputação do indivíduo deve ser registrado, e a reputação deve ser estimada de acordo com as avaliações mais recentes. Se o histórico da reputação do usuário for exibido para determinados períodos de tempo, é possível perceber uma mudança repentina nas suas notas mais recentes. Nesse caso, pode ser feita uma análise de frequência de possíveis avaliadores desonestos. Esse mecanismo pode ser utilizado para identificar falsas avaliações negativas e falsas avaliações positivas.

3.8 REPUTAÇÃO EM P2P

O funcionamento eficiente e correto de um sistema P2P depende da participação voluntária de seus usuários. Quando os *peers* de um sistema P2P não colaboram, as consequências são sérias e podem causar danos individuais a usuários

ou mesmo levar o sistema ao colapso. Sistemas tradicionais assumem que usuários são “obedientes”, que aderem a um protocolo especificado sem questionar a utilidade do mesmo para si. Esta obediência não é uma premissa realística em sistemas P2P (FELDMAN e CHUANG, 2005). É necessário, portanto, que o sistema controle de certa maneira os *peers*, responsabilizando-os pelas suas ações quando eles se recusam a colaborar, e recompensando-os quando colaboram adequadamente. Popularmente, *peers* que usam muito mais recursos do que oferecem (quando oferecem) são conhecidos como *free-riders* . Estudos em (HUBERMAN e ADAR, 2000) mostram que *free-riders* são comuns em diversas aplicações P2P de compartilhamento de arquivos, e a explicação para tal fenômeno está relacionada à “Tragédia dos Comuns” que argumenta que as pessoas tendem a abusar do uso de certos recursos se elas não têm que pagar por eles de alguma forma.

Os *free-riders* podem denegrir o desempenho do sistema para os pares que contribuem com recursos, obtendo recursos em detrimento destes. Considerando que há um custo para a doação de recursos para a comunidade, pode se tornar mais interessante para um par ser um *free-rider* do que colaborar para o sistema. Quanto menor a quantidade de pares colaborando, menor a quantidade de recursos disponíveis, e maiores as contenções de recursos

Existem diferentes abordagens para se incentivarem *peers* a colaborarem. (THEOTOKIS e SPINELLIS, 2004) identificam duas classes gerais de mecanismos de incentivo: baseados em confiança (*trust*) e reputação, e baseados em comércio(*trade*), que incluem mecanismos de micro-pagamento (em que um *peer* do overlay P2P que oferece um serviço para outro é explicitamente renumerado)e esquemas de troca de recursos. Uma terceira categoria é mencionada em (FELDMAN e CHUANG, 2005): generosidade inerente, na qual usuários decidem se contribuem para o sistema ou não baseados no nível global de contribuição dos demais usuários.

MojoNation (IBM, 2000) é um esquema baseado em comércio em que usuários que oferecem recursos como tempo de processamento e espaço em disco podem acumular unidades de *mojo*, e posteriormente, gastá-las. Outro é a “rede de favores” (ANDRADE, BRASILEIRO, *et al.*, 2004) proposta para o sistema de grade P2P *OurGrid*, no qual a decisão de acolher ou não uma tarefa de um *peers* remoto é tomada segundo o histórico anterior entre os peer envolvidos.

Os esquemas de confiança e reputação estão amparados na *reciprocidade* (FELDMAN e CHUANG, 2005)*:* cada *peer* possui uma reputação associada, que parte de um estado inicial e é construída ao longo da vida de um *peer*, com base em suas interações com outros *peers*. Essa informação de reputação pode influenciar a decisão

de um *peer* sobre os parceiros de sua próxima interação, buscando reciprocidade. Por exemplo, um *peer* pode preferir solicitar um serviço a um *peer* que lhe tem executado correta e eficientemente as últimas solicitações, ou deixar de considerar certos *peers* que têm retornado arquivos com conteúdo corrompido. A reciprocidade é obtida em um sistema P2P por meio de um sistema (de gerência) de reputação.

O grau de confiança em uma operação tem relação inversamente proporcional ao seu risco. Sobre a relação entre os termos, um sistema de gerência de reputação determina a reputação de *peers* baseado no histórico das ações dos mesmos e permite que sejam formadas opiniões sobre o grau de confiança de outros *peers*.

As relações de confiança podem ser de um para outro (confiar em um *peer* para execução de um serviço), de um para vários (confiar em um conjunto de *peers* com que informações podem ser seguramente trocadas), de vários para um (tal como confiar em um líder), e de vários para vários (quando um grupo confia em outro). Uma forma, expressa no Modelo de Opinião de Josang (JOSANG, ISMAIL e BOYD, 2006), é empregar uma tripla de valores, *c*, *d* e *i*, que correspondem à “crença”, “descrença” e “ignorância”, com $c + d + i = 1(1 >= c, d, i >= 0)$.

Um dos principais desafios na área de P2P é o projeto de um sistema de reputação que consiga determinar, precisa e eficientemente, valores de confiança adequados para peers de um sistema descentralizado de larga escala, e em que peers entram e saem, autonomamente, e a todo o momento do sistema, potencialmente sob identidades diferentes, podendo forjar mensagens e identidades.

3.9 VULNERABILIDADES DO SISTEMADE GERÊNCIA DE REPUTAÇÃO

Em termos de ação de resposta, um sistema de gerência de reputação fornece como serviço um valor de reputação sobre outros *peers*, que pode ser usado, então, em diversos contextos em um sistema P2P. Por exemplo, ao buscar um *peer* que execute, de maneira confiável um determinado serviço, ele pode-se usar a informação de reputação para inferir qual a probabilidade de o serviço ser executado corretamente. Outro exemplo é, ao buscar um objeto(como um arquivo) e encontrar múltiplos *peers* como candidatos a fonte, usar o valor de reputação (aliado ao desempenho) na escolha do *peer*.

Segundo (FELDMAN e CHUANG, 2005), existem duas questões principais que têm sido tratadas em diversos trabalhos na literatura:

- como peers novos, ditos estranhos, devem ser tratados em esquemas de reciprocidade (ou seja, em esquemas de reputação)?

- estratégias que são baseadas em reciprocidade indireta são vulneráveis a comportamento de conluio?

Essas questões são importantes em termos de vulnerabilidades do sistema de gerência de reputação. Existem diferentes formas de ataque a um sistema de reputação; os três principais ataques são discutidos a seguir.

3.9.1 Ataque de whitewashing

O primeiro ataque é conhecido como *whitewashing* e apenas ocorre quando peers podem trocar sua identidade facilmente (o que é o caso de muitos sistemas P2P). Um *peer* pode deixar o sistema e voltar, em seguida, com uma nova identidade, em uma tentativa de se livrar de qualquer reputação ruim que ele tenha acumulado. Se a política dos *peers* em relação a estranhos é permissiva, eles podem "usar" a reputação inicial, deixar o sistema e reingressar com nova reputação inicial. Se um *peer* não consegue distinguir um novo correto de um antigo, então *whitewhashers* podem causar o colapso do sistema se nenhuma contramedida for tomada ((FELDMAN e CHUANG, 2005).

3.9.2 Ataque de conspiração contra sistemas de reputação

O segundo tipo de ataque é o de conspiração contra sistemas de reputação. Este tipo de ataque é frequentemente efetivo, porque em sistemas de reputação típicos um *peer* deve consultar outros sobre a reputação de um terceiro. Se muitos estão comprometidos, então *peers* podem prover falso testemunho, no sentido de aumentar a reputação de um *peer* malicioso, ou de atacar um correto, reduzindo a sua reputação. Em princípio, o atacante deveria dispor de recursos em massa, fazendo com que boa parte dos *peers* do *overlay* fosse seus; entretanto, em muitos sistemas P2P não existe um esquema seguro de autenticação, possibilitando que *peers* adquiram múltiplas identidades falsas (criando *peers Sybil*) com um único *peer* físico, conforme explorado em (CHENG e FRIEDMAN, 2005).

3.9.3 Ataque de peer Traidor

O terceiro tipo de ataque é o de *peer traidor* (MARTI e GARCIA-MOLINA, 2006). Em tal ataque, um *peer* se comporta adequadamente por um tempo, de forma a construir uma boa reputação, e então explora o sistema valendo-se da mesma. Este

ataque é especialmente efetivo quando os *peers* ganham privilégios à medida que conquistam reputação. Um exemplo do ataque de traidor é quando um usuário no eBay constrói uma reputação, com muitas transações de pequeno valor, e então lesa alguém em uma transação de grande valor. Em termos sistêmicos, um *peer* traidor pode surgir não de uma mudança comportamental de um usuário, mas de uma mudança no ambiente. Por exemplo, uma máquina-cliente perfeitamente correta pode ser infectada com um vírus estilo Cavalo de Tróia, que então poderia, aleatoriamente, abusar da boa reputação do *peer*. A resistência a esse tipo de ataque pode ser aumentada usando a análise da história recente de um *peer* (FELDMAN e CHUANG, 2005).

3.10 CONCLUSÕES FINAIS

Por fim, conforme anteriormente comentado, a reputação de um *peer* é tipicamente usada na política de seleção de um *peer* com que interagir. Em (DUMITRIU, KNIGHTLY, *et al.*, 2005) é caracterizado o impacto de políticas de seleção entre respostas adotadas por *peers* P2P que originam buscas. São identificadas diversas políticas, dentre as quais se incluem-me escolher o *peer* que anuncia a melhor capacidade, ou seja, com o menor atraso, que é calculado por cada um deles em função da capacidade de *upload* e o número atual de *uploads* simultâneos; aleatório, em que o cliente seleciona um *peer* aleatoriamente; e *file chunking*: o cliente divide o arquivo em múltiplos blocos, e faz *donwload* simultâneo de um pedaço de cada *peer* que anunciou o arquivo.

Sob o ponto de vista de ataques, a pior política de escolha de um *peer* é aquela que seleciona o *peer* que se anuncia como o melhor. Por exemplo, se uma informação de desempenho é facilmente falsificável, uma busca só terá sucesso quando nenhuma resposta for de um *peer* malicioso, pois um *peer* malicioso sempre será escolhido. Segundo (DUMITRIU, KNIGHTLY, *et al.*, 2005), sistemas de reputação não conseguem resolver esse problema, mesmo quando os erros do sistema de reputação são mínimos. Técnicas baseadas em aleatoriedade são efetivas para aumentar a resistência de um sistema P2P a ataques. Entretanto, aleatoriedade impacta negativamente no desempenho quando atacantes não estão presentes.

4 TEORIA ECONÔMICA

4.1 ECONOMIA DE MERCADO

Pessoas necessitam alimentar-se, vestir-se, receber educação, entre outras necessidades do ser humano. Para tanto, há os recursos financeiros, mas nem sempre a renda é suficiente na hora de conseguir todos os bens e serviços desejados para satisfazer suas necessidades.

A sociedade (conjunto de pessoas) tem também necessidades coletivas, tais como estradas, defesa, justiça etc. O mesmo ocorre, individualmente, com as pessoas, que também têm mais necessidades do que meios para satisfazê-las. A economia se ocupa das questões relativas à satisfação das necessidades dos indivíduos e da sociedade.

A satisfação de necessidades *materiais* (alimentos, roupas ou habitação) e *não-materiais* (educação, lazer, etc.), de uma sociedade obriga seus membros a se ocuparem de determinadas atividades produtivas. Por intermédio dessas atividades, produzem os bens e serviços de que necessitam, e que, posteriormente, se distribuem para seu consumo entre os membros da sociedade.

Nesse processo de produção e consumo, surgem e são solucionados muitos problemas de caráter econômico: problemas nos quais se utilizam diversos meios para se conseguir uma série de fins ou objetivos (VASCONCELOS, 2006).

Na produção, por exemplo, a empresa tem que decidir que bens irá produzir e que meios utilizará para produzi-los. No caso de uma empresa que produz automóvel, os gerentes têm de decidir o modelo de automóvel a ser lançado no mercado e se irão produzi-lo com uma tecnologia robotizada ou com uma em que se emprega mais mão-de-obra.

Em relação ao **consumo**, as famílias têm que decidir como vão gastar a renda familiar entre os diferentes bens e serviços ofertados para satisfazer suas necessidades. Assim, uma família qualquer, na hora de decidir entre um televisor e uma máquina de lavar, levará em conta suas necessidades, os preços de ambos os bens e suas próprias preferências, de forma que o resultado da escolha seja o mais apropriado.

4.1.1 Definição de Economia

Economia é a ciência social que estuda como o individuo e a sociedade decidem (escolhem) empregar recursos produtivos escassos na produção de bens e serviços, de modo a distribuí-los entre várias pessoas e grupos da sociedade, a fim de satisfazer as necessidades humanas. Conceitos que são definidos como a base de seu estudo são: escolha, escassez, necessidades, recursos, produção e distribuição (VASCONCELOS, 2006). Em todas as sociedades, os recursos ou fatores de produção são escassos, embora, as necessidades humanas sejam ilimitadas. Dessa forma, a sociedade é obrigada a escolher entre diversas maneiras de produção e distribuição dos resultados da atividade produtiva aos vários grupos da sociedade.

A economia pode ser definida, também, com sendo o estudo de como as pessoas e a sociedade decidem empregar recursos escassos, que poderiam ter utilizações alternativas, para produzir bens variados.

Sistema econômico é a forma política, social e econômica pela qual uma sociedade está organizada. Seus elementos básicos são:

- Fatores de produção: são os recursos humanos, o capital, a terra, as reservas naturais e a tecnologia;
- Unidades de produção: são as empresas;
- Instituições políticas, jurídicas e sociais: são a base de organização da sociedade.

4.1.2 A Micro a e Macroeconomia

A microeconomia ocupa-se da análise do comportamento das unidades econômicas, como as famílias, ou consumidores, e as empresas. Estuda, também, os mercados em que operam os demandantes e a oferta de bens e serviços. A perspectiva microeconômica considera a atuação das diferentes unidades econômicas como se fossem unidades individuais.

A microeconomia é a parte da teoria econômica que estuda o comportamento das unidades, tais como os consumidores, as indústrias e empresas, e suas inter-relações. A macroeconomia, pelo contrário, ocupa-se do comportamento global do sistema econômico refletido em um número reduzido de variáveis, como o produto total de uma economia, o emprego, o investimento, o consumo, o nível geral de preço etc.

A macroeconomia estuda o funcionamento da economia em seu conjunto. Seu propósito é obter uma visão simplificada da economia que, porém, ao mesmo tempo, permita conhecer e atuar sobre o nível da atividade econômica de um determinado país ou de um conjunto de países.

A microeconomia e a macroeconomia variam conforme a contextualidade em que estão envolvidas e, é importante ressaltar, na economia moderna, “tanto o poder governamental como o mecanismo de preços e de mercado desempenham papéis relevantes, não obstante haja intensidade variável nos diferentes países” (RIBEIRO, 2008).

A fronteira entre a microeconomia e a macroeconomia é uma espécie de zona cinzenta, em que o limite é ínfimo, quase imperceptível. Alguns critérios são adotados para essa distinção, como o de afirmar que a macroeconomia abarca o funcionamento da economia em sua totalidade e a microeconomia em aspectos especiais. Outro critério é a forma de comportamento das variáveis individuais e das variáveis agregadas. A microeconomia volta-se aos consumidores individuais e às firmas, isoladamente considerados, enquanto a macroeconomia abrange os grandes vultos econômicos.

A microeconomia pode ser analisada sob o viés da normatividade e da positividade. Normalmente é enquadrada no campo da economia positiva. É positiva porque ela descreve como é e não como deve ser. Ao contrário da normativa que diz como deve ser, sempre implicando juízos de valor, éticos, etc. para a economia positiva, o economista deve ter como objetivo “ determinar a alocação efetiva de recursos produtivos”. Apesar de ser uma economia positiva, ela não deixa de ter um lado normativo, em que são enunciados valores, principalmente quando os economistas deixam de ser profissionais e agem como juízes que determinam e avaliam valores.

De qualquer forma, deve-se ressaltar que a microeconomia e a macroeconomia são dois ramos da mesma disciplina, a economia, e como tais se ocupam das mesmas questões, ainda que se fixem em aspectos distintos.

4.1.3 Lei da Oferta e da Procura

Em economia, a Lei da Oferta e Procura, também chamada de Lei da Oferta e da Demanda, é a lei que estabelece a relação entre a demanda de um produto - isto é, a procura- e a quantidade que é oferecida, a oferta. A partir dela é possível descrever o comportamento preponderante dos consumidores na aquisição de bens e serviços

em determinados períodos, em função de quantidades e preços. Nos períodos em que a oferta de um determinado produto excede muito à procura, seu preço tende a cair. Já em períodos nos quais a demanda passa a superar a oferta, a tendência é o aumento do preço.

Costuma-se definir a procura, ou demanda individual, como a quantidade de um determinado bem ou serviço que o consumidor estaria disposto a consumir em determinado período de tempo. É importante notar, nesse ponto, que a demanda é um desejo de consumir, e não sua realização. Demanda é, assim, o desejo de comprar (Vasconcelos, 2006).

A Teoria da Demanda é derivada da hipótese sobre a escolha do consumidor entre diversos bens que seu orçamento permite adquirir. Essa procura individual seria determinada pelo preço do bem e o preço de outros bens a renda do consumidor e seu gosto ou preferência.

A demanda é uma relação que demonstra a quantidade de um bem ou serviço que os compradores estariam dispostos a adquirir a diferentes preços de mercado. Assim, a Função Procura representa a relação entre o preço de um bem e a quantidade procurada, mantendo-se todos os outros fatores constantes.

Quase todas as mercadorias obedecem à lei da procura decrescente, segundo a qual a quantidade procurada diminui quando o preço aumenta. Isto se deve ao fato de os indivíduos estarem, geralmente, mais dispostos a comprar quando os preços estão mais baixos.

Correspondentemente, costuma-se definir oferta como sendo a quantidade de determinado bem ou serviço que os produtores e vendedores desejam vender em determinado período (VASCONCELOS, 2006). Como na demanda, a oferta representa um plano ou intenção, neste caso dos produtores e vendedores, e não a venda efetiva.

A estabilização da relação entre a oferta e a procura leva, em primeira análise, a uma estabilização do preço. Uma possível concorrência, por exemplo, pode desequilibrar essas relações, provocando alterações de preço.

Ao contrário do que pode parecer a princípio, o comportamento da sociedade não é influenciado apenas pelos preços. O valor de um produto pode ser um estímulo positivo ou negativo para que os consumidores adquiram os serviços de que necessitam, mas não é o único.

Da mesma forma que a oferta exerce uma influência sobre a procura dos consumidores, a frequência com que as pessoas buscam determinados produtos também pode aumentar e diminuir os preços dos bens e serviços.

A curva de procura baseia-se na utilidade de determinado produto para os consumidores. Quanto maior o preço, menor a quantidade procurada, e vice-versa. São determinantes da procura: preço do produto, rendimento médio dos consumidores, dimensão do mercado, preço e disponibilidade de outros bens, gostos ou preferências. O deslocamento da curva de procura ocorre em função da alteração desses fatores.

Já a curva de oferta baseia-se nos custos de produção de um bem ou serviço. É a relação entre os preços de mercado do produto e a quantidade que os produtores estão dispostos a oferecer. Quanto menor o preço, menor a quantidade de bens que os produtores vão querer vender. São determinantes da oferta: custos de produção, monopólios, concorrências de outros bens, imprevistos meteorológicos. O deslocamento da curva de oferta ocorre em função da alteração desses fatores.

4.1.4 Permuta

Troca, permuta ou mesmo escambo são sinônimos que não remetem apenas à forma de comércio de mercadorias praticado na Antiguidade. Há muitos anos, quando não existia moeda, as negociações por meio de trocas eram frequentes. As pessoas davam o que tinham em excesso e recebiam o que lhes faltava. Esse modelo nunca saiu de moda na economia.

Na permuta, uma das partes promete um bem em troca de outro, ou seja, uma se obriga a dar um bem por outro. Em melhor definição a troca é o contrato pelo qual as partes se obrigam a dar uma coisa por outra que não seja dinheiro (GONÇALVES, 1999) (DAIUTO, 1995). Historicamente, o contrato de compra e venda é uma evolução da troca ou permuta, ou ainda, denominado de escambo, antes do surgimento da economia monetária.

Importante ressaltar a diferença da troca e permuta para a compra e venda, quais sejam, a exclusão da figura do dinheiro, do preço, da contraprestação em moeda do processo de troca ou permuta. Assim, estando presente a contraprestação em dinheiro a negociação se desfigura, tornando-se um contrato de compra e venda. Não é necessário que haja valores iguais entre os bens permutadas.

A troca, permuta ou escambo representa um negócio jurídico com as seguintes características:

- Bilateral: porque existe obrigação para as duas partes. Obrigação entre ambos de transferir, um para o outro, a propriedade de determinado bem.

- Oneroso: porque ocorre redução patrimonial para as duas partes
- Comutativo: podem-se prever as vantagens e desvantagens do negócio, que podem advir, devido ao fato de suas prestações serem certas
- Consensual: basta o consentimento para ser celebrado. É consensual e não real, pois se aperfeiçoa com o acordo de vontades, independente da tradição.

Como regra, quaisquer bens podem ser objetos de troca, a exemplo de móveis por móveis, imóveis por móveis, semovente (animal) por coisa, coisa por direito, direito por direito. Assim, qualquer coisa que possa ser alienada ou vendida pode ser trocada.

A troca pode ser realizada entre bens de valores diferentes, mesmo que haja a complementação em dinheiro por parte de uma das pessoas envolvidas na troca.

Nos dias de hoje, essa prática não só ganha novas formas, como também novos nomes: as empresas chamam de permuta. E não se troca apenas por necessidade, mas principalmente, por desejo. Muitas empresas no mundo estão se organizando e fazendo parte de redes de permuta, criando ambientes de negócios gerenciados, principalmente, por meio da Internet. Vale trocar tudo, desde produtos até serviços, e, geralmente, não é utilizado nenhum tipo de dinheiro real nas negociações.

Calcada em uma necessidade no passado, a moeda de troca é utilizada com objetivos mais específicos, como ode sobrevivência. Essa é uma maneira de deixar a inadimplência ou resolver dificuldades financeiras temporárias, mas é possível, por exemplo, reduzir custos fixos, escoar estoques ociosos, reservar caixa para despesas emergenciais e abrir novos mercados.

As permutas nada têm de inusitadas. Elas são, sim, uma forma criativa e altamente lucrativa para lidar com ociosidade. Porém, uma nova forma vem tomando lugar, isto é, o escambo bilateral transforma-se em conceito multilateral**.** Esse conceito se adapta no modelo para compartilhamento de recurso e serviços que podem ser negociados em uma rede P2P. Cada cliente opta pela quantidade, tempo de uso e pela qualidade do recurso ou serviço que “adquire”, determinando de quem o adquire, podendo ser de uma ou várias fontes ao mesmo tempo.

5 A ECONOMIA NA REDE

5.1 INTRODUÇÃO

Em uma sociedade alicerçada na informação, o valor está na abundância e não na escassez (KELLY, 2003). A abundância aumenta as chances de associação entre informação, gerando riqueza. Ao contrário do que acontece com o ouro ou pedras preciosas, nesta nova economia o valor não está tanto no material, mas na informação que se tem do material e do modo como é utilizada.

Em (KELLY, 2003), a atual revolução econômica que estamos vivenciando está criando uma economia totalmente nova, sem precedentes: “uma nova ordem econômica”.

> “A grande ironia de nosso tempo é que a era dos computadores acabou. *Peers* já usufruímos de todo o valor que um computador isolado pode nos trazer. Computadores nos fazem mais rápidos e eficientes. Nada mais. Por outro lado, todas as novas tecnologias que estamos experimentando são fundamentadas na comunicação entre computadores. Isto é, conectar tornou-se mais importante que computar”, explica Kelly.

Face à Economia da Rede, como ele prefere chamar a “nova economia”, Kevin Kelly postula algumas leis que, segundo ele, devem seguir de guia para quem deseja se aventurar nessa revolução econômica.

A primeira lei, a “Lei da Conexão”, prega que um computador isolado é apenas um cérebro dentro de uma caixa de plástico. Apenas quando são interligados milhares desses cérebros é que se cria grande inteligência e, em seguida, algum valor econômico. “Um trilhão de chips conectados em rede é o *hardware*. O *software* que opera essa máquina é a Economia da Rede”, postula Kelly.

Em uma segunda lei, Kelly afirma que “quanto mais *peers* fizerem parte da rede, maior o valor econômico. Cada novo componente (*peer*) da rede aumenta seu valor unitário e mais ainda o valor da própria rede. O poder econômico vem da abundância e não da escassez (como acontecia com o petróleo e o ouro na revolução industrial).”

Nesse sentido, esta tese apresenta um estudo sobre o uso do modelo econômico de troca em rede *Peer-to-Peer* para o compartilhamento de recursos computacionais.

5.2 MODELOS ECONÔMICOS PARA SISTEMAS DISTRIBUÍDOS

A ideia de aplicação de modelos econômicos em sistemas computacionais não é nova. Diversas soluções, baseadas principalmente na alocação de recursos, utilizam modelos econômicos para o controle de uma infinidade dos mesmos como, por exemplo, alocação de memória, controle de largura de banda, entre outros. Deseja-se permitir a obtenção de acesso e utilização de forma dinâmica desses recursos, por parte de todos que têm acesso ao sistema P2P, sem a necessidade de acordos pré-definidos entre as partes, a fim de viabilizar a formação de sistemas de grande capacidade de processamento formados por recursos amplamente distribuídos e heterogêneos.

O problema básico de fornecimento do acesso aos recursos em um sistema P2P pode ser definido pela quantidade de recursos disponível em um dado momento da computação. Este pode ser visto como um problema de oferta e demanda para o qual existem diversas teorias e soluções na área de Economia.

Embora estudos teóricos dos modelos econômicos aplicados aos sistemas de informação somente agora comecem a surgir, várias empresas, incluindo a IBM (IBM GRID computing), HP (HP Grid Computing) bem como iniciativas como a (IBM, 2000) utilizam modelos econômicos para alocação de recursos dentro de seus sistemas. Basicamente, os modelos econômicos estão sendo utilizados de duas formas. A primeira se destina a utilizar princípios econômicos para atribuir um valor financeiro aos serviços. A segunda está focada na utilização desses conceitos para aprimorar as funcionalidades dos sistemas distribuídos, principalmente, no que tange ao compartilhamento de recursos e ao escalonamento e balanceamento de carga.

A disponibilização de um serviço como um serviço utilitário (BUNKER e THOMSON, 2006) (RAPPA, 2004), já se apresenta em forma comercial, como é o caso do Sun *Grid ComputeUtility* (SUN MICROSYSTEMS, 2007), um sistema que disponibiliza poder de processamento para a execução de certos serviços pelo custo de $1 (um dólar) por uso de um processador a cada hora. Esse tipo de serviço tenta se aproximar do sistema que deu origem ao nome de *Grid* computing, que seria o sistema de Distribuição de Energia Elétrica.

O sistema econômico também favorece o aprimoramento dos sistemas de compartilhamento de recursos, balanceamento e escalonamento de tarefas. Tendo em vista que os sistemas econômicos por natureza são independentes e distribuídos (WOLSKI, PLANK e BREVIK, 2000) a utilização desses conceitos possibilita aos sistemas computacionais uma autonomia na execução dos serviços prestados, principalmente em P2P*,* possibilitando que a estrutura não necessite de sistemas centrais, passíveis de falhas e, por conseguinte, de paralisação do sistema. As dificuldades apresentadas ainda são agravadas pelas diversas configurações e tipo de arquiteturas que podem ser encontradas nos mais diferentes tipos de dispositivos que fazem parte de uma infraestrutura da rede P2P.

Apesar dessa visão financeira de um serviço prestado, a utilização de conceitos econômicos não é focada apenas nesse propósito. A princípio duas direções estão sendo tomadas no que diz respeito aos modelos econômicos aplicados a sistemas distribuídos:

- Comercialização de serviços/recurso (RAPPA, 2004): segue a ideia de Utilidade *(Utility),* que considera um serviço como uma atividade econômica e quantificada por um valor comercial. Há uma preocupação maior na qualidade do serviço prestado (APPLEBY, FAKHOURI, *et al*., 2001) e menor no tempo de negociação do preço. A infraestrutura necessária para esse tipo de serviço se torna consideravelmente maior, já que além dos componentes básicos de um sistema distribuído, há a necessidade de se desenvolverem novas partes responsáveis por atividades como controle de pagamentos, os leilões ou até mesmo agentes reguladores.
- Alocação de Recursos (KUROSE e SIMHA, 1989) (SUBRAMONIAM, MAHESWARAN e TOULOUSE, 2002) (XUE, LI e NAHRSTEDT, 2003): trata-se de aplicações pontuais; nessa modalidade computacional, os conceitos de economia são empregados para facilitar o escalonamento de tarefas entre dispositivos heterogêneos dispersos, como por exemplo, os recursos de uma rede P2P. Apresentam problemas mais restritos e há preocupação com o desempenho geral do sistema e, não raramente, são feitas menções aos sistemas com limitações temporais

Sobre o compartilhamento de recurso se o escalonamento de tarefas, a utilização de modelos econômicos provê diversas contribuições, (FERGUSON,

NIKOLAOU, *et al.*, 1996) compara o modelo econômico com os sistemas computacionais distribuídos. Um conjunto complexo de mecanismos é disposto ao modelo econômico com a finalidade de assegurar o controle descentralizado dos recursos, que não proveem mecanismos robustos e distribuído para a alocação de recursos.

O item escalonamento de recurso sempre foi um dilema dentro de sistemas distribuídos, inclusive atualmente (WOLSKI, PLANK e BREVIK, 2001) não existem políticas ou mecanismos para alocação e controle de recursos que possam assegurar alto desempenho e estabilidade nas configurações de um *Grid* computacional e de uma rede P2P. Os Métodos de escalonamento distribuído são difíceis de manter, gerenciar e incertos em garantir o desempenho à medida que *peers* são retirados ou incluídos (WOLSKI, PLANK e BREVIK, 2000).

Além desse fator, atualmente são disponíveis os mais diferentes tipos de computadores, diferenciando arquitetura, sistema operacional, capacidade dos recursos e vários outros fatores que tornam o sistema um ambiente heterogêneo e o projetista deve estar ciente de que seu programa precisa estar apto a executar nessas várias configurações de *hardware* e *software*.

Com a constatação desses problemas apresentados, a utilização de conceitos econômicos é vislumbrada como uma solução viável para ser utilizada em sistemas distribuídos, inicialmente pelo fato de que os modelos econômicos já consolidados apresentam características semelhantes aos desejados num ambiente de computação distribuída, e ainda, porque é comum encontrar no mercado econômico produtos e bens dos mais variados tipos e características sendo quantificados por um valor comum, no caso a moeda.

A utilização de modelos econômicos pode prover ainda (KENYON e CHELIOTIS, 2003):

- Flexibilidade: recursos podem ser obtidos quando o usuário necessitar e podem expressar sua necessidade com alto grau de detalhamento dos recursos necessários.
- Eficiência: o preço dos recursos reflete fatores como oferta e demanda, os quais podem representar a carga de um recurso.
- Escalabilidade: novas entidades consumidoras e fornecedoras de recursos podem ser inseridas e retiradas, mantendo a flexibilidade e eficiência do sistema.
- *Feedback*: o preço para uso de recursos e o valor dos recursos pode ser utilizado para guiar decisões de gerenciamento.

Outra vantagem que a utilização de modelos econômicos traz, é a transparência. Um usuário não precisa saber, exatamente, como um serviço irá ser disponibilizado para ele, apenas será informado do preço para a execução, e isso vale para todos os serviços que a rede P2Ppretende disponibilizar.

Essa similaridade tem impulsionado a utilização de princípios econômicos em pesquisas (FERGUSON, NIKOLAOU, *et al.*, 1996) (LAI, RASMUSSON, *et al.*, 2005) (REDDY, 2006) para prover serviços computacionais para ambientes de produção onde seriam comercializados os recursos da rede P2P. A introdução de conceitos econômicos dentro de atividades computacionais, é vista com naturalidade, já que um simples parque de computadores interligados pode ser compreendido como um sistema econômico em miniatura, onde as forças de oferta e demanda podem ser observadas.

Cada uma dessas forças é representada por dois agentes: Consumidores (usuários dos recursos) e Fornecedores (proprietários dos recursos), e cada um com seus objetos diferentes. Os Consumidores procuram otimizar seus critérios de desempenho, pela obtenção dos recursos necessários sem se preocupar com o desempenho de todo o sistema. Enquanto o Fornecedor tem o comportamento baseado em funções de produção e posição relativa no mercado.

Porém, como já pontuado anteriormente, agregar conceitos econômicos ao sistema distribuído é custoso, tanto devido à implementação de novos componentes ao sistema, como à demanda computacional extra, necessária para atender às novas funcionalidades da arquitetura. Para facilitar, Kenyon e Cheliotis (KENYON e CHELIOTIS, 2003) propõem um modelo composto de três pilhas; em cada pilha há vários níveis, cada um responsável por uma atividade dentro da arquitetura.

Analisando esse modelo, estaremos focando nossa proposta na pilha de comercialização, mais precisamente na formação de preço.

Ainda a respeito do modelo apresentado por (KENYON e CHELIOTIS, 2003), verificamos que ele se refere apenas à infraestrutura do *peer*, não abrangendo a interação entre os diversos *peers* que fazem parte de uma arquitetura. Para isso, Yeo e Buyya (YEO e BUYYA, 2006) apresentam uma taxonomia sobre a arquitetura, dividida nos seguintes tópicos:

- Modelo de mercado
- Modelo de recursos
- Modelo de tarefas
- Modelo de alocação de recursos
- Modelo de avaliação

Porém, para a elaboração desse trabalho estaremos focando apenas os dois primeiros itens que são: modelo de Mercado e de Recursos.

5.2.1 Taxonomia Modelos de Mercado

A Taxonomia Modelo de Mercado examina como a atual concepção econômica é incorporada dentro dos sistemas de computação distribuída. Segundo Piro (PIRO, 2003), um modelo econômico não é determinado apenas por um conjunto de recursos e por um conjunto de agentes (consumidores e produtores), mas também por um conjunto de regras que especificam a interação entre os recursos e os agentes. Esse modelo é composto por quatro sub-taxonomias (YEO e BUYYA, 2006):

1. Modelos Econômicos
2. Focado no Participante
3. O Ambiente Comercial
4. Atributos de Qualidade de Serviços

Porém para a elaboração desse trabalho focaremos apenas o primeiro item: Modelo Econômico.

A. Modelo Econômico

A partir dos princípios de microeconomia e macroeconomia, vários modelos econômicos foram propostos para o campo da computação, baseados no modelo derivado de Buyya (BUYYA, 2002) (YEO e BUYYA, 2006), que apresenta uma taxonomia dos modelos econômicos utilizados no ambiente computacional, conforme apresentado na Figura 8. Os conceitos e estratégias econômicas embutidos em sistemas existentes ou propostas estão também apresentados em (RAJKUMAR, HEINZ, *et al.*, 2001). Descrevem-se então:

Commodity markets os produtores anunciam os preços dos recursos e cobram os utilizadores baseando-se na quantidade de recursos utilizados.

Posted price – operação similar ao *Commodity markets.* Contudo, são efetuadas, abertamente, ofertas especiais para que os consumidores estejam cientes dos descontos e possam usufruir os mesmos.

*Auction(*Leilão)suporta negociação um-para-muitos, entre um prestador de serviços (vendedor), e muitos consumidores (compradores). A negociação se reduz a um único valor . O leiloeiro estabelece as regras do leilão, aceitáveis para os

consumidores e os fornecedores. Leilões basicamente utilizam as forças do mercado para negociar um acordo de compensação dos preços para o serviço.

Bid-based proportional resource sharing (*lance com base na partilha proporcional dos recursos)* - o percentual dos recursos alocados para o usuário aplicação é proporcional ao valor do lance em comparação a lances de outros usuários. Aos usuários são atribuídos créditos ou fichas, que podem utilizar para ter acesso aos recursos. O valor de cada crédito depende da demanda.

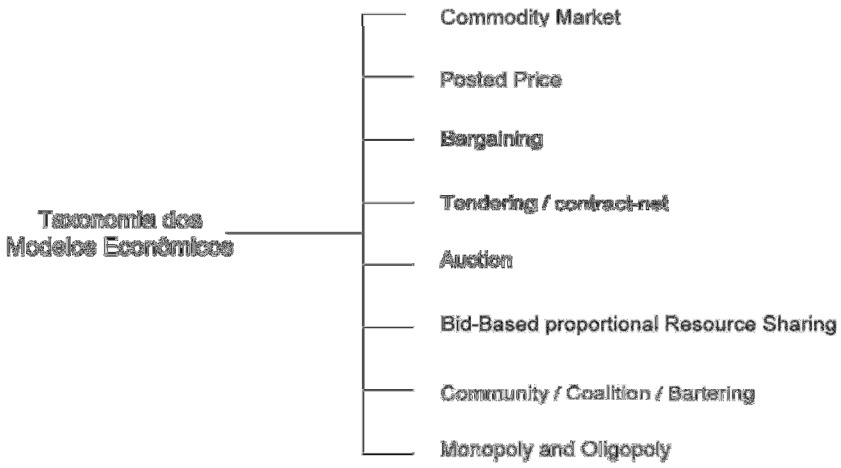

Figura 8 - Taxonomia dos Modelos Econômicos (YEO e BUYYA, 2006)

No Tendering/contract-net o consumidor primeiro anuncia as suas necessidades com intuito de atrair lances de potenciais produtores. Os produtores, em seguida, avaliam o requisito afim de determinar se os mesmos têm ou não interesse em participar da operação. Esses podem ignorar o anúncio, caso o serviço não seja do seu interesse ou caso o mesmo esteja ocupado

Bargaining (Negociação) exige negociações diretas entre produtores e consumidores até se chegar a um preço acordado mutuamente. Normalmente, os produtores iniciam a operação com os valores mais altos para maximizar seus lucros, enquanto os consumidores iniciam como os valores mais baixos para minimizar os seus custos.

Community/coalition/bartering agrupa consumidores/produtores que compartilham seus recursos criando um ambiente cooperativo compartilhado. Este modelo é normalmente adotado em ambiente computacional onde os consumidores são também produtores e assim tanto contribui em quanto utilizam os recursos. São necessários mecanismos de regulação para que os participantes atuem de forma equitativa em ambos os papéis de produtores e consumidores

Monopoly/Oligopoly não representa um mercado competitivo, pois somente um (*Monopoly*) ou um pequeno número de produtores (*Olygopoly*) determina o preço do mercado. Nesse modelo, o Consumidor não é capaz de negociar e nem consegue afetar o preço dos produtos.

A maioria dos sistemas está baseada no modelo de mercado e adota um único tipo de modelo. Entretanto, também é possível utilizar modelos híbridos ou modificados, variantes dos modelos econômicos, a fim de aproveitar os pontos fortes dos diferentes modelos e proporcionar uma melhor personalização do usuário, com base em critérios específicos de cada aplicação. Como exemplo, temos *Stanford Peers* (COOPER e GARCIA-MOLINA, 2002) como uma iniciativa que adota um modelo híbrido de leilão e troca.

5.2.2 Taxonomia dos Modelos de Recursos

O modelo de recurso descreve as características de arquitetura de um sistema distribuído. Nesse modelo, os sistemas distribuídos são classificados conforme o controle de gerenciamento, a composição dos recursos, o suporte, a escalonamento e o mecanismo de contabilidade, conforme mostrado na Figura 9.

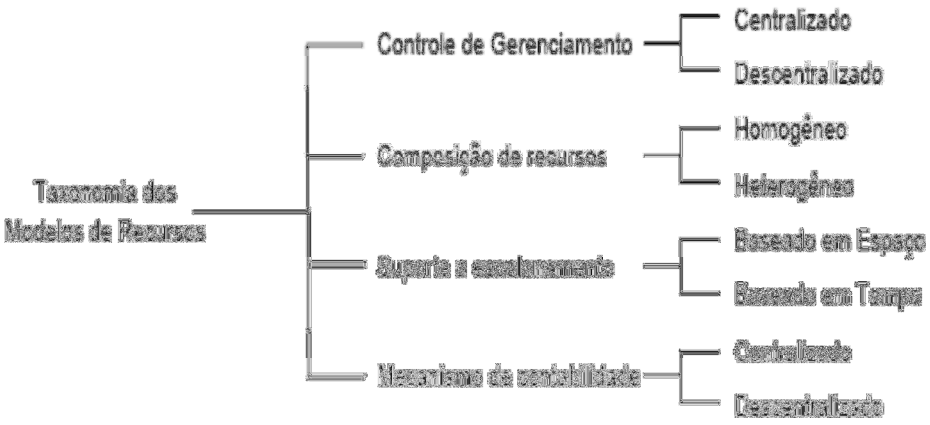

Figura 9- Taxonomia dos Modelos de Recursos. Fonte (YEO e BUYYA, 2006)

A. Controle de Gerenciamento

O controle de gerenciamento descreve como um recurso é controlado e gerenciado em um sistema distribuído (YEO e BUYYA, 2006). Um sistema com o

controle de gerenciamento centralizado apresenta um ponto central que administra os recursos e as tarefas. Enquanto isso, um sistema com controle de gerenciamento descentralizado tem controle capaz de gerenciar um subconjunto de recursos em um sistema distribuído.

Cada um apresenta suas vantagens e desvantagens, como é de se esperar. O controle de gerenciamento centralizado apresenta maior facilidade de implementação, porém é mais suscetível a gargalos e falhas devido à sobrecarga e mau funcionamento, em comparação ao controle de gerenciamento descentralizado, que apresenta um modelo com menores chances de paralisação geral do sistema, por possuir administradores que podem atuar para compensar falhas individuais no sistema.

Como ponto negativo do controle de gerenciamento descentralizado está a dificuldade de implementação, consideravelmente maior que num controle centralizado e, ainda, atenção especial deve ser desprendida quanto ao sincronismo entre os controles, o que exige protocolos e serviços de sincronização e demanda uma maior quantidade de dados trafegando via rede.

Segundo (YEO e BUYYA, 2006), o mais apropriado modelo para sistemas distribuídos, baseado em modelos de mercado, é o modelo descentralizado, por garantir melhor escalabilidade e confiabilidade.

B. Composição de recursos

(YEO e BUYYA, 2006) definem a composição de recursos como a combinação dos recursos que são disponibilizados em um sistema distribuído. No caso de um sistema distribuído, em que todos os *peers* possuem a mesma configuração e tipos de recursos, esse sistema é considerado homogêneo. Esse tipo de sistema é direcionado para facilitar processamento paralelo que requer o mesmo tipo de recurso a ser processado. A vantagem de se ter uma composição homogênea de recursos é a simplicidade do código e a velocidade de execução em comparação ao sistema com recursos heterogêneos.

Em compensação, o sistema com composição de recursos heterogêneos, que possui diferentes composições de recursos e configurações, tem a vantagem de poder disponibilizar serviços que podem ser executados de forma concorrente.

A utilização de recursos heterogêneos em um sistema econômico apresenta algumas dificuldades técnicas, principalmente no que se refere à definição de um preço comum para todos os recursos. Vários fatores influenciam em cada um dos

recursos e, conforme a utilização de cada um deles, a curva de variação de preços pode ser extremamente diferente. Esses recursos apresentam diferentes funções de utilidade e, consequentemente, estimar uma função de custo único para todos os recursos tem sido um desafio para os pesquisadores que estão ligados à área.

C. Suporte a escalonamento

O suporte a escalonamento determina o tipo de processamento que é suportado pelo sistema do Cluster (YEO e BUYYA, 2006). O escalonamento baseado em espaço permite que apenas uma simples tarefa seja executada em um período do processador, enquanto o escalonamento baseado em tempo suporta que múltiplas tarefas sejam executadas em um período do processador.

O escalonamento baseado em espaço, com uma tarefa a ser designada para executar em um *peer*, irá utilizar todo o processamento desse *peer* até que a tarefa tenha sido concluída. Com isso, o tempo em que a tarefa demorará para ser executada irá ser menor, já que não é necessário chaveamento das tarefas. As tarefas que chegarem posteriormente nesse *peer*, deverão aguardar até que a tarefa que está executando libere o uso do processador, assumindo que nessa abordagem, a preempção não é suportada.

Em contrapartida, o escalonamento baseado em tempo, permite que tarefas sejam chaveadas e processadas, concorrentemente, no processador. A vantagem de se utilizar um sistema de escalonamento baseado em tempo é que uma tarefa pode ser alocada para utilizar o processador enquanto outra está esperando pela resposta de um dispositivo de entrada/saída. Com isso, em um período de tempo maior tem-se uma maior vazão de tarefas sendo executadas.

D. Mecanismo de Contabilidade

Os mecanismos de contabilidade servem para manter e armazenar informações sobre a execução de tarefas em um sistema distribuído. Essas informações podem ser utilizadas para decisões de mudança de planos de alocação no futuro (YEO e BUYYA, 2006).

Quanto à classificação, um mecanismo de contabilidade pode ser centralizado, quando um único *peer* é responsável por manter os dados concentrados e armazenados. E um mecanismo descentralizado provê múltiplos *peers* monitorando e armazenando um conjunto de informações.

Ter um mecanismo de contabilidade centralizado garante que a recuperação dos dados seja mais simples, contudo é menos confiável e expansível se comparado a um mecanismo de contabilidade descentralizado.

5.2.3 Definição de Recursos

A base de toda a negociação se dá em nível de recurso. A identificação dos recursos necessários, pelo cliente, e a definição de custo de cada fornecedor é o que forma o mercado de negociação que será estabelecido entre ambos. Para consolidar essa definição Narayanan (NARAYANAN, 2002) estabelece associações de um recurso *r,* com:

- Demanda instantânea *dr(t)*, é a especificação de recurso que um usuário utilizará no espaço de tempo $t+\delta t$.
- Fornecimento instantâneo *sr(t)*, é a capacidade de um fornecedor para prover recursos nesse mesmo período de tempo $t+\delta t$.
- Função de custo *cr(t)*, o conjunto de recursos utilizados/providos acarreta em uma função de custo Esse custo pode ser características temporais, como tempo para executar uma tarefa, características funcionais, como consumo de bateria, ou ainda com uma unidade de permuta universal, que seria a estipulação de uma moeda para a transação.

Aplicando esse conceito a um sistema de economia Grid, temos duas entidades que já foram discutidas anteriormente, porém agora contextualizadas na definição de recursos:

- Fornecedores: são aqueles que são capazes de disponibilizar um conjunto de recursos; cada recurso disponibilizado apresenta uma demanda limitada e é delimitada por um conjunto de características inerentes a cada recurso.
- Consumidores: são os interessados pela disponibilidade de um serviço. Devem ser capazes de expressar suas necessidades e exigências, em troca da execução do serviço; recebem um custo, que pode ser financeiro, temporal ou baseado em algumas características computacionais, como consumo de energia (bateria no caso de dispositivos móveis).

Tanto fornecedores quanto consumidores são entidades independentes e a definição de qual papel terá um dispositivo computacional, não exatamente precisa ser

pré-definida no tempo de projeto. Por serem entidades dinâmicas, os papéis ainda podem se trocar durante a execução, porque um *peer* que forneça certo serviço, pode requisitar que a execução de algo de que ele necessite seja em outro *peer*.

A relação de recursos utilizados pode ser uma função desempenhada pelos consumidores, que precisariam fazer um levantamento de suas necessidades baseados em características da tarefa que desejam executar. Ou também pode ser considerada uma tarefa do fornecedor, em que é mantido um histórico das execuções e, baseada nesses valores, é realizada uma estimativa de quanto será necessário que seja utilizado de um determinado recurso para a execução da tarefa.

Cada tarefa, portanto, consiste em um conjunto de recurso R, desmembrado da solicitação da atividade e que irá influenciar no desempenho final do sistema, e da quantidade W de cada recurso ri que irá utilizar durante sua execução:

$$\boldsymbol{x = \{\omega r1, \omega r2, \omega r3, \ldots\ldots, \omega rn\}}$$

ω = representa a quantidade de recurso que será alocado

ri= representa o recurso alocado.

Apesar de parecer uma tarefa trivial, especificar todos os possíveis recursos pode se tornar uma tarefa não computável (NP completo), devido à quantidade elevada de fatores que podem impactar na quantidade de cada recurso, e ainda à influência considerável do inter-relacionamento entre os recursos, como por exemplo, em um dispositivo móvel; a disponibilidade de ω que representa a quantidade de processamento poderá refletir no consumo de energia de forma acelerada.

Por isso, o grau de exatidão que se espera que seja fornecido do preço deve ser levado em consideração durante o projeto da infraestrutura porque, dependendo da situação, a estimativa dos recursos necessários poderá exigir maior quantidade de processamento que a execução do serviço em si.

5.2.4 Taxonomia dos recursos

Os recursos naturais são componentes, materiais ou não, da paisagem geográfica, mas que ainda não tenham sofrido importantes transformações pelo trabalho humano e cuja própria gênese é independente do Homem, e aos quais foram atribuídos, historicamente, valores econômicos, sociais e culturais. Portanto, só podem ser compreendidos a partir da relação homem-natureza. Os recursos naturais são muito importantes para o Mundo.

Economicamente, são frequentemente classificados como recursos renováveis e não-renováveis, quando se tem em conta o tempo necessário para que se dê a sua reposição. Os não renováveis incluem substâncias que não podem ser recuperadas em um curto período de tempo, como por exemplo, o petróleo e minérios em geral. Os renováveis são aqueles que podem se renovar ou serem recuperados, com ou sem interferência humana, como as florestas, luz solar, ventos e a água.

Também podem ser classificados de energéticos e não energéticos, se atendermos à sua capacidade de produzir energia. Os carvões e o petróleo são recursos naturais energéticos. Por vezes a água é também considerada um recurso energético, pois as barragens transformam a força da água em energia.

A maioria dos minerais são recursos não energéticos, com exceção do volfrâmio, o urânio e o plutônio por se tratarem de substâncias radioativas e usadas para a geração de energia.

Narayanan (NARAYANAN, 2002) classifica os recursos computacionais em três categorias básicas:

- Baseados no Tempo
- Baseados no Espaço
- Esgotáveis

Segundo Narayanan essa classificação não é exaustiva, mas suficiente tanto para o trabalho do mesmo quanto para o nosso trabalho.

A. Recursos Baseados em Tempo

São recursos que possuem as características de disponibilidade ao longo de um tempo t. É representado por duas condições de utilização ao longo do tempo - utilização nula e utilização na sua totalidade disponível. São recursos que se encaixam nessa classificação, a utilização do processador e da rede.

Portanto, os recursos baseados em tempo apresentam-se definidos pela seguinte expressão(1):

$$s_r(t) = \begin{cases} R_t(t) Se\ aplicação\ utilizar\ o\ recurso \\ 0 \end{cases} \quad (1)$$

Em que

$s_r(t)$é a quantidade de recursos disponibilizados para o consumidor no tempo t

$R_t(t)$é o total de recurso disponível no tempo t

A demanda pela utilização do recurso pressupõe que no momento *t*, o recurso esteja disponível para uso do requisitante(2):

$$d_r(t) = \begin{cases} \infty \; se \; aplicação \; utiliza \; o \; recurso \\ 0 \end{cases} \quad (2)$$

Segundo Narayanan (NARAYANAN, 2002), o fornecimento de recursos baseados em tempo $R_r(t)$, com o custo de chaveamento pequeno, pode ser aproximado com um modelo GPS (General Processor Share) (3):

$$\sum_{AllApp} s_r(t) = R_r(t) \quad (3)$$

Devemos lembrar que diversos fatores inerentes foram omitidos nos cálculos apresentados para simplificar a definição de custo, porém, na prática, valores elevadores desses parâmetros podem ocasionar distorção no resultado.

B. Recursos baseados em espaço

Referem-se aos recursos que utilizarão determinado espaço lógico em determinado espaço de tempo. Um exemplo clássico dessa classificação é o espaço em disco.

Segundo Narayanan (NARAYANAN, 2002), há três tipos de objetos que poderão estar contidos num espaço:

- Objetos permanentemente armazenados em disco: estão armazenados em disco e não serão utilizados para qualquer cálculo de estimativa de fornecimento ou demanda.
- Cópias cachê de objetos: estão originalmente armazenadas em servidores remotos e são copiadas para a máquina local para processamento.
- Objetos temporários: são criados para a execução de um serviço e depois da conclusão, são apagados.

A quantidade que pode ser solicitada será limitada pela capacidade de fornecimento.

A definição da capacidade disponível pelo fornecedor é definida por (4):

$$d_r(t) = \sum_{x \in temp(t)} size(t) \quad (4)$$

Em que :

$temp(t)$ é o conjunto de objetos temporários pertencentes a uma aplicação no tempo t.

Por limitações de espaço, no caso de uma memória secundária e de desempenho em uma memória principal, Narayanan (NARAYANAN, 2002), define o custo para a execução de um serviço baseado na equação (5) :

$$c_r(t) = \begin{cases} \propto se\ d(t) > s(t) \\ 0 \quad casocontrário \end{cases} \quad (5)$$

Onde:

$s(t)$é a disponibilidade do fornecedor de recursos.

$d(t)$é a demanda por esse tipo de recurso por parte do consumidor

Constata-se que o custo para armazenar os dados é igual ao custo do fornecimento para o caso de o fornecimento ser maior que a demanda Uma situação que exigiria um armazenamento de dados maior que a capacidade do dispositivo, por exemplo, seria uma situação que não poderia ser atendida.

C. Recursos Esgotáveis

São os recursos por assim considerados não renováveis, sob a óptica da computação. Um recurso típico que se encaixa nessa classificação é a bateria. Existe uma quantidade limitada de energia que o dispositivo pode requerer. Após esse prazo, é necessária a troca ou recarga da bateria.

Segundo Narayanan (NARAYANAN, 2002), para caracterizar o fornecimento de energia são necessárias informações sobre o nível de recurso $E_r(t)$. Caso o dispositivo esteja sendo carregado no momento, teremos a taxa de recarga $R_r(t)$, e o nível possível de recurso $M_r(t)$, que seria a capacidade de fornecimento de energia. Ainda deve ser considerado que o recurso pode apresentar um limitador $A_r(t)$, no caso da bateria, o máximo que um cliente pode drenar de corrente elétrica.

Com esses parâmetros, o fornecimento de recursos esgotáveis é definido por (6):

$$s_r(t) = \left(E_r(t), R_r(t), M_r(t), A_r(t)\right) \quad (6)$$

D. Recursos relevantes

Com a classificação dos recursos computacionais, podemos definir uma lista de recursos que seriam interessantes que fossem tratados em comparação a uma possível checagem na operação de troca de recursos. Dentre esses recursos define-se (NARAYANAN, 2002) (CENDRON, 2008):

Tabela 2–Classificação de Recursos (CENDRON, 2008)

Recurso	Classificação	Unidade de demanda	Unidade de fornecimento
CPU	Baseado em tempo	Ciclos	Ciclos/s
Energia	Esgotável	Joules	Joules
Taxa de transmissão de rede	Baseado em tempo	Bytes	Bytes/s
Memória física	Baseado em espaço	Bytes	Bytes
Gravação em Disco	Baseado em tempo	Bytes	Bytes/s
Espaço em Disco	Baseado em espaço	Bytes	Bytes

Devemos destacar que os recursos apresentados na Tabela 2 se referem a recursos baseados em características físicas dos dispositivos computacionais. Num ambiente de rede podem ser abordadas outras características baseadas em características lógicas, como métricas de software.

Para exemplificar essa situação, imaginamos um serviço de impressão. Uma impressão pode apresentar características, como demora de impressão, que é um recurso baseado em tempo e a quantidade de tinta utilizada que é um recurso esgotável.

A lista de recursos não se esgota nessa tabela, devido à grande quantidade de serviços que podem estar envolvidos num ambiente de rede a heterogeneidade de dispositivos envolvidos e a criação de novas tecnologias podem se fazer necessárias para a classificação de novos recursos.

5.2.5 Combinando Recursos

A possibilidade de que um serviço requeira apenas um tipo de recurso é remota. E em grande parte das requisições, o serviço irá utilizar vários recursos de forma conjugada, como por exemplo, processamento e comunicação de dados por meio da rede de dados (CENDRON, 2008).

Com isso, podemos especificar os recursos necessários para um serviço por meio da declaração dos recursos utilizados $r_1(t), r_2(t), \ldots . r_n(t)$.

Para cada recurso que fará parte da solicitação, deverá ser especificado um valor de demanda, que seria a quantidade de recurso utilizado. Esse quantificador é uma tentativa de expressar a real quantidade de recursos de que um serviço irá necessitar. Aplicando esse valor ao conjunto de recursos utilizados obteremos a quantidade de recursos de que um serviço irá necessitar (7)

$$d_n(T) = \beta R_n(T) \quad (7)$$

Em que :

β representa a quantidade de recurso n necessário durante o tempo de execução T

Até esse ponto, obtivemos um vetor de recursos utilizados pelo serviço, representado por $\overline{D}$, contendo a relação dos recursos e suas respectivas demandas. Esse vetor é criado pelo consumidor, no momento em que necessita do fornecimento de um serviço ou, caso o fornecedor tenha essa capacidade, ela decompõe a solicitação e estipula os devidos valores.

Criada a devida lista $\overline{D}$, o peer fornecedor deverá montar a relação de preço para os recursos, representados pelo vetor $\bar{P}$. Cada recurso é representado dentro do vetor por p sendo que:

$$\forall p \in \bar{P}, p \geq 0$$

Com isso, a função de custo para um serviço $C_s$descrito na equação (7) sintetiza o custo como sendo a somatória dos preços individuais de todos os recursos pela sua quantidade de recurso necessária.

$$C_{s=\sum_i^n D_i p_i} \quad (7)$$

Por fim, ressaltamos que essa é uma equação genérica para a função de utilidade, já que os valores de demanda e preço podem sofrer alterações durante a execução.

Para a demanda seria em função da grande quantidade de fatores que pode influenciar nos valores pré-estabelecidos. Para esse caso, seria interessante manter um histórico das execuções e baseando-se nesses valores, estipular um desvio padrão para os valores solicitados pelo cliente.

5.3 ESTADO DA PRÁTICA

Neste item, apresentaremos as soluções existentes para o compartilhamento de recursos usando modelos econômicos em ambientes P2P. Essas soluções serviram de base de conhecimento para o desenvolvimento da implementação e do modelo de proposto.

5.3.1 Computacional Co-op

A Computacional Co-op (CIRNE e MARZULLO, 1999) permite que *sites* independentes se unam, formando um *grid*. A metáfora utilizada é uma cooperativa de compartilhamento de recursos. Para garantir benefícios a todos os *sites* envolvidos no *grid* de forma justa, o Co-op introduz um mecanismo que permite controlar a quantidade de recursos locais que estão sendo destinados ao *grid* e controlar a quantidade de recursos do *grid* obtido pelo site.

O mecanismo utilizado é um escalonamento proporcional (proporcional-share sheduling). Neste mecanismo, cada recurso apresenta uma determinada quantidade de *tickets*, e cada usuário recebe um número destes *tikets*. Em cada solicitação, o usuário emprega um número deles na submissão da aplicação. O algoritmo se baseia na proporção da quantidade de *tickets* possuída aplicação que deseja acessar o recurso e o número total de *tickets* existentes referente a ele para dar prioridade às aplicações.

O Co-op é um trabalho pioneiro ao prover garantias no escalonamento de *sites* compartilhados utilizando um modelo econômico. Todavia, o modelo utilizado não é suficientemente flexível para ambientes dinâmicos. O fato de a distribuição dos *tickets* ser feita no momento da criação da cooperação torna inflexível a participação do peer pois sua proporção é dada nesse momento e não poderá ser mais alterada.

5.3.2 *Grid Architecture for Computational Economy* (GRACE)

A *Grid Architecture for Computational Economy* (GRACE) (BUYYA, ABRAMSON e GIDDY, 2000) é uma arquitetura para o suporte à economia computacional em grids. A GRACE foi pensada levando em consideração os requisitos que uma infraestrutura de economia computacional deve preencher. Logo, inspirado pela ideia de mercados, os princípios de projeto da arquitetura são:

1. Um diretório onde seja possível publicar informações sobre as entidades que formam o Grid (i.e. consumidores e provedores);
2. Modelos para o estabelecimento de valores para os recursos/serviços;
3. Esquemas de cotação e mecanismos de oferta de serviços;
4. Modelos econômicos e protocolos de negociação de contratação de serviços;
5. Mediadores que atuam como reguladores e estabelecem valores para os recursos /serviços, criam moeda padrão e ajudam na resolução de impasses entre os negociadores;
6. Mecanismos para contabilização, cobrança e pagamento.

Os componentes da arquitetura são um gerente de negociação (*trade manager*), um conjunto de protocolos e APIs de negociação e um servidor de negociação. Dentro desse conjunto de componentes podemos citar:

1. Grid Resource Broker (e.g., Nimrod/G);
2. Grid Resource and Market Information Server;
3. Grid Open Trading Protocols and API;
4. Trade Manager;
5. Grid Trade Server.

O *Resource Broker* funciona como um buscador do usuário (ou de sua aplicação) perante a *grid*. Sendo assim, o *Resource Broker* desempenha atividades que permitem a execução da aplicação do usuário atendendo os seus requisitos (e.g. menor preço pelo serviço de execução). Além disso, um aspecto importante é que o *Resource Broker* exibe o *grid* para o usuário como um conjunto unificado de recursos. Essa abstração facilita a visão do usuário sobre o ambiente.

Certamente, o *Resource Broker* depende da existência de vários serviços. Por exemplo, serviços de informação sobre os recursos que são oferecidos no grid, seus preços e condições de uso. Esse serviço é fornecido pelo *Grid Resource and Market*

Information Server, o qual utiliza como base o serviço de descoberta de serviços do Globus .

Além de obter informação sobre serviços disponíveis e suas cotações, é necessário que haja um padrão (um protocolo bastante conhecido pelo cliente e pelo provedor de serviços) para a negociação. Logo, a arquitetura da GRACE apresenta um conjunto de protocolos e uma API que define regras e o formato para troca de comandos entre o cliente GRACE (i.e. *Trade Manager*) e o provedor do serviço (*Trade Server*).

Vale mencionar que o *Trade Manager* é uma parte importante do *Resource Broker*, pois tem o papel de guiar a seleção dos recursos de que a aplicação necessita para atingir seus objetivos. Por outro lado, o *Trade Server é* o agente de negociação do lado do provedor e sua função é maximizar a utilização dos recursos e o lucro do provedor.

Portanto, a negociação entre os *Trade Managers* (clientes) e os *Trade Servers* (provedores) ocorrerá de acordo com algum modelo econômico. Uma das implementações possíveis do GRACE utiliza como *broker* o Nimrod/G (BUYYA, ABRAMSON e GIDDY, 2000). O modelo econômico implementado foi o *Posted Price Market Model* descrito anteriormente. Nesse caso, os vários *Trade Servers* do sistema devem divulgar seus preços, de forma a atrair consumidores, esperando que os *Trade Managers* requisitem o serviço, tomando como base para escolha a comparação de preços entre os preços divulgados.

O gerente de negociação é o cliente GRACE. Ele utiliza as APIs de negociação GRACE para interagir com os servidores de negociação e negociar o acesso a recursos ao menor custo possível. Do lado do provedor de recursos, o servidor de negociação é um agente que negocia e vende o acesso ao recurso. Ele tenta maximizar a utilidade deste e prover lucros ao seu dono. Ele utiliza políticas de preço definidas pelo dono do recurso, que podem ser guiadas pela oferta e demanda do sistema. Os protocolos e APIs de negociação definem as regras e formatos para a troca de comandos e mensagens entre o gerente de negociação do cliente e um servidor de negociação.

Para viabilizar a gerência de recursos baseada em modelos econômicos, a GRACE é inserida na arquitetura do *grid* como um todo. São adicionados à arquitetura definida as aplicações dos usuários, o *middleware* utilizado e os gerentes de recursos locais, formando uma *Arquitetura de Gerência de Recursos em Grids Baseada em Economia*.

Nesta arquitetura, que é a composição da GRACE com a arquitetura proposta para o *grid* hoje , o gerente de recursos passa a ser um componente de um módulo maior do sistema, o *resource broker*. Internamente, o *resource broker* é composto de cinco entidades lógicas: um agente de controle de aplicações (*job control agent*), um conselheiro de escalonamento (*schedule advisor*), um explorador do grid (*grid explorer*), um gerente de negociações (*trade manager*) e um agente de *deployment* (*deployment agent*) .

Quando o *broker* recebe a submissão de uma aplicação, o agente de controle de aplicações analisa a aplicação e solicita um mapeamento de suas tarefas para recursos ao conselheiro de escalonamento. O conselheiro representa a política de escolha de recursos a ser utilizada pelo *broker*. É possível, por exemplo, priorizar os recursos capazes de prover um menor custo ou um menor prazo para a execução de uma tarefa.

De posse dos requisitos da tarefa, o conselheiro de escalonamento comunica-se com o explorador do *grid*, que localiza recursos que satisfazem os requisitos da aplicação no *mesmo*. Uma vez localizados os recursos, o gerente de negociações é encarregado de descobrir e negociar o custo da execução das tarefas neles, possivelmente reservando-os antecipadamente. Após a definição, por parte do conselheiro, de escalonamento de onde as tarefas devem ser executadas, o agente de controle de aplicações encarrega agentes de *deployment* de, efetivamente, executarem as tarefas, consumindo os recursos.

A GRACE ataca o problema gerencial do estímulo à participação e do ganho de acesso por parte dos usuários no *grid*. Como é proposta apenas uma arquitetura, existe a possibilidade de instanciá-la em diversos modelos econômicos, adequados a diferentes cenários. Da mesma forma, os problemas operacionais inerentes às suas implementações são postergados.

A. Nimrod-G

O Nimrod-G é um sistema para a modelagem e execução de aplicações de varredura de parâmetros em *grids* (BUYYA, ABRAMSON e GIDDY, 2000). Ele possui um *resource broker* implementado que utiliza mecanismos de negociação propostos na GRACE, suportando escalonamento de aplicações baseado em prazos e custos. A intenção do *resource broker* é prover acesso a quaisquer *middleware* presentes no *grid* por meio da negociação com seu dono.

A arquitetura em camadas do *broker* do Nimrod-G é composto de um *task farming engine* (TFE), um escalonador que faz a descoberta de recursos, negociação e escalonamento (*Meta-Scheduler*), um despachante, atuadores (*dispatcher and actuators*) e de agentes para gerenciar a execução das tarefas nos recursos.

Como proposto na GRACE, o escalonador do *broker* conta com um conselheiro de escalonamento - que possui os algoritmos de escalonamento -, um explorador do *grid* - que faz a descoberta dos recursos - e um gerente de negociação, que, efetivamente, entra em contato com os servidores de negociação dos recursos. O despachante e os atuadores são responsáveis por instalar agentes nos recursos selecionados para o monitoramento das tarefas que serão executadas.

O *broker* é utilizado por meio de clientes Nimrod-G. Os clientes podem ser ferramentas de criação automática de aplicações de varredura de parâmetros, mecanismos de controle e monitoração ou outras aplicações desenvolvidas pelo usuário. Por meio destes clientes é possível, além de, efetivamente, submeter as aplicações para serem rodadas no *grid*, definir restrições relativas a prazos e custos que influenciarão a forma como o escalonador selecionará os recursos.

Apesar de estar fora do escopo do trabalho do Nimrod-G, para a efetiva utilização de um sistema como esse, é necessária uma moeda e mecanismos de garantia de seu funcionamento, cobranças e pagamento entre as partes envolvidas nas negociações. O Nimrod-G se propõe a aguardar a maturidade de tecnologias de dinheiro eletrônico para resolver este problema. Uma vez tendo encontrada a solução, uma possibilidade de infraestrutura para viabilizar esta utilização é a criação de um banco, o *GridBank*, distribuído por meio de instituições participantes no *grid.*

5.3.3 OurGrid

Apesar de ser bastante pertinente, a introdução de modelos econômicos a fim de controlar o compartilhamento de recursos entre sites, e um grande número de aplicações, que, igualmente, necessitam de uma infraestrutura de recursos/serviços computacionais de larga escala, não requerem um contrato tão forte entre clientes e provedores, como as providas por uma arquitetura baseada em modelos econômicos. Ao manter o foco neste tipo de aplicação, torna-se possível desenvolver uma solução prática que pode ser usada, efetivamente, por uma comunidade de usuários.

Claramente, estamos apresentando um dilema entre se ter uma infraestrutura de escopo mais geral, porém mais complexa, o que dificultaria sua implantação efetiva, e uma infraestrutura mais simples, o que facilitaria sua implantação, porém

com um escopo mais restrito. Pensando nisso, foi desenvolvida a solução *OurGrid* (FEDERAL UNIVERSITY OF CAMPINA GRANDE, 2003) (ANDRADE, BRASILEIRO, *et al.*, 2003) . A ideia é apresentar uma solução simples que permita a criação de Grids Computacionais que fornecem poder computacional seguindo a política *best-effort*. O foco está em aplicações cujas tarefas são independentes as chamadas *Bag-of-Tasks*. O *OurGrid* é formado por três componentes: *MyGrid Broker* (CIRNE, PARANHOS, *et al.*, 2003), *OurGrid Peer* (ANDRADE, BRASILEIRO, *et al.*, 2003) e uma solução de *Sanboxing* baseada no Xen (BARHAM, DRAGOVIC, *et al.*, 2003).

O *OurGrid* explora a ideia de que um *grid* é composto de vários *sites* que têm o interesse em trocar *favores computacionais* entre si. Portanto, existe uma rede P2P de troca de favores que permite que os recursos ociosos de um *site* sejam fornecidos para outro quando solicitado. Para manter o equilíbrio do sistema, em uma situação de contenção de recursos, *sites* que doaram mais recursos (quando estes estavam ociosos) deverão ter prioridade junto à comunidade quando esta solicitar recursos.

A Figura 10 ilustra a ideia da rede de favores, onde cada peer controla um conjunto de recursos de um *site*. Ao surgir uma demanda interna por recursos que o peer de um determinado *site* não consegue suprir, este *peer* irá fazer requisições à comunidade.

A ideia é que os *peers* utilizem um esquema de prioridade baseado em quanto eles consumiram dos outros. Os resultados mostram que haverá um compartilhamento justo de recursos entre os *peers*.

A característica mais importante do *OurGrid* é conseguir prover uma solução útil e eficiente para uma comunidade de usuários em produção, apesar de se basear em soluções simples e de escopo limitado (i.e. apenas aplicações do tipo *Bag-of-Tasks*).

É importante notar que o objetivo do *OurGrid* contrasta com o objetivo do *Globus*, que fornece um conjunto de serviços para a construção da infraestrutura do Grid. Portanto, o *OurGrid* é uma solução que complementa o *Globus* provendo um *broker* (i.e. *MyGrid*) e abstrações que permitem ao usuário usar recursos *Globus* e não-*Globus*. Por outro lado, *Globus* complementa o *OurGrid* ao fornecer a infraestrutura de serviços para execução de aplicações em larga escala.

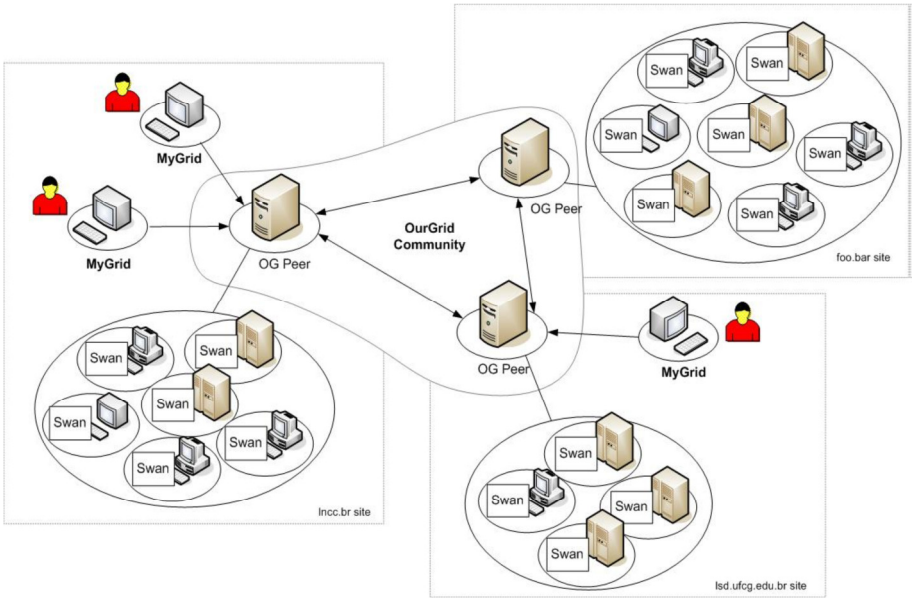

Figura 10 – Arquitetura *OurGrid* (FEDERAL UNIVERSITY OF CAMPINA GRANDE, 2003)

O *OurGrid* persegue um objetivo diferente do que seria prover uma solução genérica para computação em Grid. Com o foco em aplicações BoT, foi possível produzir uma solução efetiva para uma comunidade de usuários em produção. Não se quer dizer, com isso, que não se pretende introduzir novas funcionalidades, que aumentem o escopo da solução. Ao contrário, a ideia é, gradativamente, permitir que mais aplicações possam utilizar a solução. Por exemplo, suporte a *workflow*, suporte a *data-mining*, etc.

5.4 CONSIDERAÇÕES FINAIS

Durante o levantamento bibliográfico desta tese, uma parte fundamental foi a análise de abordagens para o incentivo de compartilhamento de recursos em redes *peer-to-peer* através do modelo econômico de troca, de modo que essas pudessem compor os trabalhos relacionados deste trabalho de pesquisa. Por isso, técnicas e arquiteturas de softwares para sistemas distribuídos foram pesquisados, de modo que esses pudessem compor uma proposta, tanto arquitetural quanto metodológica, para o compartilhamento de serviço por meio da troca multilateral, formadas sobre uma infraestrutura de rede *P2P.*

Todos estes trabalhos contribuíram para a elaboração de uma abordagem, tanto arquitetural, quanto algorítmica, para o uso de redes *P2P* como meio de compartilhamento de recursos.

6 MODELO ARQUITETURAL DECOMPARTILHAMENTO DE SERVIÇOS EM REDES P2P UTILIZANDO PERMUTA MULTILATERAL

6.1 INTRODUÇÃO

Neste capítulo, será apresentada uma proposta de uma arquitetura de software para lidar como o compartilhamento de recursos e serviços baseada no modelo econômico de troca, visando apoiar a definição de uma arquitetura de referência para esta funcionalidade. A abordagem proposta comtempla a definição de um processo de negociação que envolve a troca de recursos e/ou serviços computacionais. Esta proposta abrange, também, o estabelecimento de critérios e técnicas para a comparação dos objetos (serviços ou recursos) envolvidos na negociação, visando à definição de uma arquitetura de referência para o domínio do problema.

Chamado de Cnossos, a arquitetura transformou-se é uma ferramenta de software livre que permite a comunicação entre peers pertencente a um grupo. Os resultado permite aumentar na oferta de serviços que estarão disponíveis nos grupos dos peers. O nome foi retirado da mitologia grega. Cnossos (também grafado Knossos ou Knossus; em grego: Κνωσός, AFI: [knǫˈsǫs]) é o maior sítio arqueológico da Idade do Bronze da ilha grega de Creta, provável centro cerimonial e político da cultura e civilização minóica. Situado próximo da cidade moderna de Heraklion, atualmente Cnossos é visitado por muitos turistas, que visitam a "reconstrução" imaginativa feita a partir das ruínas que existiam no local.

6.2 CENÁRIO DE USO

Segundo (MATTOSO, WERNER, *et al.*, 2008), faltam técnicas de computação distribuída que escalem para configurações muito grandes, e que, ao mesmo tempo, tratem da autonomia, dinâmica e heterogeneidade dos recursos em ambientes computacionais P2P. Com base nisso, um exemplo de cenário de uso da abordagem proposta nesse trabalho é descrito a seguir. Nesse cenário, 4 (quatro)usuários necessitam compartilhar serviços para realizarem, cada um, tarefas distintas, em cada local onde essas pessoas trabalham. Todos precisam compartilhar serviços entre si e esse compartilhamento será feito por meio da a permuta multilateral desses serviços.

Os computadores pessoais de cada usuário possuem conexão com a Internet. Assim, a proposta desse trabalho é que cada computador pessoal se transforme em um *peer*, isto é, cada computador pode atuar como cliente ou servidor ao mesmo tempo, iniciar uma conexão com outros computadores conectados a uma rede P2P, a qualquer instante de tempo e, ainda, compartilhar os serviços computacionais existentes em computadores pessoais. Isso deve permitir que serviços computacionais sejam agregados (CHAPELL, 2008) a fim de que os serviços compartilhados pelos pesquisadores sejam mantidos por uma entidade computacional.

Quando os computadores pessoais dos pesquisadores se conectam a uma rede P2P,eles recebem um identificador único, construído com base no modelo URN (*Uniform Resource Name*). Com isso, cada computador pessoal passa a fazer parte de uma infraestrutura descentralizada de sistema distribuído. Nela, cada usuário pode compartilhar livremente os serviços computacionais de seus computadores. Esses serviços também receberão um identificador único. Apesar de nesse cenário somente existirem cinco pesquisadores, uma rede P2P pode ser formada pelos computadores de outros usuários.

Uma vez que seus computadores passam a fazer parte da rede P2P, os usuários podem compartilhar serviços computacionais de qualquer tipo e categoria. Assim, quando um deles compartilha um serviço sobre a rede P2P, informações sobre o compartilhamento poderão ser acessadas, a todo o momento, por qualquer outro usuário pertencente a essa rede. Assim, da mesma forma que um usuário compartilha um serviço por meio de palavras-chave que classificam o item compartilhado, ele também pode explicitar o seu interesse por algum outro serviço, bastando informar as possíveis palavras-chave relacionadas ao serviço idealizado. Nesse momento, cada usuário deverá também definir a relação de equivalência conforme descrito no item 7.2.1 existente entre cada um dos serviços que será disponibilizado na rede P2P.

Ao inicializar o sistema, à medida que os *peers* vão se conectando à rede, podem ser designados como *super*p*eers* ou *peer* comum da rede. As diferentes capacidades dos *peers* em relação ao poder da CPU, largura de banda ou capacidade de armazenamento, são levadas em conta quando se promove um *peer* à *super*p*eers*. Conforme os *peers* vão se conectando à rede, esses deverão associar-se a um *super*p*eers*. Em primeira instância os *peers* somente conhecerão o endereço dos *super*p*eers* aos quais eles estão associados. Somente por ocasião da permuta, os *peers* conhecerão os endereços dos outros *peers* que farão parte do processo. Em contrapartida, os *super*p*eers* deverão conhecer os endereços dos *peers* a eles associados como também o endereço de outro *super*p*eers* com os quais componha

uma rede baseando-se na topologia de Hipercubo (item 2.6.6). Um exemplo dessa topologia é apresentado na Figura 11.

.

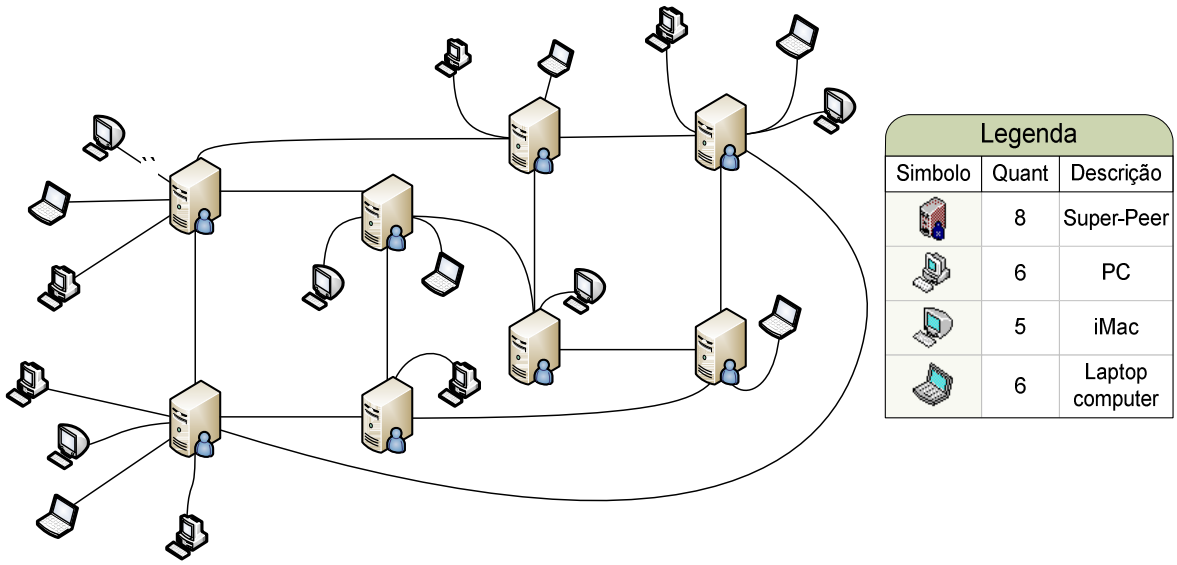

Figura 11 – P2P baseado em *Superpeers*

O processo de permuta, conforme pode ser observado na Figura 12, inicia-se com a consulta do consumidor *Peer* A (i), que envia uma requisição ao *super*p*eers* ao qual o *peer* está associado. Por sua vez, no primeiro momento, o *super*p*eers* envia uma solicitação para todos os outros *peers*, provedores do serviço solicitado, que tem cadastrado em sua tabela local de *peers* filhos(ii). De posse da requisição, os *peers* efetuam o processo da análise a fim de verificar a viabilidade da proposta. O *peer* fará essa análise tendo com base todos os serviços os quais seja proprietário, ou seja, possua o certificado de propriedade. Serão levados em consideração até mesmo os serviços que estão localizados originalmente em outros *peers*. Caso exista essa viabilidade, esse *peer* então envia diretamente para o *peer* consumidor uma resposta à requisição(iii). Por sua vez, ao receber as respostas de sua requisição, o consumidor efetuará a análise para determinar com qual *peer* o mesmo irá efetuar a troca. Ao final dessa análise, o *peer* enviará mensagens tanto para o *peer* vencedor(v) quanto para os *peers* que não irão participar da troca(iv). Em ambas as análises serão levadas em consideração as específicas equivalências dos serviços e a confiança/reputação que os *peers* possuem sobre o serviço em questão. O último passo do processo será a

troca de certificado de propriedade do serviço entre os *peers* que participam da troca (vi).

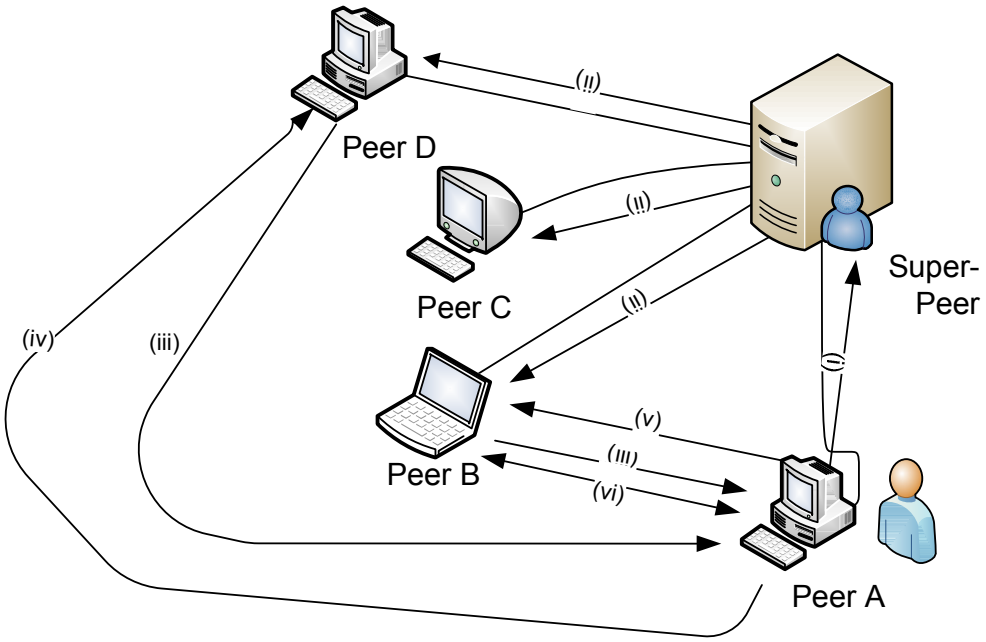

Figura 12 – Processo de Permuta – Cenário I

No cenário da Figura 12, descrevemos a situação na qual os serviços e seus certificados de propriedade estão localizados e pertencem aos *peers* que efetuaram a negociação de permuta. Em um outro cenário, poderá ocorrer uma negociação em que haverá a troca de certificados, porém a localização original do serviço (proprietário original) está sediada em outro *peer*. Nesse caso, fica caracterizada a permuta multilateral. A Figura 13 exemplifica uma situação em que o serviço esteja localizado fisicamente no *peer* C, embora seu direito de uso pertença ao *peer* B. No momento da troca dos certificados entre o *peer* A e o *peer* B, esse envia um certificado que já tenha sido negociado com o *peer* C (i). Uma vez de posse do certificado, o *peer* A entrará em contato com *peer* C para utilizar o recurso negociado (ii). Por sua vez, o *peer* C irá verificar a legitimidade do certificado para poder viabilizar a utilização do serviço pelo *peer* A.

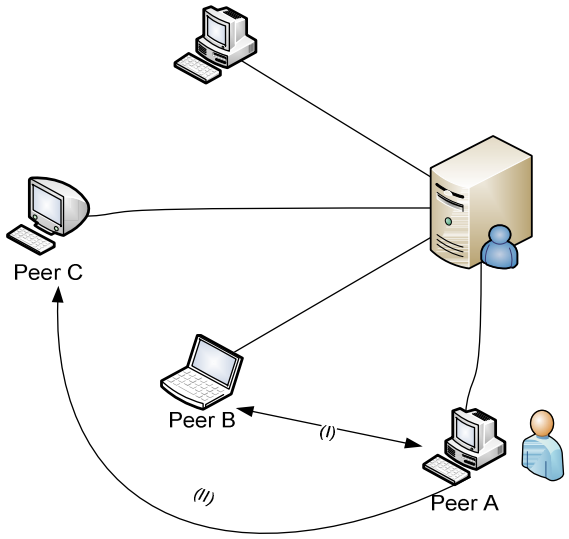

Figura 13 – Cenário Permuta Multilateral

Por fim, à medida que novos usuários ingressam nas redes computacionais, a disponibilidade dos conteúdos compartilhados mais importantes aumenta, pois, como mencionado anteriormente, cada usuário compartilha parte dos seus recursos computacionais para auxiliar troca dos mesmos. A escala da rede P2P cresce conforme o número de troca entre os usuários na mesma, ou seja, à medida que os relacionamentos de trocas são criados, quantidades de serviços computacionais são ofertadas para uso na rede.

6.3 MODELO ARQUITETURAL

O fato de existirem diversos tipos de recursos envolvidos em diferentes níveis de funcionalidade de sistema, torna o problema de compartilhamento de serviços uma questão bem desafiadora. Começando pela decomposição dos serviços e recursos que contribuem em suas diferentes dimensões e finalizando na análise dos custos correspondentes de cada um deles (MOURA, OLIVEIRA, *et al.*, 2007). A partir desta imagem detalhada, vamos, então, fazer as abstrações necessárias, a fim de gerir um modelo simples para o compartilhamento desses serviços e recursos.

A arquitetura proposta, mostrada na Figura 14, utiliza o modelo de Troca (Permuta) na concepção de seus componentes para resolver o problema de compartilhamento de serviços e recursos em uma rede P2P (MOURA, OLIVEIRA e

FRANCA, 2009). O modelo consiste na permuta de serviços entre os *peers*, não havendo a utilização de nenhum tipo de moeda. A troca é efetuada em função de fatores que serão analisados no ato de negociação como, por exemplo, a oferta, a demanda e a sazonalidade do serviço, entre outros. Poderá ser levada em consideração também, na negociação, uma maior distribuição da demanda, possibilitando que haja um possível equilíbrio entre a quantidade de recursos ofertados e a quantidade demandada.

A arquitetura possui componentes que são responsáveis por controlar a troca e o fluxo de serviços entre os participantes da rede P2P. A implementação dos mesmos visa aumentar a disponibilidade de serviços computacionais.

Dependendo das características da negociação, a arquitetura conta com componentes que funcionaram como consumidor e como provedor de serviços, em um mesmo *peer*, que são responsáveis em efetuar a transação de troca entre os serviços. Sendo assim, o *peer*, dentro da negociação poderá ter apenas uma função: como provedor ou como consumidor do serviço. Para que um determinado *peer* possa fazer uso de um serviço, basta que o mesmo disponibilize quaisquer tipos de serviços pertencentes a ele.

A arquitetura está baseada na ideia de que um peer apenas detém acesso aos serviços, se possuir outros serviços para disponibilizar no ambiente colaborativo (MOURA, FRANÇA, *et al.*, 2007). Essa característica irá permitir que mais de um *peer* tenha acesso aos diversos serviços e recursos da rede, e a garantia de consistência para permitir que os serviços disponíveis sejam sempre os mais atuais. Uma outra finalidade da arquitetura é diminuir os *free-riders* objetivando, assim, reduzir os *peers* chamados de maliciosos e egoístas da rede.

Cada *peer* opera independentemente e de forma assíncrona e é identificado de forma única por meio do seu PID. Os *peers*, dependendo de suas funcionalidades e complexidade, podem prover serviços e/ou usar serviços de outros pares.

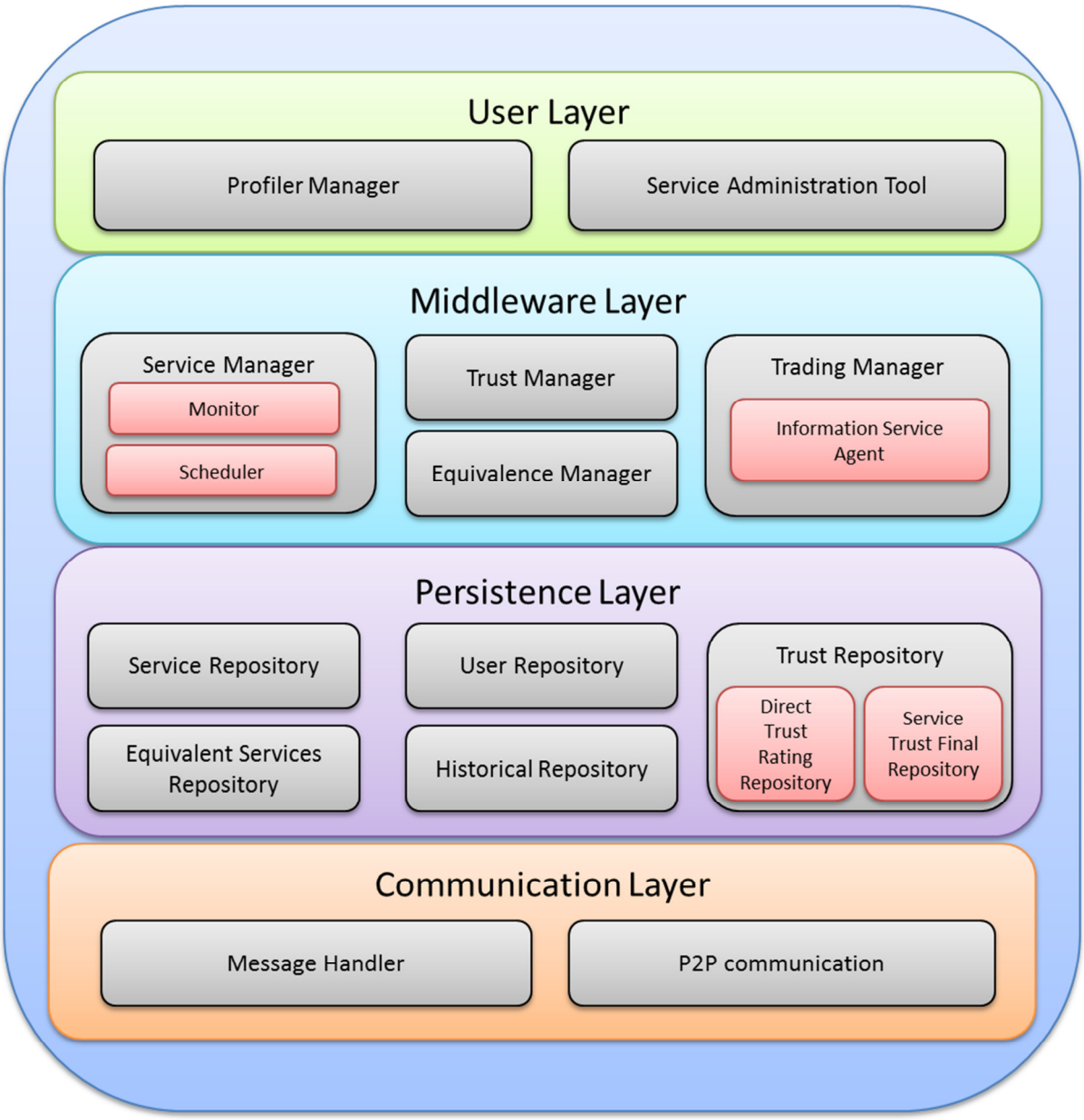

Figura 14 - Arquitetura Proposta

Para lidar com esta necessidade arquitetural, é necessário criar uma arquitetura de software que seja capaz de se adaptar, facilmente, aos diversos padrões e tecnologias utilizados pelas diferentes redes P2P. Sendo assim, propõem o uso de uma arquitetura de software chamada de *microkernel* (BUSCHMANN, MEUNIER, *et al.*, 2001), que oferece um conjunto de funcionalidades básicas, permitindo que estas sejam estendidas por aplicações específicas de software. Um *microkernel* também serve como um ambiente para plugar e coordenar estas aplicações específicas de software. Isto faz com que um desenvolvedor de software se preocupe somente com detalhes específicos do domínio de aplicação, isto é, não é

necessário se preocupar com detalhes de baixo nível de abstração. A justificativa para isto é que uma arquitetura *microkernel* oferece benefícios como portabilidade, flexibilidade e extensibilidade, separação de políticas e mecanismos, escalabilidade, confiabilidade e transparência. Por esses motivos, a proposta de arquitetura de software para compartilhamento de recursos através da permuta multilateral, em redes *peer-to-peer* está organizada na forma de um *microkernel* distribuído.

Para construir o *microkernel*, foi necessário separar dois aspectos importantes: comportamento e conteúdo. Para lidar com o comportamento, foi elaborada uma arquitetura de software orientada a serviços móveis, devido à natureza heterogênea dos dispositivos computacionais fixos, formadores do ambiente de computação distribuída que estamos estudando. Após isto, foi criada uma proposta que pudesse fornecer um arranjo de componentes para lidar com o aspecto conteúdo. O principal objetivo desta arquitetura de software é fornecer elementos arquiteturais para lidar com a criação de redes *peer-to-peer* e, em seguida, utilizar o modelo de permuta multilateral para o compartilhamento de serviços dos participantes.

Desta forma, o *microkernel* foi organizado em quatro camadas, de modo que os elementos arquiteturais pertencentes ao mesmo nível de abstração pudessem ser agrupados em uma mesma camada. Seguindo este critério de organização, o *microkernel* é composto pelas seguintes camadas: ***User Layer, Middleware Layer, Persistense Layer e Communication Layer.***

Um cliente é alguma aplicação que faz uso dos serviços disponibilizados pelo *microkernel*. Estes serviços compõem duas categorias de serviços, que são: serviços básicos e serviços de aplicações. Serviços básicos são aqueles relacionados ao funcionamento interno do *microkernel*. Esta categoria de serviços não precisa de uma aplicação para disponibilizá-los, pois os mesmos são disponibilizados pelo próprio *microkernel*. Já os serviços de aplicações são aqueles desenvolvidos para atender os requisitos de alguma aplicação. Para utilizá-los, é necessário que a aplicação seja instalada no *microkernel*. Baseado no modelo arquitetural lógico exibido pela Figura 15, um usuário, por meio de um cliente do *microkernel*, faz uma requisição para execução de um serviço à instância do *microkernel que* está em execução no computador pessoal do usuário. Esta perpassa pelas camadas do *microkernel* até chegar à camada responsável pelo seu tratamento. Logo após o tratamento da requisição, uma resposta é enviada ao cliente, que extrai os dados retornados pelo serviço requisitado.

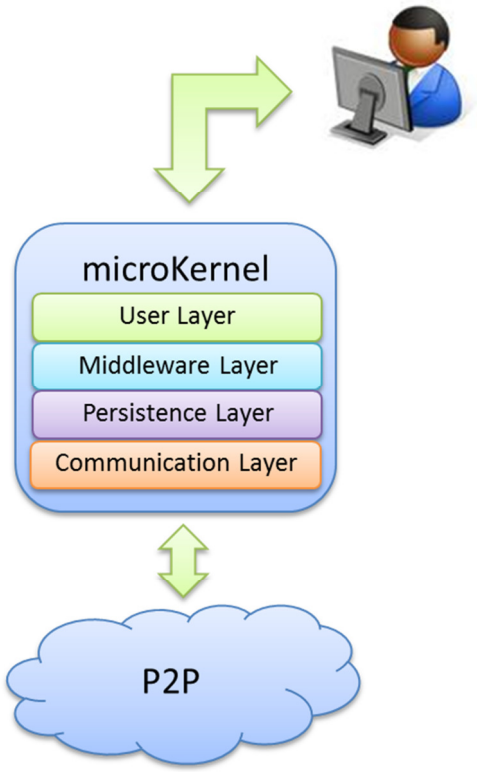

Figura 15 - Modelo arquitetural Lógico

Se a requisição for local como, por exemplo, recuperar a quantidade de espaço livre em disco do computador, ela será tratada no próprio computador do usuário e uma resposta será enviada até a camada de nível mais alto. O oposto à realização de uma requisição local se constitui quando uma requisição remota é realizada. Este tipo de requisição perpassa todas as camadas da arquitetura de software, mas diferentemente de uma requisição local, a requisição remota é submetida ao sistema distribuído. Em seguida, esta requisição recebida por um ou mais *peers* conectados à rede *peer-to-peer*. Neste momento, cada um destes *peers* trata a requisição remota e devolve uma resposta ao *peer* dono da requisição. Isto é possível, pois os computadores pessoais de usuários conectados ao ambiente distribuído são transformados em *peers*, uma vez que cada um deles executa uma instância do *microkernel.* Quando a resposta à requisição remota chega ao seu destinatário, é tratada e, em seguida, tem seus dados encaminhados até a camada de mais alto nível do modelo arquitetural.

O encaminhamento de respostas de uma camada de baixo nível para outra de nível superior subsequente é feito por meio de *callbacks* (BUSCHMANN, MEUNIER, *et al.*, 2001). Desta forma, os elementos arquiteturais de camadas de um nível mais alto de abstração podem reagir às respostas dadas por elementos arquiteturais

pertencentes às camadas de um nível de abstração menor. Então, para sincronizar a comunicação entre as camadas, foi utilizado o padrão de projeto *Observer* e o *Decorator* (GAMMA, HELM, *et al.*, 2001). Estes padrões permitem definir uma dependência entre uma camada e outra, de maneira que se uma camada de nível de abstração menor muda o seu estado, a camada de um nível de abstração acima é notificada e atualizada automaticamente.

Após toda a apresentação da estruturação do modelo arquitetural proposto por esta tese, as próximas seções detalharão cada camada e seus respectivos elementos arquiteturais, de modo que fique claro como se dá o funcionamento interno de uma arquitetura de software para compartilhamento de recursos em redes sociais *peer-to-peer*.

6.3.1 *UserLayer*

O número de serviços que um *peer* disponibiliza é dinâmico e pode mudar no decorrer da execução da aplicação. A *UserLayer* é responsável pelo recebimento e encaminhamento das requisições disparadas pelo usuário, a partir de um cliente do *microkernel*, extração de informações presentes nas requisições submetidas pelos clientes e envio de requisições para a inferior, de modo que esta possa iniciar o gerenciamento de alguma interação entre um cliente e o *microkernel.* Para tanto, é necessário que o usuário esteja conectado ao sistema distribuído, pois a camada precisa acessar informações sobre o usuário, objetivando tomadas de decisões corretas no que diz respeito à propagação das requisições para as camadas de níveis mais baixos da arquitetura de software.

Quando um usuário se conecta ao ambiente distribuído, a *User Layer* é notificada. Neste momento, a camada recebe todas as informações relacionadas ao usuário conectado, fazendo com que seu único elemento arquitetural seja ativado. Este, por sua vez, é chamado de *Profile User*, conforme apresentado pela Figura 16. Cada membro da rede virtual possui sua própria política de permissão de serviços. Essa peculiaridade possibilita a existência de *peers* mais rígidos e outros mais permissivos. Por esse motivo, são avaliadas as características dos recursos disponibilizados a cada pedido diferente aberto pelo cliente. Essa ação permite que o usuário da aplicação P2P altere seu repositório de serviço e, no mesmo instante, todo o sistema se adapte à nova condição.

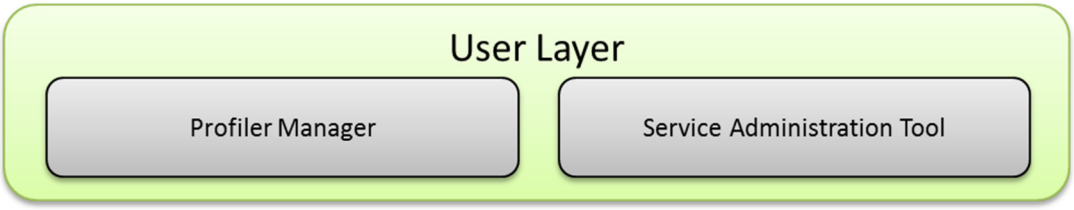

Figura 16 - User Layer

As informações mantidas pelo *Profile Manager e pela Service Administration Tool (SAT)* não podem ser alteradas, pois elas são somente para leitura. Caso seja necessário alterar alguma informação, o usuário deve disparar uma requisição para um serviço do *microkernel* responsável em tratá-las. Portanto, *essa camada* atua como um *cache*, mantendo as informações sobre o usuário mais próximas dos clientes do *microkernel*.

Para que um cliente possa acessar as informações mantidas pela c*amada*, ele deve utilizar a interface do componente plataforma. Nesta interface, existem operações que retornam informações sobre o usuário que estiver conectado no momento. Então, ao utilizar esta operação, o cliente faz com que a plataforma solicite ao *microkernel* as informações do usuário. Consequentemente, o *microkernel* acessa a *User Layer* que, por sua vez, retorna as informações requeridas pelo cliente.

Além de manter informações sobre o usuário, a camada também tem a responsabilidade de interagir com a *Midlleware Layer*. Essa responsabilidade consiste na formação de caminhos que possam guiar respostas até um cliente do *microkernel*. Neste caso, a abertura de caminhos consiste em solicitar à *Midlleware Layer* a criação de sessões, com o objetivo de manter informações sobre o tratamento das requisições submetidas a partir de um cliente do *microkernel*. No entanto, não são todas as requisições que necessitam de abertura de sessões, e sim somente aquelas que são submetidas de maneira assíncrona ao ambiente distribuído. Para requisições submetidas de maneira síncrona, as respostas são dadas por meio dos retornos das operações.

A execução de serviços básicos ou de aplicações pode ser requisitada por meio de uma requisição por serviço, submetida de maneira assíncrona. Por este motivo, *essa camada* verifica em suas informações se o serviço requisitado é um serviço básico ou de aplicação. Logo após, o *Usuário* solicita à *Midlleware Layer* a criação de sessões adequadas para os dois tipos de serviços. Por fim, uma vez que as sessões tenham sido criadas pela *Midlleware Layer*, a *SAT* mantém uma lista de referências para as mesmas.

Nesse contexto, a camada do Usuário (*User Layer*) fornece os mecanismos e as interfaces necessárias para que o administrador do *peer* possa registra informações correspondentes a cada uma de suas características. É por meio dessa camada que o administrador informa os serviços que serão disponibilizados na rede pelo *peer*. Esses registros são feitos por meio de uma interface RIA (WIKIPEDIA). Essa camada possui dois componentes: *Profile Manager* e o *Service Adminstration Tools.*

A. Profile Manager

Componente responsável por capturar e por prover o contexto do usuário de dentro da arquitetura. Captura as propriedades pessoais do usuário como, por exemplo: nome, login, senha e as armazena no *User Repository* da camada de Persistência (*Persistence Layer*). Ele também é responsável por gerar um identificador e fornecer para o peer participante essa identificação -PID(*Peer Identification*).

B. Service Administration Tools

Mantém um serviço de informações sobre serviços compartilhados de maneira que um consumidor interessado em utilizar um determinado serviço localize os provedores que, potencialmente, atendam as suas necessidades. Nesta informação de contexto, é possível obter informações sobre que os serviços básicos do *microkernel* estão disponíveis para o usuário. A *Camada de Usuário* utilizada essas informações para identificar se uma requisição, para alguns serviços do *microkernel*, pode ser propagada para as próximas camadas em sentido *top-down*;

Mantêm, também, informações sobre as aplicações do usuário instaladas no *microkernel*. Por esta informação de contexto, é possível obter os caminhos de aplicações e serviços. A partir delas, é possível identificar se uma requisição para execução de uma aplicação ou serviço pode ser encaminhada ou não. Para isto, a *Camada de Usuário* verifica se o caminho do serviço e os parâmetros de uma requisição são compatíveis com algum padrão de caminho de serviço mapeado para o usuário

Possui uma interface utilizada pelo administrador para monitorar os recursos disponíveis e pertencentes ou não ao *peer*. Nessa interface, o administrador poderá registrar, também, informações sobre a disponibilidade de cada um dos serviços.

6.3.2 Middleware Layer

A camada de Middleware (*Middleware Layer*) é responsável por gerenciar, distribuir e sincronizar as informações entre os *peers*. Ela é a camada que possui a características de gerência na arquitetura. Qualquer tarefa que deva ser executada com certa periodicidade deve ser enviada a essa camada.

Está Dividida em quatro componentes (Figura 17): Trading Manager, Trust Manger, Equivalence Manager e Service Manager.

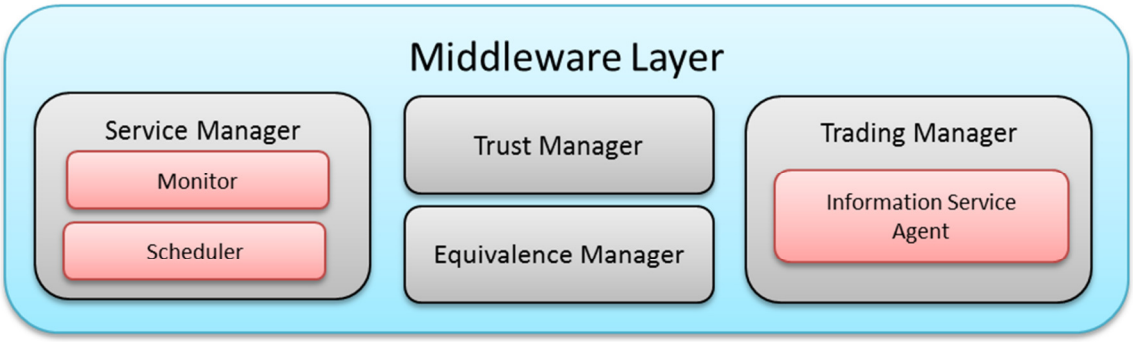

Figura 17 - Middleware Layer

A. Service Manager

Esse componente tem com responsabilidade gerenciar os serviços que serão utilizados para negociação de troca. Tem a função de gerenciar as informações sobre as execuções em um provedor, como o nível de procura por um determinado serviço, o percentual de compartilhamento do serviço, e o percentual de utilização desse serviço pelos consumidores da rede, em um dado período do dia, entre outros. A análise é feita de acordo com as informações que estão armazenadas no *Service Repository* e no *Historical Repository.*

Dividido em dois serviços, o *Monitor* e o *Scheduler*, esses componentes têm como responsabilidade a gerência dos processos que estão ou serão executados pelo *peer*.

Monitor é um serviço que permanece em execução em tempo integral. Com uma frequência definida pelo administrador, verifica a carga do processador local e armazena essa informação no repositório de informações locais do provedor, no *Services Repository*. *O Monitor* tem, também, a função de gerenciar as informações sobre as execuções em um provedor, como o nível de procura por um determinado

serviço, o percentual de compartilhamento do serviço, e o percentual de utilização desse serviço pelos consumidores da rede, em um dado período do dia, entre outros.

Scheduler é o escalonador local que tem a função de iniciar e gerenciar a execução de uma aplicação de um consumidor**.** Também é utilizado pelo gerenciador para agendar tarefas a serem executadas. Esse pode ser utilizado para agendar a execução de um serviço caso este esteja fora do ar no momento da execução.

B. Trading Manager

Junto com o *Trust Manager e o Equivalence Manager* é o componente central da arquitetura. *O Trading Manager* é responsável pela negociação dos recursos que serão utilizados. Possui um componente chamado *ISA (Information Service Agent).* Tem o papel de analisar, entre outros, os dados colhidos pelo *ISA.* Também determinar, em função dos parâmetros estabelecidos pelo administrador do provedor, quais os recursos serão negociados na troca.

O *Information Service Agent* (ISA) tem o papel de analisar e encontrar os provedores. Esses deverão possuir serviços que estejam de acordo com as características solicitadas pelo componente *Trading Manager*, em função dos parâmetros estabelecidos pelo consumidor. O componente é responsável por negociar a utilização de um serviço local. Por meio dos dados fornecidos pelo *Monitor e do Equivalence Manager*, o componente procura alocar o serviço de forma que a negociação seja feita da maneira que 0*peer* tenha um melhor benefício possível. Para essa análise o ISA irá levar em conta os serviços que o *peer* tem a disposição para a troca e que possuem também uma equivalência.

Por parte do consumidor, o componente é utilizado no momento de submeter uma requisição do serviço desejado pelo *peer*. Considerando parâmetros estabelecidos pelo usuário (como por exemplo, quantidade de recurso e prazo de execução, entre outros),o componente envia a requisição para o *ISA.*

C. Trust Manager

Componente da arquitetura que coleta, distribui e agrega informações sobre o comportamento dos participantes nas interações realizadas. Também avalia o comportamento dos serviços, determinando o nível de confiabilidade que cada um possui dentro da rede P2P. Dessa forma, auxilia o *Trading Manager* a decidir em quem confiar e com quem negociar. Tem a característica de motivar o bom

comportamento dos participantes, e procura controlar a participação daqueles que são considerados desonestos. Utiliza o *Trust Repository* para armazenar as informações.

D. Equivalence Manager

Fornece ao *Trading Manager* opções de escolha do serviço que poderá ser utilizado no processo de troca. A escolha do serviço deverá ser feita de maneira que o *peer* tenha um maior benefício na hora da negociação. A escolha é feita por meio de consulta e análise das informações que se encontram no *Equivalent Services Repository(ESR)*.

Periodicamente, o componente realiza um processo de análise das informações sobre a demanda por um serviço. Essa análise visa identificar os padrões de demanda existentes para esse serviço. As informações que serão utilizadas na análise estão contidas no *Historical Repository(HR).* Esse processo tem como finalidade determinar novos valores de equivalência para os serviços. Após essas análises, poderá haver uma atualização nos valores equivalentes armazenados no ESR.

6.3.3 Persistency Layer

A *Persistency Layer* é responsável por gerenciar o armazenamento das informações nos *peers*. É implementada por meio da utilização de um modelo de replicação proposto em (SILVA, 2008) que visa garantir consistência de forma simples, mantendo os dados atuais disponíveis na rede e facilitando sua localização em ambientes P2P com Mobilidade. A proposta apresentada consiste de algumas alterações no modelo único mestre, criando um modelo denominado múltiplo únicos mestres. A principal preocupação deste trabalho está na manutenção dos itens de dados mais atuais na rede para localização.

Essa camada é dividida em cinco componentes (Figura 18):

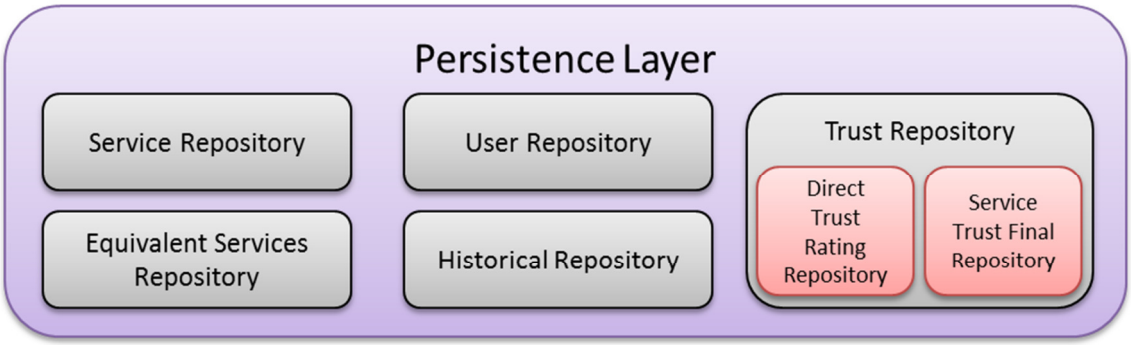

Figura 18 - Persistence Layer

A. User Repository

Responsável por armazenar as informações capturadas pelo Profile Manager. Armazena as informações pessoais do usuário como, por exemplo: nome, *login*, senha, data e hora dos acessos à rede.

B. Service Repository

Responsável por armazenar informações relativas aos serviços que serão compartilhados. O formato dessa informação está descrita na Tabela 3.

Tabela 3 - Informações presentes no descritor

Atributo	Descrição
UID	Identificador Único do Serviço
Tags	Identificadores para consulta do Serviço
Description	Descreve o propósito do Serviço
PID	Identificador do *peer* de origem do Serviço
OwnershipCertificate	Indica o dono do serviço.
Availability	Indica se o serviço está disponível ou não
Quantity	Quantidade de recurso disponível (em percentual)
Period	Intervalo de tempo no qual o recurso estará disponível

C. Equivalent Service Repository

Responsável por armazenar as informações sobre as equivalências existentes entre serviços. Como citado anteriormente, essas são utilizadas pelo componente *Equivalence Manager*. A Tabela 4 descreve a forma como essas informações serão armazenadas.

Tabela 4 – Descritor dos valores de Equivalência

ATRIBUTO	DESCRIÇÃO
UID	Identificador Único da Equivalência
Service	Descrição do Serviço pertencente à equivalência
ServiceQuantity	Quantidade de serviço a ser comparado
Equivalence	Descrição do serviço equivalente
EquivalenceQuantitty	Quantidade do serviço equivalente a ser utilizado na comparação
Schedule	Período de tempo em que os valores de equivalência serão válidos.

D. Historical Repository

Responsável por armazenar todas as negociações realizadas pelos peers, construindo, assim, um histórico de todos os processos de negociação ocorrido com ou sem sucesso. A Tabela 5 descreve a forma como essas informações serão armazenadas.

Tabela 5 – Descritor do Histórico da Negociação

Atributo	Descrição
UID	Identificador Único do Histórico
PIDBarter	Identificador do peer que solicitou a troca
TradingUUID	Identificador da Troca
ServiceResquestID	Identificador do serviço solicitado
QuantityServiceRequest	Quantidade de serviço solicitado
ServiceOfferedID	Identificador do serviço oferecido na troca
QuantityServiceOffered	Quantidade de serviço oferecido na troca
DateTimeBarter	Data e hora da negociação
Period	Período da Negociação
Status	Identificador se foi ou não efetuada a transação

E. Trust Repository.

O Trust Respository é responsável por armazenar todas as informações referentes à confiança dos *peer* e dos serviços. É dividido em dois componentes: *Direct Trust Rating Repository*, que armazena os valores das avaliações dos serviços utilizados, e o *Services Trust Final Repository* que armazena a valor final da confiança de um serviço. A Tabela 6 e a Tabela 7 descrevem a forma como a reputação do *peer* e do serviço serão armazenadas.

Tabela 6 – Descritor da *Direct Trust Rating Repository*

Atributo	Descrição
UID	Identificador Único da Avaliação
ServiceID	Identificador do Serviço Avaliado
ValiationValue	Valor da avaliação do Serviço
DateTime	Data e hora em que foi avaliado

Tabela 7 – Descritor da *Services Trust Final Repository*

Atributo	Descrição
UID	Identificador Único da reputação
ServiceID	Identificador do Serviço
DateTime	Data e hora em que foi avaliado
TrustFinal	Valor final da Confiança do Serviço

6.3.4 Communication Layer

A *Communication Layer* é responsável pelo gerenciamento da comunicação entre os peers. Cabe lembrar que não é a proposta dessa tese apresentar um novo modelo de comunicação em rede P2P porém usa modelos já consagrados para efetuar essa tarefa.

Figura 19 - Communication Layer

É composta por dois componentes: Message Handler e P2P communication.

Devido à distinção que fazemos dos *peers* em *superpeers* e *defaultpeers*, a camada de comunicação nestes dois tipos de *peers* se comporta de maneira diferente. Uma vez que *superpeers* intermedeiam a comunicação entre *defaultpeers* e monitoram mensagens que possuem informações de contexto a respeito do ambiente distribuído, os mecanismos da camada de comunicação de um *superpeer* tratam o mesmo conjunto de mensagens que um *defaultpeers* pode tratar. No entanto, os *superpeers*, quando recebem uma mensagem cujo destino não é igual ao seu identificador, a repassam para seus vizinhos *superpeers*, de modo que ela possa ser

entregue ao seu destino. Por fim, os *superpeers* monitoram mensagens que contêm requisições para criação e manipulação de informações sobre a negociação de troca.

A. Message Handler

Esse componente tem a responsabilidade de gerenciar a troca de mensagem entre os *peers* do modelo de permuta apresentado. Nele é implementado todo o protocolo de comunicação entre os *peer*, desde a simples troca de mensagem para criar um comunicação até mesmo o envio de mensagem contendo as informações sobre a concretização da permuta.

Uma mensagem só pode ser entregue se o destinatário da mensagem for vizinho do *peer* que a enviou ou se houver um caminho que possa ser percorrido até que a mensagem seja entregue em seu destino. Para lidar com isso, a *Communication Layer* conta com o elemento arquitetural *Message Handler*. Este elemento arquitetural mantém nos *superpeer* duas tabelas de vizinhos: uma de *defaultpeers* vizinhos e outra de *superpeers* vizinhos. As duas tabelas armazenam referências para os *peers* que estejam imediatamente conectados a um super*peer*. A partir de cada vizinho registrado na tabela, é possível adicionar rotas para *peers* que não sejam vizinhos imediatos. Este registro de rotas é constantemente alimentado, à medida que os super*peers e os defaultpeers* trocam mensagens uns com outros. Por meio disto, a *Communication Layer* extrai das mensagens os caminhos percorridos por elas quando passam por alguns *peers* da rede. Além disto, também é extraído o tempo gasto por uma mensagem, à medida que esta é encaminhada sobre um número finito de *peers*, até chegar ao seu destino. Estas informações são úteis, pois elas são utilizadas para avaliar o menor custo para envio de mensagens de um *peer* para outro.

B. P2P communication

Todos os elementos arquiteturais usam o elemento de P2P communication, pois ele encapsula detalhes sobre os protocolos de comunicação disponíveis no computador do usuário. Por meio desse componente, são realizados os envios e recebimentos de mensagens entre dois computadores, desde que um possa se conectar ao outro por meio de *sockets*. Assim, quando uma mensagem precisa ser enviada para outro computador, o elemento de *P2P communication* serializa esta mensagem, gerando um *stream*, que, em seguida, é enviado ao computador que deve receber esta mensagem. Quando este *stream* é recebido por este computador, o

elemento de P2P *communication* o recupera e o transforma em uma mensagem novamente.

Para viabilizar o funcionamento interno do elemento de *P2P communication*, foram utilizados três padrões de projeto de software: *Wraper Façade* ,*Forwarder-Receiver* , e *Client-Dispatcher-Server*). O *Wrapper Façade* é um padrão de projeto que encapsula funções e dados fornecidos por APIs não orientadas a objetos como, por exemplo, *sockets*. Por meio deste padrão, foi possível encapsular os protocolos de comunicação TCP e UDP, fornecendo interfaces bem definidas para o envio e recebimento de *streams* entre dois computadores. O *Forwarder-Receiver* é um padrão de projeto que fornece de maneira transparente a comunicação interprocesso em sistemas *peer-to-peer*. Este padrão utiliza dois componentes, *forwarder* e *receiver*, para desacoplar os *peers* dos mecanismos básicos de comunicação. Os *forwarders* são componentes responsáveis pelo envio de mensagens de um computador a outro. Antes de enviar uma mensagem para outro computador, um *forwarder* a transforma em um *stream* e, em seguida, o envia para seu local de destino. Já os *receivers* são responsáveis pelo recebimento de mensagens. Neste momento, um *receiver* recebe um *stream* e, em seguida, transforma esta sequência de bytes em uma mensagem novamente. Para desacoplar totalmente um *peer* de qualquer mecanismo de comunicação interprocesso, *forwarders* e *receivers* são decorados com *wraper façades*. Por fim, o *Client-Dispatcher-Server* é um padrão arquitetural que fornece transparência de localização de entidades do sistema distribuído, além de encapsular detalhes sobre a comunicação destas entidades. Para isto, um *dispatcher* mantém uma estrutura de dados que permite mapear identificadores em uma tupla, contendo endereço IP, protocolo de transporte e porta de comunicação. A partir destes mapeamentos, os elementos arquiteturais da *Communication Layer* podem solicitar ao *dispatcher* uma conexão para envio de mensagens para algum *peer*, a fim de que este execute algum serviço. No entanto, o elemento arquitetural *P2P communication* só manipula mensagens cujos descritores tenham sido registrados no *dispatcher*, que, por sua vez, associa um manipulador de mensagens a estes descritores. Detalhes sobre protocolos e trocas de mensagens entre os *peers* são apresentados no Capítulo 8.

6.4 CONSIDERAÇÕES FINAIS

O modelo arquitetural para compartilhamento de serviços computacionais em redes P2P apresentado tem como objetivo colaborar na construção de soluções para problemas relacionados à negociação de serviços dentro desse tipo de rede.

Ambientes de rede P2P são extremamente heterogêneos. Por isto, é necessário que uma arquitetura de software para compartilhamento de recursos forneça portabilidade, flexibilidade e extensibilidade, separação de políticas e mecanismos, escalabilidade, confiabilidade e transparência. Pensando nisto, este capítulo apresentou a proposta de uma arquitetura de software para compartilhamento de recursos em redes sociais *peer-to-peer*. Esta proposta fornece uma estrutura básica para construção de *middlewares* que forneçam facilidades no desenvolvimento de aplicação para *esse tipo de rede*, sem que estas dependam fundamentalmente de APIs que causem algum tipo de dependência com relação ao hardware ou ao tipo de infraestrutura de computação distribuída.

7 MECÂNICA DO PROCESSO DE PERMUTA MULTILATERAL NO COMPARTILHAMENTO DE SERVIÇOS EM REDE P2P

7.1 INTRODUÇÃO

A proposta deste trabalho, como mencionado anteriormente, é um estudo sobre o uso de redes *Peer-to-Peer* para o compartilhamento de serviços baseado no modelo de permuta multilateral em ambientes. Neste sentido, o presente capitulo apresenta os principais mecanismos necessários no processo de permuta apresentado nesse trabalho.

7.2 RELAÇÃO DE EQUIVALÊNCIA

O entendimento de como um computador funciona, em termos de hardware e software envolve conceitos de Matemática. A finalidade primeira do uso da Matemática em computação é a busca sistemática e rigorosa por padrões, partindo deum caso particular para o geral, especialmente quando tais padrões estão vinculados às mudanças circunstanciais ou ao contexto. Dentro desse contexto encontramos a Matemática Discreta que consiste no estudo de estruturas essencialmente discretas no sentido de não serem contínuas. A maioria dos objetos estudados contempla os conjuntos contáveis tais como os inteiros, grafos finitos, linguagens formais e relações.

Na Matemática Discreta, uma relação binária é uma correspondência existente entre dois conjuntos não vazios *A* e *B*. O conjunto *A* é denominado *conjunto de partida* e o conjunto *B* é denominado *conjunto de chegada*. A correspondência entre os dois conjuntos é dada em termos de pares ordenados, em que o primeiro elemento do par ordenado procede do conjunto de partida *A* e o segundo elemento do par ordenado procede do conjunto de chegada *B*. Os conjuntos de partida e de chegada não têm necessariamente que ter uma estrutura. Entretanto, segundo o tipo de estrutura que é sobreposta a esses conjuntos e o tipo de restrição que se impõe à própria relação, têm-se tipos especiais de relações, cada qual com um nome específico (WIKIPÉDIA, 2001). Em outras palavras, uma relação binária é definida como sendo um subconjunto do produto cartesiano entre dois conjuntos *A* e *B*. Isto é, uma relação *R* é um conjunto de pares ordenados. Um subconjunto de $A \times A$ pode ser chamado simplesmente de relação binária em *A*.

Nesse campo, uma relação de equivalência é a relação binária entre dois elementos de um conjunto em que os mesmos são equivalentes em alguma maneira. A relação de equivalência dá a noção de igualdade semântica, ou seja, de elementos que apresentam um mesmo significado. Relação de Equivalência é toda relação binária em um conjunto A que é, simultaneamente, *reflexiva*, *simétrica* e *transitiva*.

7.2.1 Recursos Equivalências

De maneira a flexibilizar a política de negociação dos participantes, o administrador do *peer* irá definir uma relação de equivalência para cada serviço que aquele irá disponibilizar na troca. Por serem heterogêneos e independentes, cada administrador do *peer* irá atribuir parâmetros e valores para as equivalências entre os serviços. De posse desses parâmetros, será efetuada a análise da viabilidade de troca dos serviços entre os *peers*. A Tabela 8apresenta um exemplo de como as equivalências são armazenadas.

Tabela 8 – Serviços Equivalentes do Peer A

UID	Service	ServiceQuantity	Equivalence	EquivalenceQuantity	Schedule
0326	Memória	1 Mb	Ciclos	1000	0h às 12hSeg e Qua
0258	Disco	1 Gb	Memória	256 Mb	10h às 11h Seg e Qua
0259	Disco	1 Gb	Memória	512Mb	0h às 7h Todos dias
0789	Fatorial	1	Disco	1 Gb	0h ás 23:59H Seg a Sex
0101	Memória	1Mb	Ciclos	1000	10h às 11h Seg e Qua

Esses parâmetros de equivalências são gerenciados pelo *Equivalence Manager* e ficaram armazenados no *Equivalent Services Repository*. Esse repositório irá armazenar o valor que cada recurso irá ter em relação a outro. Por exemplo, o administrador do Peer A (Tabela 8) poderá definir que 1000 ciclos de processamento serão equivalentes a 1 MB de memória RAM por hora.

O Equivalent Services Repository é o local para cada *peer*. Sendo assim, administradores diferentes poderão atribuir valores diferentes de equivalência para seus recursos. O administrador tem a possibilidade de definir, para o mesmo recurso, valores diferentes de equivalência de acordo com certas situações como, por exemplo,

finais de semana e início do mês. O mecanismo de equivalência admite que o mesmo recurso possua valores de diferentes equivalências em diferentes períodos do dia. Para cada período do dia, o administrador pode determinar um valor de equivalência que dependerá do comportamento que ele pretende com a negociação. Sendo assim, o administrador pode definir uma parametrização específica para o caso em questão no próprio registro de equivalência. Ou ainda, definir uma parametrização mais genérica para o período ou para o tipo do dia. Dessa forma, no momento de negociar com um consumidor, a componente *Trading Manager* busca o valor de equivalência que se enquadre aos parâmetros que foram requisitados.

7.2.2 Determinação de Equivalência Conforme Padrões de Demanda

A demanda por um recurso, assim como a carga local de um provedor, pode variar em função de alguns fatores como dia da semana, dia do mês, período do dia, entre outros. Portanto, é interessante que o valor de equivalência de um determinado recurso também varie conforme o dia ou horário.

Para ajustar a equivalência, o EM analisa, para cada um dos registros de equivalência, o comportamento da oferta, da demanda e da utilização do recurso (percentual de utilização da faixa disponibilizada para compartilhamento) nos diversos tipos de dias, nos quais o período apresenta demanda padrão. Quando um recurso está sendo totalmente utilizado, em um determinado período, sua taxa de utilização será cem por cento, o que causa um aumento do valor da equivalência desse recurso nesse período. Com esse aumento da equivalência, os consumidores que podem utilizar o recurso em outro horário tendem a procurar períodos mais baratos. Quando um recurso não está sendo totalmente utilizado, sua taxa de utilização é menor que cem por cento, o que causa uma queda do valor da relação de equivalência.

Por meio de consultas ao *Historical Repositoy(HR)*, o EM identifica se houve crescimento ou queda da demanda pelo serviço, o que faz com que o valor de referência da equivalência seja aumentado ou diminuído, respectivamente. De forma semelhante, o comportamento da oferta também é verificado, pois se ela aumenta, a equivalência tende a diminuir e, se essa oferta diminui, a mesma aumenta. É considerada, ainda, a relação entre a oferta e a demanda, de maneira que se a oferta for maior que a demanda, a equivalência diminui e se a demanda for maior que a equivalência, aumenta.

O processo para determinar essa demanda é efetuado, periodicamente, pelo componente *Equivalence Manager(EM)*.Para estabelecer a regra de identificação de

padrões para um período, o administrador informa ao sistema qual é o percentual máximo do desvio padrão para que um período tenha seu comportamento considerado padrão. Para o cálculo da demanda média, são consideradas as demandas dos últimos *n* dias. Para o cálculo do desvio padrão são consideradas a demanda média e as demandas dos últimos *n* dias.

Por exemplo, para a análise da demanda nas segundas-feiras, o EM busca no *Historical Repositoy(HR)* qual foi a demanda registrada para o período nas *n* últimas segundas-feiras. O valor *n* é definido pelo administrador do provedor. Com base nos dados levantados, o serviço EM efetua o cálculo estatístico (média aritmética de demanda e do desvio padrão).Sendo assim, o valor de equivalência deve ser proporcional à demanda e à taxa de utilização, e inversamente proporcional à oferta.

O valor de equivalência de um serviço em relação a outro pode influenciar na sua utilização já que consumidores tenderão a escolher entre os serviços que atendem seus requisitos e que possuem a melhor relação de Equivalência.

7.3 MECANISMOS DE CONFIANÇA E REPUTAÇÃO

Os sistemas P2P baseiam seu funcionamento em um importante fundamento: a cooperação entre os *peers* da rede, compartilhando os mais variados tipos de recursos e serviços. Entretanto, estudos como os de (ADAR e HUBERMAN, 2000) e (SAROIU, GUMMADI e GRIBBLE, 2002)demonstraram que boa parte dos usuários não obedece a esta premissa. É corriqueira a presença dos chamados usuários egoístas, que usam recursos de outros *peers* da rede e, no entanto, limitam ou impedem o acesso aos seus recursos. Também existem os *peers* maliciosos, que usam a rede apenas para prejudicar outros usuários como, por exemplo, disponibilizando arquivos infectados, corrompidos ou de conteúdo falso em uma rede de compartilhamento de arquivos.

A busca pela solução destes problemas levou ao desenvolvimento de diversas propostas de mecanismos de incentivo à cooperação. Uma das principais linhas de pesquisas explora o uso dos conceitos de confiança e reputação. A ideia é que cada *peer* tenha seu comportamento avaliado pelos outros *peers* da rede com os quais interagiu e desenvolva, ao longo do tempo, uma confiança e uma reputação.

As propostas de incentivo à cooperação baseadas em confiança e reputação podem ter duas arquiteturas: centralizada ou descentralizada. Na arquitetura centralizada, existe uma entidade central (*central authority*) responsável por calcular, manter e publicar estes conceitos de cada um dos peers que compõem a rede.

Na arquitetura descentralizada, cada *peer* da rede mantém históricos de avaliações geradas a partir de suas experiências com outros *peers*. Uma avaliação é uma nota dada pelo *peer* que requisitou algum serviço/recurso, ao comportamento do *peer* que o atendeu. Estas informações são, usualmente, conhecidas por "confiança direta". O cálculo da reputação pode ser baseado somente em informações da confiança direta. Entretanto, em uma rede com muitos *peers* como, por exemplo, em uma rede P2P de compartilhamento de arquivos, é comum a situação em que um *peer* deseja interagir com outro com quem nunca interagiu ou com quem teve poucas experiências, ou seja, o peer tem nenhuma ou pouca informação. Por esse motivo, os *peers* trocam experiências entre si. As informações recebidas de outros *peers* são comumente chamadas de "reputação".

As propostas existentes baseadas em confiança e reputação de arquitetura descentralizada se diferenciam, principalmente, pela escolha do método matemático usado para o cálculo da reputação. Algumas propostas optam pela simplicidade, usando métodos como uma média simples enquanto outros trabalhos propõem o uso de métodos mais complexos como, por exemplo, a teoria de Dempster-Shafer (YU e SINGH, 2002). A adoção de um mecanismo para contornar o problema de mau comportamento necessita de um bom entendimento das vantagens e desvantagens de cada método. Aspectos de convergência, robustez e segurança são extremamente importantes e devem ser bem conhecidos.

7.3.1 Modelo Adotado

Os mecanismos de confiança e reputação são utilizados para coletar informações sobre os comportamentos dos usuários durante suas interações e para utilizar a agregação desses dados para derivar um escore de reputação. Esse valor pode auxiliar os usuários a decidirem se devem ou não confiar na outra parte no futuro. O efeito natural desse mecanismo é incentivar o bom comportamento dos indivíduos, além de proporcionar uma melhoria na qualidade do serviço. Ele tem sido utilizado para garantir um comportamento colaborativo entre os *peers* finais em projetos de computação distribuída, que são os precursores de aplicações em grade. É utilizado, também, para aumentar a confiança e o desempenho de sociedades virtuais. Diferentes modelos de reputação descentralizados foram propostos baseados em interações entre *peers*.

Com o objetivo de avaliar os *peers*, os mecanismos de reputação possuem um papel importante nas redes P2P permitindo que todos os *peers* que habitam o sistema

possam identificar aqueles que não estão se comportando como deveriam, os chamados *peers* defeituosos ou mal intencionados e, desta forma possam inibir a proliferação de violações que podem levar o sistema ao caos (GRIZARD, VERCOUTER, *et al.*, 2006).

A confiança é baseada no comportamento passado ocorrido nas negociações entre os peers. Caso não exista esse histórico é utilizado apenas à reputação para avaliar a conduta entres os participantes do processo de negociação.

Em nosso trabalho, utilizamos como base um modelo de reputação que inclui um modelo de confiança adaptativo para quantificar e comparar a confiança de *peers* baseado em um sistema de transações com *feedback*, e uma implementação descentralizada de tal modelo em uma rede P2P.

A reputação de um *peer* determina a possibilidade que este possui de ter suas requisições atendidas, e quanto maior a reputação, maior a chance de sucesso. Dessa maneira, os *peers* possuem um incentivo claro em se mostrarem confiáveis e manter a maior e melhor reputação possível. A reputação de cada *peer* é construída baseada nas *transações diretas (confiança)*com o *peer* e baseada nos *testemunhos(reputação) de outros peers* sobre ele, conceitos que detalharemos a seguir.

Cada *peer* pode estar associado a uma ou mais classes de serviços. Um *peer* associado a uma dada classe de serviço pode apresentar um comportamento diferente quando associado a outra classe de serviço. Consequentemente, esse *peer* possuirá valores de reputação diferentes para cada classe de serviço à qual está associado.

7.3.2 Confiança Direta

Iremos definir como confiança a experiência direta que um *peer* u possuia respeito da confiabilidade de um determinado serviço de um *peer* v, baseado nas transações ocorridas diretamente entre eles. Conceitualmente, sempre que u tenta fazer uso do serviço de um v, ele atualiza sua confiança utilizando experiência diretas com v, aumentando-a se $v$concretiza a negociação e diminuindo-a em caso contrário.

Formalmente, confiança pode ser definida da seguinte maneira:

Seja TE_{uv}^{t} o número de transações efetuadas com sucesso e NTE_{uv}^{t} o número de transações efetuadas sem sucesso entre os *peers*u e v,e t um período de tempo em que ocorreram as transações. Seja $PeerConf_{uv}^{t}$ definida como sendo a confiança que o *peer* u possui em relação ao *peer* v em um tempo t, onde $0 \leq PeerConf_{uv}^{t} \leq 1$, mostrada na equação (1).

$$PeerConf_{uv}^{t} = \frac{TE_{uv}^{t}}{TE_{uv}^{t} + NTE_{uv}^{t}} \quad (1)$$

Seja $Rating_{usv}^{t'}$ o valor da avaliação do serviço s do *peer* v feito pelo *peer* u, no tempo t', em que $0 \leq Rating_{usv}^{t'} \leq 1$.E, $NRating_{usv}^{t}$ o número de avaliações efetuadas no período t. Seja $SConfDir_{usv}^{t}$ a Confiança direta de um *peer* u em relação a um serviço s de um *peer* v,em um tempo t,em que $0 \leq SConf_{usv}^{t} \leq 1$, mostrada na equação (2).

$$SConfDir_{usv}^{t} = PeerConf_{uv}^{t} * \left(\frac{\sum_{i}^{NRating_{usv}^{t}} Rating_{usv}^{t'}}{NRating_{usv}^{t}} \right) \quad (2)$$

Como podemos notar em (1), a confiança direta de um *peer* em relação aos outros irá variar com o passar do tempo. A equação é simples e baseada nas transações e nas avaliações efetuadas entre os *peers* u e v. Ela leva em consideração um período de tempo t que é estabelecido pelo administrador do *peer*. Podemos notar ainda em (2), que a confiança direta de um determinado serviço depende tanto da confiança direta do peer ao qual o serviço pertence quanto da avaliação que esse serviço possui.

Ao receber serviço de outro elemento da rede, cada *peer* registra o fato, melhorando sua avaliação sobre o *peer* que prestou o serviço. Essa informação pode, então se propagar pela rede e o *peer* provedor passa a ter melhores chances de obter serviço no futuro, o que se torna um incentivo para que os participantes forneçam serviço para a rede.

Após um *peer* utilizar um serviço de outro, é atribuído ao serviço desse *peer* uma avaliação $Rating_{usv}^{t'}$ pela utilização do mesmo. De acordo com as interações que um *peer* possui, pode ser gerado um indicador sobre possíveis ações que um *peer* pode realizar no futuro.

7.3.3 Reputação

Para nosso propósito, iremos definir reputação como sendo os testemunhos que um *peer u* possui a respeito da confiabilidade do *peer v*. Sempre que *u* tenta fazer uso dos recursos de um *v*, e u ainda não efetuou nenhuma transação direta com v, o *peer* u solicita informações(testemunhos) a respeito de v a todos os outros *peers* com os quais ele já efetuou algum tipo de transação. Serão consultados todos os *peer s*em que u já tenha computado uma confiança direta comeste.

Formalmente, reputação pode ser definida da seguinte maneira:

Seja Rp_{uvs} a reputação de um serviço s deum *peer*v,em que $0 \leq Rp_{usv} \leq 1$. Seja $PeerConf_{up}^{t}$ a confiança do *peer* u em relação ao *peer* p,em que p é um *peer* do qual u já tenha computado uma confiança direta, e $TrustFinalValue_{psv}^{t}$ a confiança final do *peer*$p$em relação ao serviço s pertencente ao *peer* v, mostrado no item 7.3.4. Seja $TotPeer$ o número total de *peer's* p . Rp_{uv} é definida como (3):

$$Rp_{usv} = \frac{\sum_{z=1}^{TotPeer} \left(PeerConf_{up}^{t} * TrustFinalValue_{psv}^{t} \right)}{TotPeer} \quad (3)$$

É importante lembrar que em (3), só será computado o testemunho existente em p. Isto quer dizer que mesmo que u possua confiança direta com p, esse deverá ter algum tipo de informação a respeito de v para que o testemunho possa ser computado.

Os valores de confiança são computados de forma global e recente e comparados. A ideia é que uma boa reputação seja difícil de ganhar, leve tempo para construir, porém possa ser destruída rapidamente após poucas transações incorretas.

7.3.4 Valor Final da Confiança

Também para nosso propósito, iremos definir $TrustFinalValue_{usv}^{t}$ como sendo o valor final de confiança de um *peer*u em relação a um serviço s de *peer*v, definido abaixo:

$$TrustFinalValue_{usv}^{t} = \left(\delta \times SConfDir_{usv}^{t} \right) + \left(\sigma \times Rp_{usv} \right) \quad (4)$$

Em que δ e σ possuem valores diferenciados e correspondem ao peso que a confiança direta e a reputação, respectivamente, terão no valor final da confiança. Esses pesos deverão seguir as seguintes características:

$$\begin{cases} \delta - \sigma \geq 0 \\ \delta + \sigma = 1 \\ 0 \leq \delta, \sigma \leq 1 \end{cases}$$

Essa definição de Valor Final de Confiança foi apresentada em (WANG, LU e DUAN, 2005). Entretanto essa foi apresentada apenas para peer. Em nossa proposta, a utilizamos na definição de confiança de recurso, também.

7.3.5 Implementação

Para implementação do modelo de confiança utilizado nessa proposta, cada *peer* possui um gerente do mecanismo de reputação representado pelo componente *Trust Manager*. Esse componente é responsável pela avaliação de confiança e reputação por meio das informações contidas no *Trust Repository*. O *Trust Manager* executa entre outras tarefas, a avaliação do nível de confiança e reputação de um *peer* e serviço/recurso desse *peer*. Esse trabalho é executado em dois passos: primeiro, ele coleta informações de confiança e reputação sobre o *peer* e serviço/recurso em questão, e então, calcula o valor final da confiança.

Para evitar problemas de segurança relativos ao armazenamento e transmissão das informações de confiança, emprega criptografia com chaves públicas/privadas. Cada *peer* é obrigado a ter um par de chaves e assinar suas mensagens de *feedback* com sua chave privada, e fornecer a chave pública, garantindo a integridade e autenticidade.

7.4 CERTIFICADOS DE PROPRIEDADE

O modelo de troca tem a característica da permuta da propriedade dos serviços entre os negociantes. Nesse modelo arquitetural, como estamos referenciando os serviços computacionais, é efetuada a troca dos certificados de propriedade desses serviços, pois na maioria dos casos, os serviços não são móveis (exemplo: ciclo de

CPU). Esse certificado será utilizado por ambos os participantes como forma de identificar o proprietário do recurso.

Os certificados de propriedade assemelham-se muito ao título ao portador, que são aqueles que não expressam o nome da pessoa beneficiada. Têm como característica a facilidade de circulação, que se processa com a simples tradição (COELHO , 2000). Os títulos ao portador, segundo a legislação brasileira são uma classificação de títulos de créditos. Título de crédito é o documento necessário para o exercício do direito, literal e autônomo, nele mencionado.

Os elementos fundamentais para se configurar o crédito decorrem da noção de **confiança** e **tempo**. A confiança é necessária, pois o crédito se assegura numa promessa da disponibilidade do serviço negociado. A temporalidade é fundamental, visto que subentende-se que o sentido do crédito é, justamente, a disponibilização futurado serviço combinado.

Os certificados de propriedade propostos nessa arquitetura, será capazes de identificar as características dos serviços, principalmente o *p*eer em que o mesmo se encontra. Eles serão criados no momento de sua disponibilização e somente o *peer* que os criou será capaz de determinar a sua autenticidade.

7.5 CONSIDERAÇÕES FINAIS

Os componentes do modelo proposto serão utilizados no projeto e construção deum *microkernel* que trate os detalhes de políticas para criar uma rede computacional de troca de serviços entre usuários, gerenciar o compartilhamento de serviços computacionais, e controlar a hospedagem colaborativa de conteúdos sobre os recursos computacionais de armazenamento, a partir da replicação de conteúdos e o controle de réplicas dos mesmos. Por fim, a partir da prova e conceito do modelo arquitetural, pretende-se desenvolver esse *microkernel* para oferecer APIs para que ferramentas de manipulação de dados possam ser construídas, sem que desenvolvedor tenha que se preocupar com as políticas do ambiente distribuído e de como os conteúdos compartilhados serão hospedados nos serviços computacionais.

8 INFRAESTRUTURA COMPUTACIONAL DISTRIBUÍDA PARA COMPARTILHAMENTO DE RECURSO EM AMBIENTE P2P

8.1 INTRODUÇÃO

No Capítulo 6 foi apresentada uma proposta de arquitetura software, na qual foram levantados os elementos arquiteturais necessários para a construção de *microkernel* cuja finalidade é o auxílio no desenvolvimento de aplicações distribuídas. Estas aplicações tiram proveito de recursos compartilhados em redes *peer-to-peer*, quando estes se conectam em um ambiente computacional criado sobre uma rede *peer-to-peer* não estruturada. Para isto, a arquitetura de software deve oferecer serviços básicos cuja finalidade é tirar proveito de recursos locais e distribuídos.

Nesse sentido, são necessários mecanismos que viabilizem o processo de permuta multilateral dentro da computação distribuída quando estes formam uma rede *peer-to-peer*, a fim de colaborarem uns com os outros. Após ter sido formada, uma rede *peer-to-peer* deve permitir a permuta e o compartilhamento de recursos computacionais além de executar serviços de aplicações de maneira distribuída por meio de objetos móveis.

Para criar uma infraestrutura computacional distribuída que permita a satisfação dos requisitos de redes *peer-to-peer*, é necessário um conjunto de algoritmos de rede *peer-to-peer que* satisfaçam requisitos como: organização da infraestrutura de computação *peer-to-peer*; desconexão programada de *peers*; descoberta de *peers*; roteamento de mensagens entre os *peers*; e sincronização de informações de troca do ambiente distribuído. Tanto as políticas de computação distribuída quanto os algoritmos de rede *peer-to-peer* dependem de estruturas de dados, nas quais informações de controle da infraestrutura de computação *peer-to-peer* e contextos de rede *peer-to-peer* são mantidas.

A proposta apresentada por esta tese baseia-se em uma rede *peer-to-peer* não estruturada, onde *superpeers* e *defaultpeer* se conectam uns aos outros, formando uma infraestrutura computacional distribuída e descentralizada. O modelo conceitual desta infraestrutura de computação é apresentado no diagrama de classes exibido pela Figura 20.

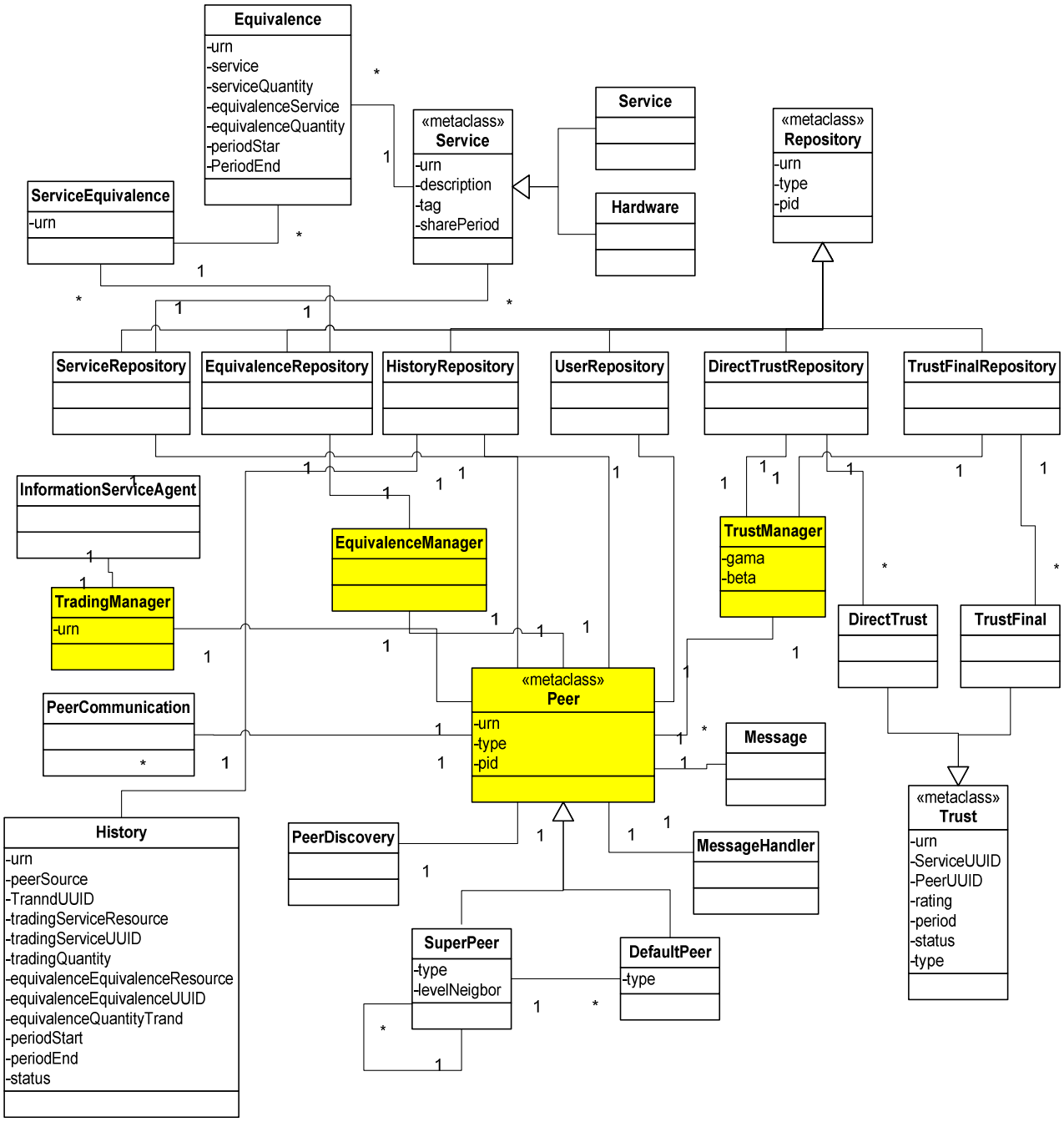

Figura 20 - Classes de Domínio

Para organizar a formação desta infraestrutura de computação, devem ser respeitados os limites e regras para construção das conexões entre os *peers*. Os limites de conexões definem o máximo de conexões entre os tipos de *peers* definidos para a organização do sistema distribuído. Assim, o valor configurado em *maxConnectionSuperPeers* define um número máximo de vizinhos *superpeers* que um *superpeer* pode suportar. Já o valor configurado em *maxConnectionsSuperpeerDafaultPeer* define o número máximo de vizinhos *defaultpeers* que um *superpeer* pode suportar. Para este trabalho, foram adotados os

valores onde um *superpeer* pode ter até o valor definido pela dimensão do hipercubo (item 2.6.6) de vizinhos *superpeers*. Por exemplo, caso tenhamos a dimensão do hipercubo igual a 3(três), esse também será o valor máximo de vizinhos suportados onde termos também no máximo 8(oitos) *superpeers* $(2^3 = 8)$. Adotamos para o no nosso modelo o valor máximo de *10* vizinhos *defaultpeers* para cada *superpeer* e cada *defaultpeer* só poderá estar ligado a um único *superpeer*.

De maneira geral, todo *peer* é identificado de maneira única, de modo que uma mensagem possa ser enviada a partir de uma origem e ser entregue em um destino. A identificação única dos *peers* também ajuda na configuração das informações de vizinhança dos *peers*. Além disto, cada *peer* monitora, quando conectado à rede *peer-to-peer*, valores de relógio do computador do usuário, de modo que esta informação seja utilizada em um processo de sincronização de informações de contexto no sistema distribuído. Por fim, qualquer *peer* usa endereços para enviar e receber mensagens. Cada endereço é configurado usando um protocolo de transporte, endereço IP e porta de comunicação.

As informações de vizinhança são mantidas pelos *superpeers* por duas tabelas de vizinhos: uma de *defaultpeer* e outra de *superpeer*. Cada vizinho *superpeer* mantém as informações sobre *peers* conectados diretamente a ele. Tais informações consistem no identificador único, tipo e lista de endereços do *peer* vizinho. Um vizinho *superpeer* também é capaz de manter informações cuja finalidade é auxiliar o processo de encaminhamento de mensagens enviadas de um *peer* para outro. Estas informações compõem uma tabela de rotas, onde cada uma de suas entradas está relacionada a um *peer* conectado à rede *peer-to-peer*. Além disto, a cada entrada na tabela de roteamento está associado um conjunto de rotas, e cada rota mantém o identificador único do *peer* de destino, o caminho a ser percorrido por uma mensagem e o custo de envio de uma mensagem sobre o caminho configurado na rota.

Quando um *peer* deseja se comunicar com outro que esteja conectado à rede *peer-to-peer*, uma mensagem é criada, contendo identificador único, rótulo, origem, destino, saltos, caminho e parâmetros. As mensagens devem conter identificadores únicos, pois esta informação é utilizada para controlar o número de mensagens encaminhadas de um *peer* para outro, dependendo do algoritmo de rede *peer-to-peer* que estiver sendo executado em um determinado instante. O rótulo auxilia no processo de identificação e execução dos algoritmos de rede *peer-to-peer* ou na execução de mecanismos específicos da camada de comunicação. A Tabela 9 apresenta os descritores das mensagens definidas para este trabalho.

Tabela 9 - Mensagens definidas para o sistema distribuído

Mensagem	Descrição
LISTSPEERS	*Defaultpeer* solicita a lista de *superpeer* para conexão
INSERTPEER	*Defaultpeer* solicita conexões a fim de criar entradas na tabela de vizinhos do *superpeer*.
INSERTSPEER	*Superpeer* solicita conexões a fim de criar entradas na tabela de vizinhos de outro *superpeer*
SUPERPEER	*Superpeer* sinaliza o aceite da inclusão do *defaultpeer* em sua tabela de vizinhos
REPLY	Sinaliza o aceite de um pedido de conexão.
ERROR	Rejeita um pedido de conexão.
PEERFULL	Informa que o *superpeer* não possui mais conexões válidas tanto para outro *superpeer* quanto para o *defaultpeer*
PEEREXIT	Desconecta um *peer*da rede *peer-to-peer* de maneira programada.
FIRSTSP	Solicita que o *peer* seja elevado à condição *de superpeer*
UPDATEPEERLEVEL	Solicita aos *superpeer* vizinhos a mudança do nível de conexão conforme o algoritmo do Hipercubo
RESQUETTRUSTFINAL	Solicita o *Trust Final* do serviço a ser negociado
RESPONSETRUSTFINAL	Devolução de resposta em função do tratamento requisição de solicitação de *TRUSTFINAL* e informa o *Trust Final* do serviço.
TRADINGSP	*Defalutpeer* submete uma requisição ao *superpeer* para criação deuma negociação de troca.
TRADINGCH	*Superpeer* submete uma requisição aos *defaultpeer* .
TRADINGST	Devolução de resposta em função do tratamento requisição de criação de uma negociação de troca.
TRADINGCP	Sinaliza que a negociação de troca foi completada informando o *peer* com o qual a troca deverá ser efetuada
TRADINGOC	Sinaliza da troca dos certificados de propriedade
TRADINGSN	Sinaliza que a negociação de troca não foi completada

8.2 ALGORITMOS DE REDE *PEER-TO-PEER*

Antes de apresentar as políticas de computação distribuída baseadas em redes P2P, é necessário apresentar a forma como a infraestrutura do sistema distribuído é organizada. Por isto, esta seção explica o funcionamento dos algoritmos responsáveis pelas seguintes funções: estruturação da rede *peer-to-peer*; e desconexão programada de *peers.* Todos os algoritmos apresentados nesta seção compõem os mecanismos da *Communication Layer* apresentados na seção 6.3.4. Por fim, os detalhes relacionados à autenticação e autorização no ambiente distribuído, onde cada *defaultpeer* necessita utilizar informações de segurança mantidas pelos seus vizinhos *superpeers*, são tratados pelo mecanismo de P2P Communication.

8.2.1 Estruturação da Rede *Peer-to-Peer*

Para criar uma infraestrutura de computação distribuída, optou-se, nesta tese, pelo uso de redes *peer-to-peer* não estruturadas, pois este tipo de rede oferece maior autonomia para ambientes de computação distribuída, uma vez que sua organização não fica limitada a uma topologia específica. Neste caso, não é necessária a intervenção do usuário, de modo que este tenha que reconfigurar toda a rede *peer-to-peer* quando esta receber um novo *peer*. Esta limitação é inerente às redes *peer-to-peer* estruturadas, pois estas são formadas a partir de configurações pré-estabelecidas por algum administrador de sistemas como, por exemplo, a quantidade total de *peers* e os identificadores de cada *peer*. Apesar disto, redes *peer-to-peer* estruturadas, comparadas às não estruturadas, impõem uma sobrecarga menor, gerando um número menor de mensagens.

Para reduzir este problema, este trabalho utiliza o conceito de *superpeers* cujos objetivos são manter índices de metadados, comuns à rede *peer-to-peer*, e atuar como intermediários na comunicação entre *defaultpeers*. Para isto, é necessário, em primeiro lugar, formar uma rede entre *superpeers*, a fim de que um *superpeer* possa ter rotas para os demais *superpeers* da rede *peer-to-peer*. Complementar a isto, é necessário que *defaultpeers* se conectem a um *superpeer* com o objetivo de completar o processo de formação da infraestrutura de rede. Para formar a rede *peer-to-peer*, este trabalho utiliza um algoritmo de rede *peer-to-peer* baseado nas especificações do *HyperBone* (BONA, 2006) e do *PeerCube* (ANCEAUME, BRASILEIRO, *et al.*, 2008).

Apesar de existir uma distinção entre os *peers* formadores da infraestrutura computacional do ambiente distribuído, o algoritmo para estruturação da rede *peer-to-peer* é o mesmo para *superpeers* e *defaultpeers*.

O processo de formação de uma rede *peer-to-peer* inicia-se quando um *peer* executa algoritmo *conectar.* Por meio deste algoritmo, solicita a lista de *superpeer* enviando a mensagem *LISTSPEERS*. De posse dessa lista, o *peer* percorre as entradas da mesma, de modo que seja enviada, de maneira síncrona, uma solicitação para se conectar à rede *peer-to-peer* através do *superpeer*. Este envio de mensagem é interrompido somente quando a iteração da lista acaba ou uma conexão entre o *peer* e *superpeers* é atingida. *O peer* envia uma mensagem *INSERTPEER p*ara o primeiro *superpeer* da lista*.* Ao receber esta mensagem, um *superpeer* verifica o número máximo de conexões permitido para o tipo de *peer* que a enviou. Se houver disponibilidade, o *superpeer* obtém o identificador, tipo e lista de endereços do *peer* que solicitou a conexão. Com tais informações, uma entrada na tabela de vizinhos é

criada e, em seguida uma mensagem *REPLAY* é enviada de volta. Caso não haja disponibilidade, o *superpeer* envia uma mensagem *PEERFULL*. Quando um *peer* recebe uma mensagem *REPLAY*, o mesmo tem a confirmação de seu pedido de conexão com a rede *peer-to-peer* por meio de um *superpeer*. Assim, o *superpeer* obtém o identificador, tipo e lista de endereços do *peer* que solicitou a conexão. Com tais informações, uma entrada na tabela *defaultpeers* de vizinhos é criada, finalizando o processo de conexão com a rede *peer-to-peer* por meio de um *superpeer*. No entanto, se um *peer* recebe uma mensagem *PEERFULL*, o processo de conexão com o *superpeer* emissor desta mensagem é cancelado e, em seguida, uma nova solicitação de conexão é feita para outro *superpeer*.

O mesmo processo é utilizado pelo *superpeer* no momento em que o mesmo deseja entrar na rede *peer-to-peer*. Entretanto, nesse caso, a tabela de *superpeer* vizinho é percorrida para efetua a conexão entre os *superpeers* e a mensagem *INSERTSPEER* que é enviada como solicitação de inclusão naquela tabela.

Uma vez que as tabelas de vizinhança tenham sido construídas nos super*peers*, é de suma importância que cada *peer* consiga alcançar os demais *peers* conectados à rede *peer-to-peer*. Para isto, é necessário que uma rede *peer-to-peer* seja conexa, fazendo com que os *peers* envie suas mensagens e que estas possam ser entregues a um destino com o menor custo possível.

Uma vez construídas, tabelas de vizinhanças são utilizadas no envio de mensagens para *peers* que não são vizinhos imediatos de um emissor de mensagem. Neste caso, mensagens percorrem no máximo n de saltos até chegar ao seu destino, sendo n o número da dimensão do hipercubo (BONA, 2006). Porém, quando ocorre o processo de compartilhamento de serviço através do modelo permuta multilateral e após a descoberta de *peer* que estejam interessados na negociação, as mensagens são trocadas entres estes *peers*.

8.2.2 Desconexão de *Peers*

Durante o envio ou a passagem adiante de uma mensagem, um *superpeer* pode lidar com problemas de indisponibilidade de algum vizinho imediato, causados por uma queda de conexão, falha de equipamento, queda de energia, entre outros. Nesta situação, em que um vizinho sai abruptamente do sistema distribuído, o *superpeer* remove a entrada correspondente ao vizinho indisponível da lista de vizinhos e, em seguida, tenta enviar ou passar adiante a mensagem. Sempre que um *peer* (seja ele um *superpeer* ou um *peer* simples) perde algum vizinho, ele imediatamente tenta se conectar a outro vizinho. Isto é feito para evitar que o grafo de

rede *peer-to-peer* se parta, causando a indisponibilidade de recursos e serviços oferecidos pelos vários *peers* pertencentes a este grafo. Portanto, quando um *peer* se conecta a outro, no caso outro *superpeer*, a fim de suprir a ausência de um vizinho, uma nova descoberta de *peers* é feita e, consequentemente, as informações de roteamento são atualizadas.

Contrário ao contexto apresentado acima, um *peer* pode se desconectar da rede *peer-to-peer* de maneira programada. Neste caso, todos os vizinhos do *peer* que está para se desconectar da rede *peer-to-peer* são notificados. Quando a mensagem *PEEREXIT* é recebida por um *peer*, é verificado se existe um vizinho cujo identificador é igual à origem da mensagem. Se verdadeiro, a entrada na tabela de vizinhos é removida. Por se tratar de um algoritmo síncrono, uma vez que todos os vizinhos tenham recebido a notificação de desconexão e, consequentemente, removido a entrada na tabela de vizinhos correspondente à origem da mensagem, o *peer* que está se desconectando da rede *peer-to-peer* exclui todas as entradas em sua tabela de vizinhos.

8.2.3 Autenticação e Autorização

Nesta tese, a formação de redes s *peer-to-peer* depende de que usuários sejam criados no sistema distribuído, de modo que possa ser verificada a autenticidade dos mesmos, a fim de autorizar a execução de serviços e acesso a recursos compartilhados nas redes *peer-to-peer*. Autorização e autenticação são conceitos fundamentais no que diz respeito a controle de acesso. Tais conceitos são distintos, mas interdependentes, de forma que a autorização de acesso a um determinado recurso ou serviço é, na verdade, dependente da autenticação. Uma vez que a autenticação é o processo para determinar quem é o usuário no sistema, a autorização determina o que um usuário pode fazer.

Tratar questões de segurança em sistemas distribuídos, cuja infraestrutura é formada por uma rede *peer-to-peer*, é uma tarefa difícil, pois as informações de usuários devem ser confidenciais. Diferentemente de sistemas distribuídos cliente-servidor, onde informações de usuários são mantidas em um índice de dados centralizado, sistemas distribuídos *peer-to-peer* precisam oferecer meios para tornar informações de controle de acesso sempre disponíveis. Isto tem como objetivo permitir que o mecanismo de controle de acesso possa autenticar o usuário e, em seguida, autorizá-lo para acessar recursos e executar serviços no sistema distribuído. Além disto, quando informações de autenticação e autorização não estão disponíveis para o

mecanismo de controle de acesso, usuários podem ser impossibilitados de acessar recursos e executar serviços no sistema distribuído. Para lidar com este requisito, é necessário criar um índice distribuído de informações de autenticação de usuário, de modo que os *peers* possam manter cópias deste índice, permitindo que tais informações estejam disponíveis sempre que for necessário autenticar e autorizar o usuário. Porém, esta abordagem não satisfaz os requisitos de controle de acesso plenamente, pois ela não oferece garantias de integridade e confidencialidade dos dados do usuário, uma vez que informações de autenticação de usuários podem ser obtidas por qualquer usuário, pois as mesmas não estão sob algum tipo de controle administrativo.

Para tratar isso, esta tese concentra informações de autenticação nos *superpeers*, pelo fato de estes oferecerem suporte à cooperação entre os vários *peers* do sistema distribuído. Isto facilita o uso de técnicas para contabilidade de recursos compartilhados e autenticação de usuários do sistema. Então, os *superpeers* devem ser computadores que tenham alta disponibilidade e acesso restrito a pessoas autorizadas, quando se refere ao acesso a informações de segurança. Os *superpeers*, na forma em que estamos propondo, servirão aos *defaultpeers* normalmente, como em qualquer abordagem que separe entidades de redes *peer-to-peer* em *superpeers* e *defaultpeer*. Somente informações sigilosas, como aquelas utilizadas para autenticar e autorizar usuários, devem ser manipuladas por administradores de sistemas nos locais onde os *superpeers* estão fisicamente instalados. Ainda assim, existe um risco para o controle de acesso do sistema, pois arquivos de senhas ficariam expostos para os administradores.

Para evitar a violação de arquivos de senhas, este trabalho faz um *hash* irreversível dos nomes e as senhas de cada usuário do sistema. Para isto, é aplicado o algoritmo SHA-1 para criptografar o nome concatenado com a senha do usuário e, em seguida, armazenar-se o resultado da criptografia no arquivo de senhas. Com isto, não é possível descobrir os nomes e senhas dos usuários do sistema distribuído. Dessa forma, então, é possível fornecer um serviço básico de identificação e autorização para os usuários do sistema distribuído. No entanto, como pode ser visto na seção anterior, os *peers*, sejam estes *superpeers* ou *defaultpeers*, se conectam uns aos outros utilizando uma lista de *superpeers* confiáveis. Isto pode comprometer o processo de autenticação de usuário, uma vez que um *peer* nem sempre se conectará ao mesmo conjunto de *superpeers* confiáveis.

Nesse contexto, cada *superpeer* deve manter uma cópia atualizada das informações de autenticação de usuários. Porém, isto pode causar problemas de

segurança ao sistema distribuído, pois as réplicas podem ser interceptadas e os dados copiados. Quando o meio utilizado para transportar informações é um meio não seguro, a informação trocada entre duas ou mais entidades computacionais deve ser criptografada e assinada digitalmente, para que se assegure a integridade e confidencialidade das informações. Isso pode ser implementado a partir do protocolo SSL, cujo objetivo é fornecer um canal seguro, ou seja, com garantia de privacidade, autenticidade dos pares e integridade de informações. Ao estabelecer a conexão, o SSL estabelece um identificador de sessão e um conjunto de algoritmos criptográficos. O algoritmo para troca de chaves será um algoritmo de criptografia de chave pública que será utilizado para enviar uma chave privada do algoritmo de criptografia de dados. Assim, o SSL utiliza-se de um algoritmo assimétrico apenas para criar um canal seguro para enviar uma chave secreta, a ser criada de forma aleatória e que será utilizada para criptografar os dados utilizando-se de um algoritmo simétrico. O algoritmo simétrico é utilizado para efetivamente criptografar os dados oriundos de alguma aplicação cliente. Por fim, o algoritmo de inserção de redundância é utilizado para garantir a integridade da mensagem. Assim, cada usuário possui um par de chaves, pública e privada utilizado no funcionamento do SSL. A chave pública é disponibilizada a qualquer pessoa disposta a corresponder-se com o proprietário dopar de chaves. A chave pública pode ser usada para ler uma mensagem assinada com a chave privada ou para criptografar mensagens que só podem ser descriptografadas com a chave privada. A segurança das mensagens criptografadas assimetricamente depende da segurança da chave privada, que deve estar protegida contra uso não autorizado.

Apesar de este trabalho adotar uma infraestrutura computacional descentralizada, é necessário manter atividades administrativas que promovam a segurança do ambiente. Por isto, os usuários devem ser criados por pessoas responsáveis pela administração do sistema distribuído. No entanto, apesar de terem sido investigadas as formas de implementação de mecanismos de segurança como foi mostrado anteriormente, este trabalho não trata detalhes de segurança, ficando este aspecto para ser tratado em trabalhos futuros. Portanto, assume-se que cada usuário registrado no sistema distribuído tem suas informações de autenticação mantidas pelos *superpeers* conectados à rede *peer-to-peer.*

.

8.3 POLÍTICAS DE COMPARTILHAMENTO DE SERVIÇOS POR MEIO DE PERMUTA MULTILATERAL

Na seção anterior, apresentamos os algoritmos básicos da infraestrutura computacional criada por uma rede *peer-to-peer* não estruturada. Estes algoritmos, no entanto, não são suficientes. Deste modo, esta seção apresenta as políticas de computação distribuída que viabilizam o compartilhamento de serviços por meio do modelo econômico de permuta multilateral. Sobre estes compartilhamentos, é possível executar tarefas sobre tais compartilhamentos, a fim de processar dados que requerem computação intensiva. A políticas apresentada na próxima seção é um algoritmo distribuído assíncrono, que envia requisições e recebe respostas após as requisições terem sido tratadas nos *peers*. Por se tratar de algoritmos distribuídos de um nível de abstração maior que os algoritmos de rede *peer-to-peer* apresentados na seção anterior, as políticas de computação distribuída são apresentadas sem que sejam abordados detalhes de baixo nível como, por exemplo, reenvio de requisições, quando respostas não são devolvidas durante certo espaço de tempo. O tratamento deste tipo de detalhe é requisito básico para qualquer implementação de *middleware* para computação *peer-to-peer*.

8.3.1 Negociação por um serviço

Em uma rede baseada em modelos econômicos, é necessário que os consumidores e os provedores de recursos negociem de maneira a determinar a troca de um recurso.

Na arquitetura proposta na seção 6.3, quando um *defaultpeer* consumidor deseja executar uma aplicação utilizando um recurso remoto, ele invoca o componente *TradingManager (TM).*Esse componente é responsável pela condução do processo de troca. Esse como os outros componentes descritos nessa seção são apresentados na Figura 20 da seção 8.1.

Ao iniciar uma negociação, o *defaultpeer* consumidor envia mensagem *TRADINGSP*. Nessa mensagem, o *defaultpeer* informa o período de tempo necessário para executar o serviço e a quantidade do mesmo. Também envia a quantidade e o serviço que o *defaultpeer* consumidor irá disponibilizar na troca. A seleção do serviço, que será oferecido em troca, é feita por meio de uma análise que o componente *EquivalenceManager* executa junto ao *Equivalent Service Repositoy*. Ao selecionar esse serviço, a TM envia uma mensagem para a *Monitor* com a intenção de bloquear

o serviço durante a negociação. Com esse procedimento, o serviço não pode ser usado em outro evento de troca. Esses são os primeiros processos efetuado pelo *TradingManager*

Essa mensagem é enviada para o *superpeer* no qual o *defaultpeer* está conectado. Por sua vez, o *superpeer* envia a mesma mensagem, em primeira instância, para todos os membros de sua tabela de *defaultpeers* vizinhos com o rótulo de *TRADINGCH.* Em uma segunda instância, o *superpeer* envia a mesma mensagem, porém agora para todos os *superpeers* pertencentes a sua tabela de *superpeers* vizinhos que, por sua vez, enviam para todos os seus *defaultpeers* e *superpeers* vizinhos. Esta instância ocorre quando nenhum dos *defaultpeers* pertencentes à tabela de *defaultpeer* vizinho deseja participar da negociação ou quando é decorrido o tempo definido pela variável de controle *maxTime* que é o tempo de espera de resposta da solicitação.

Com base na informação enviada pelo *defaultpeer* consumidor e utilizando o compomente *EquivalenceManager* local, o *defaultpeer* provedor fará uma análise para determinar a viabilidade da troca. Ele irá examinar se o serviço oferecido na negociação é equivalente ao serviço solicitado. Esta análise é feita devido ao fato de que cada *peer* tem a liberdade de determinar a equivalência de cada um dos seus serviços (seção 7.2.1). Nela são levadas em conta as características de equivalência definida por cada *peer*. Nesse momento, também, é efetuada a verificação da confiança que o *peer* pode ter pelo serviço negociado. Nessa fase são enviadas e recebidas mensagens *RESQUETTRUSTFINAL* para todos os *peers* com os quais o *peer* tenha efetuado negociação. Essa mensagem é uma solicitação do valor *Trust Final,* apresentado na *seção 7.3.4,* do serviço a ser negociado. E em resposta, o *peer* receberá a mensagem a *RESPONSETRUSTFINAL* com o valor solicitado. O *defaultpeer* provedor, após a análise, envia uma resposta, por meio da mensagem *TRADINGST,* ao *defaultpeer* consumidor, em que informará se irá ou não aceitar a permuta.

O *defaultpeer* consumidor, após receber as respostas das solicitações feitas, também faz a mesma análise que o *defaultpeer* provedor para definir com qual irá realizar a negociação. Em sequência desta análise, o consumidor notifica o *defaultpeer* provedor selecionado também utilizando a mensagem *TRADINGST*. No entanto, um *defaultpeer* provedor pode ter participado em vários processos de negociação. Neste caso, quando receber as mensagens de aceitação pelos *defaultpeer* consumidores, ela cria uma árvore de decisão contendo todas as possibilidades de execução e escolhe a configuração que maximiza seus ganhos.

Após definir se o procedimento será executado ou não, ele envia uma mensagem para confirmar ou anular a proposta para os consumidores que são as respectivas mensagens *TRADINGCP e TRADINGSN.* Em caso de cancelamento, o *defaultpeer* consumidor poderá optar por usar o próximo provedor existente na lista.

Uma vez concluída a negociação, o *defaultpeer* consumidor e o provedor irão fazer a troca dos recursos, usando o certificado de propriedade do serviço. Este certificado será utilizado por ambos os participantes, a fim de identificar o proprietário do serviço. Um *peer* só será capaz de trocar os recursos que tem em sua posse. Neste sentido, o modelo permite que um *peer* possa negociar os serviços que estão localizados em outros pares. Esse procedimento é efetuado pela da troca de mensagem *TRADINGOC.*

Por meio do *Monitor* o sistema fornece aos participantes informações referentes à rede P2P. Os provedores registram seus serviços no *Monitor* de maneira que esses possam ser localizados pelos consumidores interessados em utilizá-los. O *Monitor* fornece, ainda, dados referentes à situação econômica da rede como, por exemplo, informações sobre demanda. A demanda é registrada em um *Monitor* no momento em que um consumidor efetua uma busca pelo serviço por meio do *ISA*. Toda vez em que uma negociação por um recurso é concluída com sucesso ou não, o *TradingManager* informa ao *Monitor* as características da negociação. Dessa forma, o serviço de informações mantém as informações dos serviços negociados.

Para facilitar a implementação do modelo, é necessário que todos os *peers* que desejem compartilhar seus serviços assumam um modelo único para publicação dos seus serviços.

A Figura 21 mostra o diagrama de sequencia de todo o processo de de negociação entre dois *peer* (Peer A e Peer B). As Figura 22 e Figura 23 mostram o processo de negociação no Peer A e no Peer B, respectivamente.

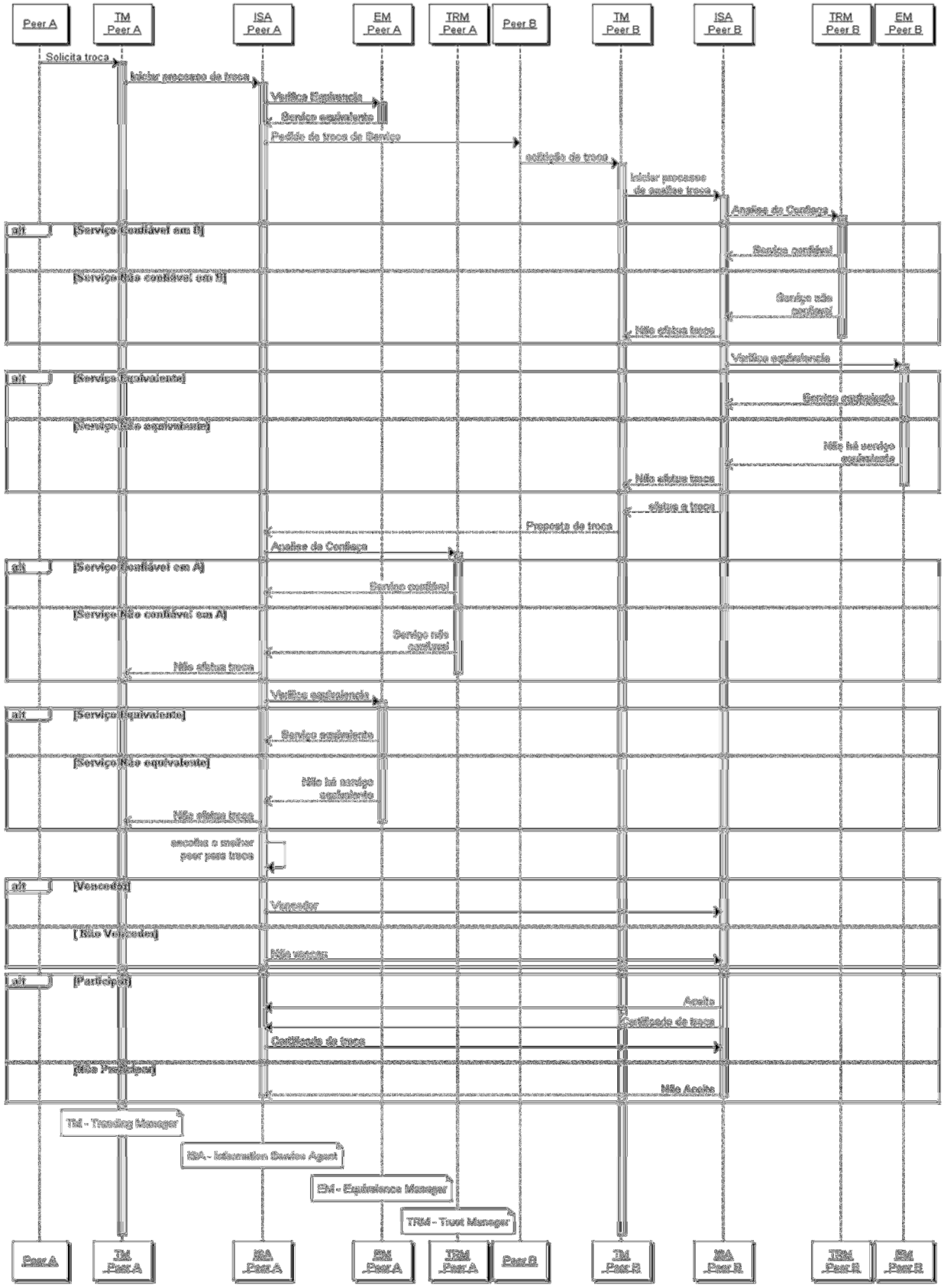

Figura 21 – Diagrama de Sequência de todo o processo de troca

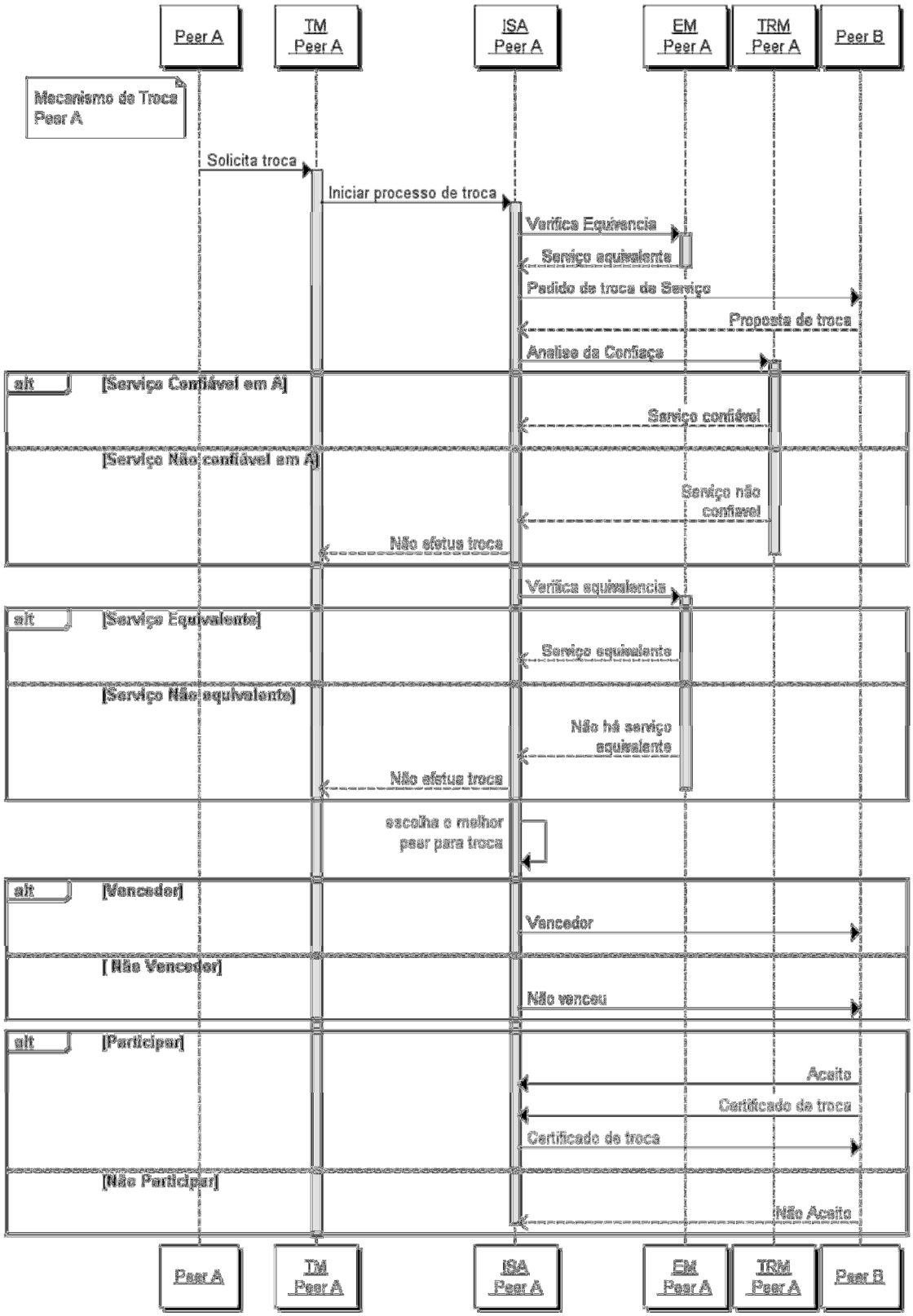

Figura 22 – Diagrama de Sequencia do processo de troca no Peer A

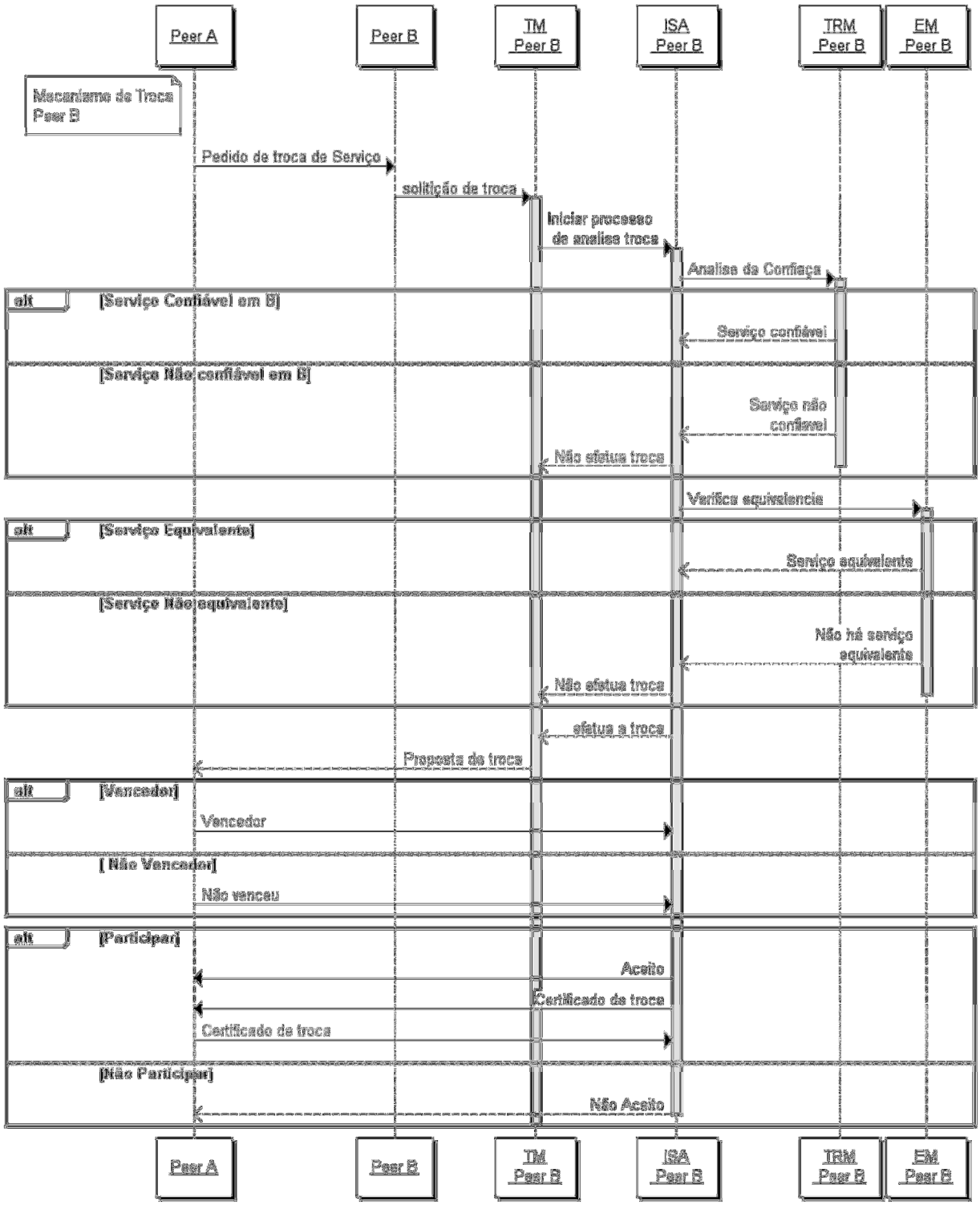

Figura 23 - Diagrama de Sequencia do processo de troca no Peer B

8.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a principal colaboração deste trabalho, que é a definição de políticas de computação distribuída baseadas em redes, de modo que estas possam permitir o compartilhamento de serviços utilizando o modelo econômico de troca

Para atingir os objetivos desta tese, é necessário avaliar experimentalmente os principais algoritmos de rede *peer-to-peer* e as políticas de computação distribuída baseadas em redes P2P, objetivando a obtenção de resultados que permitam considerar ou não o uso do modelo troca como uma alternativa para o compartilhamento de serviços em ambientes computacionais P2P.

9 SIMULAÇÕES E RESULTADOS

9.1 INTRODUÇÃO

Para a maioria das pesquisas em computação distribuída e, principalmente, em ambientes P2P, a execução é difícil e talvez até mesmo impossível, em diferentes cenários, na busca de uma avaliação de desempenho de modelo e algoritmos por meio de forma controlável. A gestão de recursos e as estratégias de programação, em larga escala, nesse contexto de desenvolvimento, necessita de uma estrutura simples para modelagem determinística e de simulação nas quais possam ser feitas avaliações dessas estratégias. Em sua maioria, os pesquisadores não têm acesso a uma infraestrutura adequada de teste devido ao fato de que construí-la é considerado processo caro e demorado. Além disso, mesmo para aqueles que têm acesso, o tamanho da mesma é quase sempre limitado a poucos recursos.

Diante dessas dificuldades, os pesquisadores reconhecem a importância e a necessidade da utilização de ferramentas para modelagem e simulação de ambientes com o intuito de provar a eficácia dos modelos propostos em suas pesquisas (BUYYA, ABRAMSON e GIDDY, 2000).

Com o constante desenvolvimento das novas tecnologias, administradores de redes ou responsáveis por novas implementações tecnológicas necessitam de métodos para avaliar as propostas de melhorias ou soluções de problemas trazidas por essas tecnologias. Existem três possibilidades de se avaliar, ou obter resultados: experimentação, método analítico e simulação.

A **experimentação** nada mais é do que a avaliação na prática do objeto a ser analisado. Este método tende a ser o mais preciso, porém nem sempre é possível de se realizar. Seguindo o exemplo deste trabalho, não seria possível obter uma estrutura de rede complexa o suficiente para a execução dos testes. Além do mais, seria bastante perigoso utilizar para este fim a estrutura de um provedor de serviço da Internet para implementação de um protocolo, correndo-se o risco de paralisação da rede ou instabilidade, ou mesmo o custo de novos equipamentos para suportar estes testes.

Outro método consiste em modelar o sistema, contendo as variáveis de interesse de forma que esse modelo possa ser tratado matematicamente. Com esse tratamento é possível obter diversos resultados de acordo com as situações desejadas. A definição das variáveis de interesse, no caso de um experimento em a nível QoS de rede seriam: atraso de pacotes, *jitter* ou performance da rede em si. Este

tipo de análise matemática é também conhecido como **método analítico** e é feita por meio de uma série de equações matemáticas e normalmente, implica um profundo conhecimento matemático para sua elaboração de forma a obter resultados precisos. Exigindo bastante tempo no desenvolvimento das fórmulas e cálculos e, dependendo do cenário a ser tratado, o processo torna-se extenuante, o que obriga à simplificação do cenário para a extração de resultados, o que pode acabar comprometendo o resultado final (PORTNOI, 2002).

Outro método que é usado para análise é por meio de **simulações**. É similar ao método experimental, mas ao invés de testar na prática, usa-se um programa como modelo . Ao contrário do método analítico, este oferece mais dinamicidade, por ser de mais simples aplicação. É possível jogar com as variáveis e obter resultados mais rapidamente. Portanto, permite cenários mais complexos com menos envolvimento matemático.

A utilização de simuladores vem crescendo, pois permitem o estudo e avaliação a custos reduzidos.

9.2 SIMULADOR

A simulação tem sido amplamente utilizada para modelagem e avaliação de sistemas do mundo real, a partir de processos de negócios, linha de montagem de fábrica e para projetar sistemas de computador. Assim, ao longo dos anos, modelagem e simulação surgiu como uma disciplina importante e, muitos padrões e ferramentas de aplicação específica e tecnologias foram construídos.

Nessa vertente e na busca de avaliar o modelo proposto nessa tese, foi desenvolvida uma ferramenta de simulação. Chamamos, assim, essa ferramenta de pydssim (*Python Distribute System Simulator*) (MOURA, 2009). O pydssim foi totalmente desenvolvido na linguagem Python em conjunto com a biblioteca SimPy (UNIVERSITY OF CALIFORNIA AT DAVIS , 2002). Trata-se de um pacote Python que permite que sejas criados facilmente modelos de sistemas de eventos discretos. Essa ferramenta foi projetada para fornecer suporte à modelagem e simulação de recursos heterogêneos em rede P2P. Ela fornece primitivas para a criação de tarefas de aplicação, o mapeamento das tarefas aos recursos. Nessa, que foram efetuadas todas as simulações do modelo de compartilhamento de recursos proposto. Diversos motivos motivaram o desenvolvimento de um simulador próprio, são eles:

- Garantir todos os requisitos do tipo de ambiente distribuído que se deseja criar a partir dos resultados desta dissertação;

- Permitir a implementação dos componentes arquiteturais de software propostos na arquitetura apresentada no Capitulo 6;
- Permitir a implementação de protocolos de comunicação *peer-to-peer*;
- Permitir implementar políticas de computação distribuída baseadas em redes sociais *peer-to-peer*;
- Facilitar o encadeamento de todas as fases envolvidas na simulação e obtenção de resultados de acordo com os objetivos definidos para este trabalho de pesquisa.

9.2.1 Linguagem Python

O Python é uma linguagem de programação poderosa e de fácil aprendizado. Possui estruturas de dados de alto nível e eficientes, bem como adota uma abordagem simples e efetiva para a programação orientada a objetos. Sua sintaxe elegante e tipagem dinâmica, em adição à sua natureza interpretada, tornam Python ideal para scripting e para o desenvolvimento rápido de aplicações em diversas áreas e na maioria das plataformas.

O interpretador de Python e sua extensa biblioteca padrão estão disponíveis na forma de código fonte ou binário para a maioria das plataformas, a partir do site http://www.python.org/, e deve ser distribuído livremente. No mesmo sítio estão disponíveis distribuições e referências para diversos módulos, programas, ferramentas e documentação adicional contribuídos por terceiros.

O interpretador de Python é facilmente extensível incorporando novas funções e tipos de dados implementados em C ou C++ (ou qualquer outra linguagem acessível a partir de C). Python também se adequa como linguagem de extensão para customizar aplicações.

9.2.2 SimPy

Simpy (***Sim****ulation in* ***Py****thon*) é pacote de simulação baseada em processos de eventos discretos utilizando linguagem *Python*. Ele é distribuído sob a licença LGPL, começando com a versão 1.5.1. Fornece um conjunto de componentes que são utilizados na concepção de um modelo de simulação. Entre eles estão incluídos os processos, mensagens e os recursos. Ele também oferece de monitoramento de

variáveis por meio de informações estatísticas (UNIVERSITY OF CALIFORNIA AT DAVIS , 2002).

Em vez de usar threads, como é o caso da maioria dos pacotes de simulação orientados a processos, *Simpy* faz uso de *gernerator* do *Python*. G*ernerator* é um procedimento especial que pode ser usado para controlar iteradores de loops. Um *generator* é muito similar para funções que retornam vetores. *Generators* podem ter parâmetros, que também podem ser referenciados e geram uma sequência de valores. Entretanto, em vez de construir uma sequência que contenha todos os valores e os retornam de uma só vez, um *generator* utiliza a palavra-chave *yield* para retornar os valores um de cada vez, que utiliza menos memória e permitindo o processamento de poucos valores rapidamente. Um *generator* é uma função mas comporta-se como um iterador. *Generator* permite ao programador especificar que uma função pode ser encerrada prematuramente e, posteriormente, reinserida no ponto de última saída, permitindo sub-rotinas, ou seja, funções em que a execução se alterna com as outras. A saída / pontos de reentrada são marcadas por palavra-chave do Python rendimento. Cada nova chamada para a função faz com que a retomada da execução da função no ponto imediatamente após a produção última executado nessa função (UNIVERSITY OF CALIFORNIA AT DAVIS , 2002).

As principais classes do *SimPy* são:

- *Process* - simula uma entidade que evolui no tempo, por exemplo, um cliente que precisa ser servido por uma máquina;
- *Resource* – simula a fila de determinado recurso. Existem três tipos de unidades de recursos (*Resource, Level e Store*).

Durante a simulação, os objetos de processo poderão entrar em estado de espera por tempo fixo ou aleatório. Poderão entrar na fila de espera de um determinado recurso, e poderão ser interrompidos por ou interagir de outras maneiras com outros processos e componentes.

Um script *Simpy* contém a declaração de uma ou mais classes de *Process* e também mais objetos são criados, a partir dessas classes. Cada objeto de processo executa seu método de execução de processo (*Process Execution Method* -PEM). O PEM é um método que determina as ações dos processos. Cada PEM corre em paralelo e poderá interagir com PEMs de outro processo. O PEM define as ações que são executadas por seus objetos de processo. Cada PEM deve conter, pelo menos, uma das declarações *yield*. *Yield* são os comandos de simulação que afetam o ciclo de vida dos objetos do processo. Esse comando controla a execução e sincronização de vários processos. Elas podem atrasar o processo, colocá-lo para dormir, pedir um recurso compartilhado ou oferecer um recurso. Eles podem adicionar novos eventos

ao escalonamento de eventos da simulação, cancelar os já existentes, ou fazer processos esperar por uma mudança de estado.

9.2.3 *Python Distribute System Simulator - PyDSSim*

O *pyddsim* é um simulador de redes orientado a eventos discretos, desenvolvido especificamente para essa pesquisa. Oferece suporte à simulação de rede P2P baseado no modelo de *SuperPeers* e utilizando o algoritmo do hipercubo virtual.

Podemos destacar as seguintes características do *pydssim*:

- Permite a modelagem de tipos heterogêneos de serviços;
- Serviços podem ser modelados de forma a serem compartilhados tanto em nível de espaço quanto em nível de tempo;
- Serviços estarão disponíveis a todo tempo;
- Diferentes aplicações podem ser simuladas em paralelo, não havendo limites no número dessas aplicações;
- Torna transparentes os serviços de rede;
- Configurável e personalizável, de modo que as características dos protocolos, geradores de tráfego, enlaces e outros elementos de rede possam ser livremente modificados, estendidos, controlados ou desabilitados;
- Flexível nas configurações, de forma que comportamentos diferentes possam ser mais facilmente simulados;
- De uso direto e pragmático, ou seja, evitando gasto de tempo em aprendizado de inúmeros recursos talvez não utilizados ou interfaces muito complexas, com diversas linguagens;
- De código livre, aberto e gratuito, permitindo assim sua modificação e uso sem necessidade de licença.

O simulador é constituído por três módulos principais, a saber: gerador de eventos, motor de simulação e rede *peer-to-peer*.

O gerador de eventos permite basicamente a criação de eventos cujo objetivo é determinar os instantes em que alguma ação acontece durante o período configurado para simulação. Com este objetivo, o módulo de geração de eventos fornece componentes básicos que permitem distribuir instantes de tempos de modo que uma topologia de rede possa ser gerada a partir das entradas frequentes de *peers*.

A geração dos instantes em que novos *peers* entram pela primeira vez no sistema distribuído determina somente em que ponto da linha de tempo definida para a simulação o evento irá acontecer. Os detalhes sobre a organização da topologia de rede são tratados pelo módulo de rede *peer-to-peer*.

São estipulados os tempos de vida que cada *peer* terá, a partir do momento em que este se conecta ao sistema distribuído. Para gerar os tempos em que os *peers* se mantêm desconectados, o módulo de geração de eventos oferece um gerador de eventos capaz de distribuir os instantes de tempo em que os *peers* se mantêm *off-line*. A partir dos tratamentos dos eventos gerados por cada um dos geradores de eventos oferecidos pelo simulador, são gerados os dados de saída de simulação, permitindo a extração de dados que, futuramente, poderão auxiliar na composição de uma geração de resultados

O motor de simulação é responsável pela execução do processo de simulação propriamente dito, retirando e submetendo para tratamento os eventos de simulação gerados pelo módulo gerador de eventos. Dessa forma, os eventos são retirados conforme a evolução do relógio de simulação e, em seguida, são submetidos a tratamento, ocasionando a execução de algum comportamento programado. Esse motor foi implementado utilizando-se a *API SimPy* apresentada na seção 9.2.2.

O módulo de rede *peer-to-peer* é responsável pelo fornecimento de componentes e algoritmos relacionados às operações de redes *peer-to-peer* e a políticas de computação distribuída. Além disto, também são oferecidos componentes que permitem gerenciar a topologia da rede *peer-to-peer* que tenha sido criada por algum protocolo de rede configurado na simulação. Atualmente, o módulo de rede *peer-to-peer* contempla somente um protocolo de rede não estruturada, baseado no HyperBone (BONA, 2006) e no PeerCube (ANCEAUME, BRASILEIRO, *et al.*, 2008). Apesar disso, é possível desenvolver outros protocolos, de modo que estes possam ser utilizados nos processos de simulação que necessitem de topologias específicas para redes *peer-to-peer*.

Outro ponto importante, no que diz respeito ao módulo de redes *peer-to-peer,* é a possibilidade de se criarem protocolos para aplicações ou políticas de computação distribuída. Isto é feito por meio da implementação de mensagens com rótulos específicos e tratadores de mensagens, que permitem a execução de algum comportamento quando um *peer* recebe uma mensagem de um de seus vizinhos.

Por fim, o módulo de rede *peer-to-peer* ainda contempla um conjunto de componentes que implementam, de maneira básica, o funcionamento dos recursos computacionais existentes em um *peer.* A partir destes componentes de recursos

computacionais, são configuradas as quantidades disponíveis para cada tipo de recurso existente em um *peer*.

Além dos módulos, o simulador de rede também oferece um conjunto de parâmetros de entrada para configuração das simulações, são eles:

- Número de *superpeers*;
- Número de *peers* simples;
- Número de conexões entre *superpeers* e *defaultpeers*;
- Taxa de chegada de *superpeers*;
- Taxa de chegada de *defaultpeers*;
- Distribuição dos tempos de vida dos *defaultpeers*
- Distribuição dos tempos em que *defaultpeers* ficam *off-line*
- Duração de uma simulação;

A Figura 24 apresenta o código fonte do programa principal do processo de simulação. Nela podemos observar os passos de inicialização das principais variáveis nessa tarefa.

```
'''
Created on 27/10/2009

@author: LGustavo
'''
from pydssim.simulation.reciprocal_trade_simulation_p2p import ReciprocalTradeSimulationP2P
from pydssim.simulation.process.factory.new_portalpeers_simulation_process_factory import
NewPortalPeersSimulationProcessFactory
from pydssim.simulation.process.factory.new_superpeers_simulation_process_factory import
NewSuperPeersSimulationProcessFactory
from pydssim.simulation.process.factory.begin_simulation_process_factory import BeginSimulationProcessFactory
from pydssim.simulation.process.factory.new_peers_simulation_process_factory import NewPeersSimulationProcessFactory
from pydssim.simulation.process.factory.new_trading_simulation_process_factory import NewTradingSimulationProcessFactory
from pydssim.simulation.process.factory.out_peers_simulation_process_factory import OutPeersSimulationProcessFactory
from pydssim.simulation.process.factory.out_superpeers_simulation_process_factory import
OutSuperPeersSimulationProcessFactory

simulation = ReciprocalTradeSimulationP2P()
simulation.setSimulationTime(3850)
simulation.setResourcePeer(7)
simulation.initializeTrust(20, "1/1/2009 1:30", "1/12/2009 1:32")
simulation.initializeNetwork(640, 3000, 10)

simulation.addSimulationProcessFactory(NewPortalPeersSimulationProcessFactory())
simulation.addSimulationProcessFactory(NewSuperPeersSimulationProcessFactory())
simulation.addSimulationProcessFactory(NewPeersSimulationProcessFactory())
simulation.addSimulationProcessFactory(OutPeersSimulationProcessFactory())
simulation.addSimulationProcessFactory(OutSuperPeersSimulationProcessFactory())
simulation.addSimulationProcessFactory(BeginSimulationProcessFactory())
simulation.addSimulationProcessFactory(NewTradingSimulationProcessFactory())


print simulation.start()
```

Figura 24 – Código Fonte inicio da Simulação

Portanto, por meio dos módulos e dos parâmetros de configuração de simulação do simulador de rede, foi possível preparar cada um dos cenários de execução dos experimentos apresentados neste capítulo.

9.3 OBJETIVOS DOS EXPERIMENTOS

Para a execução dos experimentos no simulador, foi necessário definir previamente os objetivos dos experimentos e planejar a execução de cada um deles, selecionando os geradores de eventos para cada cenário de execução de simulação, além dos parâmetros relacionados a cada um dos cenários criados e executados.

Antes de executar um experimento é necessário definir os objetivos que se deseja alcançar com o mesmo, a fim de garantir que os seus resultados sejam relevantes. Uma vez que estes objetivos são definidos, as simulações são criadas e executadas e, em seguida, os resultados destas são coletados e analisados. Portanto, os objetivos foram definidos, a fim de avaliar os seguintes aspectos:

1. Avaliar o *overhead* gerado pelos algoritmos de rede *peer-to-peer* e a política de troca de recurso sem ambiente distribuído, capturando o número de mensagens geradas;
2. Avaliar a quantidade de recursos agregados à medida que o tamanho da rede *peer-to-peer* aumenta;
3. Avaliar o processo de negociação da permuta de serviço compartilhado entre membros de uma rede *peer-to-peer*, variando o número de membros detentores do serviço;

A partir desses objetivos, foi possível nortear todo o planejamento dos experimentos, o que permitiu que fosse elaborada a metodologia de execução, que é repetida para cada experimento realizado neste trabalho.

9.4 METODOLOGIA DE EXECUÇÃO

Para que os resultados dos experimentos sejam relevantes aos objetivos apresentados anteriormente, é necessário elaborar um planejamento das atividades para que seja possível extrair os resultados e, em seguida, analisá-los.

Nesse sentido, a metodologia de execução dos experimentos consiste na execução das simulações de cada um dos cenários apresentados na seção 9.6, de

maneira repetitiva, de modo que cada cenário e/ou situações específicas de cenário sejam executados dez vezes. Como visto antes, cada cenário possui um propósito, o que permitirá a comparação dos resultados, a fim de permitir fazer uma avaliação do desempenho das políticas de compartilhamento de recursos baseadas em permuta multilateral. Então, para cada execução de simulação, são gerados arquivos de texto com os resultados gerados na simulação. Este formato pode ser aberto em uma planilha eletrônica para análise dos dados e criação de gráficos dos resultados, como foi feito neste trabalho.

9.5 CONFIGURAÇÃO DAS SIMULAÇÕES

As configurações utilizadas nas simulações foram criadas levando em consideração a infraestrutura computacional apresentada na Tabela 10

Tabela 10 - Configuração da infraestrutura do ambiente distribuído

Parâmetro	Configuração
Algoritmos da rede *peer-to-peer*	*HyperBone/PeerCube*
Nº de *superpeers*	2^n *onde* n *é o numero de conexões.* $Ex.: n = 6 => 64$
Nº de *defaultpeers*	10 por superpeers
Conexões entre *superpeers*	Log_2(Nº de $superpeers$)
Número máximo de saltos dados por uma mensagem	Conexões entre *superpeers*
Tempo de entrada de *superpeer*	50 a 70 seg
Tempo o de saída de *superpeer*	200 a 240 seg
Tempo de entrada de *peer*	5 a 12 seg
Tempo de saída de *peer*	24 a 48 seg
Número de serviços oferecidos por *peer*	4 podendo variar de ± 3 serviços
Tempo para inicio de uma nova negociação de troca	25 a 50 seg

Para definir os instantes de entrada de saída dos *peers* no ambiente distribuído, foram utilizadas configurações distintas, uma para *superpeers* e outra para *defaultpeers*. Esses instantes seguem uma distribuição de Uniforme. Os parâmetros dessa distribuição variam entre 50 a 70 segundos para o tempo de entrada e entre 200 a 240 segundos para o tempo de saída de um *superpeers* e, entre 5 a 12 segundos para o tempo de entrada e entre 24 a 28 segundos para o tempo de saída de um *defaultpeer*. Todos os instantes de tempo serão gerados entre 1 a 3600 s, sendo este último valor a duração de cada rodada de simulação, o que equivale a1h.

O valor definido em saltos é utilizado para definir o número de vezes que uma mensagem pode ser repassada de um *peer* para outro. Este valor deve ser sempre igual ao tamanho do caminho percorrido pela mensagem, à medida que esta é

repassada de *peer* para *peer*. Por fim, os parâmetros de uma mensagem são utilizados para transportar informações de um *peer* a outro.

Além dos tipos de eventos já mencionados até agora, também são gerados outros eventos, cuja finalidade é definir os instantes em que oportunidades de permuta são publicadas no ambiente distribuído simulado e o momento em que os *peers* compartilham hardware. A geração destes dois tipos de eventos se dá quando um evento primário é tratado ou no momento em que os *peers* reagem ao recebimento de uma mensagem enviada por algum outro *peer* que esteja ativo durante a simulação. Assim, quando uma oportunidade de permuta é criada, são gerados eventos de publicação desta oportunidade, de modo que os mesmos aconteçam de maneira frequente. Já os eventos de compartilhamento de hardware são criados após um *peer* ingressar em uma rede *peer-to-peer* juntamente com outros *peers*.

Além das configurações para geração de eventos, também são configuradas as disponibilidades de serviços. Dentro da disponibilidade de cada um dos tipos de serviço, o percentual de compartilhamento varia de 15% a 90%, ou seja, um *defaultpeer* não compartilhará totalmente a quantidade de serviço disponível, tendo em vista que o mesmo precisa de uma quantidade de recursos disponíveis para manter seu funcionamento. Tais percentuais podem ser observados à medida que um usuário comum utiliza seu computador pessoal.

Para facilitar um processo de sincronização, as solicitações tanto de troca quanto de inclusão na rede são capturadas pelos *superpeers* que intermedeiam a comunicação entre dois ou mais *defaultpeers*. Além disto, os *superpeers* também são utilizados para manter informações para autenticação e autorização dos usuários no sistema distribuído. Por se tratar de um assunto amplo, nesta tese somente é apresentada uma discussão sobre questões envolvendo segurança de informação no que diz respeito ao uso de redes *peer-to-peer* como infraestrutura de computação. Por não fazer parte do escopo deste trabalho, políticas de computação distribuída envolvendo autenticação e autorização de usuários serão propostas em trabalhos futuros.

9.6 DEFINIÇÃOS DE CENÁRIOS E SUA SIMULAÇÃO

A etapa de definição de cenários e avaliação do sistema proposto tem como objetivo testar o sistema em um ambiente acadêmico distribuído, focalizando todos os pontos pertinentes ao seu correto funcionamento, tais como as funcionalidades referentes ao ambiente P2P e ao processo de negociação dos dados compartilhados.

Estes possibilitaram a avaliação do sistema sob o ponto de vista da formação da rede P2P, comunicação entre os *peers*, atualizações dos índices de roteamento e processo de permuta dos serviços, levando-se em conta o tempo e a corretude dos resultados retornados aos usuários.

Analisando os objetivos na seção 9.3, foram projetados quatro cenários que são os seguintes:

1. **Estabilização da rede *peer-to-peer***: os *peers* formam a infraestrutura do ambiente de computação, sem apresentarem falhas. Com este cenário, pretendesse obter os seguintes resultados: quantidade de mensagens geradas à medida que os *peers* se conectam à rede *peer-to-peer*; à medida que mensagens são trocadas entre os *peers* ao longo do processo de estabilização; média de tempo em que os *peers* levam para entrar e sair da rede;
2. **Compartilhamento de apenas dois serviços em uma rede *peer-to-peer* por meio da permuta Multilateral :** serão compartilhados apenas dois tipos de serviços na rede *peer-to-peer* utilizando-se o modelo econômico de permuta multilateral. Com este cenário, pretende-se obter resultados que permitam avaliar o desempenho do uso de uma rede *peer-to-peer* em um processo de compartilhamento de serviços, limitado a apenas dois serviços, por meio desse tipo de permuta;
3. **Compartilhamento de n serviço em uma rede *peer-to-peer* por meio da permuta Multilateral :** serão compartilhados n tipos de serviços na rede *peer-to-peer* utilizando o modelo econômico de permuta multilateral. Com este cenário, pretende-se obter resultados que permitam avaliar o desempenho do uso de uma rede *peer-to-peer* em um processo de compartilhamento de serviços por meio desse tipo de permuta;
4. **Compartilhamento de serviço entre membros de uma rede *peer-to-peer* por meio da permuta bilateral :** serviços serão compartilhados na rede *peer-to-peer* utilizando-se o modelo econômico de permuta bilateral. Com este cenário, pretende-se obter resultados que permitam avaliar o desempenho do uso de uma rede *peer-to-peer* em um processo de compartilhamento de serviços por meio desse tipo de permuta;

A partir dos objetivos traçados e dos cenários projetados, podemos definir uma metodologia de execução dos experimentos, a fim de obter resultados capazes de serem analisados de acordo com os objetivos definidos.

Como mencionado anteriormente, os experimentos foram executados a partir do simulador de redes *peer-to-peer* desenvolvido ao longo deste trabalho. Os resultados foram coletados a partir dos *logs* de cada uma das simulações. As entradas nos *log s*eram geradas à medida que eventos eram tratados por componentes do motor de simulação.

É importante destacar que a implementação do simulador de redes *peer-to-peer* teve como objetivo a realização de uma prova de conceito. Portanto, os resultados que serão apresentados podem ser melhorados em versões futuras de cada um dos algoritmos *peer-to-peer* e das políticas de computação distribuída baseadas em redes sociais ou em implementações destes em algum protótipo no futuro.

9.6.1 Cenário 1: Estabilização da Rede *Peer-to-Peer*

A partir das dez rodadas de simulação realizadas para o cenário de estabilização da rede *peer-to-peer,* foi possível obter resultados relevantes quanto ao sistema distribuído, uma vez que a infraestrutura deste é formada por uma rede *peer-to-peer* não estruturada.

Neste cenário, a rede *peer-to-peer* apresenta oscilações no número de *peers*, ou seja, durante o período de simulação, ocorreram entradas e saídas de *peer* na rede. Como mencionado anteriormente na Seção 9.5, o número de *peer* e *superpeer* está diretamente relacionado com a dimensão do hipercubo. Nas simulações desses cenários, foi adotado um hipercubo de dimensão $n = 6$, podendo a rede chegar ao número máximo de 64 *superpeers* e de no máximo 640 *peers*, conforme tabela da seção mencionada.

Essa implementação apresentou bons resultados no que diz respeito à quantidade de mensagens geradas de acordo com a chegada de *peers* na rede. Comparando-se a outros protocolos, como por exemplo Gnutella, destaca-se que a quantidade de mensagens geradas, pelo protocolo Gnutella, cada vez que um *peer* se conecta a rede *peer-to-peer* é exibida na Figura 25. Note-se que o número de mensagens é exibido com escala exponencial diferentemente da escala do protocolo Gnutella (GONÇALVES, 2010) mostrada na Figura 26.

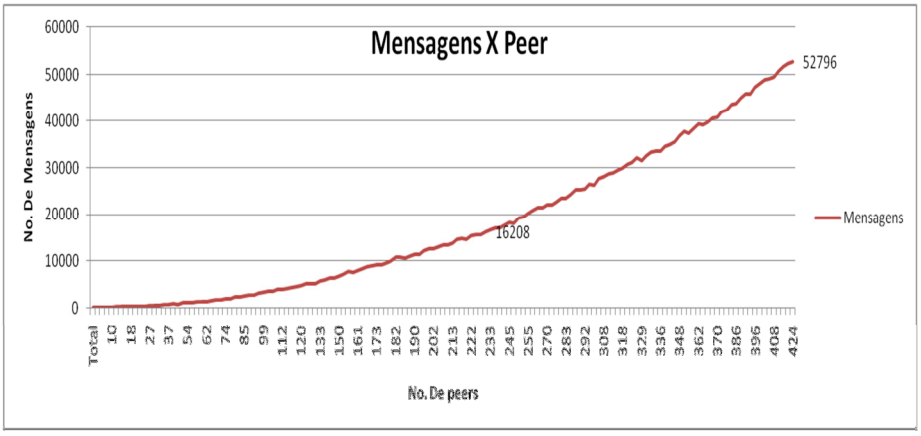

Figura 25 – Evolução do número de mensagem em relação ao número de peers

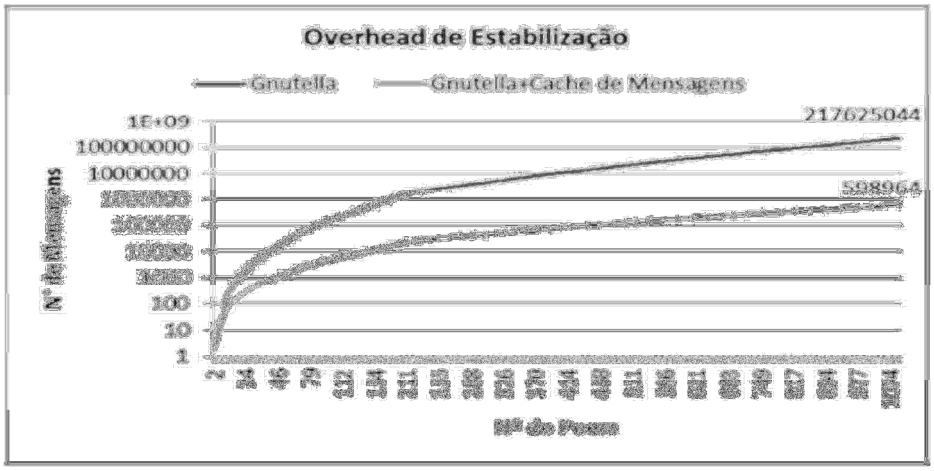

Figura 26 - Overhead de estabilização da rede peer-to-peer (GONÇALVES, 2010)

A descrição do funcionamento do protocolo utilizado nesta tese, tendo como base a especificação demonstrada em (BONA, 2006) e (ANCEAUME, BRASILEIRO, *et al.*, 2008), tem seu detalhamento apresentado na seção 8.2.

O grande *overhead* da implementação padrão do Gnutella e devido ao fato de que quando um *peer* se conecta à rede de maneira não determinística, são disparadas mensagens tipo PING, a fim de notificar todos os outros *peers* conectados à rede. À medida que cada PING é enviado e, em seguida, repassado sucessivamente até atingir o número máximo de saltos, mensagens PONG são enviadas de volta, de modo que o *peer* recém conectado à rede possa ter informações sobre outros *peers* da rede. O grande problema no uso do Gnutella não são as mensagens de *feedback* que os *peers* já conectados enviam para aquele recém chegado à rede, mas a quantidade de mensagens PING que inundam a rede.

O benefício obtido, com o uso dos algoritmos do Hipercubo, na redução do número de mensagens empregadas pelos *peers,* no momento de usar conexão na

rede, pode ser visto como um incentivador do uso dessa topologia. Essa redução ocorre uma vez que não há o envio de mensagem para todos os outros *peer* da rede. Essas mensagens são enviadas somente para os *superpeers.*

Cabe lembrar que no decorrer das execuções de todos os cenários, foram assumidos valores médios de entrada e saída de cada *peer* na rede. Esses valores foram gerados levando-se em consideração a distribuição uniforme dos valores definidos para cada evento descrito na Tabela 10. A Figura 27 apresenta essa evolução do número de *peer à* medida que irão entrando e saindo da rede seguindo as definições da tabela citada. Por meio dessa evolução, foi constatada uma entrada, em média, de 517 *peer* dos quais entraram 461 *defaultpeer* e 56 *superpeer*. Também foi constatada a saída, em média, de 93 peer dos quais 19 eram *superpeer* e 74 *defaultpeer* tendo-se, assim ,ao final da execução 426 peers no total.

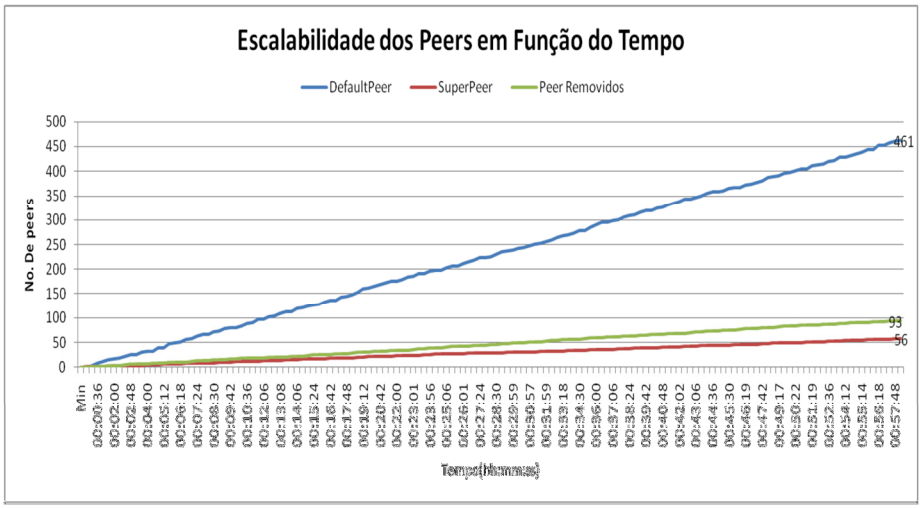

Figura 27 – Escalabilidade do número de Peers

9.6.2 Cenário 2: Compartilhamento de apenas dois serviços em uma rede peer-to-peer por meio da permuta Multilateral :

No cenário desta seção, foram realizados os experimentos para geração dos resultados nos quais se pretende avaliar o desempenho de compartilhamento com apenas dois tipos de serviço, em redes P2P, por meio do modelo econômico de permuta multilateral. Foram utilizados os serviços de memória e ciclo de processamento na execução dos experimentos desse cenário.

Também foram levadas em consideração as condições criadas pelo primeiro cenário no que diz respeito à quantidade *peer* na topologia da rede.

Para esse cenário foram efetuados três tipos de simulações distintas, todas elas variando a ocorrência ou não da análise de confiança e da análise equivalência dos serviços negociados. Como foi visto no Capitulo 7, confiança/reputação e equivalência são os principais mecanismos necessários no processo de permuta apresentado nesse trabalho.

A. Primeira Simulação - Compartilhamento de serviço sem a utilização do mecanismo de análise da confiança/reputação e de equivalência dos serviços

Essa simulação foi concebida com o intuito de avaliar como os *peers* se comportariam em um ambiente onde a negociação seria efetuada sem nenhuma forma de analise, havendo apenas o processo de troca direta. Para tanto, foram desabilitados, dos *peers,* os mecanismos de análise da confiança e da equivalência dos serviços solicitados. Com isto, os serviços passaram a ter valores comuns de confiança e valores comuns de equivalência, tornando assim todos iguais para ambas as características.

Essa primeira simulação não possui nenhuma configuração extra, que beneficie a permuta, ou seja, visa ser o mais convencional possível, como acontece em uma troca direta. A Figura 28 apresenta a evolução do processo de negociação dos recursos no decorrer do tempo. Notar-se, nesta simulação, que 51,59% das negociações submetidas foram concretizas e 38,89% das negociações efetuadas foram feitas multilateralmente. Esses valores estão diretamente ligados à quantidade de serviços disponíveis. Como trabalhamos com apenas dois tipos de serviço nota-se que a taxa de efetuar a permuta é bem próxima ao percentual das permutas não efetuadas. A Figura 29 apresenta o gráfico onde o número de processo de negociação de permuta é comparado com a evolução do número de serviços disponibilizados pelos *peers* na rede até a finalização do processo de troca. A Tabela 11 demonstra a relação existente entre o número de *peers* na rede e o número total de permutas. Já Tabela 12 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o número de permutas efetuadas.

Tabela 11 – Relação de Permutas submetidas pelo número de *Peers* sem a utilização de análise de Confiança e Equivalência com dois tipos de serviços

	Total de *Peers*	**430**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	126	65	49	58
Média por *peer* por Permutas	3,41	6,62	8,78	7,41

Tabela 12 - Relação de Permutas submetidas pelo número de serviços disponíveis sem a utilização de análise de Confiança e Equivalência com dois tipos de serviços

	Total de Serviços	**804**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	126	65	49	58
Média por Serviços Oferecidos	6,38	12,37	22,97	14,72

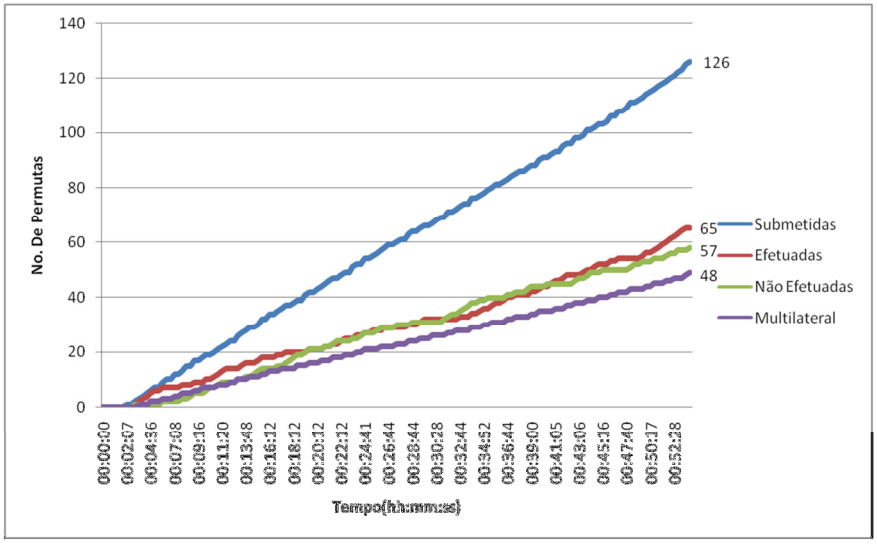

Figura 28 – Evolução das Permutas no Tempo sem Confiança e Equivalência com apenas dois tipos de serviços(min)

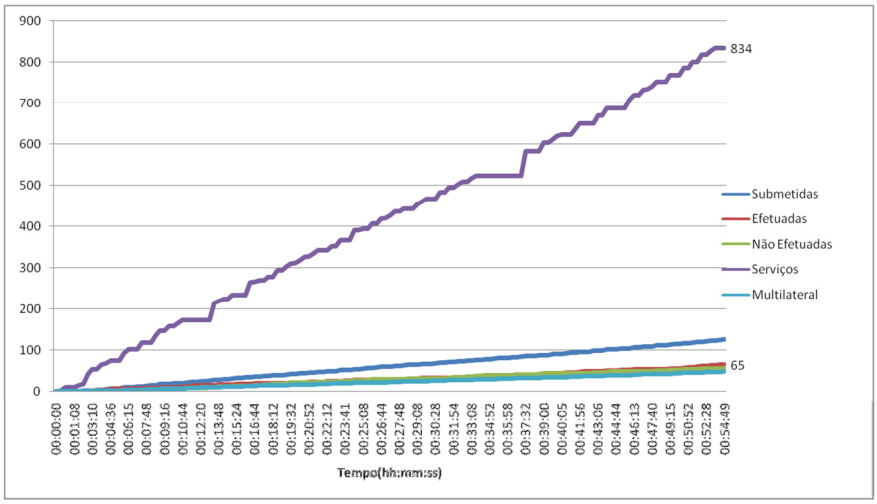

Figura 29 – Comparativo entre o número de serviços e o número de permutas sem uso de mecanismo de confiança e equivalência e com 2 tipos de serviços apenas

B. Segunda Simulação - Compartilhamento de serviço em uma rede peer-to-peer com a utilização da análise da Confiança e sem utilizar a análise de equivalência dos serviços:

Essa simulação foi idealizada com a finalidade de avaliar os *peers* de uma rede P2P em um ambiente onde a negociação de permuta de serviços onde é apenas efetuado o processo de análise de confiança. Para essa avaliação, foi desabilitado dos *peers* o mecanismo de análise de equivalência dos serviços solicitados mantendo apenas o mecanismo de análise de confiança. Esse fato fez com que os serviços passassem a ter valores diferenciados de confiança e valores comuns de equivalência.

A Tabela 13 demonstra a relação existente entre o número de peers na rede e o número total de permutas. Já a Tabela 14 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o número de permutas efetuadas. A Figura 30 apresenta a evolução do processo de negociação dos recursos no decorrer do tempo. Observa-se nesta simulação que 46,27% das negociações submetidas foram concretizadas. Como já era esperado houve uma diminuição no número de transações efetuadas devido ao fato de a essa simulação termos acrescentado o mecanismo de confiança. A Figura 31 apresenta o gráfico onde o número de processo de negociação de permuta é comparado com a evolução do número de serviços disponibilizados na simulação. Para melhor avaliação dessa comparação, foi efetuada uma transformação dos valores iniciais, utilizando-se como base log_{10}.

Tabela 13 - Relação de Permutas submetidas pelo número de *Peers*, com a análise de Confiança e sem a Análise de Equivalência para dois tipos de serviços

	Total de Peers	**458**		
	Submetidas	efetuadas	Multilateral	não Efetuadas
Total de Permutas	134	62	42	69
Média por *peer* por Permutas	3,42	7,39	10,90	6,64

Tabela 14 - Relação de Permutas submetidas pelo número de serviços disponíveis, com a análise de Confiança e sema análise de Equivalência para dois tipos de serviços

	Total de Serviços	**906**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	134	62	42	69

Média por Serviços Oferecidos	6,76	14,61	21,57	13,13

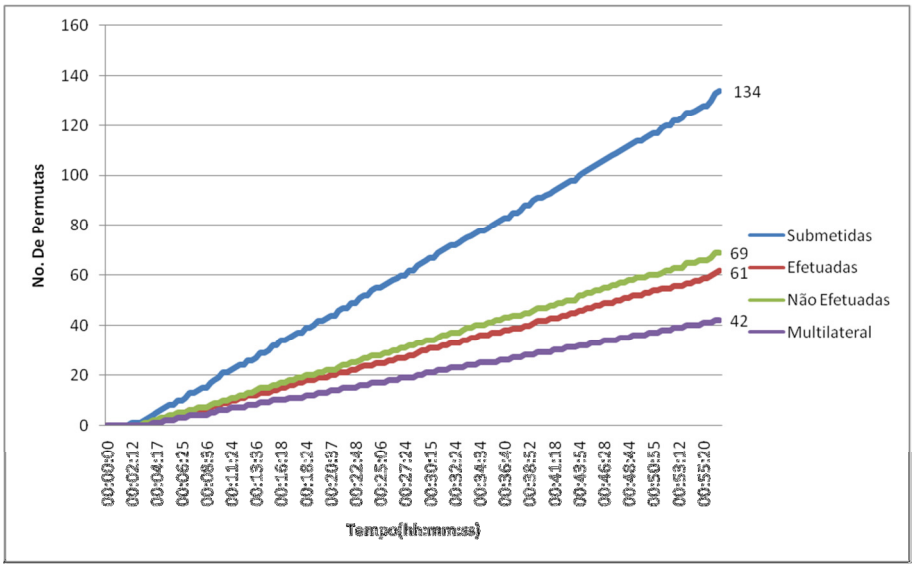

Figura 30 - Evolução das Permutas com Confiança e sem Equivalência com dois serviços

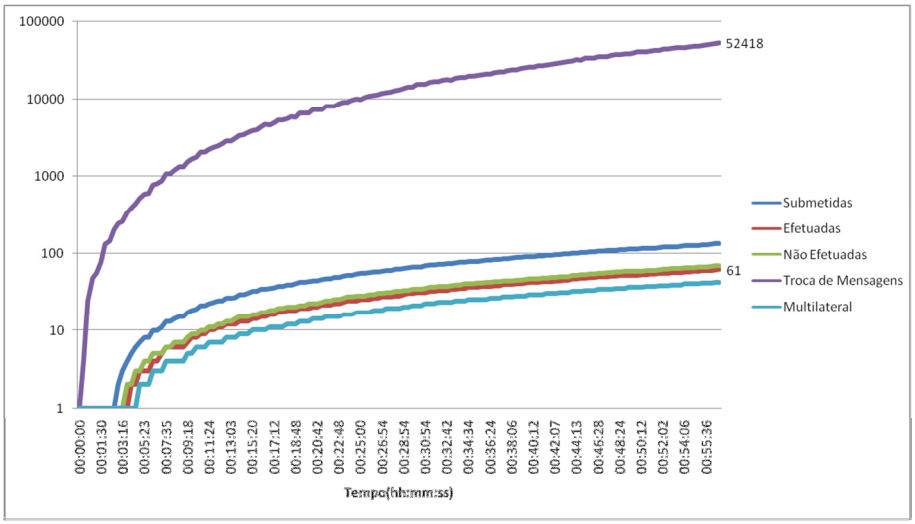

Figura 31 - Comparativo entre o número de serviços e o número de permutas sem uso de mecanismo de equivalência para dois tipos de serviços

C. Terceira Simulação - Compartilhamento de serviço de uma rede peer-to-peer utilizando os mecanismos de Confiança e Equivalência para dois tipos de serviços:

Essa simulação teve o propósito de medir o desempenho dos *peers* em um ambiente onde foram efetuadas, na negociação, todas as análises propostas nessa tese, porém utilizando-se apenas dois tipos de serviços. Com esse intuito, foram

ativados os mecanismos de analise da confiança e da equivalência dos serviços solicitados.

A Figura 32 apresenta processo de negociação dos recursos no decorre do tempo. Notar-se, nesta simulação, que 43,31% das negociações submetidas foram concretizadas. A Figura 33 apresenta o gráfico comparativo entre o número de permutas e o número de serviços disponibilizados durante o processo de permuta. Novamente foi efetuada uma transformação dos valores iniciais, utilizando-se como base log_{10}. Essa transformação gerou um gráfico em escala logarítmica. Já a Figura 34 mostra um gráfico comparativo entres os motivos - falta de confiança ou falta de equivalência - que causaram a não efetuação da permuta. Esse gráfico nos mostra que no início da simulação há uma quantidade maior de permutas não concretizadas motivada pela falta de confiança. Essa propriedade se inverte à medida que novos *peers* participam do processo de permuta. Com isto, novas informações sobre a confiança dos *peers* são criadas e disponibilizadas. Essa inversão também ocorre devido ao maior número de serviços disponibilizados na rede causando assim uma maior dificuldade de se encontrarem serviços que possuam as mesmas equivalências em *peers* diferentes..

A Tabela 15 procura demonstrar a relação existente entre o número de *peers* e o número total de permutas. Já a Tabela 16 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o numero de permutas efetuadas.

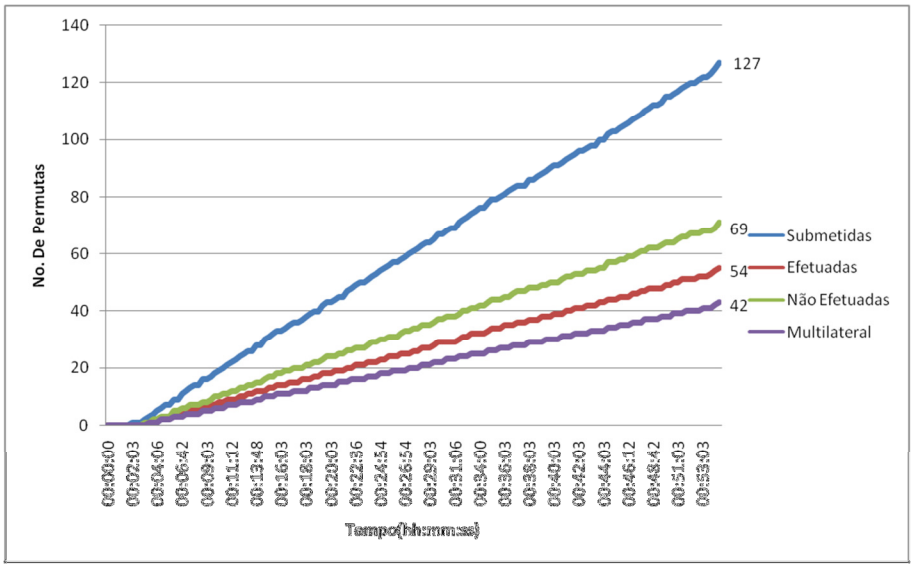

Figura 32 – Evolução de permutas Multilateral levando em consideração o compartilhamento de apenas dois tipos de serviços

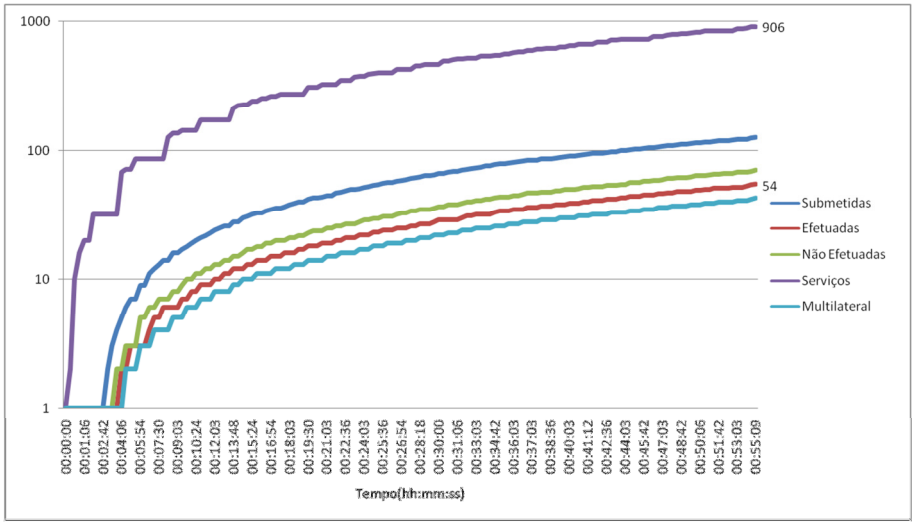

Figura 33 – Comparativo entre quantidade e Permutas e quantidade de serviços levando em consideração o compartilhamento de apenas dois tipos de serviços

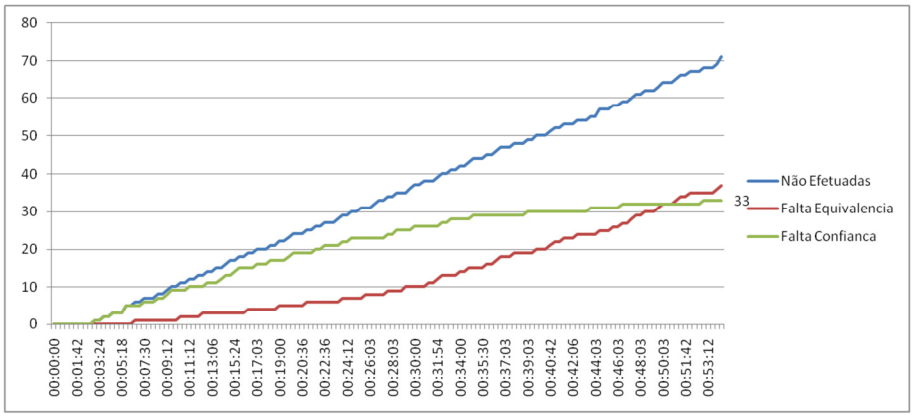

Figura 34 – Comparativo entre os motivos das permutas não efetuadas levando em consideração o compartilhamento de apenas dois tipos de serviços

Tabela 15 - Relação de Permutas submetidas pelo número de Peers para dois tipos de serviços

	Total de *Peers*	**433**		
Permutas	Submetidas	Efetuadas	Multilateral	Não Efetuada
Total de Permutas	127	55	43	71
Média por *Peer* por Permutas	3,41	7,87	10,07	6,10

Tabela 16 - Relação de Permutas submetidas pelo número de serviços disponibilizados para dois tipos de serviços

	Total de *Peers*	**906**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	127	55	43	71
Média por Serviços Oferecidos	7,13	16,47	21,07	12,76

D. Quarta Simulação - Compartilhamento de serviço de uma rede peer-to-peer utilizando os mecanismos de Confiança e Equivalência para dois tipos de serviços sem a utilização de períodos de tempo na negociação:

Essa simulação teve o propósito de medir o desempenho dos *peers* em um ambiente onde foram efetuadas, na negociação, todas as análises propostas nessa tese, utilizando-se apenas dois tipos de serviços. Porém, nessa simulação os serviços foram negociados sem a limitação de horário de uso. Isso quer dizer que não foi levado em consideração o horário de disponibilidade dos mesmos. Com esse intuito, foram ativados os mecanismos de analise da confiança e da equivalência dos serviços solicitados.

A Figura 35 apresenta processo de negociação dos recursos no decorre do tempo. Notar-se, nesta simulação, que 62,20 % das negociações submetidas foram concretizadas. A Figura 36 apresenta o gráfico comparativo entre o número de permutas e o número de serviços disponibilizados durante o processo de permuta. Novamente foi efetuada uma transformação dos valores iniciais, utilizando-se como base log_{10}. Essa transformação gerou um gráfico em escala logarítmica. Já a Figura 37 mostra um gráfico comparativo entres os motivos - falta de confiança ou falta de equivalência - que causaram a não efetuação da permuta. Esse gráfico nos mostra que no início da simulação há uma quantidade maior de permutas não concretizadas motivada pela falta de confiança. Essa propriedade se inverte à medida que novos *peers* participam do processo de permuta. Com isto, novas informações sobre a confiança dos *peers* são criadas e disponibilizadas. Essa inversão também ocorre devido ao maior número de serviços disponibilizados na rede causando assim uma maior dificuldade de se encontrarem serviços que possuam as mesmas equivalências em *peers* diferentes..

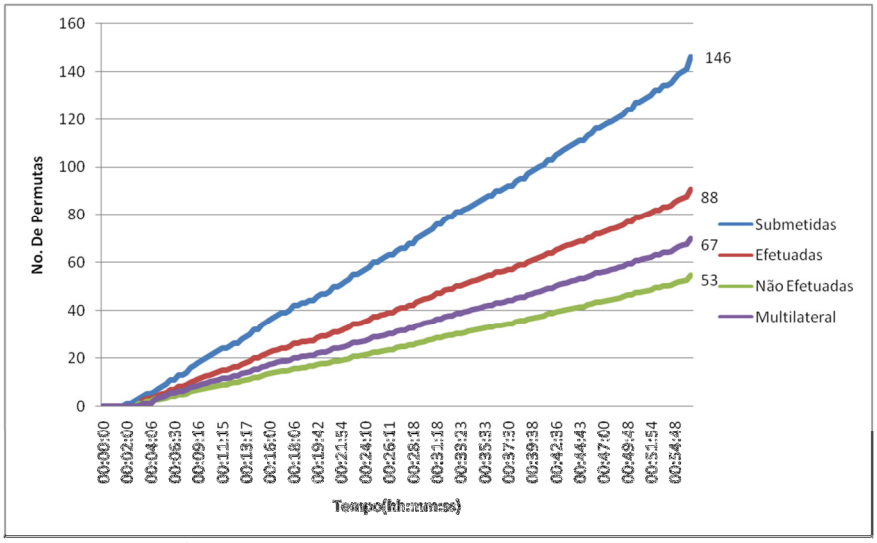

Figura 35 - Evolução de permutas Multilateral levando em consideração o compartilhamento de apenas dois tipos de serviços sem limite de tempo

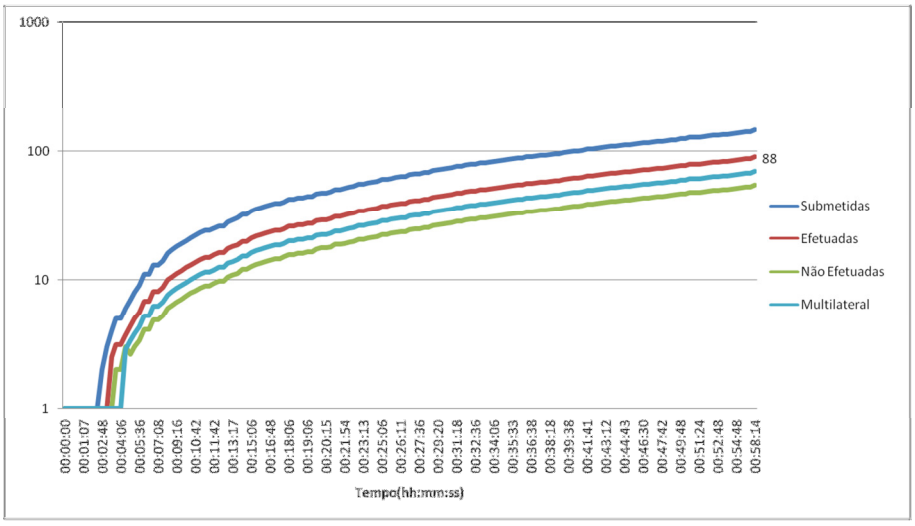

Figura 36 - Comparativo entre quantidade e Permutas e quantidade de serviços levando em consideração o compartilhamento de apenas dois tipos de serviços sem limite de tempo

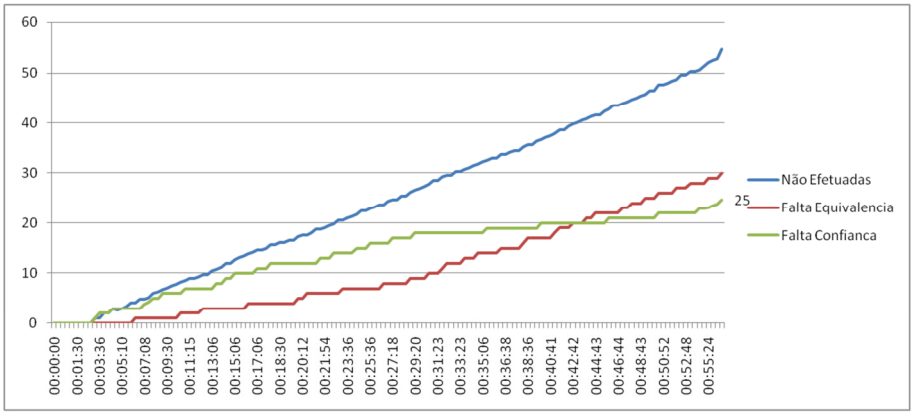

Figura 37 - Comparativo entre os motivos das permutas não efetuadas levando em consideração o compartilhamento de apenas dois tipos de serviços sem limite de tempo

9.6.3 Cenário 3: Compartilhamento n de serviço em rede *peer-to-peer* por meio da permuta Multilateral

Neste cenário foram realizados os experimentos para avaliar o desempenho do compartilhamento de n tipos de serviço na rede P2P, por meio do modelo econômico de permuta multilateral. Também para esse cenário, foram efetuados três tipos de simulações distintas, todas elas variando a ocorrência ou não da análise de confiança e da análise equivalência dos serviços negociados. As mesmas configurações que foram utilizadas na seção 9.6.2 foram aproveitadas neste cenário.

A. Primeira Simulação - Compartilhamento de serviço em uma rede peer-to-peer sem a utilização da análise da Confiança e de equivalência dos serviços

Essa simulação teve o intuito de avaliar o comportamento dos *peers,* no processo de negociação de serviços compartilhados, em um ambiente totalmente livre de análise. As considerações efetuadas na seção 9.6.2A, no que diz respeito às análises efetuadas, foram aproveitadas nesta simulação.

Seguindo as mesmas configurações usadas na seção 9.6.2A, essa simulação não possui nenhuma configuração extra, que beneficie ou prejudique o processo de permuta. A Figura 38 apresenta a evolução do processo de negociação dos recursos no decorrer do tempo. Notar-se, nesta simulação, que 46,83% das negociações submetidas foram concretizadas e 32,89% das negociações efetuadas foram feitas multilateralmente. Visto que não foi utilizada nenhuma forma de comparação entre os serviços e os mesmo foram negociados livremente, esses valores podem ser atribuídos à quantidade de serviços disponíveis. Podemos constatar que quanto maior essa quantidade mais difícil será encontrar membros da rede dispostos à troca. A Figura 39 apresenta o gráfico onde o número de processo de negociação de permuta é comparado com a evolução do número de serviços disponibilizados pelos *peers*. Para melhor avaliação dessa comparação, foi efetuada uma transformação dos valores iniciais utilizando-se como base log_{10}, de onde se tem um gráfico logarítmico. A Tabela 17 demonstra a relação existente entre o número de peers na rede e o número total de permutas. Já a Tabela 18 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o número de permutas efetuadas.

Tabela 17 – Relação de Permutas submetidas pelo número de *Peers* no compartilhamento de serviços sem a utilização de análise de Confiança e Equivalência

	Total de *Peers*	**430**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	126	59	41	67
Média por *Peer por permutas*	3,41	7,29	10,49	6,42

Tabela 18 - Relação de Permutas submetidas pelo número de serviços disponíveis no compartilhamento sem a utilização de análise de Confiança e Equivalência

	Total de Peers	**2580**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	126	59	41	67
Média por Serviços Oferecidos	20,48	43,73	62,93	38,51

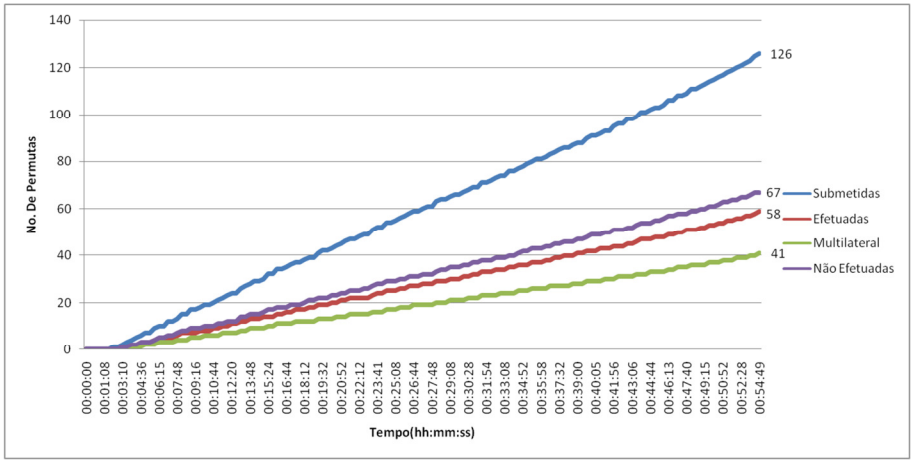

Figura 38 – Evolução das Permutas no Tempo sem Confiança e Equivalência(min)

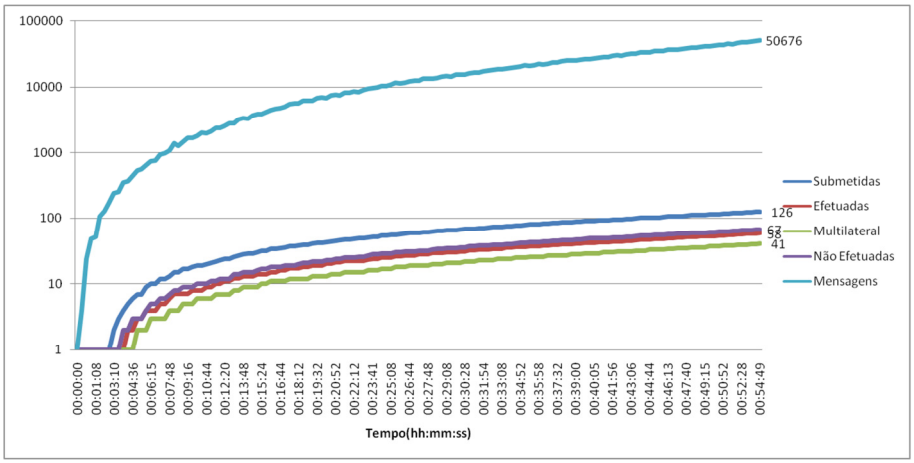

Figura 39 – Comparativo entre o número de Mensagem e o número de permutas sem uso de mecanismo de confiança e equivalência

B. *Segunda Simulação - Compartilhamento de serviço em uma rede peer-to-peer com a utilização da análise da Confiança e sem utilização da análise de equivalência dos serviços:*

Essa simulação foi concebida para avaliar o comportamento dos *peers* em um ambiente onde o processo de negociação de troca de serviços é efetuado apenas com mecanismo de análise de confiança, como também foi definido na seção 9.6.2B.

A Figura 40 apresenta a evolução do processo de negociação dos recursos no decorre do tempo. Ressalta-se nesta simulação que 40,30% das negociações submetidas foram concretizadas e 26,87% da negociações usaram a permuta multilateral. Da mesma maneira que aconteceu na simulação do seção 9.6.2B, com a utilização do mecanismo citado, ocorreu uma diminuição no número de permutas efetuadas com sucesso e um aumento no número de permutas não efetuadas. A Figura 41 apresenta o gráfico onde o número de processo de negociação de permuta é comparado com a evolução do número de mensagens enviada sem todos os eventos da simulação. Para melhor interpretação dos valores do gráfico, novamente foi efetuada a transformação desses valores iniciais para uma escala logarítmica na base 10. A Tabela 19 demonstra a relação existente entre o número de *peers* na rede e o número total de permutas. Já a Tabela 20 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o número de permutas efetuadas.

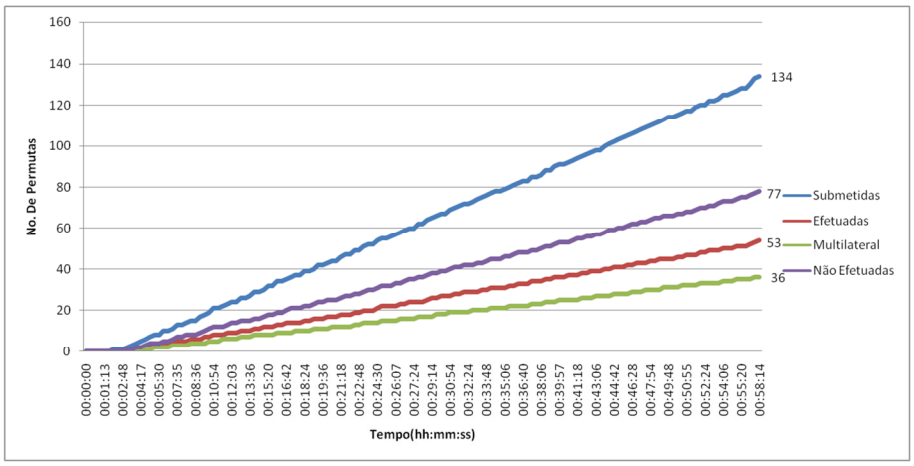

Figura 40 - Evolução das Permutas no Tempo com Confiança e sem Equivalência

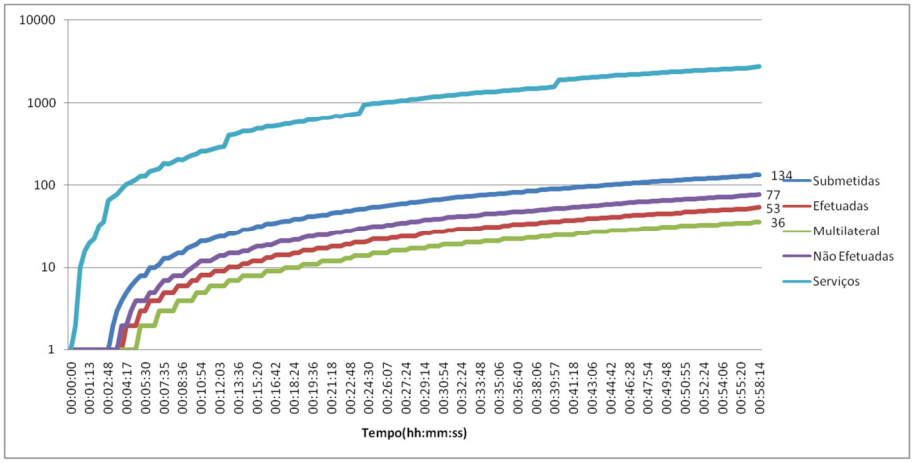

Figura 41 -Comparativo entre o número de serviços e o número de permutas sem uso de mecanismo de equivalência

Tabela 19 - Relação de Permutas submetidas pelo número de *Peers* no compartilhamento de serviços com análise de confiança e sem de análise de equivalência

	Total de Peers	**458**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	134	54	36	78
Média por Serviços Oferecidos	3,42	8,48	12,72	5,87

Tabela 20 - Relação de Permutas submetidas pelo número de serviços disponíveis no compartilhamento sem a utilização de análise de Confiança e Equivalência

	Total de Peers	**2748**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	134	54	36	78
Média por Serviços Oferecidos	20,51	50,89	76,33	35,23

C. Terceira Simulação - Compartilhamento de serviço entre membros de uma rede peer-to-peer utilizando de Confiança e Equivalência dos serviços:

Considerada a simulação mais importante, essa foi concebida com o propósito de aferir o comportamento de todos dos *peers* em um ambiente com os mecanismos de análise de confiança e equivalência ativados. A Figura 42 apresenta processo de negociação dos recursos no decorrer do tempo. Nessa figura é observado que um total de 36,57% das negociações submetidas foram concretizadas. A procura demonstrar a relação existente entre o número de *peers* e o número total de permutas.

Já Tabela 22 apresenta a relação existente entre o número de serviços oferecidos no processo de compartilhamento e o número de permutas efetuadas.

A Figura 43 apresenta o gráfico comparativo entre o número de permutas e o número de serviços disponibilizados durante o processo de permuta. Novamente foi efetuada uma transformação dos valores iniciais, utilizando-se como base log_{10}. Essa transformação gerou um gráfico em escala logarítmica. Já a Figura 44 mostra um gráfico comparativo entres motivos - falta de confiança ou falta de equivalência - que causaram a não efetuação da permuta. Esse gráfico nos mostra que também acontece uma inversão nas quantidades de referentes falta de confiança e falta de equivalência em relação ao inicio da execução da simulação. Esse fato novamente acontece com ocorrência das mesmas características citadas na seção 9.6.3-C.

Tabela 21 - Relação de Permutas submetidas pelo número de *Peers*

	Total de Peers	**458**		
Permutas	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	134	49	33	83
Média por Peer	3,42	9,35	13,88	5,52

Tabela 22 - Relação de Permutas submetidas pelo número de serviços disponibilizados

	Total de Peers	**2748**		
	Submetidas	Efetuadas	Multilateral	Não Efetuadas
Total de Permutas	134	49	33	83
Média por Serviços Oferecidos	20,51	56,08	83,27	33,11

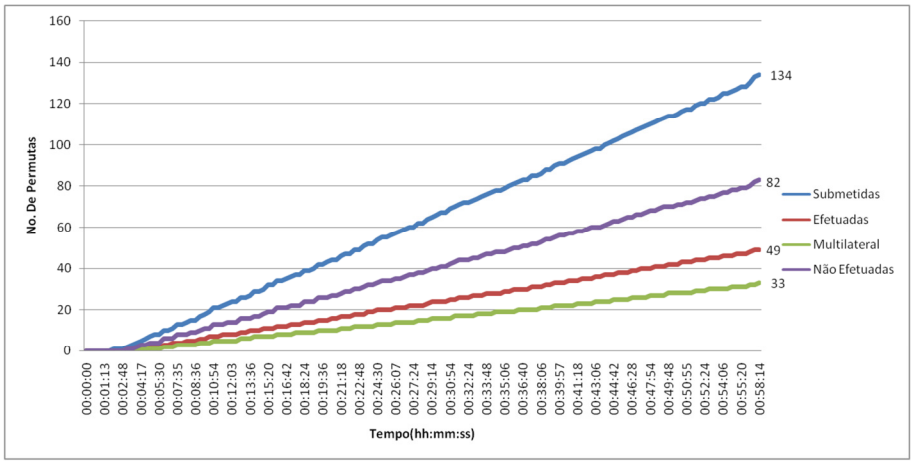

Figura 42 - Evolução de permutas Multilateral

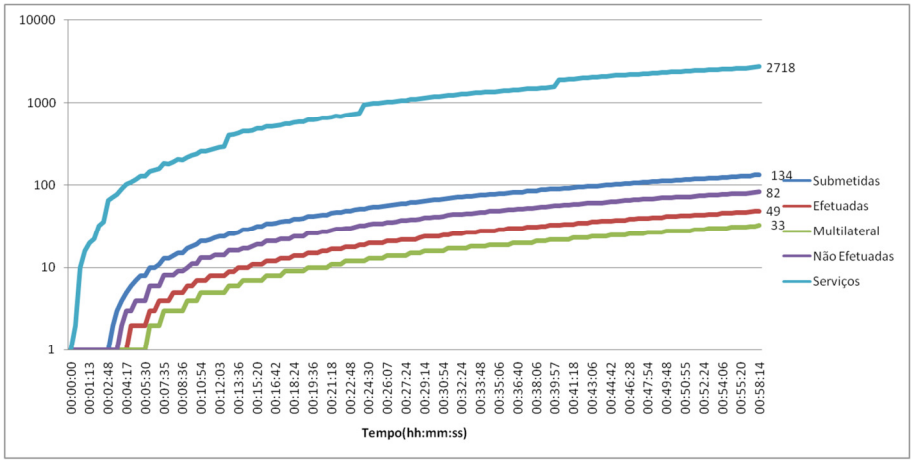

Figura 43 - Comparativo entre quantidade de Permutas e quantidade de serviços

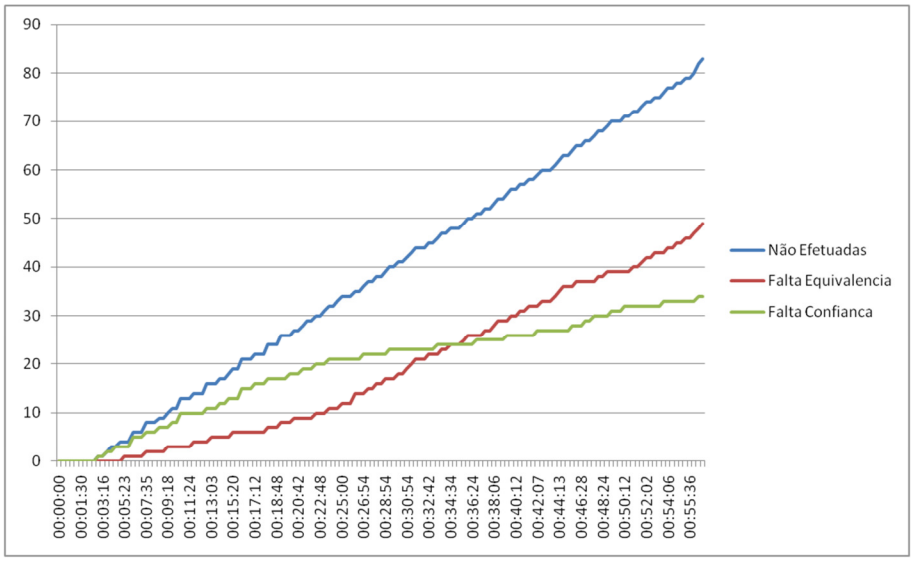

Figura 44 - Comparativo entre os motivos das permutas não efetuadas

D. Quarta Simulação - Compartilhamento de serviço de uma rede peer-to-peer utilizando os mecanismos de Confiança e Equivalência para n tipos de serviços sem a utilização de períodos de tempo na negociação:

Essa simulação teve o propósito de medir o desempenho dos *peers* em um ambiente onde foram efetuadas, na negociação, todas as análises propostas nessa tese, utilizando-se n tipos de serviços. Porém, nessa simulação os serviços, da mesma forma que na seção 9.6.2D, foram negociados sem a limitação de horário de uso dos mesmos. Isso quer dizer que não foi levado em consideração o horário de disponibilidade dos mesmos. Os serviços forma negociados de acordo com a sua

disponibilidade. Com esse intuito, foram ativados os mecanismos de analise da confiança e da equivalência dos serviços solicitados.

A Figura 45 apresenta processo de negociação dos recursos no decorre do tempo. Notar-se, nesta simulação, que 55,15 % das negociações submetidas foram concretizadas. A Figura 46 apresenta o gráfico comparativo entre o número de permutas e o número de serviços disponibilizados durante o processo de permuta. Novamente foi efetuada uma transformação dos valores iniciais, utilizando-se como base log_{10}. Essa transformação gerou um gráfico em escala logarítmica. Já a Figura 47 mostra um gráfico comparativo entres os motivos - falta de confiança ou falta de equivalência - que causaram a não efetuação da permuta. Esse gráfico nos mostra que no início da simulação há uma quantidade maior de permutas não concretizadas motivada pela falta de confiança. Essa propriedade se inverte à medida que novos *peers* participam do processo de permuta. Com isto, novas informações sobre a confiança dos *peers* são criadas e disponibilizadas. Essa inversão também ocorre devido ao maior número de serviços disponibilizados na rede causando assim uma maior dificuldade de se encontrarem serviços que possuam as mesmas equivalências em *peers* diferentes.

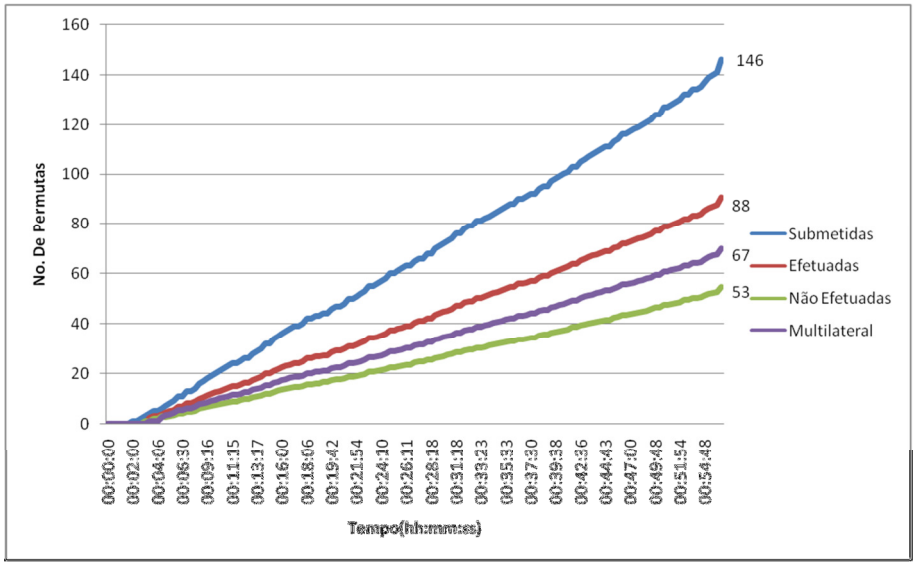

Figura 45 - Evolução de permutas Multilateral levando em consideração o compartilhamento de n tipos de serviços sem limite de tempo

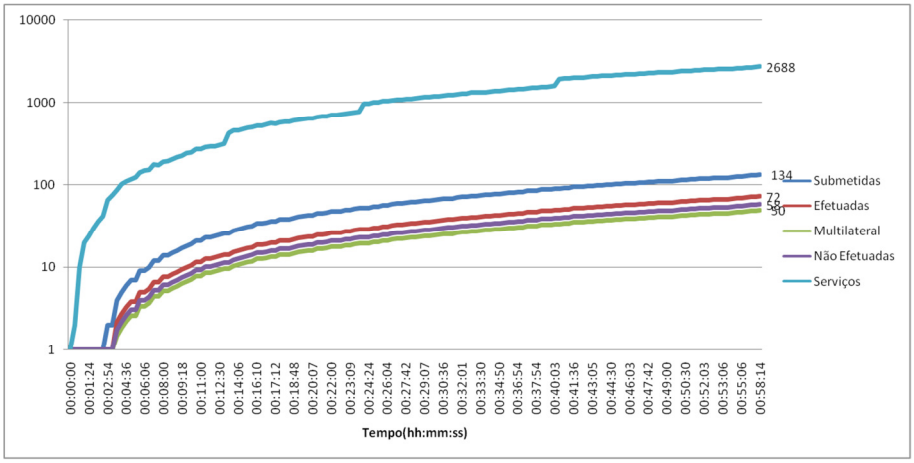

Figura 46 - Comparativo entre quantidade e Permutas e quantidade de serviços levando em consideração o compartilhamento de n tipos de serviços sem limite de tempo

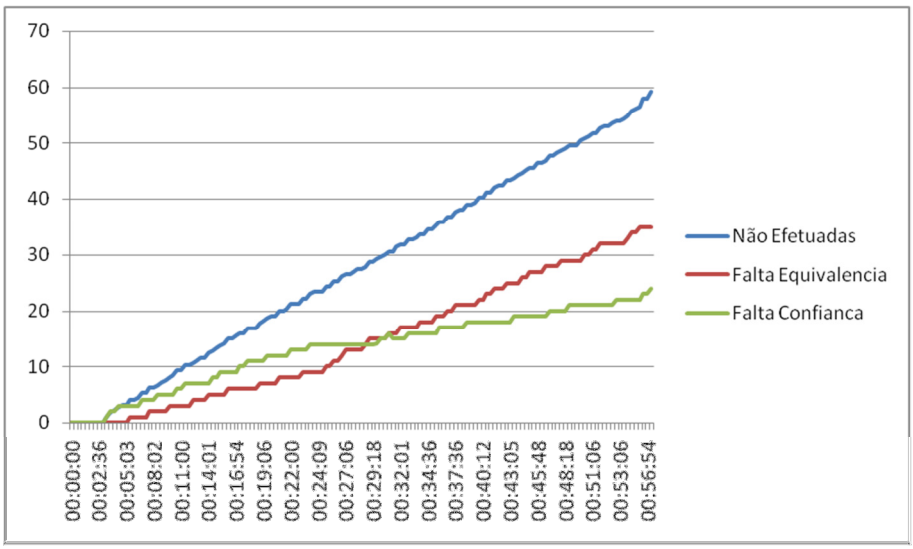

Figura 47 - Comparativo entre os motivos das permutas não efetuadas levando em consideração o compartilhamento de apenas dois tipos de serviços sem limite de tempo

9.6.4 Cenário 4: Compartilhamento de serviço entre membros de uma rede peer-to-peer por meio da permuta bilateral :

A. Com Limitador de tempo de utilização

Neste cenário foram realizados os experimentos para avaliar o desempenho do compartilhamento para dois e para n tipos de serviço na rede P2P, por meio do modelo da permuta bilateral utilizado os períodos de tempo para uso de cada serviço. Como visto na seção 4.1.4, nesse tipo de negociação os serviços permutados não

poderão ser renegociados com terceiros, ficando assim restrito à negociação do objeto de troca com um terceiro negociante. Também para esse cenário, foram efetuadas da análise de confiança e a análise equivalência dos serviços negociados.

Como visto nas nos cenários anteriores, os experimentos foram feitos visando analisar o comportamento do mecanismo de permuta, porém nessa simulação a troca ocorre bilateralmente. A Figura 48 e a Figura 49 apresentam o gráfico do comportamento levando em consideração o número de permutas submetidas. A Figura 50 e a Figura 51 exibem no gráfico um comparativo entre as permutas efetuadas e a quantidade de serviços disponibilizada pelos *peers* para a negociação. Já a Figura 52 e a Figura 53 apresentam o gráfico da relação existente entre os motivos pelos quais não foram efetuadas as trocas

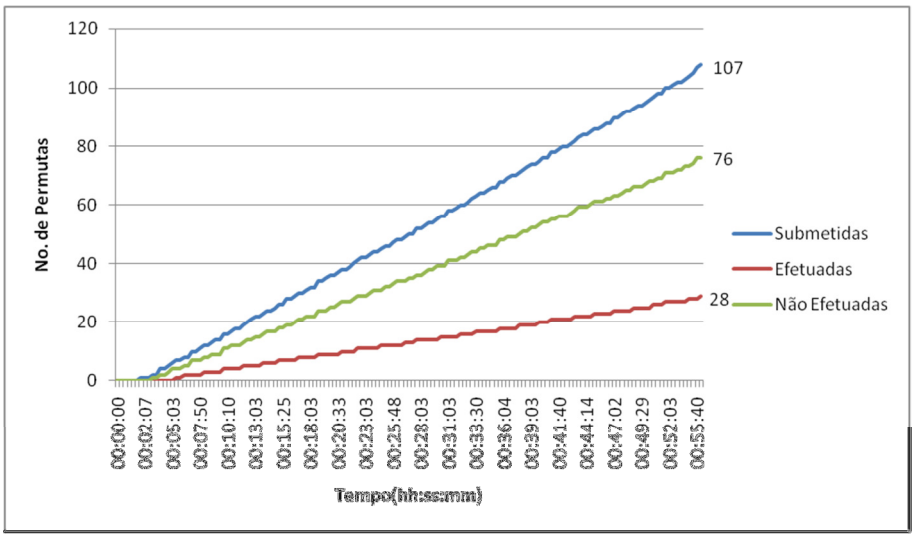

Figura 48 - Evolução da Negociação de Permutas no modelo Bilateral para 2 serviços

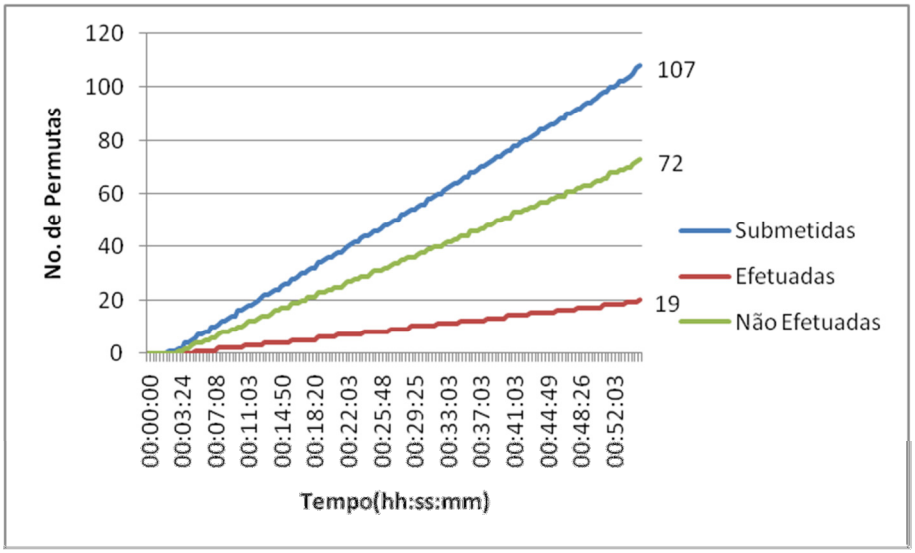

Figura 49 – Evolução da Negociação de Permutas no modelo Bilateral para n serviços

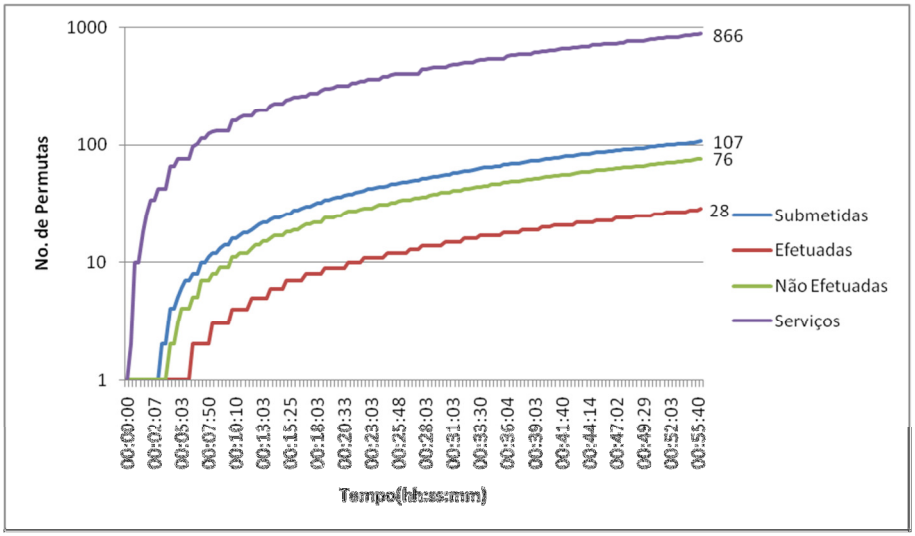

Figura 50 - Comparativo entre a quantidade de permutas e o número de serviços oferecidos no modelo Bilateral para 2 serviços

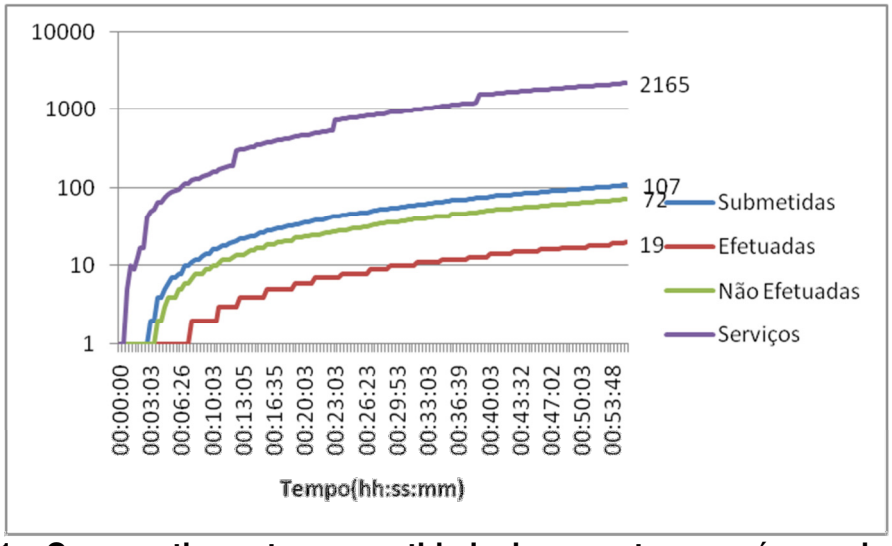

Figura 51 – Comparativo entre a quantidade de permutas e o número de serviços oferecidos no modelo Bilateral para n serviços

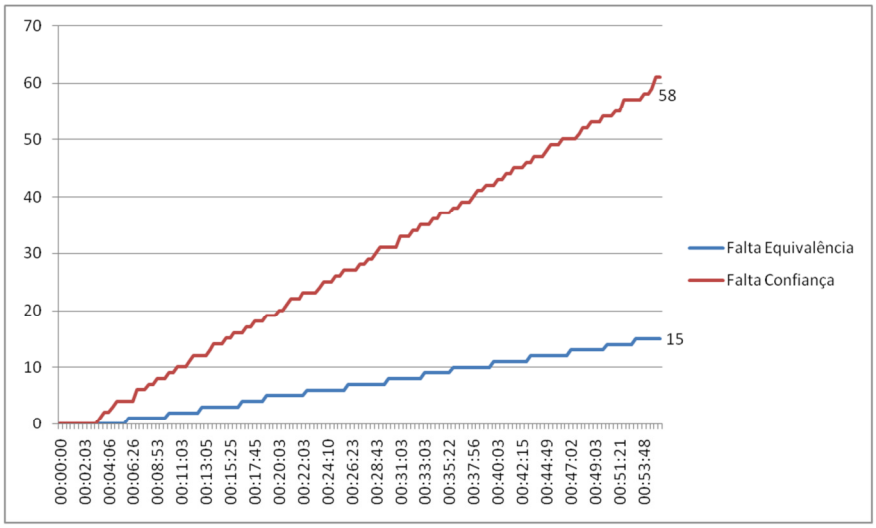

Figura 52 - Motivos de não concretização das permutas no modelo Bilateral para 2 serviços

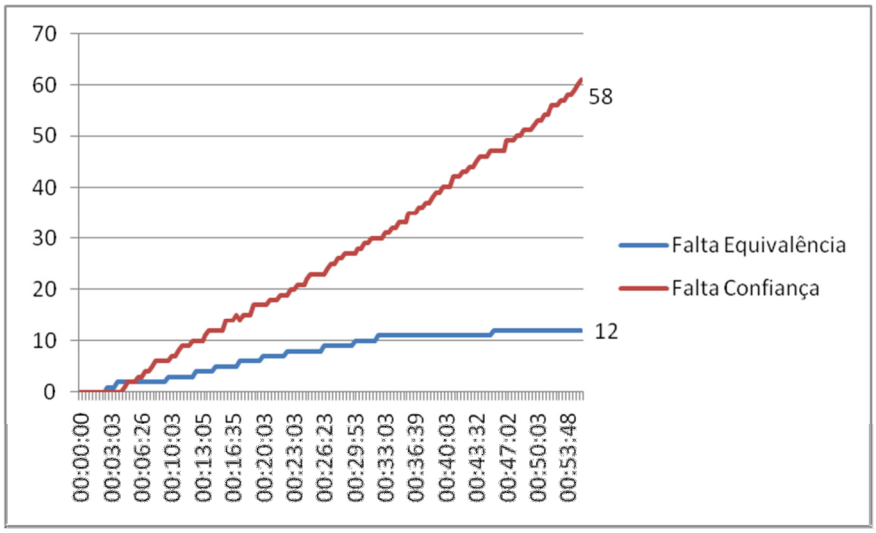

Figura 53 – Motivos de não concretização das permutas no modelo Bilateral para n serviços

B. Sem limitador de disponibilidade tempo de uso

Neste cenário foram realizados os experimentos para avaliar o desempenho do compartilhamento para dois e para n tipos de serviço na rede P2P, por meio do modelo da permuta bilateral. Entretanto, os períodos de tempo para uso de cada serviço forma ignorados nesta simulação. Também para essa simulação, foram efetuadas da análise de confiança e a análise equivalência dos serviços negociados.

Como visto nas nos cenários anteriores, os experimentos foram feitos visando analisar o comportamento do mecanismo de permuta porem nesse cenário a troca ocorre bilateralmente. A Figura 54 e a Figura 55 apresentam o gráfico do comportamento levando em consideração o número de permutas submetidas. A Figura 56 e a Figura 57 exibem no gráfico um comparativo entre as permutas efetuadas e a quantidade de serviços disponibilizada pelos *peers* para a negociação. Já a Figura 58 e a Figura 59 apresentam o gráfico da relação existente entre os motivos pelos quais não foram efetuadas as trocas

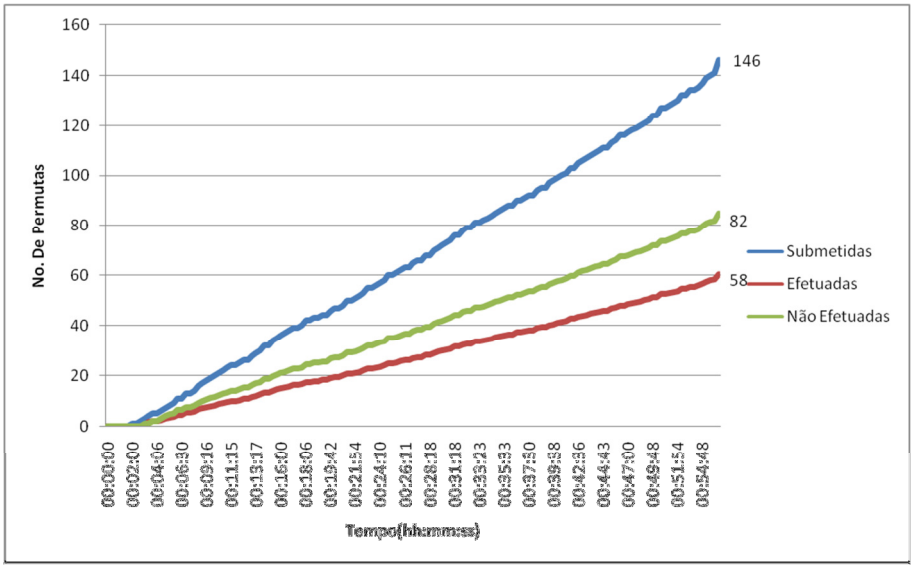

Figura 54 - Evolução da Negociação de Permutas no modelo Bilateral para n serviços sem limites de tempo

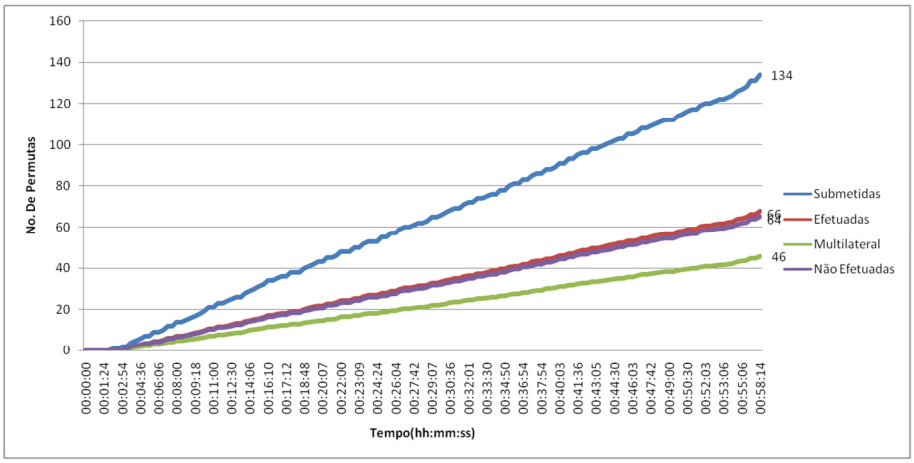

Figura 55 - Evolução da Negociação de Permutas no modelo Bilateral para n serviços sem limite de tempo

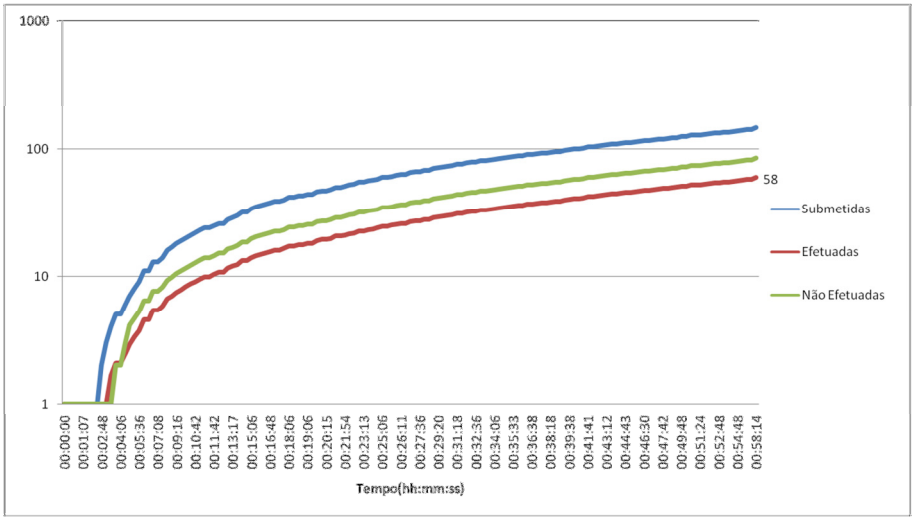

Figura 56 - Comparativo entre a quantidade de permutas e o número de serviços oferecidos no modelo Bilateral para 2 serviços sem limite de tempo

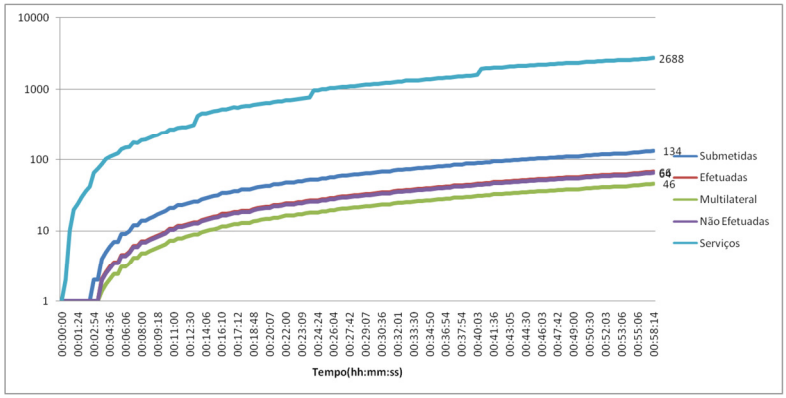

Figura 57 - Comparativo entre a quantidade de permutas e o número de serviços oferecidos no modelo Bilateral para n serviços sem limite de tempo

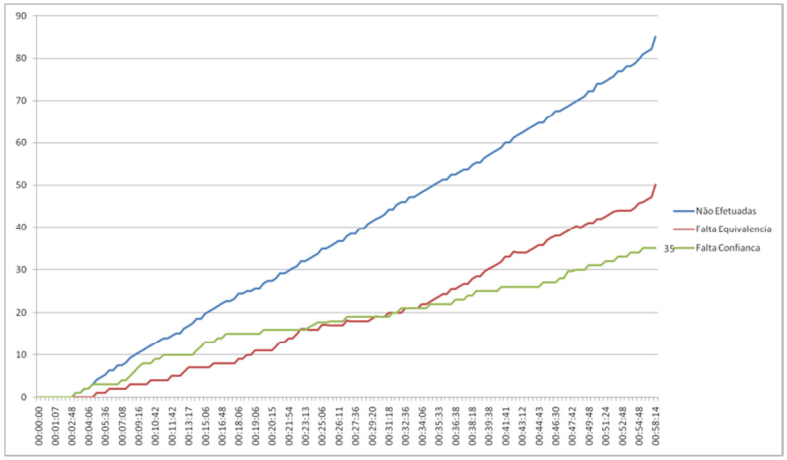

Figura 58 - Motivos de não concretização das permutas no modelo Bilateral para 2 serviços sem limite de tempo

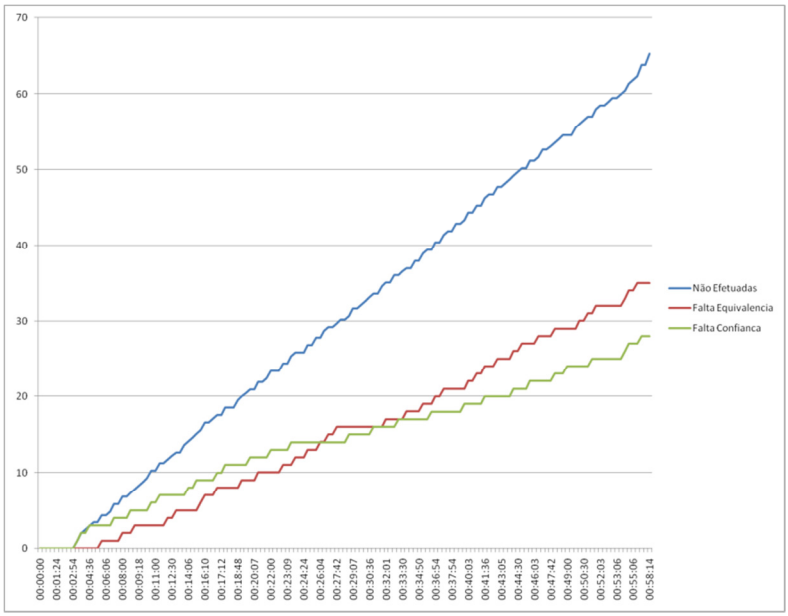

Figura 59 - Motivos de não concretização das permutas no modelo Bilateral para n serviços sem limite de tempo

9.7 ANÁLISE DOS RESULTADOS

A partir dos valores obtidos das simulações, podemos avaliar de forma geral o comportamento da diferenciação dos cenários simulados.

Os resultados apresentados na Tabela 23 demonstram o comportamento das permutas efetuadas, com dois tipos de serviços, nas quatro diferentes simulações executadas (SCSE – Sem Confiança e Sem Equivalência; CCSE – Com Confiança e Sem Equivalência; CCCE– Com Confiança e Com Equivalência; CCCEST– Com Confiança e Com Equivalência Sem Tempo;) do cenário 2 (seção 9.6.2). O gráfico da Figura 60 demonstra a evolução das permutas em cada uma das simulações. Com base nessas duas informações, podemos constatar que à medida que são inseridos mecanismos de confiança e equivalência, propostos nesse trabalho, há um aumento no número de transações de permutas não efetuadas. Em contrapartida, há uma diminuição na quantidade de permutas efetuadas. Esse fato também é verificado nas simulações efetuadas no cenário 3 onde temos simulações com n serviços, como mostram a Tabela 24 e a Figura 61.

Também podemos constatar, com essas informações, que a diferença entre as simulações SCSE e CCCE, no que diz respeito às permutas não efetuadas e às efetuadas são em média, respectivamente, 9,32% e em média de 9.27% levando-se em conta os resultados obtidos nos cenários 2 e 3. Sendo assim, é possível chegar à conclusão de que, à medida que são utilizados os mecanismos de confiança e equivalência, o processo de permuta torna-se mais qualitativo na escolha dos *peers* e dos serviços que serão utilizados nesse processo e, com isso, há um decréscimo o número de transações efetuadas com sucesso.

Vale ressalvar que, em média de 70,80% das permutas efetuadas foram feitas multilateralmente, ou seja, foram concretizadas utilizando-se serviços que já haviam sido transacionados em um processo anterior e que, inicialmente não pertenciam ao provedor do serviço. O provedor detinha apenas o certificado de propriedade do serviço permutado.

Tabela 23 - Comparativos das simulações do Cenário 2

Permuta com 2 serviços								
	Submetidos		Não efetuado		Efetuados		Multilateral	
SCSE	126	100,00%	58	46,03%	65	51,59%	49	38,89%
CCSE	134	100,00%	69	51,49%	62	46,27%	42	31,34%
CCCE	127	100,00%	71	55,91%	55	43,31%	43	33,86%
CCCEST	146	100,00%	55	37,67%	91	62,33%	70	47,95%

Tabela 24 - Comparativos das simulações do Cenário 3

Permutas com N serviços								
	Submetidos		Não efetuado		Efetuados		Multilateral	
SCSE	126	100,00%	67	53,17%	59	46,83%	41	32,54%
CCSE	134	100,00%	78	58,21%	54	40,30%	36	26,87%
CCCE	134	100,00%	83	61,94%	49	36,57%	33	24,63%
CCCEST	134	100,00%	59	44,03%	74	55,22%	50	37,31%

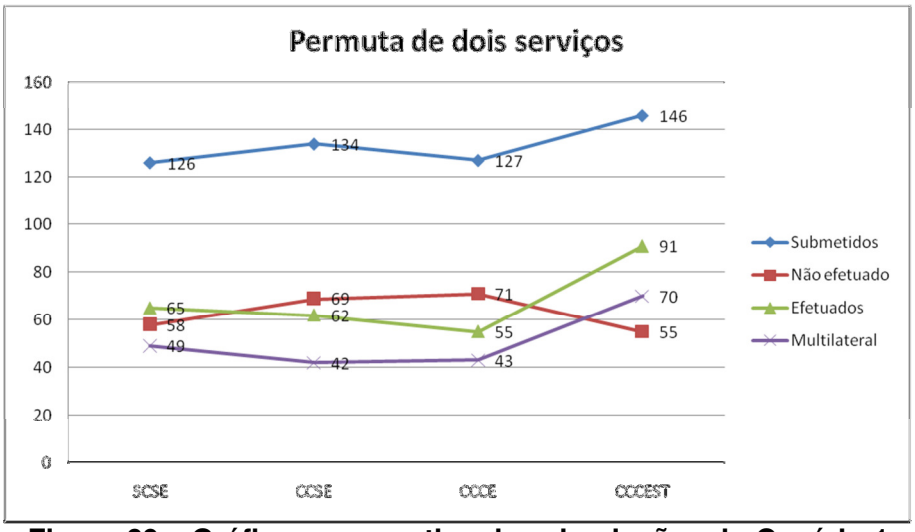

Figura 60 – Gráfico comparativo das simulações do Cenário 1

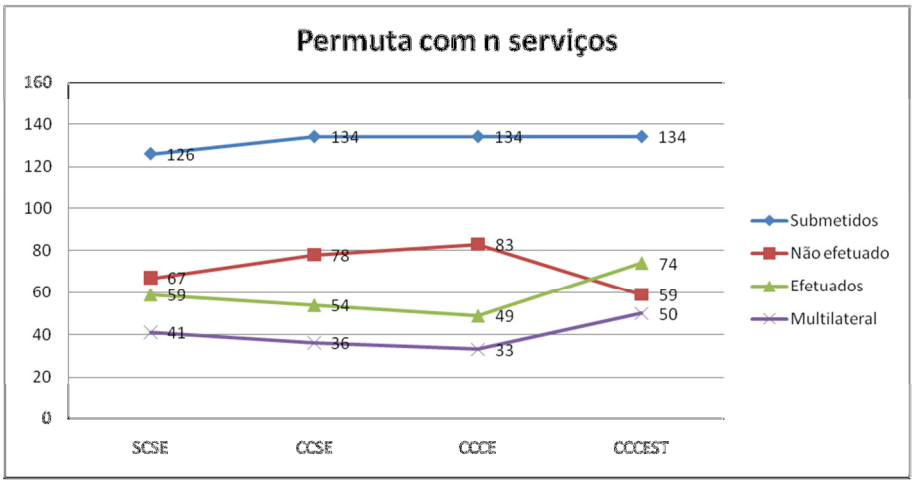

Figura 61 -Gráfico - Comparativos das simulações do Cenário 2

A Tabela 25 apresenta um comparativo entre os quatro diferentes tipos de simulação executada em cada cenário. Nela podemos observar a influência do número de tipos distintos de serviço, disponibilizados pelos *peers*, no processo de permuta. Por meio de observações, é aceitável concluir que com um número maior e diversificado de serviços torna-se mais difícil encontrar *peers* que possuam serviços

disponíveis com as mesmas particularidades que as desejadas por uma determinada transação de troca. A Figura 62, Figura 63, Figura 64 e a Figura 65 demonstram graficamente essa consideração.

Tabela 25 – Comparativo entre os três diferentes tipos de simulação executada em cada cenário

Sem Confiança e sem Equivalência								
	Submetidos		Não efetuado		Efetuados		Multilateral	
Permuta com 2 serviços	126	100,00%	58	46,03%	65	51,59%	49	38,89%
Permutas com N serviços	126	100,00%	67	53,17%	59	46,83%	41	32,54%
Com Confiança e sem Equivalência								
	Submetidos		Não efetuado		Efetuados		Multilateral	
Permuta com 2 serviços	134	100,00%	69	51,49%	62	46,27%	42	31,34%
Permutas com N serviços	134	100,00%	78	58,21%	54	40,30%	36	26,87%
Com Confiança e Com Equivalência								
	Submetidos		Não efetuado		Efetuados		Multilateral	
Permuta com 2 serviços	127	100,00%	71	55,91%	55	43,31%	43	33,86%
Permutas com N serviços	134	100,00%	83	61,94%	49	36,57%	33	24,63%
Com Confiança e Com Equivalência Sem Tempo								
	Submetidos		Não efetuado		Efetuados		Multilateral	
Permuta com 2 serviços	146	100,00%	55	37,67%	91	62,33%	70	47,95%
Permutas com N serviços	134	100,00%	59	44,03%	74	55,22%	50	37,31%

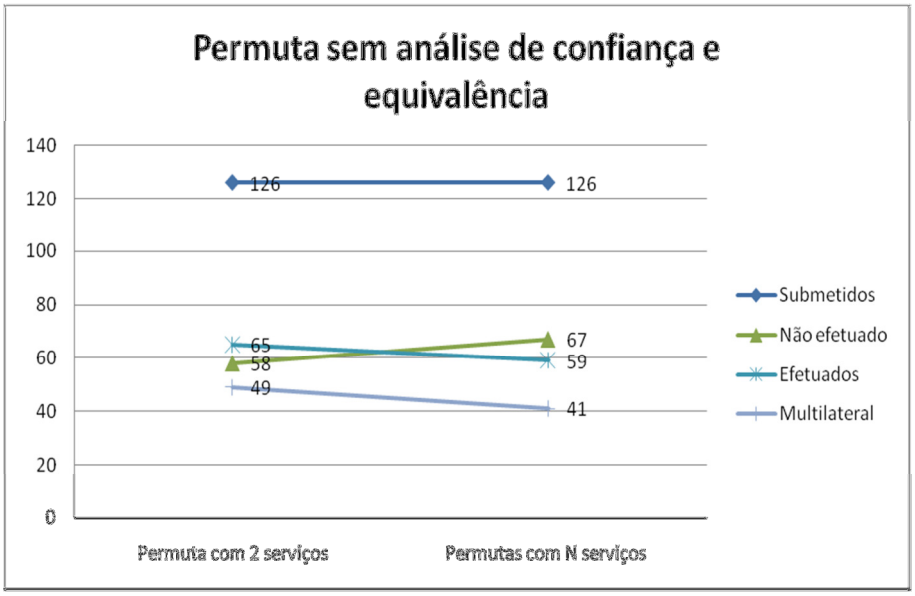

Figura 62 – Comparativo entre Cenários na Simulação SCSE

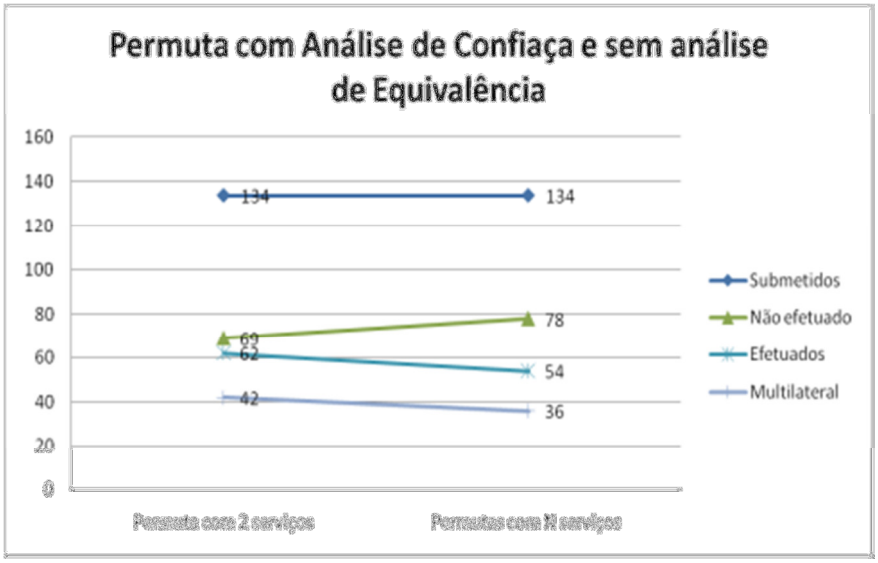

Figura 63 - Comparativo entre Cenários na Simulação CCSE

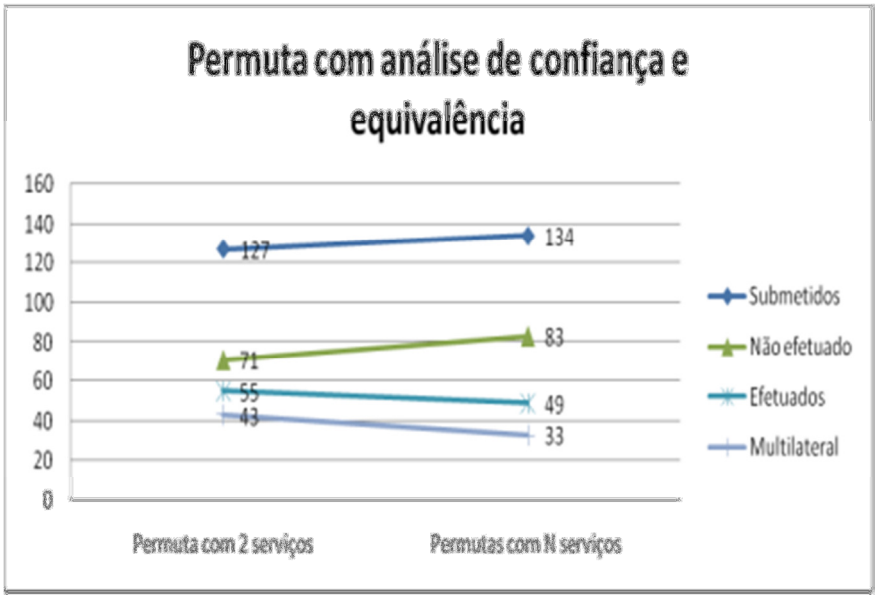

Figura 64 -Comparativo entre Cenários na Simulação CCCE

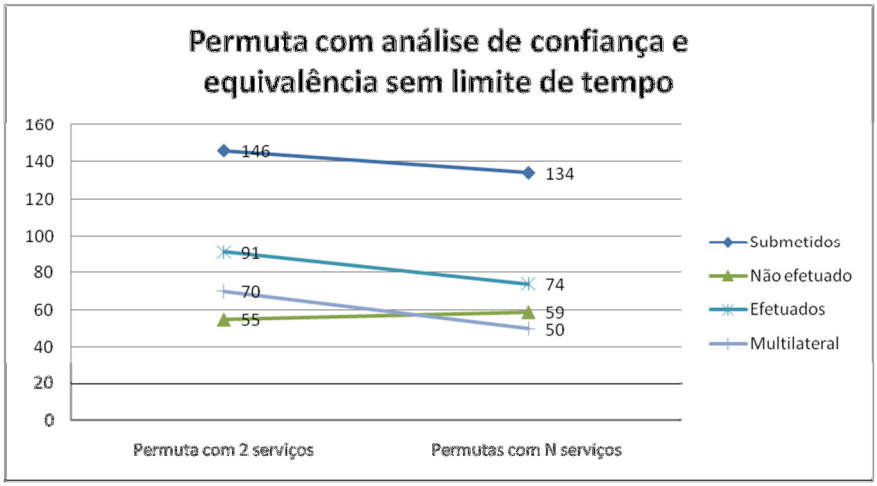

Figura 65 - Comparativo entre Cenários na Simulação CCCEST

Com a finalidade de apresentarmos as diferenças entre a permuta bilateral e a multilateral, a Tabela 26 traz um comparativo entre essas duas modalidades de troca. Podemos constatar que o modelo permuta multilateral mostra-se mais eficiente. É observado que esse fato ocorre, devido à característica da permuta bilateral de que um serviço que já tenha sido permutado, não possa ser negociado com terceiros. Essa característica faz com que um provedor que tenha adquirido a posse de um serviço em transações anteriores não possa negociá-lo em uma nova transação. A Figura 66 apresenta gráfico que ajuda na análise desse comparativo. Já a Tabela 27 apresenta um a comparação também entre essas modalidades de troca, porém leva em consideração os motivos pelos quais não foram efetuadas as permutas. A Figura 67 exibe um gráfico com esse comparativo.

Tabela 26 – Comparativo entre a Permuta bilateral e a Multilateral com análise de confiança e equivalência

	Submetidos		Não efetuados		Efetuados	
Bilateral com 2 serviços	108	100,00%	76	70,37%	29	26,85%
CCCE com 2 serviços	127	100,00%	71	55,91%	55	43,31%
Bilateral com n serviços	108	100,00%	73	67,59%	20	18,52%
CCCE com n serviços	134	100,00%	83	61,94%	49	36,57%

Tabela 27 – Comparativo entre os motivos de não haver a efetuação da permuta

	Não efetuados		Falta de Equivalência		Falta de Confiança	
Bilateral com 2 serviços	76	100,00%	15	19,74%	61	80,26%
CCCE com 2 serviços	71	100,00%	21	29,58%	50	70,42%
Bilateral com n serviços	73	100,00%	12	16,44%	61	83,56%
CCCE com n serviços	83	100,00%	23	27,71%	60	72,29%

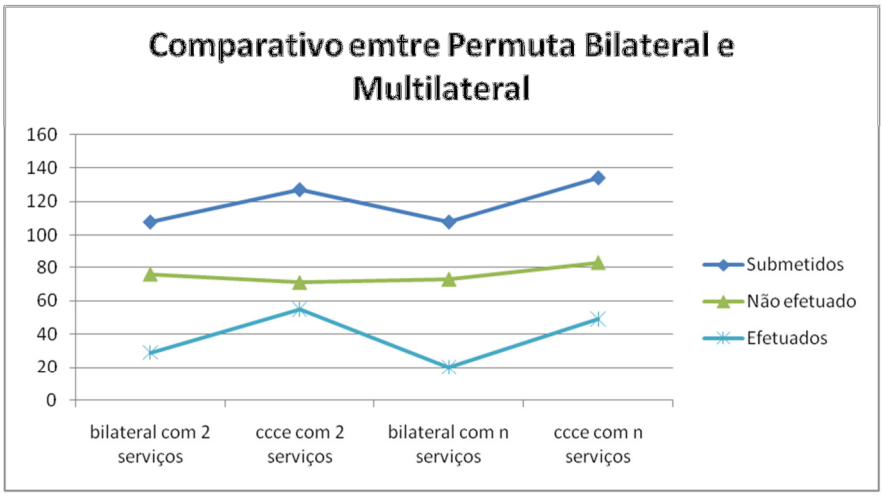

Figura 66 - Comparativo entre a Permuta bilateral e a Multilateral com análise de confiança e equivalência

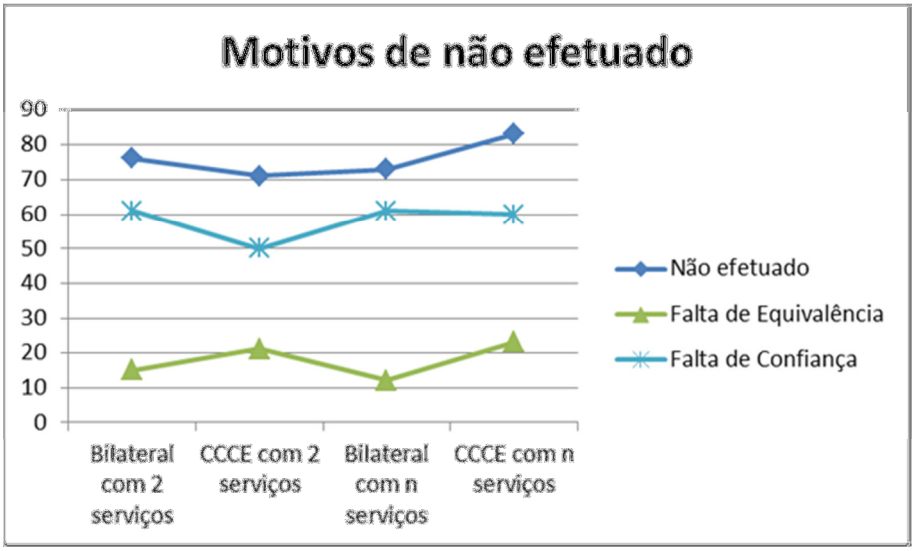

Figura 67 – Coparativos dos motivos de não haver a permuta

9.8 CONSIDERAÇÕES FINAIS

Neste capítulo, foi primeiramente avaliado o algoritmo para estruturação de redes *peer-to-peer*, dada a importância do mesmo no que diz respeito ao impacto do mesmo sobre a infraestrutura de comunicação utilizada para interconectar computadores em ambientes computacionais de larga escala. Logo após, foram avaliadas aa políticas de computação distribuída tendo como estudo o compartilhamento de serviços utilizando-se o modelo de permuta multilateral proposta nesta tese. A avaliação destas políticas teve como objetivo principal obter resultados que pudessem apresentar os prós e contras do uso do modelo permuta multilateral como uma alternativa para compartilhamento de recursos na rede P2P. Assim, é possível concluir que, por meio de uma infraestrutura computacional dinâmica como é uma rede *peer-to-peer* não estruturada, é possível formar redes de trocas. Um ponto negativo, com relação ao uso de redes *peer-to-peer* para compartilhamento de serviços, é a dependência da colaboração que deve haver entre os membros de uma rede. Caso isto não ocorra, o uso de redes *peer-to-peer* para a finalidade proposta neste trabalho fica comprometido. Mesmo que usuários tenham interesses em comum e agreguem uma grande quantidade de recursos compartilhados, isto não é suficiente para haver permuta, pois os membros de uma rede social *peer-to-peer* devem realmente ser capazes de colaborar.

10 CONCLUSÃO

Com a crescente demanda por maior poder de processamento, a computação em grade tem um potencial de crescimento bastante grande. Esse trabalho tem como referência uma arquitetura para compartilhamento de serviços, em ambientes P2P, utilizando o modelo econômico baseado em troca. A arquitetura apresentada nesse projeto trata das questões de confiança e também do controle do comportamento egoísta dos participantes.

Para promover o crescimento da rede P2P, é necessário enfrentar os problemas inerentes à arquitetura distribuída. A existência de participantes independentes, geograficamente distribuídos e sob administrações diferentes, propicia um grande número de questões que devem ser tratadas.

A inserção de modelos econômicos, na troca de recursos, colabora de forma significativa com o poder computacional da rede P2P, levando-se em consideração que cada participante tem a liberdade e flexibilidade para administrar seus recursos da maneira que melhor lhe convier.

O problema existente no fato de um único participante, muitas vezes, ser incapaz de oferecer recursos capazes de gerar a quantidade de créditos necessária para a execução de uma aplicação que consuma um valor de créditos bastante grande foi solucionado por meio da inserção do conceito de grupos de trabalho. Dessa forma, uma determinada instituição pode organizar seu grupo de trabalho de maneira a atender suas necessidades e garantir que os recursos remotos necessários possam ser utilizados.

10.1 PRINCIPAIS CONTRIBUIÇÕES

Esta tese apresentou um estudo sobre o uso do modelo econômico de permuta multilateral para compartilhamento de serviços em rede *peer-to-peer.* O objetivo foi atingido por meio de políticas de computação distribuída, executadas sobre uma infraestrutura computacional formada por uma rede *peer-to-peer* não estruturada. A partir destas políticas, foi possível criar redes *peer-to-peer*, agregar serviços compartilhados pelos membros da rede, e executar aplicações de cargas de trabalho divisíveis.

As principais contribuições do sistema proposto podem ser sintetizadas sob dois aspectos. Do ponto de vista de projeto, a proposta oferece um novo

paradigma para o compartilhamento de serviço, baseado no modelo econômico de permuta multilateral, e trazendo todos os benefícios de flexibilidade e interoperabilidade inerentes a essa abordagem. Do ponto de vista de um sistema de *middleware*, a proposta procura incorporar todas as características e funcionalidades necessárias, tanto para as aplicações clientes, como capacidades de inspeção e ciência de contexto, como para os desenvolvedores dessas aplicações, livrando-os da tarefa de lidar com decisões de infraestrutura e assim facilitando, assim, o seu trabalho.

Nesse sentido, as principais contribuições são:

a. Políticas de computação distribuída baseadas em redes *peer-to-peer* para compartilhamento de serviços. Estas políticas permitem a criação de redes sobre uma infraestrutura computacional formada por uma rede *peer-to-peer* não estruturada. Além disso, possibilitam a formação de um ambiente computacional cuja escala aumenta à medida que novos membros são admitidos na rede. O ambiente computacional é formado a partir da agregação dos recursos computacionais compartilhados pelos membros de uma rede *peer-to-peer*. Isto permite que novos recursos sejam adicionados, sem que o ambiente seja reconfigurado manualmente, ou seja, a capacidade computacional é aumentada de maneira transparente e sem a intervenção humana no que tange à configuração do ambiente computacional. Sobre o agregado de recursos computacionais, as políticas oferecem meios para que serviços sejam compartilhados e armazenados de maneira colaborativa, permitindo que informações comuns aos membros da rede estejam sempre disponíveis, anão ser que todos os membros da rede se mantenham desconectados. Por fim, as políticas ainda oferecem meios de uma aplicação de carga de trabalho divisível ser executada sobre os compartilhamentos de tempos de ocupação de processadores e espaços de memória.
b. Proposta de uma arquitetura para implementação de um *middleware* baseado nas políticas de computação distribuída baseadas em redes P2P. A arquitetura é responsável por disponibilizar interfaces e componentes que permitam a criação de aplicações de computação distribuída, de modo que estas possam tirar proveito da infraestrutura computacional criada a partir dos compartilhamentos realizados por

usuários, no momento da formação e, posteriormente, na evolução de uma rede*peer-to-peer*.

c. Resultados experimentais de desempenho das políticas de computação distribuída baseadas em rede. Os experimentos realizados com o simulador de redes mostraram que é possível por meio de redes peer-to-*peer*, criar ambientes computacionais distribuídos de grande escala.
d. Implementação de um simulador de redes *peer-to-peer*, onde algoritmos de redes *peer-to-peer* podem ser testados antes de serem utilizados na implementação de uma aplicação ou *middleware* de computação distribuída. Além disso, o simulador também oferece suporte à prototipação e simulação de políticas de computação distribuída de maneira fácil, pois não é necessário investir uma grande quantidade de tempo para aprender e usar o conjunto de componentes de software existentes no simulador.

Apesar de a abordagem proposta apresentar uma grande flexibilidade no que tange ao aumento da escala de recursos para ambientes computacionais, esta é frágil com relação ao modelo utilizado para compor a infraestrutura de computação, pois redes *peer-to-peer* normalmente são instáveis. No entanto, pesquisadores se comprometem uns com os outros, quando todos buscam um objetivo comum, diferente do susuários de sistemas *peer-to-peer* para compartilhamento de serviços.

10.2 TRABALHOS FUTUROS

Os trabalhos futuros a serem realizados serão conduzidos ao longo de diferentes etapas. A seguir são detalhadas as principais atividades a serem realizadas em cada etapa e suas metas.

10.2.1 Plataforma de Computação *Peer-to-Peer* Orientada a Objetos Móveis

Esta fase compreenderá o desenvolvimento de uma plataforma de computação *peer-to-peer* orientada a serviços móveis. Uma das motivações para este trabalho futuro é desenvolver aplicações distribuídas e isso é muito complexo, pois o desenvolvedor precisa ter um grande conhecimento, tanto teórico quanto prático, no que diz respeito a tipos de redes *peer-to-peer* e seus respectivos protocolos;

Por este motivo, é proposto o desenvolvimento de uma plataforma de computação *peer-to-peer*, que além de oferecer todos os mecanismos para provisão de computação *peer-to-peer*, também é orientada a objetos móveis. Por isto, serviços de aplicações podem ser executados de maneira distribuída em computadores, onde tais serviços não foram configurados. Com isto, funcionalidades de aplicações que precisam lidar com o processamento intensivo de dados podem ser desenvolvidas utilizando serviços móveis. Isto deve permitir que os objetos móveis executem sobre um conjunto de compartilhamentos de tempos de processamento e espaços de memória, sem haver intervenção humana no processo de configuração do ambiente computacional e execução dos serviços de aplicações.

Para viabilizar a proposta de desenvolvimento dessa plataforma, o modelo arquitetural de software apresentado no Capítulo 6 servirá como especificação de base para o projeto e desenvolvimento da plataforma. Como mencionado anteriormente, o modelo arquitetural proposto segue o padrão arquitetural *microkernel*, a fim de possibilitar a inversão de dependência no que diz respeito aos serviços básicos e serviços de aplicações. Entre estes dois tipos de serviços não deve existir acoplamento forte na configuração das dependências entre um e outro. Por este motivo, cada camada do *microkernel* e seus respectivos componentes deverão oferecer interfaces bem definidas, a fim de possibilitar a adaptação da plataforma a requisitos relacionados aos ambientes de rede P2P.

10.2.2 Políticas de Computação Distribuída Baseadas em Redes P2P como Serviços de Plataforma

Uma vez implementada a plataforma de computação *peer-to-peer* orientada a serviços móveis, o principal objetivo desta etapa é introduzir as políticas de computação distribuída na forma de serviços básicos de plataforma. Portanto, ficarão previstas para esta etapa as seguintes atividades:

- Implementação de serviços de registro de usuários no sistema distribuído;
- Implementação de serviços de identificação e autorização de usuários no sistema distribuído;
- Implementação das políticas de computação distribuída baseadas em redes P2P.

- Execução das políticas em um ambiente computacional real, por meio de protótipos de aplicações desenvolvidas sobre a plataforma de computação.
-

10.2.3 Refinamento das Políticas de Computação Distribuída Baseada em Redes P2P

Após a coleta dos resultados obtidos durante a execução em um ambiente computacional real, o principal objetivo desta etapa é avaliar o desempenho das políticas de computação distribuída baseadas em redes sociais. De antemão, propomos estudos em duas frentes:

- **Sistemas de recomendação –** estudos e propostas nesta área melhorariam os mecanismos de criação de redes P2P e a admissão de membros nestas redes. Com isto, novos algoritmos de combinação das confianças podem ser desenvolvidos, com o objetivo de trazer maior robustez no processo de análise de compatibilidade entre restrições de confiança dos serviços ofertados pelo usuários do sistema distribuído;
- **Balanceamento de carga –** estudos e propostas nesta área colaborariam para a melhoria da qualidade de serviço da plataforma. A partir disto, algoritmos e estratégias para balanceamento de carga podem ser propostos para otimizar a obtenção de grandes volumes de dados compartilhados em uma rede *peer-to-peer* e a divisão das cargas de trabalho processadas sobre compartilhamentos de ciclos de processadores e espaços de memória.

11 BIBLIOGRAFIA

ADAR, E. A.; HUBERMAN, B. A. Free Riding on Gnutella. **Peer-Reviewed Journal of the Internet**, Palo Alto, 5(10), 2000.

AMIR , ; DANILOV, C. Reliable Communication in Overlay Networks, 2003. 511-520.

ANCEAUME, E. et al. PeerCUBE: An HyperCube-Based P2P Overlay Robust Against Collusion A Churn, Rennes, 2008.

ANDERSEN, D. et al. **Resilient Overlay Networks**. Massachusetts Institute of Technology. Boston, Massachusetts, p. 86. 2001.

ANDRADE, N. et al. **Ourgrid:** An approach to easily assemble grids with equitable resource sharing. the 9th Workshop on Job Scheduling Strategies for Parallel Processing. [S.l.]: [s.n.]. 2003.

ANDRADE, N. et al. **Discouraging Free-riding in a Peer-to-Peer Grid**. Proceedings of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing. [S.l.]: [s.n.]. 2004.

APPLEBY, K. et al. **Oceano - SLA Based Management of a Computing Utility**. International Symposium on Integrated Network Management Proceedings. [S.l.]: [s.n.]. 2001. p. 855-868.

ATC. Ensuring dependability of P2P applications at architectural level. **P2P_Architect project**, 30 abr. 2002. Disponivel em: <http://www.atc.gr/p2p_architect/results/0101F05_P2P Survey.pdf>. Acesso em: 20 nov. 2007.

BARCELLOS, A. M.; GASPARY, L. P. **Segurança em redes P2P:** princípios, tecnologias e desafios. Simposio Brasileiro de Redes de Computadores. Curitiba, PR: [s.n.]. 2006. p. 211-260.

BARHAM, P. et al. **Xen and the art of virtualization**. the ACM Symposium on Operating Systems Principles (SOSP). [S.l.]: [s.n.]. 2003.

BONA, L. C. E. D. **HyperBone: Uma Rede Overlay Baseada em Hipercubo Virtual para Computação Distribuida na Internet**. UTFPR/ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL. Curitiba, p. 187. 2006.

BOSS, G. et al. Cloud Computing. **Lanzhou University**, 2007. Disponivel em: <grid.lzu.edu.cn/resource/article_download.jsp?id=33>. Acesso em: 13 out. 2009.

BUNKER, G.; THOMSON, D. **Delivering Utility Computing:** Business-driven IT Optimization. [S.l.]: Wiley, 2006.

BUSCHMANN, et al. **Pattern-Oriented Software Architecture:** A System of Patterns. [S.l.]: John Wiey & Sons, v. 1, 2001.

BUYYA, R. **Service-Oriented Grid Architecture for Distributed Computational Economies**. Monash University. Melbourne. 2002.

BUYYA, R.; ABRAMSON, D.; GIDDY, J. **An Economy Driven Resource Management Architecture for Global Computational Power Grids**. The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000). Las Vegas, USA: [s.n.]. 2000.

BUYYA, R.; ABRAMSON, D.; GIDDY, J. **Nimrod/G:** An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000). Beijing, China: IEEE Computer Society Pres. 2000.

CENDRON, M. M. **Escalonamento de Tarefas Baseado Em Leílão no Middlare Grid-M**. Universidade Federal de Santa Catarina. Florianópolis- SC. 2008.

CHAPELL, D. A Short introdution to cloud platforms: an enterprise-oriented view. **David Chappell & Associates**, ago. 2008. Disponivel em: <http://www.davidchappell.com/CloudPlatforms--Chappell.pdf>. Acesso em: 15 jan. 2009.

CHENG, A.; FRIEDMAN, E. **Sybilproof reputation**. P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems. New York, NY, USA: ACM. 2005. p. 128–132.

CIRNE, W. et al. **Running bag-of-tasks applications on computational grids:** The mygrid approach. the ICCP'2003 - International. [S.l.]: [s.n.]. 2003.

CIRNE, W.; MARZULLO, K. **The computational Co-op:** Gathering clusters into a metacomputer. IPPS/SPDP'99 Symposium. [S.l.]: [s.n.]. 1999. p. 160 - 166.

COELHO , U. **Curso de Direito Comercial**. São Paulo: Saraiva, 2000.

COOPER, B. F.; GARCIA-MOLINA, H. **Bidding for storage space in a peer-to-peer data preservation system**. Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002). Vienna: IEEE Computer Society Press. 2002. p. 372-381.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems:** Concepts and Design. [S.l.]: Addison-Wesley, 2006.

DAIUTO, R. R. **Introdução ao estudo do contrato**. São Paulo: Atlas, 1995.

DELLAROCAS, C. S. The Digitization of Word-Of-Mouth: Promise and Challenges of Online Reputation Mechanisms. **Management Science**, p. 1407-1424, 2003.

DOVAL, D.; O'MAHONY,. Overlay networks: A scalable alternative for P2P. **Internet Computing**, 7, 2003.

DUMITRIU, D. et al. Denial-of-service resilience in peer-to-peer file sharing systems. **Denial-of-service resilience in peer-to-peer file sharing systems ACM SIGMETRICS'05**, 2005.

EBAY. Ebay. **Ebay**, 1995. Disponivel em: <http://www.ebay.com/>. Acesso em: 20 fev. 2008.

FEDERAL UNIVERSITY OF CAMPINA GRANDE. OurGrid, 2003. Disponivel em: <http://www.ourgrid.org>. Acesso em: 20 Fev 2008.

FELDMAN, M.; CHUANG, J. **Overcoming free-riding behavior in peer-to-peer systems**. ACM SIGecom Exchanges. New York: [s.n.]. 2005. p. 41-50.

FERGUSON, D. F. et al. Economic models for allocating resources in computer system, S. Clearwater, 1996.

FISCHER, M.; LYNCH, N.; PATERSON, M. Impossibility of Distributed Consensus with One Faulty Process, 32, 1985.

FOSTER, I.; IAMNITCHI, A. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In: _____ **Peer-to-Peer Systems II**. [S.l.]: [s.n.], v. 2735/2003, 2003.

GAMMA, E. et al. **Padrões de projeto:** soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2001.

GENÇ, Z. SFG: smart flooding by gossiping, Toulouse, 2005. 246-247.

GOLBECK, J.; HENDLER, J. **Inferring Binary Trust Relationships in Web-based Social Networks**. ACM Transactions on Internet Technology. [S.l.]: [s.n.]. 2006. p. 497–529.

GOLBECK, J.; HENDLER, J. Inferring Trust Relationships in Web-based Social Networks. **ACM Transactions on Internet Technology**, November 2006. 497 - 529.

GONÇALVES, C. A. **Direito civil brasileiro**. Sào Paulo: Saraiva, v. 3, 1999.

GONÇALVES, F. B. **UM ESTUDO SOBRE O USO DE REDES P2P SOCIAIS PARA O COMPARTILHAMENTO DE RECURSOS EM AMBIENTES DE E-SCIENCE**. NCE/UFRJ. Rio de Janeiro, p. 156. 2010.

GRANDISON, T.; SLOMAN, M. A survey of trust in internet applications. **IEEE Communications Surveys,** 2000.

GRIZARD, A. et al. **A peer-to-peer normative system to achieve social order**. Autonomous Agents and Multi-Agent Systems. [S.l.]: [s.n.]. 2006.

GUPTA, M.; JUDGE, P.; AMMAR, M. **A Reputation System for PeertoPeer Networks**. NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video. New York: ACM Press. 2003. p. 144–152.

HP Grid Computing. **HP**. Disponivel em: <http://www.hp.com/techservers/grid>. Acesso em: 20 jan. 2008.

HUBERMAN, B. A.; ADAR, E. Free Riding on Gnutella. **First Monday (Peer-Reviewed Journal of the Internet),** 2000.

IBM. MojoNation project, 02 ago. 2000. Disponivel em: <http://freshmeat.net/projects/mojonation/>. Acesso em: 28 jan. 2008.

IBM GRID computing. Disponivel em: < >. Acesso em: 20 dez. 2007.

IRTF. Peer-to-Peer Research Group. **Internet Research Task Force**, 29 set. 2006. Disponivel em: <http://www.irtf.org/charter?gtype=rg&group=p2prg>. Acesso em: 03 dez. 2007.

JENSEN, C.; DAVIS, J.; FARNHAM, S. **Finding Others Online:** Reputation Systems for Social Online Spaces. the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. Conference on Human Factors in Computing Systems. Minneapolis, Minnesota: [s.n.]. 2002.

JOSANG, A.; ISMAIL, R.; BOYD, C. A Survey of Trust and Reputation Systems for Online Service Provision. **Decision Support Systems**, 43 (2), 2006. Disponivel em: <http://sky.fit.qut.edu.au/~josang/papers/JIB2007-DSS.pdf>. Acesso em: 20 abr. 2008.

KELLY, K. New Rules for the New Economy. **New Rules for the New Economy**, 11 mar. 2003. Disponivel em: <http://www.kk.org/newrules/index.php>. Acesso em: 14 dez. 2007.

KENYON, C.; CHELIOTIS, G. Grid resource management: state of the art and future trends. In: _____ **Grid Resource Management:** State of the Art and Research Issues. Norwell, MA, USA: Kluwer Academic Publishers, 2003. p. 465 - 478.

KUROSE, J. F.; SIMHA, R. A microeconomic approach to optimal resource allocation indistributed computer systems, 38, n. 5, May 1989. 705-717.

LAI, K. et al. Tycoon: An implementation of a distributed, market-based resource allocation system. **Multiagent and Grid Systems**, 1, n. 3, Aug 2005. 1574-1702.

LEUF, B. **Peer to Peer:** Collaboration and Sharing over the Internet. Bouston: Addison-Wesley Longman Publishing, 2002.

LIU, H.; QIU, Y. **A Reputation Model base on Transactions in Peer-to-Peer networks**. Third International Conference on Semantics Knowledge and Grid. Washington, DC, USA: IEEE Computer Society. 2007. p. 398-401.

LIV, Q. et al. Search and Replication in Unstructured Peer-to-Peer Networks, New York, 2002.

LIVRE, M. Mercado Livre. **Mercado Livre**, 1998. Disponivel em: <http://mercadolivre.com.br>. Acesso em: 20 fev. 2008.

MARTI, S.; GARCIA-MOLINA, H. Taxonomy of trust: Categorizing p2p reputation systems. **Computer Networks**, 2006. 472–484.

MARTINS, V.; PACITTI, E.; VALDURIES, P. **Survey of data replication in P2P systems**. [S.l.]. 2007.

MATTOSO, M. et al. **Gerenciando Experimentos Científicos em Larga Escala**. SEMISH - Seminário Integrado de Software e Hardware. Belém: [s.n.]. 2008.

MCKNIGHT, L. W.; HOWISON, J. **Towards a Sharing Level Protocol for Distributed**. International Conference on Computer, Communication. Orlando, Florida: [s.n.]. 2003.

MILOJICIC, D. S. et al. **Peer-to-Peer Computing**. Palo Alto. 2003.

MOURA, L. G. L. pydssim - Distributed System Simulator in Python. **Google Code**, 05 jan. 2009. Disponivel em: <http://code.google.com/p/pydssim/>. Acesso em: 15 jun. 2010.

MOURA, L. G. L. et al. **A Software Architecture for Provisioning of Mobile Services in Peer-to-Peer Environments**. The Second International Conference on Internet and Web Applications and Services. Mauritius: [s.n.]. 2007. p. 13-19.

MOURA, L. G. L. et al. **An Architectural Model for Applications Based on Mobile Services**. International Multi-Conference on Computing in the Global Information Technology. Guadaloupe: [s.n.]. 2007.

MOURA, L. G.; OLIVEIRA, C. E. T.; FRANCA, F. M. G. **The Use of Reciprocal Trade as a Model of Sharing Resources in P2P Networks**. Fifth International Conference on Networking and Services. [S.l.]: [s.n.]. 2009. p. 91-96.

MUI, L.; HALBERSTADT, A.; MOHTASHEMI, M. **Notions of Reputation in Multi-Agents Systems:** A Review. Proceedings of the first international joint conference on Autonomous agents and multiagent systems. Bologna, Italy: [s.n.]. 2002. p. 280-287.

MUTKA, M. W.; LIVNY, M. Profiling Workstations Available Capacity for Remote Execution, Amsterdam, 1987. 529-544.

MUTKA, M. W.; LIVNY, M. The available capacity of a privately owned workstation environment. **Elsevier Science Publishers B. V.**, 12, 1991. 269 - 284.

NARAYANAN, D. **Operating System Support For Mobile Interactive Applications**. Carnegie Mellon University. Pittsburgh, PA. 2002.

NASCIMENTO , S. R. D. Algoritmos distribuidos para localização de falhas e difusão de mensagens em hipercubos defeituosos, Campinas, SP, 2000.

PARAMESWARAN, M.; SUSARLA, A.; WHINSTON, A. B. P2P Networking: An Information-Sharing Alternative. **IEEE Computer**, 2001. 31-38.

PARASHAR, M.; HARIRI,. Autonomic Computing: An Overview. In: _____ **Unconventional Programming Paradigms**. [S.l.]: Springer Verlag, 2005.

PELISSARI, F. R.; RIGHI, R. D. R.; WESTPHALL, C. M. **RBRP:** Protocolo de Reputação Baseado em Papéis para Redes P2P. International Information Technologies Symposium. São Carlos, SP: [s.n.]. 2004.

PETERSON, L. L.; DAVIE, B. S. **Computer Networks:** A Systems Approach. [S.l.]: Morgan Kaufmann Publishers Inc., 2003.

PIRO, R. M. **Simulation of Economy-based Load Balancing in Computational Grid for Large-Scale Scientific Applications**. University of Torino. Torino. 2003.

PORTNOI, M. **Network Simulator – Visão Geral da Ferramenta de Simulação de Redes**. Universidade de Salvador. Bahia. 2002.

QBONE. QBone. **QBone Home Page**, 2005. Acesso em: 15 jul. 2009.

RAJKUMAR, B. et al. **Economic Models for Management of Resources in Peer-to-Peer and Grid Computing**. SPIE Int. Conf. on Commercial Applications for High-Performance Computing. Denver: [s.n.]. 2001.

RAPPA, M. A. The utility business model and the future of computing services. **IBM Systems Journal**, 43, n. 1, 2004. 32-42.

RATNASAMY, S. et al. A Scalable Content-Addressable Network, San Diego, California, 2001. 161-172.

REDDY, S. R. **Market Economy Based Resource Allocation in Grids**. School of Information Technology. Kharagpur. 2006.

RESNICK, P. et al. Reputation Systems. **Communications of the ACM**, 43(12), 2000. 45-48.

RIBEIRO, A. B. ESTUDO MICROECONÔMICO: A MICROECONOMIA NO CONTEXTO DA TEORIA ECONÔMICA. **Blog André Gilberto Boelter Ribeiro**, 15 jan. 2008. Disponivel em: <http://andreboelter.blogspot.com/2008/01/estudo-microeconmico-microeconomia-no.html>. Acesso em: 10 mar. 2008.

ROBINSON, P.; VOGT, H.; WAGEALLA, W. Some Research Challenges in Pervasive Computing. In: _____ **Privacy, Security and Trust within the Context of Pervasive Computing**. [S.l.]: Springer US, 2005. p. 1-16.

SABATER, J.; SIERRA, C. Social ReGreT, a reputation model based on social relations. **ACM SIGecom Exchanges**, 2001. 44 – 56.

SAROIU, S.; GUMMADI, P. K.; GRIBBLE, S. D. **A Measurement Study of Peer-to-Peer File Sharing Systems**. Seattle, WA. 2002.

SILVA, I. P. **Um Modelo de Replicação para Localização de Dados em Ambientes P2P com Mobilidade**. Rio de Janeiro. 2008.

STOICA, I. et al. Chord: A Scalable peer-to-peer lookup service for internet applications, San Diego, California, 2001. 149-160.

SUBRAMONIAM, K.; MAHESWARAN, M.; TOULOUSE, M. **Towards a micro-economic model for resource allocation in Grid computing systems**. IEEE International Symposium on High Performance Distributed Computing. [S.l.]: [s.n.]. 2002. p. 782-785.

SUN MICROSYSTEMS. Network.com., 2007. Disponivel em: <http://www.network.com>. Acesso em: 10 fev 2008.

TANEMBAUM, ; VAN STEEN,. **Sistemas distribuídos:** princípios e paradigmas. São Paulo: Pearson Prentice Hall, 2007.

TANENBAUM, A. S. **Computer NetWork**. [S.l.]: Prendice-Hall, 2002.

TANG, L. et al. **FreeRank:** implementing independent ranking service for multiplayer online games. 4th ACM SIGCOMM workshop on Network and system support for games. Hawthorne, NY: [s.n.]. 2005. p. 1-7.

THEOTOKIS, S. A.; SPINELLIS, D. A survey of peer-to-peer content distribution technologies, 39, 2004.

TOWSLEY, D. Peer-peer Networking. **Centro de Informática da UFPE**, 2003. Disponivel em: <http://www.cin.ufpe.br/~gprt/gtp2p/tutoriais/p2p03-tutorial_towsley.pdf>. Acesso em: 24 nov. 2007.

UNIVERSITY OF CALIFORNIA AT DAVIS. Simulation with SimPy - In Depth Manual. **SimPy Simulation Package Homepage**, 2002. Disponivel em: <http://simpy.sourceforge.net/SimPyDocs/Manuals/Manual.html>. Acesso em: 10 set. 2009.

VALDURIEZ, P.; PACITTI, E. Data Management in Large-Scale P2P Systems, Valencia, 2005. 104-110.

VASCONCELOS, M. A. S. **Economia:** Micro e Macro. 4o. ed. São Paulo: Atlas, 2006.

WANG, T.; LU, X.; DUAN, H. A Novel Behavior-Based Peer-to-Peer Trust Model. In: _____ **Grid and Cooperative Computing - GCC 2005**. [S.l.]: Springer Berlin / Heidelberg, v. 3795/2005, 2005.

WIKIPEDIA.

WIKIPÉDIA. Relação (matemática). **Wikipédia**, 15 jan. 2001. Disponivel em: <http://pt.wikipedia.org/wiki/Rela%C3%A7%C3%A3o_%28matem%C3%A1tica%29>. Acesso em: 10 dez. 2009.

WOLSKI, R.; PLANK, J. S.; BREVIK, J. **g-Commerce - Building Computational Marketplaces for the Computational Grid**. University of Tennessee. [S.l.]. 2000. (UT-CS-00-439).

WOLSKI, R.; PLANK, J. S.; BREVIK, J. Analyzing Market-Based Resource Allocation Strategies for the Computatio. **International Journal of High Performance Computing Applications**, 15, 2001. 258-281.

XUE, Y.; LI, B.; NAHRSTEDT, K. **Price-based Resource Allocation in wireless ad hoc networks**. Eleventh International Workshop on Quality of Service. Berkeley: Springer Berlin. 2003.

YEO, C. S.; BUYYA, R. A taxonomy of market-based resource management systems for utility-driven cluster computing. **Software—Practice & Experience**, 36, n. 13, Nov 2006. 1381-1419.

YU, B.; SINGH, M. P. Distributed Reputation Management for Electronic Commerce. **Computational Intelligence**, 2002.

APÊNDICE A - CÓDIGO FONTE DAS PRINCIPAIS FUNÇÕES

Neste apêndice, será apresentado o código fonte das principais funções utilizada no nessa tese. A ferramenta de pydssim (*Python Distribute System Simulator*) encontra-se disponível para utilização em (MOURA, 2009). O pydssim foi totalmente desenvolvido na linguagem Python em conjunto com a biblioteca SimPy (UNIVERSITY OF CALIFORNIA AT DAVIS , 2002). Ela fornece primitivas para a criação de tarefas de aplicação, o mapeamento das tarefas aos recursos. Nessa, que foram efetuadas todas as simulações do modelo de compartilhamento de recursos proposto. Seguindo a organização de como foram propostos, as próximas seções detalharão o de cada um dos algoritmos.

A1 – SETSERVICEFORTRANDING

O código abaixo demonstra a função que dá inicio do processo de troca de recursos.

```
setServiceForTrading(self,trading):


    self.getTradings().addElement(trading)
    self.__isa= InformationServiceAgent(self)
    self.__isa.searchServiceForTrading(trading)

    tradingUUID = trading.getUUID()
    timeStart =time.time()

    peer = ""

    while (trading.getStatus() == AbstractTrading.STARTED and ((time.time() - timeStart) < 20)) :

       if trading.getStatus() == AbstractTrading.NOTCOMLETE:
          continue


       if (time.time() - timeStart) > 10 and trading.getAttempt() == 1:
          timeStart =time.time()
          trading.setAttempt(trading.getAttempt() +1)
          self.__isa.searchServiceForTrading(trading)

       peer,trust = trading.definyPeerTrading()

       if trust >= 0.5 or len(trading.getPeersTrading())>3:
          if self.__isa.sendResponseToPeerWinner(trading,self.getPeer().getPID(),peer)== AbstractTrading.ACK:
             trading.setStatus(AbstractTrading.COMPLETE)

             ownershipCertificate = self.__isa.sendOwnershipCertificate(trading,self.getPeer().getURN(),peer)
             trading.setOwnershipCertificate(ownershipCertificate)
          else:
             trading.getPeersTrading().pop(peer)


    self.__isa.sendResponseToPeerAll(trading,self.getPeer().getPID(),peer)
    TradingLogger().resgiterLoggingInfo("****** Final Trading ,Peer = %s"%(self.__peer.getPID()))
```

A2 - GETALLEQULIVALENCEPERIOD

A função getAllEquivalencePeriod retorna todos os serviços equivalentes dento de um especificado.

```
def getAllEquivalenceInPeriod(self,periodStart,periodEnd):
      return dict([(equivalenceID,equivalence) for equivalenceID, equivalence in self.getEquivalences().iteritems()
                  if(equivalence.getPeriodStart() <=periodStart) and (equivalence.getPeriodEnd()>= periodEnd)])


   def hasEquivalencesForTag(self,serviceTag,periodStart,periodEnd):

      return dict([(equivalenceID,equivalence) for equivalenceID, equivalence in self.getEquivalences().iteritems()
                  if (equivalence.getEquivalence().getResourceTag() == serviceTag) and
                    (equivalence.getPeriodStart() <=periodStart) and (equivalence.getPeriodEnd()>= periodEnd)])

   def hasEquivalences(self,recourseEquivalence,periodStart,periodEnd):

      return dict([(equivalenceID,equivalence) for equivalenceID, equivalence in self.getEquivalences().iteritems()
                  if (equivalence.getEquivalence().getResourceUUID() == recourseEquivalence) and
                    (equivalence.getPeriodStart() <=periodStart) and (equivalence.getPeriodEnd()>= periodEnd)])
```

A3 – TRUSTFINALVALUECALCULATION

A função TrustFinalValueCalculation calcula o confiança final de um determinado serviço.

```
def TrustFinalValueCalculation(self,peerID,serviceID,startDate,stopDate):

      trustFinalValue = 0.5
      confDir, t ,tt = self.directTrustCalculation(peerID, serviceID, startDate, stopDate)

      gamaConfDir    = (self.getGama()*confDir)
      betaReputation = (self.getBeta()*self.reputationCalculation(peerID, serviceID, startDate, stopDate))
      trustFinalValue = (gamaConfDir+betaReputation)
      hoje = datetime.today()
      period =  hoje.strftime('%d/%m/%Y %H:%M')


      self.getTrustFinal().addElement(TrustFinal(peerID,serviceID,serviceID,AbstractTrust.TRUSTF,trustFinalValue,perio
d,True))

      return trustFinalValue
```