UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
DEPARTAMENTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GERALDO PEREIRA ROCHA FILHO

IMPLEMENTAÇÃO E AVALIAÇÃO DO PROTOCOLO TAPESTRY USANDO O NETWORK SIMULATOR 3

VITÓRIA DA CONQUITA, BA
2012

GERALDO PEREIRA ROCHA FILHO

IMPLEMENTAÇÃO E AVALIAÇÃO DO PROTOCOLO TAPESTRY USANDO O NETWORK SIMULATOR 3

Trabalho de Conclusão de Curso apresentada à Universidade Estadual do Sudoeste da Bahia, como requisito para obtenção do título de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marlos André Marques Simões de Oliveira

VITÓRIA DA CONQUITA, BA
2012

GERALDO PEREIRA ROCHA FILHO

IMPLEMENTAÇÃO E AVALIAÇÃO DO PROTOCOLO TAPESTRY USANDO O NETWORK SIMULATOR 3

Monografia aprovada como requisito parcial para obtenção do grau de bacharel, no curso de graduação em Ciência da Computação na Universidade Estadual do Sudoeste da Bahia.

Banca Examinadora:

_____

Prof. Ms. Marcelo Barbosa de Almeida

_____

Prof. Dr. Marlos André Marques Simões de Oliveira

_____

Prof. Dr. Roque Mendes Prado Trindade

Dedico este trabalho a minha irmã,
Thiaquelliny Teixeira Pereira.

AGRADECIMENTO

Agradeço:

Aos meus pais, Geraldo Pereira Rocha e Adenita Teixeira Pereira, pelo apoio que sempre ofereceram para mais uma conquista, mostrando desde cedo que com perseverança e dedicação os objetivos são alcançados.

Aos meus familiares pela força para seguir em frente.

À minha amiga e irmã, Thiaquelliny Teixeira Pereira, pelos ensinamentos no decorrer do curso, principalmente nesta etapa final.

À minha namorada, Cristiane de Azevedo Guimarães, pela paciência, compreensão e por estar ao meu lado em mais uma vitória.

Ao meu Orientador, Professor e Auxiliador, Marlos Marques, que sempre teve paciência e atenção nos momentos em que necessitei de ajuda; além da confiança depositada em mim.

A todos os professores, do curso de Ciência da Computação pela contribuição do conhecimento transmitido nestes últimos anos de estudos, e a todos que colaboraram de forma direta ou indireta neste trabalho.

Mas principalmente a Deus por mim dar sabedoria e saúde para concluir este trabalho.

“A mente que se abre a uma nova idéia jamais volta ao seu tamanho original”. (Einstein)

RESUMO

A estrutura *Peer-to-peer* vem ganhando notoriedade nos últimos anos em consequência do sucesso de suas aplicações, que exigem acesso a recursos compartilhados por usuário. Além disso, o rápido crescimento da internet colaborou para o surgimento do paradigma P2P. A sua grande vantagem em relação ao modelo Cliente/Servidor, é a colaboração direta entre os usuários sem a necessidade de depender de um servidor gerenciado por terceiros. Logo, estratégias para encontrar recursos localizados em maquinas de toda uma rede devem ser avaliadas, por isso protocolos para a distribuição e localização de objetos é fator determinante para demonstrar a escalabilidade e robustez em um sistema. Portanto este trabalho implementa o protocolo Tapestry; sendo avaliado em um ambiente virtual realístico utilizando as métricas *Stretch e Stress*, por meio de uma simulação, usando a ferramenta Network Simulator 3.

PALAVRAS-CHAVES: Tapestry; Network Simulator 3; Rede *Peer-to-peer*; Protocolo.

ABSTRACT

The *Peer-to-peer* structure has gained notoriety in last years due to the success of their applications, which require access to shared resources per user. Moreover, the rapid growth of the Internet contributed to the emergence of the paradigm P2P. Its great advantage over the Client/Server model, is the collaboration direct between users without having to rely on a server managed by third parties. Thus, strategies for finding resources located on machines across a network should be evaluated, so protocols for the distribution and location of objects is an important factor to demonstrate the scalability and robustness in a system. Therefore this work implements the protocol Tapestry; being evaluated in a realistic virtual environment using metrics *Stretch* and *Stress*, through a simulation using the tool Network Simulator 3.

KEYWORDS: Tapestry; Network Simulator 3; Network *Peer-to-peer*; Protocol.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALM	(*Application Layer Multicast*) - Comunicação multiponto na camada de aplicação.
AVC	Ambiente Virtual Colaborativo.
CPU	(*Central Processor Unit*) – Unidade de processamento central.
CAN	(*Content-Addressable Network*) – Rede de conteúdo endereçável.
DHT	(*Distributed Hash Table*) – Tabela hash distribuida.
GNS-3	(*Graphical Network Simulator 3*) – Simulador de rede grafico 3.
GUID	(*Globally Unique Identifier*) – Identificador global exclusivo.
ICQ	(*I Seek You*) – Eu procuro você, programa de comunicação instantâneo.
ID	(*Identify*) – Identificador.
IP	(*Internet Protocol*) – Protocolo Internet.
IPv4	(*Internet Protocol version* 4) – Protocolo Internet versão 4.
IPv6	(*Internet Protocol version* 6) – Protocolo Internet versão 6.
MPLS	(*Multiprotocol Label Switching*) – Chaveamento de rótulo multi protocolo.
NAM	(*Network Animator*) – Animador de Rede.
NS-2	(*Network Simulator* 2) – Simulador de rede 2.
NS-3	(*Network Simulator* 3) – Simulador de rede 3.
OPNET	(*Optimizing Network Engineering Tools*) – Ferramenta de Engenharia para Otimização de Rede.
OS	(*Operation System*) – Sistema Operacional.
OTCL	(*Olde Towne Civic League*) – Linguagem utilizada para simulações em redes.
P2P	(*Peer-to-peer*) – Par-a-par, qualquer forma de comunicação não-hierárquica.
POP	Ponto de Presença.
QoS	(*Quality of Service*) – Qualidade de serviço.
RDP	(*Relative Delay Penalty*) – Penalidade de atraso relativo.

RDPM Penalidade de atraso relativo médio.

RNP Rede Nacional de Pesquisa.

SHA (*Secre Hash algorithm*) – Função *hash*.

TCC Trabalho de Conclusão de Curso.

TCP (*Transmission Control Protocol*) – Protocolo de Transmissão de Controle.

TCP/IP (*Transmission Control Protocol/Internet Protocol*) – Protocolo de transmissão e de controle/Protocolo Internet.

UDP (*User Datagram Protocol*) – Protocolo de datagrama do usuário.

SUMÁRIO

CAPÍTULO 1

Introdução

O uso da tecnologia *Peer-to-peer* (P2P) vem ganhando notoriedade nos últimos anos, principalmente pelo sucesso de certas aplicações como Napster, KaZaa, ICQ e BitTorrent. Para Rocha et. al. (2004) o resultado do crescimento dessas aplicações se deve ao fato delas estarem relacionadas à tecnologia P2P, estimulando os usuários no momento em que eles percebem que podem compartilhar dados e contribuir para a disseminação das informações.

O sistema P2P tem como finalidade permitir a distribuição de recursos a qualquer *host* que tenha acesso a uma rede de comunicação. Os *hosts* participantes podem atuar tanto como cliente quanto servidor de recursos, segundo Junior et. al. (2005), essa característica dispõe aos integrantes, denominado de *peers* (pares) ou nós, as mesmas capacidades e responsabilidades.

O paradigma cliente/servidor administra os recursos situados em uma única máquina, considerada servidor, essa particularidade pode tornar a capacidade do serviço limitada por causa dos componentes de *hardware* do servidor, além de não exigir muito da distribuição e gerenciamento dos recursos. Em contrapartida tais desvantagens podem ser superadas pelo conceito de rede *Peer-to-peer*. Para Couloris, G., Dollimore, J., Kinberg (2007), este paradigma fornece acesso aos recursos de informações localizados em máquinas que pertencem a uma rede; por isso, os algoritmos para a distribuição e recuperação de informação são aspectos importantes a ser considerado.

Este trabalho visa a implementação e avaliação do protocolo Tapestry, que possui como estratégia um roteamento baseado em prefixo, usando a ferramenta NS-3. O roteamento é feito por meio de uma DHT e encaminha as mensagens para os nós Tapestry com base nos GUIDs que estão associados aos objetos, mostrando que uma camada *routing overlay* obtêm bom desempenho e alta confiança em uma rede envolvendo muitos nós.

1.1. Motivação

A tecnologia P2P surgiu para mudar o paradigma existente de Cliente/Servidor, pois essa independe de uma organização central ou hierárquica. Dispondo aos seus usuários as mesmas capacidades e responsabilidades, uma vez que os integrantes não possuem maquinas dedicadas. "Através dessa tecnologia qualquer dispositivo pode acessar diretamente os recursos de outro, sem nenhum controle centralizado." (JUNIOR ET. AL., 2005)

De acordo com os pesquisadores Couloris, G., Dollimore, J., Kinberg (2007) os projetos de sistemas de sobreposição de roteamento começou em 2000 e continua sendo uma área de pesquisa bastante ativa. Isso surgiu por causa do crescimento muito rápido da Internet, no qual os aplicativos distribuídos compartilham recursos resultantes da colaboração de muitos *hosts* na rede. Vários protocolos na camada de aplicação sugiram neste ínterim, propondo: gerenciamento, confiabilidade, qualidade de serviço (QoS), escalabilidade e controle de congestionamento. Estes aspectos em nível de aplicação possuem seus benefícios como: maior escalabilidade; mais flexibilidade e adaptabilidade, pois não necessitam de uma infra estrutura como na camada de rede, tornando as aplicações distribuídas e disponíveis na internet.

Para o pesquisador Oliveira (2010) a natureza altamente dinâmica das aplicações P2P, se dá por causa dos aspectos comportamentais dos usuários que utilizam esse tipo de serviço, tornando a rede muito volátil. Isto requer estratégias para a construção de redes virtuais confiáveis, escaláveis e em tempo real. Logo o conhecimento de como os *hosts* estão interligados na rede física real, pode beneficiar na construção de uma rede sobreposta virtual, tornando mais adaptável, mais auto organizável e mais eficiente na sua escalabilidade.

Por ser um assunto recente, com relevância tanto no meio acadêmico quanto para ambientes corporativos, este trabalho se justifica pela possibilidade de oferecer um aprofundamento sobre o tema.

1.2. Objetivos

1.2.1. Geral

Implementar e avaliar um protocolo distribuído, Tapestry, conhecido como *routing overlay*, utilizando a ferramenta de simulação Network Simulator 3.

1.2.2. Específicos

- Estudar conceitos chaves do paradigma *Peer-to-peer*;
- Estudar rede de sobreposição que oferece um serviço de roteamento no nível da aplicação;
- Estudar estruturas *Peer-to-peer* que podem ser classificadas quanto a sua organização;
- Estudar sistemas de compartilhamento de dados em redes *Peer-to-peer*;
- Estudar as características do Network Simulator 3, para usá-lo como ferramenta base para a implementação do protocolo Tapestry;
- Construir um modelo de simulação para o protocolo Tapestry utilizando a ferramenta Network Simulator 3;
- Analisar os resultados obtidos da simulação através de métricas bem definidas.

1.3. Estrutura do trabalho

Esta monografia está organizada em 5 capítulos. O capítulo 1 trata-se da introdução do trabalho. No capítulo 2 se retrata ao referencial teórico, em que é descrito, de forma panorâmica, os conceitos chaves P2P, com o objetivo de proporcionar uma boa compreensão da rede *overlay* no nível de aplicação e suas organizações. Ainda, na referida seção, também é feita uma analise dos principais simuladores existentes e do escolhido para ser utilizado. No capítulo 3 são analisados e estudados os protocolos que implementam um roteamento em nível de aplicação, junto com trabalhos atuais relacionados ao tema da monografia. No capítulo 4 é apresentado a contribuição deste TCC, bem como a avaliação do protocolo Tapestry. O capítulo 5 apresenta a conclusão e algumas possibilidades de trabalhos futuros.

CAPÍTULO 2
Referencial teórico

Neste espaço, dedica-se ao estudo bibliográfico do tema: Implementação e avaliação do protocolo Tapestry utilizando o network simulator 3.

2.1. Conceitos chaves *peer-to-peer*

Segundo o autor Barcellos & Gaspary (2006) não há um consenso na literatura sobre os aspectos fundamentais dos sistemas P2P. Para eles, originalmente, o P2P se refere a um estilo de arquitetura distribuída que contrapõem com a arquitetura cliente/servidor; sendo os sistemas distribuídos descentralizados, no qual os nodos correspondem igualmente em termos de funcionalidades e tarefas que realizam. Esta definição é purista e exclui diversas aplicações aceitas hoje em dia como *Peer-to-peer*. Mais recentemente, P2P passou a ser associado a uma classe de aplicações que aproveita recursos como disco e CPU presentes nas bordas da Internet.

Theotokis & Spinellis (2004) propõem um conceito mais abrangente sobre sistema P2P: sistemas distribuídos composta por nodos interconectados capazes de se auto-organizar em topologias *overlay*, com a finalidade de compartilhar recursos tais como ciclos de CPU, armazenamento, largura de banda e dados; sendo capaz de se adaptar a falhas, mantendo a conectividade aceitável e um bom desempenho, sem exigir uma intermediação ou apoio de uma entidade centralizada.

A característica fundamental P2P é de que um sistema é construído com base na cooperação dos nós de forma “voluntaria”, ou seja, estabelecendo ligações diretas entre pares de *hosts* e interagindo através dos mesmos, formando uma rede *overlay* no nível de aplicação. Logo, o *routing overlay* compartilha muita semelhança com o roteamento na camada de rede.

2.2. Routing overlay *versus* roteamento IP

O *routing overlay* e o roteamento IP possuem os mesmos princípios. Ambos têm como base achar um caminho "ótimo" entre dois nós que se comuniquem. Diante dessa característica pergunta-se: por quê é necessário usar um roteamento no nível da aplicação, visto que tanto o *routing overlay* quanto o roteamento IP compartilham do mesmo objetivo?

Para Couloris, G.; Dollimore, J.; Kinberg T. (2007), existem várias razões para a existência de um roteamento em nível de aplicação. Os argumentos destes autores são resumidos na tabela abaixo.

Tabela 2.1 - Distinções entre IP e Overlay em nível de aplicação.

Propriedades	Roteamento IP	Routing Overlay
Escala	O IPv4 é limitado a 232 nós endereçáveis. O IPv6 é mais generoso 2128, mas nas duas versões os endereços são estruturados hierarquicamente e grande parte do espaço é previamente alocado de acordo com os requisitos administrativos	Os P2P podem endereçar mais objetos. O espaço de nomes do GUID é muito grande e plano (>2128), podendo ser ocupado de forma muito mais completa.
Equilíbrio da carga	As cargas sobre os roteadores são determinadas pela topologia da rede e pelos padrões de tráfego vigentes.	Os objetos podem se posicionar aleatoriamente e com isso os padrões de tráfego não têm nada a ver com a topologia da rede.
Dinâmica da rede	Tabelas de roteamento de IP são atualizadas de forma assíncrona.	Tabelas de roteamento podem ser atualizadas de forma síncrona ou assíncrona.
Tolerância falhas	A redundância é projetada na rede IP por seus gerentes, além de a replicação ser dispendiosa.	Rotas e referências de objetos podem ser replicadas por um fator n.
Identificação do destino	Cada endereço IP é mapeado em exatamente um nó de destino.	As mensagens podem ser direcionadas para a réplica mais próxima de um objeto de destino
Segurança e anonimato	Obtido quando todos os nós são confiáveis	Obtidos mesmo em ambientes de confiança limitada.

2.3. Modelo de arquitetura peer to peer

Na literatura sobre modelos de arquitetura P2P existem diversas classes, para Rocha et. al. (2004), a mais utilizada principalmente no mundo acadêmico nos últimos anos, divide-se em duas categorias quanto a sua organização: Arquitetura estruturada e não-estruturada.

2.3.1. Arquitetura não-estruturada

O modo no qual os vários elementos em uma rede são interconectados, na arquitetura Não-Estruturada, é determinado de modo *ad hoc*. Esta estrutura é uma "rede que não possui servidor centralizado, nem controle preciso sobre a topologia e localização/busca de documentos." (ROCHA ET. AL, 2004, p. 5). A comunicação entre os nodos é dada de forma arbitrária a medida que os *hosts* entram e saem da estrutura *overlay*. De acordo com Barcellos & Gaspary (2006), por este motivo, a localização de recursos, deve ser completamente independente da topologia da rede *overlay*. Consequentemente uma dificuldade agregada com esta estrutura é o processo de localização de recursos na rede.

Técnicas como inundação do overlay foram utilizadas nos primeiros sistemas P2P. "A ineficiência deste método fomentou a pesquisa e desenvolvimento com ênfase na escabilidade" (BARCELLOS & GASPARY, 2006, p. 6). Como consequência apareceram sistemas não-estruturados que usam técnicas mais eficazes em termo de uso de serviços, como exemplo o Napster, BitTorrent, FreeNet.

O Napster foi o sistema precursor responsável por propagar a tecnologia P2P, o seu sucesso se deu por admitir compartilhamento principalmente arquivos de músicas. Estes eram baixados diretamente de um *host*, podendo ser tanto cliente quanto servidor, de maneira descentralizada. "A arquitetura do Napster incluía índices centralizados, mas eram os usuários que forneciam os arquivos, os quais eram armazenados e acessados em seus computadores pessoais." (COULORIS, G.; DOLLIMORE J.; KINBERG T.; 2007, p.353)

A estrutura do Napster pode ser representada pela Figura 2.1.

Figura 2.1 - Napster compartilhamento de arquivos *Peer-to-peer* com um índice replicado centralizado. (COULORIS, G.; DOLLIMORE J.; KINBERG T., 2007, p.354).

O BitTorrent é um protocolo próprio da camada de aplicação, que permite ao usuário executar *downloads*, de maneira geral indexados em *websites*, utilizando o conceito P2P para conectar os pares de *hosts*. A finalidade é compartilhar os arquivos que já foram baixados, aumentando o desempenho e proporcionando altas taxas de transferência, mesmo que o arquivo esteja sendo baixados por vários usuários. Isso acontece, pois cada usuário coopera com um pedaço do arquivo que esta baixando, como não há uma ordem no recebimento de dados todos os *hosts* contribuem com a rede, mesmo que o arquivo não esteja completo. O arquivo terá os pedaços organizados, quando o *download* for concluído.

O FreeNet é um projeto P2P que permite compartilhar arquivos de forma anônima, a rede de comunicação torna as trocas de mensagens um processo completamente anônimo por meio da criptografia. De acordo Barcellos & Gaspary (2006), o FreeNet é baseado fracamente em DHT usando palavras-chaves e textos para reconhecer o objeto procurado. O roteamento assegura achar o objeto fazendo busca de nodo em nodo através de identificadores ou cadeias de caracteres contendo texto descritivo. Barcellos & Gaspary (2006) o FreeNet é um sistema criado com a finalidade de oferecer: privacidade para nodos que publiquem, recuperem e armazenem objetos; resistência à censura; altas disponibilidade e confiabilidade; armazenamento e roteamento eficientes, escaláveis e adaptativos.

2.3.2. Arquitetura estruturada

A arquitetura Estruturada, de maneira geral usa busca fundamentada em DHT. Para Rocha et. al. (2004) essa estrutura não contém um servidor centralizado de informação, porém contém uma organização expressiva entre os nós. Barcellos & Gaspary (2006) partinha do mesmo conceito, mas acrescenta que a topologia *overlay* é dita por um esquema de alocação de chaves e IDs de nodos; associando um determinado recurso, objeto ou serviço, a um nodo específico de maneira determinística e conhecida integralmente no *overlay*. O DHT pode exercer a função de uma tabela de roteamento distribuída no nível de aplicação, com a finalidade de encontrar um objeto em um numero de passos limitados. Contudo Barcellos & Gaspary (2006) argumenta que ao utilizar DHTs é essencial uma correspondência entre a chave buscada e a chave oferecida como parâmetro na busca, ou seja, o nodo requerente precisa saber perfeitamente a chave do objeto procurado, e nem sempre isso é possível, pois a manutenção do *overlay* em *hosts* altamente instável é difícil.

De acordo com Theotokis & Spinellis (2004), essa categoria de estrutura pode ser separada da infra estrutura, com o objetivo de fornecer melhores serviços para a aplicação. Por isso diferentes protocolos como Tapestry, Chord e CAN têm sido desenvolvidos. A respeito dos referidos protocolos, ressalta-se que eles são mais detalhados no capítulo 3, em que é também realizado, juntamente, um estudo dos trabalhos relacionados.

2.4. Simuladores existentes

Existem vários programas de simulador de redes com o intuito de avaliar o sistema antes de sua implantação. Entre os simuladores que mais se destacam são: Graphical Network Simulator (GNS-3), Network Simulator 3[1] (NS-3), Network Simulator 2 (NS-2), Optimizing Network Engineering Tools (OPNET), Parsec, SSF e QNet. Esses setes simuladores possuem propriedades semelhantes, pois todos empregam simulação baseada em eventos discretos e utilizam uma variedade de protocolos TCP/IP, porém cada um possui foco diferente.

OPNET é um simulador amplamente utilizado no âmbito corporativo, por causa das suas funcionalidades e alta precisão nos resultados. Devido ao seu alto custo, ele se torna restrito em alguns ambientes, como a exemplo pesquisas acadêmicas. GNS-3 é um simulador gráfico de rede que permite a simulação de redes complexas. Elas só são possíveis por causa

[1] O NS-3 será tratado nos tópicos seguintes. Uma boa fonte é o sítio: http://www.nsnam.org/.

dos Dynamips, Dynagen e Qemu, componentes necessário para o desenvolvimento de modelos de simuladores [GNS3, 2011]. O NS-2, NS-3, Parsec, SSF e QNet estão mais focados no projeto de protocolos e mecanismos de rede. O Parsec é uma linguagem de simulação baseada em C, focada para simulações de redes sem fio e permite simulações paralelas [KAMIENSKI, 2002]. O SSF permite simulações paralelas e é mais utilizado para simulações de protocolos de roteamento. O QNet é um simulador de rede IP que foi desenvolvido com a finalidade ajudar trabalhos de pesquisas, projeto e gerenciamento de redes convergentes. A contribuição do QNet está em integrar numa única plataforma de simulação, recursos para tratar de redes convergentes, tecnologia de QoS e engenharia de tráfego para protocolos IP [FIDALGO, 2004].

Ainda, no que ser refere ao estudo de simuladores, analisamos, no próximo tópico, o simulador NS-3 pelo fato de ele ter uma distribuição gratuita de código aberto. Por esse motivo é comum a utilização desse simulador por pesquisadores e para uso educacional, tornando justificáveis as situações onde é necessário desenvolver novas funcionalidades.

2.4.1. NS-3

O projeto do simulador NS-3 teve início no ano de 2006 com o objetivo de ser uma nova alternativa ao popular NS-2. De acordo com Franco et. al. (2009), isso ocorreu porque o NS-2 já não apresentava, há algum tempo, evoluções técnicas, como, por exemplo, uma simulação em paralelo. O NS-3 é um simulador baseado em eventos discretos e orientado a objetos, que tem como funcionalidade proporcionar um ambiente para o desenvolvimento de pesquisa em torno de protocolos que utilizam a pilha TCP/IP, em contexto de redes fixas, móveis, com fio, sem fio, redes locais e de satélite. Abaixo, tratamos o ambiente de desenvolvimento do NS-3, usando como fonte básica a documentação do próprio.

2.4.2. Ambiente de desenvolvimento do NS-3.

No NS-3 os scripts, que contém os comandos e o modelo a ser simulado, são desenvolvidos em C++ ou Python, o que requer que o usuário tenha conhecimento básico em pelo menos um das linguagens, com o objetivo de prover flexibilidade sem prejudicar o desempenho do modelo de simulação, como também, na programação orientada a objetos. Esta tem o objetivo de facilitar os scripts.

Para Kamienski et. al. (2002) o usuário precisa ter em mente o conhecimento de blocos - comandos, classes dos objetos que implementam os protocolos em uma determinada linguagem - para o desenvolvimento de uma simulação. Porém, quando o usuário deseja incluir característica aos protocolos implementados no NS-3 e implementar novos protocolos, é recomendado, a implementação direta utilizando C++ devido a eficiência da linguagem.

O NS-3 está disponível para vários sistemas operacionais como Linux, OS X e Windows. Ressalva-se que no ultimo caso é necessário a instalação do Cygwin, um ambiente Linux para Windows, que tem como objetivo auxiliar a migração de aplicativos do UNIX/Linux para a plataforma Windows. O Cygwin fornece muitos dos comandos mais populares do sistema Linux; no entanto, pode haver problemas de emulação. Logo, uma alternativa seria a instalação da maquina virtual Linux e depois a instalação VMware Server. Este por sua vez, permite a emulação de um servidor inteiro ao invés de funcionar como um simples sistema operacional.

A interação do NS-3 com o usuário se dá por meio de um terminal baseado em texto, apesar de ele não possuir uma interface amigável, o seu sítio dá suporte para os primeiros passos, tornando possível, ainda que de forma custosa, a interação "Homem – Computador".

2.4.3. NS-3 versus NS-2.

O fato de o NS-2 e o NS-3 serem simuladores semelhantes nos conduz a tratar os pontos de destaque de cada um. A mudança mais notável do NS-2 para o NS-3 é a escolha de uma linguagem de script. No NS-2 não é possível fazer uma simulação com apenas C++, ou seja, deve-se fazer um misto entre C++ e OTcl; em contraste, o NS-3 pode ser escrito tanto em C++ quanto em Phyton. O resultado da simulação feita no NS-2 e no NS-3 pode ser visualizado através da Network Animator Nam – NAM. Uma das principais semelhanças entre esses dois simuladores é que ambos são baseados em orientação a objetos - C++ - e alguns códigos do NS-2 já foram aderidos pelo NS-3.

Como dito, ambos os simuladores são semelhantes, e a pergunta que surge frente a essa situação é: qual dos dois simuladores utilizar? De acordo com a documentação do NS-3, essa resposta irá depender do projeto de simulação a ser desenvolvido. O NS-3 não tem todos os métodos que o NS-2 possui, mas por outro lado, ele contém novas funcionalidades como a manipulação de múltiplas interfaces em nós, o uso do endereço IPv4 ou IPv6 e suporte aos

padrões 802.11. Ressalva-se que os desenvolvedores do NS-3 afirmam que a ferramenta está pronto para uso, independente do projeto a ser simulado.

2.4.4. Vantagens do NS-3.

As principais vantagens que levam a utilização do NS-3 são [KAMIENSKI ET. AL., 2002] e [SIMULATOR, 2011] :

- Grande quantidade de protocolos e tecnologia existentes como *multicast*, MPLS, redes *ad-hoc*, redes de satélites, redes com e sem fio;
- Disponibilidade de um simulador padrão para a comunidade acadêmica;
- Exploração de estudos de interações de protocolos em um ambiente controlado;
- Boa base para desenvolvimento, tanto para testar novos protocolos, como para criá-los;
- Código aberto e gratuito.

2.4.5. Limitações do NS-3.

Existem situações que torna o NS-3 desfavorável, como falta de documentação apropriada para iniciantes, tornando a curva de aprendizagem difícil. Caso o usuário necessite de uma funcionalidade que não esteja presente no NS-3, é necessário que o próprio usuário a desenvolva. Para Kamienski et. al. (2002) deve-se não apenas ter apenas intimidade com o desenvolvimento em C++, mas também ter um bom conhecimento de como é a estrutura de dados do NS-3 e seu esquema de hierarquias de classes, o que não é nada simples para um usuário iniciante.

Outra limitação do NS-3 está relacionada com a versão para plataforma Windows. Apesar de existir uma versão para a distribuição Windows muitos adotam a versão para a plataforma Linux, pois o simulador apresenta funcionamento mais eficiente, pelo fato de sua plataforma ser nativa do UNIX.

2.4.6. Estrutura do NS-3.

O NS-3 é um simulador de rede de eventos discretos em que o núcleo de simulação e modelos são implementados em C++. Ele é construído como uma biblioteca que pode ser estática ou dinâmica ligada a um programa principal - escrito em C++ - que define a topologia de simulação e inicia o simulador.

Como já dito, um dos tipos de simulação utilizado para simuladores é o baseado em evento discreto. “Em simulação de redes, o método mais utilizado é o de eventos discretos. Esse método consiste em analisar o estado da rede apenas em um determinado tempo, podendo ignorar os outros estados sem que isto interfira no resultado da análise.” (FUGIMOTO, 2007, p.5). Como exemplo, pode-se representar a transmissão e o recebimento de um pacote, ignorando outros fatores, como cálculos de detecção de erros.

O código fonte para NS-3 é organizado principalmente no diretório src e pode ser representado pela Figura 2.2.

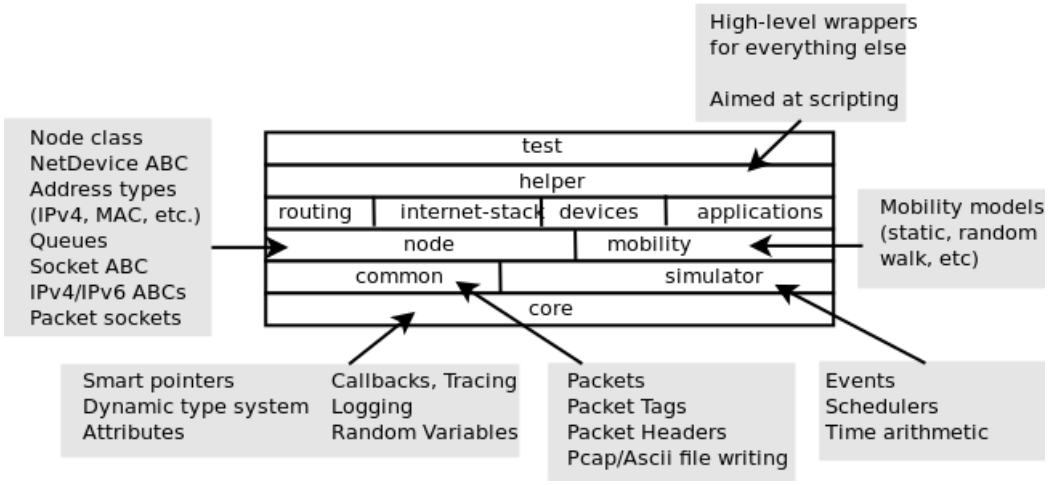

Figura 2.2 - Ilustração da organização do NS-3.

Os pacotes do NS-3 contem uma vasta funcionalidade de protocolos como TCP, UDP, IPv4, IPv6, redes móveis com suporte aos padrões 802.11. Além disso, as bibliotecas proporcionam funções para a criação de alguns tipos de tráfego como o CBR que é utilizado para simular tráfego constante de voz, ON-OFF para tráfego em rajada e voz comprimida, tráfego para transferência de arquivos. Tudo isso, torna o NS-3 um ótimo simulador para trabalhos acadêmicos.

CAPÍTULO 3

O estado da arte

Vários protocolos de roteamento em nível de aplicação que possui arquitetura descentralizada e estruturada P2P como Chord, CAN e Tapestry serão estudados neste capítulo. Esses protocolos serão analisados, bem como as suas estratégias adotadas, com o intuito de permitir o roteamento de mensagens de forma distribuída. Além disso, um estudo de trabalhos atuais relacionado à proposta da monografia será realizado.

3.1. Chord

Para Barcellos & Gaspary (2006) o Chord é um algoritmo de roteamento que usa *hashing* consistente (SHA-1) para associar chaves de objetos a nodos. Os 2^B identificadores, na qual "B" representa a quantidade de bits, estão organizados em um espaço de chaves circular. Os identificadores dos nodos podem ser alcançados fazendo um *hash* do endereço IP, ao passo que as chaves dos objetos são obtidas fazendo um *hash* da descrição do objeto.

A Figura 3.1 mostra o espaço de IDs de nodos circular para B = 5, ou seja, 2^5 identificadores. Os nós 1, 4, 7, 12, 15, 20 e 27 que estão sombreados, são os nós que fazem parte da rede.

Rocha et. al. (2004), os identificadores dos *hosts* são modelados como um círculo de números de 0 a $2^B - 1$, portanto o Sucessor(Chave) é definido como sendo o identificador de nó do primeiro nó real seguinte a Chave, no sentido horário. A exemplo da Figura 3.1, o Sucessor(5) = 7 e Sucessor(9) = 12. O roteamento das mensagens no decorrer do anel é unidirecional sendo recursivo ou iterativo. Além disso, a falha de um *host* não causa uma falha global, pois pode haver replicações de objetos em outros *hosts*.

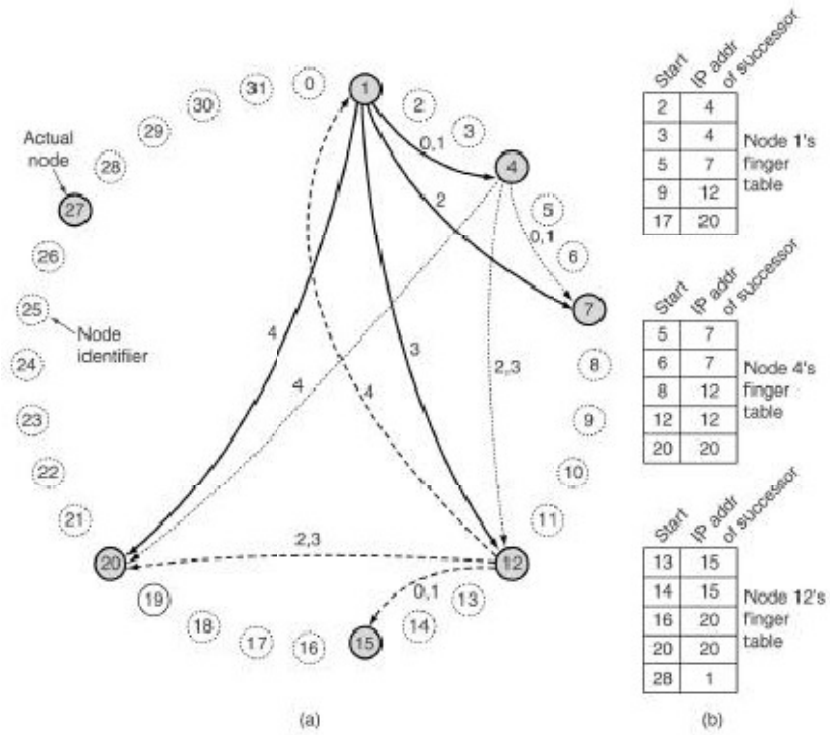

Figura 3.1 - (a) Um conjunto de 32 identificadores de nós organizados em um círculo. (b) Exemplos de tabelas *finger*. (ROCHA ET. AL, 2004, p. 19).

3.2. *Content Addressable Network* (CAN)

O CAN [RATNASAMY et al., 2001] é uma infra-estrutura distribuída projetada para tornar a busca de dados mais eficiente através da utilização da estrutura de tabela *hash* distribuída. Dentre suas principais características estão a escalabilidade, a tolerância a falhas e a auto-organização. Outro aspecto importante é a possibilidade de implementação totalmente no nível de aplicação.

O sistema CAN assemelha-se a uma tabela *hash*, onde são realizadas operações de inserção, pesquisa e exclusão dos pares (chave, valor). Este sistema apresenta-se como um espaço virtual de coordenadas cartesianas d-dimensional, sendo composto por vários nós individuais. Cada nó é responsável por uma porção (zona) deste espaço virtual e detém um pequeno número de informações a respeito das zonas adjacentes (ou vizinhas). O exemplo da Figura 3.2 mostra um espaço virtual bi-dimensional composto por 5 nós.

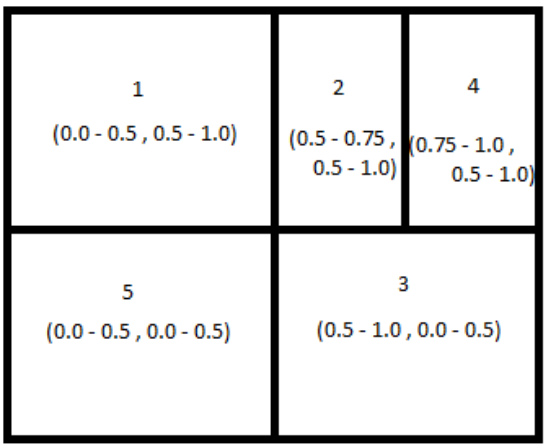

Figura 3.2 - Rede CAN bi-dimensional composto por 5 nós.

3.2.1. Operações básicas em um sistema CAN

3.2.1.1. A inserção de um novo nó

A inserção de um novo nó na rede é feita através do seu mapeamento em algum ponto P aleatório do espaço d-dimensional que estará sob a responsabilidade de um nó já existente na configuração e que por sua vez deverá dividir sua zona em partes iguais cedendo uma delas ao novo nó. Por conseqüência, as informações de vizinhos devem ser atualizadas a fim de manter a consistência do sistema. A Figura 3.3 mostra a inserção do nó 7 em um espaço de coordenadas com vários nós participantes.

3.2.1.2. A busca por um dado

A busca por um dado na rede CAN ocorre pela aplicação da função *hash* ao dado, o que resultará na localização de um ponto X no espaço d-dimensional. A partir deste ponto X é possível a localização da exata zona em que o dado se encontra e finalmente qual o nó responsável por ele. Para um espaço d-dimensional divido em n zonas iguais, as buscas são feitas em (d / 4) * (n 1/d) *hops* e os nós individuais possuem (2 * d) vizinhos.

3.2.1.3. A exclusão de um nó

A qualquer momento um nó CAN pode deixar o sistema. A operação de exclusão afeta a estrutura e desta forma é necessário assegurar que os nós próximos ao nó excluído ocupem sua zona e herdem suas informações.

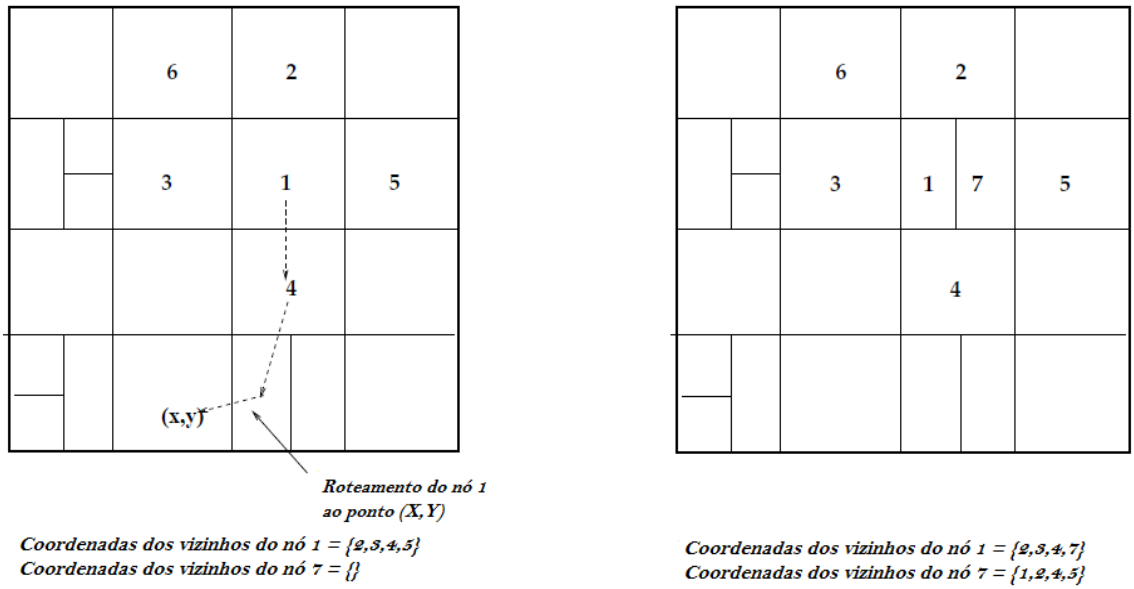

Figura 3.3 - Entrada do nó 7 na rede CAN. [RATNASAMY et al., 2001]

3.2.1.4. Roteamento no CAN

O roteamento em uma rede CAN ocorre por meio da análise das informações contidas nas tabelas de roteamento de cada nó participante da estrutura d-dimensional. Cada nó possui duas informações essenciais em sua tabela que são: o endereço IP e a zona de coordenadas virtuais de cada um de seus vizinhos. Com o uso dessas informações um nó pode enviar uma mensagem a um destino qualquer utilizando-se do mecanismo de encaminhamento de mensagem para o vizinho mais próximo às coordenadas de destino.

Neste tipo de roteamento as mensagens tendem a seguir por um caminho em linha reta, correspondendo ao caminho mais curto entre origem e destino. Fato que não ocorre quando um dos nós que compõem o caminho mais curto falha, obrigando o roteamento da mensagem por um caminho alternativo dentre os vários existentes entre dois pontos no espaço de coordenadas virtuais.

3.3. Tapestry

De acordo com Barcellos; Gaspary (2006), o Tapestry é uma estrutura P2P que permite o roteamento de mensagens de forma distribuída, auto-administrada e tolerante a falhas. Nesse ambiente as informações de roteamento e a localização são partilhadas entre os *hosts* da rede. O protocolo em questão tem como objetivo encaminhar, de maneira correta, uma mensagem endereçada para qualquer nó em O(logN) etapas em uma rede com N *hosts* ativos. Vale ressaltar que o protocolo implementa uma DHT e encaminha as mensagens para os nós com base nas suas chaves, calculadas por meio de uma ***S**ecure **H**ash **A**lgorithm* -SHA- ou simplesmente função *hash*, usando roteamento baseado em prefixo.

Segundo Rocha et. al. (2004), para cada nó da rede é atribuído um identificador, NodeId, gerado através de uma função *hash* aplicado ao seu endereço IP ou a sua chave pública. O identificador, NodeId, é usado para localizar a posição do nó em um espaço de chave circular, organizado entre 0 à $2^{128} - 1$. Uma chave é mapeada para um nó cujo NodeId é, numericamente, o mais próximo da identificação da chave. Para Couloris, G.; Dollimore, J.; Kinberg T. (2007), embora existam no máximo 128 linhas na tabela, apenas $\log_{16}n$ linhas serão preenchidas, em média, em uma rede com N nós ativos.

O protocolo Tapestry faz o roteamento das mensagens para o nó, NodeId, que, como dito, esteja numericamente mais próximo da chave desejada. Para Rocha et. al. (2004), isto é feito a cada etapa do roteamento, quando um *host* envia as mensagens para outro *host* no qual o seu identificador deve compartilhar com a chave, no mínimo um digito a mais do que é compartilhado com o *host* atual. Caso nenhum *host* seja conhecido, a mensagem é enviada para o *host* cuja sua identificação seja numericamente mais próxima do *host* atual. Isso só é possível, porque cada *host* mantém uma tabela de roteamento com o conjunto de seus vizinhos.

Com o intuito de facilitar o entendimento do protocolo Tapestry, a Tabela 3.1 mostra uma DHT para um nó especifico da rede. Tal tabela é organizada da seguinte forma: as chaves são vistas como valores hexadecimais e a tabela ordena as chaves com base em seus prefixos hexadecimais.

Tabela 3.1 - DHT do protocolo Tapestry em um *host* (45678) especifico.

P	Prefixos de Chaves e manipuladores de nós n correspondentes															
0	0 n	1 n	2 n	3 n	**4**	5 n	6 n	7 n	8 n	9 n	A n	B n	C n	D n	E n	F n
1	40 N	41 n	42 n	43 n	44 n	**45**	46 n	47 n	48 n	49 n	4A n	4B n	4C n	4D n	4E n	4F N
2	450 n	451 n	452 n	453 n	454 n	455 n	**456**	457 n	458 n	459 n	45A n	45B n	45C n	45D n	45E n	45F n
3	4560 n	4561 n	4562 N	4563 n	4564 n	4565 n	4566 n	**4567**	4568 n	4569 n	456A n	456B n	456C n	456D n	456E n	456F n
4	45670 n	45671 n	45672 n	45673 n	45674 n	45675 n	45676 n	45677 n	**45678**	45679 n	4567A n	4567B n	4567C n	4567D n	4567E n	4567F n

A Tabela 3.1 esta situada em um nó cuja identificação é 45678. As letras n denotam pares (Chave e Endereço IP) especificando o próximo *hop* a ser dado pelas mensagens endereçadas aos *hosts* que correspondem ao prefixo dado na tabela. Como exemplo, uma mensagem do *host* 45678 para o *host* 65A12 poderia passar pelos seguintes *hosts*: nnnn2 –> nnn12 –> nnA12 –> nn5A12 –> 65A12, ou seja, dado uma mensagem, primeiramente o nó deve verificar se a chave está dentro da faixa de endereço coberto pela DHT. Dessa forma "a tabela de roteamento é usada e a mensagem é encaminhada para o nó que compartilha um prefixo comum com a chave (pelo menos um ou mais dígitos)." (ROCHA ET. AL., 2004, pag. 22). Contudo, em algumas situações pode ocorrer de o nó não ser alcançado, neste caso, a mensagem é enviada para o nó que compartilhe um prefixo com a chave e esteja numericamente mais próxima da chave do que o nó atual.

Com o objetivo de detalhar a execução do protocolo, a Figura 3.4 ilustra as ações do algoritmo de roteamento de uma mensagem do nó 45678 para 65A12, com a ajuda de uma tabela de roteamento adequadamente preenchida. Os pontos representam nós ativos no qual o espaço é considerado circular com o nó 0 adjacente ao nó $2^{128} – 1$.

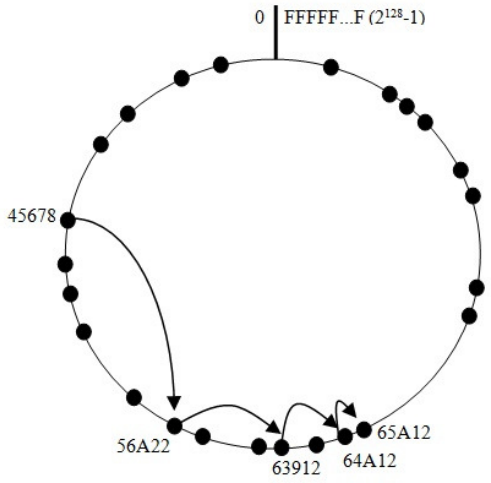

Figura 3.4 – Ilustração das ações do protocolo de roteamento.

3.4. Trabalhos relacionados

O objetivo deste tópico é apresentar trabalhos científicos atuais relacionados ao tema proposto desta monografia. São mostrados estudos cujo foco corresponde à avaliação e análise dos protocolos distribuídos, Tapestry e Pastry. Além disso, são descritas pesquisas relacionadas ao uso do simulador de rede, NS-3.

O Zhao; Kubiatowicz; Joseph (2001) em Tapestry: An *Infrastructure for Fault-tolerant wide-area Location and Routing*, avaliam a importância de um protocolo distribuído, Tapestry, de roteamento na camada de aplicação, usando apenas *links* ponto-a-ponto e sem possuir recursos centralizados. Segundo eles o roteamento e as informações de diretórios dentro desta infra-estrutura é mais fácil de administrar e reparar. Isto ocorre, pois o protocolo é implementado em nível de aplicação, ficando mais próximo do usuário programador, tornando mais auto organizável, com mais facilidade de escalonar, mais adaptável e mais eficiente. Além disso, Zhao; Kubiatowicz; Joseph (2001) mostra a arquitetura do protocolo Tapestry e explora suas vantagens através de uma série de experimentos.

Papadopoulos (2009), *NS3Pastry A PASTRY DHT port to the NS3 Network Simulator*, implementa o protocolo Pastry dentro do simulador de rede NS-3, no qual os aplicativos possam usar estruturas overlay. O NS-3 foi o escolhido por ser uma referência para a simulação na internet, além de ser baseado em eventos discretos o simulador utiliza uma variedade de protocolos, por isso Papadopoulos (2009) propôs uma implementação para o protocolo Pastry. Todavia, ele ressalva que a versão NS3Pastry 0.5 apresenta defeitos com pendências de ponteiros a memória, depois de um certo tempo simulando uma aplicação.

Zhao et. al. (2004), Tapestry: *A Resilient Global-scale Overlay for Service Deployment*, fornece detalhe completo sobre o protocolo de roteamento e do gerenciamento de tabelas de roteamento do Tapestry em face da chegada e saída de nós. Explorando o comportamento da implantação do protocolo no PlanetLab, com um teste global de 100 máquinas . Os resultados apresentam um comportamento estável e um bom desempenho sobre o *routing overlay*, apesar da instabilidade da camada de rede subjacente.

A pesquisa realizada por Zhao et. al. (2004) assemelha-se a proposta deste trabalho. Contudo o ambiente simulado será diferente, usando um simulador de rede

NS-3 para implementar e avaliar o protocolo Tapestry com métricas mais adequada para o escopo da simulação.

CAPÍTULO 4

Implementação e avaliação

A construção de um ambiente virtual bem como sua topologia, aplicação e serviços será modelada para simular e avaliar os resultados do protocolo Tapestry, usando o simulador de rede NS-3. As métricas *Stretch* e *Stress*, foram escolhidas por serem mais adequadas para a avaliação do protocolo em questão. Além disso, serão apresentadas a estratégia utilizada, o cenário desenvolvido e os resultados alcançados. Os objetivos são expor a facilidade de modelagem de um ambiente físico na ferramenta NS-3, mostrar a utilidade para fazer testes de novas funcionalidade como a implementação de novos protocolos de rede, e contribuir tanto para a comunidade de desenvolvimento do Network Simulator 3 *project* quanto para o grupo de pesquisa NS-3 *reviews*.

4.1. Planejamento da Simulação

Cada projeto de simulação possui características próprias, porém existem alguns passos que são comuns, Kamienski et. al. (2002). Neste caso, o primeiro passo foi definir com clareza o sistema a ser modelado, TapestryApplication, situação que, inevitavelmente, exige um conhecimento aprofundado do sistema de simulação a ser desenvolvido. A segunda etapa é definir o cenário do modelo a ser simulado, pois um modelo incorreto pode não apresentar o sistema de forma adequada, por causa das considerações incorretas tomadas sobre o comportamento do ambiente físico. Por ultimo na qual se refere a avaliação de desempenho do sistema, ela só é possível com a utilização de métrica. Está deve ser selecionada de acordo com os objetivos do sistema a ser modelado, para não influenciar nos resultados, por meio das escolhas de métricas inadequadas.

4.1.1. Cenário de simulação

Foi elaborado um cenário virtual realístico, da Figura 4.1, a fim de avaliar o protocolo Tapestry utilizando a ferramenta NS-3. Esse cenário representa o *backbone* da RNP, que interliga os principais centros de pesquisa do país por meio de seus roteadores conhecidos como Ponto de Presença (POPs). Para esses roteadores serão atribuídos os endereços de IPs da Tabela 4.1, na qual cada um será conectado a uma rede local padrão com dez estações de trabalho, tendo a Figura 4.2 como um exemplo. Além disso, foi desenvolvido uma aplicação, TapestryApplication, para esta pesquisa, com o objetivo de gerar trafego de dados encaminhando de maneira eficiente, uma mensagem endereçada para um nó que esteja ativo.

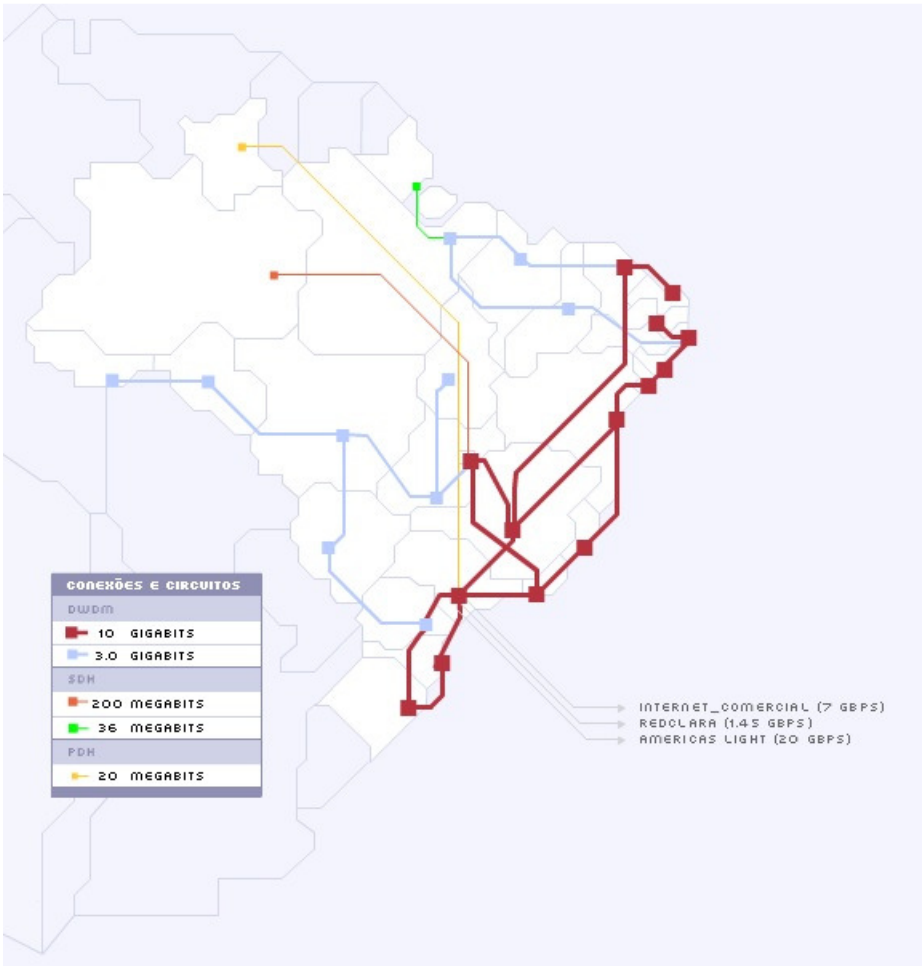

Figura 4.1 – Backbone da RNP de Dezembro de 2011 [RNP 2011]

Tabela 4.1 – Endereços IP dos POPs [OLIVEIRA 2010]

POP	DNS	IP
Acre	www.pop-ac.rnp.br	200.129.175.129
Alagoas	www.pop-al.rnp.br	201.76.50.45
Amapá	www.pop-ap.rnp.br	200.129.167.34
Amazonas	www.pop-am.rnp.br	200.129.156.1
Bahia	www.pop-ba.rnp.br	200.18.234.22
Ceará	www.pop-ce.rnp.br	200.129.0.45
Distrito Federal	www.pop-df.rnp.br	200.19.119.125
Espírito Santo	www.pop-es.rnp.br	200.137.64.23
Goiás	www.pop-go.rnp.br	200.17.58.7
Maranhão	www.pop-ma.rnp.br	200.137.129.22
Mato Grosso	www.pop-mt.rnp.br	200.129.240.1
Mato Grosso do Sul	www.pop-ms.rnp.br	200.129.207.34
Minas Gerais	www.pop-mg.rnp.br	200.131.1.80
Pára	www.pop-pa.rnp.br	200.129.132.130
Paraíba	www.pop-pb.rnp.br	200.129.64.130
Paraná	www.pop-pr.rnp.br	200.134.255.12
Pernambuco	www.pop-pe.rnp.br	200.133.0.136
Piauí	www.pop-pi.rnp.br	200.137.160.188
Rio de Janeiro	www.pop-rj.rnp.br	200.159.252.82
Rio Grande do Norte	www.pop-rn.rnp.br	200.137.0.40
Rio Grande do Sul	www.pop-rs.rnp.br	200.132.0.151
Rondônia	www.pop-ro.rnp.br	200.129.139.136
Roraima	www.pop-rr.rnp.br	200.129.143.4
Santa Catarina	www.pop-sc.rnp.br	200.237.192.3
São Paulo	www.pop-sp.rnp.br	200.133.192.62
Sergipe	www.pop-se.rnp.br	200.17.118.193
Tocantins	www.pop-to.rnp.br	200.129.179.6

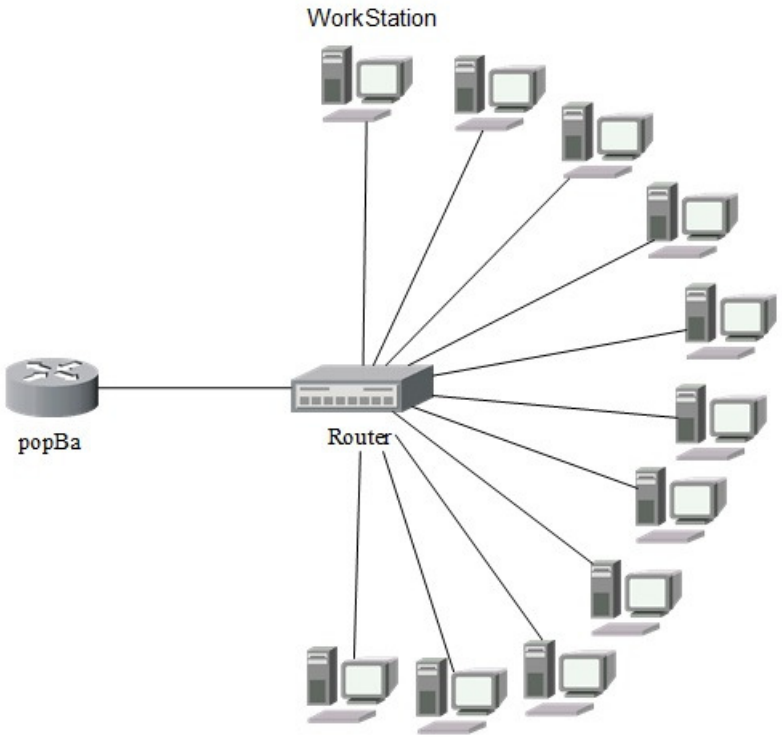

Figura 4.2 – POP da Bahia conectado a uma rede local padrão.

A implementação do ambiente virtual foi realizada em uma máquina com as seguintes configurações:

- Processador Core2 Duo P8700 2,53GHz (velocidade do *clock*).
- Memória cachê 3 MB.
- Memória RAM 3 GB DDR3 1066MHz.
- Sistema Operacional Ubuntu Server 9.10

4.1.2. TapestryApplication

A aplicação TapestryApplication foi desenvolvida incorporando um protocolo distribuído, Tapestry. Tal estratégia foi utilizada porque torna a construção do protocolo mais flexível, ao invés de usar uma aplicação do próprio NS-3, já que esta não foi construída para o protocolo a ser implementado, consequentemente não possui um serviço para uma implementação adequada. Além do mais, foi feito uma interface de programação básica para a construção de uma DHT, conforme o desenvolvimento da aplicação. Isso proporcionará uma flexibilidade adicional para o TapestryApplication, colocando réplicas próximas aos requerentes, reduzindo a latência e minimizando as cargas de rede. Esta aplicação, implementada - na linguagem C++ - no ambiente de

desenvolvimento do NS-3, possui a finalidade de encaminhar mensagem para o nó que esteja numericamente mais próximo ao requerente.

O código fonte a seguir corresponde ao método SendMessage, Figura 4.3, da aplicação TapestryApplication, que usa o protocolo Tapestry encaminhando de maneira correta, uma mensagem endereçada para qualquer nó em uma rede com N *hosts* ativos.

```
void TapestryAplicacao::SendMessage(void){
  Ptr<Packet> pacote = Create<Packet> (m_tamanhoPacote);
  bool flag = true;
  // checa o Ipv4Address, caso tenha, faz o roteamento da mensagem para o nó correto com a ajuda da DHT.
  Ipv4Address prox = check(g_objeto, g_cli);
  if(operator!=(Ipv4Address::GetZero(),prox)){
        g_cli = prox;
        m_soquete->SendTo(pacote,0,prox);
  }
  //sucessor da DHT na qual seja numericamente mais proximo do nó atual.
  else if(operator==(Ipv4Address::GetZero(),prox)){
      string auxiliar = NodeId(g_cli);
      for(int i = 0; i<g_contador; i++){
          //mensagem é enviada para o nó cuja identificação seja numericamente mais proxima do nó atual.
          if(auxiliar==host[i].nodeId){
              auxiliar = host[i+1].nodeId;
              //procurar o objeto no host numericamente mais proximo ao host[i+1]
              prox = check(g_objeto, auxiliar);
              //caso tenha! enviar e sair do for, caso não tenha ira achar o nó mais proximo no do host atual
              if(operator!=(Ipv4Address::GetZero(),prox)){
                  flag = false; // o host pode ser alcancado, não é necessario usar SucNeighbor
                  m_soquete->SendTo(pacote,0,prox);
                  break; // saindo do for
              }
          }
      }
        //pode ocorrer de o nó não ser alcancado.
        if(flag){
          g_cli = SucNeighbor(n0);
          m_soquete->SendTo(pacote,0,g_cli);
      }
  }
  // para manter o agendamento da cadeia de eventos.
  if(++m_pacoteEnviado<m_nPacote){
      //Agenda um outro evento de transmissão até que TapestryAplicacao descide que enviou
      ScheduleTx();
  }
}
```

Figura 4.3 – Método SendMessage.

4.1.3. Métricas

Na tentativa de medir o desempenho do protocolo na aplicação TapestryApplication de forma justa, adotou-se duas métricas, *Stretch* e *Stress*, que foram apresentadas pelos autores Chu, Rao & Zhang (2000); Pendarakis et. al. (2001); Banerjee, Bhattacharjee & Kommareddy (2002) ambas estão descritas abaixo:

- **Penalidade de Atraso Relativo (RDP):** Conhecida como *Stretch*, razão entre o atraso de dois nós na sobreposição e o atraso que esses dois nós sofreriam na rede subjacente.
- ***Stress*:** Conhecido como *Link Stress*, que seria o número de cópias idênticas de um pacote trafegando em um mesmo enlace físico;

Fahmy & Kwon (2007), faz uma distinção entre o RDP e o *Stretch*, sendo o primeiro a razão entre o atraso do caminho na rede virtual com a rede real, e o ultimo como sendo a razão entre o número de saltos nessas duas redes.

A Figura 4.4, apresenta um exemplo do cálculo da métrica *Stretch* e *Stress*, que servirá como base para avaliar o protocolo Tapestry no NS-3, cujo o mesmo é avaliado na sessão 4.2. Na Figura 4.4a obtém-se o *Stretch* médio 1,25 (média do *Stretch* individual dos *links* AB=3/3=1, AC=6/4=1,5), enquanto que o *Stress* máximo é 1. No caso da Figura 4.4b tem-se uma árvore diferente, o *Stretch* médio passa a ser 1 (média do *Stretch* individual dos *links* AB=3/3, AC=4/4), ao passo que o S*tress* máximo muda para 2.

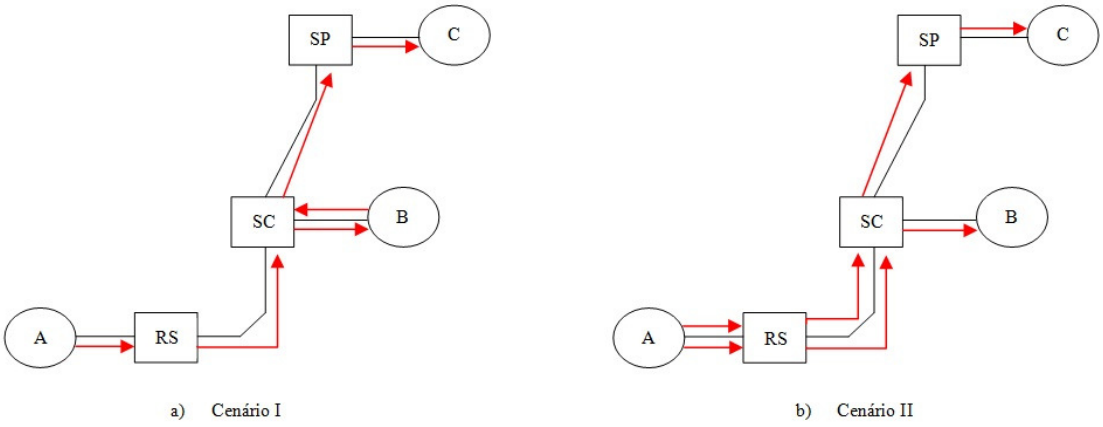

Figura 4.4 – Exemplo de cálculo de *Stretch* e *Stress*.

4.2. Avaliação dos resultados obtidos

Os resultados a serem apresentados nesta sessão, foram obtidos através das análises de capturas de pacotes geradas pela aplicação TapestryApplication, em conjunto com a ferramenta Wireshark. A média de cada métrica foi avaliada através de um conjunto de 70 experimentos, a fim de observar a estabilidade dos dados obtidos.

A Figura 4.5 e 4.6 exibe os resultados das métricas *Stretch* e *Stress* médio. O IP *multicast* nativo que possui uma solução ótima com o valor de *Stretch* 1 e *Stress* 1 [Chu, Rao & Zhang; 2000], foi adicionado nos Gráficos RPDM e Stress Médio, servindo como referencia para avaliar o *Stretch* e *Stress* do protocolo Tapestry.

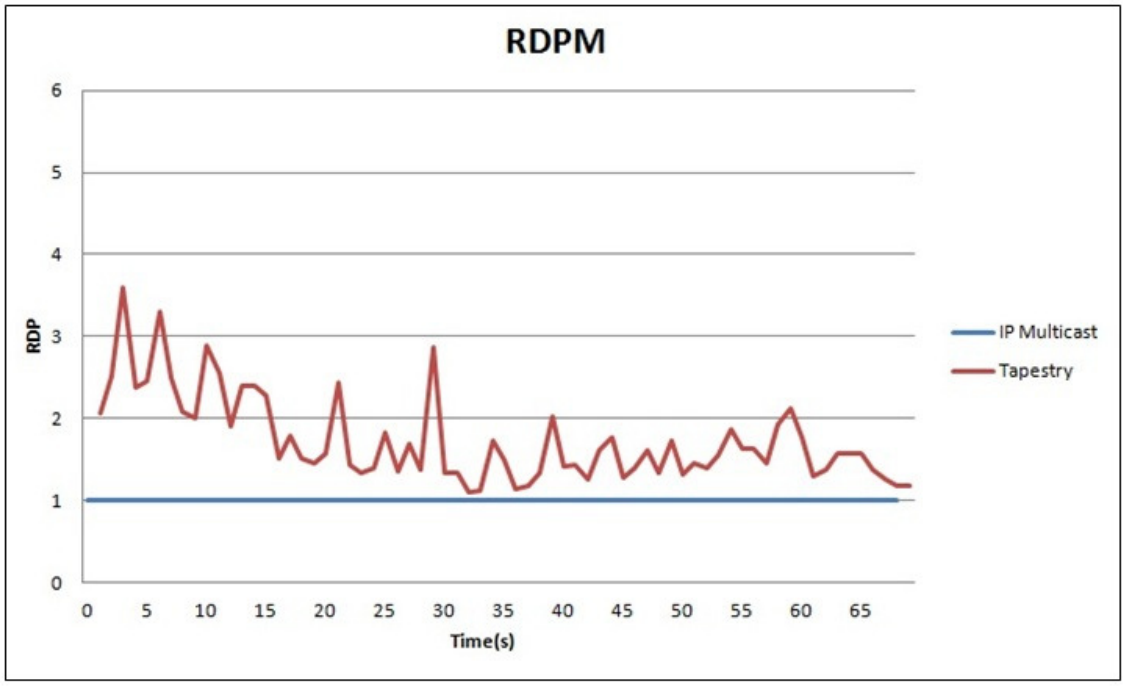

Figura 4.5 – Resultado do *Stretch* médio.

O gráfico acima, representado na Figura 4.5, mostra que quando o sistema se estabiliza obtêm se menos de 2 RDPM para cada par de membro, um dos motivos é o uso de DHT que suaviza a relação entre o caminho que um pacote percorre na rede virtual, com o caminho que ela percorre na rede física. Além disso, o emprego do protocolo TCP para o encaminhamento de mensagens ajuda a amenizar o *Stretch*. No entanto, existem alguns pares de membros com RDP alto. Isto se dá, pois no inicio da simulação do sistema alguns *hosts* ainda não estão ativos, levando a uma arvore de distribuição não ótima, ocorrendo um RDP alto em comparação ao IP *multicast*.

Na Figura 4.6, observa-se que a árvore de distribuição de dados melhora gradualmente durante a entrada de mais *hosts* na rede, ou seja, está convergindo mais rapidamente para um valor estável tendo o Stress médio menor que 2, consequentemente pode-se presumir também que o encaminhamento de dado permaneça mais estável ao longo de um período. Além disso, a curva do Gráfico Stress Médio está ligado à estrutura de dados *hashing* permitindo nodos entrarem e saírem da rede causando pouco estresse ao *overlay* P2P [Barcellos & Gaspary, 2006]. Isso acontece, pois quando um nodo ingressa na rede após o nodo n, ele assume a responsabilidade sobre uma parcela das chaves que estavam com n.

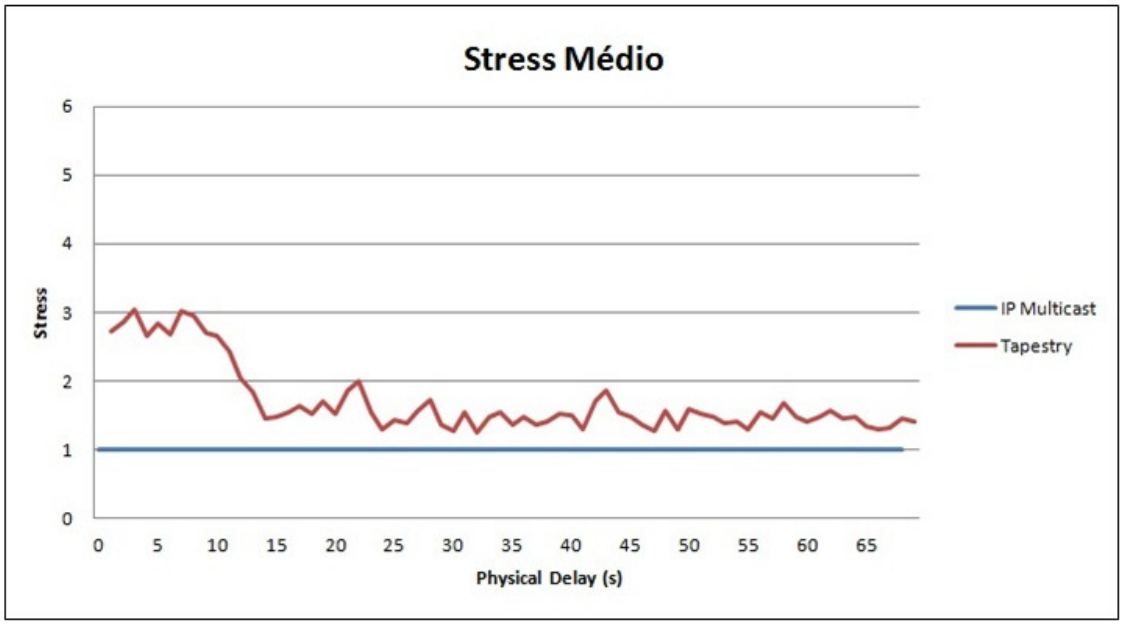

Figura 4.6 – Resultado do *Stress* médio.

Os resultados apresentados acerca da Figura 4.5 e 4.6 são similares aos dos autores, Zhao et. al. (2004); Zhao, Kubiatowicz & Joseph (2001), que faz uma avaliação do protocolo Tapestry apresentado na sessão 3.4. Trabalhos Relacionados. Além disso, os pesquisadores Chu, Rao & Zhang (2000); Banerjee, Bhattacharjee & Kommareddy (2002), avaliam a importância de um protocolo distribuído como o NARADA e NICE chegando a resultados semelhantes.

CAPÍTULO 5

Conclusões e trabalhos futuros

As redes *peer-to-peer* surgiram para mudar o paradigma existente da arquitetura Cliente/Servidor tradicional, pois essa independe de uma organização central ou hierárquica. O modelo Cliente/Servidor pode possuir serviço limitado por causa dos componentes de *hardware* do servidor, além de exigir poucas decisões sobre o gerenciamento e a distribuição dos recursos. Contudo, tais pontos negativos podem ser superados pela tecnologia P2P. Esta fornece acesso aos recursos de informações situados em máquinas que pertencem a uma rede, logo os algoritmos distribuídos são aspectos importantes a ser considerado.

Durante o desenvolvimento deste TCC foi possível perceber a importância da simulação e o uso do NS-3 na pesquisa científica. Visto que o NS-3 é uma ferramenta de construção e análise usada para prever o efeito de mudanças em um mundo real. Por esse motivo é comum o seu uso por pesquisadores, no estudo científico em situações na qual é necessário tanto desenvolver novas funcionalidades quanto simular sistemas existentes.

Este trabalho de conclusão de curso implementou o protocolo Tapestry e avaliou o seu desempenho em um ambiente virtual realístico, *backbone* da RNP com cada POP conectado a uma rede local padrão, usando o simulador de rede NS-3. A avaliação foi feita pelas métricas *Stretch* e *Stress*, para proporcionar um resultado mais confiável ao protocolo implementado. Este protocolo permite o roteamento de mensagens de forma distribuída, auto-administrada e tolerante a falhas, tendo como estratégia um roteamento baseado em prefixo que é feito por meio de uma DHT.

Através da análise dos resultados, foi possível constatar que o protocolo Tapestry melhora o seu desempenho detectando rapidamente mudanças em seu ambiente. Além disso, cópias adicionais de objetos em *host* ativos podem melhorar significativamente a consulta dos mesmos. Também foi possível observar que, o roteamento dentro da

estrutura do protocolo Tapestry é mais fácil de administrar e reparar. Pois o protocolo foi implementado na camada de aplicação, tornando o mais próximo do usuário programador, mostrando que uma camada *routing overlay* obtêm bom desempenho e alta confiança em uma rede envolvendo muitos nós.

Finalmente, a análise demonstrou que o protocolo Tapestry gerencia informações de roteamento de forma eficiente, e ainda, mostrou-se um protocolo simples, de fácil entendimento e implementação.

5.1. Sugestões de trabalhos futuros

Dando continuidade ao TCC desenvolvido podemos citar como trabalho futuro, a implantação de uma biblioteca com os serviços que o protocolo Tapestry fornece dentro do simulador de rede NS-3, no qual os aplicativos possam usar estruturas *overlay*. Outra sugestão seria uma abordagem de pesquisa comparativa entre o protocolo Tapestry e CAN que possui estratégias diferentes, porém com o mesmo objetivo, evidenciando qual seria o melhor em determinado cenário. Outra proposta está relacionada com o trabalho dos autores Pereira et. al. que avalia o protocolo CAN *Multicast* no suporte a **A**mbientes **V**irtuais **C**olaborativos - AVCs -, neste caso seria a avaliação do protocolo Tapestry no suporte a AVCs. Pois o *Multicast* na camada de aplicação - ALM - tem recentemente surgido como uma boa alternativa ao *Multicast* na camada de rede.

Referências Bibliográficas

AL-FUQAHA, A., GUIZANI, M., KHAN, B., RAYES, A., **Network Modeling and Simulation: A Pratical Perspective**. 2010.

ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, M., MORRIS, R., Resilient Overlay Networks. In: Proc. ACM SOSP, oct. 2001.

ANDROUTSELLIS, S. T., SPINELLIS, D., **A survey of peer-to-peer content distribution technologies**, ACM Computing Surveys (CSUR), vol. 36, pp. 335-371, 2004.

BARCELLOS, Marinho P.; Gaspary, Luciano P. **Segurança em Redes P2P: Princípios, Tecnologias e Desafios.** Simposio Brasileiro de Redes de Computadores, 24 maio 2006, PR. Anais. Curitiba: SBRC, 2006.

BITTORRENT, BitTorrent Delivering the Word's Content. Disponível em <http://www.bittorrent.com>. Acesso em 14 Agosto 2011.

CS, UCLA Parallel Computing Laboratory. Disponível em <http://pcl.cs.ucla.edu/projects/parsec/ >. Acesso em 01 de abr, 2011.

COULOURIS, G., Dollimore, J., Kinberg. **Sistemas Distribuídos**: Conceitos e Projetos. 4. ed. Bookmann, 2007.

CURTI, J. C., **Análise de segurança em aplicações que utilizam plataformas UNIX e MS-Windows como Clientes e Servidores**, Dissertação de Mestrado, UNICAMP, 2004.

CYGWIN, Cygwin. Disponível em < http://www.cygwin.com/ >. Acesso em 31 de mar, 2011.

FIDALGO, Joseane Farias; et. al**. Qnet – Um Simulador Gráfico de Tráfego IP para Redes Convergentes**. 22o Simpósio Brasileiro de Redes de Computadores, maio 2004, Gramado. Anais. Rio Grande do Sul, Gramado: UFPE/SBRC, 2004.

FRANCO, Carlos Eduardo Roriz; et. al.. **Uso de Sistemas Multicore para Simulações de Rede com o NS-3 através de Paralelização**. Out. 2009. (Proposta de Projeto Final de Curso). Instituto de Informática, Universidade Federal de Goiás, Goiânia, Goiás, 2009.

FREENETPROJECT, The Freenet Project. Disponível em <http://freenetproject.org>. Acesso em 14 de Agosto de 2011.

FUGIMOTO, M.; PERUMALLA, S.; RILEY, F.. Network Simulation. 2007.

GNS3, Graphical Network Simulator. Disponível em < http://www.gns3.net/>. Acesso em 01 de abr, 2011.

GOMES, E., **Hidra: Arquivamento Digital de Alta-Confiabilidade Utilizando Auditoria em Redes Peer-to-Peer**, Dissertação de Mestrado, UFPR, 2010.

ICQ, ICQ 7.6 – The latest ICQ. Disponível em <http://www.icq.com>. Acesso em 10 Agosto 2011.

IME, Simulação. Disponível em <http://www.ime.usp.br/~gold/cursos/2002/mac2301/ep2/ep2/node2.html>. Acesso em 29 de mar, 2011.

JUNIOR, João Rocha; et. al.. **XPeer: Middleware for Peer-to-Peer Applications.** Proceedings of Peer-to-Peer Workshop at Brazilian Symposium of Computer Networks, 2005.

JUNIOR, João Rocha; et. Al.. **Peer-to-Peer: Collaborative Computing in the Internet.** Proceedings of the Brazilian Symposium of Computer Networks (SBRC), 2004.

KAMIENSKI, Carlos Alberto; et. al.. **Simulando a Internet: Aplicações na Pesquisa e no Ensino**. 21ª Jornada de Atualização em Informatica, jul. 2002, Congresso da SBC, Florianópolis, SC.

KAZAA, Listen to Music Online at KaZaa. Disponível em < http://www.kazaa.com>. Acesso em 10 Agosto 2011

KLINGBERG, T., MANFREDI, R., Gnutella 0.6, URL http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html, june 2002. Último acesso em 18/09/2011.

LV, Q., CAO, P., COHEN, E. et al. **Search and Replication in Unstructured Peer-to-Peer Networks**, 16th ACM International Conference on Supercomputing, jun. 2002.

MADRUGA, M. **Uma Arquitetura P2P Baseada na Hierarquia do Endereçamento IP com Roteamento Unificado**, Tese de Doutorado, UFRN, 2006.

MILOJICIC, D. S., KALOGERAKI, V., LUKOSE, R., NAGARAJA, K., PRUYNE, J., RICHARD, B., ROLLINS, S., XU, Z., **Peer-to-Peer Computation**, HP Laboratories Palo Alto, jul., 2003.

NAPSTER, Napster Page. Disponível em <http://free.napster.com/napsterhomemain>. Acesso em 10 Agosto 2011.

OBELHEIRO, R. R. **Um Overlay de Roteamento Tolerante a Intrusões**, Tese de Doutorado, UFSC, nov. 2006.

OLIVEIRA, M. A. M. S. **Um Protocolo de Comunicação *Multicast* na Camada de Aplicação com Consciência de Localização**. Tese de Doutorado, UFRN, 2010.

OPNET, Soluções para Gerenciamento de Desempenho de Aplicações. Disponível em <http://www.opnet.com>. Acesso em 1 abr, 2011.

PACESTAR, LAN diagram software – LanFlow. Disponível em <http://www.pacestar.com/lanflow/>. Acesso em 08 Dezembro 2011.

PAPADOPOULOS, Charilaos, **NS3Pastry A PASTRY DHT port to the NS3 Network Simulator.** Department of Informatics; Athens University of Economics and Business. 16 Fev. 2009.

PEREIRA, et. al., **Avaliação do Protocolo CAN *Multicast* no Suporte a Ambientes Virtuais Colaborativos**.V Workshop em Desempenho de Sistemas Computacionais e de Comunicação, 2007.

RATNASAMY, Sylvia; Francis, Paul; Handley, Mark; Karp, Richard; Shenker, Scott **A Scalable Content-Addressable Network.** In ACM SIGCOMM, 2001.

REZENDE, E. S. **Modelo Estrutural para Compartilhamento de Arquivos Peer-to-Peer**, Dissertação de Mestrado, UNESP, 2009.

RNP, Rede Nacional de Ensino e Pesquisa. Disponível em <http://www.rnp.br/>. Acesso em 04 de Dezembro 2011.

ROCHA, João; et. al.. **Peer-to-Peer: Computação Colaborativa na Internet.** Proceedings of the Brazilian Symposium of Computer Networks (SBRC), Maio 2004.

ROWSTRON, A., DRUSCHEL, P. **Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems**. In In IFIP/ACM Internacional Conference on Distributed Systems Platforms (Middleware), November 2001.

SETI@HOME, Search for Extraterrestrial Intelligence at Home. Space Science Laboratory, University of California, Berkley, 2002.

SIMULATOR, Network. The ns-3 network Simulator. Disponível em <http://www.nsnam.org/>. Acesso em 2 abr 2011.

SIMULATOR, Network. Ns-3 Tutorials and Manual. Disponível em <http://www.nsnam.org/docs/release/3.10/doxygen/index.html>. Acesso em 28 mar 2011.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., BALAKRISHNAN, H. **Chord: A Scalable peer-to-peer lookup service for internet applications**, In Proceedings of the ACM SIGCOMM Conference, 2001.

SUBRAMANIAN, R., GOODMAN, B. D. **Peer-to-peer computing: the evolution of a disruptive technology**, Idea Group Inc., 2005.

TANENBAUM, A.S. **Redes de Computadores**, 4ª Edição, Editora Campus, 2003.

TAYLOR, I. J. **From P2P to Web services and grids: peers in a client/server world,** London, Springer, 2005.

THEOTOKIS, Stephanos A.; Spinellis, Diomidis. **A Survey of Peer-to-Peer Content Distribution Techonologies.** Journal. ACM Computing Surveys - CSUR - . 4 Dez 2004.

WMWARE SERVER, Vmware. Disponível em <http://www.vmware.com/products/server/overview.html>. Acesso em 31 de mar, 2011.

ZHAO, Ben; Kubiatowicz, John; Joseph, Anthony. **Tapestry: An infrastructure for fault-tolerant wide-area location and routing.** Computer Science Division University of California, Berkeley. 2001.

ZHAO, Ben; Huang, Ling; Stribling Jeremy; Rhea, Sean C.; Joseph, Anthony D.; Kubiatowicz, John D. **Tapestry: A Resilient Global-scale Overlay for Service Deployment.** IEEE Journal on Selected Areas in Communications. 07 Jan. 2004.