UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2007.1

SIMULAÇÃO DE UMA REDE DHT PARA *PEER*S SEMÂNTICOS NO SPEED

TRABALHO DE GRADUAÇÃO

Aluno – Guilherme Barros de Souza, gbs2@cin.ufpe.br.
Orientadora – Profa. Dra. Ana Carolina Salgado, acs@cin.ufpe.br.

24 de Julho de 2007

UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2007.1

GUILHERME BARROS DE SOUZA

SIMULAÇÃO DE UMA REDE DHT PARA PEERS SEMÂNTICOS NO SPEED

Este trabalho foi apresentado à graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Ana Carolina Salgado

24 de Julho de 2007

Dedico,

Aos meus pais
Arquimedes e Marlene

À minha irmã
Flávia

Agradecimentos

Agradeço aos meus pais, por terem me proporcionado a oportunidade de cursar uma faculdade e o apoio incondicional para que eu conseguisse, enfim, terminar o curso na UFPE e obter meu diploma.

À minha orientadora, a professora Doutora Ana Carolina Salgado, pela confiança e oportunidade de trabalhar neste importante projeto.

À professora Doutora Patrícia Cabral de Azevedo Restelli Tedesco pela boa vontade de me atender diversas vezes no horário do almoço.

Aos meus companheiros de projeto: Carlos Eduardo, Damires Souza, Juliano Souza e Domingos Mendes, que sempre me deram grande apoio.

Aos meus amigos pelo companheirismo e apoio prestado durante o curso.

A todas as pessoas que contribuíram com este trabalho, meus sinceros agradecimentos.

E finalmente, a Deus, por sempre me dar forças nos momentos difíceis para superar os desafios que encontro em minha vida.

Muito Obrigado!

Sumário

Lista de Figuras

Lista de Quadros

1. Introdução

Com o crescimento exponencial da quantidade de dados [Lyman et al. 2003] e uma grande variedade de sistemas de informações, os problemas da integração de dados [Hull 1997, Florescu et al. 1998, Nica 1999] entre estes sistemas têm se tornado um desafio crescente. Vários sistemas foram discutidos, em [Pires et al. 2006], como alternativas para este problema: sistemas de bancos de dados distribuídos, sistemas de bancos de dados federados, sistemas de bancos de dados com esquemas globais e sistemas de integração de dados com esquemas de mediação.

Paralelamente à evolução dos sistemas gerenciadores de dados, houve o desenvolvimento dos sistemas *peer-to-peer*. Este desenvolvimento foi possibilitado pelo aumento da banda na internet e pelo crescimento do poder computacional dos *peer*s da rede [García et al. 2005]. Assim, com uma grande quantidade de *peer*s e uma banda disponível entre eles, o ambiente para desenvolver, comercialmente, aplicações massivas *peer-to-peer* estava montado.

Na tentativa de atingir um modelo mais adequado para um ambiente de integração de dados armazenados em fontes autônomas, heterogêneas e dinâmicas, foram criados os Sistemas de Gerenciamento de Dados *Peer-to-peer* (*Peer Data Management Systems* – PDMS) [Sung et al. 2005]. Estes sistemas possuem as seguintes características: compartilhamento de dados descentralizado, processamento e armazenamento de dados distribuídos em *peer*s autônomos, escalabilidade e mapeamentos semânticos dos dados armazenados nos *peer*s.

Este trabalho se baseia no SPEED, um PDMS que utiliza semântica e uma topologia, com dois níveis de agregação: uma DHT e uma *super-peer*. Essas características facilitam o processamento de consultas e os mapeamentos semânticos entre esquemas.

1.1 Objetivos

Nesse contexto, este trabalho de graduação tem como objetivo fundamentar, documentar e implementar uma simulação para uma rede DHT (*Distributed Hash Table*) [Sung et al. 2005] de *peer*s semânticos como parte de um PDMS. Cada *peer* semântico representa um domínio de conhecimento. A implementação será baseada na arquitetura do SPEED [Pires et al. 2006] e colaborará com duas teses de doutorado e um trabalho de graduação em andamento no CIn/UFPE [Pires 2007, Souza 2007, Mendes 2007].

Basicamente, este trabalho de graduação dará ênfase a três aspectos:

1 – Arcabouço teórico sobre PDMS, *peer* semântico e DHT.

2 – Definição e especificação dos detalhes de implementação e do comportamento do sistema;

3 – Escolha de um simulador [García et al. 2005] para redes *peer-to-peer* e implementação de uma simulação utilizando os conceitos do SPEED;

4 – Desenvolvimento de uma interface gráfica para a exibição dos resultados da simulação.

Além destes aspectos, serão abordados conceitos de *peer-to-peer*, de PDMS e de simuladores de redes *peer-to-peer*, pois são indispensáveis para um melhor entendimento do tema, já que os mesmos fazem parte do contexto deste trabalho.

1.2 Estrutura

Além deste capítulo introdutório, que explanou sobre o contexto e objetivos, este trabalho possui mais seis capítulos.

O Capítulo 2 define os conceitos relativos à *peer-to-peer*, *overlay*, distribuição de informações utilizando DHT, algoritmos para a geração de *overlays*, o algoritmo escolhido: o Symphony [Manku et al. 2003] e ainda uma

introdução sobre PDMS, suas principais funcionalidades e um comparativo entre os PDMS mais conhecidos.

O Capítulo 3 descreve as características de um simulador de rede e suas vantagens de utilização neste trabalho. Além disso, há um comparativo e um detalhamento das principais características do PlanetSim [García et al. 2005], que foi o simulador escolhido para a implementação deste projeto.

O Capítulo 4 mostra os fundamentos do SPEED, suas principais características, o seu funcionamento e detalha seus componentes.

Já o Capítulo 5 aborda a implementação de uma simulação de uma rede DHT para *peers* semânticos, a metodologia adotada e o ambiente de desenvolvimento. Essa seção também detalha a estrutura dos módulos do projeto e suas respectivas especificações.

O Capítulo 6 define o experimento a ser executado, os detalhes da implementação e os resultados obtidos.

Por fim, têm-se o Capítulo 7 que descreve as conclusões com o trabalho aqui apresentado, as contribuições, as principais dificuldades encontradas e propõe trabalhos futuros.

2. *Peer-to-peer* & PDMS

Este capítulo será responsável por abranger a principal teoria necessária ao entendimento da implementação de uma simulação de uma rede DHT para *peers* semânticos no sistema SPEED. Nele abordaremos, inicialmente, conceitos de *peer-to-peer*, os quais são a base do sistema SPEED. Depois serão abordados assuntos relacionados à PDMS, pois assim poderemos absorver e fundamentar algumas das estratégias propostas pelo SPEED. Por fim, analisaremos os principais simuladores de redes *peer-to-peer*, para ratificar a escolha do PlanetSim na realização deste projeto de implementação.

2.1 Peer-to-peer

Sistemas e aplicações *peer-to-peer* são sistemas distribuídos sem qualquer forma de controle centralizado ou hierarquia organizacional, onde o software que está sendo executado em cada *peer* é equivalente em funcionalidade [Stoica et al. 2001].

Esses sistemas têm crescido exponencialmente e, conforme ilustrado na Figura 1, sua utilização com transmissões de arquivos de vídeos são responsáveis por cerca de 60% de todo o tráfego da internet.

O paradigma *peer-to-peer* tem como funcionalidade mais difundida o compartilhamento de arquivos pela internet. Essas redes de compartilhamento são resumidamente redes *overlay* dinâmicas com participação voluntária onde todos os *peer*s participantes trocam arquivos diretamente. [Rocha 2003].

A aplicação *peer-to-peer* que teve maior divulgação foi o SETI@Home [SETI@Home 2002], que utiliza o poder de processamento entre os *peer*s da rede, para executar tarefas de processamento massivo, tais como análise de sinais de rádio em busca de sinais de vida extraterrestre. Desde 2002 ele está em

funcionamento e suas estatísticas até 2006 indicam um total de: 2.433.979,781 anos de CPU e 7,745086 * 10^{21} operações de ponto flutuante.

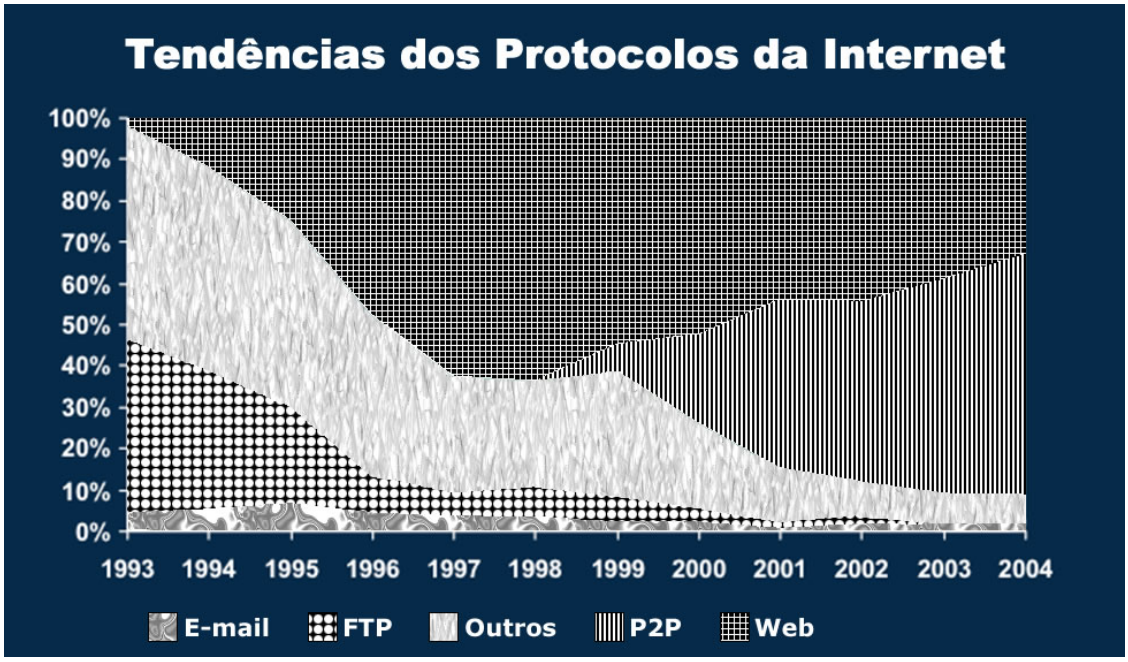


Figura 1: Tendências dos Protocolos da Internet. Adaptado de [CacheLogic 2005].

Dadas as diferentes utilizações das redes *peer-to-peer*, elas foram classificadas nas seguintes topologias: pura ou descentralizada; híbrida ou parcialmente centralizada; e *super-peer*.

Uma topologia pura é composta por *peer*s em um mesmo nível de hierarquia. Todos os *peer*s exercem as mesmas funções na rede. Alguns exemplos de sistemas *peer-to-peer* baseados em topologia pura são: Gnutella [Gnutella 2007] e FreeNet [FreeNet 2007].

A topologia pura, ilustrada na Figura 2b, pode ser classificada em não estruturada e estruturada. A não estruturada se caracteriza pelo uso de *broadcast (flooding)* para o envio de mensagens, tornando a rede menos eficiente e escalável. Já uma topologia pura estruturada é caracterizada pela DHT, que é um conjunto de estrutura de dados que propicia uma melhor busca e acesso a dados distribuídos. Logo o sistema torna-se escalável e mais confiável.

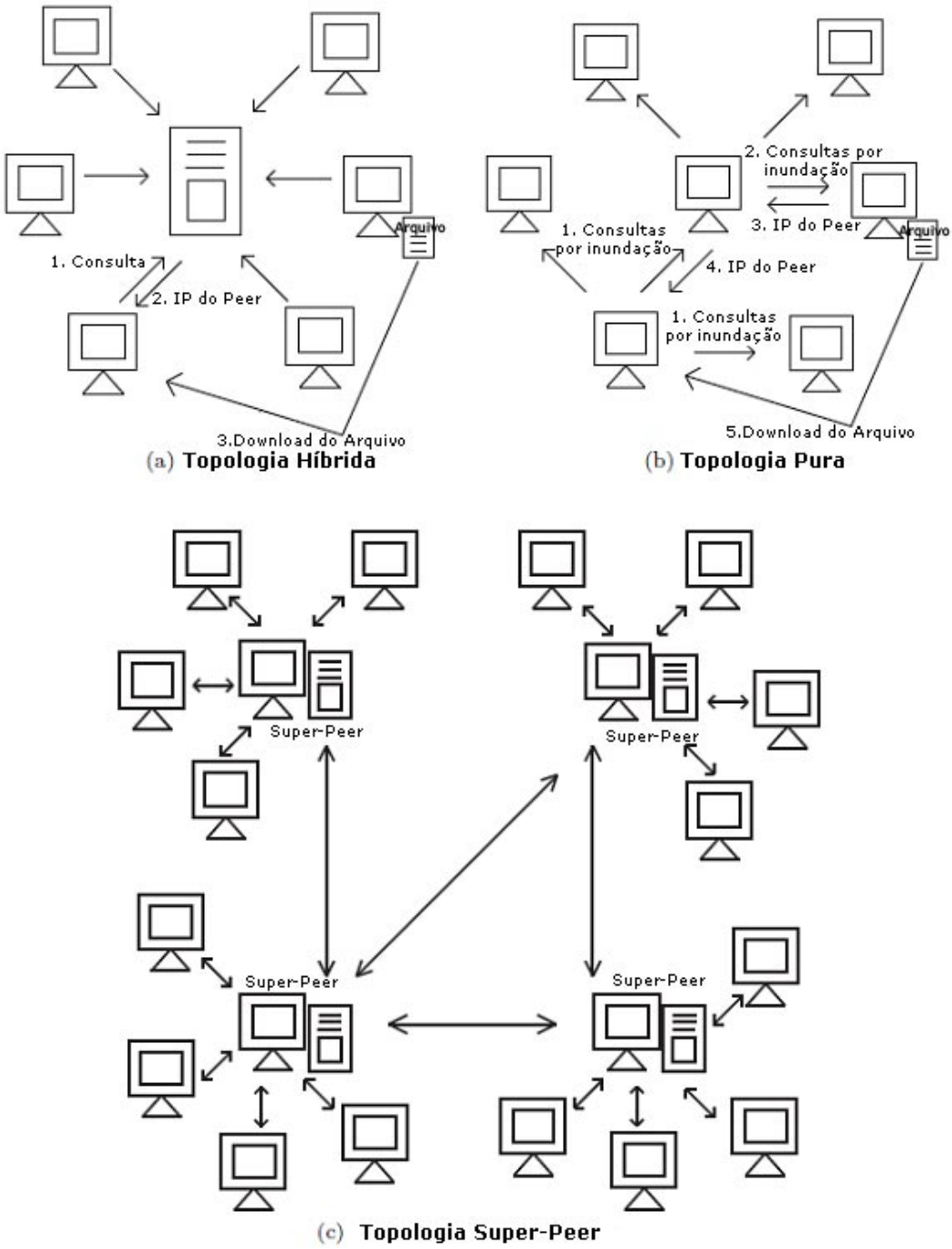


Figura 2: Topologias de sistemas *peer-to-peer*. Adaptado de [Sung et al. 2005].

Já a topologia híbrida, visualizada na Figura 2a, é caracterizada pela existência de uma ou várias entidades servidoras que centralizam o controle da rede. Elas são responsáveis pelas buscas e indexação. Essa topologia é menos escalável e as entidades servidoras representam pontos de falhas. Sem elas, a

rede não opera. Os seguintes aplicativos foram desenvolvidos utilizando a topologia híbrida: Napster [Napster 2007], BitTorrent [BitTorrent 2007] e Instant Messaging [Day et al. 2000].

Por fim, temos a topologia *super-peer*, observada na Figura 2c, onde são eleitos *peer*s de maior capacidade computacional para coordenar os demais *peer*s. Esse *peer* é chamado de *super-peer*. Caso um *super-peer* saia da rede, outro *peer* normal será eleito como um novo *super-peer* para substituí-lo. As aplicações KaZaA [Kazaa 2007] e Morpheus [Morpheus 2007] são baseadas na topologia *super-peer*.

O paradigma *peer-to-peer* possibilita a implementação de sistemas com as seguintes características: volatilidade, auto-organização, tolerância à falhas, descentralização, balanceamento de carga, escalabilidade e roteamento.

2.1.1 Overlay

Uma rede *overlay* é uma rede lógica dentro de uma outra rede. Ou seja, a rede *overlay* é um subconjunto da rede onde se localiza. Os *peer*s compartilham as duas redes, porém, os dados da rede *overlay* são isolados logicamente da outra rede. As linhas tracejadas da Figura 3 representam as conexões entre os *peer*s da camada *overlay*.

A abstração representada pela camada *overlay* faz com que as distâncias entre os *peer*s sejam baseadas na capacidade das conexões entre os *peer*s da rede *overlay* e não na distância física.

Algumas características que podem ser incorporadas a uma rede *overlay* são: maior disponibilidade e desempenho [Amir 2003, Andersen et al. 2001]; e tolerância a faltas e intrusões [Obelheiro 2003].

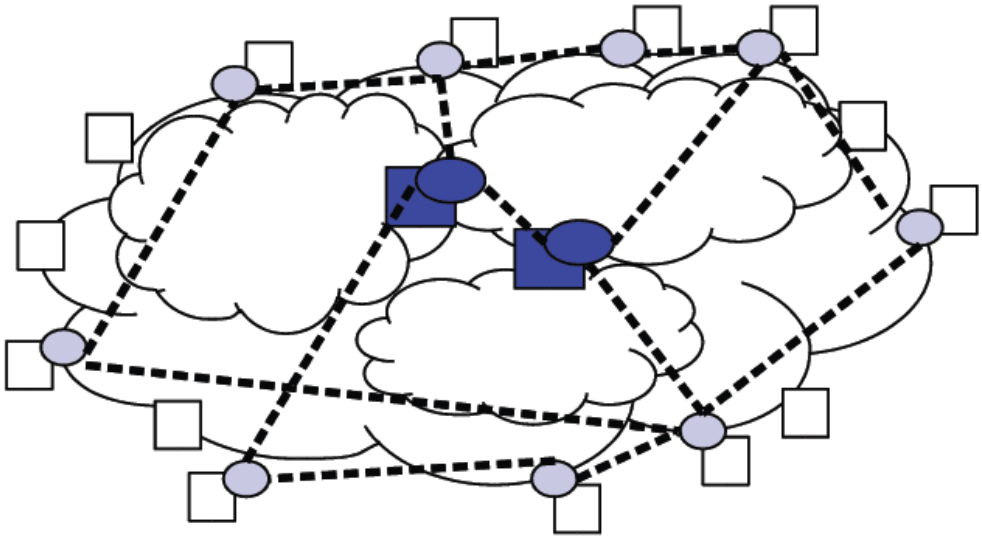


Figura 3: Rede *Overlay* [Neto 2004]

2.2 Distribuição de Informações - DHT

Uma Tabela *Hash* Distribuída (DHT) é um conjunto de tabelas com valores *hash* e dados, distribuídas em diferentes locais, que mapeia chaves em valores. Esse mapeamento se dá através do uso de uma função *hash*, tal como: MD5 [Rivest 1992] ou SHA-1 [Jones 2001], em uma chave, que irá gerar um valor *hash*. No contexto *peer-to-peer* [Stoica et al. 2001], a DHT é responsável pela descentralização do controle da rede, pois, a informação estará distribuída em diversos *peers*.

A DHT tem a seguinte característica: para uma chave K, qualquer peer pode usar a DHT para obter, através da chave K, o endereço do peer responsável. Uma DHT possui duas operações básicas: inserir e procurar. Suas interfaces são definidas da seguinte forma: *inserir (chave, valor)* e *procurar (chave)*.

A operação *inserir* funciona da maneira representada na Figura 4. Já a operação *procurar* funciona, como demonstrado na Figura 5. Ambas as operações sempre têm a chave como entrada e encaminham mensagens, utilizando um algoritmo de roteamento, para o *peer* responsável pela chave.

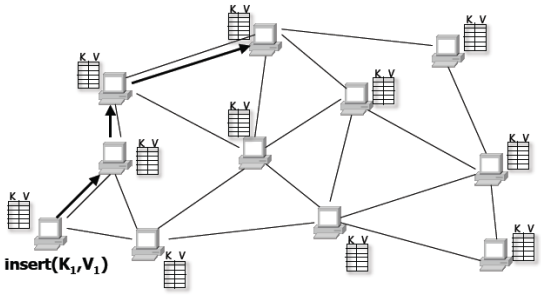


Figura 4: Inserção de um valor [Hellerstein 2004].

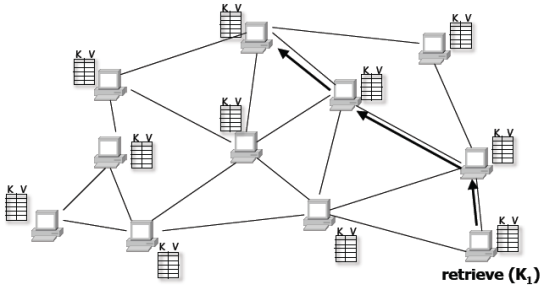


Figura 5: Procura de um valor [Hellerstein 2004].

Sob a DHT há um mecanismo de distribuição de responsabilidade das chaves. Esse mecanismo pode ser provido por diversos algoritmos que formam as geometrias mais conhecidas: *d*-Torus [Ratnasam et al. 2001], anel [Stoica et al. 2001], mesh [Xie 2003], de Bruijn [Sung et al. 2005] e Butterfly [Xie 2003]. Os principais são CAN [Ratnasam et al. 2001] e Chord [Stoica et al. 2001]. Suas formas de operação podem ser visualizadas na Figura 6. Ainda existem outros algoritmos tais como: Tapestry [Zhao et al. 2004], Kademlia [Maymounkov 2002], Viceroy [Malkhi et al. 2002], Koorde [Kaashoek 2003], Pastry [Rowstron et al. 2001] e Symphony [Manku et al. 2003].

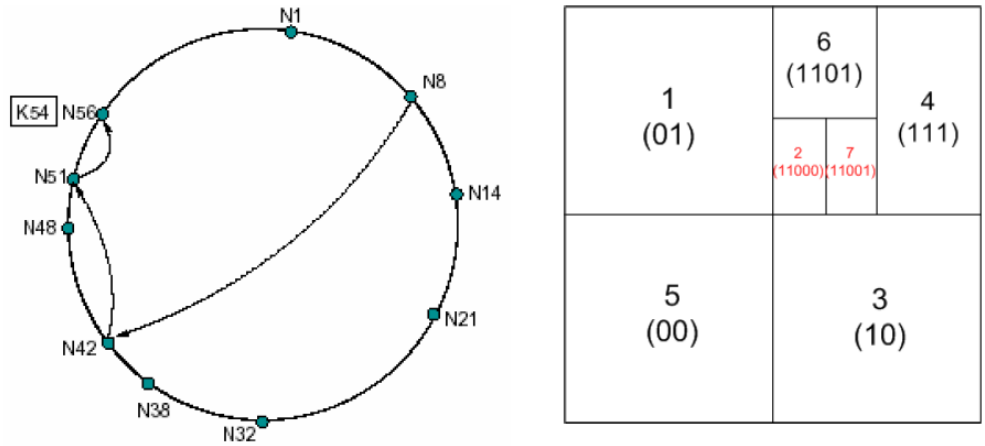


Figura 6: Funcionamento dos Algoritmos Chord e CAN [Pires et al. 2006].

O fator primordial na escolha de um algoritmo para a implementação de uma rede DHT é a sua complexidade computacional. Esta complexidade está descrita no Quadro 1 abaixo, sob as métricas de complexidade no espaço e complexidade no tempo.

Quadro 1: Complexidade dos Algoritmos. Adaptado de [Sung et al. 2005].

DHT	Complexidade de Espaço (Número de Peers)	Complexidade de Tempo (Número de Saltos)
Chord	O(log(n))	O(log(n))
Tapestry (b = base dos IDs)	$O(\log_b(n))$	$O(\log_b(n))$
CAN (d = n. de dimensões)	2d	$O(n^{1/d})$
Pastry	$O(\log_2{}^b(n))$	$O(\log_2{}^b(n))$
Viceroy	7	O(log(n))
Koorde (Degree-2 DeBruijn)	2	O(log(n))
Kademlia	O(log(n))	O(log(n))
Super-peers Estruturados	$O(\sqrt{n})$	O(1)
Symphony	O(log(n))	O(log(n))

Nesse contexto, o algoritmo escolhido para a implementação da rede DHT de *peer*s semânticos do SPEED foi o Symphony devido à sua baixa complexidade, à sua baixa latência [Braunisch 2006], à simplicidade de seu funcionamento e à existência de um exemplo genérico no simulador de redes *peer-to-peer* escolhido. A função da DHT no SPEED é ajudar *peer*s com interesses comuns a encontrarem uns aos outros e facilitar a formação das comunidades semânticas [Souza 2007]. Essas comunidades semânticas representam domínios.

2.2.1 Symphony

O protocolo Symphony é um protocolo de topologia randômica para geração de *overlays* e roteamento de mensagens. Ele se baseia no fato de um *peer,* mesmo sem saber onde está o *peer* destinatário de uma mensagem, mas com informações sobre esse *peer* destinatário, poder encaminhar esta mensagem para outro *peer* que esteja mais próximo do *peer* destinatário [Manku et al. 2003]. Assim, à medida que a mensagem é encaminhada, ela se aproxima mais do *peer* destinatário até encontrá-lo.

As informações dos *peers* destinatários são armazenadas em tabelas *hash* distribuídas (DHT). Essas tabelas são chamadas de tabelas de roteamento [Manku et al. 2003]. As tabelas de roteamento são atualizadas quando um *peer* entra ou sai da rede. As entradas de *peers* podem necessitar de um balanceamento de carga. Já nas saídas é possível ocorrer problemas tais como: perda de informação, desconexão de partes da rede e falta de balanceamento de carga.

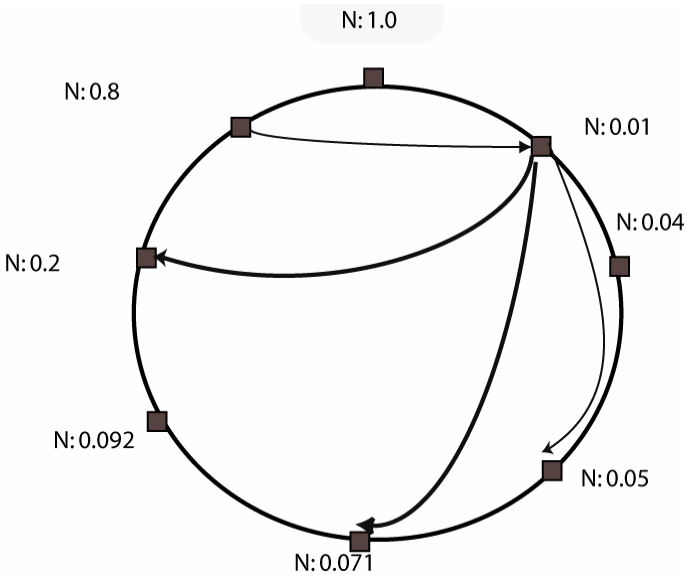


Figura 7: Exemplo de uma tabela de roteamento Symphony [Braunisch 2006].

Os *peers* da rede Symphony possuem identificadores (IDs) que servem como parâmetro para a localização do *peer* na rede. Esses IDs têm valores entre 0 e 1 [Manku et al. 2003].

A Figura 7 exibe as conexões curtas e longas do *peer* N: 0.01 com os outros *peers* da rede. A quantidade de conexões de uma rede é 2k+2, onde k é o número de conexões de longa distância [Manku et al. 2003]. Os outros *peers* dessa rede também possuem tabelas de roteamento. O Quadro 2 detalha a tabela de roteamento Symphony do *peer* N: 0.01 da Figura 7.

Quadro 2: Componentes de uma tabela de roteamento do *peer* N: 0.01 da Figura 7. Adaptado de [Braunisch 2006].

***Peer* de início**	**Conexões de curta distância**		**Conexões curtas reforçadas com tabelas "do vizinho do vizinho", no sentido horário**	**Conexões de longa distância com distribuição harmônica**
	Predecessor	**Sucessor**		
N: 0.01	N: 1.0	N: 0.04	N: 0.05	N: 0.071 ; N: 0.2

O Symphony possui as seguintes características: balanceamento de carga, através da distribuição produzida pela função *hash* que gera os identificadores e do protocolo de estimativa; baixo custo de manutenção, já que são necessárias poucas mensagens para manter a rede; tolerância à falhas, através da redundância de ligações de sucessores, predecessores e *links* de longa distância; e ainda flexibilidade em permitir que cada *peer* gerencie uma quantidade diferente de conexões.

O protocolo de estimativa, que influencia o balanceamento de carga, estima a distância aproximada entre os *peers*, baseada na distância dos seus outros *peers* vizinhos.

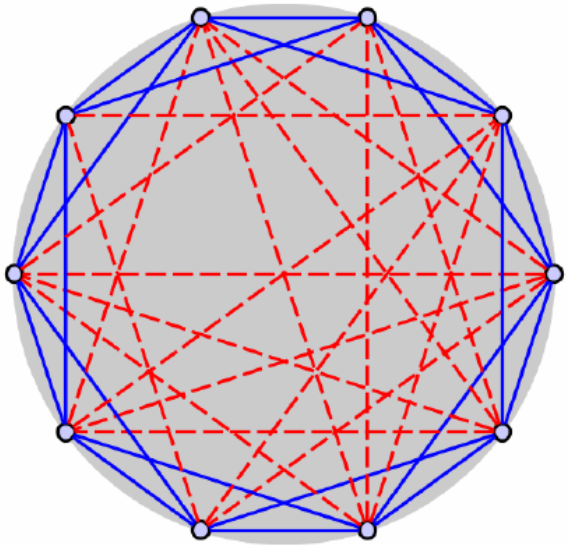


Figura 8: Rede Symphony com 10 *peers* e com no máximo duas conexões de longa distância por *peer*. [Braunisch 2006].

A Figura 8 mostra uma rede com 10 *peers*. Cada *peer* Symphony possui duas conexões curtas com seus vizinhos: a do sucessor e a do predecessor. Estas conexões curtas estão representadas como arestas contínuas. Já os *links* de longa

distância estão representados como arestas tracejadas e são ligações randômicas com alguns *peers* que não são seus vizinhos. Estas ligações randômicas são obtidas através do protocolo de estimativa.

Para uma aplicação executando sobre o Symphony, ele suporta apenas uma operação: dado uma chave, ele mapeia a chave em um *peer*. Esse mapeamento é realizado através de uma função *Hash*, tal como SHA-1 [Jones 2001] e será utilizado pela DHT conforme citado na Seção 2.2. A aplicação utilizará essa operação tanto na inserção de alguma chave, quanto na busca de dessa chave.

2.3 PDMS

Os PDMS são Sistemas de Gerenciamento de Dados *Peer-to-Peer* (*Peer Data Management Systems*) que provêem um ambiente de integração de dados armazenados em fontes autônomas, heterogêneas e dinâmicas.

Um PDMS é composto por uma quantidade excessiva de *peer*s. Cada *peer* representa uma fonte de dados e disponibiliza seus dados através de esquemas exportados, ou seja, a parte do esquema em que um *peer* deseja compartilhar com os demais *peer*s [Pires 2007]. Estes sistemas possuem as seguintes características: compartilhamento de dados descentralizado, processamento e armazenamento de dados distribuídos em *peer*s autônomos, escalabilidade e mapeamentos semânticos dos dados armazenados nos *peer*s.

O mapeamento de dados descentralizado é mister para evitar *peer*s de falhas que inviabilizem a rede. Caso algum *peer* do PDMS fique indisponível, os outros *peer*s irão continuar a funcionar, sem que a rede fique inoperante.

A escalabilidade é necessária devido à grande quantidade de *peer*s que compõem a rede. A rede deverá manter serviços de buscas e indexação sem que haja um grande custo computacional associado a cada operação.

Os *peer*s que compõem um PDMS são considerados autônomos, pois não há controle sobre a entrada ou saída destes *peer*s, nem mesmo sobre as alterações dos esquemas de suas bases de dados.

Um dos diferenciais entre os outros sistemas de integração de dados é a substituição de um único esquema global por uma coleção de mapeamentos semânticos entre os esquemas individuais dos *peer*s que representam as fontes de dados.

Para que um PDMS seja eficiente, ele precisa ter um sistema de indexação e processamento de consultas eficazes. Segundo [Hose et al. 2006], estes são os maiores desafios no desenvolvimento de PDMS para redes de larga escala.

2.3.1 Principais PDMS

Com o objetivo de destacar as principais semelhanças e diferenças entre os PDMS, o Quadro 3 mostra os principais PDMS: Edutella, Local Relational Model, Piazza, Chatty Web, PeerDB, Xpeer, DBGlobe, RDFPeers, NeuroGrid e Mutant Query Plans [Risson 2004]. Os critérios da classificação foram os seguintes: topologia da rede, linguagem/esquema de marcação e linguagem de consulta.

2.4 Considerações

Neste capítulo abordamos as principais características dos PDMS que serão desejáveis em nossa implementação da rede DHT para o SPEED. Também fizemos uma análise entre os PDMS mais conhecidos para verificar estratégias de funcionamento desses PDMS em relação ao funcionamento do SPEED [Pires 2007].

Quadro 3: Comparativo entre os principais PDMS. Adaptado de [Risson 2004].

Nome	Topologia da Rede	Linguagem de Esquema/Marcação	Linguagem de Consulta
SPEED	DHT e *super-peer*.	–	–
Edutella, Semantic Overlay Clusters, Hypercup	Hipercubos *super-peer*s, agrupamentos baseados em ontologias, consultas ou regras.	Resource Description Framework (RDF) e Schema (RDFS).	Edutella's RDF Query Exchange Language (QEL).
Local Relational Model	–	XML	–
Piazza	–	XML ou RDF; XML Schema ou Ontologias OWL.	Baseada em XQuery.
Chatty Web	Descoberta semelhante ao ping/pong Gnutella.	XML	Seleção genérica, projeção e operadores de mapeamento, XQuery.
PeerDB	Pequenos saltos aos *peer*s que mais recentemente proveram respostas.	–	SQL
Xpeer	Rede auto-organizada de *peer*s e *super-peer*s.	XML	XQuery FLWOR
DBGlobe	Hierarquia em três camadas: Objetos Primários Móveis, Servidores Administradores de células e Filtros.	Extensões do XML Schema	Extensões da XQuery
RDFPeers, Mult-Attribute Addressable Network	Armazena tuplas em uma extensão de Chord; *super-peer*s Shuns; Usa visão do sucessor para balanceamento de carga.	RDFS	Consultas Nativas ou RDQL
NeuroGrid	Cada *peer* tem uma base de conhecimento e uma tabela de Identificadores Únicos Globais para identificar as consultas; mecanismos semelhantes ao Gnutella para previnir *loopings*; Aprende a partir das interações dos usuários.	RDF	–
Mutant Query Plans	Os *peer*s recebem uma ou mais funcionalidades – servidor base, servidor de indexação, servidor de meta-indexação, servidor de categorias; Consultas roteadas por catálogos hierárquicos distribuídos.	XML	Operadores e dados de consultas que acumulam resultados parciais antes de retornar para o solicitante com o resultado completo.

3. Simuladores de Redes *Peer-to-peer*

Os simuladores são as ferramentas mais populares para investigar as redes *overlay* e as aplicações *peer-to-peer* [Naicken et al. 2007]. Essa popularidade se deve ao alto custo de montar um ambiente *peer-to-peer* para testes em larga escala. Além disso, a validação de algoritmos e soluções em um ambiente controlado tem um resultado mais preciso. Entende-se por um ambiente controlado, um ambiente sem problemas de capacidade de processamento, banda e com quantidade de máquinas suficientes. Este ambiente pode ser provido e/ou abstraído por simuladores.

Apesar do resultado de uma simulação não ser igual ao da execução em um ambiente real, os simuladores tentam, através da simulação de falhas e eventos, deixar esse resultado mais próximo da realidade.

No caso dos simuladores *peer-to-peer*, os mais conhecidos [Brown 2006] foram avaliados. Eles foram os seguintes: Narses [Narses 2003], 3LS [Ting 2003], P2PSim [P2PSim 2005], NeuroGrid [Neurogrid 2001], PlanetSim [García et al. 2005], PeerSim [PeerSim 2005], OMNet++ [OMNet++ 2006], NS2 [NS-2 2007] e SSFNet [SFFNet 2004].

Quadro 4: Quadro comparativo de simuladores *peer-to-peer*. Adaptado de [Brown 2006].

Simulador	Quantidade Máxima de *Peers*	Código Fonte Disponível	Versão	Linguagem	Dependente de Plataforma
3LS	10^3	não	-	-	sim / Linux
Narses	6.10^2	sim	0.1	Java	sim / Linux
NS2	-	parcialmente	2.0	Otcl	-
NeuroGrid	3.10^5	sim	0.0.5	Java	não
OMNeT++	10^3	sim	3.3	C++	-
P2PSim	3.10^3	sim	0.3	C++	sim / Linux
PeerSim	10^6	sim	1.0.1	Java	não
PlanetSim	10^5	sim	3.0	Java	não
SimJava	-	sim	2	Java	não
SSF	$3{,}3.10^4$	não	2	Java/C	-

Os critérios de avaliação foram estes: disponibilidade do código fonte, linguagem de programação, sistema operacional, simulação de *overlay*, escalabilidade, funcionamento, utilização de padrões de projeto, usabilidade do simulador, documentação existente, licença e se projeto ainda está em desenvolvimento.

Após análises a partir do quadro acima, o PlanetSim foi o escolhido para simular o projeto de uma rede DHT para *peer*s semânticos no sistema SPEED, já que é escalável, possui código fonte disponível, está em desenvolvimento, é programado em Java, é independente de plataforma e segue padrões de desenvolvimento [Pressman 2006]. A escolha da linguagem Java ocorreu devido à maior familiariedade do autor com ela. As principais características do PlanetSim serão descritas na Seção 3.1. Outro simulador que obteve bom desempenho foi o PeerSim. Ele será utilizado na simulação da rede *super-peer* [Mendes 2007], em outra camada da arquitetura do SPEED [Pires et al. 2006].

3.1 O Simulador PlanetSim

O PlanetSim é um framework orientado a objetos para simulações de redes *overlays* e serviços [García et al. 2005]. Este simulador apresenta uma arquitetura modular e bem definida, seguindo os padrões clássicos de projeto [Pressman 2006]. Assim é necessário um menor aprendizado para se implementar novas simulações neste ambiente. A arquitetura do simulador pode ser visualizada na Figura 9.

Por ser dividido em camadas (rede, *overlay* e aplicação) e possuir um baixo acoplamento, podemos alterar os dados de uma camada, que as demais continuarão com o mesmo funcionamento. Isto possibilita a implementação apenas de um serviço ou apenas de uma nova rede *overlay*, sem que seja necessário alterar as outras partes do sistema.

Outra característica importante deste simulador é a construção baseada em interfaces, facilitando suas alterações ou extensões. Também é possível, por meio da substituição das implementações das interfaces, a migração do ambiente de simulação para um ambiente com uma estrutura de rede.

Essas interfaces servem para dividir o simulador em camadas, conforme disposto na Figura 9. As três camadas são as seguintes: aplicação, *overlay* e rede. Estas camadas possuem uma comunicação bidirecional com suas camadas vizinhas. A camada de aplicação foi padronizada baseada na implementação de interfaces do FreePastry [FreePastry 2007] na *Common API*. As interfaces oriundas desta implementação foram as seguintes: *Application*, *EndPoint*, *Message*, *RouteMessage*, *Id* e *NodeHandle*. Esta API serve de "fachada" para o sistema de roteamento. A camada de aplicação se comunica diretamente com a camada de *overlay*.

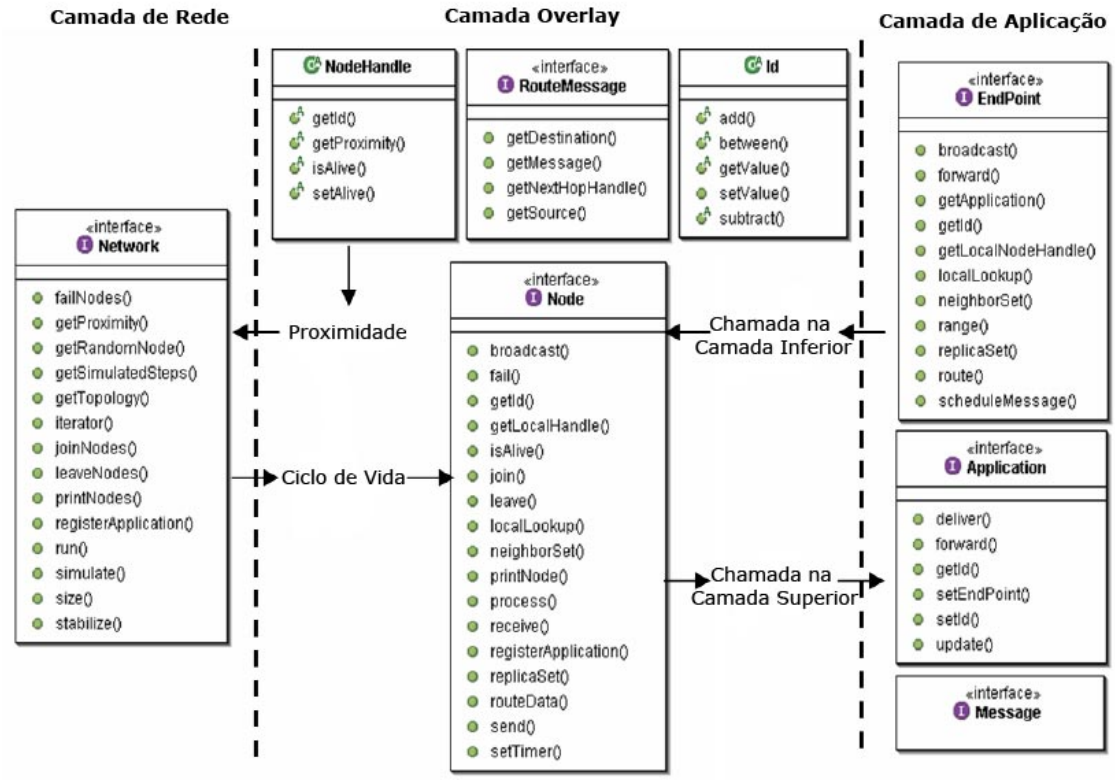


Figura 9: Arquitetura em Camadas do PlanetSim. Adaptado de [García et al. 2005].

A camada de *overlay* é a responsável pela organização entre os *peer*s. Ela possui filas de entrada e de saída de mensagens. Também possui métodos para enviar, receber e processar essas mensagens. Cada *peer* desta camada deve possuir especificações de como enviar e de como reagir à chegada de mensagens. Esta camada se comunica tanto com a camada de aplicação, quanto com a camada de rede.

Juntamente com a camada de rede, a camada de *overlay* caracteriza o ciclo de vida dos *peer*s pelos métodos padrões (*join*, *leave*, *fail* e *process*). Esses *peer*s são identificados principalmente por três entidades: *Id, IdFactory* e *NodeHandle*.

A camada de rede é a camada que determina o ciclo geral da simulação. O ciclo é dividido em passos, em cada passo mensagens são processadas, encaminhadas e recebidas. Esta camada representa o local onde as mensagens são roteadas. Esses roteamentos podem ter custos associados e reorganização da topologia. A camada de rede pode ser estendida com o uso de conexões TCPs e UDPs, sobre uma rede com endereços IPs reais.

O PlanetSim possui um *kernel* modularizado que exerce as principais atividades do simulador. A Figura 10 mostra a comunicação entre os componentes do *kernel* do PlanetSim.

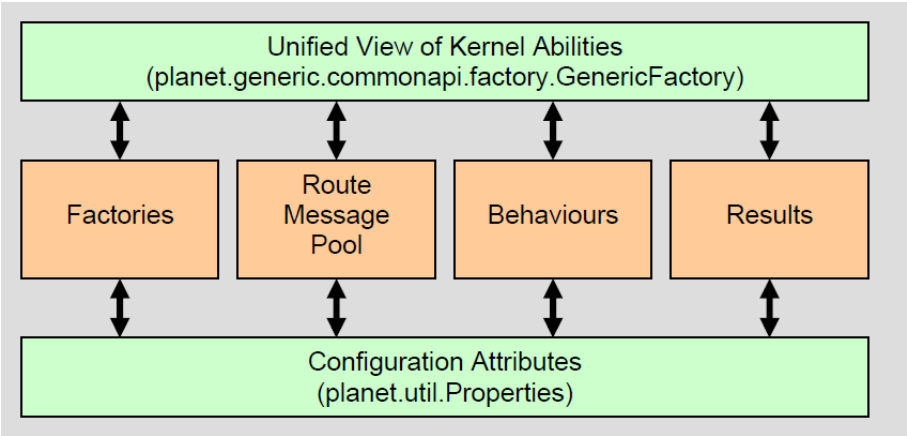


Figura 10: Componentes do *kernel* do PlanetSim [García et al. 2005].

As *factories* são advindas do padrão de projeto *Factory Method* e são uma coleção de classes com a habilidade de gerar instâncias de algum tipo. As

seguintes *factories* estão implementadas e podem ser estendidas: *NetworkFactory, NodeFactory, NodeHandleFactory, IdFactory, EndPointFactory* e *ApplicationFactory*.

O componente *RouteMessagePool* é um *pool* e uma *factory* das mensagens que são roteadas para estabilizar a rede. Este *pool* é necessário para minimizar a utilização de recursos do sistema, através da reutilização de mensagens.

O módulo *Behaviours* define um processamento personalizado, utilizando a noção de comportamento, para as mensagens que chegam a um *peer*. Assim, com o uso de arquivos de configurações, é possível adicionar e remover comportamentos sem a necessidade de recompilar o sistema.

O componente *Results* é um gerador de resultados em diferentes formatos (GML e Pajek) que foca na geração de grafos para representar a topologia de uma rede. O funcionamento desse módulo é possível através da visita dos *peer*s e da verificação de suas ligações com outros *peer*s. Diferentes formatos de resultados podem ser gerados em uma mesma simulação.

O *Unified View of Kernel Abilities* é uma camada pública e genérica que agrupa as características de uma simulação de rede para os desenvolvedores. Ela é baseada na classe *planet.generic.commonapi.factory.GenericFactory*.

O componente *Configuration Attributes* é usado para especificar o contexto de uma simulação em arquivos textos. Logo, mudando-se os arquivos de configuração é possível mudar a simulação, sem que seja necessário recompilar o código. A hierarquia dos arquivos de configurações pode ser vista na Figura 11.

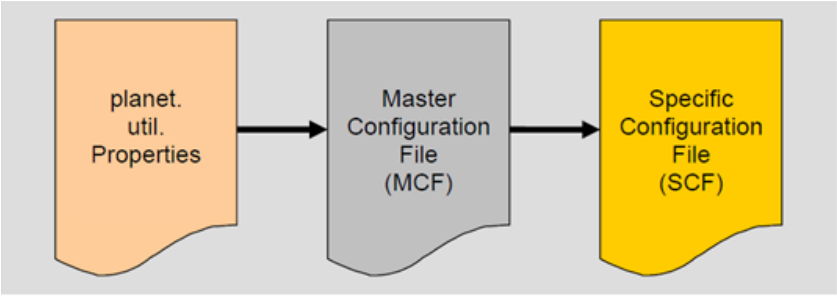


Figura 11: Hierarquia das configurações da simulação no PlanetSim [García et al. 2005].

A classe que representa as configurações é a *planet.util.Properties*, já o MCF indica qual SCF deve ser usado para cada teste. O SCF possui todos os atributos referentes ao *kernel* e ao teste específico.

3.2 Considerações

Neste capítulo abordamos o funcionamento do PlanetSim, um ambiente controlado, de baixo custo e com um perfil que suporta uma implementação do SPEED. O PlanetSim simulará os conceitos teóricos da arquitetura SPEED. Esses conceitos serão detalhados no próximo capítulo.

4. SPEED

O sistema SPEED vem sendo pesquisado e desenvolvido com o propósito de prover soluções para problemas críticos de gerenciamento de dados em sistemas *peer-to-peer*, como conectividade, mapeamentos entre esquemas, processamento de consultas e qualidade de serviços. Para isso, utiliza semântica como base para o desenvolvimento e gerenciamento de seus serviços [Souza 2007].

Nas próximas seções apresentaremos a arquitetura do SPEED e os conceitos de todos os seus componentes.

4.1 Arquitetura do SPEED

A arquitetura do Sistema *Peer-to-peer* de Gerenciamento de Dados Baseado em Semântica (ou Semantic Peer Data Management System – SPEED) foi formulada por [Pires et al. 2006] e é composta por diferentes categorias de *peer*s e relacionamentos entre eles. Existem três tipos de *peers*: os *peers* semânticos, os *peers* de integração e os *peers* de dados. O SPEED foi projetado para utilizar duas topologias de redes distintas: a DHT como um anel superior e mais abstrato, que organiza os *peers* semânticos, e a *super-peer* para o gerenciamento de pontos de integração e pontos de dados. A arquitetura completa do sistema SPEED pode ser observada na Figura 12.

A topologia DHT foi abordada na Seção 2.2 e forma a camada de mais alto nível do sistema. Ela tem a funcionalidade de localizar e encaminhar os novos *peer*s e as consultas para serem processados no melhor *peer* semântico. Já a topologia *super-peer* é abordada em [Mendes 2007] e realiza a localização dos *peers* de integração, o balanceamento dos *clusters*.

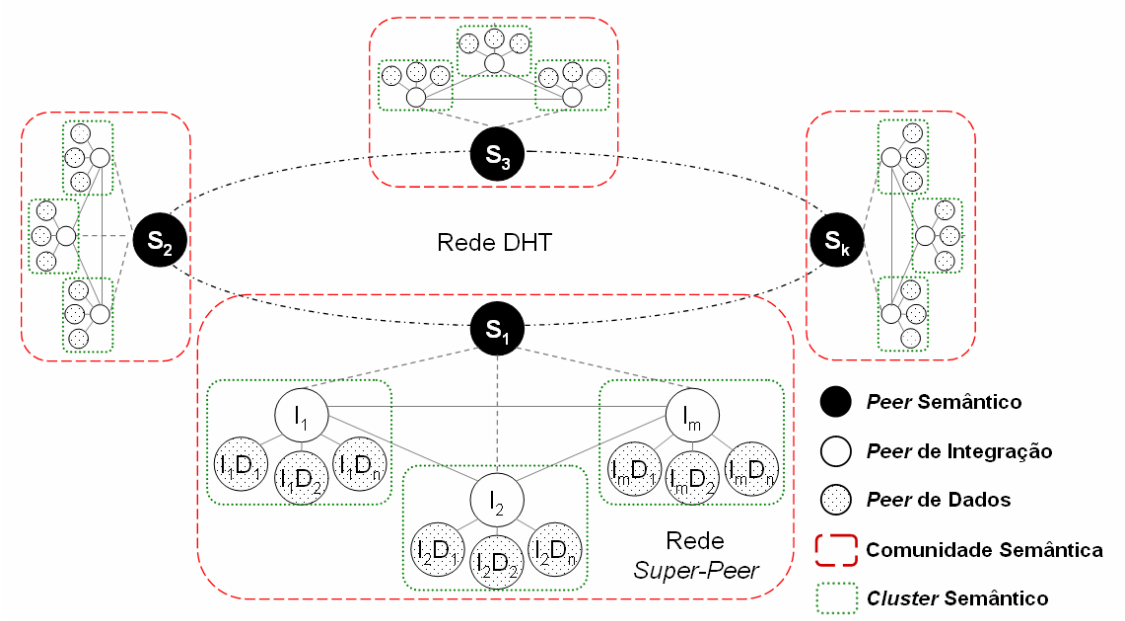


Figura 12: Visão Geral da Arquitetura SPEED. Adaptado de [Pires 2007].

Fazem parte da topologia DHT os *peers* semânticos, representados na Figura 12 por S_1, S_2, S_3 e S_k. Esses *peers* atuam como servidores de ontologias padrões específicas de domínios. Estas ontologias de domínio são usadas para enriquecer semanticamente esquemas exportados de *peers* de dados e, eficientemente, distribuir *peers* de dados dentro de comunidades semânticas e *clusters* [Souza 2007].

Os *peers* semânticos servem de pontos de entrada das comunidades semânticas. Este gerenciamento é composto pela entrada e saída de *peers*, balanceamento de *peers* e resolução de onde um *peer* que deseja se conectar a rede deverá ser conectado.

Os *peers* de integração estão ilustrados por I_1, I_2 e I_m na Figura 12. Eles são *peers* de dados diferenciados em um *cluster* semântico por oferecer melhores recursos computacionais. Este *peer* possui um conhecimento detalhado sobre os *peers* de dados de um *cluster* [Pires 2007]. Os *peers* de integração se comunicam tanto com os outros *peers* de integração quanto com os *peers* de dados que estão conectados a eles. A necessidade de possuir melhores recursos computacionais é

justificada pela maior capacidade computacional exigida para efetuar controle e processamento de consultas sobre os *peers* de dados.

Já os *peers* de dados são quaisquer *peers* (por exemplo, um simples computador ou um servidor) que pode conectar-se e/ou desconectar-se frequentemente da rede *peer-to-peer*. Um *peer* de dados corresponde a uma fonte de dados cujas informações são compartilhadas com outros *peers* mediante o estabelecimento de mapeamentos semânticos [Pires 2007]. Os *peers* de dados são os locais onde são executadas as consultas. Estes *peer*s podem ser visualizados na Figura 12 sob a legenda de I_1D_1, I_1D_2, I_1D_n, I_1D_1, I_1D_2, I_1D_n, I_mD_1, I_mD_2 e I_mD_n.

O *cluster* semântico é um conceito lógico e serve para associar um conjunto de *peers* de dados a um *peer* de integração com interesses semânticos semelhantes. Cada *cluster* semântico está associado a um interesse comum entre *peer*s de dados e possui um *peer* de integração que é responsável por tarefas como indexação de metadados, processamento de consultas e integração dos dados [Souza 2007]. Os clusters servem para prover um ambiente estável onde serão aplicadas as técnicas de associação dos esquemas exportados.

Por fim a comunidade semântica é o mais alto nível de um conjunto semântico. Ela é um conceito lógico para o agrupamento de *clusters* semânticos que possuem interesses semânticos em comum.

4.2 Principais Componentes

Nesta seção será apresentada a composição dos principais componentes da arquitetura do sistema SPEED. A Figura 13 representa os componentes dos três tipos de *peers* presentes no sistema.

O *peer* semântico é um componente formado por uma camada responsável pela comunicação na rede *peer-to-peer*, um comparador semântico e uma base de conhecimento. A arquitetura do *peer* semântico está representada na Figura 13. A camada *peer-to-peer* é utilizada para a comunicação na rede DHT com os outros

*peer*s semânticos e os *peers* semânticos de sua comunidade. O comparador semântico serve para localizar a melhor comunidade para o *peer* de dados entrante. Também ajuda no balanceamento e na manutenção da conectividade nos *clusters* semânticos. Por fim, a base de conhecimento serve de repositório para as informações que mantêm a comunidade semântica.

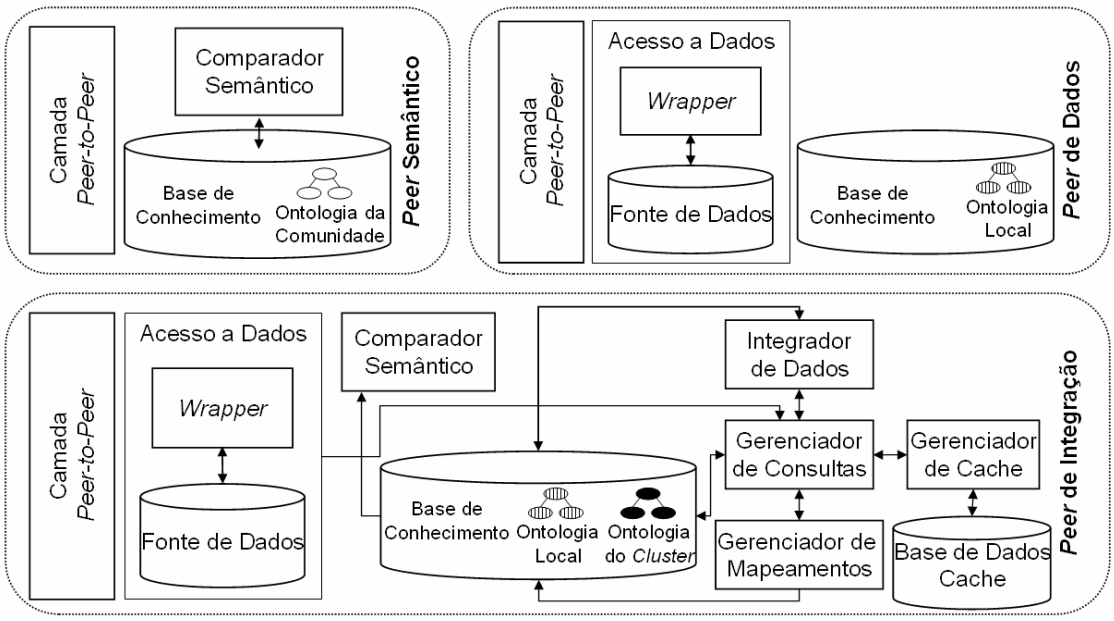


Figura 13: Componentes das arquiteturas dos *peer*s. Adaptado de [Pires et al. 2006].

O *peer* de dados é também formado por uma camada de comunicação com a rede *peer-to-peer*, que realiza a comunicação com um *peer* de integração. Além dessa camada, ele possui uma camada de acesso a dados e uma base de conhecimento. A camada de acesso a dados é composta por uma fonte de dados e um wrapper. Esta fonte de dados contém os dados disponíveis no *peer*. O *wrapper* é utilizado para compatibilizar a comunicação entre a linguagem da consulta do sistema e a linguagem do *peer* de dados. Já a base de conhecimento contém informações sobre a ontologia local. A ontologia local é responsável por representar os dados de um *peer* de dados.

Por fim, o *peer* de integração pode ser considerado uma extensão do *peer* de dados, pois o mesmo possui as mesmas funcionalidades do *peer* de dados.

Além disso, o *peer* de integração possui os seguintes componentes: o comparador semântico, o integrador de dados, o gerenciador de mapeamentos, o gerenciador de consultas, o gerenciador de cache e a base de dados cache.

O integrador de dados é o componente que integra os resultados dos *peers* de dados e dos *peer*s de integração. Em seguida, encaminha esses resultados para o *peer* que originou a consulta. Já o gerenciador de mapeamentos gera e mantém mapeamentos semânticos entre esquemas exportados (mapeados para ontologias) com a ontologia de referência do *cluster,* e entre ontologias de *peers* de integração [Souza 2007]. O gerenciador de consultas recebe as consultas e, de acordo com a base de conhecimento, processa as consultas e/ou encaminha para os *peers* de dados ideais e para os outros *peers* de integração. Já a camada *peer-to-peer* realiza a comunicação entre os *peers* de integração, os *peers* de dados e os *peers* semânticos. A base de conhecimento armazena informações úteis para a realização de tarefas como associação de esquemas exportados, processamento de consultas e integração de dados [Souza 2007].

4.3 Considerações

Neste capítulo mostramos os detalhes da arquitetura SPEED. Sua arquitetura é formada por duas topologias. Uma rede DHT e uma rede *super-peer*. Os principais componentes são os *peers* semânticos, os *peers* de integração e os *peers* de dados. Este PDMS possui o uso da semântica e a topologia mista como diferenciais. O uso da semântica facilita os mapeamentos entre esquemas e o processamento de consultas.

5. Rede DHT para *Peer*s Semânticos

Neste capítulo vamos detalhar o funcionamento de uma rede DHT para os *peers* semânticos do sistema SPEED que será analisado com o uso do simulador PlanetSim, visto na Seção 3.1.

A rede DHT para *peers* semânticos é uma rede com informações de roteamento distribuídas em *peers* com significado semântico. Essa distribuição de informações possui vantagens como observado nas Seções 2.1 e 2.2.

Essa rede DHT é a topologia da camada superior na arquitetura SPEED, descrita na Seção 4.1. O diferencial desta camada é o uso da semântica para agrupamento de seus *peers* semânticos. Assim, o processamento de consultas e os mapeamentos semânticos entre esquemas são facilitados.

Este capítulo aborda inicialmente o funcionamento e a implementação, em módulos, da simulação da rede DHT na arquitetura SPEED. Após essa abordagem, a metodologia e o ambiente de desenvolvimento serão detalhados.

5.1 Módulos do Projeto

Utilizando a estratégia de dividir-para-conquistar [Demaine 2001] e os bons princípios da programação orientada a objetos [Pressman 2006], a modularização do projeto foi realizada. Esta modularização provê uma maior coesão e um menor acoplamento do sistema. Assim, futuras alterações ou implementações serão realizadas mais facilmente.

O projeto foi dividido em três módulos: o módulo DHT foi desenvolvido a partir de um exemplo presente no simulador PlanetSim, o módulo semântico foi totalmente desenvolvido baseado na arquitetura SPEED e o módulo gráfico foi todo elaborado utilizando componentes gráficos Java e um exemplo da API *prefuse* [Prefuse 2007]. Esses módulos serão detalhados nas próximas seções.

5.1.1 Módulo Semântico

O módulo semântico é o responsável por realizar a decisão, a qual *peer* semântico o determinado *peer* pertence. Assim, ele possibilita que o *peer* ou a consulta sejam encaminhados para o destino correto. O resultado do módulo semântico sempre será apenas uma palavra ou identificador que classifique unicamente e uma dada entrada, pois essa classificação exata será utilizada para encaminhar a mensagem na rede DHT.

Essa decisão deverá ser apoiada em ontologias e comparação de esquemas para um resultado mais confiável. Devido à alta complexidade do assunto, o presente trabalho não aborda o uso de ontologias, nem de comparação entre esquemas. Neste trabalho, a decisão será baseada em mapeamentos de palavras – domínio, onde o domínio é representado por estas palavras.

Especificação

Neste trabalho o módulo semântico terá como principal componente uma função. Esta função é responsável por determinar o *peer* semântico de destino ideal para uma dada entrada. A função de classificação é baseada em um mapeamento palavra – domínio. Este mapeamento vai simular a operação com o uso de ontologias e com a comparação de esquemas. Assim, como exemplos, podemos citar o parâmetro "corn" que fará parte do domínio "agriculture" e o parâmetro "god" do domínio "religion".

Caso a rede tenha outro significado semântico mais adequado, basta que haja a substituição desta função para que a classificação seja realizada de uma nova maneira.

5.1.2 Módulo DHT

O módulo DHT é o responsável pela organização da rede de *peer*s semânticos e pelo encaminhamento de mensagens entre esses *peer*s. A organização da rede é realizada através das tabelas de roteamento de cada *peer* semântico. A tabela de roteamento possui valores e os identificadores (IDs) responsáveis por esses valores. Além destas informações, cada *peer* semântico possui informações sobre os IDs vizinhos. Tanto as informações das tabelas, quanto as dos vizinhos são essenciais para a manutenção da rede.

Especificação

As operações realizadas neste módulo podem ser classificadas como: na *overlay* e sobre a *overlay*. As operações na *overlay* são a entrada, a busca e a saída de *peer*s semânticos. Já as ações realizadas sobre a *overlay* são a busca, a remoção e a inserção de palavras-chave que representam o domínio associado ao *peer* semântico.

Quando o *peer* semântico "C" entra na rede, tanto as tabelas de roteamento dos *peers* semânticos "A" e "B" que são responsáveis pela faixa de ID, quanto a tabela de roteamento do *peer* "C" serão atualizadas. A atualização consiste em atribuir ao *peer* semântico entrante a responsabilidade do valor que representa a faixa de IDs vizinhos. Um exemplo de mudança das responsabilidades após a inserção de um *peer* na rede pode ser visualizado na Figura 14.

Ao sair da rede, os IDs que estavam sob responsabilidade do *peer* semântico que saiu deverão ser redistribuídos para os seus vizinhos, através da atualização de suas tabelas de roteamento. A função utilizada para geração destes IDs foi a SHA-1 [Jones 2001]. Esta função é considerada segura, pois ainda não há registros de colisões de *hashes*, ou seja, *hashes* iguais gerados a partir de entradas diferentes.

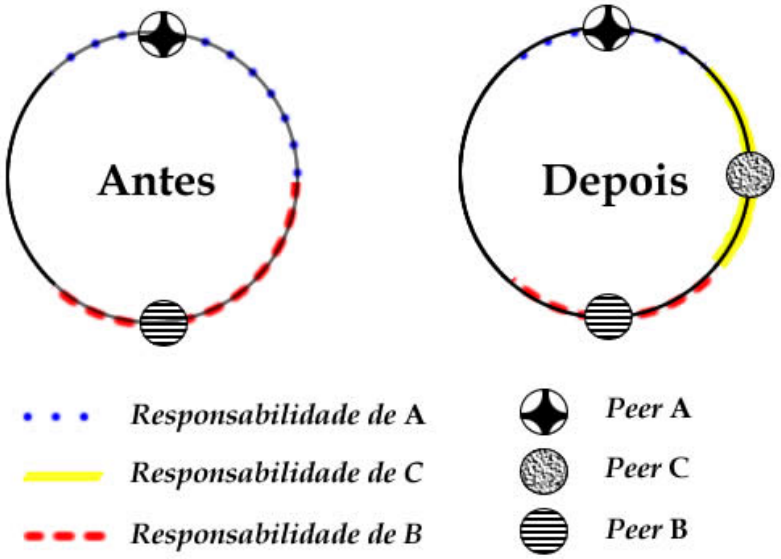


Figura 14: Mudança das responsabilidades na inserção de um *peer*.

Os *peers* semânticos são identificados por IDs. Como entrada para a geração de IDs, utilizamos a palavra-chave do domínio que representa este *peer* semântico. Por exemplo, o domínio "art" com o ID 0,7767297815567344 representa a palavra-chave "modern art".

A inserção de *peers* de dados e de integração ocorre a partir de qualquer *peer* semântico da rede e tem como objetivo inseri-lo no *peer* semântico que melhor represente o domínio deste *peer* entrante. Caso não exista um domínio que possa ser representado por um *peer* semântico, o *peer* entrante não será inserido na rede.

Já a busca por *peers* de dados e de integração também pode ser realizada a partir de qualquer *peer* semântico e deverá retornar a mensagem de sucesso, caso o *peer* tenha sido encontrado. A informação de falha é exibida, caso o *peer* semântico representante não exista ou se existir, caso o *peer* de dados ou de integração não esteja no *peer* semântico.

5.1.3 Módulo Gráfico

O módulo gráfico é o módulo responsável por exibir a topologia da rede DHT de *peer*s semânticos. Esta topologia é composta por um anel de *peer*s semânticos. Os *peers* de integração e de dados, que representarão a camada *super-peer* em futuras implementações, estarão conectados aos *peers* semânticos que representam os seus domínios. Possibilitando uma personalização da exibição do gráfico, existe um painel de controle, onde é possível alterar opções que refletem alterações no gráfico e na topologia exibida. Este módulo também deverá exibir as mensagens oriundas da simulação realizada no PlanetSim.

Especificação

O módulo gráfico irá plotar topologias baseado no resultado da simulação realizada no PlanetSim. O resultado original do PlanetSim é a exibição de textos, porém esta não representa bem uma rede. Assim, este módulo gera grafos com *peers* semânticos que caracterizam o anel da rede DHT e com *peer*s de integração e de dados ligados a este anel facilitando a visualização da rede.

Além da área da topologia, existe um painel de controle com as seguintes opções visuais: mostrar os *peers* de integração e de dados, e visualizar arestas oriundas das tabelas de roteamento. Assim, podemos observar diferentes gráficos, a partir de uma mesma simulação.

A cada simulação, a tela da topologia é exibida. Quando as opções da exibição são alteradas e aplicadas, a topologia é redesenhada com os novos parâmetros. Os conceitos de rede DHT, topologia em anel e das camadas da arquitetura SPEED, são representados pela interface gráfica. Para que as mensagens exibidas, durante a simulação, no simulador PlanetSim fossem visualizadas, foi necessário adicionar um *console* à interface gráfica.

As operações disponibilizadas pelo módulo gráfico serão as seguintes: componente para a inserção de *peers* semânticos, componente para a inserção de

peers de integração e de dados, componente para remoção de *peers* de integração e de dados e, componente para a busca por *peers* de integração e de dados.

5.2 Ambiente de Desenvolvimento

O ambiente foi o Eclipse 3.1.2 [Eclipse 2005] e teve sua linguagem de programação escolhida o Java 5 [Java 2007], pois essa era a linguagem utilizada para o desenvolvimento do PlanetSim, o simulador de redes *peer-to-peer*. Outra vantagem percebida com a utilização de Java foi o uso de seus componentes, juntamente com a API *Prefuse* [Prefuse 2007], disponível apenas em Java, que facilitou a criação da representação da topologia da rede.

5.3 Considerações

Neste capítulo a implementação da rede DHT na arquitetura SPEED foi detalhada em módulos. Esses módulos foram divididos em DHT, semântico e gráfico. Cada módulo foi detalhado e especificado. Por fim, a metodologia e o ambiente de desenvolvimento foram descritos.

6. Experimento e Resultados

Neste capítulo vamos detalhar o experimento da implementação de uma rede DHT para *peers* semânticos no sistema SPEED em duas etapas e depois faremos considerações a respeito de seus resultados.

Os dados utilizados nesse experimento são um subconjunto do Urban Dictionary [UD 2007]. Este dicionário é composto por definições e relacionamentos entre palavras, ou seja, a palavra "art" possui sua definição e palavras relacionadas presentes no Quadro 5.

Quadro 5: Mapeamento de palavras relacionadas com o domínio [UD 2007].

Domínio	Palavras Relacionadas
art	music, graffiti, artist, painting, design, emo, photography, poetry, beauty, briggsy, goth, paint, rock, tattoo, vandalism, architecture, art school, artistic, artists, deviant art, literature, modern, beautiful, college, creativity, drawing, fashion, film, gothic, talent, abstract, books, art apprenticeship, cool, expression, fun, indie, love e modern.

As palavras relacionadas do Quadro 5 formam o domínio "art", ou seja, existe na rede DHT um *peer* semântico que representa este domínio. O conjunto de dados possui 18 domínios: "agriculture", "anthropology", "art", "astronomy", "biology", "chemistry", "economics", "engineer", "geography", "history", "journalist", "law", "medicine", "paleontology", "philosophy", "physics", "psychology" e "religion". Esses domínios são compostos por 460 palavras-chave que representam termos dos domínios associados aos *peer*s semânticos.

Todos os dados deste experimento tiveram que ser pré-processados, ou seja, palavras-chave que representavam mais de um domínio foram excluídas. Isto foi necessário para permitir que a função de classificação fosse implementada de acordo com o método de classificação da interface, logo, com apenas uma classificação.

Esta função de classificação do módulo semântico também foi baseada no Urban Dictionary [UD 2007] e conseguiu obter até 18 saídas, pois esta foi a

quantidade de domínios selecionados do Urban Dictionary. Caso outra função de classificação seja adotada, basta criar uma classe que implemente a interface *IFuncaoDeClassificacao*.

Os dados que são classificados representam os *peers* de dados e de integração. Neste protótipo os *peers* de dados e de integração são representados por palavras-chave, já que este trabalho não aborda a implementação da camada *super-peer*, onde se localizam os *peers* de dados e de integração.

6.1 Construção do Experimento

A construção do experimento iniciou com a execução realizada através do arquivo *DHTTest.java.* Ela se inicia com a criação da rede de *peers* semânticos. Há a leitura da massa de dados para suprir a infra-estrutura do simulador de redes *overlays*, PlanetSim, na criação de uma rede com cinco *peers* semânticos. Estes cinco *peers* são subconjuntos dos dados utilizados no módulo semântico. Os domínios escolhidos foram: "art", "chemistry", "engineer", "geography" e "philosophy". Os *peers* originais do PlanetSim sofreram modificações para receber um nome, o qual será a base para a construção do ID do *peer* semântico. Este nome representa a classificação gerada pela função de classificação do módulo semântico descrito na Seção 5.1.1.

A mensagem exibida na criação da rede *overlay* é mostrada a seguir:

```
Starting creation of 5 nodes...
5 nodes created OK with [60] steps and [0.031] seconds.
```

Após a criação da rede DHT com os cinco *peers*, há a inserção dos *peers* de dados, representados por palavras-chave. Essas palavras-chave são subconjuntos aleatórios das palavras que compõem cada um dos cinco domínios. Essas palavras são oriundas de um arquivo de texto com o nome do domínio. Um dos subconjuntos utilizados foi o "chemistry", representado pelo arquivo

chemistry.txt, com as seguintes palavras: “organic”, ”organic chemistry”, “hydrogen”, “lab”, “periodic table”, “random”, “acid”, “analytical”, “thermodynamics” e “wacky”. A implementação deste exemplo permite o carregamento de massas de dados maiores. Para isto, basta apenas adicionar mais arquivos de texto na pasta de dados. O carregamento de dados original do simulador não leva em consideração que esses *peers* de dados são classificados e distribuídos de acordo com a função de classificação do módulo semântico. Esta funcionalidade foi totalmente desenvolvida.

O mecanismo de roteamento presente no PlanetSim também foi modificado para que o roteamento fosse realizado baseado no resultado do módulo semântico e não na palavra-chave.

Os *peers* semânticos já recebem o nome oriundo das possíveis classificações da função classificadora representada pela classe *FuncaoUrbanDictionary*.java. Já que todos os *peers* semânticos foram inseridos e a rede encontra-se estabilizada, a primeira etapa da construção está concluída.

Uma parte da mensagem de saída após a execução da inserção dos *peers* de dados representados por palavras-chave está disposta a seguir:

```
Starting key insertion...
...
INSERT At 'art' 0.3096701149232435'(Id Gerado)  Palavra-Chave = 'modern
art'

      INFO: 'love' Stored At art ('0.7767297815567344')

 INSERT At 'chemistry' 0.5083190194166411'(Id Gerado)       Palavra-
Chave = 'organic'

      INFO: 'modern art' Stored At art ('0.7767297815567344')

 INSERT At 'chemistry' 0.5083190194166411'(Id Gerado)       Palavra-
Chave = 'organic chemistry'

      INFO: 'organic' Stored At chemistry ('0.7767297815567344')
```

O formato da saída acima foi incrementado com as palavras-chave e o nome dos domínios que representam os *peers* semânticos.

Com a inserção de todos os *peers* de dados, a rede encontra-se povoada. Para verificar a corretude destas inserções, é necessário realizar a busca por todas estas palavras. Novamente foi necessário alterar o mecanismo de buscas presente no PlanetSim, já que o original procurava pelas palavras-chave e realizava o mecanismo de roteamento baseado nas mesmas. Esta construção foi alterada para que o roteamento fosse realizado através do resultado do módulo semântico, ou seja, do domínio. Ainda houve a necessidade de incluir os dados do *peer* semântico na mensagem roteada, pois era necessário verificar se a palavra-chave tinha sido encontrada no *peer* semântico de destino.

A seguir, uma parte da mensagem de saída exibida no *console* após a realização de buscas por *peers* de dados. Estes *peers* de dados foram representados por palavras-chave. As buscas foram realizadas a partir de um *peer* semântico qualquer:

```
Starting lookups...
...
LOOKUP Of Target Palavra-Chave 'lake' PS 'geography' at
'0.6972074154027985'
      INFO: 'williams' Binded to geography '0.6972074154027985'  ....OK

 LOOKUP Of Target Palavra-Chave 'wind' PS 'geography' at
'0.6972074154027985'
      INFO: 'lake' Binded to geography '0.6972074154027985'....OK

 LOOKUP Of Target Palavra-Chave 'existentialism' PS 'philosophy' at
'0.00273414830472365'
      INFO: 'wind' Binded to geography '0.6972074154027985'....OK
```

O formato de saída acima também foi incrementado com as palavras-chave e o nome dos domínios que representam os *peers* semânticos.

A Figura 15 mostra a rede DHT de *peers* semânticos formada, antes da inserção dos *peers* de dados. Também podemos observar na Figura 15 o porquê da classificação da rede DHT como uma topologia anel.

Com a rede de *peers* semânticos formada, há a inserção dos *peers* de dados. O conteúdo dos *peers* de dados a serem inseridos já se encontra em uma estrutura de dados, que foi povoada pela leitura dos arquivos de texto presentes na pasta

de dados. A partir de qualquer *peer* semântico, a inserção é realizada através do método *insert(palavraChave)*, o qual gera uma mensagem com a *palavraChave* e a encaminha na rede. Há uma consulta à função de classificação da *palavraChave* presente no módulo semântico, depois uma função *hash* será aplicada a *nomeDoPeerSemantico* para gerar um identificador (ID). Este ID será comparado com as entradas da tabela de roteamento (sucessor, predecessor e *links* de longa distância) do *peer* semântico, onde as operações estão sendo realizadas. Caso não haja o ID exato, o *peer* semântico irá encaminhar para outro *peer* semântico onde o ID seja mais próximo do ID gerado.

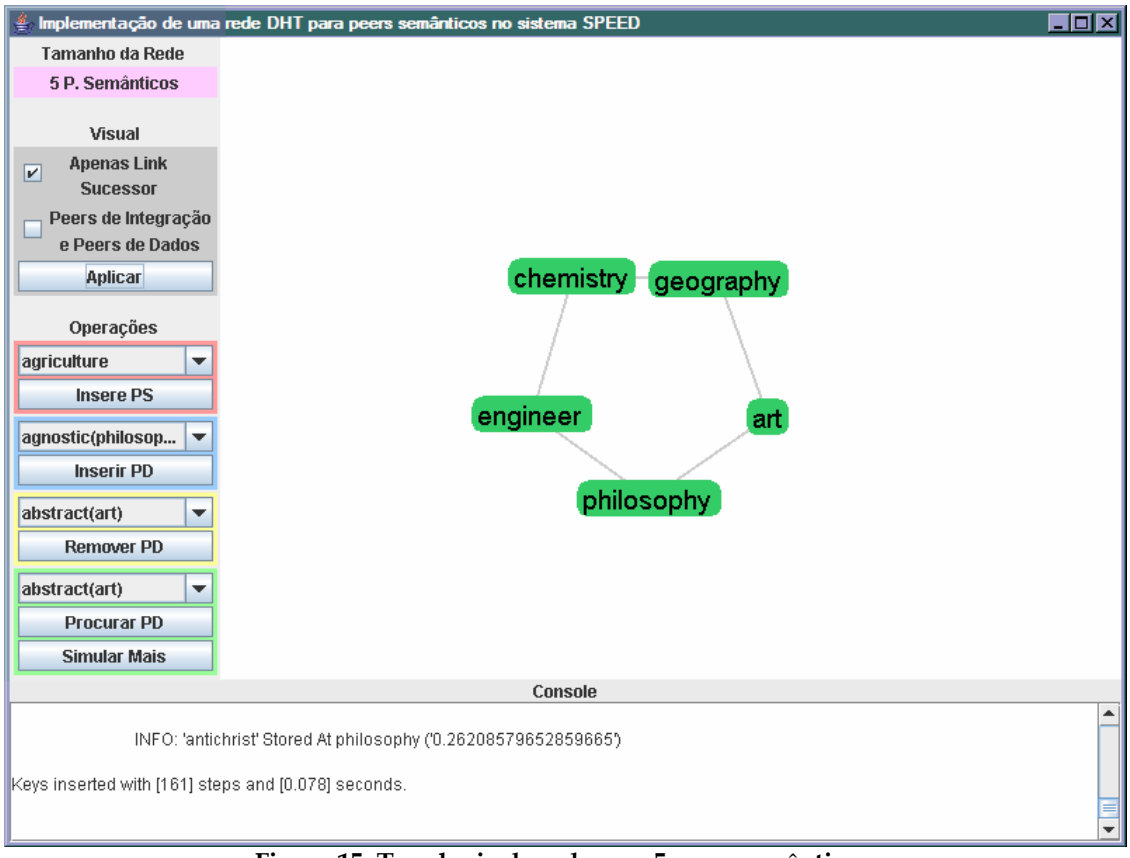


Figura 15: Topologia da rede com 5 *peer*s semânticos.

Assim, como os IDs presentes em cada entrada das tabelas de roteamento são organizados com o protocolo de estimativa através de uma função de distribuição probabilística, o custo para localizar um *peer* nesta implementação será de O(log n). Portanto, temos um sistema escalável.

Depois da inserção de todas as palavras-chave, que representam os *peer*s de dados, nos respectivos *peer*s semânticos, uma lista com todos os *peers* de dados será iterada e estes são procurados a partir de um *peer* semântico qualquer.

Após a verificação de que todos os *peers* de dados encontram-se inseridos, o módulo gráfico é inicializado. Este módulo está disposto na Figura 16 e é composto por três partes: o painel de topologia da rede, o painel de controle e o *console*. O painel de controle e o *console* foram totalmente construídos com a utilização de componentes Java. O painel de topologia foi completamente construído utilizando, além dos componentes Java, a API *Prefuse* [Prefuse 2007]. Esta API tem a funcionalidade de gerar grafos a partir de arquivos XML. Este XML contém a descrição (nome e valor) de todos os *peer*s pertencentes ao grafo e suas respectivas arestas. Logo, o XML representa a topologia da rede.

A API é configurada por parâmetros que representam: a cor das arestas, a cor dos *peers*, a forma dos *peers*, nível de interação e a força de atração entre os *peers*. Através do método *new GraphMLReader().readGraph("/graph.xml")*,

O XML foi gerado pela iteração das estruturas de dados que compunham a rede. Assim, com uma entrada do tipo *NetworkImpl*, os *peers* da rede são iterados e suas ligações com outros *peers* são descobertas. Essas ligações são observadas ao acessar a tabela de roteamento. Ao final destas iterações, o arquivo XML é gerado, com seus respectivos *peer*s e suas ligações.

O painel de controle permite que o usuário visualize ou oculte os *links* da tabela de roteamento e os de longa distância; visualize ou oculte os *peers* de integração e de dados; realize a operação de inserção de novos *peers* semânticos e inserir, procurar e remover *peers* de dados. Há também o *console* que foi criado a partir de um painel de texto que recebe todas as mensagens que foram encaminhadas ao *console* padrão de Java. Podemos observar o painel de controle à esquerda e o da topologia à direita.

Como resultado inicial do experimento, temos a interface gráfica presente na Figura 16.

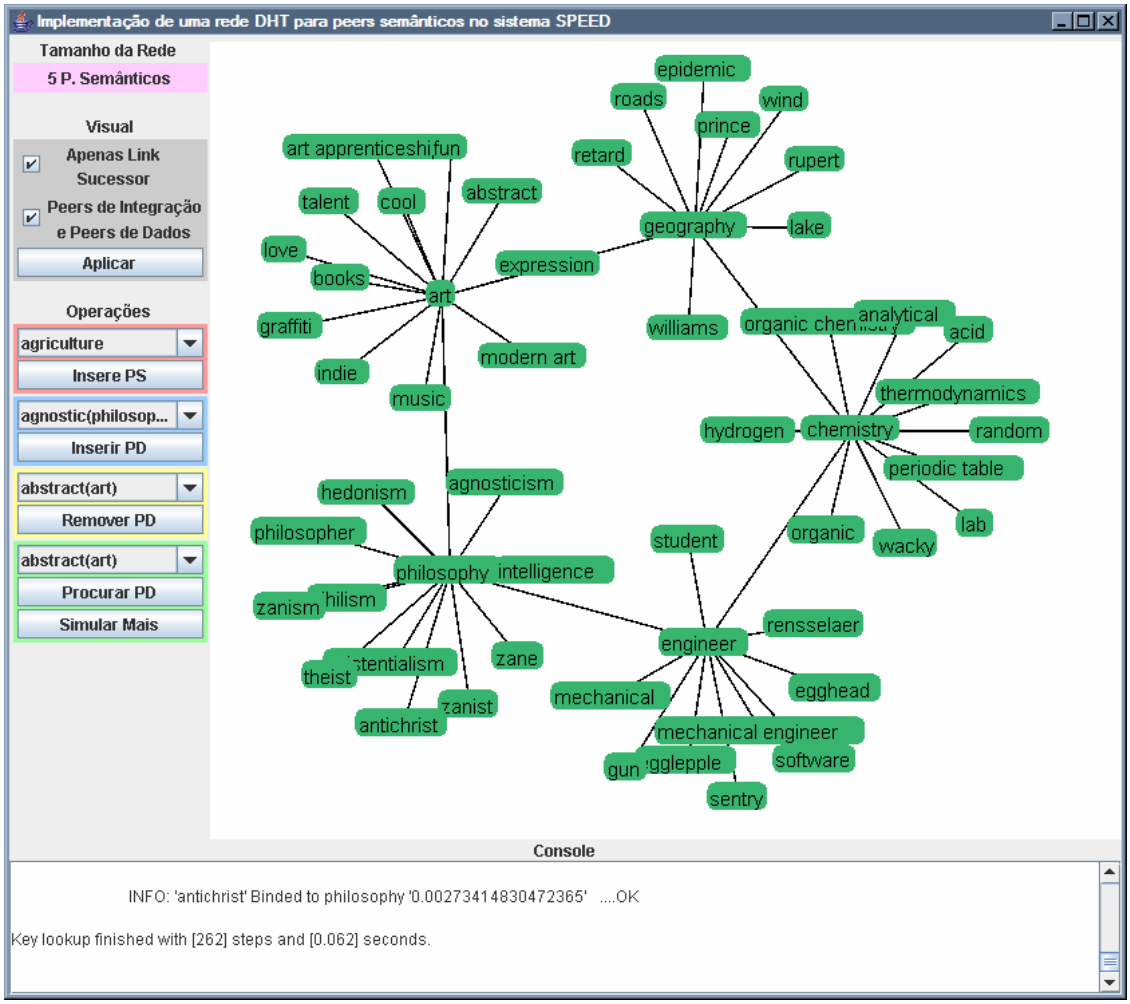


Figura 16: Rede DHT com 5 *peers* semânticos e 66 *peers* de dados.

A interação do administrador do sistema SPEED com o painel de topologia da interface gráfica se dá através das seguintes maneiras:

- Clicar e arrastar em uma área vazia com o botão esquerdo, o qual irá movimentar todo o grafo;
- Clicar e arrastar um *peer* com o botão esquerdo, o qual irá movimentar apenas o *peer* no grafo;
- Clicar e arrastar com o botão direito: para cima, diminuirá o zoom; para baixo, aumentará o zoom.

A topologia é traçada inicialmente de forma aleatória, porém, com o passar do tempo, os *peer*s vão se repelindo e a forma de anel vai sendo caracterizada.

Apesar de simples, a presente interface gráfica atende aos requisitos, já que consegue representar e manipular os conceitos de rede DHT, topologia em anel e de uma das camadas da arquitetura SPEED.

6.2 Manipulação do Experimento

A manipulação do experimento é realizada através do uso de todas as opções do painel de controle. Este fato marca a segunda fase do experimento. O estado inicial do experimento pode ser visualizado na Figura 16. As opções são divididas em visuais e de operação.

6.2.1 Opções Visuais

As opções de visuais padrões são o *checkbox* "Apenas Link Sucessor" e o *checkbox* "*Peers* de Integração e *Peers* de Dados". Nesta seção, os *peers* de integração e de dados serão ocultados e as ligações da tabela de roteamento e do predecessor serão exibidas. O resultado pode ser visualizado na Figura 17.

A topologia da Figura 17 também foi submetida ao aumento de zoom, para que a conexão entre os *peers* semânticos fosse melhor visualizada.

6.2.2 Opções de Operação

Esta seção aborda as quatro operações disponíveis: inserção de *peers* semânticos, inserção de *peers* de dados (PD), remoção de *peers* de dados e busca por *peers* de dados.

A operação de inserção de *peer* semântico é executada através da seleção de "astronomy" presente no primeiro elemento da Figura 18a e do clique no botão "Insere PS". O resultado é o aumento do tamanho da rede de *peers*

semânticos de cinco para seis e o aparecimento do *peer* "astronomy", no grafo que representa a topologia da rede, entre os *peers* semânticos "philosophy" e "art". Estes resultados podem ser vistos na Figura 19.

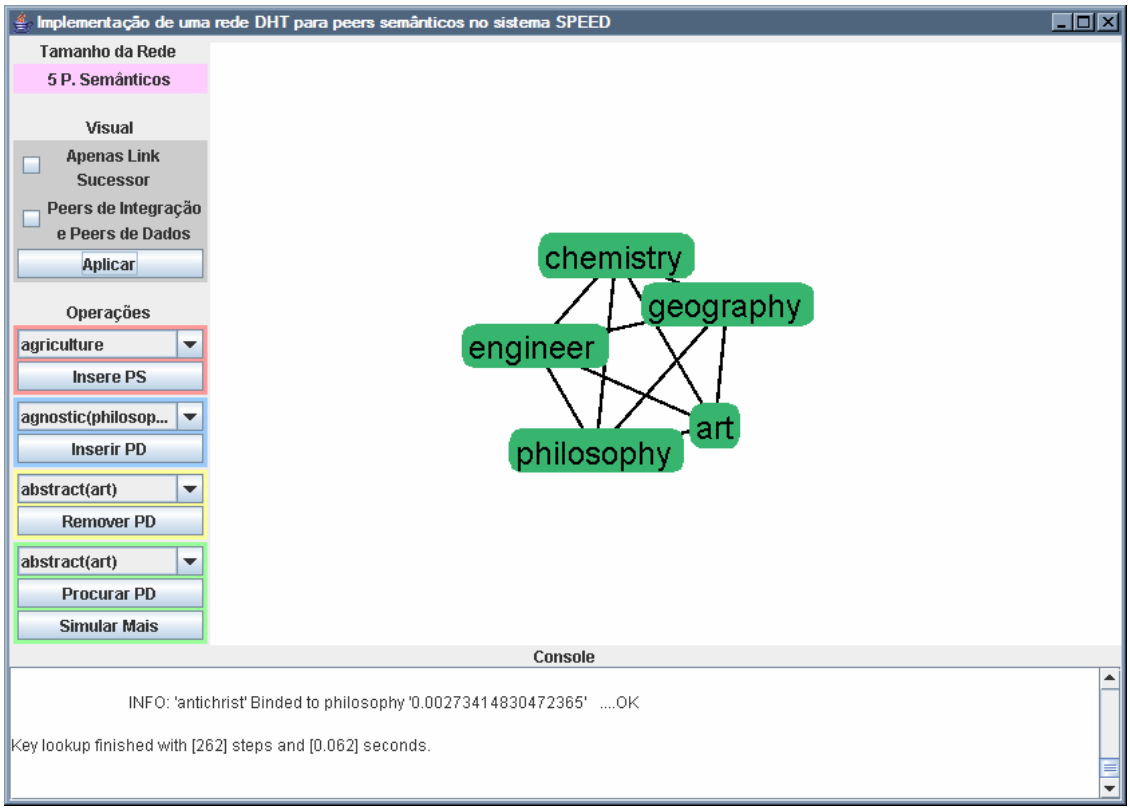
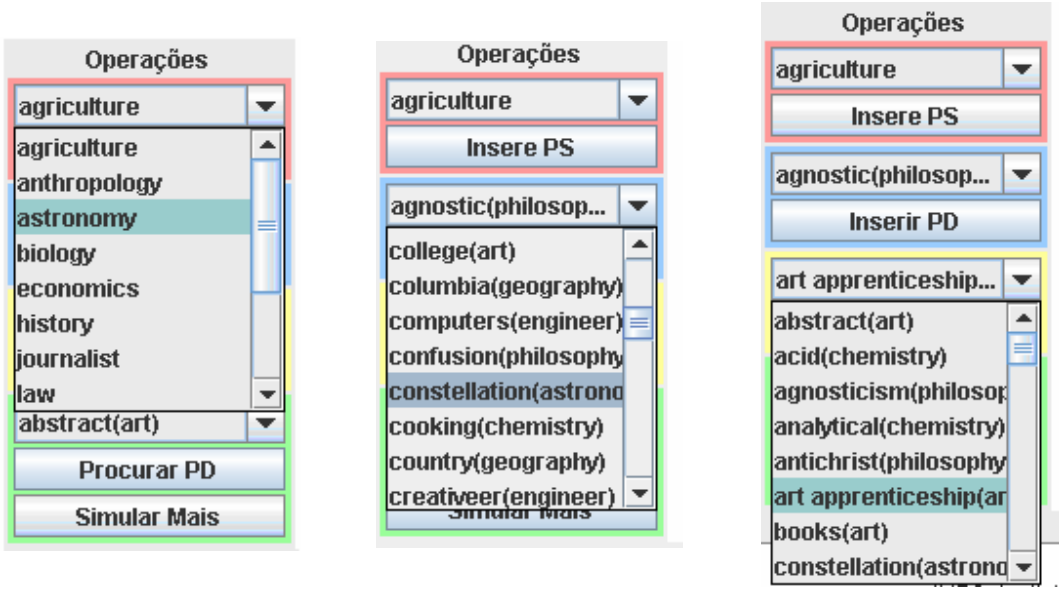


Figura 17: Palavras-chave ocultas e ligações da tabela de roteamento visíveis.

A inserção do *peer* de dados é efetuada pela escolha do item "constellation" na Figura 18b e do clique no botão "Inserir PD". O resultado é exibido no grafo que representa a topologia da rede na Figura 19 e mostra um novo componente "constellation" anexado ao *peer* semântico "astronomy".

A remoção de um *peer* de dados é possibilitada pela escolha de "art apprenticeship" no elemento da Figura 18c e pelo clique no botão "Remover PD". A remoção é visualizada na Figura 19, pela ausência do componente "art apprenticeship" no *peer* semântico "art".



(a) Seleção do *peer* semântico a ser inserido.

(b) Escolha do *peer* de dados a ser inserido.

(c) Seleção do *peer* de dados a ser removido.

(d) Escolha do *peer* de dados a ser encontrado.

(e) Seleção do *peer* de dados a ser localizado.

Figura 18: Operações do Painel de Controle.

Duas buscas por *peers* de dados na rede são realizadas. A primeira acontece com a seleção de "acid" na Figura 18d e com um clique em "Procurar PD". Como a mensagem com a busca deve ter encaminhada na rede, há a

necessidade de apertar no botão “Simular Mais” para que a operação seja concluída.

O resultado da primeira operação de busca foi o seguinte:

```
LOOKUP Of Target Palavra-Chave 'acid' PS 'chemistry' at
'0.5083190194166411'
      INFO: 'acid' Binded to chemistry '0.5083190194166411'....OK
```

Este resultado informa que a mensagem contendo a busca pelo *peer* de dados representado pela palavra-chave “acid” foi encaminhada pela rede até o *peer* semântico “chemistry” e foi encontrada.

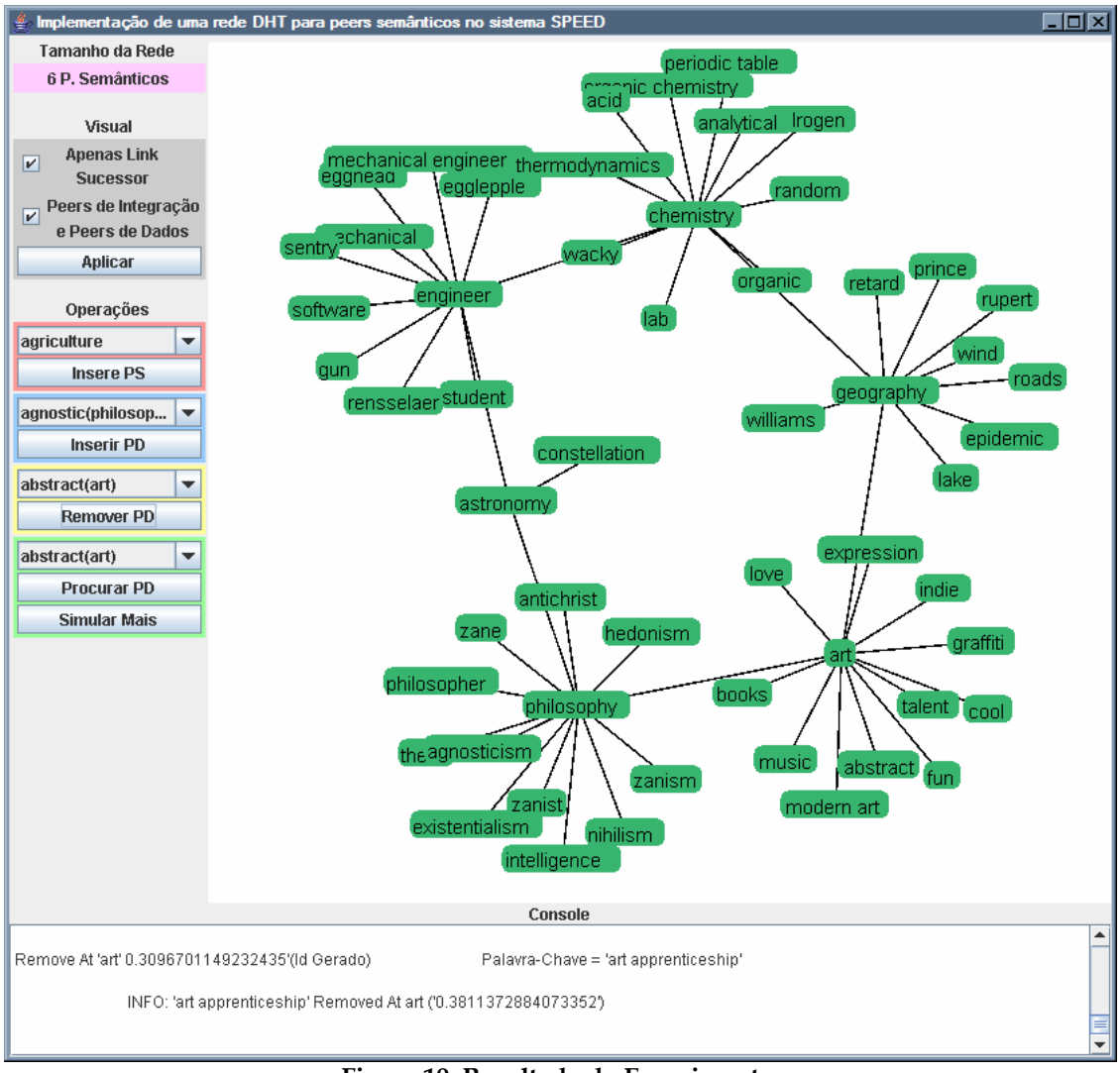


Figura 19: Resultado do Experimento.

A segunda busca é pelo *peer* de dados "atom". O item presente na Figura 18e é selecionado e os botões "Procurar PD" e "Simular Mais" são pressionados. As informações relativas a esta busca são as seguintes:

```
LOOKUP Of Target Palavra-Chave 'atom' PS 'chemistry' at
'0.5083190194166411'
      INFO: '[chemistry]' 'atom' Not Found     ....FAIL
```

As informações mostram que a mensagem com a busca pela *peer* de dados "atom" foi redirecionada até o *peer* semântico "chemistry". Ao chegar ao *peer* semântico, o *peer* de dados não foi encontrado e a operação de busca falhou.

Após a execução de todas as operações presentes no painel de controle, a topologia da rede pode ser visualizada na Figura 19.

6.3 Considerações

Neste capítulo vimos os detalhes de cada etapa da execução do experimento e o funcionamento de todas as operações do painel de controle. As principais fases foram: o carregamento da massa de dados, a criação da rede de *peers* semânticos, a inclusão de *peers* representados por palavras-chave, a busca por *peers* inseridos e a geração de uma representação gráfica da topologia da rede. Já as opções do painel de controle utilizadas foram: a opção de exibição dos *peers* na topologia, a opção de exibição das ligações da tabela de roteamento e dos predecessores, o componente para a inserção de *peers* semânticos, o componente para a inserção de *peers* de dados, o componente para e remoção de *peers* de dados e o componente para a busca por *peers* de dados.

7. Conclusões

No presente trabalho, observamos as principais características de *peer-to-peer*, *overlay*, DHT e Symphony, que fundamentaram a construção da simulação de uma rede DHT, com características semânticas em seus *peer*s, que será utilizada no protótipo do sistema SPEED.

Os PDMS e simuladores de redes *peer-to-peer* foram abordados para fundamentar a escolha do PlanetSim, o simulador que mais se adequou aos conceitos do SPEED. A simulação realizada no PlanetSim foi criada a partir de conceitos existentes em outros exemplos de simulações, já que, não existia uma simulação adequada à arquitetura SPEED.

A função de classificação implementada no módulo semântico foi simples, mas pode cumprir com o seu papel de classificar *peers* em domínios representados por *peers* semânticos. Em implementações com massas de dados maiores e com um significado semântico maior, a função poderá ser substituída por funções baseadas de ontologias e em comparação de esquemas.

A interface gráfica foi concebida para possibilitar uma melhor compreensão do resultado da simulação, permitindo assim, que o usuário navegue entre o resultado e personalize a exibição de informações.

Este trabalho foi implementado de uma forma modularizada e extensível, pois o mesmo foi concebido para integrar-se com o restante da topologia do SPEED. Assim, o restante da arquitetura deverá obedecer ao padrão de troca de mensagens e ao de funcionamento do simulador baseado em passos.

7.1 Contribuições

Este trabalho foi o pioneiro na implementação do sistema SPEED, logo, várias indefinições precisaram de fundamentos para que pudessem ser esclarecidas e implementadas. Essas indefinições foram solucionadas em etapas.

A primeira etapa foi a análise do SPEED para possibilitar a pesquisa e escolha de um simulador de redes *peer-to-peer* que fosse adequado ao sistema. O simulador era capaz de simular o SPEED, porém, era necessário criar uma simulação no PlanetSim.

A segunda fase foi a implementação de uma simulação com a reutilização de partes de outros exemplos de simulações já existentes, já que os exemplos do simulador não tinham o completo perfil necessário ao SPEED. Ainda foi necessário pesquisar e criar de uma função de classificação provisória à arquitetura SPEED, pois não havia nenhuma definição de implementação quanto ao método de classificação dos dados em *peers* semânticos.

Por fim, as últimas etapas foram: modularizar todo o projeto, para facilitar futuras implementações; e criar uma interface gráfica, que pudesse exibir e manipular o resultado da simulação.

7.2 Dificuldades Encontradas

Dificuldades surgiram devido à grande quantidade de simuladores. Após uma longa seleção fizemos a escolha do PlanetSim [García et al. 2005], pois o mesmo se adequava melhor à implementação de uma rede DHT. Na implementação da rede *super-peer* [Mendes 2007], o simulador que melhor atendia as suas necessidades foi o PeerSim. Logo, chegamos a um consenso de desenvolver e documentar o funcionamento das duas redes em diferentes simuladores para obter um maior conhecimento do domínio.

Houve ainda problemas quanto à corretude do algoritmo de geração de *overlay* Chord implementado pelo PlanetSim. Nesta implementação, em algumas pesquisas/buscas, não se conseguia atingir um *peer* que estava presente na rede. A solução foi utilizar o algoritmo Symphony.

Mesmo utilizando Java, padrões de desenvolvimento e boa documentação, o entendimento do simulador PlanetSim foi conseguido após bastante

perseverança. Para melhor dimensionarmos este simulador, o mesmo possui aproximadamente 200 (duzentas) classes.

Uma dificuldade ocorreu devido à simulação que trabalha com números aleatórios relativos à geração dos IDs de cada *peer* em uma rede, logo, não foi possível estabelecer um exemplo que produzisse cenários semelhantes entre as execuções.

7.3 Trabalhos Futuros

O desenvolvimento de uma rede DHT para *peers* semânticos no PlanetSim foi apenas uma das partes necessárias para o completo funcionamento do SPEED. Paralelamente a esta implementação, a topologia *super-peer* vem sendo desenvolvida [Mendes 2007] no simulador PeerSim [PeerSim 2005]. Na próxima fase de implementação do SPEED, uma das duas implementações deverá migrar para o simulador da outra rede. Assim, com a documentação provida por estes trabalhos, a migração será realizada com menos dificuldades. Outra parte a ser aperfeiçoada é a questão semântica no sistema. A solução atual é simples e deve ser estendida, utilizando comparação de esquemas e ontologias [Souza 2007].

A interface gráfica do sistema é um ponto a ser aprimorado, a atual é simples. Com uma interface gráfica mais interativa, o funcionamento do sistema passa a ser mais transparente para o usuário. Novas interações podem ser criadas para aumentar o detalhamento da rede em resposta a alguma ação do usuário.

Referências

[Amir 2003] Amir,Y., Danilov,C., 2003. Reliable Communication in Overlay Networks. In Proc. Int. Conf. on Dependable Systems and Networks (DSN'2003), pp. 511–520, San Francisco.

[Andersen et al. 2001] Andersen,D.G., Balakrishnan,H., Kaashoek,F., Morris,R., 2001. "Resilient Overlay Networks". In Proc. ACM Symp. on Operating Systems Principles, pp. 131–145, Banff, Canada.

[AIM 2007] AOL Instant Messenger. http://www.aim.com/, último acesso em: 02/05/2007.

[BitTorrent 2007] BitTorrent. http://www.bittorrent.com/, último acesso em: 02/05/2007.

[Braunisch 2006] Braunisch,M.H., 2006. Chord and Symphony: An Examination of Distributed Hash Tables and Extension of PlanetSim. A Thesis in the Field of Information Technology for the Degree of Master of Liberal Arts in Extension Studies, Harvard University, June 2006.

[Brown 2006] Brown,A., Kolberg,M., 2006. Tools for *Peer-to-Peer* Network Simulation. University of Stirling, UK.

[CacheLogic 2005] CacheLogic Research "P2P in 2005," 2005. http://www.cachelogic.com/home/pages/studies/2005_01.php, último acesso em 24/05/2007.

[Day et al. 2000] Day,D., Aggarwal,S., Mohr,G., Vicent,J., 2000. Instant Messaging / Presence Protocol Requirements, RFC 2779, IETF.

[Demaine 2001] Demaine,E.D., Leiserson,C.E., 2001. Introduction to Algorithms.

[Eclipse 2005] Eclipse 3.1.2, 2005. http://www.eclipse.org/, último acesso em 30/05/2007.

[Florescu et al. 1998] Florescu,D., Levy,A., Mendelzon,A. 1998. Database techniques for the world wide web: a survey, ACM SIGMOD Record, v. 27, n. 3, p. 59-74.

[FreeNet 2007] FreeNet. http://freenetproject.org/, último acesso em 30/05/2007.

[FreePastry 2007] FreePastry, 2007. Pastry - a substrate for *peer-to-peer* applications. http://freepastry.org/, último acesso em 04/06/2007.

[García et al. 2005] García,P., Pairot,C., Mondéjar,R., Pujol,J., Tejedor,H., Rallo,R., 2005. PlanetSim: A New Overlay Network Simulation Framework. T. Gschwind and C. Mascolo (Eds.): SEM 2004, LNCS 3437, pp. 123–136.

[Gnutella 2007] Gnutella, 2007. http://www.gnutella.com/, último acesso em 30/05/2007.

[GTNetS 2002] The Georgia Tech Network Simulator (GTNetS), 2002. http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/, último acesso em 30/05/2007.

[Hellerstein 2004] Hellerstein,J., 2004. Architectures and Algorithms for Internet-Scale (P2P) Data Management. EECS Computer Science Division, UC Berkeley Intel Research, Berkeley.

[Hose et al. 2006] Hose,K., Job,A., Karnstedt,M., Sattler,K., 2006. An Extensible, Distributed Simulation Environment for *Peer* Data Management Systems. Department of Computer Science and Automation, Ilmenau, Germany.

[Hull 1997] Hull,R., 1997. Managing semantic heterogeneity in databases: a threorical perspective, In: ACM Symposium on Principles os Database Systems, Procedings..., p. 51-61.

[Java 2007] Java, 2007. http://www.java.com/, último acesso em 30/05/2007.

[Jones 2001] Jones, P., 2001. US Secure Hash Algorithm 1 (SHA1), RFC 3174. http://www.faqs.org/rfcs/rfc3174.html/, último acesso em 30/05/2007.

[Kazaa 2007] KaZaA. http://www.kazaa.com/, último acesso em 24/05/2007.

[Kaashoek 2003] Kaashoek,M.F., Karger,D.R., 2003. Koorde: A simple degree-optimal distributed hash table. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 2735/2003, Pages 98-107.

[Lyman et al. 2003] Lyman,P., Varian,H.R., Dunn,J., Strygin,A., Searingen,K., 2003. How much information. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/, último acesso em 30/05/2007.

[Malkhi et al. 2002] Malkhi,D., Naor,M., Ratajczak,D., 2002. "Viceroy: A scalable and dynamic emulation of the Butterfly". In Proceeding of the 21st ACM Symposium on Principles of Distributed Computing (PODC'02).

[Manku et al. 2003] Manku, G.S., Bawa,M., Raghavan,P., 2003. Symphony: Distributed hashing in a small world. In Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003), pages 127–140.

[Maymounkov 2002] Maymounkov,P., Mazières,D., 2002. Kademlia: A *Peer-to-peer* Information System Based on the XOR Metric. Electronic Proceedings for the 1st International Workshop on *Peer-to-Peer* Systems, MIT Faculty Club, Cambridge, MA, USA.

[Mendes 2007] Mendes,D.S.R., 2007. Implementação de uma Rede Super-Ponto para *Peer*s de Integração no Sistema SPEED. Monografia e Proposta de Trabalho de Graduação. CIn, UFPE, Brasil.

[Morpheus 2007] Morpheus. http://www.morpheus.com/, último acesso em 24/05/2007.

[Naicken et al. 2007] Naicken,S., Livingston,B., Basu,A., Rodhetbhai,S., Wakeman,I., Chalmers,D., 2007. The State of *Peer*-to-*Peer* Simulators and Simulations. ACM SIGCOMM Computer Communication Review 95 Volume 37, Number 2.

[Napster 2007] Napster, 2007. http://www.napster.com/, último acesso em 02/05/2007.

[Narses 2003] Narses Network Simulator, 2003. http://sourceforge.net/projects/narses/, último acesso em 30/05/2007.

[NATO Phonetic Alphabet 2007] NATO Phonetic Alphabet, 2007. http://en.wikipedia.org/wiki/NATO_phonetic_alphabet/, último acesso em 25/06/2007.

[Neto 2004] Neto,D.G.O., 2004. PlanetLab: Serviços de Rede em Escala Planetária – WNRP 2004. Departamento de Ciência da Computação, UFMG.

[Neurogrid 2001] NeuroGrid Network Simulator, 2001. http://www.neurogrid.net/, último acesso em 30/05/2007.

[Nica 1999] Nica,A., Rundensteiner,E.A., 1999. View maintenance after view synchronization, In International Database Engineering and Application Symposium (IDEAS'99), 1999. Procedings..., p. 213-215.

[NS-2 2007] The Network Simulator (NS-2), 2007. http://sourceforge.net/projects/nsnam/, último acesso em 30/05/2007.

[Obelheiro 2003] Obelheiro,R.R., Fraga,J.S., 2003. Uma Rede Overlay Tolerante a Intrusões: Arquitetura e Análise. Departamento de Automação e Sistemas Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil.

[OMNet++ 2006] OMNeT++ is a discrete event simulation environment, 2006. http://www.omnetpp.org/, último acesso em 30/05/2007.

[P2PSim 2005] P2Psim: A Simulator for *Peer-to-Peer* (P2P) Protocols, 2005. http://pdos.csail.mit.edu/p2psim/, último acesso em 30/05/2007.

[PeerSim 2005] PeerSim P2P Simulator, 2005. http://peersim.sourceforge.net/, último acesso em 30/05/2007.

[Pires et al. 2006] Pires,C.E.S., Lóscio,B.F., Salgado,A.C., 2006. Data Management in P2P Systems. In Proc. of the 21º Simpósio Brasileiro de Banco de Dados, Florianópolis, SC, Brazil.

[Pires 2007] Pires,C.E.S., 2007. Um Sistema P2P de Gerenciamento de Dados com Conectividade Baseada em Semântica. Monografia de Qualificação e Proposta de Doutorado. CIn, UFPE, Brasil.

[Prefuse 2007] The prefuse visualization toolkit, 2007. http://prefuse.org/, último acesso em 22/06/2007.

[Pressman 2006] Pressman, R.S., 2006. Software Engineering: A Practitioner's Approach, 6/edition.

[Ratnasam et al. 2001] Ratnasamy,S., Francis,P., Handley,M., Karp,R., Shenker,S., 2001. A Scalable Content-Addressable Network. SIGCOMM'01, San Diego, California, USA.

[Risson 2004] Risson,J., Moors,T., 2004. Survey of research towards robust *peer-to-peer* networks: search methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, Sydney, Austrália.

[Rivest 1992] Rivest,R., 1992. The MD5 Message Digest Algorithm, RFC1321. ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt/, último acesso em 30/05/2007.

[Rocha 2003] Rocha,R.R., 2003. Redes *Peer-to-Peer* para Compartilhamento de Arquivos na Internet. Grupo de Teleinformática e Automação PEE/COPPE - DEL/POLI, Universidade Federal do Rio de Janeiro.

[Rose 2007] IBM Rational Rose Enterprise. http://www-306.ibm.com/software/awdtools/developer/rose/enterprise/index.html/ , último acesso em 30/05/2007.

[Rowstron et al. 2001] Rowstron,A., Druschel,P., 2001. Pastry: Scalable, decentralized object location and routing for large-scale *peer-to-peer* systems. 18th IFIP/ACM International Conference on Distributed Systems Platforms.

[RUP 2007] Rational Unified Process (RUP). http://www-306.ibm.com/software/awdtools/rup/, último acesso em 30/05/2007.

[SETI@Home 2002] SETI@Home, 2002. http://seticlassic.ssl.berkeley.edu/totals.html/, último acesso em 24/05/2007.

[SFFNet 2004] SSF (Scalable Simulation Framework), 2004. http://www.ssfnet.org/, último acesso em 30/05/2007.

[Souza 2007] Souza,D.Y., 2007. Reformulação de Consulta Baseada em Semântica para PDMS. Monografia de Qualificação e Proposta de Doutorado. CIn, UFPE.

[Stoica et al. 2001] Stoica,I., Morris,R., Karger,D., Kaashoek,M.F., Balakrishnan,H., 200X. Chord: A Scalable *Peer-to-peer* Lookup Service for Internet Applications. MIT Laboratory for Computer Science.

[Sung et al. 2005] Sung,L.G.A., Ahmed,N., Blanco,R., Li,H., Soliman,M.A., Hadaller,D., 2005. A Survey of Data Management in *Peer-to-Peer* Systems. School of Computer Science, University of Waterloo.

[Ting 2003] Ting,N.S., Deters,R., 2003. 3LS - A *Peer-to-Peer* Network Simulator. Department of Computer Science, University of Saskatchewan, Canada.

[UML 2007] Unified Modeling Language 2.0 (UML). http://www.omg.org/technology/documents/formal/uml.htm/, último acesso em 30/05/2007.

[UD 2007] Urban Dictionary, 2007. http://www.urbandictionary.com/, último acesso em 12/07/2007.

[Xie 2003] Xie,M., 2003. P2P Systems Base don Distributed Hash Table. Computer Science, University of Ottawa.

[Zhao et al. 2004] Zhao,B.Y., Huang,L., Stribling,J., Rhea,S.C., Joseph,A.D., Kubiatowicz,J.D., 2004. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on selected areas in communications, Vol. 22, No. 1.

Assinaturas

Ana Carolina Salgado

Orientadora

Guilherme Barros de Souza

Aluno