Sistemas P2P e PDMSs

Trabalho Individual I

CIn-UFPE

Edemberg Rocha da Silva
Agosto de 2010

Sistemas P2P e Sistemas de Gerenciamento de Dados em Ambientes P2P

1. Introdução

Como um meio de prover uma infra-estrutura para um compartilhamento de recurso na *web*, surgem os sistemas P2P. Esses sistemas partem do princípio que um controle centralizado inexiste e que em cada *peer*, os softwares uma vez executados detêm características funcionais equivalentes e os *peers* atuam como cliente e/ou servidor. A motivação inicial desses tipos de sistemas foi a possibilidade de compartilhar arquivos. Facilmente um usuário podia fazer a busca por documento hipermídia, através do uso de palavras-chave, sem a necessidade de existir descrições semânticas para que esses documentos fossem encontrados.

Embora tenham sido concebidos para o compartilhamento de dados não estruturados, nos últimos anos os sistemas P2P têm sido direcionados a ambientes mais sofisticados (HOSE, 2006), dando suporte ao compartilhamento de dados estruturados. É nesse cenário que surgem os Sistemas de Gerenciadores de Dados em P2P (*Peer Data Management Systems - PDMS*). Os PDMSs visam unir o poder semântico existente nos bancos de dados aliado aos benefícios providos pelos sistemas P2P. Com isso, os PDMSs permitem realizar tarefas em ambientes P2P que vão desde o armazenamento, indexação e busca, a questões relacionadas à segurança e privacidade (OTTO, 2007).

Este trabalho objetiva apresentar a natureza dos sistemas P2P no tocante às suas características, arquiteturas, tipos de buscas que realizam e topologias existentes. E através desses parâmetros, os PDMSs serão abordados, exemplificados e comparados mediante suas características gerais.

2. Sistemas Peer-to-Peer

Inúmeras definições existem sobre sistemas P2P, mas todas elas buscando uma condensação de sua definição. Nowell (NOWELL et. al, 2002) define sistemas P2P da seguinte forma:

> *"Peer-to-peer (P2P) systems are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality."*

No foco deste trabalho, os sistemas P2P são vistos também como uma rede de nós interconectados que cooperam entre si, trocando informações e serviços sem a intervenção de um servidor. Vários são os sistemas P2P implementados para os mais diversos fins, como podemos citar: computação distribuída, troca de mensagens, trabalho colaborativo e compartilhamento de dados. Em especial, sistemas que permitem a troca de arquivos tornaram as aplicações P2P bastante populares e difundidas entre usuários. São exemplos desses sistemas o Kazza(KAZAA, 2010),Napster (NAPSTER, 2010), GNutella(GNUTELLA22, 2010), E-mule (E-MULE, 2010) dentre outros.

ANDROUTSELLIS-THEOTOKIS et al. (2004) apontam as características básicas de um sistema P2P como sendo:

- *Peers* se conectam diretamente com outros *peers*;
- *Peers* são responsáveis pelos seus próprios dados;
- *Peers* podem entrar e sair da rede a qualquer momento;
- *Peers* podem atuar tanto quanto clientes quanto como servidores;
- *Peers* são autônomos com relação ao controle e estruturação da rede, ou seja, não existe autoridade central.

Se comparados à arquitetura cliente-servidor, onde o número de servidores é fixo, sistemas P2P são mais flexíveis e extensíveis, visto que quando novos *peers* entram na rede, a capacidade total do sistema aumenta, enquanto que na arquitetura cliente servidor, a adição de novos clientes reduz o desempenho do mesmo (ZHAO, 2006). Devido a essas vantagens presentes em sistemas P2P, isso faz com que eles tenham robustez em falhas, extensivo compartilhamento de recursos, auto-organização, balanceamento de carga, persistência de dados, entre outras características.

2.1.Arquiteturas

Sistemas P2P têm suas arquiteturas diferenciadas sobre diferentes fatores. Nesta seção abordaremos as arquiteturas existentes para sistemas P2P, considerando esses fatores.

a. Quanto à forma de compartilhamento dos índices dos objetos

De acordo com (SUNG, 2005) são diversas as categorias de arquiteturas de sistemas P2P. E considerando a forma como os índices dos objetos compartilhados são armazenados, temos as seguintes classificações:

1. Centralizados

Nestes sistemas existe um índice central, conforme Figura 1, e com informações que ficam sendo constantemente atualizadas. Ao entrar na rede, o usuário se conecta a um ou mais servidores e estes atuam como um índice, para busca de dados disponíveis. Ao se conectar, o usuário informa ao servidor todos os dados que ele está disponibilizando na rede. Neste momento o servidor atualiza seus índices para disponibilizar esses dados para outros possíveis usuários.

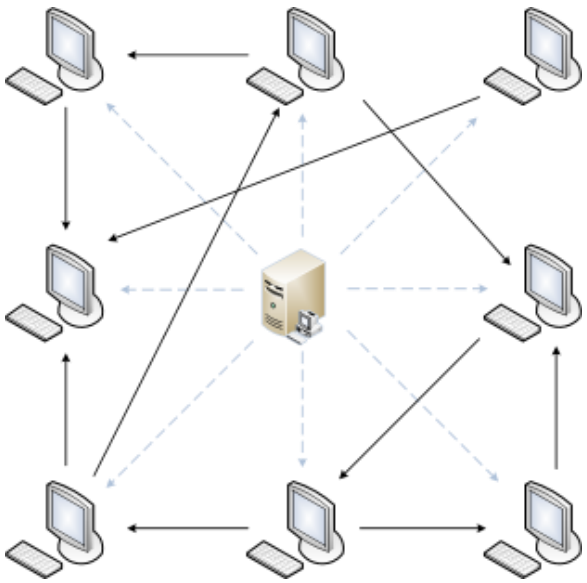

Figura 1 - Sistema P2P Centralizado

Quando um usuário realiza a busca por um dado elemento, ele primeiro conecta-se a um servidor e submete sua busca a ele. Após realizar a busca, o servidor retorna ao usuário os IPs dos *peers* que possuem os elementos procurados pelo usuário. Tanto o

sistema de compartilhamento de arquivos do NAPSTER (2010) quanto do SOULSEEK (2010) fazem uso desse tipo de arquitetura.

2. Descentralizados

Nesta arquitetura inexiste o papel de um servidor que contém um índice central. Sua arquitetura pode ser visualizada conforme ilustra a Figura 2.

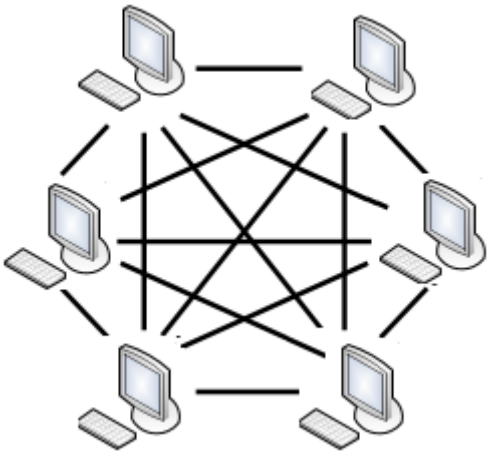

Figura 2 - Sistema P2P Descentralizado

Sistemas P2P descentralizados subdividem-se em não estruturados e estruturados. Um não estruturado não possui um controle preciso sobre a topologia, localização e busca de documentos. O KAZAA (2010) e o GNUTELLA2 (2010) são exemplos de sistemas P2P descentralizados e não estruturados. Em contrapartida, um sistema P2P descentralizado e estruturado possui uma significativa estruturação entre os nós. Nessa, os documentos são postos em locais e, posteriormente, torna-se fácil sua localização e existe um controle da topologia da rede. Esse tipo de arquitetura utiliza busca baseada em DHT (*distributed* hash table). Uma DHT fornece um alicerce para a construção de sistemas distribuídos descentralizados, provendo um serviço de *lookup* similar a uma tabela hash (chave e valor), sem estabelecer uma hierarquia fixa (GHODSI, 2006). Cabem aos peers a responsabilidade de manter o mapeamento de chaves para valores. Os sistemas BitTorrent (SUNG, 2005), Chord (STOICA, 2001), Pastry (DRUSCHEL, 2001) e CAN (RATNASAMY, 2001) fazem uso de uma arquitetura descentralizada e estruturada.

3. Modelo Hierárquico

O modelo hierárquico faz uso de nós com características especiais, os super nós, para a execução de tarefas como a manutenção de índices, como ocorre no Kazaa (KAZAA, 2010). Este modelo pode ser visto como um intermediário entre os dois modelos anteriores (REZENDE, 2009).

b. Quanto ao tipo de busca

Quanto ao mecanismo de busca em sistemas P2P, (TOWSLEY, 2003) divide-os em três categorias:

- Serviço de localização centralizada (*Centralized Service Location* - CSL): neste mecanismo, existe um *peer* central e a ele são submetidas consultas provenientes dos nós existentes na rede. É através do nó central onde as trocas de informações são realizadas entre os *peers*. Podemos citar como exemplo de sistema P2P que realiza uma busca centralizada, o NAPSTER (2010).
- Serviço de localização por inundação (*Flooding-based Service Location* - FSL): neste mecanismo, não existe uma estruturação na rede, tão pouco um *peer* central como no mecanismo anterior. Ao entrar na rede, um *peer* entra e conecta-se aos demais *peers* existentes na rede e as buscas são realizadas através de mecanismos de inundação[1]. O GNUTELLA2 (2010) é um exemplo de sistema P2P que realiza busca por inundação.
- Serviço de localização baseado em DHT (*DHT Service Location*): rede onde os nós têm autonomia e usam um mecanismo de busca semelhante ao de uma tabela *hash*, porém mantém partes dessa tabela em diversos nós que compõem a rede (REZENDE, 2009). Os sistemas P2P Chord (STOICA, 2001), CAN (RATNASAMY, 2001), Tapestry (ZHAO, 2001) e Pastry (DRUSCHEL, 2001) realizam buscas por DHT.

Além dos três mecanismos citados anteriormente, (COSTA, 2009) classifica mais dois mecanismos:

- Rede Semântica Overlay (*Semantic Overlay Network* - SON): existe uma rede virtual onde encontram-se agrupados *peers* de acordo com ligações semânticas. *Peers* unem-se formando uma sub-rede com valor semântico associado, podendo cada sub-rede se sobrepor. A Figura 3 ilustra uma divisão semântica baseada em categorias de filmes que cada *peer* possui. E, no caso de um *peer* possuir filmes de mais de uma categoria, ele entra em diversas sub-redes semânticas das quais ele

[1] O termo inundação refere-se ao número de mensagens que são enviadas a rede até que o nó procurado seja encontrado (REZENDE, 2009).

faz parte da categoria. O *peer* X possui filmes tanto da categoria “ação” quanto “suspense”.

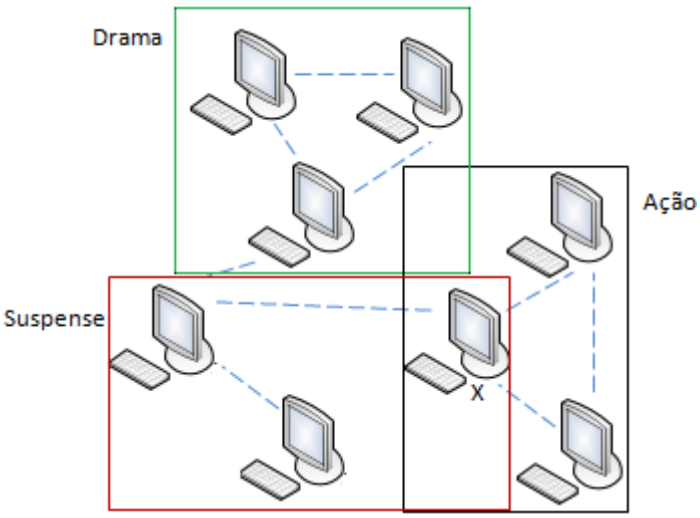

Figura 3 - Busca baseada em SON

As consultas realizadas são enviadas somente aos grupos semânticos relacionados, ignorando aqueles que não possuem qualquer relação à semântica estabelecida na consulta (COSTA, 2009).

- Colônia de formigas (*Ant Colony Optimization System*- ACO): surgiu com o intuito de melhorar o desempenho de busca por inundação (COSTA, 2009). Algumas pesquisas (MICHLMAYR, 2006) (CIGLARIC & VIDMAR, 2006) têm indicado o uso do algoritmo de colônia de formigas para a otimização dos caminhos que devem receber uma consulta. O algoritmo ACO é baseado no comportamento de formigas e de suas colônias. Alguns experimentos mostram que as formigas possuem um mecanismo de comunicação indireta que permite traçar uma melhor rota para chegar até o seu alimento, deixando marcas no caminho através do feromônio liberado por elas. Utilizando várias trilhas, as formigas detectam o melhor caminho pela maior intensidade de feromônio detectado por elas (COSTA, 2009).

  O problema do caixeiro viajante (DORIGO & GAMBARDELLA, 1997) e a otimização de algoritmos para alinhamento múltiplo de seqüências em bioinformática (ZAFALON, 2009) são exemplos de problemas que podem ser resolvidos através de colônia de formigas.

  Devido à absorção de mudanças dinâmicas em um grafo, o ACO permite ser utilizado no roteamento em redes de computadores dinâmicas, como no caso das redes P2P, fato este que pôde ser observado em (CIGLARIC & VIDMAR, 2006). No caso de redes que fazem o uso de técnicas de inundação para o roteamento, o

ACO pode predizer caminhos que possuem a melhor probabilidade de encontrar melhores resultados (COSTA, 2009). O AntNet (CARO & DORIGO, 1998) utilizou colônia de formigas para a geração de tabelas de roteamento em redes P2P. Através do uso de palavras-chave definidas em cada *peer*, foram criados vários tipos de feromônio, que resultam em diferentes caminhos para cada palavra-chave fornecida na busca desejada (COSTA, 2009).

c. Outras classificações

No tocante a características relacionadas à confiabilidade, escalabilidade, desempenho, dentre outros, Fiorano (2003) compara e avalia diferentes topologias, objetivando determinar a arquitetura mais adequada para problemas que necessitam de computação distribuída. No trabalho (FIORANO, 2003), ele apresenta as seguintes categorias para sistemas P2P:

- Pura

Modelo descentralizado onde não existe um ponto central de controle. Os *peers* são auto-suficientes, atuam tanto como cliente quanto como servidor e estão conectados, podendo comunicar-se uns aos outros. Um exemplo de funcionamento pode ser visualizado através da Figura 4, onde o *peer* X se interliga aos outros *peers* os quais possuem outras conexões. Por sua vez, estes se conectam a outros *peers* com os quais mantém conexão e repetem a solicitação inicial. Tal procedimento é realizado repetidamente até que o recurso procurado seja encontrado, no caso, em "Y" (REZENDE, 2009). Quando o recurso é encontrado, uma conexão é criada entre o nó onde o recurso foi encontrado e o nó que fez a requisição, executando-se assim a transferência do recurso solicitado sem a existência de nenhum intermediário (LAFFRANCHI, 2004).

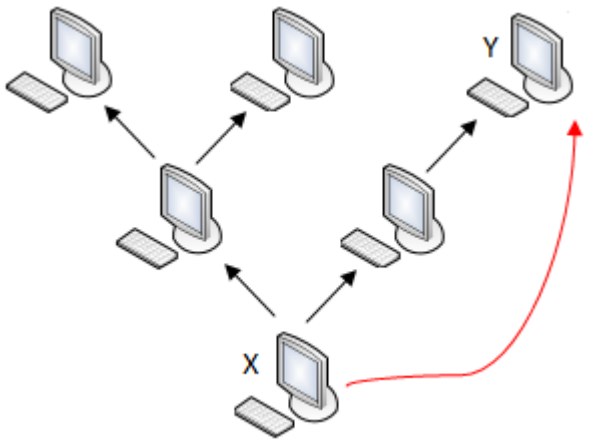

Figura 4 - Arquitetura pura de um sistema P2P

Essa arquitetura possui como característica a escalabilidade, uma vez que qualquer *peer* pode entrar e trocar informações com outros *peers* na rede (FIORANO, 2003). E, considerando que a queda em um nó não afeta o restante do sistema, podemos considerar também que existe tolerância a falhas na arquitetura pura. Um dos principais sistemas que faz uso desta arquitetura é o sistema GNUTELLA2 (GNUTELLA2, 2010).

- Híbrida

O modelo descentralizado é raramente utilizado devido a sua complexidade (REZENDE, 2009). Em uma arquitetura híbrida, os dados trafegam entre os nós como na arquitetura pura, porém a informação de controle é tratada por um servidor central, que realiza o monitoramento para os *peers* e garante a coerência das informações. A Figura 5 ilustra uma situação onde o *peer* "X" entra em contato com o servidor para localizar um determinado recurso. Em seguida, o servidor coleta as informações necessárias para a conexão entre "X" e o *peer* que possui o recurso desejado, no caso o *peer* "Y". Uma vez que "X" e "Y" estabelecem uma conexão, a transação entre os *peers* será executada.

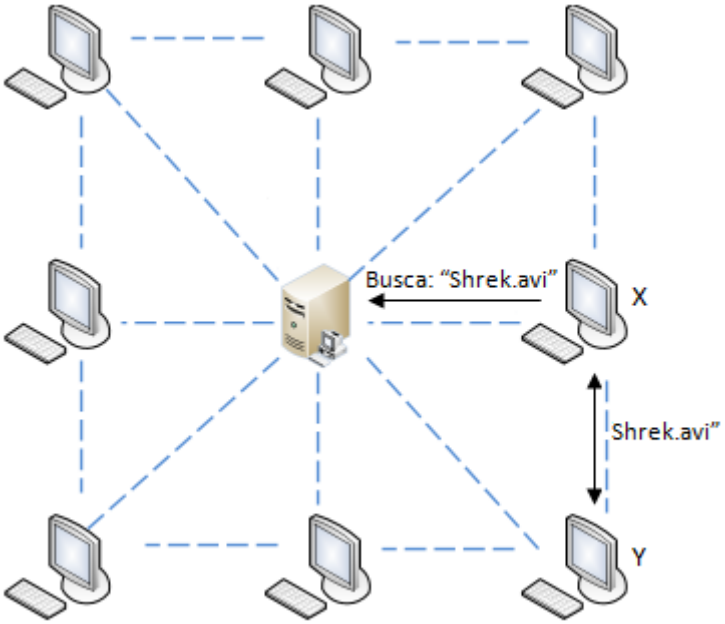

Figura 5 - Arquitetura híbrida

Sistemas híbridos têm sido utilizados principalmente em aplicações críticas, geralmente limitadas a um pequeno conjunto de problemas (FIORANO, 2003). O Groove (GROOVE, 2010) é um sistema de gerenciamento de projetos colaborativos, que faz uso da arquitetura híbrida.

- Super-*peer*

Nesta arquitetura existe um *peer* global que gerencia outros *peers* que estão conectados a ele. Esse *peer* global é chamado de super-*peer*. A idéia é de ter sub-redes P2P, cada uma com um super-*peer*, e para que haja uma comunicação entre um *peer* de uma sub-rede com outro *peer* de outra sub-rede, os super *peers* de ambas devem estar conectados. A Figura 6 ilustra uma arquitetura de super peers, onde estão conectados três super-*peers*, A, B e C.

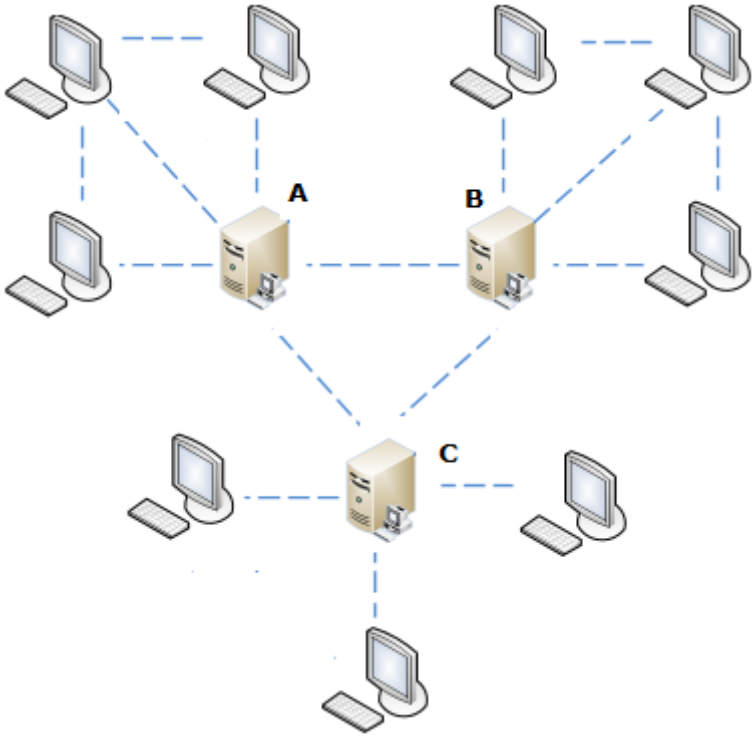

Figura 6 - Arquitetura de super-*peer*

Nos sistemas que implementam uma arquitetura de super *peers*, existe uma agilidade quanto à busca da informação desejada, se comparada às outras arquiteturas. Tal agilidade deve-se ao fato que os sistemas são divididos em agrupamentos (clusters) onde cada um destes possui informações indexadas sobre seus *peers*. De acordo com (FIORANO, 2003), uma busca que leva O(n) em uma rede pura ou híbrida, levará O(n/m) (sendo m o número médio de *peers* conectados a um super-*peer*) em uma arquitetura de super-*peers*. O sistema Kazaa (KAZAA, 2010) é um exemplo que faz uso de uma arquitetura como esta.

Caso ocorra alguma falha em alguns dos super-*peers*, os clientes poderão sentir essa falha. E para resolver esse problema, redundância de super-*peer (super-peer redundancy)* pode ser utilizada. Nesta solução *peers* reservas são definidos para assumiram o papel de um super-*peer*, caso ocorra algum tipo de falha. Logo, uma arquitetura de super *peer* redundante que combina vantagens tanto de sistemas centralizados quanto de sistemas descentralizados é considerada a mais apropriada para sistemas que necessitam de computação distribuída (FIORANO, 2003).

2.2.Roteamento

Um dos principais objetivos de um sistema P2P é prover um mecanismo de busca de dados através de consultas utilizando-se palavras-chave (SUNG, 2005). E, a fim de identificar os *peers* que podem responder às consultas, um mecanismo de roteamento se faz necessário e sua implementação pode variar de acordo com a arquitetura utilizada, como visto na seção anterior. Algoritmos de buscas e roteamento procuram otimizar o encaminhamento de um *peer* a um outro, assim como as estratégias de roteamento dependerão da característica dos sistemas de serem estruturados ou não.

As estratégias de roteamento são baseadas nos seguintes modelos: centralizado, inundação e DHT.

- Modelo Centralizado

  A estratégia do modelo centralizado é manter um ponto central onde ficam armazenados nele o conteúdo que cada *peer* na rede tem a oferecer. Logo, quando uma consulta é submetida por um *peer* ao ponto central, este último escolhe o *peer* que for mais adequado a atender à solicitação da consulta, e em seguida a troca de dados é realizada entre os *peers*.

  No modelo centralizado, se um dado existir na rede, ele será encontrado e, como o mecanismo de busca por palavras-chave é fácil de implementar, isso traz vantagens ao modelo centralizado. Porém, esse modelo torna-se susceptível a falhas devido à concentração do conteúdo dos *peers* em um único ponto.

- Modelo de Inundação

  Neste modelo é realizada uma busca por inundação controlada por um tempo (*time to live* - TTL). Ao solicitar uma busca por um determinado dado, um *peer* manda uma mensagem a todos os *peers* conectados a ele. Se um dos *peers* vizinhos possuir o dado desejado, o IP deste é retornado ao solicitante. Caso contrário, os *peers* mandam mensagens aos seus *peers* vizinhos e o procedimento de busca vai se repetindo até que o dado seja encontrado ou o TTL seja alcançado.

  Devido à necessidade de um grande processamento e largura de banda, a escalabilidade do modelo de inundação fica comprometida. Além disso, pode haver casos onde o dado existe, porém não seja encontrado devido a sua

distância, ao *peer* solicitante da busca, ser grande e o tempo do TTL ser logo alcançado.

- Modelo DHT

Nesse modelo, um ID randômico é associado a cada *peer* que conhece um determinado número de *peers* como poder ser visto na Figura 7. Ao ser publicado um documento no sistema P2P, um valor de ID é gerado através de uma função *hash* tomando como entrada o conteúdo dos documentos e o seu nome. Em seguida, o *peer* repassa o documento ao *peer* cujo ID é o mais próximo do ID do documento. Esse processo é repetido até que o ID do nó atual seja o mais próximo do ID do documento (LOEST, 2007). E uma vez encontrado, o documento será transferido ao *peer* solicitante, enquanto que cada *peer* que participou do roteamento ficará com uma cópia local do documento. Com isso, podemos afirmar que cada operação de roteamento também garante que uma cópia do documento seja mantida (LOEST, 2007) (LUA et. al, 2004).

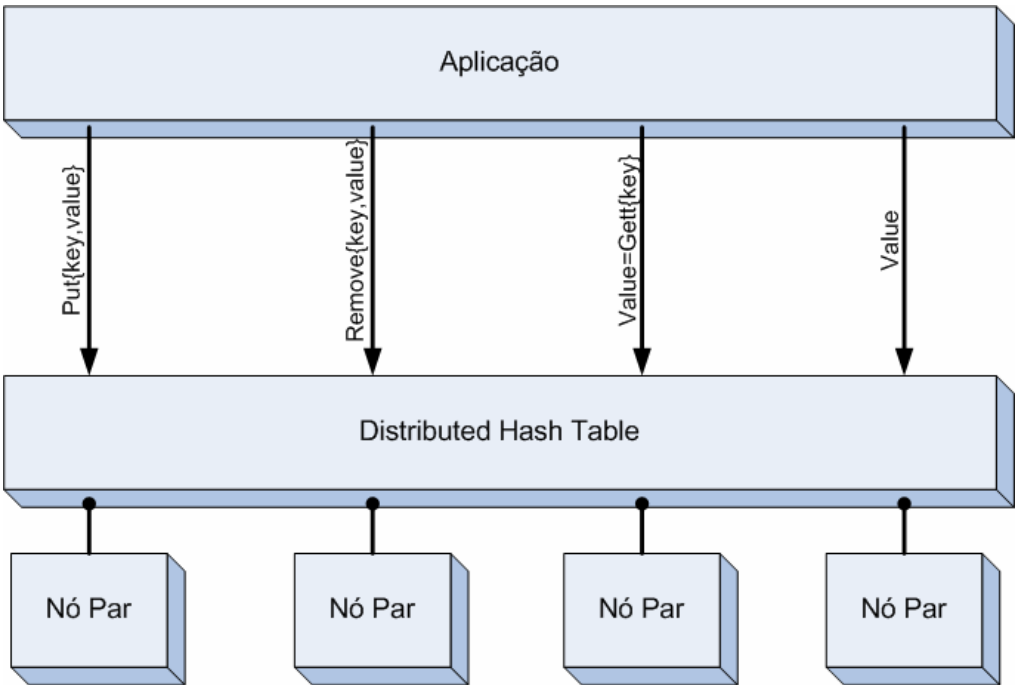

Figura 7 - Sistema DHT (LUA et. al, 2004)

Embora o modelo DHT seja considerado eficiente para comunidades grandes e globais, ele apresenta um problema relacionado ao ID do documento. Este precisa ser conhecido antes que uma requisição do documento seja realizada. Assim, é

mais difícil implementar uma pesquisa nesse modelo que no modelo de inundação. Além disso, pode ocorrer a formação de “ilhas”, onde a comunidade se divide em subcomunidades que não possuem nenhuma ligação entre si (KAMIENSKI et. al, 2005) (LOEST, 2007).

3. Sistemas Gerenciadores de Dados em Ambientes P2P

A evolução da tecnologia de banco de dados permitiu a sua saída de uma arquitetura centralizada para outras distribuídas, federadas, com esquemas globais únicos e para sistemas de integração de dados que propiciam aos seus usuários total transparência em seu uso. Então, foi proposto pela comunidade científica aliar a evolução tecnológica dos bancos de dados aos sistemas P2P, para a partilha de dados (HALEVY et al. 2006). Dessa união surgiram os sistemas gerenciadores de dados em ambientes P2P (*Peer Data Management Systems- PDMS*) com o intuito de prover aos usuários os benefícios de sistemas P2P e as facilidades pertencentes aos SGBDs como linguagens de consultas, modelos semânticos e facilidade de uso.

Uma vez que possui como alicerce um sistema P2P, todas as características deste são herdadas pelos PDMS. Como exemplo de tais características podemos citar a autonomia que cada *peer* possui para entrar e sair da rede. E no tocante ao ponto de vista de gerenciar dados, um PDMS deve tratar aspectos tais como (SUNG et al. 2005):

- Localização de dados: *peer* capazes de identificar e localizar dados armazenados em outros *peers*.
- Processamento de consulta: dada uma consulta realizada em um *peer*, o sistema deverá ser capaz de descobrir os *peers* que podem contribuir com informações relevantes ao que se é esperado pela consulta e processá-la de forma eficiente para os usuários.
- Integração de dados: uma vez que uma consulta foi submetida e os *peers* que atendam à consulta foram localizados, os dados acessados precisam ser integrados e retornados aos usuários, mesmo que as fontes apresentem diferentes esquemas e representações.
- Consistência de dados: em caso de replicação e uso de *cache*, a consistência dos dados deverá ser preservada.

Em um PDMS cada *peer* está associado a um esquema que representa o seu domínio de interesse, e as relações semânticas entre os pontos são organizadas entre

pares ou conjuntos pequenos de *peers*. Percorrendo caminhos obtidos através de mapeamentos, as consultas submetidas em um *peer* podem obter dados relevantes e complementares de qualquer outro *peer* na rede (HAVELVY & TATARINOV, 2004).

3.1.Requisitos

De acordo com (VALDURIEZ & PACITTI, 2004) alguns dos requisitos necessários ao se projetar um PDMS são:

- Autonomia: um *peer* dever ser capaz de entrar e sair do sistema a qualquer momento e sem qualquer restrição. Um *peer* também deve controlar seus dados armazenados e saber quais outros *peers* confiáves também podem armazenar os seus dados.
- Expressividade da consulta: um PDMS deve permitir fazer o uso de uma linguagem de consulta expressiva, com um nível adequado de detalhes. Em um ambiente onde os dados estão estruturados, uma linguagem como SQL se faz necessária.
- Eficiência: o uso eficiente de recursos de sistemas P2P (poder computacional, escalabilidade, armazenamento, tolerância a falhas, largura de banda dentre outros) deve resultar em um menor custo, possibilitando um maior processamento de consultas em um determinado intervalo de tempo.
- Qualidade do serviço: em um PDMS deve ser considerada a completude dos resultados de uma consulta, assim como o tempo de sua resposta. A disponibilidade e a consistência dos dados também devem ser consideradas.
- Tolerância a falhas: a qualidade do serviço e a sua eficiência devem existir mesmo na ocorrência da falhas. Portanto, uma solução seria a replicação de dados devido à natureza dinâmica existente nos PDMS (*peers* podem entrar e sair a qualquer momento).
- Segurança: devido à natureza aberta dos sistemas P2P, isso põe em riscos PDMS uma vez que *peers* não muito confiáveis podem entrar no sistema. Considerando esse aspecto, questões como controle de acesso e propriedade intelectual deverão ser consideradas.

3.2.Classificação dos PDMS

Além dos requisitos citados anteriormente, VALDURIEZ et al. (2004) classificam os PDMSs em:

- Não estruturados: cada *peer* pode se comunicar diretamente com seus vizinhos.
- Estruturados: são baseados em DHT.
- Super-*peer*: alguns *peers* podem realizar a tarefa mais complexa como indexação, processamento de consultas e gerenciamento de metadados.

A Tabela 1 ressalta a análise dos diversos tipos de PDMS de acordo com os requisitos definidos em (VALDURIEZ & PACITTI, 2004):

Tabela 1 - Requisitos versus tipos PDMSs

Requisito\Tipo PDMS	Não Estruturado	Estruturado	Super *peer*
Autonomia	Alta	Baixa	Média
Expressividade da consulta	Alta	Baixa	Alta
Eficiência	Baixa	Alta	Alta
Qualidade do serviço	Baixa	Alta	Alta
Tolerância a falhas	Alta	Alta	Baixa
Segurança	Baixa	Baixa	Alta

3.3.Processamento de consultas

Em se tratando de PDMS, cada *peer* disponibiliza seus dados através de seus esquemas e, para que os dados sejam encontrados se faz necessária a conexão dos *peers* através de caminhos semânticos. Nesse cenário, uma consulta submetida por um usuário tanto pode ser respondida no próprio *peer* ou por outros que fazem parte da rede. Isso se torna possível devido ao conjunto de mapeamentos entre esquemas de *peers* distintos, que permitem que consultas submetidas à um *peer* possam ser reformuladas e executadas em outros *peers*.

Em alguns casos, um *peer* pode agir apenas como um mediador para repassar uma consulta a outros *peers*. Esse tipo de *peer* não disponibiliza dados, ele apenas possui um esquema virtual que permite estabelecer o mapeamento transitivo entre *peers* que possuem dados a serem disponibilizados.

Devido ao dinamismo existente em PDMS, onde vários *peers* podem se conectar e desconectar da rede, o volume de esquemas e mapeamentos podem aumentar ou diminuir, podendo gerar problemas referentes à perdas semânticas, caminhos extensos e restrições ao tempo de respostas. E, para resolver o problema de mapeamento, ontologias podem ser utilizadas para gerar atalhos entre *peers* não-adjacentes, mas que possuem conteúdos relacionados semanticamente. O uso de ontologia em PDMSs pode ser observado em (PIRES, 2009).

Em uma arquitetura pura, um processamento de consulta tem como vantagem a escalabilidade, uma vez que qualquer *peer* pode conectar-se ou desconectar-se sem que haja interferência nos demais *peers* e no andamento da consulta. Outra vantagem refere-se à inexistência de falhas, pois as respostas são geralmente encontradas mesmo que nem todos os *peers* estejam conectados, a não ser que a consulta retorne um resultado vazio. E como desvantagens, o algoritmo pode seguir por caminhos não ótimos, sem recuperar respostas, seguir por caminhos redundantes obtendo consultas desnecessárias e, o tempo de resposta tornar-se bastante alto.

No caso de uma arquitetura de super-*peer*, o processamento de consulta trás como vantagem um melhor desempenho na execução da consulta, se comparado à arquitetura pura, pois na super-*peer* o sistema é fracionado em pequenos grupos que possuem interesses em comum, facilitando o roteamento da consulta. E, como os *peers* que pertencem a um domínio de conhecimentos comuns são agrupados próximos uns aos outros, existe uma unidade controladora que se encarrega de integrar os resultados das consultas. Já as desvantagens estão no fato dos super-*peers* poderem se tornar pontos críticos de falhas e, ao mesmo tempo, degradarem o desempenho da resposta da consulta.

3.4.Alguns PDMSs

Atualmente, diversos são os PDMSs existentes, porém cada um com suas características diferenciadas pelas suas arquiteturas e pela forma de processar suas consultas. Nesta seção, serão apresentados alguns desse PDMSs.

3.4.1 PeerDB

Desenvolvido pela Universidade de Singapura, o PeerDB é um PDMS implementado sobre a plataforma BestPeer (NG et al., 2002), baseado na topologia pura não-estruturada, onde cada *peer* possui dados que são gerenciados por um SGBD relacional, proporcionando buscas por conteúdo. Os dados gerenciados são compartilhados sem a presença de um esquema global e acessado através da linguagem SQL.

A Figura 8 ilustra a arquitetura do PeerDB que basicamente é composta por três camadas:

- Camada P2P: camada responsável por ofertar serviços de descoberta de recursos na rede e por compartilhar dados.
- Camada de agentes inteligentes: sistema multi-agentes que viabiliza uma infra-estrutura para que agentes móveis possam operar nos diversos *peers* da rede. E, em cada *peer* existe um agente responsável por gerenciar as consultas dos usuários, esse agente é chamado de DBAgent. Além disso, o DBAgent monitora estatísticas sobre *peers* vizinhos e gerencia políticas de configuração da rede.
- Camada de gerenciamento de dados: camada responsável pela manipulação dos dados disponíveis no *peer*. Nesta existe mecanismos de *cache* para proporcionar o armazenamento temporário dos resultados provenientes de diversos *peers*, com o intuito de minimizar o tempo de resposta às consultas subsequentes. Para o gerenciamento de *cache*, o PeerDB faz uso do algorítimo LRU (O'NEIL et al., 1993).

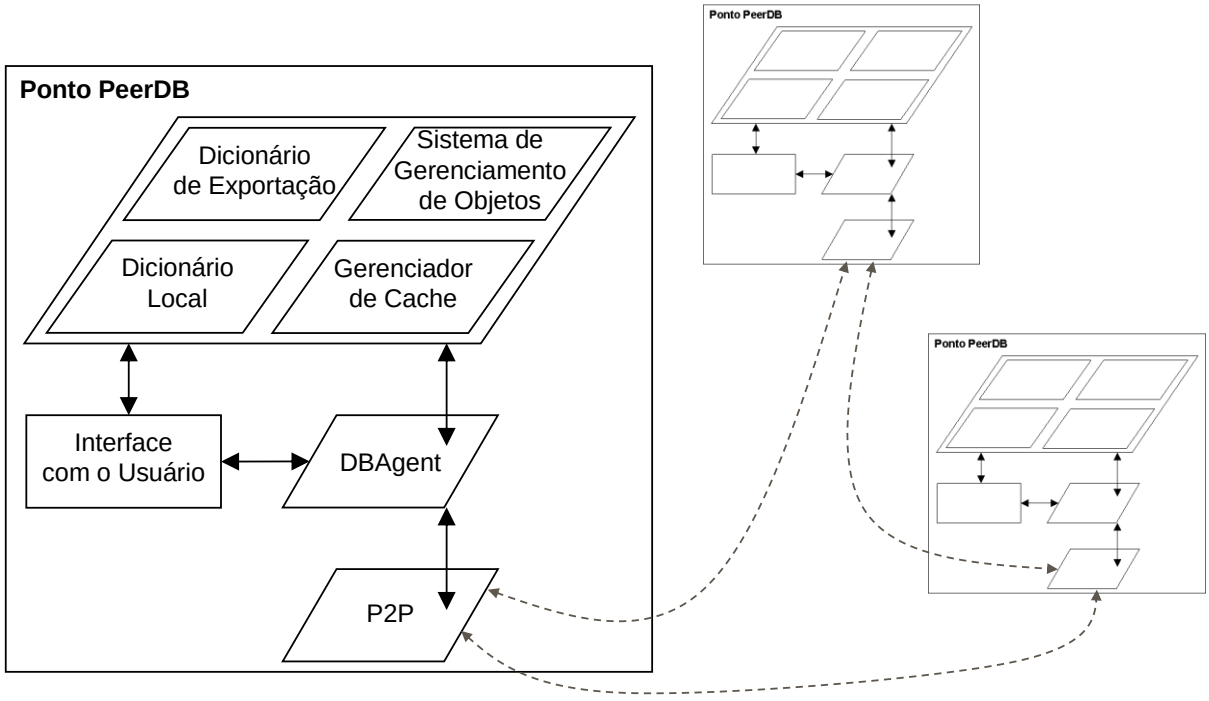

Figura 8-Arquitetura do PeerDB (OOI et al., 2003)

3.4.1.1 Processamento de consulta

Uma vez submetida uma consulta em um *peer*, um agente DBAgent pertencente este *peer* cria agentes auxiliares e os enviam ao conjunto de *peers* vizinhos. No entanto, apenas os agentes auxiliares aptos à responderem a consulta retornam metadados referentes às tabelas e colunas envolvidas. O DBAgente, situado no *peer* onde a consulta foi submetida, compara sintaticamente os metadados recebidos com os metadados do seu *peer*. E se surgir conflitos semânticos, o DBAgente interage com o usuário a fim de resolve-los. No final, os dados são combinados e apresentados ao usuário.

Através de comparações sintáticas entre os nomes de tabelas e de colunas da consulta e as palavras-chave contidas no dicionário de exportação (possui metadados referentes às tabelas e as colunas comparilhadas) dos *peers* envolvidos, é que um *peer* é escolhido. E, durante o processamento de consultas o usuário necessita confirmar os pontos que devem ser consultados. Com isso, o PeerDB fica dependente da interseção do usuário. Além disso, a qualidade do resultado das consultas depende do processo de definição de palavras-chave e, no caso de ocorrência de homônimos, o resultado da consulta pode ficar comprometido (OOI et al., 2003).

3.4.2 Piazza

Desenvolvido pela Universidade de Washington e da Pensilvânia, o Piazza foi concebido com o intuito de ser um PDMS escalonável em um ambiente heterogêneo e distribuído. O sistema assume que usuários participantes interessados em compartilhar seus dados, devem estar dispostos a definir mapeamentos entre seus esquemas, conforme ilustra a Figura 9. Cada *peer* compartilha seu dado na forma de relações armazenadas. O *peer* define o seu esquema de forma que outros *peers* possam acessar suas relações armazenadas. Além disso, o *peer* mantém dois tipos de mapeamentos de esquemas. O primeiro refere-se às relações armazenadas e ao esquema do *peer*. O segundo refere-se ao mapeamento entre o esquema do *peer* com os esquemas de seus vizinhos (HALEVY et al., 2003a).

Em (HALEVY et al., 2003b) encontra-se relatado que o Piazza suporta compartilhamento de dados em XML/RDF para suporte a aplicações de *web* semântica. Portanto, o esquema do *peer* é descrito usando XML *Schema* ou ontologias na liguagem OWL para dados XML e RDF respectivamente. Além disso, a linguagem para mapeamento e para consulta é baseada em XQuery.

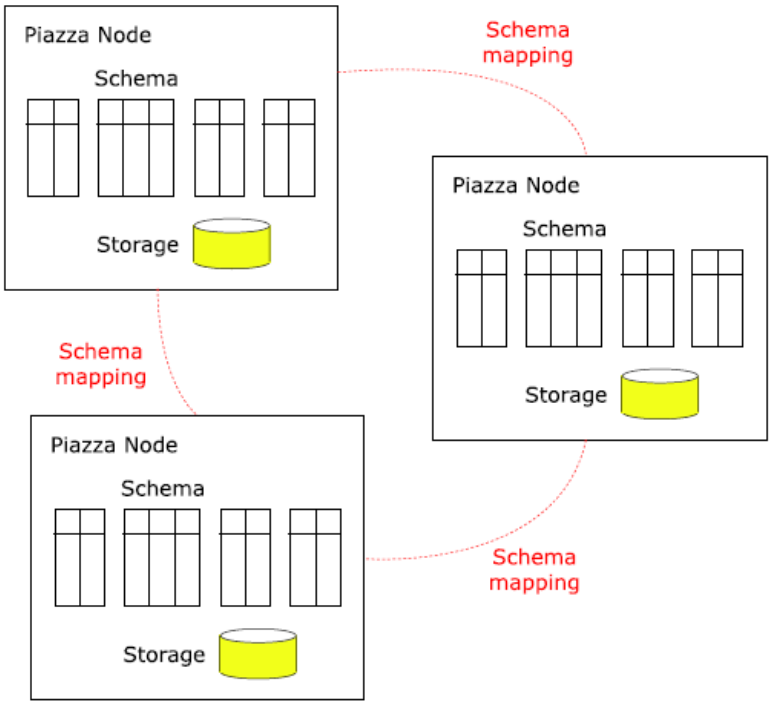

Figura 9-Arquitetura do Piazza (VU et al., 2010)

O Piazza possui uma arquitetura puramente descentralizada, inexistindo a presença de um esquema global único. O conjunto de mapeamentos definidos por este sistema visa definir a semântica do mesmo. O mapeamento é implementado através de um gráfico arbitrário de esquemas conectados, sendo alguns destes esquemas definidos virtualmente para propósitos de consulta e de mapeamento. E, para a construção dos mapeamentos, o Piazza baseia-se no uso conjunto de heurísticas e algoritmos a fim de resolver problemas semânticos. Estes por sua vez são baseados em técnicas de aprendizagem de máquina e na exploração de experiências anteriores, onde informações sobre mapeamentos válidos já existentes são utilizados para o mapeamento de novos esquemas.

3.4.2.1 Processamento de consulta

Para que uma consulta Q seja processada e seu resultado retornado ao usuário, o Piazza reformula Q em uma consulta Q_1 que referencia apenas tabelas armazenadas. Depois o sistema avalia Q_1 de forma a obter o resultado, através de técnicas de processamento de consultas adaptativas (TATARINOV et al., 2003). O algoritmo utilizado pelo Piazza aceita como entrada um conjunto de mapeamentos de *peers* e descrições de armazenamento, além da consulta Q propriamente dita, e retorna como saída uma consulta reformulada Q_1.

Através do envio de consultas reformuladas, um *peer* pode obter resultados de outros *peers* que tenham respostas e eliminar resultados redundantes. E, para identificar os *peers* relevantes ao usuário, o sistema constrói um índice para explorar informações sobre os reais dados de um *peer*. Esse índice é implementado como uma máquina de busca *web*, diferentemente de outros sistemas que adotam uma estratégia descentralizada baseada em DHT.

3.4.3 SEWAISE

SEWASIE (*SEmantic Webs and AgentS in Integrated Economies*) é um projeto de pesquisa, financiado pela Comissão Européia, que visa implementar um mecanismo de busca avançado e que permita acesso inteligente a fontes de dados heterogêneas espalhadas na *web*, através de enriquecimento semântico (SEWAISE, 2010).

SEWAISE foi projetado utilizando sistemas multi-agentes com base segura, escalável e com foca na arquitetura de um sistema distribuído, para busca semântica com ontologias específicas da comunidade multilíngüe. Ontologias são utilizadas em camadas de inferência e baseadas em padrões W3C (BENEVENTANO et al., 2003).

A Figura 10 especifica os componentes da arquitetura do SEWAISE, e são (BENEVENTANO et al., 2003):

- *User interface*

  Possui um conjunto de módulos integrados que juntos propiciam ao usuário uma interação com um sistema de busca semântica. Esta interface é personalizada e configurada com o perfil de um usuário específico e com uma referência para as ontologias que são comumente usadas por esse usuário.

  Há dois elementos extras nesse componente que são as ferramentas de visualização e de comunicação. O primeiro é responsável por monitorar as fontes de informações de acordo com os interesses do usuário definidos em *"monitoramento de perfis"*. Nesse caso, um agente fica responsável por realizar o monitoramento dos interesses do usuário. Já a segunda ferramenta fornece suporte a negociação entre o usuário e as outras partes. Nesse caso, agentes podem interceder ajudando a encontrar parceiros de negócios, solicitando ofertas e realizando *ranking* delas (FILLOTTRANI, 2005).

- SINode (SEWAISE *Information Node*)

Os grupos de SINode unem os módulos os quais trabalham para definir e gerenciar uma ontologia global integrada apresentada na rede. Um único SINode pode abranger diferentes sistemas.

Os SINodes são baseados em sistemas mediadores, onde cada um inclui uma visão global da informação detida. As fontes de informações são heterogêneas (dados estruturados, não-estruturados ou semi-estruturados). O SINode realiza o processamento de enriquecimento semântico de forma semi-automática, para criar a ontologia SINode. Por sua vez, esta ontologia do SINode também está integrada com outros componente similares aos do mapeamento de ontologias usados pelos *broker agents* (FILLOTTRANI, 2005).

- *Broker agent (BA):*

São os *peers* responsáveis por gerenciar uma visão do conhecimento manipulado pela rede, assim como a informação específica de cada SINode. Um *broker agent* trabalha como um intermediário detendo do controle direto sobre o número de SINodes, e fornecendo meios para publicar um manifesto no âmbito da rede, no tocante as informações mantidas localmente em um perfil semântico (FILLOTTRANI, 2005).

- *Query agent (QA)*

O QAs são os portadores das consultas a partir da interface do usuário até os SINodes. Eles interagem com um ou vários *broker agents* no intuito de resolver uma consulta e integrar as respostas ao usuário. Um QA traduz a consulta de acordo com o mapeamento ontológico do BA e pede, diretamente aos SINodes, a coleção dos resultados parciais.

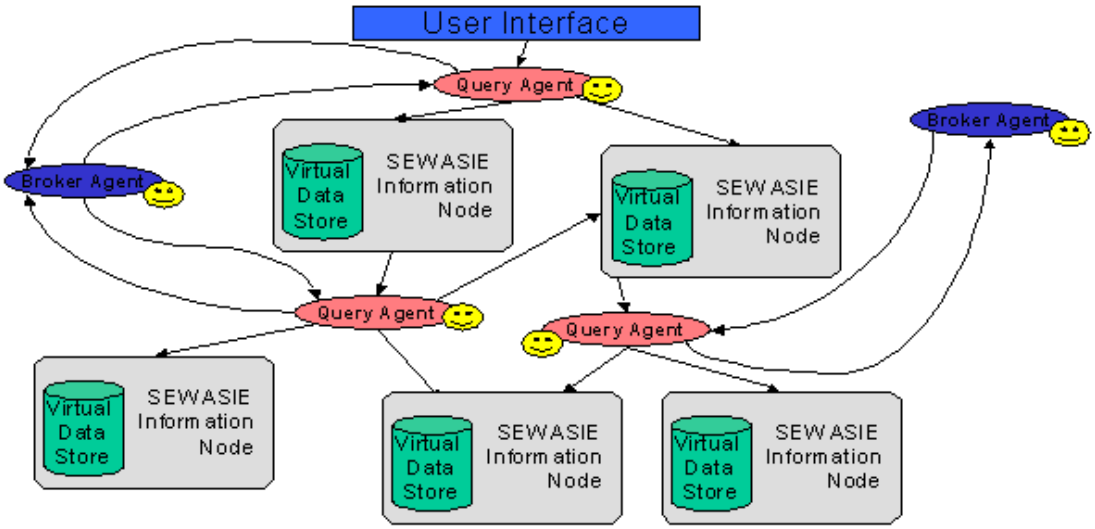

Figura 10 - Arquitetura SEWAISE (SEWAISE, 2010)

3.4.3.1 Processamento de Consulta

O processamento de consulta no SEWASIE leva em consideração dois diferentes níveis de mapeamento. O primeiro é feito dentro de um SINode, mapeando as fontes de dados para o GVV (Global Virtual View)[2] do SINode. O segundo mapeamento é feito no nível BA, mapeando vários GVVs dos SINodes para uma ontologia de um BA. Em geral, o mapeamento das fontes de dados para ontologia do BA não é uma simples composição de mapeamentos. No caso em que os mapeamentos são definidos como GAV, os mapeamentos podem ser compostos e, a reformulação da consulta para uma ontologia de BA pode ser feita em dois passos: primeiro, a consulta é expandida e desdobrada em restrições w.r.t. e mapeadas para ontologia do BA. Então, um processo semelhante é feito dentro de um w.r.t. do SINode para a ontologia do SINode. O resultado deste processo é uma consulta expandida e um conjunto de consultas sobre os SINodes. No final, um agente de consulta é responsável pela execução dessas consultas desdobradas e pela fusão dos resultados obtidos pela consulta (SEWAISE, 2010). O processo de consulta no SEWASIE pode ser visualizado na Figura 11.

[2] O mapeamento GVV pode ser visto com detalhes em (FILLOTTRANI, 2005).

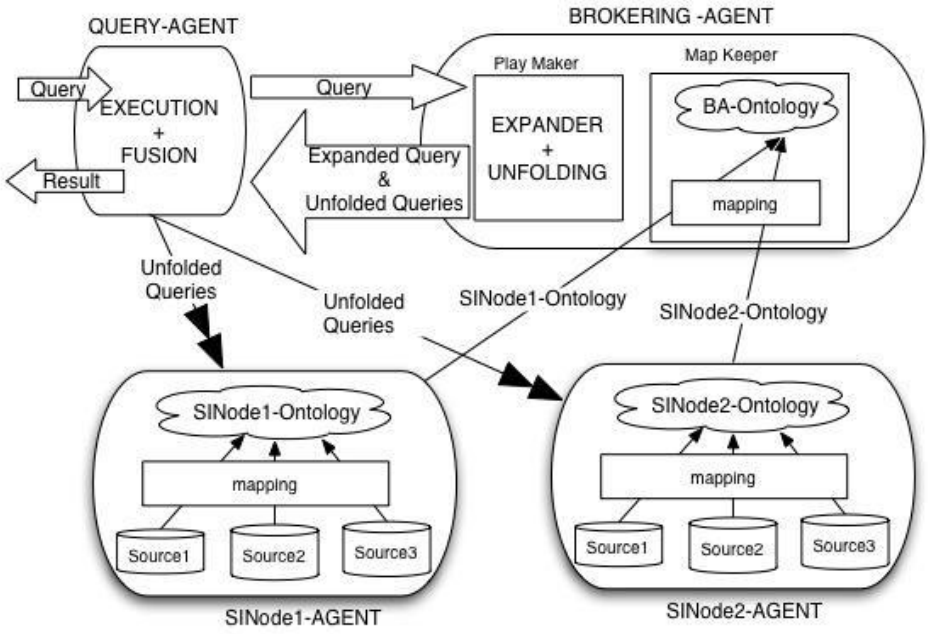

Figura 11-Processamento de consulta no SEWASIE (SEWAISE, 2010)

3.4.4 SPEED

Desenvolvido pela Universidade Federal de Pernambuco, o SPEED (*Semantic PEer-to-peer Data Management System*) é um PDMS que possui uma topologia de rede mista: DHT, super *peer* e não estruturada (PIRES, 2009). O SPEED realiza a "clusterização" de *peers* que possuem similaridades semânticas, com o intuito de facilitar o mapeamento semântico entre *peers* e como conseqüência, facilitar o processamento de consulta sobre o grande volume de fontes de dados.

A Figura 12 ilustra a arquitetura do SPEED. Nela podemos observar (PIRES, 2009):

- Ponto de dados: *peers* quem possuem dados a serem compartilhados com outros *peers*.
- Ponto de integração: *peer* responsável por processamento de consulta, indexação de metadados e integração de dados. Pontos de integração são pontos de dados com alta disponibilidade e poder computacional, além disso, eles nomeiam seus respectivos clusters semânticos.
- Ponto semântico: *peer* que armazena e oferece uma ontologia padrão de um domínio específico (por exemplo, "saúde", "educação", "engenharia" etc.), e nomeia suas comunidades semânticas correspondentes.
- Cluster semântico: está associado a um interesse comum entre pontos de dados e possui um ponto de integração.

- Comunidade semântica: uma comunidade semântica é construída através da composição de clusters que possuem interesses e semânticas comuns. Ao entrar no sistema, um ponto de dados é selecionado, por um ponto semântico, para entrar em um cluster semântico apropriado, onde o ponto de dados deve estar conectado.
- Topologia de super-*peer*: com o intuito de fragmentar a rede em porções mais fáceis de gerenciar, o conceito de cluster alia-se ao de super-*peer*. Com o emprego desta topologia, existe uma maior exploração da heterogeneidade dos pontos participantes.
- Topologia DHT: utilizada para auxiliar *peers* que possuem interesses comuns a encontrar um ao outro e construir comunidades semânticas. Essa topologia é composta pelos pontos semânticos com boa largura de banda e que permanecem na rede por longos períodos de tempo.

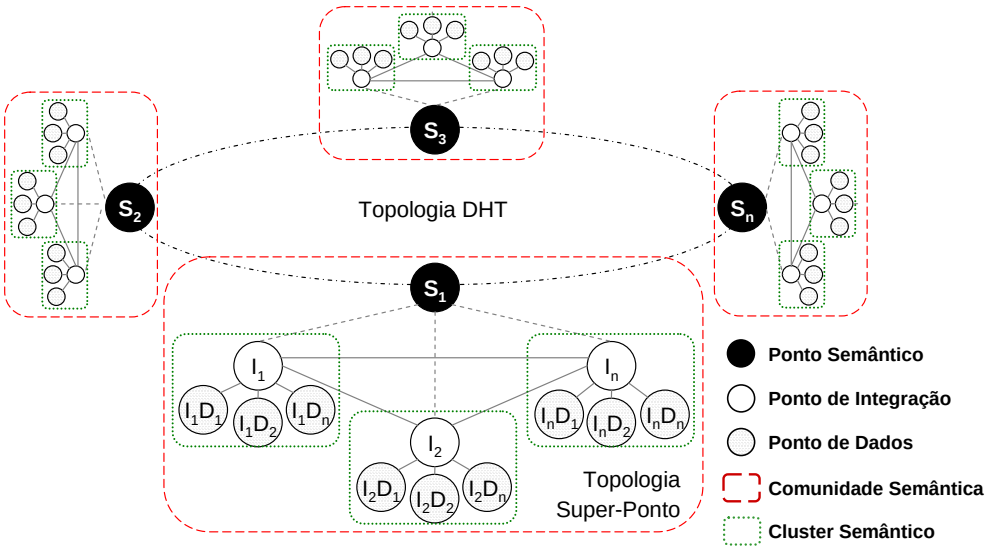

Figura 12-Arquitetura do SPEED (PIRES, 2009)

3.4.4.1 Processamento de consulta

No SPEED, uma consulta submetida por um *peer* é reformulada de acordo com o esquema exportado por este *peer*, e traduzida em um modelo de consulta comum. E, essa consulta tanto pode ser respondida tanto por um *peer* de dados quanto por um *peer* de integração. A consulta é disseminada somente entre os clusters da comunidade semântica onde ela foi submetida (PIRES, 2009). Os *peers* semânticos não participam do processamento de consulta, logo se uma consulta for submetida a uma determinada comunidade semântica, a consulta não é encaminhada para outras comunidades. Por onde a consulta passar na rede, ela é reformulada de acordo com o mapeamento semântico previamente estabelecido. Os *peers*

de integração se responsabilizam por integrar os resultados retornados pelos *peers* de dados e de integração.

3.4.5 iXPeer

Desenvolvido através de pesquisas da Universidade de Montpellier, o iXPeer é um PDMS baseado na arquitetura de super-*peer* e oferece tempo de resposta aceitáveis aos usuários. O iXPeer é uma extensão do XPeer (BELLAHSÈNE et al., 2004a) (BELLAHSÈNE et al., 2004b) resolvendo problemas susceptiveis a falhas existentes neste ultimo. Tais problemas podem ser vistos em (BELLAHSÈNE et al., 2006a) (JUNIOR, 2008). O iXPeer lida com a integração de dados nos vários níveis de abstração, onde em um nível mais baixo define mapeamentos precisos entre esquemas de fontes de dados, e em um nível mais alto as associações são feitas baseadas em palavras-chave.

Neste sistema, os *peers* que compartilham os mesmos esquemas são agrupados em um único *cluster*, uma vez que eles compartilham informações sobre um domínio único. Onde em cada um deste, o esquema compartilhado é gerenciado por um super-*peer*, e todos os *peers* de um mesmo *cluster* comunicam-se através do super-*peer*. A Figura 13 ilustra *peers* ligados a super-*peers* dentro de um mesmo domínio, porém é possível ter super-*peers* agrupando *peers* que compartilham informações sobre o mesmo domínio. Neste caso, uma aplicação que realiza uma consulta sobre um determinado domínio, o resultado para ela poderá estar em vários *clusters*.

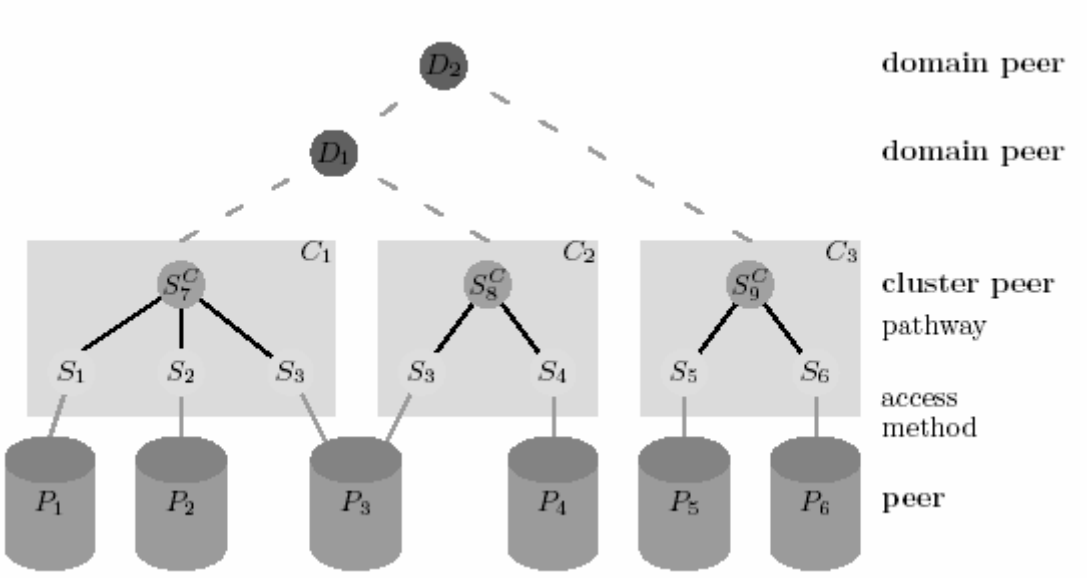

Figura 13-Arquitetura do iXPeer (BELLAHSÈNE et al., 2006a)

A implementação do iXPeer foi realizada utilizando as principais características do AutoMed (*Automatic Generation of Mediator Tools for Heterogeneous Database Integration*) (BOYD et al., 2004). Este faz uso de uma metodologia onde o processo de integração é visto como uma seqüência de transformações reversíveis que modificam tanto o esquema como a extensão do banco de dados. Logo, o AutoMed provê uma implementação de uma abordagem em mediador e que permite mapeamentos bidirecionais entre fontes de dados e um mediador.

3.4.5.1 Processamento de consulta

No iXPeer (BELLAHSÈNE, 2006b), o usuário formula sua consulta *Q* em um *peer* utilizando um esquema mediador, que possui a visão deste *peer*. Em seguida, esta consulta é submetida ao *cluster*, onde um *peer* irá processá-la. A consulta *Q* é reescrita em um conjunto de sub-consultas $\{Q_1, Q_2, ..., Q_k\}$, onde *k* é o número de esquemas de *clusters* que foram integrados.

Cada consulta Q_1 está relacionada a um esquema de *cluster*. Cada *cluster* pode agora processar sua consulta relacionada da seguinte forma: dentro de *cluster*, cada sub-consulta é traduzida para outro subconjunto de consultas, um para cada *peer* local pertencente ao *cluster*. Essa tradução toma por base o esquema desses *peers* locais, de acordo com as regra de mapeamentos definidas. No final, os resultados enviados por esses *peers* locais são integrados pelo *peer* de domínio (*domain peer*), através *peers* dos *cluster*, em um único resultado que será devolvido ao *peer* que iniciou a consulta.

3.4.6 SUNRISE

Com a proposta de possuir uma completa infra-estrutura, a qual estende cada *peer* com funcionalidades para capturar uma aproximação semântica proveniente de um esquema heterogêneo, e explorá-lo para uma organização de uma rede semântica e roteamento de consulta, foi proposto o SUNRISE (*System for Unified Network Routing, Indexing and Semantic Exploration*) (MANDREOLI et al., 2008). Estee PDMS foi desenvolvido pela Universidade de Modena e Reggio Emilia, na Itália.

O SUNRISE suporta a criação e manutenção da organização de uma rede flexível que realiza *clusters* dos *peers* que estão semanticamente relacionados (de forma semelhante ao iXPEER, porém em *Semantic Overlay Networks*-SONs). Além disso, o sistema fornece um

conjunto de técnicas que podem ser usadas para a efetiva e eficiente exploração da rede como pode ser observado em (MANDREOLI et al., 2007).

A Figura 14 ilustra duas SONs com dados relacionados a “cinema” e cada *peer* está representado por um tópico principal de interesse derivado do seu esquema. Podemos observar que alguns *peers* são monotemáticos (só lidam com um assunto), como é o caso dos *peers* B e F. Outros *peers* estão preocupados com ambos os temas, como o caso do *peer* C.

Podemos observar que um *peer* pode estar na base de uma ou mais SONs, e para um PDMS esta operação é um desafio devido a falta de um entendimento comum entre os dicionários dos *peers* locais. Para isso, o SUNRISE assiste a entrada de novos *peers* em duas fases: realiza a escolha da SON mais próxima; escolhida a SON, dentro dela uma seleção refinada dos melhores vizinhos entre os *peers* mais relacionados semanticamente é realizada (LODI et al., 2008).

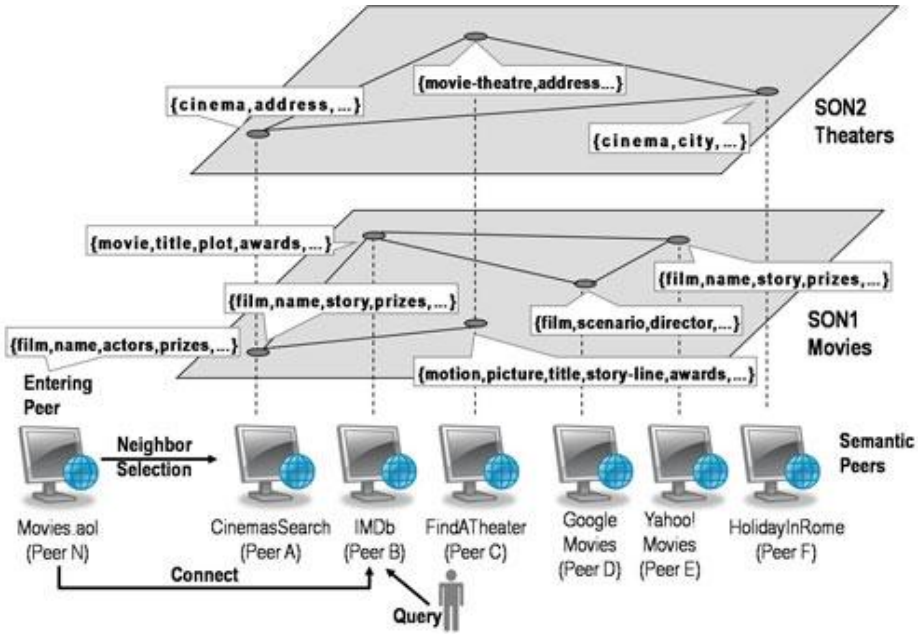

Figura 14-Exemplo da organização da rede no SUNRISE (MANDREOLI et al., 2008)

As descrições resumidas sobre as SONs disponíveis na rede estão descritas em uma estrutura chamada *Access Point Structure* (APS), a fim de ajudar os novos *peers* à decidirem que SONs participar ou se for o caso, formar novas SONs.

3.4.6.1 Processamento de consulta

O SUNRISE evita que as consultas façam broadcasting, pois isto sobrecarregaria a rede e provavelmente a consulta retornaria resultados irrelevantes. Além disso, o sistema explora técnicas para selecionar a direção que é mais provável oferecer os melhores resultados para uma consulta (LODI et al., 2008). Cada *peer* mantêm um índice de roteamento semântico (*Semantic Routing Index*-SRI) que sumariza, para cada conceito em seu esquema, o conhecimento semântico sobre as sub-redes de seus vizinhos, podendo fornecer sugestões onde o dado pode ser alcançado em cada trajeto, conforme ilustrado na Figura 15.

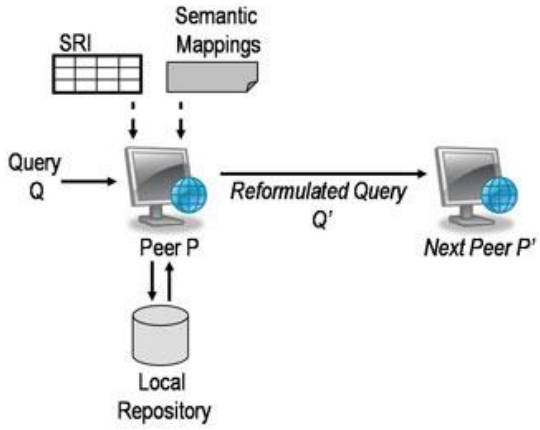

Figura 15-Processamento de consulta no SUNRISE (SUNRISE, 2010)

3.4.7 PEPSINT

Desenvolvido pela Universidade de Illinois, em Chicago, o PEPSINT (*PEer-to-peer Semantic INTegration*) foi proposto para realizar integração semântica em fontes heterogêneos, as quais estão em XML e RDF (CRUZ et al., 2004). Este PDMS foi construído em uma arquitetura híbrida, a qual a ontologia RDF global (construída usando GAV) no super-*peer* comporta-se não apenas como um ponto de controle central sobre os *peers*, mas também como um mediador de consulta de um *peer* para outro.

A proposta do PEPSINT é de preservar a estrutura do domínio existente em RDF e em XML, implementando a integração semântica em nível de esquema (através de um processo de correspondência de esquemas) e de instância (através do processo de resposta de uma consulta).

Por ser baseado na arquitetura híbrida, o PEPSINT contém os dois tipos de *peers:* super-*peer* contendo a ontologia global em RDF e os *peers* contendo esquemas e fontes de dados locais. Conforme ilustra a Figura 16, a arquitetura do PEPSINT é composta por quatro principais componentes: **XML *to* RDF *wrapper*** usado para transformar o esquema XML para um esquema RDF local, e este mapeado para a ontologia global. ***Local* XML *and* RDF *schemas***

estão presentes em *peers* que possuem dados e metadados, e possuem o propósito de integração semântica. ***Global* RDF *ontology*** localizada no super-*peer* com o propósito de atuar como um esquema mediador virtual integrado dos esquemas RDFs locais. Trata-se de uma simples ontologia que não possui altos níveis de axiomas. ***Mapping table*** armazena os mapeamentos entre os esquemas locais e a ontologia global.

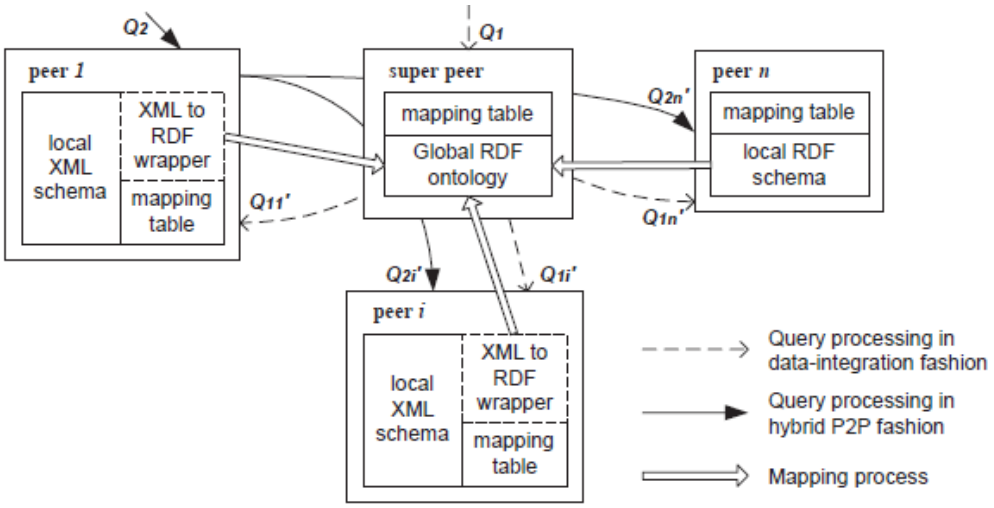

Figura 16-Arquitetura do PEPSINT (CRUZ et al., 2004)

3.4.7.1 Processamento de consulta

Segundo XIAO (2006), o usuário pode submeter uma consulta a uma fonte de dados local (XML ou RDF), em qualquer *peer*. Localmente, a consulta será executada na fonte local para obter uma resposta local. Entretanto, a consulta submetida em uma fonte é reescrita em uma consulta direcionada a cada *peer* conectado. A reescrita de consulta utiliza a ontologia global, e a composição de mapeamentos a partir do *peer* inicial ao super-*peer,* com mapeamentos do super-*peer* para os *peers* alvos. Ao executar a consulta, os *peers* aos alvos retornam a resposta ao *peer* inicial, essa resposta é chamada de "resposta remota". As respostas locais e remotas são integradas e retornadas ao usuário. As formas de processamento de consulta em *data-integration* e *hybrid P2P fashion*, conforme ilustrado na Figura 15, estão descritas em (CRUZ et al., 2004).

3.4.8 Outros PDMSs

3.4.8.1 Edutella

O projeto Edutella (NEJDL et al., 2003), desenvolvido pela Universidade de Hannover, na Alemanha, provê uma infra-estrutura onde os metadados encontram descritos em RDF, em uma rede baseada no framework JXTA. Neste PDMS, as conexões entre os *peers* são codificadas em uma topologia de super-*peer*, a qual se assemelha a arquitetura híbrida usada no PEPSINT. Além disso, existe uma linguagem chamada RDF-QEL usada para definir um formato comum de intercambio para realização de consultas. Assim, um *wrapper* traduz uma consulta local escrita em XPath ou SQL para RDF-QEL. Porém, o Edutella não oferece suporte direto a fontes em XML, embora fontes de dados em RDF possam ser serializadas em formatos XML.

3.4.8.2 ROSA– P2P

Desenvolvido pelo Instituto Militar de Engenharia (IME-RJ), o Rosa-P2P (BRITO et al., 2005) é um PDMS concebido para compartilhar dados, cujo conteúdo é acadêmico. Trata-se da evolução do sistema ROSA (*Repository of Objects with Semantic Access*). O ROSA é um sistema centralizado que armazena objetos de aprendizagem com conteúdos instrucionais. No entanto, o ROSA-P2P levou o ROSA a um ambiente P2P, inspirado no Edutella. O ROSA-P2P é baseado na arquitetura de super-*peer* e faz uso de uma estrutura semântica complexa para resolver conflitos semânticos. A otimização do processamento de consulta é feito através do uso de agregações e agrupamentos de *peers*.

3.4.8.3 Humboltd Discoverer

O Humboldt Discoverer provê um índice semântico para uma arquitetura PDMS estendida (nas dimensões semântica, *web* e qualidade), conforme pode ser visto em (HERSCHEL et al., 2005). Esse índice é utilizado para localizar fontes de informações relevantes que não são alcançáveis por meio de caminhos de mapeamentos convencionais. O *Humboldt Discoverer* implementa uma infra-estrutura híbrida combinando DHT com uma rede não estruturada, cujos esquemas dos *peers* são indexados por tecnologias da *web* semântica. HERSCHEL et al. (2005) acreditam que com a indexação das informações dos esquemas e com a escolha de uma arquitetura híbrida, os índices são insensíveis as alterações na estrutura da rede, assim como sobre a atualização dos dados (HERSCHEL et al., 2005). A dimensão de qualidade abordada no *Humboldt Discoverer* influencia no processamento de consultas, pois a qualidade do resultado da consulta é diretamente dependente dos caminhos de mapeamentos

providos por dois *peers*. E na dimensão *web*, métricas para localizar e realizar *ranking* de *peers* são definidas para responder uma consulta.

3.4.8.4 GridVine

Baseado em uma estrutura de acesso descentralizada, o GridVine (MAUROX et al., 2007) foi projetado seguinte o principio de independência de dados , separando a camada lógica, onde dados e esquemas e seus mapeamentos semânticos são gerenciados, da camada física que consiste de uma rede P2P *overlay*, a qual suporta um eficiente mecanismo de roteamento de mensagens e balanceamento de carga de índices. De acordo com ABERER et al. (2004), essa independência permite processar operações lógicas na sobreposição semântica usando diferentes estratégias de execução. Em (ABERER et al. 2004) são identificadas técnicas, iterativas e recursivas, para "*transversalizar*" uma SON. Este PDMS detém um conjunto de algoritmos que organizam automaticamente a rede de mapeamento de esquemas. Além disso, segundo ABERER et al.(2004), o GridVine oferece uma solução completa para implementação de SONs com suporte a interoperabilidade e escalabilidade do sistema.

3.4.8.5 Hyperion

O Hyperion (ARENAS et al., 2003) foi proposto para ser um PDMS para bancos de dados relacionais (um em cada *peer*). O Hyperion possui um mecanismo semelhante de *mapping table*, adotado no PEPSINT, para armazenar as conexões entre os esquemas locais nos *peers*. Além disso, possui um gerenciador de consulta que faz uso do *mapping table* para reescrever uma consulta postada em termos de um esquema local. O processo de reescrita produz uma consulta que é executada sobre *peers* próximos.

O quadro a seguir resume e compara algumas características gerais dos PDMSs descritos nesta seção.

PDMS	Representação dos dados	Modelo de Arquitetura	Linguagem de Consulta	Processamento de Consultas	Middleware para ambiente distribuído	Outras características
Piazza	XML, RDF e DAML+OIL	Pura	XQuery	Consulta reescrita com base em mapeamentos; Técnicas de aprendizagem de máquina e exploração de casos anteriores	JXTA (HALEVY et al., 2004)	Estrutura de índice para auxiliar na identificação e otimização da busca.
SPEED	Estruturado e semi-estruturado	Super-*peer*	SPARQL	Uso de índice de consultas baseados em ORC (ontologia de referencia de *cluster*); Merge entre ontologia do *peer* com a ORC.	RMI (protótipo)	Formação de comunidades semânticas, envolvendo clusters que possuem o mesmo domínio.
SUNRISE	XML	Pura	XQuery	Cada peer mantêm um índice de roteamento semântico; Indices que fornecem sugestões onde o dado pode ser alcançado em cada trajeto.	Não identificado	Rede organizada em um conjunto de SONs; SONs com *peers* relacionados semanticamente; Algoritmo de organização de *peers* inspirado no algoritmo *incremental clustering*. APS para ajudar a um *peer* entrar na rede.
SEWASIE	Estruturado, semi-estruturado e não estruturado	Pura	Nativa do SEWASIE	Agentes auxiliam no processamento de consultas, integração dos resultados e mapeamento dos esquemas.	JADE (FILLOTTRANI, 2005)	Possui interface de consulta inteligente, já que possui um raciocinador *online*;Integração com ferramentas OLAP; Sistemas multi-agentes.
PeerDB	Estruturado	Pura	SQL	Agentes inteligentes; Função de similaridade; Forte dependência e interatividade	JADE	Possui agentes que monitoram estatísticas sobre *peers* vizinhos e que gerenciam políticas de configuração na rede; Sistemas

				com o usuário.		multi-agentes
Rosa-P2P	XML e RDF	Super-*peer*	RosaSQL	Utiliza uma máquina de execução chamada de MEC ROSA. Cada ponto processa a consulta da mesma maneira e o ponto solicitador, de forma autônoma.	Sockets (BRITO, 2005)	Agrupamentos baseados em assunto e localização.
iXPeer	Dados relacionais, XML, RDF e textos semi-estruturados tal como o CSV.	Super-*peer*	XQuery	Consultas reescritas pelos clusters e os resultados são integrados pelos *domain peers* via *cluster peers*.	JXTA (BELLAHSÈNEet al., 2006)	Gerenciamento de metadados através de um modelo de repositório; cluster de p*eers* agrupados por domínios em comum; Utilização do framework AutoMed.
Hyperion	Relacional	Pura	SQL	Consultas reescritas através do uso de tabelas e expressões de mapeamento.	JXTA (protótipo) (GIANOLLI, et al., 2005)	Tabelas de mapeamentos e de expressões são usadas para armazenar conexões entre esquemas locais no *peers*.
PEPSINT	XML e RDF	Híbrida	XQL e RDQL	Reescrita de consulta através de ontologia global; Data-integration *fashion* e hybrid P2P *fashion.*	Não identificado	Tabelas de mapeamento para armazenar correspondência entre esquemas.
Edutella	XML e RDF	Super-*peer*	RDF-QEL	Geração de plano de consulta a partir de índices, metadados e mapeamentos; Uso de índices de roteamento para determinar *peers* relevantes.	JXTA (CRUZ, et al., 2004)	Regras baseadas em *clustering*.

Tabela 2 - Quadro comparativo entre PDMSs

4. Considerações Finais

Neste trabalho foram apresentados conceitos referentes à sistemas P2P e aos PDMSs, enfatizando a realidade e importância de seus usos no dias atuais. Vimos que enquanto os sistemas P2P são adequados para compartilhamento de arquivos, troca de mensagens e de trabalho colaborativo, porém são ineficazes em caso de abstração de dados em fontes autônomas e distribuídas por não se preocuparem com essa abstração. Nesta ineficácia, surgem os PDMSs para resolverem essa situação, a fim de integrar dados. Considerados uma evolução dos sistemas de integração de dados, os PDMSs provêem escalabilidade e flexibilidade, em um ambiente onde não há a necessidade de um esquema global e tão pouco uma autoridade central.

Foram descritas aqui as características arquiteturais tanto em sistemas P2P quanto em PDMSs. Exemplos de implementações desses dois tipos de sistemas foram descritos, considerando os seus aspectos estruturais e características específicas. E por fim, um quadro resumido comparou alguns dos PDMSs pesquisados neste trabalho.

5. Referências Bibliográficas

ABERER, K., MAUROUX, P., HAUSWIRTH, M., & PELT, T. (2004). GridVine: Building Internet-Scale Semantic Overlay Networks. Proceedings of the 3rd International Semantic Web Conference (ISWC2004) , pp. 107–121. Hiroshima, Japão.

ANDROUTSELLIS-THEOTOKIS, S., & SPINELLIS, D. (2004). A survey of peer-to-peer content distribution techonologies. *ACM Computing Survey* , vol. 36, pp. 335–371.

ARENAS, M., KANTERE, V., KIRINGA, I., MILLER, R., MYLOPOULOS, J., & KEMENTSIETSIDIS, A. (2003). The hyperion project: from data integration to data coordination. ACM SIGMOD Record, vol. 32, pp. 53–58.

BELLAHSÈNE, Z., ROANTREE, M. (2004a). Querying Distributed Data in a Super-Peer Based Architecture. Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA'04), vol. 3180, pp. 296-305. Zaragova, Espanha.

BELLAHSÈNE, Z., KING, N., & ROANTREE, M. (2004b). Services for Large Scale P2P Networks. European Research Consortium for Informatics and Mathematics News Journal (ERCIM'04)

Disponível em Disponível: http://www.computing.dcu.ie/~nking/ercim04.pdf. Acessado em 10 de agosto de 2010.

BELLAHSÈNE, Z., LAZANITIS, C., McBRIEN, P., & RIZOUPOLOS, N. (2006a). iXPeer: Implementing layers of abstraction in P2P Schema Mapping using AutoMed. Proceedings of the 2nd Workshop on Innovations in Web Infrastructure (IWI 2006). Edinburgh, United Kingdom. Disponível em http://pubs.doc.ic.ac.uk/ixpeer-layers-of-abstraction-p2p/ixpeer-layers-of-abstraction-p2p.pdf. Acessado em 13 de agosto de 2010.

BELLAHSÈNE, Z., LAZANITIS, C., McBRIEN, P., & RIZOUPOLOS, N. (2006b). Querying Distributed Data in a Super-Peer Based Architecture. Proceedings of the 2nd Workshop on Innovations in Web Infrastructure (IWI 2006). Edinburgh, United Kingdom. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.511&rep=rep1&type=pdf. Acessado em 31 de agosto de 2010.

BENEVENTANO, D., BERGAMASCHI, S., FERGNANI, A., GUERRA, F., VINCINI, M., & MONTANARI, D. (2003). A Peer-To-Peer Agent-Based Semantic Search Engine. Proceedings of the 11th Italian Symposium on Advanced Database Systems, pp. 367-378. Cosenza, Italy.

BERGAMASCHI, S., & GUERRA, F. (2002). Peer-to-peer paradigm for a semantic search engine. Proceedings of the 1st international conference on Agents and peer-to-peer computing, pp. 81-86. Bologna, Italy.

BOYD, M., KITTIVORAVITKUL, S., LAZANITIS, C., McBRIEN, P., & RIZOPOULOS, N. (2004). AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. Proceedings of the 6th Advanced Information Systems Engineering Conference (CAiSE), pp. 82-97. Riga, Latvia.

BRITO, G. A. (2005). INTEGRAÇÃO DE OBJETOS DE APRENDIZAGEM NO SISTEMA ROSA - P2P. Dissertação de Mestrado. Instituto Militar de Engenharia. Rio de Janeiro, Brasil.

BRITO, G., & MOURA, A. M. (2005). ROSA-P2P: a Peer-to-Peer System for Learning Objects Integration on the Web. Proceedings of the 11th Brazilian Symposium on Multimedia and the Web (WebMedia'05), pp. 1-9. Poços de Caldas, Brasil.

CARO, G. D., & DORIGO, M. (1998). Antnet: Distributed stigmergetic control for communications network. Journal of Artificial Intelligence Research, vol. 9 ,pp. 317–365.

CIGLARIC, M., & VIDMAR, T. (2006). Ant-inspired query routing performance in dynamic peer-to-peer networks. Parallel and Distributed Processing Symposium, 20th International IEEE Computer Society , pp. 287-292.

COSTA, L. R. (2009). Roteamento de consultas em bancos de dados peer-to-peer utilizando colônias de formigas e ontologias. Dissertação de Mestrado.Universidade Estadual Paulista. São José do Rio Preto, Brasil.

CRUZ, I., XIAO, H., & HSU, F. (2004). Peer-to-peer Semantic Integration of XML and RDF Data Sources. Third International Workshop on Agents and Peer-to-Peer Computing (AP2PC), vol. 3601, pp. 108-119. Heidelberg, Germany.

DORIGO, M., & GAMBARDELLA, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, vol. 1 , pp. 53–66.

DRUSCHEL, P. A. (2001). Past: Persistent and anonymous storage in a peer-to-peer networking environment. Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII) , pp. 65–70.

E-mule. (2010). Acesso em 28 de julho de 2010, disponível em: http://www.emule-project.net/home/perl/general.cgi?l=30

FILLOTTRANI, P. R. (2005). The multi-agent system architeccture in SEWASIE. Journal of Computer Science &Tecnology, vol.5, n° 4 , pp. 225-231.

FIORANO. (2003). Super-Peer Architectures for Distributed Computing. White Paper, Fiorano Software, Inc. Disponível em http://www.fiorano.com/whitepapers/superpeer.pdf.

GHODSI, A. (2006). Distributed k-ary System: Algorithms for Distributed Hash Tables. PhD Thesis. KTH-Royal Institute of Technology. Stockholm, Sweden.

GIANOLLI, P. R., GARZETTI, M., JIANG, L., KEMENTSIETSIDIS, A., KIRINGA, I., MASUD, M., et al. (2005). Data Sharing in the Hyperion Peer Database System. Proceedings of the 31st international conference on Very large data bases, pp. 1291-1294. Trondheim, Norway.

GNUTELLA2. (2010). *Gnutella2 Developer Network*. Acesso em Julho de 2010, disponível em http://g2.trillinux.org

GROOVE. (2010). Acesso em 15 de Julho de 2010, disponível em http://technet.microsoft.com/en-us/ee263742.aspx#tab=1

HALEVY, A. Y., IVES, Z., MADHAVAN, J., MORK, P., SUCIU, D., & TATARINOV, I. (2004). The Piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering* , pp. 787-798.

HALEVY, A., & TATARINOV, I. (2004). Efficient query reformulation in peer-data management systems. Proceedings of the SIGMOD Conference International Conference on Management of Data, pp. 539-550.Paris, France.

HALEVY, A., IVES, Z., MONK, P., & TATARINOV, I. (2003b). Piazza: data management infrastructure for semantic. Proceedings of the 12th World Wide Web Conference (WWW), pp. 556-567. Budapest, Hungary.

HALEVY, A., IVES, Z., SUCIU, D., & TATARINOV, I. (2003a). Schema mediation in peer data management. Proceedings of the 19th IEEE International Conference on Data Engineering.

HALEVY, A., RAJARAMA, A., & ORDILLE, J. (2006). Data Integration: The Teenage Years. Proceedings of the 32nd International Conference on Very large data bases, pp 9-16. Seoul, Korea.

HERSCHEL, S., & HEESE, R. (2005). Humboldt Discoverer: A semantic P2P index for PDMS. Proceedings of the International Workshop Data Integration and the Semantic Web. Porto, Portugal. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.144.503&rep=rep1&type=pdf.

HOSE K., J. A. (2006). An Extensible, Distributed Simulation Environment for Peer Data Management Systems. *Conference on Extending Database Technology, LNCS 3896* , pp. 1198-1202.

JUNIOR, H. C. (2008). Ontologias Emergentes: Uma abordagem para construção de ontologias a partir de mapeamentos ponto-a-ponto. Dissertação de Mestrado. Instituto Militar de Engenharia. Rio de Janeiro, Brasil.

KAMIENSKI, C., SOUTO, E., ROCHA, J., DOMINGUES, M., CALLADO, A., & SADOK, D. (2005). Colaboração na Internet e a Tecnologia Peer-to-Peer. XXV Congresso da Sociedade Brasileira de Computação – SBC2005. São Leopoldo, Brasil.

KAZAA. (2010). Acesso em 14 de Julho de 2010, disponível em http://www.kazaa.com/us/index.htm

LAFFRANCHI, M. M. (2004). Uma solução peer-to-peer para a implantação de jogos multiusuário baseada no padrão emergente MPEG-4 MU. Tese de Doutorado. Universisade Federal de São Carlos. São Carlos, Brasil.

LODI, S., MANDREOLI, F., MARTOGLIA, R., PENZO, W., & SASSATELLI, S. (2008). Semantic Peer: Here are the Neighbors You Want! Proceedings of the 11th International Conference on Extending Database Technology (EDBT '08), pp. 26-37. Nantes, França.

LOEST, S. R. (2007). Um sistema de backup cooperativo tolerante a intrusões baseado em redes P2P. Dissertação de Mestrado. Pontifícia Universidade Católica do Paraná. Curitiba, Brasil.

LUA, E., CROWCROFT, J., PIAS, M., SHARMA, R., & LIM, S. (2004). A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Survey and Tutorial, pp. 72-93.

LV, Q. e. (2002). Search and Replication in Unstructured Peer-to-Peer Networks. Proceedings of the 16° ACM International Conference on Supercomputing(ICS'02), pp. 84-95. New York, United State of America.

MANDREOLI, F., MARTOGLIA, R., PENZO, W., SASSATELLI, S., & VILLANI, G. (2007). SRI@work: Efficient and Effective Routing Strategies in a PDMS. Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE '07), pp. 285-297. Nancy, França.

MANDREOLI, F., MARTOGLIA, R., SASSATELLI, S., & VILLANI, G. (2008). Building a PDMS Infraestrutura for XML Data Sharing with SUNRISE. *Proceedings of the 2008 EDBT workshop on Database technologies for handling XML information on the web*, pp. 51-59. Nantes, France.

MAUROX, P. C., AGARWAL, S., BUDURA, A., HAGHANI, P., & ABERER, K. (2007). Self-Organizing Schema Mappings in the GridVine Peer Data Management System. Proceedings of the 33rd international conference on Very large data bases, pp. 1334-1337. Vienna, Austria.

MICHLMAYR, E. (2006). Ant algorithms for search in unstructured peer-to-peer networks. In: ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops. Washington, DC, USA: IEEE Computer Society. , pp. 142–146.

NAPSTER. (2010). Acesso em 14 de Julho de 2010, disponível em http://www.napster.com

NEJDL, W., SIBERSKI, W., & SINTEK, M. (2003). Design issues and challenges for RDF- and schema-based peer-to-peer systems. ACM SIGMOD Record, vol. 32, pp. 41 - 46.

Ng, W. S., Ooi, B. C., & Tan, K.-L. (2002). BestPeer: A Self-Configurable Peer-to-Peer System. In Proceedings of the 18th International Conference on Data Engineering, p. 272. San Jose, United State of America.

NOWELL, D. L., BALAKRISHNAN, H., & KARGER, D. (2002). Analysis of the Evolution of Peer-to-Peer. Proceedings of the 21st annual symposium on Principles of distributed computing, pp. 233 - 242. Monterey, United State of America.

O'NEIL, E., O'NEIL, P., & WEIKUM, G. (1993). The LRU-K page replacement algorithm for database disk. Proceedings of the 1993 ACM Sigmod Internacional Conference on Management of Data, vol. 22, pp. 297-306.

OOI, B. C., SHU, Y., & TAN, K.-L. (2003). Relational Data Sharing in Peer-based Data Management Systems. ACM SIGMOD Record, vol. 32, pp. 59-64.

OTTO F., M. P. (2007). A Model for Data Management in Peer-to-Peer Systems. International Journal of Computing and ICT Research, ISSN 1996-1065 (online),vol.1, No. 2 ,pp. 67.

PIRES, C. (2009). Ontology-based Clustering in a Peer Data Management System. Tese de Doutorado. Universidade Federal de Pernambuco.Recife, Brasil.

RATNASAMY, S. F. (2001). A scalable content-addressable network. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161-172. San Diego, United State of America.

REZENDE, E. D. (2009). Modelo Estrutural para Compartilhamento de Arquivos Peer-to-Peer. Dissertação de Mestrado. Universidade Estadual Paulista. São José do Rio Preto, Brasil.

SEWAISE. (5 de agosto de 2010). *SEmantic Webs and AgentS in Integrated Economies*. Acesso em 5 de agosto de 2010, disponível em http://www.sewaise.org

SOULSEEK. (2010). *Soulseek - Slsknet - File sharing software*. Acesso em 14 de Julho de 2010, disponível em http://www.slsknet.org

STOICA, I. M. (2001). Chord: A scalable peer-to-peer lookup service. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149-160. San Diego, United State of America.

SUNG, L. G. (2005). A Survey of Data Management in Peer-to-Peer Systems. School of Computer Science, University of Waterloo. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.6553&rep=rep1&type=pdf

SUNRISE. (2010). *ISGroup - Information Systems Group*. Acesso em 26 de agosto de 2010, disponível em http://www.isgroup.unimo.it/sunriseProject.asp

TATARINOV, I., IVES, Z., MADHAVAN, J., HALEVY, A., SUCIU, D., DALVI, N. (2003). The Piazza Peer Data Management Project. ACM SIGMOD Record, vol. 32, pp. 47-52.

TOWSLEY, D. (2003). Peer-peer Networking. Tutorial no SBRC2003, disponível em http://www.cin.ufpe.br/~gprt/gtp2p/tutoriais/p2p03-tutorial_towsley.pdf.

VALDURIEZ, P., & PACITTI, E. (2004). Data Management in Large-scale P2P Systems. *Proceedings of the International Conference on High Performance Computing for Computational Science.* Valencia, Spain. Disponível em http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.4652&rep=rep1&type=pdf

VU, Q. M., LUPU, M., & OOI, B. C. (2010). Peer-to-Peer Computing - Principles and Applications. Editora Springer.

XIAO, H. (2006). *Query processing for heterogeneous data integration.* PhD Thesis. University of Illinois. Chicago, United State of America.

XPEER. (2010). *XPeer Project*. Acesso em 26 de agosto de 2010, disponível em http://www.di.unipi.it/~manghi/XPeerWeb/homexpeer.htm

ZAFALON, G. F. (2009). Algoritmos de alinhamento múltiplo e técnicas de otimização para esses algoritmos utilizando Ant Colony. Dissertação de Mestrado. Universidade Estadual Paulista. São José do Rio Preto, Brasil.

ZHAO, B. K. (Apr. 2001). *Tapestry: An infrastructure for fault-tolerant wide-area location and routing.* Techical Report: CSD-01-1141, Computer Science Division.

ZHAO, J. (2006). *Schema Mediation and Query Processing in Peer Data Management Systems.* Master Thesis. University of British Columbia. Kelowna, Canada.