Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Mestrado em Engenharia Informática

Collaborative Web Applications

Tiago João dos Santos Alberto Garcia (aluno nº 27262)

2º Semestre de 2009/10

28 de Julho de 2010

Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Collaborative Web Applications

Tiago João dos Santos Alberto Garcia (aluno nº 27262)

Orientador: Prof. Doutor Nuno Preguiça
Co-orientador: Prof. Doutor Sérgio Duarte

Trabalho apresentado no âmbito do Mestrado em Engenharia Informática, como requisito parcial para obtenção do grau de Mestre em Engenharia Informática.

2º Semestre de 2009/10
28 de Julho de 2010

Agradecimentos

Quero agradecer, em primeiro lugar, ao Professor Doutor Nuno Preguiça e ao Professor Doutor Sérgio Duarte por todo o conhecimento e apoio transmitidos durante a toda a elaboração desta dissertação, sem os quais a conclusão com sucesso deste trabalho não teria sido possível. Um agradecimento também a todos os professores e colegas do departamento que, de uma forma ou de outra, contribuíram com o seu conhecimento e dedicação para a minha formação académica e pessoal ao longo dos últimos anos.

A todos os colegas, de curso, de trabalho, de danças e de vida, que me apoiaram durante os dias maus, e com quem partilhei os dias bons, um grande abraço. O apoio de todos vós foi indispensável. Não podendo nomear todos, ficam os que mais me marcaram ao longo dos anos: Jorge Graça, Maria João Paiva, Fátima Bernardes, João Águas, Vanessa Violante, Carla Violante, Jessica Fernandez, Sónia Garcias e Laura Gouveia.

Um agradecimento especial àqueles amigos que, nem sempre estando fisicamente por perto, estão sempre comigo. Yann, Phillip G., Tom, Amaia, Philippe S., Sónia, Manuela, Haruki, Eduardo & Cristoph: Thank you.

Aos meus pais, João e Anabela Garcia, um muito obrigado pelo incondicional apoio ao longo destes 25 anos. Um 'miado' para a minha gata, eterna e paciente companheira.

Resumo

O crescimento da *web* nos últimos anos tem levado a um significativo aumento da informação disponível *online*. No sentido de se manterem informados, os utilizadores podem recorrer a mecanismos de disseminação de eventos, que lhes permitem ser informados de actualizações que ocorram nos *sites* da sua preferência. Um dos protocolos desenvolvidos neste sentido é o RSS.

A disseminação de eventos RSS é feita de forma centralizada. Cada utilizador consulta, de forma automática, através de uma aplicação específica para o efeito, a fonte da informação para ser informado sobre alterações à mesma.

Este tipo de aproximação pode ser estendida através da colaboração entre múltiplos utilizadores na divulgação da informação. Isto permite tornar o processo de disseminação de notícias mais eficiente, ao mesmo tempo que permite acrescentar funcionalidades relevantes ao protocolo RSS.

Esta dissertação apresenta um sistema que permite aos subscritores de *feeds* RSS colaborarem na sua disseminação, bem como cooperarem entre si para divulgar novas *feeds*. Tal permite aos utilizadores ter acesso a novas fontes de informação sugeridas com base nas suas preferências pessoais.

Cada vez mais, existe o interesse por parte dos utilizadores em receberem sugestões de conteúdos. Sites que disponibilizam esse serviço têm surgido nos últimos anos, tendo por base uma componente centralizada que gere as sugestões a fornecer aos utilizadores. Essa mesma funcionalidade pode ser suportada num sistema de disseminação colaborativa de *feeds* RSS, pelo que o sistema desenvolvido inclui mecanismos que permitem aos utilizadores colaborarem na geração de sugestões de conteúdos.

Palavras-chave: RSS, disseminação colaborativa, P2P, recomendação de conteúdos

Abstract

Over the last few years, the growth of the web has allowed more and more information to be posted online. In order to be kept up to date about their favorite sites, users may rely on event dissemination services. RSS is one of the most used protocols for that purpose.

Event dissemination using RSS is based on a centralized model. Each user must have an application that periodically polls the information's source, in order to be informed about updates.

This kind of approach can be extended to allow collaboration between different subscribers of RSS feeds in the information propagation process. This allows for the dissemination process to be more efficient and, at the same time, allowing new and interesting features to be added.

In this dissertion, we present a system that will allow RSS users to cooperate in feed disseminations, as well as collaborate in content rating and suggestion. The goal is to allow users to have faster and easier access to more relevant feeds sources.

User interest in automatic content suggestions has grown in the last few years, leading to the creation of several sites providing that service. These services have a centralized architecture, where all users get their suggestions from a single server. A similar service can be deployed in a collaborative-based RSS feeds dissemination system. The system presented in this work lets users share information directly with each other, allowing the system to provide feeds suggestions.

Keywords: RSS, collaborative dissemination, P2P, content recomendation

Conteúdo

1. Introdução

1.1 Introdução geral

A Internet é, cada vez mais, um meio importante de troca de informação. Criada com o intuito de permitir a troca de dados entre computadores, tornou-se um dos principais meios de comunicação entre pessoas.

Durante os últimos anos assistimos à massificação do acesso à Internet. Com o crescente número de utilizadores, surge também um crescente número de interesses a serem explorados na *web*. Consequentemente, a oferta de informação é cada vez maior, não só em diversidade, mas também em quantidade.

Durante a evolução da Internet foram desenvolvidas tecnologias que permitem ao utilizador comum colocar online informação que pretende partilhar com os restantes utilizadores da rede. Exemplos disso são os sistemas de *newsgroups*, *wikis* e, mais recentemente, blogues e *sites* de redes sociais. Estas fontes de informação vieram complementar as fontes de informação já existentes, como *sites* institucionais, de empresas ou organizações.

Com o crescimento do número de fontes de informação surgiu o problema de permitir ao utilizador manter-se actualizado. Para dar resposta a esse problema, foram desenvolvidos protocolos que permitem que um utilizador se mantenha informado relativamente a actualizações que ocorram aos conteúdos disponibilizados, por exemplo, num dado *site*. Ao subscrever esse serviço, o utilizador final recebe resumos das actualizações ocorridas, mantendo-se informado sobre conteúdos disponibilizados pelo autor dos mesmos. Estes sistemas facilitam o acesso à informação, permitindo que esta chegue automaticamente até ao utilizador, pouco tempo após ser divulgada.

Neste contexto, a escolha de fontes de informação passa a ter um papel crucial. Os utilizadores querem fontes com informação sobre os tópicos do seu interesse. A abundância de informação na Internet torna este processo de selecção complexo. Para encontrar informação sobre todos os seus tópicos de interesse um utilizador deve investigar uma grande quantidade de fontes, descartando grande parte das mesmas. Este processo é moroso e incómodo. Por outro lado, uma pesquisa rápida levará a que apenas parte dos tópicos de interesse do utilizador sejam cobertos.

Assim, torna-se útil desenvolver um sistema que permita aos utilizadores aceder rapidamente à informação relevante sobre os seus tópicos de interesse, independentemente da proveniência da mesma. Em particular, seria útil que esse sistema permitisse que os utilizadores com gostos semelhantes colaborassem entre si para determinar quais as fontes mais pertinentes. Como tal, passa a ser importante incluir neste sistema mecanismos de classificação colaborativa de conteúdos, e sugestão de fontes e informações com base nas semelhanças nas preferências e gostos dos utilizadores.

1.2 Motivação

O protocolo RSS [6] permite divulgar eventos (ex. alterações a um site ou blogue), permitindo aos seus utilizadores manterem-se informados. Este protocolo tem por base um modelo editor/assinante (*publish/subscribe*), que se caracteriza por permitir que diversas fontes de informação comuniquem, de forma assíncrona, um conjunto de eventos aos seus subscritores. Uma das características deste modelo é que não implica necessariamente que subscritores e editores se conheçam. Ambos podem apenas conhecer uma entidade intermédia, denominada *broker*, que fica responsável por receber informação dos editores e por a disponibilizar/encaminhar para os subscritores.

As subscrições de *feeds* RSS são geridas por aplicações específicas, que consultam regularmente os provedores desse serviço, obtendo deles informações. Em particular, no caso do protocolo RSS, os editores actuam em simultâneo como *brokers*, e fornecem informações sobre actualizações aos conteúdos disponibilizados.

Este sistema permite aos utilizadores manterem-se actualizados relativamente a um conjunto de fontes de informação por eles seleccionadas. No entanto, não possibilita directamente que diversos utilizadores se conheçam ou troquem informação entre si. Como tal, não possibilita que os utilizadores partilhem informação sobre as suas fontes e a relevância e interesse das mesmas.

Assim, seria relevante e útil permitir que, de algum modo, os utilizadores trocassem informação entre si (de forma implícita ou explícita) de modo a possibilitar o acesso a novas fontes de informação, sugeridas implicitamente por outros utilizadores, com base nas semelhanças entre os seus gostos pessoais e tópicos de interesse.

Sistemas com este tipo de funcionalidades já existem, mas com objectivos distintos. Alguns exemplos disso são o StumbleUpon (www.stumbleupon.com), Reddit (www.reddit.com) e Digg (www.digg.com), que permitem aos utilizadores classificarem *sites* com base na sua opinião pessoal sobre os mesmos. Essas classificações são então compiladas pelo provedor de cada um destes serviços, e são oferecidas sugestões aos utilizadores com base nos *sites* já visitados por eles. Estes sistemas destinam-se a partilhar e divulgar páginas *web* tradicionais, não se aplicando directamente a *feeds* RSS.

Assim, tornar-se-ia relevante a existência de um sistema capaz de permitir aos utilizadores colaborarem entre si para classificarem e partilharem essas classificações sobre *feeds* RSS. A principal funcionalidade deste sistema será a componente cooperativa, onde os utilizadores comunicarão entre si (de forma directa ou indirecta) para partilhar informação sobre as *feeds*.

A plataforma LiveFeeds [1] é um sistema que aborda uma questão semelhante. Nesta plataforma, os utilizadores comunicam entre si, de modo a poderem disseminar as *feeds*, sem que todos os utilizadores tenham de recorrer directamente aos provedores das mesmas. Adicionalmente, esta plataforma permite que essa disseminação seja feita de forma filtrada, ou seja, a própria plataforma incorpora mecanismos de filtragem dos conteúdos disseminados, com base nas preferências dos utilizadores. Fica assim definida uma potencial base para o sistema que se pretende desenvolver.

Embora permita a disseminação colaborativa de *feeds*, esta plataforma por si só não acrescenta novas funcionalidades ao processo de subscrição e acesso a conteúdos. Tal como foi apontado anteriormente, a crescente quantidade de informação disponível na Internet criou no utilizador final a necessidade de despender mais tempo na sua procura e filtragem, de modo a manter-se informado. Por outro lado, os sistemas tradicionais de disseminação de informação não oferecem qualquer resposta a esse problema. Assim, surge a oportunidade de complementar as funcionalidades oferecidas pelas *feeds* RSS com mecanismos de recomendação de conteúdos.

Este tipo de mecanismos, denominados Recommender Systems [12], são utilizados em diversas aplicações práticas. A sua utilização mais conhecida é nos sistemas de lojas *online*, onde são propostos ao utilizador um conjunto de novos produtos com base na navegação efectuada. Na sua essência, estas ferramentas pretendem sugerir ao utilizador final um conjunto de elementos do seu potencial interesse. Na literatura [12] foram propostas diversas abordagens para obter este tipo de recomendações. Por exemplo, nas lojas *online* as sugestões tendem a ser efectuadas com base no histórico de compras: a um utilizador são sugeridos livros adquiridos por outros utilizadores com históricos de compras que se intersectam. Em geral, a abordagem a adoptar varia consoante a natureza dos elementos a sugerir, a informação automaticamente disponível ou inferível sobre os mesmos e a informação disponível ou inferível sobre o utilizador do sistema. No cenário que se considera neste trabalho, poder-se-iam sugerir recomendações com base na informação acedida anteriormente, ou de utilizadores que acederam ao mesmo tipo de informação

Assim, torna-se vital obter informações sobre os utilizadores, de modo a tornar possível a geração de sugestões fiáveis. Existem várias abordagens a este problema [4] [10]. Note-se que, em todas elas, foi feito um significativo esforço para tornar o sistema o mais automático possível, requerendo o mínimo de alterações possíveis ao comportamento do utilizador. Assim, pretende-se obter o máximo de informação com o mínimo de esforço possível. Sistemas que forcem o utilizador a fornecer informação explicitamente são frequentemente pouco adoptados/abandonados [10].

Para suportar a recolha de informação automática é necessário que exista um modo de permitir ao utilizador consultar os conteúdos a apresentar, ao mesmo tempo que se recolhem as informações necessárias à geração de sugestões. Existem algumas plataformas capazes de suportar essas funcionalidades [2] [4] [10]. Sendo capazes de apresentar ao utilizador as informações que este pretende consultar, estas plataforma permitem que, sobre elas, sejam implementados directamente mecanismos de recolha de informação.

1.3 Solução Apresentada

O trabalho desenvolvido consiste um sistema cooperativo para melhorar o acesso à informação disponível na *web*, em particular a *feeds* RSS. Este sistema cooperativo permite tornar o mecanismo de disseminação de notícias mais eficiente, dado que permite aos utilizadores colaborarem

para difundirem as actualizações entre si.

Este mecanismo de disseminação é complementado com ferramentas de selecção e sugestão de informação relevante. Tal permite aos utilizadores receberem informações sobre novas fontes de informação, determinadas como sendo de potencial interesse, com base nos seus gostos pessoais.

Este sistema opera com um *proxy web*, construído usando o sistema Comanche [2]. Esta plataforma permite executar localmente *Java Web Applications*, capazes de efectuar operações de monitorização e transformação sobre pedidos e respostas trocadas entre o *browser* e os servidores *web* a que este acede.

Para permitir aos utilizadores colaborarem na disseminação de conteúdos, recorreu-se ao LiveFeeds, que permite a disseminação de *feeds* RSS de forma colaborativa e filtrada. Recorrendo a estas funcionalidades, é possível fazer com que as actualizações aos conteúdos RSS sejam trocadas entre os diversos membros da rede, deixando de ser necessário recorrer ao servidor a cada verificação dos conteúdos. Dado que o LiveFeeds dispõe de informações sobre todos os utilizadores do sistema, poderá ainda ser utilizado para facilitar a comunicação entre os módulos de Tratamento de Sugestões presentes em cada instância da plataforma.

A interacção com o utilizador é feita de modo transparente, não implicando que este altere os seus hábitos de utilização da *web* ou de consulta de *feeds*. A informação apresentada pode ser consultada em qualquer aplicação que suporte a visualização de conteúdos RSS, bastando para tal alterar as configurações da mesma.

Existe a necessidade de recolher informação sobre as preferências dos utilizadores. Um caso particular disto é a necessidade que existe em determinar se um utilizador considerou uma *feed* relevante ou não. De modo a tornar a plataforma o mais atraente possível ao utilizador, tal é feito de forma automática e transparente, dado que os utilizadores nem sempre estão dispostos a fornecer este tipo de informações explicitamente.

Para garantir que essa informação é recolhida de forma automática, o módulo de extracção de informação utiliza a capacidade do Comanche de interceptar os dados resultantes da navegação do utilizador na *web*. A partir da intercepção desses dados é possível inferir quais as *feeds* de interesse do utilizador, de um modo cómodo e transparente para este.

Adicionalmente é possível, com base na informação recolhida sobre o utilizador e os seus interesses, gerar sugestões de conteúdos. Tal é feito integrando um componente responsável por utilizar a informação recolhida sobre os utilizadores como base para trocar sugestões entre as diversas instâncias da plataforma. Este componente integra um algoritmo de recomendação (*recommender system*), assim como primitivas de comunicação entre elementos da rede, de modo a permitir a partilhar da informação que é utilizada como base para a geração destas sugestões. Esta componente de comunicação recorre ao LiveFeeds, dado que este permite comunicar com qualquer elemento presente na rede.

1.4 Principais Contribuições Previstas

A principal contribuição prevista deste trabalho consiste no desenho e implementação de um sistema que permita aos subscritores de *feeds* RSS colaborarem na sua disseminação. Neste contexto, a disseminação cooperativa deste tipo de conteúdos pretende ser uma alternativa viável à consulta directa por parte de todos os utilizadores aos servidores responsáveis por alojar os conteúdos. A principal vantagem desta abordagem é a optimização do processo de divulgação deste tipo de conteúdos. É ainda relevante referir que será possível, através da utilização desta plataforma, reduzir a carga sobre os servidores *web* que disponibilizam *feeds* RSS, bem como diminuir o tempo de acesso por parte dos utilizadores finais aos conteúdos desta forma divulgados.

O sistema Comanche [2], que servirá de base a este trabalho, actua como um *proxy web* capaz de analisar e manipular os pedidos e respostas gerados durante a navegação na rede. Estas funcionalidades permitem, não só identificar de forma automática e transparente as preferências do utilizador, como também apresentar conteúdos *web* resultantes da troca cooperativa de informação entre os subscritores de *feeds*. As funcionalidades do Comanche permitirão ainda que a interacção com ColFeeds ocorra através dos clientes RSS já existentes, não necessitando modificações aos mesmos ou de alterações ao comportamento do utilizador final para operar.

De forma a comunicarem entre si, as diversas instâncias do ColFeeds recorrerão ao sistema de disseminação de *feeds* LiveFeeds [1]. Este sistema permitirá a troca de conteúdos RSS entre os utilizadores, assim como a divulgação de sugestões de novos conteúdos relevantes, determinados com base nas preferências do utilizador.

1.5 Estrutura do Documento

Após este capítulo em que se apresenta o contexto deste trabalho e se descreveu, de forma sucinta, a solução desenhada e implementação desenvolvida, este documento prossegue com o capítulo 2, se analisa o estado da arte. O capítulo 3 apresenta, em detalhe, as plataformas e tecnologias que serviram de base a esta implementação. No capítulo 4 é apresentado o desenho da solução, discutindo-se a arquitectura escolhida. No capítulo 5 são analisados os detalhes relevantes da implementação desenvolvida como parte deste trabalho. O capítulo 6 apresenta os resultados das avaliações qualitativas e quantitativas a este trabalho, sendo as conclusões desta dissertação apresentadas no capítulo 7.

2. Trabalho relacionado

Neste capítulo iremos analisar trabalho relacionado com os diversos aspectos do trabalho a desenvolver. Serão apresentadas trabalhos que oferecem resposta a alguns dos problemas identificados no decorrer da análise do problema em estudo.

2.1 Propagação de Informação

Um dos problemas chave deste trabalho é o modo como os diversos elementos comunicam entre si. Dada a natureza colaborativa da solução proposta, uma abordagem descentralizada, como aquela oferecida pelos sistemas *peer to peer* [11], poderá dar resposta aos principais desafios: tolerância a falhas, escalabilidade, ausência de infraestrutura centralizada e não alteração dos protocolos e infraestruturas em utilização.

2.1.1 Sistemas P2P (*peer to peer*)

Os serviços organizados segundo o modelo *peer to peer* têm como principal característica a inexistência de uma infraestrutura centralizada para manter o serviço disponibilizado. Em vez disso, os vários componentes comunicam entre si, disponibilizando serviços, alternando nos papeis de cliente e servidor.

Os sistemas baseados em *peer to peer* têm associados a si um conjunto de características e problemas próprios. Os elementos que os compõem são típicamente pobres em recursos (CPU, memória, disco, etc), quando comparados com os servidores dedicados utilizados no modelo cliente/servidor tradicional. Isto leva a que um bom sistema procure/deva garantir que a carga computacional seja distribuída justamente por entre os participantes. Garantias de que o serviço opera correctamente sem sobrecarregar os elementos que o compõem são essenciais para a boa adopção e funcionamento de soluções baseadas neste tipo de arquitecturas.

Adicionalmente, os elementos que compõem uma rede deste tipo não são equipamentos dedicados. Tipicamente, os utilizadores não estão ligados em permanência à rede, pelo que o sistema deve ser capaz de lidar com a entrada e saída frequente de elementos da rede.

Dado que não existe um servidor central que disponibiliza o serviço e actua como ponto de encontro de todos os elementos da rede, cada nó deve manter informação suficiente para poder comunicar com os restantes elementos da rede. A topologia da rede sobreposta e o modo como é formada determinará algumas das características chave de um sistema P2P, como, por exemplo, o custo do encaminhamento ou a rapidez com que é possível realizar pesquisas no sistema.

Consequentemente, o tipo de organização adoptada permite classificar as redes *peer to peer* de uma de duas formas:

- Redes sobrepostas não estruturadas [11]: Os nós conhecem um conjunto aleatório de

outros nós, e comunicam com os restantes através destes. Isto leva a que se formem grafos aleatórios, dado que não são usados identificadores lógicos para guiar a formação de uma topologia em particular;

- Redes sobrepostas estruturadas [11]: Cada elemento da rede assume um identificador lógico único. Com base nesse identificador, existem regras que guiam a selecção dos parceiros a que um nó se deve ligar. Esta organização pretende introduzir determinismo na topologia, de modo a formar uma estrutura organizada entre os elementos que a compõem. Tal é feito de modo a tornar o encaminhamento potencialmente mais eficaz, bem como atribuir à rede um conjunto de características que facilitam a implementação de funcionalidades sobre estas. É de realçar que, na maior parte dos casos, a estrutura criada entre os nós nada tem a ver com a organização física da rede real subjacente.

Esta última característica é a causadora de uma das principais fraquezas apontadas dos sistemas *peer to peer*, independentemente da natureza da estrutura adoptada pela sua rede. Dado que as ligações criadas entre os nós em nada se relacionam com a sua proximidade geográfica, estes sistemas tendem a sofrer de uma elevada latência entre nós.

Outra das características negativas deste tipo de arquitecturas é a necessidade de despender recursos para lidar com a entrada e saída frequente de nós do sistema. Esta questão é de particular relevância nas redes sobrepostas estruturadas, devido à necessidade de reparar continuamente a rede sobreposta, de modo a não perder o determinismo subjacente e as características que nele assentam.

2.1.2 *Distributed Hash Table*

As *Distributed Hash Table*, ou DHTs, são um tipo de rede sobreposta estruturada. Esta estrutura destina-se a operar como um repositório distribuído de dados. Estes dados têm como chave identificadores do mesmo domínio dos usados para organizar a rede sobreposta onde são armazenados. Isto facilita o acesso aos mesmos, dado que permite determinar e alcançar rapidamente qual o nó responsável por armazenar um certo conjunto de dados, e aceder ao mesmo com custo limitado. Tal facilita o acesso aos dados, bem como pesquisas com algumas características determinísticas. Estas estruturas implementam ainda mecanismos de replicação de dados, de modo a permitir que a saída de nós do sistema não afecte a disponibilidade dos dados.

Cada nó assume um identificador único num espaço N, o que lhe permite determinar uma posição relativa em relação aos restantes e, consequentemente, uma noção de distância. Tipicamente, os nós mantêm uma tabela de encaminhamento para nós distantes, que lhes permite encaminhar com rapidez dados para nós em posições afastadas da rede sobreposta. Adicionalmente, podem manter tabelas onde guardam o endereço de nós que estão próximos de si, para permitir dar resposta a problemas de replicação de dados e tolerância a falhas. O tamanho e organização destas tabelas variam com as diferentes soluções.

2.1.2.1 Chord

O Chord [16] é uma DHT onde os nós se organizam numa estrutura circular. Cada nó deve conhecer, no mínimo, o seu sucessor, sendo isto suficiente para garantir que o encaminhamento é feito com custo linear. Cada nó mantém ainda uma tabela de encaminhamento para grandes saltos. Esta tabela, de dimensão $log(N)$, é preenchida segundo a seguinte fórmula: $n+2^{i-1}$, para todo o i tal que: $1 < i < k$ (sendo n o identificador do nó corrente e k o tamanho do espaço de identificadores, em bits. Caso não exista um nó com o identificar determinado, é guardado o endereço do nó existente seguinte. Esta tabela de grandes saltos permite que o custo de encaminhamento seja melhorado, passando de linear para $O(log(N))$.

2.1.2.2 Pastry

O Pastry [14] é um outro tipo de DHT. Embora, tal como o Chord, tenha custo de encaminhamento $log(N)$ com tabelas de tamanho $log(N)$, esta rede sobreposta analisa e tem em conta outras métricas para melhorar o seu desempenho. A mais frequentemente utilizada é a da latência entre nós. Assim, cada nó mantém 3 tabelas de encaminhamento:

- Tabela de Vizinhos: com V apontadores para nós vizinhos segundo uma dada métrica (por exemplo, nós com baixa latência em relação ao corrente);
- Tabela de Encaminhamento: Tabela onde figuram nós cujos identificadores têm prefixos (em base B) com determinadas propriedades. Esta tabela, de dimensão $logB(N)$ linhas e $B-1$ colunas, mantém um conjunto de nós numericamente distantes do actual. A primeira linha mantém identificadores onde o primeiro dígito varia em relação ao do identificador do nó corrente. A segunda linha contém identificadores onde o primeiro dígito é igual ao do nó actual, e o 2º difere. As restantes linhas são preenchidas de forma análoga. Note-se que esta tabela pode incluir nós retirados da tabela de vizinho, logo serão nós com baixa latência em relação ao actual;
- Tabela de Folhas: com F apontadores para nós numericamente adjacentes ($F/2$ nós numericamente menores e $F/2$ numericamente superiores), permitindo encaminhamento nós próximos.

Ao encaminhar dados, o algoritmo analisa o endereço de destino. Se for próximo ao do nó corrente e estiver contido na tabela de folhas, esta é usada para encaminhar a informação. Se tal não se verificar, utiliza a tabela de encaminhamento para enviar a informação para o nó com o prefixo mais semelhante ao do destinatário. Se ambos os passos anteriores não forem aplicáveis, a informação é encaminhada para o nó numericamente mais próximo do destino. O encaminhamento termina quando deixa de ser possível reduzir a distância ao destino.

Esta estratégia de encaminhamento, embora tenha a mesma complexidade da do Chord, tende a ser mais eficaz, dado que as tabelas são construídas preferencialmente com nós para os quais se tem baixa latência na comunicação.

2.1.2.3 DHTs 1-hop e 2-hop

As DHTs *1-hop* [13] e *2-hop* [7] mantêm uma tabela de encaminhamento maior (respectivamente de tamanho N e $N^{\frac{1}{2}}$, em que N é o número total de nós presentes no sistema) de modo a encurtarem a distância a percorrer no encaminhamento.

As *1-Hop* e *2-Hop* representam uma abordagem distinta ao problema do encaminhamento em DHTs. Utilizando maior quantidade de recursos (nomeadamente, memória para armazenar as tabelas de encaminhamento e capacidade de comunicação para as adquirir e manter), estas estruturas conseguem reduzir drasticamente a distancia percorrida no encaminhamento. Ao passo que as DHTs anteriormente apresentadas têm custo $log(N)$ de encaminhamento, estas têm custo constante. No caso da *2-Hop*, o custo de encaminhamento é 2. No caso da *1-Hop*, o custo de encaminhamento é 1, ou seja, é utilizado o melhor caminho oferecido pela camada de rede na qual assenta a rede sobreposta.

2.1.3 Sistemas de Propagação de Informação *Web* Usando Sistemas P2P

O Squirrel [8] é um sistema de *caching* HTTP, que permite disponibilizar aos seus utilizadores o acesso a uma cache partilhada, sem ser necessário recorrer a hardware dedicado para tal.

Desenvolvido para ser utilizado em redes locais, onde a largura de banda entre os diversos *hosts* é elevada quando comparada com a largura de banda de acesso à *web*, este sistema permite que os utilizadores partilhem a sua *cache web* entre si.

A aplicação é instalada no computador de cada utilizador final, e actua como um *proxy*, interceptando os pedidos realizados e as respectivas respostas. As várias instâncias da aplicação em execução numa rede local interligam-se entre si, através de uma DHT Pastry.

Todos os pedidos feitos são interceptados pela *proxy* local. Se a informação pedida existir na *cache* desse nó, a resposta é imediata. Caso contrário, e com base no endereço do pedido, o Squirrel determina qual o nó da rede responsável por guardar a página em *cache*. A esse nó dá-se o nome de "*home node*". O "*home node*" poderá operar num de dois modos distintos:

- Home store: O "*home node*" guarda localmente as páginas pelas quais é responsável. Ao receber um pedido, responde imediatamente se tiver os dados em *cache*. Caso contrário, encaminha o pedido para o servidor, guarda a resposta, e encaminha-a para o nó que originou o pedido;

- Directory: O "*home node*" guarda uma lista de nós que têm em *cache* a página pedida. Ao receber um pedido, verifica se conhece algum nó que tenha essa página em *cache*, e requisita o conteúdo a esse nó. Se não conhecer, encaminha o pedido para o servidor, enviando a resposta para o nó que o originou. Neste processo, guarda a informação da existência de um nó com esse conteúdo.

Análise Crítica Embora se destine a um ambiente de execução distinto do alvo deste trabalho, o Squirrel é uma abordagem possível à partilha de conteúdos usando um modelo *peer to peer*. É

de evidenciar o seu modelo de indexação de conteúdos, que poderá ser relevante para o trabalho em curso.

O Squirrel apresenta a grande limitação de só ser viável em redes locais, onde a velocidade de ligação entre *hosts* é muito superior à de acesso à internet. O facto de a cache ser partilhada por diversos utilizadores através de uma DHT Pastry poderá ainda levar a que ocorram problemas de latência acumulada na comunicação entre os diversos elementos da rede. Uma possível solução para este problema seria a utilização de DHTs *1-Hop* ou *2-Hop*. A partilha de dados pode ainda introduzir problemas a nível de segurança e privacidade, dado que cada instância da plataforma fica a conhecer quem acedeu às páginas para as quais foi designado como "*home node*".

2.2 Interacção com o utilizador

Pretende-se que o sistema a desenvolver obtenha e forneça ao utilizador final o máximo de informação possível, requerendo o mínimo de esforço. Assim, responder a este requisito de forma óptima tornará a plataforma apelativa ao utilizador final, potenciando o seu crescimento e consequente melhoria de desempenho.

2.2.1 UsaProxy

A plataforma UsaProxy [4] permite monitorizar, com elevado nível de detalhe, a utilização de uma página *web* por parte de um utilizador final, sem necessitar de alterações quer no cliente, quer no servidor.

Tal é conseguido recorrendo a um *proxy* entre cliente e servidor. Ao ser encaminhada para o cliente, a página passa pelo *proxy* que nela insere automaticamente código JavaScript. Este código destina-se a monitorizar alguns parâmetros de utilização da página:

- Eventos "*load*" e "*resize*" da página, bem como a largura e altura da janela do browser;
- Eventos "*focus*", "*blur*" e "*unload*";
- Eventos de rato, como "*click*" e "*hover*", propriedades que permitem identificar sobre que objecto da página tal ocorreu, bem como as coordenadas do rato;
- Movimentos do rato;
- Eventos de *scrolling*, assim como a posição da barra de *scroll*;
- Teclas pressionadas.

De modo a reduzir o volume de tráfego gerado na rede, a informação obtida é modificada, de modo a ser transmitida informação suficiente para ser possível reconstruir os passos do utilizador, sem que se transmita toda e qualquer alteração ao estado da navegação (por exemplo, apenas se monitorizam modificações significativas da posição da barra de scroll ou do rato).

O processo de inserção deste código de monitorização é automático, pelo que não requer a intervenção do programador. Por outro lado, o utilizador final não tem de instalar qualquer software adicional no seu computador, desde que tenha JavaScript activo. Isto permite obter dados sobre a interacção de um utilizador com uma página, de forma totalmente transparente e automática.

Os dados obtidos podem ser analisados de dois modos: individualmente ou em grupo. À luz do trabalho a desenvolver, ambas as abordagens podem ser relevantes:

- Analisar os dados individuais pode ser útil para inferir as preferências de um dado utilizador;
- Uma análise do comportamento de um grupo é relevante para inferir, na globalidade da página, quais as secções de maior e menor interesse.

Análise Crítica O UsaProxy define um modo de obter implícita e automaticamente informação a partir da simples utilização de uma página. Este tipo de abordagem é de extrema relevância, dado que permite inferir informação relativa à opinião de um utilizador sobre uma página (ou secção da mesma) sem que este tenha de efectuar qualquer acção. No entanto, esta abordagem é propícia a erros nos resultados. Um utilizador que, por exemplo, deixe uma janela aberta e abandone o computador poderá ser identificado como tendo uma preferência por aquele conteúdo, quando tal não se verifica.

2.2.2 AjaxScope

O AjaxScope [9] é uma plataforma que permite monitorizar a execução do código JavaScript contido numa página *web*, inserindo neste instruções de monitorização.

Este sistema tem por base um *proxy web*, que intercepta as comunicações entre cliente e servidor. Ao encaminhar uma página do servidor para o cliente, a plataforma modifica dinâmica e automaticamente o código JavaScript nela contido, acrescentando-lhe primitivas de monitorização (detecção de tempos de execução de excertos de código, detecção de ciclos infinitos, detecção de *memory leaks*, etc).

O código colocado na página pelo AjaxScope envia periodicamente *logs* do cliente para o *proxy*, permitindo aos programadores obter informações que lhes permitam melhorar o desempenho do código ou corrigir bugs.

Ao permitir manipular o código dinamicamente, é possível distribuir um alargado conjunto de testes por inúmeros utilizadores, gerando resultados mais fiáveis, ao mesmo tempo que reduz o esforço exigido de cada utilizador. Outra grande vantagem deste sistema é permitir que a monitorização seja feita sem que os programadores tenham de inserir essa funcionalidade manualmente.

Análise Crítica Esta plataforma demonstra como é possível injectar código JavaScript em páginas HTML no sentido de obter informações sobre o comportamento das mesmas. Este

paradigma pode ser estendido de forma a permitir detectar comportamentos do utilizador face à página apresentada, sendo útil para extrair informação sobre as suas preferências ou opinião sobre os conteúdos que visualiza.

2.3 Gestão e Classificação de conteúdos

2.3.1 *Recommender Systems*

Os sistemas de recomendação [12], ou *recommender systems*, tem sido uma importante área de investigação ao longo dos últimos anos. Estes mecanismos são utilizados para sugerir a um utilizador um conjunto de elementos a ele desconhecidos, com base em informação obtida sobre este. Este tipo de sistemas podem ser encontrados em lojas online, sites de notícias, entre outros, sob a forma de sugestões de produtos potencialmente interessantes, notícias relevantes, etc.

A operação destes sistemas tem por base informação obtida sobre o utilizador. Assumindo um exemplo, a informação recolhida poderá ser as classificações atribuídas por um utilizador a um conjunto de filmes. Com base nas classificações obtidas a partir deste e dos restantes utilizadores, estes sistemas devem ser capazes de estimar níveis de relevância para um utilizador de conteúdos não classificados por ele. Os *recommender systems* podem ser agrupados numa de três possíveis categorias, consoante o modo como estimam este tipo de classificação:

- *Content-Based*: as recomendações são elaboradas com base em semelhanças com produtos já classificados pelo utilizador;
- *Collaborative-Based*: as recomendações são elaboradas com base em classificações atribuídas por utilizadores com preferências semelhantes;
- Soluções híbridas: combinam ambos os métodos descritos anteriormente.

Content-Based Estes sistemas baseiam-se unicamente na informação extraída dos diversos elementos a sugerir e nas preferências do utilizador que consulta o sistema. O sistema extraí um conjunto de informações sobre os elementos a sugerir, de modo a poder inferir níveis de semelhança entre eles. Exemplificando, ao analisar um filme, o sistema poderá ter em conta o realizador, actores ou género de filme para determinar parecenças entre filmes distintos. Posteriormente, sugere ao utilizador elementos a ele desconhecidos que sejam semelhantes a elementos classificados como desejáveis. Recuperando o exemplo acima apresentado, poderá sugerir filmes do mesmo realizador, com os mesmos actores, etc.

Este tipo de sistemas requer a existência de informação sobre os elementos a recomendar. Idealmente, esse tipo de informação deve ser extraída automaticamente. No entanto, tal poderá não ser possível. No exemplo anterior, este tipo de sistemas é aplicável a sugestões sobre filmes, desde que o sistema tenha acesso à informação sobre os filmes (realizador, actores, etc). No

entanto, o esforço de inserção dessa informação no sistema pode tornar a sua implementação inviável.

Um outro problema deste tipo de sistemas é que estes apenas recomendarão elementos semelhantes aos já classificados pelo utilizador, independentemente da qualidade dos restantes elementos. Por exemplo, um utilizador que apenas classifique comédias dificilmente receberia como sugestão um drama, independentemente da qualidade deste. Assim, alguns destes sistemas incorporam mecanismos com factores aleatórios, de modo a sugerirem conteúdos com alguma diversidade.

Por outro lado, o utilizador poderá receber recomendações de filmes muito semelhantes aos já vistos, o que nem sempre será desejável. Em alguns sistemas tal é prevenido excluindo da lista de recomendações elementos que sejam demasiado semelhantes.

Novos utilizadores do sistema poderão ainda ter dificuldades em obter sugestões viáveis, dado que a eficácia do sistema é proporcional ao número de classificações atribuídas pelo utilizador.

Collaborative-Based Uma outra abordagem é a utilização de sistemas collaborative-based. Estes sistemas não se baseiam na informação extraída a partir dos elementos a sugerir, mas sim nas semelhanças entre utilizadores. Para inferir um valor de relevância para um elemento, o sistema tem por base as classificações atribuídas a esse elementos por outros utilizadores com preferências semelhantes. Assim, passa a ser necessário obter informação sobre as preferências dos utilizadores, de modo a tornar possível determinar níveis de semelhança entre eles. Tal poderá ser obtido explicitamente (através de questionários, por exemplo) ou implicitamente (através das classificações atribuídas pelo utilizador).

À semelhança do que ocorre nos sistemas content-based, o sistema poderá ter pouca informação sobre o utilizador, o que poderá gerar recomendações incorrectas.

Outro dos problemas destes sistemas são a inserção de novos elementos. Estes elementos só serão sugeridos aos utilizadores após obterem um número significativo de avaliações. Estes mecanismos têm ainda de lidar com problemas de esparsidade de dados. Dado que o número de classificações atribuídas é tendencialmente menor do que o número de classificações que têm de ser inferidas, é necessário garantir que o sistema é capaz de gerar sugestões baseando-se num pequeno volume de dados.

Soluções híbridas As soluções híbridas combinam as abordagens anteriormente apresentadas. Subdividem-se em quatro grupos:

- Combinação de sistemas independentes: O sistema infere classificações com base em dois sistemas independentes (cada um baseado num dos modelos anteriormente apresentados) e combina essa informação para gerar as sugestões finais;
- Acrescentar características *content-based* a sistemas *collaborative-based*;
- Acrescentar características *collaborative-based* a sistemas *content-based*;

- Sistemas que unificam as duas abordagens num mecanismo único, com base nos dois tipos de classificações.

2.3.2 GroupLens

O GroupLens [10] é um sistema que permite às aplicações clientes do serviço Usenet (*newsgroups*) incorporarem mecanismos de recolha, apresentação e partilha de avaliações de artigos.

O sistema é composto por duas partes. A primeira delas é uma biblioteca para integração nos clientes de *newsgroups*. Esta biblioteca define um conjunto de primitivas que permitem receber e transmitir avaliações sobre conteúdos. O modo como estas avaliações são obtidas e apresentadas ao utilizador fica a cargo dos implementadores dos clientes de *newsgroups*, não sendo definido pela biblioteca. Esta apenas permite comunicação, utilizando um protocolo aberto, definido especificamente para este efeito.

A biblioteca permite comunicar com os servidores GroupLens, a segunda parte do sistema. Estes servidores são responsáveis por executar um conjunto de operações:

- Gerar sugestões de conteúdos para apresentar aos utilizadores, com base nas avaliações atribuídas a conteúdos, e um critério de semelhança entre utilizadores;
- Armazenar informações recebidas dos clientes, contendo avaliações a conteúdos;
- Gerar, a partir das avaliações feitas por cada utilizador, um valor de semelhança entre utilizadores. Este valor é calculado com base nas avaliações feitas pelos utilizadores a um mesmo artigo (avaliações semelhantes para um mesmo artigo indicam interesses/opiniões em comum);

Para lidar com problemas de esparssidade de avaliações e elevado volume de notícias a classificar, o modelo proposto opta por particionar os conteúdos em subconjuntos. Tal é facilmente alcançável se tivermos em conta que o próprio modelo *newsgroup* é, por natureza, um modelo hierárquico de notícias. Ao recorrer a este mecanismo de particionamento obtém-se maior fiabilidade nas sugestões, bem como menor carga sobre o servidor GroupLens.

Análise Crítica O GroupLens define uma plataforma que permite diversos utilizadores participarem na classificação de conteúdos, bem como propagarem e apresentarem essa informação. Embora opere num modelo cliente/servidor, o seu objectivo é semelhante ao do trabalho em curso, podendo ser utilizada uma abordagem semelhante.

2.3.3 *Automatic News Construction*

Wang et al. [17] definem um modelo para agregação automática de notícias. O algoritmo proposto consegue identificar semelhanças entre notícias provenientes de diversas fontes, agrupando-as com base no seu conteúdo. Tal permite gerar documentos subordinados a um dado tópico (ou conjunto de tópicos), contendo informação completa e detalhada sobre este(s).

O modelo proposto assume que as diversas notícias sobre o mesmo acontecimento são emitidas num curto espaço temporal. O algoritmo processa todas as notícias recebidas durante um intervalo de tempo, tentando identificar pontos em comum. Caso encontre, agrupa essas notícias em tópicos. Esses tópicos passam a possuir um conjunto de *keywords* associadas, geradas a partir das *keywords* das notícias que contêm.

Os novos tópicos gerados a partir das novas notícias são então comparados com os tópicos já existentes, de modo a determinar se são realmente ”novos”. Caso sejam semelhantes a tópicos já existentes, são agrupados.

Um processo semelhante é realizado para agrupar diferentes tópicos que estejam relacionados, gerando assim um conjunto de notícias que estão relacionadas com um dado tema (tema esse que engloba diversos tópicos).

O algoritmo proposto prevê ainda a possibilidade de recorrer a e incluir informação proveniente de outras fontes. Tais fontes, como blogues ou fóruns, possuem características especiais, como incluírem mais informação irrelevante ou as suas fontes serem menos credíveis. Assim, é necessário recorrer a mecanismos de filtragem de conteúdos mais finos. A proposta dos autores é limitar estas fontes a blogues/fóruns conhecidos, ou a autores credíveis. O algoritmo de agrupamento de conteúdos é semelhante ao acima descrito.

A grande vantagem deste algoritmo é que permite agrupar informação sobre um dado tópico ou tema, tornando o acesso à informação mais rápido e menos trabalhoso. A possibilidade de processar e agrupar informação independentemente da fonte torna-o especialmente útil. O conteúdo final gerado permite ao utilizador ter acesso a informação agrupada e condensada, tornando a sua leitura mais apelativa e informativa. Este conceito poderá ser aplicável ao trabalho a desenvolver, de modo a permitir ao utilizador um mais fácil acesso a conteúdos relacionados.

Análise Crítica Esta ferramenta permite agrupar a informação reunida de diversas fontes, de modo a tornar o acesso do utilizador final à mesma mais simples e rápido. Este é um dos critérios relevantes para garantir a usabilidade e aceitação da plataforma a desenvolver. No entanto, esta abordagem apenas permite agrupar informação. Cabe ao utilizador final filtrar por si qual a relevante e qual a desinteressante.

2.3.4 Yes, There is a Correlation - From Social Networks to Personal Behavior on the Web

Singla et ál [15] apresenta um estudo das relações entre utilizadores de IM (*instant messaging*) que comunicam entre si e as suas preferências pessoais. Embora tenha por base dados extraídos a partir de aplicações de IM, os autores do artigo acreditam que as conclusões obtidas se mantêm válidas para redes sociais.

Do cruzamento de dados entre as conversas mantidas entre utilizadores de uma aplicação de IM e as suas pesquisas num motor de busca (neste caso, respectivamente, Windows Messenger e Windows Live Search) é possível inferir alguns dados interessantes, e que poderão ter influência na área dos *recommender systems*:

- Utilizadores que mantêm uma relação próxima aquando da utilização de aplicações de IM (falam frequentemente, muitas vezes ou mantêm conversas de longa duração) tendem a efectuar pesquisas pelos mesmos termos, o que indica um conjunto de interesses semelhantes;
- Esses mesmos utilizadores situam-se frequentemente na proximidade uns dos outros (tipicamente na mesma cidade ou região);
- As relações acima indicadas acentuam-se com o aumento da proximidade dos utilizadores (mais tempo passado a conversar, maior número de mensagens trocadas, etc);
- O tempo despendido por mensagem é inversamente proporcional à proximidade dos utilizadores (em relações mais formais, os utilizadores tendem a demorar mais tempo por mensagem, e vice-versa);
- Dois utilizadores que partilhem um terceiro como amigo tendem a ter interesses em comum.

Este tipo de relações entre dados de IM e interesses pessoais dos utilizadores poderão ser úteis para inferir *ratings* a partir de classificações atribuídas por outros, em que existe uma relação numa rede social ou numa ferramenta de IM entre ambos os utilizadores.

Análise Crítica Este artigo demonstra um outro modo de inferir semelhanças entre as preferência pessoais dos utilizadores. Embora o seu foco de análise sejam os comportamentos na utilização de motores de busca e aplicações de conversação em tempo real, as suas conclusões podem ser transpostas para a utilização de redes sociais (Hi5, Facebook, etc). Este tipo de informação poderá ser útil como fonte adicional de informação para a classificação de conteúdos e/ou a geração de sugestões.

3. *Background*

3.1 Comanche

O Comanche [2] é um sistema que permite que, num contexto *web*, algumas operações computacionais sejam executadas no cliente, ao invés de no servidor. Para tal, o sistema actua como um *proxy*, instalado na máquina cliente. Esse *proxy* obtém aplicações (sob a forma de *Java Web Applications*) do servidor do prestador de um serviço. Essas aplicações efectuam operações sobre os pedidos e respostas que transitam através desse *proxy*.

3.1.1 Funcionamento geral

O principal objectivo do Comanche [2] é tirar partido dos recursos computacionais disponíveis nos computadores pessoais que acedem a serviços disponibilizados na *web*, de modo a acelerar o tempo de resposta aos pedidos e reduzir o tráfego gerado e a carga sobre os servidores. Para tal, o Comanche executa no cliente, sob a forma de um *proxy*. De modo a tratar os pedidos interceptados, o Comanche permite a instalação de *Java Web Applications* disponibilizadas pelos provedores de serviços *web*. Essas aplicações ficam associadas a um determinado endereço ou expressão regular, que delimitam o seu domínio de acção.

As operações possíveis de serem executadas por estas aplicações vão desde simples funcionalidades de *logging*, passando por transformações dos pedidos que são efectivamente encaminhados para o servidor, até à supressão do pedido ao servidor, sendo a resposta totalmente gerada localmente.

3.1.2 Tipos de aplicações

Consoante as funções que desempenham e as funcionalidades a que se destinam, as aplicações enquadram-se numa de três categorias:

3.1.2.1 *Final Web Applications*

Estas aplicações são capazes de interceptar os pedidos realizados ao servidor. São ainda capazes de utilizar dados locais para gerar a totalidade das respostas, ou contactar um servidor para obter ou submeter dados. Estas aplicações podem ainda definir um conjunto de outras aplicações como dependências. Essas aplicações, caso ainda não estejam instaladas, são-no durante o processo de instalação da aplicação final. O objectivo das aplicações finais é disponibilizar serviços que pouco ou nada dependem da comunicação com o servidor.

3.1.2.2 *Transform Web Applications*

Estas aplicações interceptam e podem modificar o pedido efectuado e a resposta do servidor. Ao contrário das *Final Web Applications*, a sua principal função não é disponibilizar serviços localmente, mas sim diminuir a carga sobre o servidor através, por exemplo, da supressão de parte do pedido realizado ou respondendo a um pedido com dados existentes localmente. Podem ainda manipular os dados apresentados numa página com base em informação obtida a partir de outras fontes.

3.1.2.3 *Observation Web Applications*

Estas aplicações são capazes de monitorizar os pedidos e respostas trocados entre cliente e servidores. Destinam-se a aplicações de *logging*/análise estatística e outras onde apenas é relevante recolher informação. O sistema Comanche garante que aplicações desta natureza não são capazes de modificar os pedidos e as respostas. São apenas capazes de os analisar e armazenar localmente dados sobre eles.

3.1.3 Funcionalidades e segurança

De modo a garantir a segurança do utilizador final, as aplicações executam em isolamento. Adicionalmente, cada aplicação está associada a um conjunto conhecido de endereços. No caso das aplicações finais, estas apenas podem operar sobre endereços que tenham um prefixo idêntico ao da sua instalação. As aplicações de transformação e observação têm associadas a si uma expressão regular que define sobre que pedidos podem operar. As aplicações encontram-se ainda limitadas no acesso ao sistema de ficheiros.

Para oferecer suporte à execução, o Comanche disponibiliza às aplicações um sistema de armazenamento de dados de forma persistente, suportado por uma base de dados relacional JavaDB. É garantido o isolamento entre aplicações dos dados armazenados desta forma. No entanto, é possível às aplicações finais acederem às bases de dados de aplicações de transformação e observação das quais dependam.

3.1.4 Análise de características

As funcionalidades do Comanche permitem a execução local de operações computacionais que, de outro modo, teriam de ser executadas num servidor *web*. Do ponto de vista do utilizador final, é possível reduzir o tráfego gerado e o tempo de resposta durante a utilização de um serviço que recorra a uma aplicação Comanche, à custa do tráfego e tempo despendidos na instalação da aplicação.

A utilização de um *proxy web* poderá acarretar um custo adicional, dado que existe um elemento intermédio no acesso à rede. No entanto, como demonstrado em [2], tal é desprezável.

3.1.5 Extensibilidade

O Comanche foi desenvolvido como forma de complementar os serviços disponibilizados por servidores centrais, onde estão alojados conteúdos e/ou serviços. No entanto é possível, através do desenvolvimento de aplicações específicas, disponibilizar serviços que apenas dependam das próprias aplicações. Neste trabalho, recorre-se a esta abordagem para implementar aplicações Comanche que comunicam entre si de modo a disponibilizar uma funcionalidade específica.

3.2 LiveFeeds

O LiveFeeds [1] é um sistema de disseminação cooperativa de *feeds* RSS, que permite diminuir a carga sobre os servidores destes conteúdos. Criando uma estrutura *peer to peer* entre os subscritores de *feeds*, é possível criar canais que permitem disseminar esta informação, sem que todos os subscritores tenham de contactar o servidor que a disponibiliza.

3.2.1 Funcionamento geral

O funcionamento do LiveFeeds tem como primitiva base a optimização do processo de disseminação de conteúdos. O encaminhamento é feito apenas para os nós que estão interessados na informação a propagar, e apenas esses intervêm no processo de disseminação desse conteúdo.

Cada nó da rede é responsável por conhecer o endereço e as preferências de todos os utilizadores da rede. Isto permite-lhe saber exactamente quais os nós interessados nessa informação, e quais os que não a desejam receber. Ao ser identificado um novo conteúdo, o nó que o detecta encaminha-o para outros nós nele interessados. Este processo é repetido pelos receptores, e assim sucessivamente, até que todos os interessados tenham sido informados.

Dado que todos os nós do sistema conhecem as preferências de todos os utilizadores, as *feeds* são encaminhadas apenas para os nós cujos utilizadores estão interessados nestas. Assim, nenhum elemento da rede é encarregue de encaminhar mensagens relativamente a conteúdos nos quais não está interessado. Esta característica garante que os recursos são utilizados de forma justa: um utilizador que subscreva poucas *feeds* terá uma carga menor no seu sistema do que um utilizador que subscreva muitas.

3.2.2 Em detalhe - canais e nós

De modo a dar suporte ao modo de funcionamento acima descrito, cada instância do LiveFeeds deve especificar qual(ais) o(s) canal(ais) a que se deseja juntar. A cada canal estão associados um modelo de mensagem e um modelo de filtro. O modelo de mensagem define a estrutura dos conteúdos/eventos a transitar nesse canal. Num cenário simplista, o modelo de mensagem seria, por exemplo, uma estrutura que permitisse armazenar o conteúdo de uma *feed* RSS. O modelo de filtro determina o modo como os nós especificam as suas preferências face aos conteúdos do canal. São estes filtros que permitem determinar se um nó está ou não interessado num dado

conteúdo. Numa abordagem ingénua, este filtro poderia ser a lista dos URLs das *feeds* em que o nó está interessado.

Para cada canal a que adere, o nó deve especificar as suas preferências relativamente aos conteúdos que nele transitam, instanciando um filtro de acordo com o modelo do canal. Esse filtro é posteriormente passado aos restantes nós. É através desse filtro que é possível determinar o interesse de um nó em receber um dado conteúdo.

Todos os nós contêm uma estrutura interna que armazena grupos de três dados: identificador de um nó na rede, identificador de um canal a que este pertença e filtro que especifica as preferências desse nó relativamente aos conteúdos que transitam nesse canal. Note-se que cada nó guarda a informação relativa a todos os nós e todos os canais, inclusive sobre nós com os quais não partilha canais e canais aos quais não pertence.

3.2.3 Comunicação

Durante o processo de divulgação de preferências, e de forma a garantir que essa informação chega a todos os nós, alguns deles assumem o papel de "*Slice leaders*". Cada um destes nós é responsável por receber as alterações às preferências de um subconjunto da totalidade dos nós. Periodicamente, essas alterações são agrupadas por cada um dos "*Slice leaders*", e disseminadas para todos os nós da rede. Tal permite difundir para todos os nós a informação completa, com possibilidade de usar compressão e, através da serialização desses anúncios, reparar eventuais falhas.

Para garantir a eficácia no encaminhamento e não incorrer em problemas de latência acumulada, o LiveFeeds opta por utilizar uma DHT *1-Hop*, ou seja, todos os nós conhecem os endereços de todos os restantes elementos presentes na rede. Combinando esta informação com o conhecimento das preferências de todos os nós, o LiveFeeds consegue disseminar de forma óptima os eventos: encaminhamento apenas entre os nós interessados no conteúdo a ser disseminado, e directo (um salto), o que reduz significativamente a latência nas comunicações.

3.2.4 Análise de características

O facto de todos os nós da rede conhecerem as preferências de todos os utilizadores permite que a informação seja optimamente difundida. Em particular, permite garantir que, ao receber uma *feed* a partir de um outro elemento da rede, esta é relevante (de acordo com as preferências especificadas no filtro). Assim, elimina-se o problema de falsos positivos (*feeds* que são recebidas mas não subscritas). Por outro lado, garante-se também a inexistência de falsos negativos, dado que as *feeds* relevantes serão sempre entregues a todos os que nelas estão interessados. Estas duas características conferem a esta plataforma um elevado grau de fiabilidade.

O ponto negativo mais relevante desta plataforma prende-se com o custo da actualização dos filtros, que ocorre quando um nó entra ou sai da rede ou de um canal, ou modifica as suas preferências. Sempre que tal ocorre, todos os nós da rede têm de ser informados dessa alteração e do conteúdo do novo filtro, o que leva à geração de tráfego proporcional ao tamanho do filtro

especificado e do número de nós existentes na rede. Tal pode ser colmatado utilizando mecanismos de compressão de informação ou, no caso de entradas e saídas de nós da rede, utilizando um hash do filtro actual como mecanismo de comparação com o último filtro conhecido pelos restantes elementos da rede (evitando a difusão do filtro caso seja idêntico ao já conhecido).

Embora eficazes, estas estratégias apenas permitem minimizar o problema, não o eliminando. Assim, verifica-se que o principal factor que limita a escalabilidade da plataforma não é o número de elementos presentes na rede, mas sim a taxa de actualização das preferências dos utilizadores.

3.2.5 Extensibilidade

Embora o intuito original da plataforma LiveFeeds seja a disseminação colaborativa de *feeds*, esta pode ser facilmente estendida para suportar outras funcionalidades de interesse. Com base no conhecimento disponível em cada nó (endereço e preferências de todos os elementos da rede), é possível completar a plataforma de forma a disponibilizar outros serviços de troca de informação entre nós.

Em particular, para o âmbito deste trabalho, torna-se útil a criação de um serviço de *anycast*: encaminhamento de um pedido para um elemento de uma lista de nós que cumprem um dado requisito. Esta característica será útil para permitir a implementação de um mecanismo que permita a um nó pedir explicitamente aos seus vizinhos um dado conjunto de conteúdos (p.ex.: histórico de uma *feed*).

Embora tenha sido desenvolvido com o intuito de disseminar *feeds*, não existe nenhuma limitação que impeça a disseminação de outro tipo de conteúdos através do LiveFeeds. Assim, será possível implementar canais específicos para a disseminação de informação relativa às preferências do utilizador, de modo a permitir a troca de sugestões de *feeds* de interesse.

É ainda possível manipular a informação transmitida sobre as *feeds*, de modo a que se adeqúe a alguns cenários particulares. Por exemplo, será possível enviar um resumo das *feeds* para um utilizador que aceda a partir de um dispositivo móvel de conexão limitada e/ou intermitente (telemóvel, pda, etc).

3.3 Bloom Filter

O Bloom Filter [5] é uma estrutura de dados, desenvolvida com o objectivo de permitir identificar o estado de subscrição de um conjunto de elementos. Para tal, recorre a uma estrutura interna de dimensão fixa, onde armazena informação sobre os elementos subscritos. Com base nessa informação, é possível determinar se um elemento pertence ou não a esse conjunto. Os resultados negativos obtidos são garantidamente correctos; no entanto, existe a possibilidade de ocorrência de falsos positivos.

Na base deste mecanismo encontra-se um vector de *m bits*, totalmente preenchido com zeros. A dimensão desse vector é imutável e definida no momento da criação do filtro. Associado

também ao filtro encontra-se um conjunto de uma ou várias funções de *hashing* independentes, capazes de operar sobre os dados que se pretendem identificar com o filtro e de, a partir destes, gerar valores num espaço de resultados entre 0 e m.

Durante o processo de adição de um elemento, as funções de *hash* são aplicadas ao elemento a inserir, daí resultando um ou mais valores. Esses valores identificam posições do vector de *bits* que deverão ser colocadas com o valor 1, caso ainda não estejam. O filtro permite a adição de um número infinito de elementos, embora perca precisão com o aumento deste valor. A remoção de um elemento depois de inserido não é possível.

3.3.1 Verificação de Filiação

De modo a testar a presença de um elemento no filtro, são aplicadas k funções de *hash* a este, e verificam-se se as posições desse modo obtidas se encontram todas marcadas a 1. Em caso negativo, garante-se que o elemento não pertence ao conjunto: caso esse elemento já tivesse sido inserido, todas as posições geradas pelas funções de *hash* teriam de, forçosamente, estar assinaladas a 1. No entanto, caso se detecte uma resposta afirmativa, esta não é garantidamente correcta.

Ao ser verificado cada um dos k *bits*, a probabilidade de ele já se encontrar assinalado a, como resultado de uma das n inserções anteriores, é dada pela expressão:

$$1 - \left(1 - \frac{1}{m}\right)^{kn}$$

em que m representa a dimensão em *bits* do vector do filtro. Assim, ao ser testada a presença de um elemento, a probabilidade de todos os k *bits* a ele associados estarem já assinalados a 1, com base na inserção de outros elementos, é de:

$$\left(1 - \left[1 - \frac{1}{m}\right]^{kn}\right)^{k} \approx \left(1 - e^{-kn/m}\right)^{k}$$

Caso tal ocorra, o filtro indicará que o elemento está presente, sem que o este tenha alguma vez sido inserido, ocorrendo um falso positivo.

3.3.2 Dimensionamento e Taxa de Falsos Positivos

Pela análise da expressão anterior, é possível concluir que a probabilidade de falsos positivos aumenta com o valor de n e diminui com o de m. O número de funções de *hash* a utilizar, e consequente número de *bits* associados a um elemento, também afecta a probabilidade da geração de falsos positivos. O seu valor ideal, que minimiza essa taxa, é dado pela expressão

$$\frac{m}{n} \ln 2$$

a partir do qual se obtém uma probabilidade de falsos positivos de

$$2^{-k} \approx 0.6185^{m/n}$$

Daí se extrai que, para um valor de espectável de n elementos a inserir, e para uma probabilidade p de obter falsos positivos, m é

$$m = -\frac{n \ln p}{(\ln 2)^2}$$

3.3.3 Escalabilidade

A taxa de falsos positivos aumenta com número de elementos representados no filtro, diminuindo com o aumento da dimensão física do filtro. Umas vez criados, estes filtros não podem ser redimensionados, pelo que uma análise *a priori* é necessária e vital para garantir um filtro que, simultaneamente, seja de pequena dimensão física e gere um número tolerável de falsos positivos. Como forma de contornar a esta limitação, foi desenvolvido o Scalable Bloom Filter [3], capaz de variar de dimensão após ter sido criado. Essa variação de dimensão pode ocorrer mesmo após o filtro já ter sido preenchido, sem que a informação relativa aos elementos já inseridos seja perdida. Recorrendo a este mecanismo, torna-se possível redimensionar dinamicamente o filtro, mantendo a taxa de falsos positivos a um nível desejado, sem que seja necessário conhecer *a priori* o número de elementos a inserir.

4. Desenho da Solução

4.1 Objectivos/Requisitos do Sistema

O objectivo deste trabalho é desenvolver um sistema cooperativo, denominado ColFeeds, para melhorar o acesso à informação disponível na *web*, em particular a *feeds* RSS. Este sistema cooperativo permitirá tornar o processo de disseminação de notícias mais eficiente, permitindo aos utilizadores colaborarem entre si no processo de divulgação de notícias.

Este sistema permite aos subscritores de *feeds* comunicarem directamente entre si, de modo a trocarem notícias pertencentes às *feeds* que seguem. O acesso aos dados presentes no servidor que aloja as *feeds* passa a ser esporádico, permitindo a optimização do processo de difusão de eventos, ao mesmo tempo que se reduz a carga nos servidores que albergam os dados. Todo este processo ocorre sem necessidade de recorrer a um servidor central que coordene os diversos elementos: a rede formada é estritamente *peer to peer* pura.

Um dos objectivos principais é a transparência da plataforma, tanto para os criadores de conteúdos, como para os seus leitores. Assim, o funcionamento da aplicação é totalmente transparente para os geradores de eventos, e não requer nenhum cliente específico. O utilizador poderá utilizar o mesmo cliente de *feeds* que utiliza anteriormente, desde que o configure adequadamente. Isto potencia a adopção da plataforma, dado que não implica alterações no comportamento do utilizador.

Além de permitir ao utilizador final aceder aos conteúdos RSS de forma mais eficiente, esta plataforma é capaz de elaborar sugestões de conteúdos relevantes para o utilizador final, com base em dados recolhidos sobre as suas preferências pessoais. Esses dados são obtidos de forma automática, através da monitorização dos conteúdos consultados pelo utilizador. Esses dados não só servem para determinar as suas preferências, mas também para gerarem sugestões de interesse a outros utilizadores. Este processo, além de ocorrer de forma transparente para o utilizador final, respeita a sua privacidade, pois os dados transitam de forma anónima entre utilizadores.

4.2 Arquitectura Geral

Na figura 4.1 apresenta-se a arquitectura do sistema desenvolvido. Este sistema tem por base um *proxy web*, construído usando o sistema Comanche [2]. Esta plataforma permite executar localmente aplicações capazes de efectuar operações de monitorização e transformação sobre pedidos e respostas trocadas entre o cliente e os servidores *web* a que este acede, como se descreve na secção 3.1.

Sobre o Comanche executa uma aplicação responsável pela extracção de informação, que monitoriza todo o tráfego realizado, analisando o seu conteúdo e armazenando informação sobre este. A informação recolhida desta forma permite à plataforma preencher uma base de dados

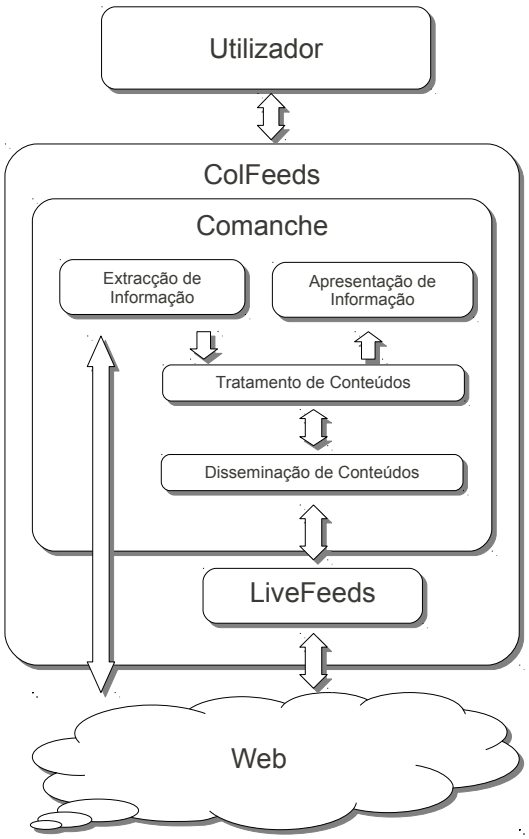

Figura 4.1 Arquitectura da solução

onde são registadas informações gerais sobre os conteúdos consultadas pelo utilizador, como a natureza dos mesmos, palavras-chave a eles associadas e números de acessos.

Esta aplicação de monitorização é ainda responsável por detectar se um determinado pedido corresponde a um acesso a uma *feed* RSS. Em caso afirmativo, esses dados são encaminhados para o módulo de Tratamento de Conteúdos. Esse componente é responsável por armazenar e disponibilizar as *feeds* RSS a que o utilizador acede, bem como tratar todos os dados necessários à manutenção do mecanismo de sugestões. Desta forma, ao aceder a uma *feed* RSS, o pedido do utilizador pode ser respondido total ou parcialmente com base em dados existentes localmente, reduzindo o número de consultas a servidores de conteúdos.

Esses dados são posteriormente encaminhados para a plataforma LiveFeeds [1], que possibilitará a comunicação com os restantes utilizadores deste serviço, e a troca das *feeds* e das sugestões geradas.

A aplicação é ainda capaz de, a partir dos dados armazenados localmente e recebidos dos restantes utilizadores, gerar e apresentar *feeds* que apresentam, no seu conteúdo, sugestões de *feeds* relacionadas, de potencial interesse para o utilizador.

4.3 Extracção de Informação

A extracção de informação ocorre como resposta à necessidade de obter informação sobre as preferências do utilizador. Recorrendo às capacidades oferecidas pelo Comanche, é possível monitorizar todo o tráfego gerado pelo utilizador, e sobre este recolher informação que permitam determinar quais os interesses pessoais deste.

A recolha de dados passa pela intercepção e análise dos pedidos e respostas HTTP gerados pelo utilizador durante a sua navegação na *web*. Esses dados são feitos passar através do *proxy* Comanche, que é capaz de os analisar, recolhendo informações diversas, como endereços acedidos, momento do acesso, conteúdo dos sites consultados, entre outros. Os dados recolhidos desta forma permitem ao sistema conhecer as preferências do utilizador: *feeds* que segue, páginas que consulta, tópicos de efectivo ou potencial interesse, entre outros. Os dados deste modo recolhidos são posteriormente utilizados na secção de Tratamento de Conteúdos, de modo a permitirem a gestão das *feeds* e sugestões.

Recorrendo às funcionalidades disponibilizadas pelo Comanche, é possível fazer esta recolha de informação de forma totalmente transparente. A recolha de dados é feita de forma totalmente automática e implícita, sem que o utilizador tenha de preencher qualquer tipo de formulário ou questionário ou, de qualquer outra forma, informar explicitamente o sistema sobre os seus gostos ou preferências. Ao evitar modificar o habitual comportamento do utilizador ao navegar na *web*, facilita-se a adopção da plataforma. Tal como identificado em [10], forçar o utilizador a modificar os seus hábitos pode dificultar a aceitação de uma plataforma ou tecnologia, ou mesmo levar ao seu abandono, o que confere a esta característica especial importância.

4.4 Tratamento de Conteúdos

O módulo de Tratamento de Conteúdos é a parte da plataforma ColFeeds que intercepta a informação pedida pelo utilizador ao consultar uma *feed* e, através das potencialidades oferecidas pelo Comanche, gera uma resposta baseada em dados obtidos através da rede ColFeeds. Neste processo, é conveniente identificar dois processos distintos e independentes, Tratamento de *Feeds* e Tratamento de Sugestões, discutidos de seguida em detalhe.

4.4.1 Tratamento de *Feeds*

A componente de tratamento de *feeds* é responsável por gerir e armazenar toda a informação associada às *feeds* RSS a que o utilizador acede.

Este módulo é activado sempre que é detectada, através das funcionalidades de intercepção e modificação de pedidos e respostas HTTP do Comanche, um acesso a uma *feed* RSS. Através dessas mesmas funcionalidades, e com base nos dados existentes localmente, provenientes dos restantes utilizadores ou através da consulta ao servidor que disponibiliza a *feed*, este módulo é capaz de responder aos pedidos realizados pelo cliente final. Para gerar esta resposta é invocada a componente de Apresentação de Informação.

4.4.2 Tratamento de Sugestões

O objectivo deste componente é gerar um conjunto de sugestões de *feeds* de interesse ao utilizador final. Para tal, baseia-se na informação recolhida pelo módulo de Extracção de Informação para inferir quais os interesses do utilizador.

Adicionalmente, este módulo troca informação com as restantes instâncias do ColFeeds, recorrendo à componente de Disseminação de Conteúdos. Essa troca de informação permite obter conhecimento sobre *feeds* desconhecidas ao utilizador local, mas que tenham sido consultadas por outros utilizadores da rede.

Combinando estes dois tipos de dados, este módulo é capaz de gerar um conjunto de sugestões de *feeds* para o utilizador. As sugestões geradas têm por base as preferências inferidas do utilizador, pelo que devem ser de interesse para este.

As sugestões geradas são encaminhadas para a componente de Apresentação de Informação, para serem apresentadas ao utilizador final.

4.5 Disseminação de Conteúdos

Para permitir que as diversas instâncias da plataforma comuniquem entre si, o ColFeeds recorre ao LiveFeeds [1]. Este sistema de comunicação permite aos diversos utilizadores formarem entre si uma rede *peer to peer*, onde diversos canais *publish/subscribe* possibilitam a troca filtrada de informação entre eles. A filtragem é feita com base em informação disponibilizada

por cada um dos nós, que reflecte as preferências do utilizador face à informação trocada.

No contexto do ColFeeds, existem dois tipos de informação que necessitam de ser trocados entre os diversos elementos da rede. O primeiro corresponde aos dados das *feeds* RSS que os nós trocam entre si, de modo a permitir dispensar a consulta constante dos servidores que as disponibilizam. O segundo tipo de informação trocada é a necessária à apresentação de sugestões de *feeds* aos utilizadores.

4.5.1 Disseminação do Conteúdo das *Feeds*

Este componente permite que as diversas instâncias do ColFeeds troquem entre si informação relativa ao conteúdo das *feeds*, possibilitando a divulgação de actualizações às mesmas e, consequentemente, a dispensa da consulta ao servidor. Dado que assenta sobre o LiveFeeds, este mecanismo herda as suas características, sendo de particular relevância a garantia de que, sempre que não ocorram falhas, não são gerados falsos negativos: a divulgação ocorre com garantias de entrega do evento detectado a todos aqueles que se declararam como estando interessados na *feed* a que este pertence. Em caso de ocorrência de uma falha no processo de disseminação, a garantia de entrega é probabilística. Este mecanismo é utilizado sempre que é obtida uma *feed* a partir do servidor, seja para divulgar eventuais alterações detectadas, ou para renovar a validade da mesma nas restantes instâncias do ColFeeds.

4.5.2 Disseminação de Sugestões

A disseminação de sugestões ocorre como forma de disponibilizar a cada um dos nós informações sobre *feeds* que os seus respectivos utilizadores desconhecem, mas que foram consultadas por outros utilizadores. Tal como acontece na disseminação de *feeds*, a disseminação de sugestões ocorre tendo por base o LiveFeeds e, por isso, a troca de informação é feita de forma filtrada, e com garantias de ausência de falsos negativos, sempre que não ocorram falhas.

4.6 Apresentação de Informação

Tal como explicitado anteriormente, sempre que uma *feed* é consultada pelo utilizador, a resposta é gerada pelo ColFeeds a partir da informação presente em base de dados. De modo a garantir que esta plataforma é utilizável sem que seja necessário um cliente de *feeds* específico, ou alterações aos já existentes, a informação gerada deve respeitar o formato definido na especificação RSS.

Adicionalmente, é necessário apresentar ao utilizador final as sugestões de conteúdos geradas pelo sistema. A apresentação desses dados também deve ser feito de modo a que o ColFeeds possa operar sob qualquer cliente de RSS, sem serem necessárias modificações ao mesmo.

Este módulo surge para dar resposta a esses problemas. Através da informação presente no

sistema relativa a *feeds* e sugestões, é capaz de gerar uma *feed* que respeita a especificação RSS, contempla todo o conteúdo da versão disponível no servidor, ao mesmo tempo que inclui, no seu interior, informação sobre as sugestões geradas pelo sistema.

As sugestões geradas e apresentadas abordam os mesmos temas que a *feed* onde se inserem, pois será nesse momento que serão mais valiosas para o utilizador final. Esta abordagem permite ainda que as sugestões geradas sejam apresentadas de forma gradual, evitando a apresentação de quantidades excessivas de informação e consequente desconforto por parte do utilizador ao usar a plataforma.

5 . Implementação

Neste capítulo descreve-se a implementação do sistema. Esta descrição não pretende ser exaustiva, mas antes focar os detalhes mais importantes, apresentando e discutindo a motivação por detrás da escolhas adoptadas.

5.1 Extracção de Informação

O módulo de Extracção de Informação pretende recolher informação sobre o utilizador, a partir da monitorização da sua navegação na *web*. A informação recolhida desta forma permite à plataforma preencher uma base de dados onde são registadas informações sobre as páginas e *feeds* consultadas, como tipo de conteúdos, palavras-chave a elas associadas e números de *hits*. Tais dados são posteriormente utilizados para determinar as *feeds* e os tópicos de interesse do utilizador, bem como gerar sugestões para outros utilizadores da rede.

A implementação escolhida recolhe informação sobre todos os endereços aos quais o utilizador acedeu, números e datas de acesso. São ainda registados os endereços que, não tendo sido explicitamente consultados, fazem parte de uma página consultada e representam uma ligação para uma *feed* RSS. Estes processos de recolha de dados ocorrem com base em informação obtida analisando e processando o conteúdo HTML e XML correspondente, respectivamente às páginas e *feeds* consultadas.

A informação recolhida por esta implementação não é muito detalhada, nem ambiciona a tal. O objectivo é demonstrar as capacidades da plataforma e fornecer uma base simples e facilmente extensível, ao mesmo tempo que recolhe dados suficientes para os mecanismos de Tratamento de *Feeds* e Tratamento de Sugestões implementados. A informação pertinente a recolher dependerá da informação requerida pelas implementações desses dois mecanismos, cujas características serão discutidas nas próximas secções.

5.2 Tratamento de *Feeds*

Este módulo é responsável por tratar os pedidos por *feeds* RSS, e é activado sempre que um é detectado. Caso essa *feed* não exista na base de dados local, o pedido é encaminhado normalmente para o servidor, e a resposta, que conterá o conteúdo da *feed*, interceptada. Os dados nela contidos são processados e armazenados localmente. Esse processamento é feito por um *parser* XML, capaz de processar o conteúdo de uma *feed*. Esses dados são armazenados em base de dados local e enviados para o módulo de Disseminação de Conteúdos, para que possam ser divulgados para os restantes membros da rede.

Caso a *feed* consultada já esteja presente na base de dados local, é verificada a actualidade da informação existente, de modo a garantir que esta é suficientemente recente. Uma *feed* é considerada como estando actual caso tenha sido actualizada nos últimos 5 minutos. Caso

tal se verifique, o pedido ao servidor é suprimido. Se, por outro lado, a informação estiver desactualizada, o pedido é encaminhado para o servidor, para que se possa obter uma versão mais recente dos dados. A resposta obtida é guardada em base de dados local e encaminhada para o módulo de Disseminação de Conteúdos, para divulgação.

Em qualquer um destes casos, a resposta final à aplicação cliente é sempre gerada a partir da base de dados local, pela componente de Apresentação de Informação, apresentada na secção 5.5.

Outra das competências deste módulo é determinar que *feeds* são subscritas ou não pelo utilizador. Dada a natureza do protocolo RSS, e do módulo no qual se baseia a sua utilização, a acção de subscrever uma fonte é um processo que ocorre exclusivamente ao nível do cliente de *feeds*. Assim, torna-se impossível, através de uma implementação com esta arquitectura, determinar com plena certeza se uma dada *feed* foi ou não efectivamente subscrita pelo utilizador final. Passa a ser necessário inferir, a partir da informação interceptada, se o utilizador subscreveu ou não uma dada *feed*.

É legítimo assumir que uma *feed* efectivamente subscrita será acedida regular e periodicamente num espaço de tempo relativamente curto, dado que esse é o único modo do qual os clientes de *feeds* tradicionais dispõem para detectar actualizações. Este comportamento espectável serve como base para tentar inferir se esta foi ou não subscrita pelo utilizador final. Nesta implementação, considera-se uma *feed* como estando subscrita caso o seu endereço tenha sido consultado três ou mais vezes nas últimas 24 horas. Numa versão final do sistema, este valor seria obtido a partir de resultados experimentais e configurável para melhor se adaptar à realidade de cada utilizador. Seria ainda necessário elaborar algoritmos mais complexos, que permitam determinar, com maior grau de fiabilidade, quais as *feeds* que o utilizador realmente segue.

5.3 Tratamento de Sugestões

O objectivo deste componente é gerar, com base nos dados obtidos sobre o utilizador e provenientes dos restantes membros da rede ColFeeds, um conjunto de sugestões de *feeds* de interesse. Este problema de inferência de interesses encontra-se amplamente estudado na literatura, sendo alvo de bastantes projectos de investigação, passados e presentes. Não sendo, na sua essência, um problema abrangido pela área de investigação deste trabalho, optou-se por utilizar uma solução simplista, cujo objectivo é demonstrar as funcionalidades da solução global, sem se comprometer relativamente à qualidade dos resultados gerados.

A implementação escolhida comunica com as restantes instâncias da plataforma ColFeeds através da componente de Disseminação de Conteúdos, divulgando um conjunto de tópicos de interesse do utilizador. Essa lista de tópicos é obtida agrupando todas as palavras-chave encontradas nos itens de *feeds* assinaladas como subscritas pelo utilizador. A divulgação dessa lista permite a recepção de resumos sobre outras *feeds*, que contenham uma ou mais palavras-chave de interesse ao utilizador.

A principal desvantagem da utilização de palavras-chave na elaboração de sugestões é

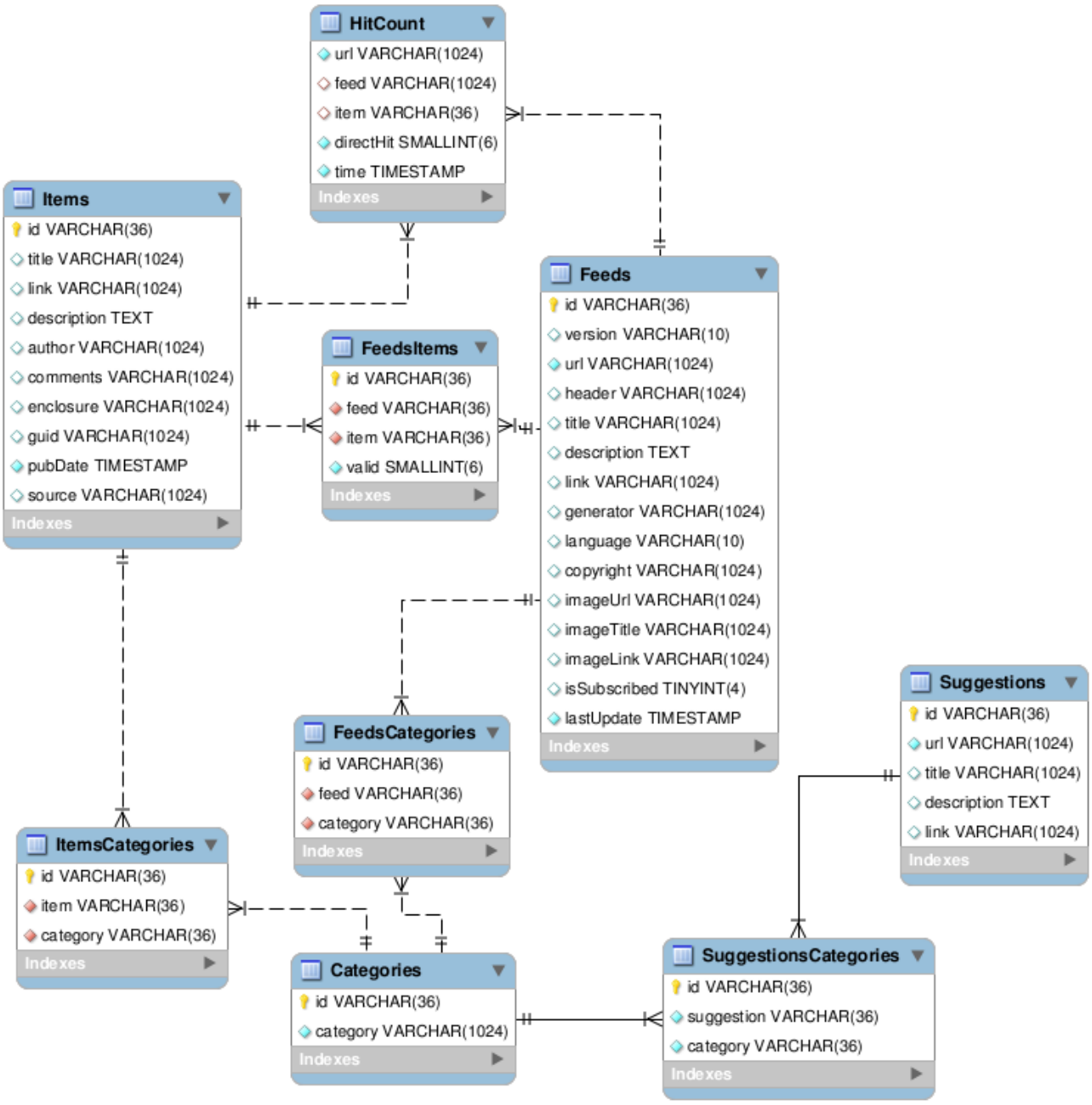

Figura 5.1 Diagrama da base de dados

o facto de que estas serem um elemento opcional de uma *feed*. Numa versão final desta plataforma, e caso se optasse por manter a abordagem de geração de sugestões com base em palavras-chave, seria necessário complementar a lista de palavras-chave com dados adicionais, obtidos, por exemplo, a partir da detecção de palavras-chave na descrição das noticias contidas na *feed*.

Uma sugestão é criada e divulgada sempre que uma nova *feed* é detectada por um nó. Adicionalmente, a publicação da totalidade das sugestões é feita por cada nó da rede a cada 60 minutos. Deste modo é possível fazer com que estas cheguem a todos os utilizadores, mesmo os que sofreram de um período de desconexão, sem que tal se torne uma sobrecarga excessiva sobre a rede e os nós que a compõem. Adicionalmente, os resumos propagados contêm o título, descrição, endereço da *feed* e palavras-chave a ela associadas, dados suficientes para que o utilizador final possa ter acesso a um resumo do conteúdo da *feed* sugerida e, caso assim entenda, aceder à mesma.

5.4 Disseminação de Conteúdos

A componente de Disseminação de Conteúdos é responsável por permitir que as diferentes instâncias do ColFeeds troquem entre si os dados que constituem uma *feed* RSS, de modo a que seja possível dispensar, na maioria dos casos, a consulta directa ao servidor que mantêm a *feed*. Permite também que sejam trocados dados que servem de base à operação da componente de Tratamento de Sugestões.

Esta troca de conteúdos ocorre sobre o LiveFeeds. Tal como analisado anteriormente, o LiveFeeds é uma plataforma desenvolvida com o intuito de possibilitar a troca filtrada de *feeds* RSS entre diversos nós. No entanto, a sua implementação permite que outros tipos de informação que não *feeds* sejam trocados. Deste modo, é possível utiliza-lo como método único de comunicação entre nós.

O LiveFeeds permite a troca de informação através da criação de canais independentes, aos quais cada participante pode aderir através das especificação de um filtro. Esse filtro irá determinar quais dos conteúdos que circulam no canal o seu autor irá receber. Dado que esses filtros são conhecidos por todos os nós da rede, é possível fazer a disseminação dos dados de forma filtrada, optimizando o processo de divulgação de informação e reduzindo o consumo de largura de banda. Assim, é necessário utilizar um modelo de filtro que garanta a inexistência de falsos negativos, ao mesmo tempo que minimiza ou elimina falsos positivos.

Embora o tráfego gerado durante a disseminação de eventos seja relevante, é também necessário minimizar a dimensão física dos filtros. Sempre que ocorre uma entrada na rede, ou que um utilizador altera as suas preferências, o seu filtro tem de ser propagado. A informação das suas preferências é feita chegar a todos os nós da rede, mesmo aqueles com os quais não tem preferências em comum ou não partilha canais.

Assim, torna-se importante minimizar não só o tráfego gerado durante a divulgação de eventos, mas também a dimensão física dos filtros a utilizar. De forma a atingir de forma eficaz estes

objectivos, é necessário dimensionar os filtros a utilizar tendo por base a informação que se pretende divulgar. Assim, optou-se por separar a disseminação de dados em dois canais independentes.

5.4.1 Disseminação do Conteúdo das *Feeds*

O processo de Disseminação do Conteúdo das *Feeds* tem atribuído a si um canal dedicado para esse efeito. Neste canal circulam os dados associados ao conteúdo das diversas *feeds* subscritas pelos utilizadores, sendo o processo de emissão executado sempre que um dos nós obtém, a partir de um servidor, uma *feed*. Nesse processo de emissão, o conteúdo completo da *feed* é feito transitar entre os diversos nós, mesmo que não existam alterações ao mesmo. Esta abordagem permite que todos os nós obtenham, a cada actualização, uma versão completa do actual estado da *feed* em questão, renovando a validade da sua cópia local da mesma. Embora não seja a mais eficiente em termos de largura de banda, esta abordagem é a que apresenta maior simplicidade, sendo também a menos propensa a efeitos negativos resultantes da ocorrência de falhas na transmissão de dados.

Dada a natureza da informação que é trocada neste canal, é necessário especificar um modelo de filtro que garanta a ausência de falsos negativos. Caso estes ocorressem, as alterações a uma *feed* não chegariam a todos os seus subscritores. Por outro lado, é tolerável que o filtro utilizado permita a existência de falsos positivos, pois estes podem ser filtrados no receptor. Embora estes tenham um impacto negativo na carga sobre a rede, não comprometem a funcionalidade da plataforma.

Assim, o canal de divulgação de *feeds* RSS tem como modelo de filtro Bloom Filters. Esta estrutura permite armazenar e testar, com garantias de ausência de falsos negativos, se um dado elemento está ou não subscrito. A dimensão física desta estrutura é fixa, e varia consoante a taxa de falsos positivos que se espera do filtro. Nesta implementação assume-se como razoável uma taxa máxima de falsos positivos de 1%. A determinação do valor óptimo desta taxa implica a recolha de dados experimentais, como tamanho médio do conteúdo de uma *feed*, taxa de actualização de filtros, entre outros, processo que não foi possível completar no decorrer deste trabalho.

Tendo este valor como valor máximo tolerável de falsos positivos, a implementação é capaz de dimensionar automaticamente o filtro, sempre que este é gerado. Tal o que ocorre quando o utilizador se junta à rede, ou sempre que as suas preferências são alteradas. Esta abordagem permite optimizar a dimensão do filtro, que crescerá com o número de *feeds* subscritas, garantindo uma percentagem baixa e constante de falsos positivos.

A informação é guardada no filtro através de mecanismos próprios de *hashing*, que devem operar sobre identificadores únicos e universais de cada *feed*, de modo a poderem ser verificados em qualquer nó. Para esse efeito, a implementação utiliza o URL da *feed* como identificador global e único da mesma.

Este filtro é utilizado pelo LiveFeeds para determinar quais os nós interessados em receber uma dada *feed*. As características do LiveFeeds, já apresentadas garantem que a informação é

entregue a todos os utilizadores cujo filtro contenha a *feed* em trânsito. No entanto, dadas as características dos Bloom Filters, podem ocorrer falsos positivos. Para além do excesso de tráfego gerado, estas ocorrências não têm qualquer impacto negativo no processo de disseminação de informação, sendo filtradas aquando da sua recepção.

5.4.2 Disseminação de Sugestões

A disseminação de sugestões é feita recorrendo a um canal independente mas idêntico ao utilizado para disseminar as *feeds*. Assim, os problemas associados à resolução deste problema são os mesmos que os discutidos anteriormente, pelo que a solução utilizada é também um Bloom Filter.

Tal como discutido anteriormente, a subscrição de sugestões ocorre com base nas palavras-chave associadas às *feeds* subscritas pelo utilizador. Assim, essas palavras-chave são utilizadas para popular o filtro que o nó utiliza para se juntar a este canal. Periodicamente, cada nó irá publicar neste canal resumos das *feeds* que conhece, com base nas palavras-chave associadas a cada uma delas. Todos os nós que tenham especificado alguma dessas palavras-chave no seu filtro irão ser informados sobre os conteúdos divulgados.

Tal como acontece no processo de Disseminação do Conteúdo das *Feeds*, também na Disseminação de Sugestões podem ocorrer falsos positivos. Dado que o resumo da *feed* que é feito transitar neste canal inclui a lista de palavras-chave associadas à *feed*, a filtragem no receptor é possível. Analogamente ao que ocorre no canal de Disseminação de *Feeds*, tem-se como tolerável uma taxa de falsos positivos de 1%.

Ao ser disseminada uma sugestão, o processo de difusão ocorre uma vez por cada palavra-chave associada à sugestão. Isto pode levar a que a mesma sugestão seja entregue por diversas vezes ao mesmo receptor. Idealmente, a disseminação e filtragem deveriam ocorrer com base na lista de palavras-chave, evitando a entrega de resultados em duplicado. No entanto, devido a uma limitação da implementação, tal não é possível.

5.5 Apresentação de Informação

Ao ser consultada uma *feed*, independentemente do processamento que ocorre sobre esse pedido, a resposta é gerada pelo ColFeeds, através da componente de Apresentação de Informação. Este módulo do sistema é responsável por gerar uma resposta HTTP que contenha, no seu interior, uma *feed* construída obedecendo ao protocolo RSS.

Para tal, este componente do sistema utiliza um gerador de XML que, a partir do identificador da *feed* pedida, acede à base de dados local e consegue reconstruir o conteúdo da mesma. As *feeds* geradas deste modo apresentam os elementos XML que fazem parte da especificação formal RSS, garantindo-se a equivalência face à *feed* original, bem como a compatibilidade com os clientes de *feeds*.

No entanto, é necessário incluir informação adicional no conteúdo da *feed*, de modo a

```
<?xml version="1.0" encoding="utf-8" ?><rss version="2.0" xml:base="http://www.javali.pt/rss.xml" xmlns:dc="http://purl.org/
dc/elements/1.1/">
  <channel>
    <title></title>
    <link>http://www.javali.pt/rss.xml</link>
    <description></description>
    <language>pt-pt</language>
          <item>
    <title>Javali procura Web Designer</title>
    <link>http://www.javali.pt/destaques/javali-procura-web-designer</link>
    <description></description>
     <pubDate>Quarta, 21 Jul 2010 19:26:03 +0100</pubDate>
 <dc:creator>nadia.guerreiro</dc:creator>
 <guid isPermaLink="false">250 at http://www.javali.pt</guid>
  </item>
  <item>
    <title>Museu do Caramulo implementa solução de POS da Javali - JavaliPOS </title>
    <link>http://www.javali.pt/destaques/museu-do-caramulo-implementa-solu-o-de-gest-o-pos-da-javali-javalipos</link>
    <description></description>
     <pubDate>Quarta, 21 Jul 2010 14:27:07 +0100</pubDate>
 <dc:creator>nadia.guerreiro</dc:creator>
 <guid isPermaLink="false">249 at http://www.javali.pt</guid>
  </item>
  <item>
    <title>Novo SugarCRM 6 já chegou!</title>
    <link>http://www.javali.pt/destaques/novo-sugarcrm-6-j-chegou</link>
    <description></description>
     <pubDate>Quinta, 15 Jul 2010 13:08:10 +0100</pubDate>
 <dc:creator>nadia.guerreiro</dc:creator>
 <guid isPermaLink="false">247 at http://www.javali.pt</guid>
  </item>
  <item>
    <title>Versão beta do Firefox 4 já está disponível para download</title>
    <link>http://www.javali.pt/destaques/vers-o-beta-do-firefox-4-j-est-dispon-vel-para-download</link>
    <description></description>
     <pubDate>Sexta, 09 Jul 2010 13:08:22 +0100</pubDate>
 <dc:creator>nadia.guerreiro</dc:creator>
 <guid isPermaLink="false">245 at http://www.javali.pt</guid>
  </item>
  <item>
```

Figura 5.2 Exemplo do código XML de uma *feed*

que seja possível apresentar sugestões ao utilizador final. Para responder a este problema, acrescenta-se, no final de cada *feed*, um item adicional, denominado '*Suggestions*'. Este item contém uma tabela, em formato HTML, onde são listadas as sugestões de *feeds* relacionadas com a que está a ser apresentada. Neste contexto, considera-se que uma sugestão e uma *feed* estão relacionadas caso tenham em comum pelo menos uma palavra-chave. Cada sugestão é apresentada sob a forma de um *link* para o endereço onde a *feed* completa está alojada, seguida do texto da descrição da mesma.

Através desta abordagem, garante-se a legibilidade da *feed* gerada, independentemente do cliente RSS usado pelo utilizador.

6. Avaliação

A avaliação do sistema ColFeeds foi feita em duas vertentes. Primeiro, realizou-se uma avaliação qualitativa do sistema, de modo a garantir que este conseguia alcançar os objectivos propostos.

De seguida realizou-se uma avaliação do seu desempenho. Esta avaliação pretende medir os custos e benefícios da utilização desta plataforma, quando comparativamente à utilização do modelo de acesso directo aos servidores de *feeds*.

6.1 Avaliação Qualitativa

A avaliação qualitativa pretende avaliar se o ColFeeds é capaz de atingir os objectivos propostos, e assim ser capaz de suportar o serviço pretendido.

6.1.1 Identificação de Conteúdos

Em primeiro lugar, foi avaliada a capacidade do ColFeeds de identificar correctamente as *feeds* consultadas pelo utilizador. Para tal, testou-se o acesso a diversos endereços, representando diversos casos-tipo:

1. Acesso a páginas HTML com referência a uma ou mais *feeds* no seu cabeçalho, onde se espera que estas sejam detectadas;
2. Acesso a páginas HTML sem referência a *feeds* no seu cabeçalho, onde se espera que o pedidos e respostas sejam analisados mas não modificados ou suprimidos;
3. Acesso ao endereço de um ficheiro XML correspondendo a uma *feed* RSS, onde se espera que esta seja detectada e analisada;
4. Acesso ao endereço de um ficheiro XML que não corresponda a uma *feed*, onde se espera um resultado idêntico ao do cenário anterior;

Este último caso-tipo é relevante dado que uma *feed* é especificada através de um ficheiro XML. Assim, o acesso a ficheiros XML que não representem *feeds* podem ser uma potencial fonte de erros, conferindo-lhes especial relevância para teste.

Em todos os casos o comportamento do ColFeeds foi comparado com o do mecanismo de detecçao de *feeds* existent no Firefox. Para além das funcionalidades de *browser*, esta aplicação disponibiliza um mecanismo de subscrição de RSS, bem como uma funcionalidade de detecção de *feeds* em páginas HTML, tornando-o na ferramenta ideal para utilizar enquanto referência neste teste. Os resultados são apresentados em duas tabelas, onde a ocorrência de um comportamento esperado é assinalado com o símbolo ✓ e um comportamento inesperado com o símbolo ×

6.1.1.1 Conteúdos HTML

A figura 6.1 apresenta uma tabela com dados resultantes desta avaliação experimental, recolhidos a 25/07/2010, referentes à análise de de conteúdos em formato HTML por parte de ambas as plataformas. Nela é possível atentar que o ColFeeds apresentou o comportamento esperado em todos excepto dois casos. Após análise detalhada, foi possível chegar às seguintes conclusões:

- http://gmailblog.blogspot.com/ - Existe, no cabeçalho desta página, a referência a uma *feed* RSS. Esse endereço, quando consultado, gera uma resposta HTTP com o código 302, correspondente a um redireccionamento. Esta resposta é encaminhada para o ColFeeds, cuja implementação não prevê este tipo de respostas. A tentativa de identificar uma *feed* gera um erro, e é abortada, continuando o ColFeeds a operar normalmente.

- http://www.microsoft.com/en/us/default.aspx - O acesso a este endereço retorna uma resposta HTTP 400 - Pedido inválido. Em testes realizados utilizando o Comanche sem a aplicação ColFeeds instalada obteve-se o mesmo erro, pelo que se comprova que o mesmo tem origem no Comanche. A correcção do mesmo não foi possível.

A análise destes resultados permite concluir que o ColFeeds é capaz de eficazmente detectar e processar os acessos a páginas contendo referências a *feeds* RSS, existindo, no entanto, espaço para melhoria desta funcionalidade. O tratamento de páginas HTML sem qualquer conteúdo RSS ocorreu, em todos os casos, de acordo com o esperado.

É ainda relevante mencionar que algumas das páginas testadas recorrem à tecnologia AJAX, que se baseia na troca de conteúdos XML entre cliente e servidor. Nos casos testados o ColFeeds identifica o trânsito de dados XML, mas é capaz de, com sucesso, detectar que estes não correspondem a *feeds* RSS.

6.1.1.2 Conteúdos XML

A figura 6.2 apresenta uma tabela com dados resultantes desta avaliação experimental, recolhidos a 26/07/2010, referentes à análise de de conteúdos em formato XML por parte das duas plataformas. Nela é possível identificar que o ColFeeds apresentou o comportamento esperado em todos excepto dois casos. A análise dos mesmos levou às seguintes conclusões:

- http://feeds.feedburner.com/OfficialGmailBlog - O acesso a este endereço provoca o lançamento de diversas excepções na plataforma. Testes posteriores demonstram que o mesmo comportamento se verifica caso se utilize o Comanche sem a aplicação ColFeeds a executar sobre este, pelo que não foi possível determinar com exactidão a origem deste erro.

- http://www.microsoft.com/presspass/rss/mscomfeed.xml - À semelhança do ocorrido no acesso à *homepage* da Microsoft, este endereço retorna uma resposta HTTP 400 - Pedido inválido. Em testes realizados utilizando o Comanche sem a aplicação ColFeeds instalada obteve-se o mesmo erro, pelo que se comprova que o mesmo tem origem no Comanche.

Páginas com Feed	Firefox 3.6.7	ColFeeds
http://edition.cnn.com	✓	✓
http://www.neowin.net/	✓	✓
http://publico.pt/	✓	✓
http://www.javali.pt/	✓	✓
http://gmailblog.blogspot.com/	✓	×
http://www.microsoft.com/en/us/default.aspx	✓	×
http://www.apple.com/hotnews/feeds/ticker.rss	✓	✓
http://www.nasa.gov/	×	✓
http://www.worldofwarcraft.com	✓	✓
http://www.rss-specifications.com	✓	✓
http://www.softwaremarketingresource.com/	✓	✓
http://brainstormsandraves.com/xml/rss.xml	✓	✓
http://www.nytimes.com/	✓	✓
http://htf.atom.com/	✓	✓
http://www.opensuse.org/en/	✓	✓
http://www.tomshardware.com	✓	✓
Páginas sem Feed	**Firefox 3.6.7**	**ColFeeds**
http://www.di.fct.unl.pt/	✓	✓
http://www.bancobpi.pt/	✓	✓
http://www.amazon.co.uk/	✓	✓
http://www.starwars.com/	✓	✓
http://www.fox.com/	✓	✓
http://www.universidade-autonoma.pt/default.aspx	✓	✓
http://www.samsung.com/pt/	✓	✓
http://www.seat.pt	✓	✓
http://www.casadamusica.com/	✓	✓
http://www.siteground.com	✓	✓
http://www.ubuntu.com/	✓	✓
http://remax.pt/	✓	✓
http://www.redhat.com/	✓	✓
http://www.oracle.com/index.html	✓	✓

Figura 6.1 Resultados da análise de conteúdos HTML

Feeds	Firefox 3.6.7	ColFeeds
http://rss.cnn.com/rss/edition.rss	✓	✓
http://www.neowin.net/news/rss	✓	✓
http://feeds.feedburner.com/publicoRSS	✓	✓
http://www.javali.pt/rss.xml	✓	✓
http://feeds.feedburner.com/OfficialGmailBlog	✓	×
http://www.microsoft.com/presspass/rss/mscomfeed.xml	✓	×
http://www.apple.com/	✓	✓
http://www.nasa.gov/rss/featured_videos.xml	✓	✓
http://www.worldofwarcraft.com/rss.xml	✓	✓
http://www.rss-specifications.com/blog-feed.xml	✓	✓
http://www.icoblog.com/feeds/posts/default?alt=rss	✓	✓
http://brainstormsandraves.com/	✓	✓
http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml	✓	✓
http://feeds2.feedburner.com/happytreefriends/site	✓	✓
http://news.opensuse.org/feed/	✓	✓
http://www.tomshardware.com/feeds/rss2/tom-s-hardware-us,18-1.xml	✓	✓
Ficheiros XML	**Firefox 3.6.7**	**ColFeeds**
http://www.w3schools.com/XML/note.xml	✓	✓
http://www.w3schools.com/XML/note_error.xml	✓	✓
http://www.w3schools.com/XML/cd_catalog.xml	✓	✓
http://www.w3schools.com/XML/simple.xml	✓	✓

Figura 6.2 Resultados da análise de conteúdos XML

Estes resultados demonstram que, embora exista lugar para algumas melhorias, o ColFeeds apresenta o comportamento espectável na maioria dos casos testados, sendo capaz de processar correctamente a maior parte das *feeds* consultadas. Foi ainda capaz de identificar correctamente diversos ficheiros XML como não correspondendo a conteúdos RSS. Os testes realizados na secção 6.1.1.1 permitem ainda concluir que o ColFeeds é capaz de lidar correctamente com conteúdos XML resultantes da execução de pedidos AJAX.

6.1.2 Tratamento de Conteúdos

Em seguida, foi testada a capacidade do ColFeeds em processar correctamente os dados contidos numa *feed*, bem como recriar o conteúdo original a partir da informação armazenada localmente. Nesta operações intervêm os mecanismos de detecção de conteúdos testados na secção anterior, que permitem encaminhar o pedido gerado pela aplicação cliente para a componente de Tratamento de Conteúdos do ColFeeds. Ao ser processado por esse módulo, o pedido é, num primeiro acesso, encaminhado para o servidor. A *feed* obtida como resposta é processada e armazenada na base de dados local. A partir dos dados obtidos desta forma, a componente de apresentação de informação existente no ColFeeds é capaz de responder a um pedido por essa *feed* sem necessidade de recorrer ao servidor, desde que esse pedido surja dentro do período de validade atribuído pelo ColFeeds aos dados. Findo esse período, uma nova versão dos dados é obtida a partir do servidor, garantindo a atempada detecção de actualizações aos conteúdos.

De modo a testar este processo, e sem recurso ao ColFeeds, acedeu-se ao endereço das diversas *feeds* identificadas na figura 6.1 e obteve-se uma cópia do seu código-fonte. Em seguida, repetiu-se o mesmo procedimento, mas fazendo passar os dados através do ColFeeds. Na figura 6.3 podemos ver excertos desses ficheiros, onde se compara um item RSS tal como disponível no servidor, com o mesmo item gerado pelo ColFeeds. Neste exemplo as diferenças são apenas estruturais, sendo o conteúdo exactamente igual. Na análise completa dos diversos casos, são visíveis pequenas diferenças de conteúdos, nomeadamente a inexistência de alguns campos opcionais e de campos não presentes no *standard* RSS na versão gerada pelo ColFeeds. No entanto, estas diferenças não afectam a informação apresentada ao utilizador final. Em testes realizados com o Firefox, não foi possível detectar diferenças visíveis nas *feeds* apresentadas. Ficam, no entanto, identificadas como trabalho futuro, dado que alguns desses dados podem ser utilizados por alguns clientes de RSS.

Este teste comprova que o ColFeeds não só é capaz de analisar e armazenar correctamente o conteúdo de uma *feed*, como também é capaz de o recriar quando necessário. Testes adicionais foram realizados, que comprovaram a resiliência destes dados entre execuções.

6.1.3 Disseminação de Conteúdos

Outra das vertentes testadas foi a de Disseminação de Feeds: troca de actualizações de conteúdos entre as diversas instâncias do ColFeeds. Com esse objectivo utilizaram-se duas instâncias do ColFeeds, executando em máquinas independentes, com o intuito de fazer transitar entre elas

```
<item>
        <title>Microsoft reports strong Q4 revenue thanks again to Windows 7 sales</title>
        <link>http://www.neowin.net/news/microsoft-reports-strong-q4-revenue-thanks-again-to-w
        <description>Microsoft announced on Thursday, a record fourth-quarter revenue of $16.0

        The financial figures are a 22% increase from the same period of the prior year. Micro
        <author>Tom Warren</author>
        <pubDate>Thu, 22 Jul 2010 21:22:22 +0100</pubDate>
        <guid>http://www.neowin.net/news/microsoft-reports-strong-q4-revenue-thanks-again-to-w
</item>
```

```
<item>
<title>Microsoft reports strong Q4 revenue thanks again to Windows 7 sales</title>
<link>http://www.neowin.net/news/microsoft-reports-strong-q4-revenue-thanks-again-to-windows-7-
<description>Microsoft announced on Thursday, a record fourth-quarter revenue of $16.04 billion

        The financial figures are a 22% increase from the same period of the prior year. Micros
<pubDate>Thu, 22 Jul 2010 21:22:22 +0100</pubDate>
<author>Tom Warren</author>
<guid>http://www.neowin.net/news/microsoft-reports-strong-q4-revenue-thanks-again-to-windows-7-
</item>
```

Figura 6.3 Em cima, parte código fonte de uma feed, proveniente do servidor. Em baixo, o mesmo excerto de código, gerado pelo ColFeeds

actualizações a uma *feed*. Como cliente final foi utilizado o Firefox, desactivando a *cache*. Para ser possível forçar a ocorrência de alterações ao seu conteúdo, optou-se por utilizar uma cópia do conteúdo de uma *feed*, alojada num servidor local.

Inicialmente, acedeu-se repetidamente à *feed* em apenas uma das instâncias. Este procedimento fez com que o ColFeeds identificasse esta *feed* como estando subscrita pelo utilizador, tendo modificado o seu filtro do canal de *feed* de acordo com essa alteração. Na segunda instância em execução do ColFeeds foi possível observar que, durante todo o processo, não foi recepcionada qualquer *feed*. Este comportamento era o esperado, dado que este nó não tinha subscrito qualquer fonte de conteúdo.

Em seguida, modificou-se o conteúdo da *feed*, removendo-lhe um item, e acedeu-se por diversas vezes a esta a partir da segunda instância do ColFeeds em execução. Foi possível verificar, através das mensagens geradas pelo ColFeeds na consola, que a *feed* foi obtida a partir do servidor. A visualização da *feed* no *browser* demonstra que a versão obtida já reflectia as alterações efectuadas ao ficheiro-fonte.

Foi ainda possível verificar, através de mensagens publicadas na consola, que a *feed* foi transmitida para a outra instância em execução do ColFeeds, dado que esta tinham subscrito a *feed*. De modo a corroborar estes dados, foi retirada do servidor a *feed*, e recorreu-se à primeira instância do ColFeeds para consultar novamente o seu conteúdo. Nesse processo, foi possível verificar que, embora a cópia alojada no servidor já não estivesse disponível, o ColFeeds era ainda capaz de responder a pedidos por esta, tendo por base a informação armazenada localmente. Na resposta gerada foi ainda possível identificar que as alterações efectuadas sobre a

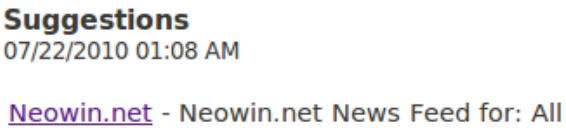

Figura 6.4 Apresentação de sugestões no corpo de uma feed

feed tinham sido propagadas. Esta consulta ao conteúdo ocorreu num prazo inferior a 5 minutos, de modo a garantir que o resultado apresentado era obtido a partir da base de dados local, e não era despoletado uma tentativa de actualização dos dados a partir do servidor.

Este teste permitiu concluir a capacidade do ColFeeds de fazer trocar entre as suas diversas instâncias actualizações detectadas a *feeds*.

6.1.4 Sugestões

Os testes à componente de sugestões visam garantir que estas são trocadas entre os diversos nós, mas também que são apresentadas de forma correcta, respeitando a especificação RSS. Assim, neste teste, recorreu-se a um cenário de teste semelhante ao anterior: duas instâncias do ColFeeds, executando em máquinas independentes. Para este teste utilizaram-se ainda cópias duas *feeds* distintas, alojadas num servidor local, às quais foram acrescentadas uma palavras-chave. Esta opção deve-se ao facto de, à data da realização destes testes, as *feeds* consultadas não apresentarem palavras-chave em comum.

Utilizando a primeira das instâncias do ColFeeds, consultou-se uma das *feeds*, forçando a sua subscrição por parte do ColFeeds. Monitorizando a consola, é possível identificar que esta sugestão foi propagada para a rede, não tendo sido recepcionada pelo outra instância em execução, tal como esperado.

Em seguida, utilizando a segunda instância do ColFeeds, consultou-se a segunda *feed*. Nova análise das mensagens de controlo demonstraram que a sugestão associada a esta *feed* foi propagada para a rede, tendo sido recepcionada pela primeira instância do ColFeeds. Ao consultar novamente a *feed* inicial na primeira instância do ColFeeds, foi possível observar, no final do seu conteúdo, a inclusão da sugestão à segunda *feed*. Na figura 6.4 é visível esse item, que representa e aponta para a *feed* consultada na segunda instância do ColFeeds.

Este teste demonstra a capacidade do ColFeeds de, com base nas *feeds* consultadas pelos utilizadores, gerar sugestões de conteúdos de potencial interesse.

6.2 Avaliação quantitativa

Na avaliação quantitativa foram analisados dois aspectos considerados relevantes para determinar o grau de sucesso desta plataforma: impacto da utilização do ColFeeds na experiência de utilização da web do utilizador, e recursos poupados nos servidores que disponibilizam as *feeds*

RSS.

6.2.1 Carregamento de Páginas no Cliente

Dado que o ColFeeds actua como um *proxy*, é necessário quantificar qual o acréscimo no tempo de carregamento das páginas consultadas pelo utilizador resultante da sua utilização. Esse acréscimo estará relacionado com o tempo despendido na intercepção e encaminhamento de pedidos e respostas HTTP pelo Comanche, bem como na extracção e processamento de dados efectuados pelo ColFeeds (a operar sobre o Comanche).

Dado que estas operações ocorrem localmente, os valores obtidos estão intrinsecamente dependentes das características do hardware onde a plataforma executa. Os valores em seguida discutidos foram obtidos num computador com processador Dual Core Intel T7700 a 2.40GHz, com 2GB de RAM, correndo Ubuntu 10.4 32bits. A ligação à internet é feita através de uma conexão ADSL com 8 *megabits* por segundo de velocidade máxima de *download*, e 1 *megabit* por segundo de *upload*.

Nos testes realizados, obtiveram-se valores para três cenários distintos:

- Ligação directa à *web*, correspondendo ao modo tradicional de operação.
- Ligação utilizando o Comanche, sem nenhuma aplicação a executar sobre este.
- Ligação utilizando o ColFeeds.

Estes cenários permitem não só determinar qual o impacto da utilização da plataforma, mas também qual a componente da mesma responsável por percas de desempenho. Os resultados apresentados reflectem a média dos tempos de acesso obtida com base em 24 acessos, tendo sido descartados os 2 piores e os 2 melhores tempos, de modo a eliminar a introdução de valores erróneos resultantes de variações no desempenho da rede. Foram ainda registados o número de pedidos HTTP resultantes do acesso a cada uma das páginas, bem como a dimensão total da mesma. Os acessos foram feitos utilizando o *browser* Firefox, com *cache* desactivada, recorrendo aos *add-ons* Firebug 1.5.4 e YSlow 2.0.7 para medição de valores.

O tipo de conteúdos solicitados pelo utilizador irá fazer variar o comportamento adoptado pelo ColFeeds no tratamento dos dado, o que, por sua vez, terá implicações no tempo de carregamento dos mesmos. Assim, de modo a melhor estrutura esta análise, os testes foram divididos em três grupos distintos, variando, entre eles, o comportamento esperado do ColFeeds no tratamento dos dados.

Endereços com dados HTML sem referências a feeds RSS Na figura 6.5 é possível observar um gráfico representativo da média dos valores de tempos de acesso a páginas HTML sem referência a *feeds* RSS. Neste cenário, o ColFeeds apenas analisa o pedido e a resposta gerados com o intuito de registar os endereços consultados pelo utilizador, não efectuando processamento adicional.

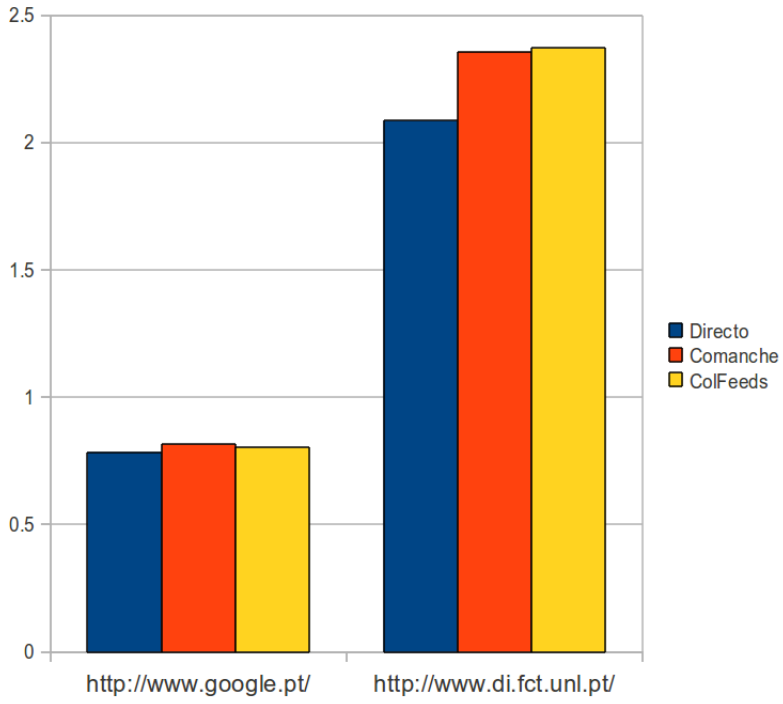

Figura 6.5 Tempos de acesso, em segundos, a endereços com dados HTML sem referências a *feeds* RSS

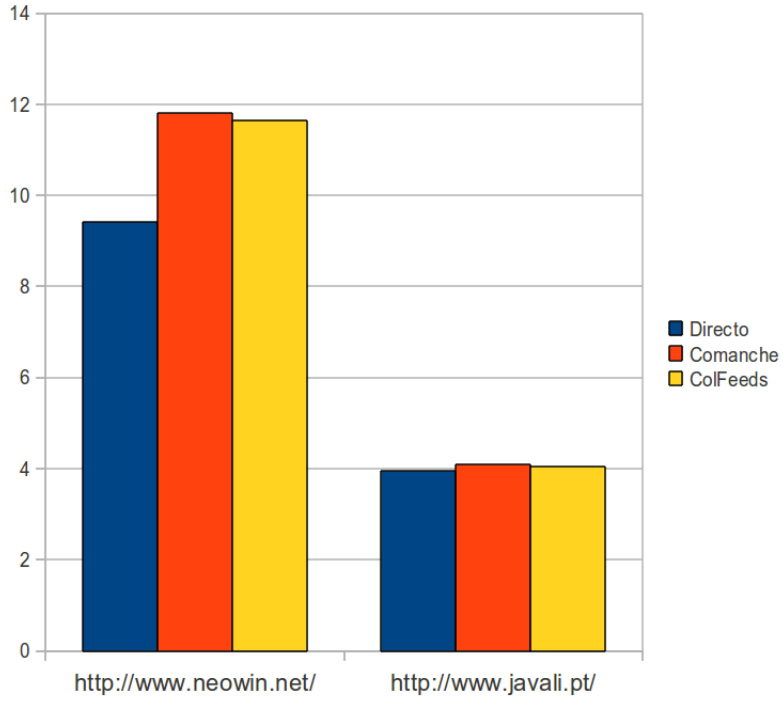

Figura 6.6 Tempos de acesso, em segundos, a endereços com dados HTML contendo referências a *feeds* RSS

Os resultados obtidos mostram que, nestes casos, a utilização do ColFeeds acarreta um custo adicional tolerável (3% e 12%, respectivamente). O custo introduzido pela utilização do ColFeeds é equivalente ao associado à utilização do Comanche sem qualquer aplicação instalada, permitindo concluir o eficiente funcionamento da implementação desenvolvida no decorrer deste trabalho.

Endereços com dados HTML contendo referências a feeds RSS A figura 6.6 representa um gráfico referente à média dos valores de tempos de acesso a páginas HTML cujo cabeçalho referencia *feeds* RSS. Neste cenário, o ColFeeds detecta a existência de conteúdos RSS, obtendo-os a partir do servidor, o que não ocorre no caso de acesso directo ou utilizando apenas o Comanche. Isto levará à geração de pedidos adicionais por parte do ColFeeds, bem como processamento local sobre o resultado desses pedidos.

Nos resultados obtidos é possível identificar, no primeiro caso, um significativo acréscimo no tempo de carregamento, tanto no cenário de utilização do ColFeeds, como no caso de utilização do Comanche. Analisando em detalhe a informação obtida, é possível identificar que este acréscimo significativo no tempo de carregamento da página possa estar associado ao elevado número de pedidos HTTP associados ao seu carregamento (mais de 100).

No segundo caso testado, os tempo de acesso sofrem um ligeiro aumento com a utilização

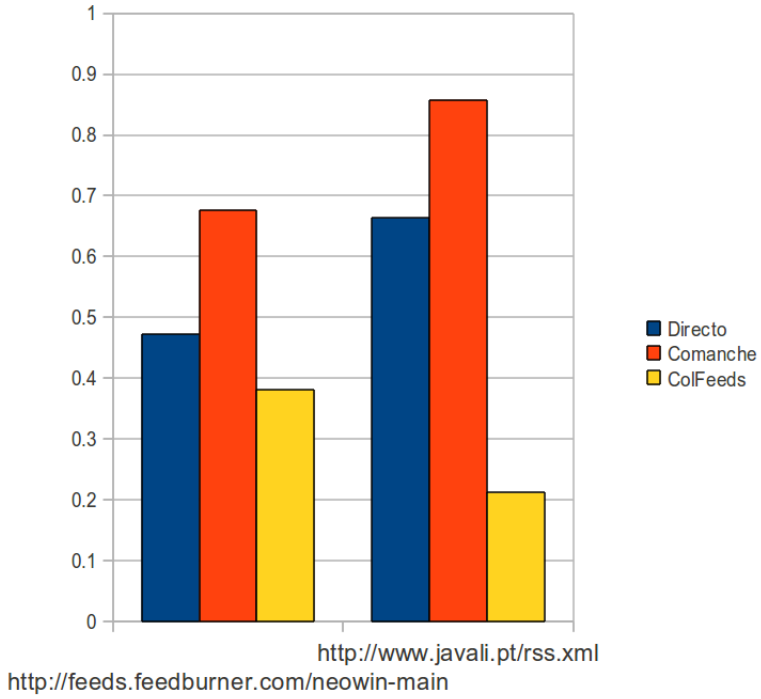

Figura 6.7 Tempos de acesso, em segundos, a endereços com dados XML referentes a *feeds* RSS

do Comanche e do ColFeeds. Os valores percentuais desse incremento são, respectivamente, 3,6% e 2.5%, sendo negligenciáveis.

Os resultados obtidos nos dois casos explicitam ainda que o tempo médio de carregamento utilizando o ColFeeds não é superior ao obtido quando é utilizado apenas o Comanche, pelo que é possível concluir que o tempo gasto na identificação e obtenção da *feed* presente na página e respectivo processamento é desprezável.

Endereços com dados XML referentes a feeds RSS A figura 6.7 representa um gráfico referente a acessos directos a *feeds* RSS. Neste cenário, e assumindo que a *feed* é desconhecida pelo ColFeeds, o primeiro acesso resultará no encaminhamento do pedido para o servidor, e intercepção, processamento e armazenamento local dos dados contidos na resposta. Os acessos seguintes ocorridos durante o período de validade da cópia local resultam na supressão total do pedido, sendo a resposta gerada apenas com base em dados existentes localmente. Um acesso efectuado a uma *feed* cujo período de validade tenha terminado resultará em novo acesso ao servidor que a aloja. Assim, torna-se pertinente referir que os tempos apresentados nesta figura referentes ao ColFeeds ilustram acessos efectuados dentro do período de validade da cópia local das *feeds* testadas.

No primeiro destes cenários, a *feed* inclui, no seu conteúdo, referência a diversos conteúdos

externos, como *banners* e outras imagens, que resultam num conjunto adicional de pedidos HTTP. Embora estes dados façam parte do conteúdo da *feed*, a actual implementação não os armazena localmente, pelo que têm de ser obtidos a partir do servidor. Este facto leva a que o ganho obtido pela utilização do ColFeeds não seja tão elevado quanto possível (diminuição do tempo de carregamento de 475ms para 381ms, representando um ganho de 20%)

No segundo caso de teste recorreu-se a uma *feed* cujo conteúdo não implica a geração de pedidos HTTP adicionais. Neste cenário, a utilização do ColFeeds permite que o pedido ao servidor seja totalmente suprimido, sendo a resposta gerada localmente na integra. Assim, foi possível reduzir o tempo de carregamento de 664ms para 212ms, correspondendo a uma diminuição de 70%.

Embora não sejam apresentados na figura 6.7, foram também recolhidos, para cada uma das *feeds*, dados relativos aos tempos de acesso no caso em que a validade dos dados armazenados localmente tinha expirado. Nesses cenários constatou-se que o tempo de carregamento aumentou entre 200% e 300%, como resultado do processamento local que ocorre sobre estes dados. No entanto, este caso apenas se verifica numa fracção muito reduzida dos acessos aos conteúdos, tornando-o tolerável.

Estes resultados permitem demonstrar a capacidade do ColFeeds em reduzir substancialmente o tempo de acesso a conteúdos RSS, sempre que seja possível recorrer à cópia local dos mesmos para responder aos pedidos do utilizador.

6.2.2 Redução de Carga no Servidor

Outra das características do ColFeeds é a sua capacidade de reduzir o número de acessos que são feitos aos conteúdos alojados nos servidores. Sendo este um dos objectivos do seu desenvolvimento e adopção, é relevante avaliar quais os recursos que podem ser poupados nos servidores, quer em termos de volume de dados, quer em termos de número de acessos. No decorrer da elaboração deste trabalho, não foi possível recolher dados experimentais que suportassem esta avaliação. Assim, faz-se uma análise da estimativa dos eventuais ganhos conseguidos com esta solução.

Sem perda de generalidade, assume-se a utilização de uma única *feed*, alojada num servidor responsável por responder a todos os pedidos por esta. Consideram-se ainda n subscritores da *feed*, ligados em permanência à Internet, recorrendo a clientes RSS que refrescam o conteúdo disponibilizado uma vez a cada p minutos. Assume-se ainda uma dimensão média de d_f para o ficheiro da *feed*. Adicionalmente, cada pedido tem um custo constante d_o, correspondente aos tráfego gerado pelos cabeçalhos HTML trocados entre cliente e servidor.

Caso os clientes acedam directamente ao servidor, que responde com o conteúdo da *feed*, o volume de tráfego gerado entre os clientes e o servidor a cada minuto, será:

$$\frac{n\left(d_f + d_o\right)}{p}$$

Utilizando o ColFeeds, o servidor será interrogado, no melhor caso, por um único utilizador a

cada período de p minutos. Neste cenário, a carga gerada sobre o servidor será de:

$$\frac{d_f + d_o}{p}$$

Terminado o período de validade das cópias locais da *feed*, é necessário obter novamente a versão do servidor de modo a renovar a validade dos conteúdos. O número de clientes que consultam efectivamente o servidor dependerá do tempo t de propagação da notificação entre os nós da rede ColFeeds. Assumindo $t < p$, o número médio de clientes que irão aceder ao servidor, a cada minuto, é dado pela expressão:

$$\frac{t}{p} \times n$$

gerando a seguinte carga no servidor:

$$\frac{n\left(d_f + d_o\right)}{p} \times \frac{t}{p}$$

Comparando esta expressão com a que define o valor de tráfego gerado caso os clientes acedam directamente ao servidor, é visível que a utilização do ColFeeds introduz um ganho proporcional, definido por:

$$\frac{t}{p}$$

Num cenário real, espera-se $t \ll p$, pelo que é possível concluir que a utilização do ColFeeds permite uma poupança considerável de tráfego gerado no servidor. Esta poupança será tão maior quanto for o período de validade atribuído pelo ColFeeds às copias locais das *feeds*, beneficiando também da redução do tempo de propagação dos eventos entre os diversos nós da rede.

Exemplo Para melhor ilustrar o potencial ganho associado à utilização do ColFeeds, foram instanciadas as expressões apresentadas. Assim, assuma-se um sistema com 1000 utilizadores, que seguem uma *feed* de dimensão 10KB. Esses utilizadores verificam, a cada 5 minutos, a existência de actualizações, gerando 1KB de tráfego em cabeçalhos HTTP. Assuma-se também como 5 minutos o período de validade atribuído pelo ColFeeds às cópias locais dos conteúdos, e que a disseminação de um evento para todos os membros da rede ColFeeds demora, no pior caso, 10 segundos. Estes dados representam estimativas, não tendo por base qualquer recolha ou análise experimental.

Através da aplicação destes valores às expressões anteriormente apresentadas, é possível determinar que a carga sobre o servidor, no modelo sem recurso ao ColFeeds, seria de 2200KB a cada minuto. Recorrendo ao ColFeeds, o valor do tráfego gerado em média por minuto é inferior a 74KB, ou 3.33% do gerado no cenário em que não se utiliza o ColFeeds.

7. Conclusões

Com o crescente aumento do volume de informação disponível na *web*, torna-se necessário recorrer a tecnologias que nos permitam poupar tempo durante a navegação. O protocolo RSS foi pensado com base nesse problema, e surgiu como um meio de tentar levar a informação desde a fonte até aos utilizadores. No entanto, a sua utilização tem por base aplicações que, de forma automática, consultam regularmente as fontes de informação em busca de actualizações. Este processo, para além de intrinsecamente ineficiente, não possibilita que os utilizadores do serviço colaborem entre si, e daí retirem benefício.

Neste trabalho, desenvolveu-se um mecanismo que, apoiando-se nas mais-valias oferecidas pelo RSS, permite estender as suas funcionalidades e melhorar o seu funcionamento. Deste modo, é possível aos utilizadores colaborarem entre si para se manterem actualizados. O ColFeeds permite que os utilizadores troquem entre si a informação contida nas *feeds* RSS, optimizando o processo de divulgação de informação.

Esta plataforma opera com base num *proxy web*, capaz de interceptar, analisar e modificar o tráfego gerado pelo utilizador durante a sua navegação. Deste modo, é possível manipular os conteúdos RSS consultados pelo utilizador, suprimindo pedidos ao servidor e gerando respostas com base em informação trocada pelos diversos membros da rede ColFeeds.

Para além de colaborarem na disseminação da informação, a plataforma ColFeeds permite ainda que os utilizadores tenham acesso a um conjunto de sugestões de novas *feeds*, com base nas suas preferências pessoais. A plataforma é capaz de, de forma automática, não intrusiva e transparente, determinar quais os temas de interesse do utilizador. Com base nestes dados, o ColFeeds comunica com as restantes instâncias da plataforma para, de forma segura e anónima, obter informações sobre *feeds* de potencial interesse para o utilizador.

Essa informação é apresentada ao utilizador final como parte das *feeds* que consulta, respeitando o standard RSS. Esta característica garante que a plataforma ColFeeds é capaz de operar com qualquer cliente de RSS, minimizando o impacto da adopção desta tecnologia.

Os testes realizados permitem concluir que a utilização do ColFeeds é uma alternativa viável à utilização do modelo de consulta de RSS tradicional. O sistema é capaz de identificar correctamente quais as *feeds* subscritas pelo utilizador, bem como os temas nos quais demonstra interesse. Com base nestes dados, o ColFeeds foi capaz de gerir a divulgação de actualizações às *feeds* subscritas, assim como fazer chegar ao utilizador um conjunto de sugestões com base nos seus tópicos de interesse.

Foi ainda possível demonstrar que a utilização do ColFeeds pode acelerar substancialmente o tempo de acesso às *feeds* RSS por parte dos clientes deste tipo de conteúdos, acarretando um custo tolerável nas restantes consultas à *web*. Ao nível dos servidores foi possível obter uma elevada redução no volume de tráfego, sem comprometer a qualidade do serviço e a periodicidade da divulgação de actualizações à informação.

7.1 Principais Contribuições

A principal contribuição deste trabalho consiste no desenho e implementação de um sistema que permite aos subscritores de *feeds* RSS colaborarem na sua disseminação. Neste contexto, a disseminação cooperativa deste tipo de conteúdos pretende ser uma alternativa viável à consulta directa por parte de todos os utilizadores aos servidores responsáveis por alojar os conteúdos. A principal vantagem desta abordagem é a optimização do processo de divulgação de actualizações a este tipo de conteúdos. É ainda relevante referir, como objectivos alcançados por esta plataforma, a redução de carga sobre os servidores *web* que disponibilizam *feeds* RSS e a diminuição no tempo de acesso por parte dos utilizadores finais aos conteúdos desta forma divulgados.

O sistema permite que a disseminação cooperativa seja feita de forma transparente, permitindo que o utilizador continue a usar o seu cliente de RSS preferido. Para tal, a solução implementada intercepta as conexões HTTP, usando o Comanche [2]. Recorrendo a esta plataforma, é possível analisar e manipular o tráfego resultante da navegação do utilizador na Internet, permitindo apresentar aos utilizadores a informação obtida a partir dos restantes utilizadores da rede.

Para além de permitir a troca de informação relativa à actualização de *feeds* RSS, o ColFeeds permite aos seus utilizadores colaborarem na elaboração e troca de sugestões de conteúdos de interesse. Tal é feito recorrendo à análise do tráfego gerado pelo utilizador, através do Comanche, que permite inferir os seus tópicos de interesse. Estas sugestões são apresentadas ao utilizador final inseridas no conteúdo das *feeds* por ele seguidas.

De forma a comunicarem entre si, as diversas instâncias do ColFeeds recorrerem ao sistema de disseminação de *feeds* LiveFeeds [1]. Este sistema permite a troca de conteúdos RSS entre os utilizadores, assim como a divulgação de sugestões de novos conteúdos relevantes.

7.2 Trabalho Futuro

Uma das principais limitações deste trabalho prende-se com a falta de algoritmos precisos e fiáveis, capazes de identificar eficazmente quais as *feeds* subscritas pelo utilizador e quais os tópicos do seu interesse.

Foi possível, durante a fase de testes, concluir que o mecanismo actual de inferência de interesses e geração de sugestões baseado em palavras-chaves nem sempre é viável, dado que não é garantida a existência dessa informação no conteúdo das *feeds*.

Estes dois problemas pertencem a um âmbito que não é o deste trabalho, pelo que a sua resolução adequada não foi tentada.

A implementação poderá ainda beneficiar de mecanismos mais complexos de extracção de informação, capazes de mais eficazmente inferir as preferências do utilizador. Dado que a informação que é necessária ser extraída desta análise depende do algoritmo que a irá consumir, optou-se por implementar apenas uma recolha de dados generalistas e pouco detalhados.

Existem outros pontos de menor importância para a completude do trabalho, mas relevantes de serem abordados no futuro:

- O actual módulo de Tratamento de *Feeds* não abrange na totalidade os campos definidos na especificação RSS. Não são ainda suportados outros formatos de *feeds*, como o Atom.
- A actual implementação depende apenas de dados recolhidos de forma transparente para o utilizador Embora tal seja uma mais valia da plataforma, poderá ser desejável que o utilizador forneça explicitamente dados à aplicação. Uma abordagem mista poderá também ser uma solução válida.
- O processo de divulgação do conteúdo das *feeds* poderá ser modificado para ser baseado numa abordagem acumulativa, em que apenas são transmitidas as alterações ocorridas e não a totalidade da *feed*. Embora esta abordagem acarrete maior complexidade e propensão a corrupção de dados derivado a falhas, é possível desenvolver mecanismos suficientemente sofisticados para lidar com essas situações.
- Será ainda útil, numa versão futura desta plataforma, a capacidade de gerar sugestões não exclusivamente sobre feeds RSS. A geração de sugestões relativas a páginas *web* tradicionais seria um valioso complemento às funcionalidades já disponibilizadas pelo ColFeeds, reforçando a ideia que motivou o seu desenvolvimento.
- A implementação de mecanismos de avaliação de conteúdos por parte dos utilizadores (de forma explicita ou implícita) poderia ainda servir como complemento às funcionalidades implementadas, eventualmente permitindo o aumento da relevância das sugestões geradas pelo sistema.
- Outra das mais valias possíveis de serem incluídas nesta plataforma seria a integração com plataformas de redes sociais ou *instant messenging*, possibilitando a inferência de relações interpessoais existentes entre os utilizadores, com o objectivo de melhor conhecer as suas preferências e complementar as sugestões fornecidas.

Bibliografia

[1] Project livefeeds - p2p dissemination of web syndication content, 2008. Projecto PTDC/EIA/76114/2006, financiado pela FCT/MCTES.

[2] Filipe M. Afonso. Comanche: Aplicações web estendidas nos clientes. Master's thesis, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, 2009.

[3] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. Scalable bloom filters. *Information Processing Letters*, 101(6):255 – 261, 2007.

[4] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 203–212, New York, NY, USA, 2006. ACM.

[5] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[6] Advisory R. S. S. Board. Rss 2.0 specification. *http://www.rssboard.org/rss-specification/. Accessed 25th March 2010*, March 2010.

[7] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues. Efficient routing for peer-to-peer overlays. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 9–9, Berkeley, CA, USA, 2004. USENIX Association.

[8] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: a decentralized peer-to-peer web cache. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 213–222, New York, NY, USA, 2002. ACM.

[9] Emre Kiciman and Benjamin Livshits. Ajaxscope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 17–30, New York, NY, USA, 2007. ACM.

[10] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[11] Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[12] Gediminas Member-Adomavicius and Alexander Member-Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[13] Rodrigo Rodrigues, Barbara Liskov, and Liuba Shrira. The design of a robust peer-to-peer system. In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 117–124, New York, NY, USA, 2002. ACM.

[14] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[15] Parag Singla and Matthew Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 655–664, New York, NY, USA, 2008. ACM.

[16] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[17] Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. Automatic online news issue construction in web environment. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 457–466, New York, NY, USA, 2008. ACM.