Análise do custo e da viabilidade de um sistema P2P com visibilidade completa

Simão Mata, J. Legatheaux Martins, Sérgio Duarte e Margarida Mamede

Departamento de Informática
Faculdade de Ciências e Tecnologia, FCT
Universidade Nova de Lisboa
2829-516 Caparica, Portugal,
simao.m@gmail.com, {jose.legatheaux, smd, mm}@di.fct.unl.pt,
http://di.fct.unl.pt

Resumo Este artigo apresenta um estudo da viabilidade de um algoritmo de filiação P2P com visibilidade completa, sendo definidos os limites para este tipo de aproximação através da análise do custo do algoritmo em função do dinamismo do sistema. É ainda proposta e avaliada a utilização de uma rede com super-nós, como forma de suportar cenários de maior dinamismo.

Abstract This paper presents a feasibility study for a P2P membership algorithm with full visibility. The limits for this kind of approach are evaluated in terms of the cost of the algorithm as a function of system churn. It is also proposed and evaluated the use of super nodes as the basis for supporting more dynamic settings.

1 Introdução

A difusão filtrada (*content-based networking*) é um problema que há largos anos interessa à comunidade científica [2] e ainda hoje é alvo de estudo. Esta forma de comunicação multi-ponto é uma generalização do paradigma editor-assinante baseado no conteúdo [1]. A particularidade deste modelo de difusão reside nos participantes poderem manifestar o interesse em receber apenas as mensagens cujo conteúdo respeita um dado padrão, o qual pode ser entendido como um *filtro*.

A difusão filtrada tem-se revelado um problema complexo e têm sido numerosas as soluções propostas ao longo dos anos [7]. De um modo geral, o problema tem sido atacado através da construção de árvores de difusão que reflectem, de alguma forma, o grau de afinidade dos filtros submetidos pelos participantes. Nesse processo procura-se minimizar a entrega de mensagens não desejadas (falsos positivos, ou simplesmente *spam*) e, ao mesmo tempo, evitar os *falsos negativos*, i.e., quando a difusão de uma mensagem falha alguns dos interessados.

O projecto LiveFeeds [1] está a desenvolver algoritmos P2P de difusão filtrada usando a disseminação cooperativa de *feeds RSS* como caso de estudo. Um dos

[1] Projecto PTDC/EIA/76114/2006, Project *LiveFeeds – P2P Dissemination of Web Syndication Content*, financiado pela FCT/MCTES.

objectivos deste projecto reside na optimização da difusão filtrada pela via de uma distribuição equitativa da carga pelos nós participantes.

LiveFeeds procura equacionar o problema da difusão filtrada como um problema de computação distribuída, em que a natureza interna dos filtros é relegada para segundo plano. O algoritmo LiveFeeds não produz falsos positivos, nem falsos negativos [2], entregando as mensagens a todos os nós interessados e apenas a esses. Além disso, o esforço despendido no encaminhamento de uma mensagem, que inclui a avaliação dos filtros, recai apenas nos nós a que ela se destina. Para tal, o algoritmo tem como requisito que cada nó participante tenha visibilidade completa e consistente da filiação do sistema. Esta característica enquadra o algoritmo LiveFeeds na filosofia *One Hop Routing* [5], exemplificado pelas DHTs que sacrificam a complexidade espacial das tabelas de encaminhamento para obter encaminhamento com custo constante. Trata-se de um requisito forte, porém a simplicidade e robustez que confere à solução para o problema da difusão filtrada justifica a abordagem que está a ser investigada.

O presente artigo pretende discutir a viabilidade da difusão filtrada LiveFeeds, através da análise do custo do algoritmo de filiação adoptado. Em concreto, pretende-se balizar o volume de tráfego investido pelos nós na manutenção de uma visão completa e consistente da filiação em função do dinamismo do sistema. Este será modelado em termos da taxa de entrada de novos nós e o seu tempo médio de vida no sistema.

Nas secções seguintes é descrito o algoritmo de filiação LiveFeeds e é feita uma análise teórica do mesmo. Segue-se uma discussão da validação e dos resultados obtidos por simulação. É, também, descrita e avaliada uma solução para o problema em que se traduz a entrada de um número elevado de participantes num curto espaço de tempo. O documento termina com as conclusões finais e uma breve descrição do trabalho que ainda resta desenvolver.

2 O algoritmo de filiação com visibilidade completa

O algoritmo utiliza a própria informação de filiação de cada nó para ir construindo árvores de difusão aleatórias com vista a disseminar rapidamente a entrada de novos nós pelo sistema. Um mecanismo epidémico complementar tem como função eliminar as inconsistências que inevitavelmente se produzem devido a entradas concorrentes e às ocasionais falhas dos nós.

Em concreto, cada nó LiveFeeds é conhecido por um identificador aleatório numérico (por hipótese, sem colisões e com uma distribuição uniforme), pelo seu endereço e por um filtro. O domínio dos identificadores está dividido num pequeno número de fatias (menos de uma dezena para cenários de utilização típicos). Em cada fatia, o nó com o identificador numericamente menor é, tendencialmente, conhecido como o líder da fatia (*slice leader*) e tem um papel destacado no algoritmo. Não existe qualquer tipo de coordenação entre os *slice leaders*. É ainda de notar que, momentaneamente, pode haver mais do que um

[2] Ignorando falhas nos nós, as quais exigem cuidados extra que não serão discutidos neste artigo.

slice leader em cada fatia, facto previsto pelo algoritmo. O sistema assume, também, que existe conectividade entre todos os nós.

Quando um nó pretende juntar-se ao sistema, fá-lo através de um nó já presente no sistema, conhecido a priori. Este nó auxiliar serve para encaminhar o pedido de entrada ao *slice leader* relevante, determinado com base no identificador do nó que pretende entrar. A função de cada *slice leader* consiste em agregar pedidos de entrada antes de iniciar a sua difusão pelos nós do sistema. Essa acumulação realiza-se até ser atingido um número razoável de eventos, ou até se esgotar o tempo máximo permitido de acumulação, fixado em cerca de 30 segundos. Nessa altura, é iniciada a difusão do evento de agregação pelo sistema através de uma árvore aleatória de grau G, gerada no decurso do processo de difusão. Para tal, o *slice leader* começa por dividir o universo dos identificadores em G sub-intervalos com a mesma cardinalidade (de nós). Para cada um deles, o *slice leader* selecciona um nó escolhido ao acaso, a quem entrega o evento de agregação a difundir e o sub-intervalo dos identificadores que ele, por sua vez, deverá tratar. O processo repete-se recursivamente, em cada nó da árvore aleatória assim gerada, até se chegar ao ponto em que os intervalos ficam tão estreitos que os nós que restam podem ser tratados simplesmente como folhas. Um nó sabe que entrou no sistema quando recebe a mensagem que difunde a sua entrada no sistema. Se tal não acontecer em tempo útil, ele repete todo o processo de entrada.

Sendo o processo de difusão anterior, essencialmente, *best-effort*, a visão da filiação do sistema que os nós assim adquirem é, inevitavelmente, imperfeita. Falhas ocasionais de nós e eventos de difusão concorrentes geram inconsistências que têm que ser corrigidas. Para tal, cada novo evento agregado de filiação, iniciado por um *slice leader*, é identificado por um número de sequência local único e uma estampilha (vectorial) global. Esta última tem como função registar o evento de filiação emitido mais recentemente por cada um dos *slice leaders* do sistema. Cada uma destas estampilhas globais representa uma ”vista‘ concreta da composição do sistema, que cada nó pode comparar com a sua própria visão, computada a partir dos eventos de filiação que efectivamente recebeu, a fim de detectar eventos perdidos. A recuperação de eventuais omissões faz-se por epidemia. O processo é simples e consiste em periodicamente seleccionar um nó escolhido ao acaso e enviar-lhe a estampilha vectorial que identifica os eventos já vistos, na esperança de receber na volta os eventos em falta.

Note-se, ainda, que assim que um nó é tido (por qualquer outro) como tendo entrado no sistema, passa a poder se seleccionado para nó interior de uma árvore de difusão. Tal facto obriga que cada nó, previamente à sua entrada, obtenha uma cópia razoavelmente completa e actualizada da informação de filiação do sistema. Com o objectivo de reduzir a dimensão dessa informação e, de forma relacionada, reduzir a dimensão das estampilhas vectoriais usadas na recuperação epidémica, é imposto um limite máximo ao tempo de sessão dos nós. Porém, o impacto destas medidas e o seu custo ainda estão a ser avaliados e estão fora do âmbito deste documento, o qual se centra exclusivamente no custo do processo de difusão da informação de filiação em função do dinamismo do sistema ignorando o custo da reparação das falhas.

Sobre a visão completa e consistente, assim obtida, é implementado o processo de difusão filtrada. Essencialmente, trata-se de uma pesquisa distribuída dos nós cujos filtros aceitam a mensagem, ficando cada nó responsável por realizar uma parte do trabalho que resta, representado sob a forma de um intervalo de identificadores. Essa pesquisa faz-se através uma árvore aleatória que cobre apenas os nós cujo filtro aceita a mensagem em questão, ou seja apenas estes realizam esta computação distribuída.

A geração da árvore tem várias semelhanças com o processo de difusão da informação de filiação, sendo a principal diferença a parte que trata da selecção dos nós de cada nível inferior da árvore. Nomeadamente, geram-se também G' sub-intervalos. Porém, em vez de seleccionar um nó ao acaso para cada sub-intervalo obtido, esses sub-intervalos são processados sequencialmente até ser encontrado (em cada um deles) um nó, se houver, que aceite a mensagem. Os G' nós (no máximo) assim produzidos irão receber a mensagem a disseminar, mais o respectivo sub-intervalo que ainda restar. Para garantir a aleatoriedade da árvore, antes da proceder à divisão inicial, é primeiro adicionado um deslocamento aleatório.

Com este algoritmo não é produzido *spam*. A menos da ocorrência de falhas nos nós, também não há falsos negativos, pois quando um evento chega a um nó é possível atrasar o seu processamento até que esse nó chegue a uma vista de filiação igual ou posterior à estampilha de emissão da mensagem.

3 Análise do custo da difusão usando árvores aleatórias

Começaremos por analisar o custo da difusão global (*broadcasting*) num sistema estável (i.e, sem *churn*) com N nós. O objectivo é ter um ponto de partida que permita estimar os valores da capacidade de *upstream* e de *downstream* que cada nó terá de investir na implementação do algoritmo de filiação.

Por hipótese, vão ser difundidas continuamente mensagens, ao ritmo de r mensagens por segundo, por todos os N nós do sistema. Cada uma das mensagens tem a dimensão de m_t bytes. Pretende-se estimar a capacidade de *upstream* e de *downstream* que tal difusão requer. Quando um nó a pretende difundir uma mensagem m a todos os outros nós do sistema, selecciona um nó aleatoriamente, nó c, para ser a raiz da árvore de difusão. c dá então início à construção de uma árvore de difusão aleatória com grau G, processo que terá lugar simultaneamente com a própria difusão.

Para realizar esta análise, um primeiro passo consiste em determinar qual a probabilidade, $P(folha)$, de um nó ser folha e $1 - P(folha)$ de ser nó interior numa árvore de difusão. Para simplificar a análise, vamos admitir que as árvores usadas pelo algoritmo de difusão são equilibradas e todos os nós interiores têm grau G. Nestas condições, sendo h a altura da árvore de difusão formada, N satisfaz:

$$N = \sum_{i=0}^{h} G^i = \frac{G^{h+1} - 1}{G - 1} \tag{1}$$

$$P(folha) = \frac{G^h}{N} = \frac{G^h(G-1)}{G^{h+1}-1} = \frac{G^{h+1}-G^h}{G^{h+1}-1} \approx \frac{G^{h+1}-G^h}{G^{h+1}} = 1 - \frac{1}{G} \quad (2)$$

O tráfego de cada nó, aquando da recepção de uma mensagem de notificação, depende da posição do nó na árvore de difusão. As folhas da árvore só têm de receber a mensagem de notificação e não necessitam de a encaminhar para G nós, como acontece com os nós interiores. Um nó interior recebe m_t e envia $G.m_t$, enquanto uma folha não envia mensagens e recebe m_t.

A capacidade de *upstream* (b_u) e *downstream* (b_d) requerida em média por cada nó é dada por:

$$b_u = P(interior) \times G.m_t \times r = \frac{1}{G} \times G.m_t \times r = m_t \times r \quad (3)$$

$$b_d = m_t \times r \quad (4)$$

Podemos assim concluir que, em média, todos os nós considerados contribuem com o mesmo tráfego de *upstream* e *downstream*, pois a probabilidade de um nó ser folha é inversamente proporcional ao grau da árvore.

Dado que a análise anterior foi realizada com base numa restrição sobre os valores possíveis do número de nós do sistema (cf. a equação 1) e pressupondo que a árvore de difusão é completa e equilibrada, foram realizadas um conjunto de simulações que permitissem aferir se este resultado é uma boa aproximação dos requisitos do algoritmo de difusão no que diz respeito à capacidade usada na rede.

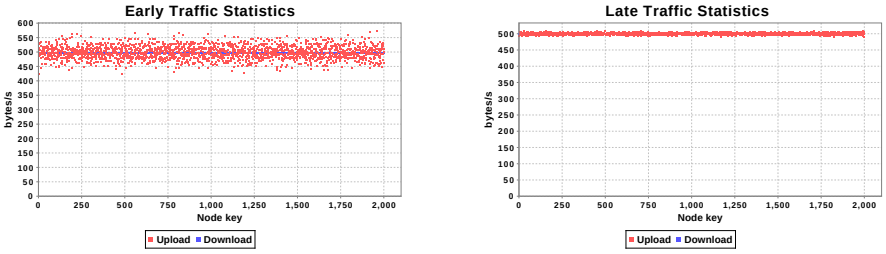
As simulações consistiram na criação de um sistema com N nós, com visibilidade completa e um tempo de sessão infinito, sobre o qual se desencadearam difusões totais, com raiz aleatória e à taxa de uma mensagem por segundo, de dimensão 500 Bytes. Os gráficos apresentados na figura 1 correspondem a uma dessas experiências, amostradas em dois momentos distintos. Esta e outras simulações, para vários valores de N, confirmaram que os valores médios do tráfego de *download* são os esperados e que os valores médios do tráfego de *upload* convergem para o mesmo valor com o número de difusões realizadas.

4 Custo da difusão da filiação num sistema dinâmico

Num sistema P2P real, os nós entram e saem do mesmo. Estes eventos devem repercutir-se em modificações da tabela de encaminhamento dos outros nós. No sistema em análise, os eventos que são relevantes são as entradas, na medida em que os eventos de saída não são propagados [3].

As entradas e saídas são eventos geralmente independentes e a repetição de eventos deste tipo cria um efeito que se designa por *churn* [11], o qual caracteriza

[3] Por hipótese, nesta fase do desenho, cada nó marca como inacessíveis os nós com que não consegue comunicar.



(a) (b)

Figura 1: Tráfego médio usado por um sistema de 2000 nós, no momento $t = 20$ segundos (a) e no momento $t = 3600$ segundos (b)

a dinâmica das alterações de filiação [3,9]. Um modelo de *churn* é composto por duas componentes. Uma distribuição que modela o tempo entre duas chegadas consecutivas de novos nós (*inter arrival time*) e uma distribuição que modela o tempo de sessão de cada nó [4]. Muitos sistemas P2P são avaliados com base em modelos simplistas, que usam distribuições uniformes do intervalo de chegada consecutiva de novos nós, e valores constantes do tempo de sessão. Isso pode ser enganador porque existem situações de *flash-crowds*, isto é, situações em que existe uma grande afluência de utilizadores num curto espaço de tempo (e que podem também ser caracterizadas por tempos de sessão muito curtos), correspondendo a um elevado *churn* que o sistema tem suportar. No contexto do LiveFeeds, uma *flash-crowd* pode corresponder a uma altura em que acontece algo importante e os utilizadores entram no sistema para procurar notícias e artigos sobre esse assunto, mas assim que obtêm o conteúdo pretendido abandonam o sistema.

Neste estudo procurou-se colmatar esse defeito de estudos anteriores (cf. [6,10]) usando um modelo mais realista. Em [4] é examinada a rede *skype* e é tanto quanto sabemos o único estudo que fornece alguma informação sobre o comportamento dos utilizadores nesta rede, que se pensa ter utilizadores que têm um comportamento semelhante aos possíveis utilizadores do *LiveFeeds*. Neste trabalho, é referido que na rede *skype* a entrada e saída de utilizadores depende fortemente da altura do dia e da altura da semana, havendo mais entradas durante a manhã, mais saídas ao fim do dia e mais utilização da rede em dias úteis. Esta ideia é também suportada por [12] que estuda o comportamento dos utilizadores de uma rede de *instant messaging* e por [8] que estuda o comportamento dos utilizadores de *e-mail*.

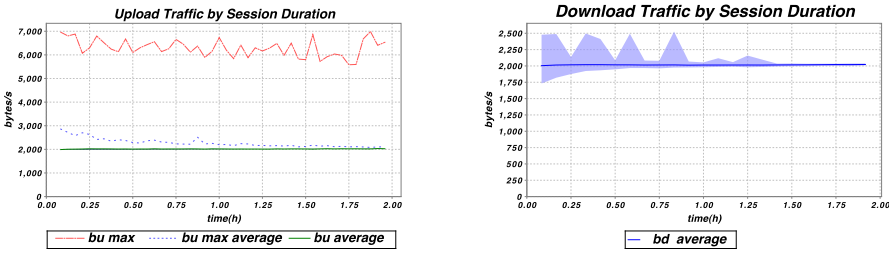
A partir destes estudos devemos modelar as chegadas de utilizadores através de um processo de *Poisson* não homogéneo, uma vez que a quantidade de entradas depende das alturas do dia e da semana, e modelar o tempo de sessão através de uma distribuição *heavy-tailed* [4], que de acordo com [12] pode ser uma distribuição tipo *Weibull*. Uma vez que apenas se pretende estudar o comportamento do algoritmo de filiação no pior caso possível, durante *flash-crowds*, é apenas necessário usar uma distribuição de *Poisson* homogénea que modele os valores de tempo entre chegadas consecutivas durante a pior altura.

O estudo dos requisitos do algoritmo de filiação para difundir as alterações da mesma, foi realizado através da simulação do algoritmo, fazendo vários testes, em que o sistema ia crescendo continuamente, com os parâmetros apresentados na tabela 1.

Tabela 1: Parâmetros usados na simulação

Parâmetro	Valor
G	4
Número de fatias	4
Periodicidade das difusões pelos SLs	25 a 30s ou 15 a 20 msgs
Tamanho de uma mensagem de filiação	500 Bytes
Tempo de sessão	300 a 7200 segundos
Distribuição de tempo de sessão	*Weibull* com $\lambda = 5000$ e $k = 0,5$
Distribuição dos tempo entre chegadas	*Poisson* com $\lambda = 1/2$, $1/4$ e $1/10$
Tempo de amostragem	25 segundos

Durante a simulação é registado o tráfego total enviado e recebido por cada nó. Estes dados permitem calcular a capacidade média (b_u *average* e b_d *average*) requerida por nó para execução do algoritmo. São também realizadas amostragens a cada 25 segundos, o que ainda permite determinar o valor máximo observado (b_u *max*) em cada período de amostragem e a média destes valores (b_u *max average*) para cada nó. A figura 2 apresenta os valores máximos observados (b_u *max*) no sistema e os médios das restantes grandezas acima descritas, para nós agrupados por tempo de vida.



(a) Tráfego de *upstream* (b) Tráfego de *downstream*

Figura 2: Resultados obtidos com distribuição do tempo entre chegadas parametrizada com $\lambda = 1/4$ (valor médio $r = 4$ *eventos*$/s$) e tempo de sessão máximo de 2 horas

Pela figura 2 observa-se que os dados obtidos para os valores médios estimados no fim de vida de cada nó são coerentes com os resultados teóricos descritos na secção 3 ($b_u = b_d = r \times m_t = 4,0 \times 500 = 2000$ B/s), apesar de não estarmos perante exactamente as mesmas condições.

Os valores obtidos para b_u *max* mostram o pior caso possível que os nós tiveram que suportar, uma vez que representa a capacidade máxima utilizada,

de entre todos os nós com o mesmo tempo de sessão. Esta métrica fornece assim um ponto extremo para o cálculo dos limites do algoritmo simulado. Neste caso, houve pelo menos um nó que necessitou de suportar o envio de $7000B/s$ num intervalo de $t_s = 25s$, tendo esse nó permanecido no sistema cerca de $1,9$ horas.

Em relação aos valores para a banda passante de *downstream* (Figura 2 (b)) não se nota uma variação significativa nos valores observados. Embora existam algumas flutuações, o valor médio situa-se sempre nos $b_d = r \times 500 = 2000B/s$, correspondente ao valor esperado no cenário ideal analizado na secção 3. Pela observação deste gráfico corrobora-se também o resultado já referido de que em média e ao fim de um número muito grande de *broadcasts* o valor da banda passante de *downstream* será igual ao valor da banda passante de *upstream*, uma vez que se verifica que $b_u = b_d \approx 2000B/s$. Tal facto é posto em evidência pelo facto de os valores mínimo e máximo de b_d *average* se aproximarem de b_d *average* com o aumento do tempo de sessão (c.f. área poligonal sombreada em (2)).

5 Introdução de super-nós

As análises anteriores mostram que os requisitos de comunicação do algoritmo são lineares com o ritmo médio de entrada de novos nós. Esta ideia foi em geral omitida nas anteriores análises de algoritmos com aproximação semelhante, c.f. por exemplo [6,10]. Tal custo não escala com aquele ritmo pelo que se torna necessário melhorar a escalabilidade do algoritmo.

Existem várias vias para melhorar este aspecto. No que se segue exploraremos uma que mantém a linearidade mas diminui a derivada e baseia-se na noção de "super-nó".

Este algoritmo suporta-se na ideia de que existem nós que são responsáveis pelos novos nós do sistema, os nós ditos *filhos* só se tornam super-nós e a sua entrada só é difundida para os todos os outros super-nós do sistema após algum tempo passado desde a sua entrada. Atenua-se assim o problema dos nós que estão muito pouco tempo dentro do sistema e provocam um aumento do *churn* observado, uma vez que cada entrada teria de ser difundida por todos os nós do sistema, mesmo que o nó que acaba de entrar tenha um tempo de sessão muito pequeno. Utilizando o novo algoritmo, a entrada de um novo nó só é difundida para todos os nós do sistema apenas quando ele se torna super-nó.

Introduz-se assim uma hierarquização do sistema, pois passam a existir nós de dois tipos distintos, os nós e os super-nós, sendo criada uma rede em que todos os super-nós conhecem os outros super-nós mas apenas conhecem os seus filhos.

No algoritmo reformulado, um nó a, para entrar no sistema, contacta um super-nó, s, enviando um pedido para que s seja pai de a. s deve decidir se pode aceitar mais um nó como filho ou reencaminhar a para outro nó. O processo repete-se até que algum nó aceite a como filho.

Após a ter encontrado um pai, deve aguardar uma mensagem de promoção vinda do pai, para que se junte à rede de super-nós. O filho fica assim a depen-

der do super-nó para receber mensagens referentes a acontecimentos na rede de super-nós.

Após algumas tentativas recusadas, o filho pode forçar a sua entrada. Ao receber um filho nestas condições, um super-nó deve promover um dos seus filhos para poder acomodar o novo nó. O critério que o super-nó utiliza para seleccionar o filho que deve promover tem grande influência no comportamento do sistema. Ao receber uma mensagem de promoção, o filho deve iniciar a rotina de entrada na rede de super-nós.

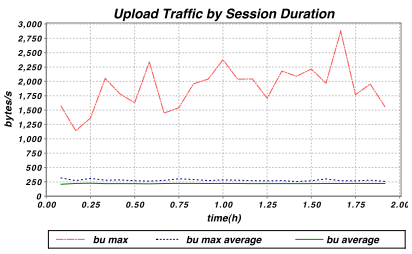
Como primeira aproximação, foram obtidas algumas simulações para que se pudesse determinar se uma aproximação deste tipo é viável. As simulações foram executadas utilizando os mesmos parâmetros utilizados durante as simulações referentes ao algoritmo normal (c.f. secção 4), bem como as mesmas métricas, de forma a que seja possível fazer uma comparação entre os dois algoritmos. Os resultados destas simulações são apresentados na figura 3. Uma vez que são medidos os valores considerados apenas para os super-nós, é considerado o tempo de sessão como super-nó e não o tempo de sessão total.

As simulações apresentadas na figura 3 correspondem a um caso em que ao entrar no sistema, um nó sabe sempre qual dos super-nós pode aceitar mais um filho e em que os super-nós sabem sempre qual dos seus filhos vai ficar mais tempo no sistema, promovendo esse nó sempre que seja necessário promover um dos seus filhos. Desta forma, o caso representado nesta simulação consiste no melhor caso possível.

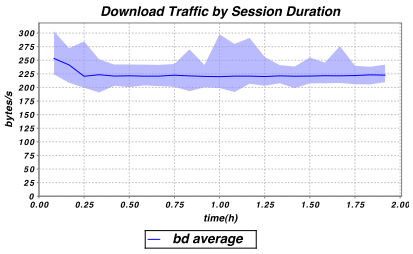
Como se pode notar pelo gráfico(a) da figura 3, o valor do tráfego de *upstream* utilizado pelos nós desceu significativamente, pois passa a depender linearmente do número de entradas por segundo na rede de super-nós e não do número de entradas por segundo em todo o sistema. A mesma descida é verificada ao observar o gráfico (b). É importante verificar que a igualdade entre o tráfego de *upstream* e *downstream* continua a ser observável, em média e ao fim de algum tempo, a quantidade de informação recebida é igual à quantidade de informação enviada.

Podemos assim concluir que a introdução de uma hierarquia deste tipo no sistema contribuiu de forma positiva para a diminuição do *churn* observado pela rede de super-nós e que esta aproximação tem assim um grande potencial de melhoramento.

Não havendo um oráculo perfeito para encontrar um super-nó disponível e para seleccionar o melhor filho a promover, como discutido anteriormente, a figura 4 apresenta os resultados obtidos para condições mais realistas. Nestas simulações, tanto a selecção do super-nó semente como dos filhos a promover são feitas através de um processo estocástico. Como seria expectável, comparativamente ao cenário correspondente à figura 3, observa-se um aumento da capacidade utilizada. Porém, pode-se verificar que continua a existir um ganho no que se refere ao tráfego de *upstream* e *downstream* que cada nó despendeu para difundir a informação de filiação dos novos super-nós, face ao algoritmo de visibilidade completa.

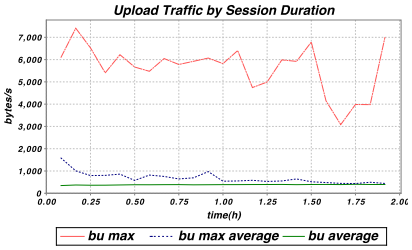


(a) Tráfego de *upstream*

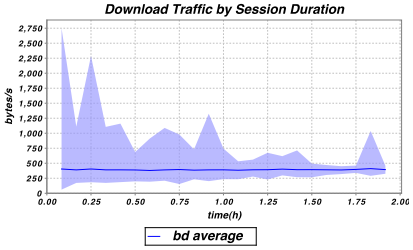


(b) Tráfego de *downstream*

Figura 3: Resultados obtidos utilizando o algoritmo de super-nós configurado de forma equivalente à dos ensaios da secção 4 perante condições óptimas de selecção de nós semente e nós a promover



(a) Tráfego de *upstream*



(b) Tráfego de *downstream*

Figura 4: Resultados obtidos utilizando escolha aleatória de nós semente e selecção aleatória de filhos a promover

Outros melhoramentos possíveis

Um dos melhoramentos a introduzir seria a implementação de novos critérios para a selecção do filho a promover quando é necessário introduzir um novo nó na rede de super-nós. Estes critérios podem tomar um papel de grande relevo no melhoramento do algoritmo. Um critério possível seria dividir os nós em classes, baseados em informação como a utilização de *NAT* por parte do nó, qual a capacidade estimada do nó ou outros factores que possam ser determinantes e seleccionar o nó mais apto para ser promovido para super-nó.

A noção de classes pode ainda ser desenvolvida para um outro modelo em que cada nó utiliza um critério de ordenação (*ranking*) para decidir qual dos seus filhos deve promover a super-nó. A posição de um nó poderia ser determinada em função da classe do nó e, também, através de conhecimento de sessões passadas do nó em questão. Nestes moldes, um nó precisaria de guardar de forma persistente vária informação sobre os nós, nomeadamente informação sobre duração de sessões passadas.

A introdução de novos nós na rede de super-nós pode ser feita de forma adaptativa de maneira a que os super-nós possam ter algum controlo sobre o número de entradas por segundo observadas. Podemos partir da ideia de que um super-nó tem conhecimento total dos eventos dentro da rede de super-nós e pode assim decidir se o número de entradas observadas no passado próximo é

baixo o suficiente para que a rede suporte uma nova entrada sem que exista uma sobrecarga. É preciso no entanto ter em consideração medidas para que o *feedback* proveniente deste tipo de mecanismo seja evitado. Se não forem implementadas tais medidas, vários nós podem detectar um baixo número de eventos de entrada e promover simultaneamente vários nós, levando a um aumento excessivo do número de entradas por segundo.

Ao ter de gerir filhos, um super-nó terá de utilizar mais capacidade de *upstream*, mas essa desvantagem poderá ser anulada pela redução do número de entradas observadas que levam a uma diminuição da capacidade necessária. Para reduzir mais a desvantagem que um super-nó tem ao ser responsável por outros nós, pode ser desenvolvido um mecanismo em que os filhos fazem o envio de mensagens no lugar do super-nó, participando na árvore de difusão no lugar do seu pai. O trabalho do super-nó pode ainda ser diminuído se os filhos difundirem entre si as mensagens vindas do super-nó.

6 Conclusões e trabalho futuro

Com base na análise feita, podemos tirar algumas conclusões sobre a aplicabilidade do algoritmo de filiação LiveFeeds com visibilidade completa. A utilização de árvores de difusão aleatórias tem como benefício que a capacidade de *upstream* necessária é proporcional ao número de eventos por segundo observada no sistema. A capacidade que cada nó deve suportar para que o mecanismo considerado possa ser utilizado não depende assim do tamanho do sistema, mas sim do comportamento dos seus utilizadores. Se esse comportamento for estável o suficiente, o mecanismo de visibilidade completa pode ser utilizado com um grande número de nós.

Neste sentido, podemos considerar o uso do mecanismo de visibilidade completa num sistema em que os nós comportam-se como *brokers* e poderão estar alojados em instituições de ensino, *ISPs*, grandes empresas, etc. e que servem os *feeds* aos utilizadores. Este tipo de nós têm tipicamente boa conectividade e tempos de sessão longos, o que leva a que, num sistema constituído maioritariamente por este tipo de nós, a taxa de eventos por segundo seja baixa o suficiente para viabilizar que seja suportado um grande número de nós. A título de exemplo, se considerarmos uma rede com $N = 10000$ nós, em que os nós têm um tempo sessão médio de cerca de 3 horas, isto equivale a uma taxa de eventos de cerca de $10000/10800 \approx 0,9$ eventos/segundo. Os nós teriam assim de contribuir com uma capacidade de *downstream* e *upstream* de $r \times m = 0,9 \times 500 = 450$ bytes por segundo para manter a informação de filiação, sendo este um custo bastante baixo para nós do tipo considerado.

Porém, para que o sistema suporte utilizadores com um comportamento mais dinâmico, é necessário encontrar outras soluções, como a introdução de super-nós. Através de super-nós afigura-se possível resolver dois problemas complicados e ignorados em muitos sistemas P2P. São eles, o elevado custo dos nós com tempos de sessão muito curtos e a entrada de muitos nós num curto espaço de tempo.

Em contrapartida, ao limitar a visão completa do sistema aos super-nós perde-se alguma da simplicidade do algoritmo de difusão filtrada inicialmente pensado no projecto LiveFeeds. Para evitar falsos positivos torna-se necessário que cada nó, até ser promovido, esteja sob a alçada de um super-nó cujo filtro seja pelo menos igual ao seu, ou em alternativa mais abrangente. Dado o conhecimento alargado que todos os super-nós possuem, esse processo terá custos de comunicação mínimos, mas obriga que o algoritmo de filiação passe a ter algum conhecimento da composição interna dos filtros.

Em termos de trabalho futuro, há ainda alguns estudos a realizar. Além de uma análise cuidada dos outros melhoramentos propostos na sub-secção 5, é preciso estudar as questões relacionadas com a dimensão da informação de filiação que cada nó precisa de receber antes de entrar no sistema. Caso o problema se revele excessivamente sério, tudo indica que uma forma de minimizar essa informação passará por dar ainda mais relevância aos super-nós, aumentado o número dos nós que deles dependem. E, como também já foi sugerido, entregando a esses nós alguma da responsabilidades de encaminhamento das mensagens de filiação.

Referências

1. A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *INFOCOM*, 2004.
2. P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.
3. P. B. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. *SIGCOMM Comput. Commun. Rev.*, 36(4):147–158, 2006.
4. S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, pages 1–6, 2006.
5. A. Gupta, B. Liskov, and R. Rodrigues. One hop lookups for peer-to-peer overlays. In *Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX)*, pages 7–12, Lihue, Hawaii, May 2003.
6. A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. *Proc. First Symposium on Networked Systems Design and Implementation*, 2004.
7. J. Legatheaux Martins and S. Duarte. Routing Algorithms for Content-based Publish/Subscribe Systems. *IEEE Communications Tutorials and Surveys – Accepted for Publication*, page 21, 2009.
8. R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. N. Amaral. A Poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences*, 105(47):18153–18158, 2008.
9. S. C. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. *Handling Churn in a DHT*. Computer Science Division, University of California, 2003.
10. R. Rodrigues and C. Blake. When Multi-Hop Peer-to-Peer Lookup Matters. In *In Proc. of IPTPS*, pages 112–122, 2004.
11. D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, New York, NY, USA, 2006. ACM.
12. Z. Xiao, L. Guo, and J. Tracey. Understanding instant messaging traffic characteristics. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, page 51. IEEE Computer Society Washington, DC, USA, 2007.