Reduzindo o Tráfego Gerado por Aplicações Par-a-Par para Distribuição de Vídeo sob-Demanda na Internet

Josilene Moreira[1], Petrônio Gomes[1], Victor Souza[2], Djamel Sadok[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brazil

[2]Ericsson Research - Stockholm, Sweden

{josilene,petronio,jamel}@cin.ufpe.br, victor.souza@ericsson.com

Abstract. *New strategies for Video on-Demand distribution (VoD) have been proposed, including the use of peer-to-peer (P2P) networks. However, this strategy transfers the cost of distribution from content providers to service providers (ISPs). One solution to minimize this cost is the use of caches that store the P2P traffic and try to keep it local to the ISP. This paper evaluates the use of caches for P2P VoD traffic, proposing an optimization for a partial cache algorithm that improves the performance of the original algorithm up to 12.7%. The major contribution is to demonstrate that the use of cooperation between ISPs improves the efficiency of the cache in more than 100%, significantly reducing transit traffic (12.6%).*

Resumo. *Novas estratégias de distribuição de Vídeo sob-demanda (VoD) têm sido propostas, entre elas o uso de redes par-a-par (P2P). Entretanto esta estratégia transfere o custo de distribuição dos provedores de conteúdo para os provedores de serviço de Internet (ISPs). Uma das soluções para minimizar este custo é o uso de caches que armazenam o tráfego P2P e procuram mantê-lo local ao ISP. Este artigo avalia o uso de caches para tráfego P2P VoD, propondo uma otimização de um algoritmo de cache parcial que melhora o desempenho do algoritmo original em até 12,7%. A maior contribuição é demonstrar que o uso de cooperação entre ISPs melhora a eficiência da cache em mais de 100%, reduzindo significativamente o tráfego de trânsito (12,6%).*

Palavras-chave*: estratégias de cache, caches cooperativas, VoD*

1. Introdução

Nos dias de hoje, a popularidade de sistemas Par-a-Par (P2P) para distribuição de conteúdo tem aumentado rapidamente. Esta classe de aplicações é responsável pela maior parte do tráfego gerado entre ISPs (Provedores de Serviço Internet) e o crescente uso desse tipo de rede de compartilhamento sugere uma tendência de aumento ainda maior no futuro [Hefeeda and Saleh 2008].

Recentemente, além de ser usada para compartilhamento de arquivos, a tecnologia P2P também tem sido largamente utilizada em aplicações de distribuição de vídeo na Internet, tanto ao vivo (*Live Stream*) como sob-demanda (*Video on-Demand*). Diversas aplicações de origem chinesa como *PPLive*, *TVU*, *SOPCast* e *PPStream*, entre

outras, utilizam esta tecnologia. Adicionalmente, algumas redes de distribuição de conteúdo (*CDNs*) também têm usado a tecnologia P2P para alavancar seus negócios e diminuir os custos de distribuição. Entre elas estão a *Velocix*[1] , a *GridNetworks* (recentemente incorporada pela *Global Media Services*[2] e a *Akamai*[3], que adquiriu a empresa especialista em P2P *Red Swosh* em 2007 e começou a oferecer serviços baseados nesta tecnologia em 2009[4].

Entretanto, os Provedores de Serviço de Internet (ISPs) não estão satisfeitos com o uso da tecnologia P2P. A principal razão é que os custos de distribuição estão sendo transferidos dos provedores de conteúdo para os ISPs [Blond et al. 2008] [Hefeeda and Saleh 2008] [Karagiannis et al. 2005] . Como a maioria dos protocolos P2P não está preocupada em selecionar nós dentro do mesmo ISP, o uso destas aplicações tem gerado um custo de tráfego de trânsito bastante significativo para os ISPs. Embora existam acordos de troca de tráfego entre ISPs (enlaces de *peering*), os quais geralmente não incorrem em custos, é impossível que um ISP tenha acordos com todos os ISPs existentes. Normalmente, um ISP paga por enlaces de trânsito que o conectam indiretamente aos outros ISPs com os quais ele não possui conexão direta. Pesquisas recentes mostram que apenas cerca de 8% do tráfego gerado por aplicações P2P permanece interno ao ISP; 92% trafega por enlaces de trânsito e de *peering* [Shen et al. 2007]. Ressalta-se ainda que a maior parte dos *bytes* transferidos em aplicações P2P decorre de grandes objetos [Gummadi et al. 2003] [Leibowitz et al. 2002] .

Com o objetivo de reduzir este tráfego, os ISPs têm tomado iniciativas como o bloqueio de aplicações P2P. Esta solução não é tão simples, visto que pode depender de técnicas sofisticadas tanto de *hardware* como de *software* e, mesmo assim, as aplicações sempre estão tentando alguma forma de esquivar-se à detecção dos seus protocolos [Feitosa et al. 2008]. Uma segunda alternativa é o uso de *caches* a fim de melhorar a localidade de tráfego, as quais atuam interceptando as requisições dos nós clientes [Dán 2009] [Karagiannis et al. 2005]. As *caches* geralmente usam o mesmo protocolo do sistema P2P e funcionam como um nó de grande capacidade que "atrai" as requisições dos usuários por ter grande capacidade e largura de banda. O conteúdo primeiro é buscado internamente ao ISP e só será acessado externamente se não for encontrado na *cache* ou se a sua capacidade for excedida.

Este trabalho estuda especificamente a contribuição das *caches* para a redução do tráfego P2P em sistemas de distribuição de Video sob-Demanda (VoD). Um sistema de vídeo geralmente distribui objetos bem maiores do que um sistema convencional de compartilhamento de arquivos como *Kazaa* e *BitTorrent*, e portanto seu uso intenso impacta fortemente no perfil de tráfego de um ISP. Apresenta-se um algoritmo de *cache* parcial modificado que melhora a eficiência da *cache* em 12,7% sobre o algoritmo original, através do armazenamento dos segmentos mais populares dos objetos [Hefeeda

---

[1] Velocix. http://www.velocix.com/. Acessado em 10/12/2009.

[2] BusinessWire.http://www.businesswire.com/portal/site/home/permalink/?ndmViewId=news_view&newsId=20090417005446&newsLang=en. Acessado em 05/04/2010

[3] Akamai. http://www.akamai.com/.Acessado em 15/04/2010

[4] Ligth Reading. http://www.lightreading.com/document.asp?doc_id=121710. Acessado em 12/12/2009.

and Saleh 2008] [Yu et al. 2006]. Entretanto, a principal contribuição deste trabalho é a avaliação da estratégia de cooperação entre *caches* de diferentes ISPs sobre a redução do tráfego de trânsito. O uso de *caches* cooperativas chega a melhorar o desempenho do algoritmo de *cache* em mais de 100%, aumenta o tráfego local em 128% e reduz o tráfego de trânsito em até 12,6%.

O artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, a Seção 3 descreve a metodologia utilizada nas avaliações, a Seção 4 apresenta os resultados e a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos relacionados

Algumas pesquisas recentes tratam das estratégias de redução de tráfego entre ISPs. Em [Karagiannis et al. 2005], os autores apresentam um dos primeiros trabalhos a avaliar o impacto do tráfego P2P sobre os ISPs, mostrando que os custos de distribuição estão sendo transferidos dos provedores de conteúdo para os provedores de serviço de Internet. As idéias do uso de *caches* e do conceito de localidade de tráfego são avaliadas a partir de coletas de tráfego BitTorrent[5] , concluindo que essas estratégias podem beneficiar significativamente os ISPs, reduzindo seus custos.

Em [Choffnes and Bustamante 2008], é proposta uma abordagem para reduzir o custo do tráfego inter-ISP sem que o desempenho do sistema P2P seja sacrificado. Na seleção, que preza por localidade, os pares se comunicam com vizinhos que teoricamente estão próximos utilizando as informações coletadas pelo mecanismo de redirecionamento de uma CDN (*Content Distribution Network*). Isso implica que não é necessária uma nova infra-estrutura e nem cooperação entre provedores de serviços de Internet para colocar em prática essa abordagem. Para avaliar a solução proposta foi realizada uma implementação de um cliente BitTorrent que tentam encontrar 'caminhos entre nós' que possibilitem reduzir o tráfego inter-ISP. No entanto, esta solução depende do uso de uma espécie de oráculo, que é a CDN, para encontrar os nós mais próximos.

Devido à tensão existente entre aplicações P2P e ISPs, os autores em [Shen et al. 2007] propuseram uma nova estratégia chamada de HPTP, *HTTP-based Peer-to-Peer*. Essa técnica propõe a utilização de *caches* já existentes nos ISPs usadas para o armazenamento de tráfego *web* para alternativamente armazenar tráfego P2P. Para isso, os autores descreveram um processo denominado '*HTTPifying*', que consiste na segmentação dos arquivos P2P em pedaços menores para serem encapsulados, transportados e tratados pelas *caches* como tráfego HTTP. Nesse trabalho, por meio de simulações, ganhos significativos foram identificados, como a redução da carga de tráfego entre ISPs e no *backbone* da Internet sem comprometer a aplicação P2P envolvida.

Em [Hefeeda and Saleh 2008], um estudo sobre características relevantes do tráfego P2P foi realizado, avaliando o benefício do uso de *caches*. O tráfego P2P foi coletado e caracterizado, analisando-se as distribuições de popularidade dos objetos. Posteriormente, um algoritmo de *cache* parcial foi proposto, baseado na modelagem realizada anteriormente. Seguindo a idéia de *cache* para objetos *web*, o algoritmo visa minimizar os principais problemas provenientes do tráfego P2P. As simulações mostram

[5] BitTorrent. http://www.bittorrent.com. Acessado em 15/10/2009.

altos índices de desempenho, sendo possível constatar a importância do algoritmo de *cache* parcial para esse tipo de tráfego. Este algoritmo serviu como base para as alterações e otimizações propostas pelo nosso trabalho.

Apesar de ter um grande potencial, esquemas de caches cooperativas envolvendo tráfego P2P não têm sido muito abordados na literatura. Em [Hefeeda and Noorizadeh 2008], é defendida a idéia de que o esquema de caches cooperativas é mais útil ao tráfego P2P que ao tráfego mais comum da *web*, pois os objetos P2P são repetitivos [Leibowitz et al. 2002] e apresentam pouquíssimas mudanças, sendo considerados praticamente imutáveis por [Gummadi et al. 2003]. Em [Dán 2009], o autor propõe um esquema de *caches* cooperativas compatível com as relações de negócio existentes entre ISPs, onde o problema é modelado usando a teoria dos grafos. Os resultados apresentados demonstram matematicamente a capacidade do esquema de *caches* cooperativas de minimizar os custos oriundos do tráfego P2P para os ISPs. O estudo é realizado para vídeo ao vivo e mostra uma análise matemática, não utilizando simulação das requisições para os objetos armazenados nas *caches* como faz o nosso trabalho.

Em [Hefeeda and Noorizadeh 2008], também é analisado o potencial de *caches* cooperativas para a redução do tráfego de trânsito causado por aplicações P2P. São propostos dois modelos de cooperação entre *caches* de diferentes ASs e *cache*s de um mesmo AS. Um *trace* com oito meses de duração foi coletado para clientes *Gnutella*[6] e diversas simulações foram executadas, alternando entre os dois modelos propostos. Por fim, foram apresentados os resultados que destacam a relevância da cooperação entre *caches* e o *overhead* gerado. Nosso trabalho é complementar a este, pois explora especificamente o potencial de *caches* para tráfego de VoD, além de usar simulações específicas que reproduzem as requisições de clientes para objetos de vídeo em *caches* cooperativas.

3. Metodologia

Para avaliação do desempenho das *caches* foram criadas carga de dados (*workloads*) sintéticas através do gerador *ProWGen* [Busari and Wiliamson 2002]. Este gerador permite modelar diversas características importantes de uma carga de dados tais como popularidade dos objetos, correlação temporal entre as requisições e tamanho médio dos objetos, proporcionando a reprodução de um conjunto de dados muito próximos da realidade.

A fim de reproduzir os cenários da maneira mais realista possível para as avaliações de desempenho, as cargas sintéticas foram construídas a partir das características de coletas de tráfego P2P reais (*traces*) descritas em [Hefeeda and Saleh 2008]. A construção das cargas sintéticas reproduz inicialmente as características de distribuição de popularidade, tráfego *cacheable* (passível de ser armazenado na *cache)* e requisições dos dois sistemas autônomos (ASs) mais representativos das coletas efetuadas. A partir dos cenários básicos destes dois ASs, estas características são então variadas, a fim de obter uma avaliação mais abrangente. As requisições dos usuários foram criadas para objetos de VoD e processadas de encontro às *cache*s usando os algoritmos propostos.

[6] Gnutella. http://www.gnutella.com. Acessado em 12/12/2009.

3.1.Cenários

Utilizando o *ProWGen*, foram simulados inicialmente dois sistemas autônomos (ASs) onde o tráfego e os objetos requisitados apresentam diferentes características de popularidade e de percentual de tráfego *cacheable* (Tabela 1).

Tabela 1. Característica dos cenários

	AS397	**AS95**
Número de usuários	9315	9316
Número de objetos	3000	3000
Número de blocos por vídeo	20	20
Número de requisições	186300	186320
Tráfego *cacheable*	48%	54%
Distribuição de popularidade	MZipf (0.62, 8)	MZipf (0.6, 50)
Tamanho dos vídeos	1 GB	1 GB

Estes ASs foram escolhidos por apresentarem características distintas e serem bastante representativos dos perfis de tráfego observados. Para o AS397 aproximadamente 4,5 TB (48%) do tráfego pode ser potencialmente armazenado pela *cache* (tráfego *cacheable*). Já no AS95, este tráfego é de 54%, o que representa cerca de 5 TB. Além disso, a distribuição de popularidade é diferente para os dois ASs. Nas Seções 4.2 e 4.3 analisamos como estas duas características podem impactar no desempenho do algoritmo.

3.2. Requisições dos usuários

As requisições de objetos de VoD são distintas daquelas de compartilhamento de arquivos em sistemas P2P. Nos principais algoritmos de VoD-P2P como PPLive [Huang et al. 2008], o controle de quais segmentos do vídeo serão requisitados é de responsabilidade dos nós usuários. Nestes sistemas, os usuários assistem ao vídeo enquanto este está sendo baixado, com um certo atraso inicial (*startup delay*) e certa falha de continuidade (*playback continuity*). A aplicação é encarregada de escalonar quais segmentos o usuário precisa baixar a fim de minimizar atrasos. Neste trabalho as requisições dos objetos foram geradas para corresponder àquelas realizadas em sistemas de VoD.

Objetos de vídeo geralmente são maiores do que objetos *web* [Leibowitz et al. 2002]. Um objeto é composto por um determinado número de segmentos, que é a menor unidade manipulada pela *cache*. As requisições dos objetos são feitas em blocos compostos por *n* segmentos. Nos cenários dos experimentos, cada bloco tem um tamanho fixo de 50 segmentos, cada segmento com 1MB. Dessa forma, um objeto de 1GB é composto por mil segmentos de 1MB, e é requisitado em 20 blocos de 50 segmentos. As requisições são geradas seqüencialmente para os blocos do mesmo objeto, considerando que o usuário assiste ao vídeo do início ao fim (não são consideradas operações de adiantar ou retroceder a reprodução).

Apesar das requisições serem geradas seqüencialmente para os blocos de um mesmo vídeo, na Seção 4.4 isola-se o efeito da seqüencialidade, analisando-se a correlação temporal sobre o desempenho do algoritmo, gerando requisições com correlação temporal fraca, média e forte.

3.3. Algoritmo de *cache*

Através da análise de um algoritmo de armazenamento parcial apresentado na Figura 1, o qual propõe a escolha dos segmentos a serem armazenados de acordo com a popularidade e o tamanho dos objetos, verificou-se que o algoritmo continha algumas falhas graves e que o desempenho da *cache* poderia ser otimizado. O algoritmo original determina a quantidade e quais segmentos serão armazenados na *cache*. A escolha é realizada dinamicamente e atualizada a cada vez que o objeto é requisitado.

A função de utilidade do objeto *i*, baseada em sua popularidade, é denotada por *Yi* enquanto que a função de utilidade do objeto mais popular é armazenada em *Y*. A filosofia do algoritmo é manter na *cache* um número de segmentos diferente para cada objeto, proporcionalmente à sua popularidade.

```
SE objeto i ∉ cache
        insira um segmento do objeto i na cache, removendo caso necessário
SENÃO
        hit = segmentos de i na cache ∩ segmentos requisitados
        Yi = Yi + (hit/segmentos de i na cache)
        cache miss = (segmentos requisitados – hit)/(tamanho do segmento)
        x = (Yi/Y)*(tamanho médio dos objetos da rede)
        k = min(cache miss, max(x, 1))
        SE espaço ocupado da cache + k segmentos > tamanho da cache
                remova k segmentos do objeto menos popular
        adicione k segmentos do objeto i a cache
FIM
```

Figura 1. Algoritmo básico

No entanto este algoritmo apresenta uma falha grave, pois um objeto *i* que foi muito popular em um determinado período não tem sua função de utilidade decrementada. Este fato pode gerar duas conseqüências; a primeira é que o objeto *i* tardará muito a sair da *cache.* A segunda é que um objeto *j* que esteja se tornando muito popular terá que ultrapassar o valor da função de utilidade do objeto *i* que já foi mais popular um dia para que seja considerado o mais 'valioso' (com a maior função de utilidade) da *cache*. Isto pode não acontecer nunca, e assim o objeto será mantido por um período muito longo na *cache*, diminuindo o seu desempenho.

O algoritmo também pode ser melhorado com relação ao seu desempenho geral, através da alteração de sua política de remoção de objetos da *cache* e do cálculo dos segmentos a serem armazenados. Na verdade, com a aplicação destas modificações, podemos considerar que um novo algoritmo foi construído. As modificações realizadas para melhorar seu desempenho e corrigir a falha na temporalidade dos objetos mais populares são:

(i) Remoção de objetos *(eviction)*: quando é necessário remover algum item da *cache,* o objeto menos popular é identificado e os segmentos são retirados do fim para o início. O algoritmo original removia quaisquer segmentos do objeto menos popular.

(ii) Modificação do total de segmentos a ser armazenado: a variável *hit* deixa de ser proporcional ao número de segmentos do objeto na *cache* e é modificada para ser proporcional ao número de acertos. Isto faz com que o algoritmo responda mais rapidamente às alterações de popularidade dos objetos.

(iii) Introdução do GDS (*Greedy Dual Size*): no algoritmo original, um vídeo que foi um dia muito popular não tem a sua função de utilidade decrementada no decorrer do tempo. Desse modo, para que este vídeo possa sair um dia da *cache,* a cada vez que um vídeo é removido, o valor da sua função de utilidade é decrementado de todos os valores das funções de utilidade dos outros objetos que ainda estão na *cache*.

Percebeu-se que, com essas modificações, o algoritmo tornou-se mais "agressivo", adaptando-se mais rapidamente às mudanças de popularidade dos objetos e assim alcançando melhor desempenho, além de corrigir o problema da longa permanência de um objeto muito popular na *cache*.

3.4. Modelos de cooperação

Este estudo analisa a cooperação entre *cache*s de ISPs diferentes, conforme a Figura 2.

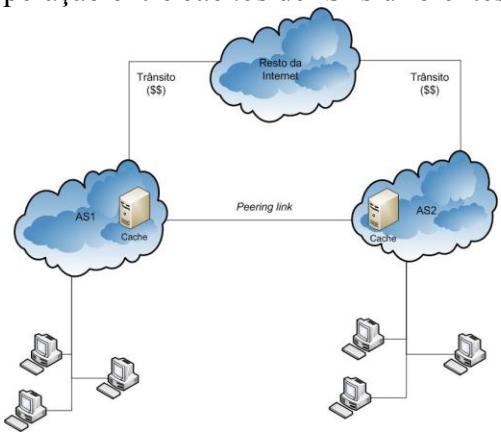


Figura 2. Modelo de cooperação entre *caches* de diferentes ASes

Cada ISP é representado por um AS distinto, AS1 e AS2. Normalmente os ISPs possuem acordos que permitem que uma quantidade de tráfego semelhante seja trocada entre eles sem incorrer em custo, conhecido como acordos de troca de tráfego ou *peering*. A comunicação com outros ISPs com os quais não existem acordos de troca (representado na figura como "resto da Internet") geralmente acontece através de enlaces de trânsito e incorre em custo para o ISP que origina o tráfego.

Para as análises da cooperação deste trabalho, cada ISP possui uma *cache* com o objetivo de armazenar os objetos mais populares e assim reduzir a quantidade de tráfego de trânsito, reduzindo o custo total de tráfego para o ISP. Quando uma determinada *cache* de um ISP recebe uma requisição de um de seus usuários, uma das seguintes situações pode acontecer:

(1) Todos os segmentos são encontrados e a requisição é completamente atendida localmente pela *cache*; o algoritmo de *cache* descrito é aplicado para decidir que segmentos do objeto devem ser armazenados;

(2) Apenas parte dos segmentos é encontrada e a requisição é parcialmente atendida pela *cache* local; o algoritmo de *cache* decide que segmentos devem ser armazenados, ou

(3) Nenhuma parte do objeto requisitado é encontrada na *cache* local.

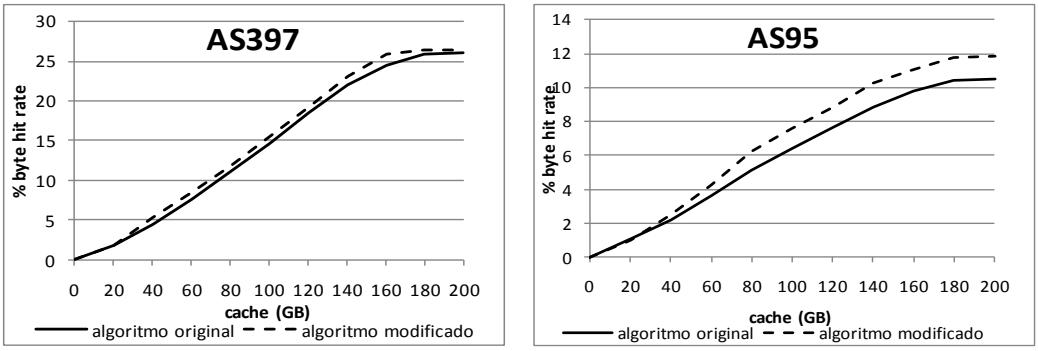
Nos casos (2) e (3) a requisição que não foi atendida (completa ou parcialmente) será redirecionada para a *cache* de um ISP vizinho, acontecendo assim a cooperação. Dessa forma espera-se que a maioria do tráfego gerado pelos usuários P2P possa ser

atendida pelo próprio ISP que originou a requisição ou por um ISP com o qual haja um acordo de troca de tráfego, minimizando o tráfego de trânsito.

4. Resultados

4.1.O Impacto do Algoritmo sobre a Eficiência da *Cache*

A fim de avaliar o desempenho do algoritmo modificado em condições reais de tráfego P2P, foram consideradas as características originais dos ASs 397 e 95 (Figura 3). Observa-se que em ambos os cenários o desempenho do algoritmo modificado é melhor do que o desempenho do algoritmo original. Entretanto, verificam-se duas situações distintas. O algoritmo modificado apresenta um melhor desempenho para o AS95, sendo 12,7% melhor que o algoritmo original, o que representa 1,18 TB de tráfego que é mantido internamente ao AS95. Já no AS397 o algoritmo modificado obtém uma diferença positiva de desempenho de 1,44%; como o tráfego total do AS95 é de 9,3 *terabytes*, mesmo este pequeno aumento na eficiência do algoritmo de *cache* representa quase 120 GBytes de tráfego que será mantido internamente ao AS.



(a) (b)

Figura 3. Desempenho do algoritmo modificado

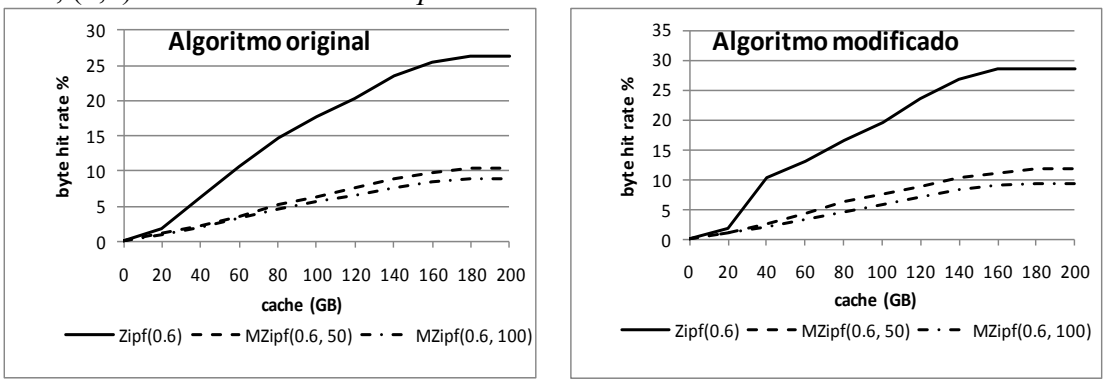
Mesmo no AS397, onde a diferença de desempenho é pouco melhor para o algoritmo modificado, a economia em termos de tráfego é bastante representativa. Ressalta-se que estamos tratando de uma simulação com um conjunto de dados que, embora representativo, é ainda bastante reduzido se comparado com dados reais. Em situações reais, onde o tráfego P2P é muito maior, a contribuição também será proporcionalmente mais alta, diminuindo ainda mais o custo de tráfego para o ISP.

Também é possível perceber que existe uma diferença no desempenho máximo alcançado em cada um dos ASs. No AS397 o desempenho máximo é de 25,83%, enquanto que no AS95 o desempenho máximo é de 11,83%. Dois fatores podem contribuir para a diferença no desempenho do algoritmo nos diferentes cenários, o percentual de tráfego passível de ser armazenado na *cache* (*cacheable*) isto é, os objetos que são requisitados mais de uma vez, e a distribuição de popularidade dos objetos, os quais investiga-se detalhadamente a seguir.

4.2.O Impacto da Distribuição de Popularidade

Embora nossos cenários sejam representativos de dois ASs reais, na prática as características dos ASs podem variar. Deste modo, nesta seção analisamos como a distribuição de popularidade afeta o desempenho do algoritmo de *cache* (Figura 4).

A distribuição MZipf possui dois parâmetros, sendo o primeiro a inclinação da curva e o segundo o fator de *plateau*. O fator de *plateau* indica a concentração das requisições para os objetos mais populares; quanto menor este fator significa que os objetos mais populares são requisitados mais freqüentemente. Quando o fator de *plateau* é zero, a distribuição iguala-se a uma Zipf. Para a avaliação do impacto da distribuição, o primeiro parâmetro foi fixado de acordo com o valor aproximado apresentado nos dois ASs, (0,6) e variou-se o fator de *plateau.*



(a) (b)

Figura 4. Impacto da distribuição de popularidade

Em primeiro lugar, observa-se que ambos os algoritmos apresentam melhor desempenho quando a distribuição de popularidade segue a Zipf(0.6). Este resultado é esperado, pois o algoritmo procura otimizar exatamente o armazenamento dos objetos mais populares. Assim, quando a distribuição é Zipf (fator de *plateau* = 0), existe um maior número de requisições para os objetos mais populares, e então o algoritmo é mais eficiente. Por outro lado, quanto maior o fator de *plateau*, maior será o 'achatamento' da curva da distribuição e, portanto, haverá menos requisições para os objetos mais populares diminuindo a eficiência do algoritmo. É o que acontece para MZipf(0.6,50) e MZipf(0.6,100), como mostram as figuras 4(a) e 4(b).

Adicionalmente se verifica que o algoritmo modificado atinge um melhor desempenho para todas as variações de distribuição de popularidade. Assim, é possível justificar o desempenho diferente para os dois cenários apresentados na seção 4.1. Esta variação decorre das diferentes distribuições de popularidade encontradas nos ASs. Enquanto que no AS397 a distribuição é MZipf (0.62, 8) e portanto há um maior número de requisições para os objetos mais populares, no AS95 a distribuição é MZipf (0.6, 50), indicando que as requisições acontecem em menor número para os objetos mais populares.

4.3. O Impacto do Tráfego Passível de Ser Armazenado na *Cache* (*cacheable*)

Na prática, nem todo objeto é requisitado mais de uma vez pelos usuários de um ISP. Se um objeto é requisitado apenas uma vez, não será vantajoso colocá-lo na *cache,* pois ocupará o espaço de outros objetos mais populares. A fim de avaliar como o volume dos objetos *cacheable* afeta o desempenho do algoritmo variou-se este parâmetro para o AS397 e AS95 (Figura 5).

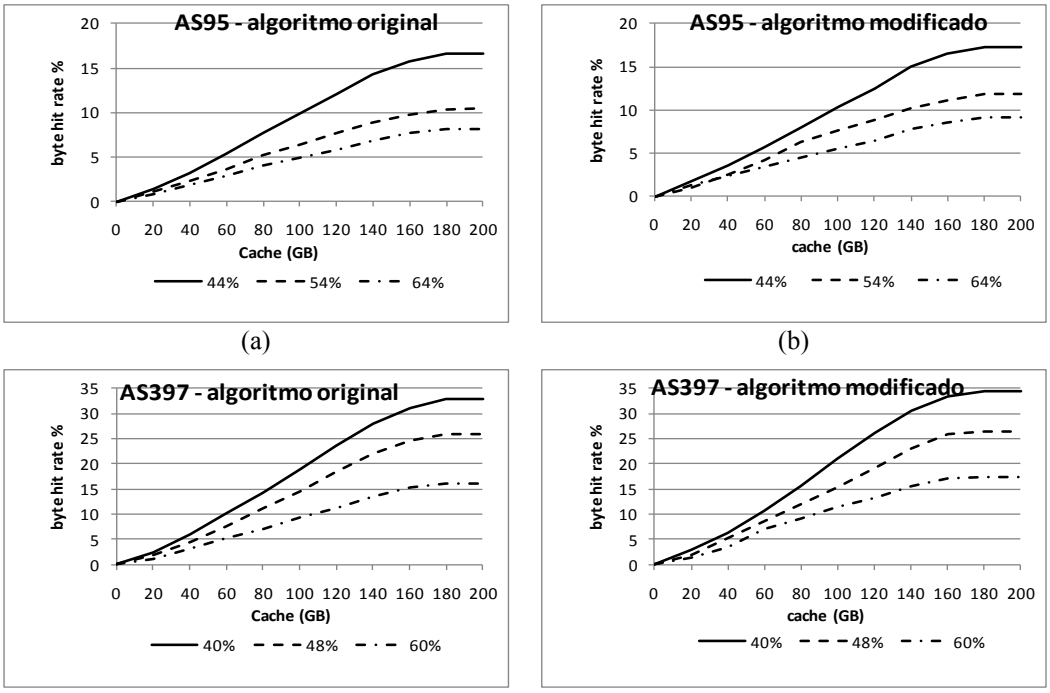


Figura 5. Impacto do tráfego *cacheable*

Primeiramente observa-se que o algoritmo modificado sempre obtém um melhor desempenho que o algoritmo original. Em todos os casos, menor tráfego *cacheable* implica em uma maior porcentagem de acerto. Isso acontece porque quanto menor a porcentagem de objetos requisitados mais de uma vez, o número de objetos mais populares tende a ser menor. Se poucos objetos são mais populares e o algoritmo guarda esses objetos, sua eficiência será maior.

4.4. O Impacto da Correlação Temporal

Como descrito na Seção 3.2, as requisições foram geradas de forma seqüencial para blocos de segmentos de um objeto. A fim de avaliar como a correlação temporal entre as requisições dos segmentos pode afetar o desempenho do algoritmo, variou-se a intensidade da correlação. A variação da correlação temporal foi obtida a através da geração de requisições pelo *ProWGen* com parâmetros de correlação fraca, média e alta para o AS397 (Figura 6).

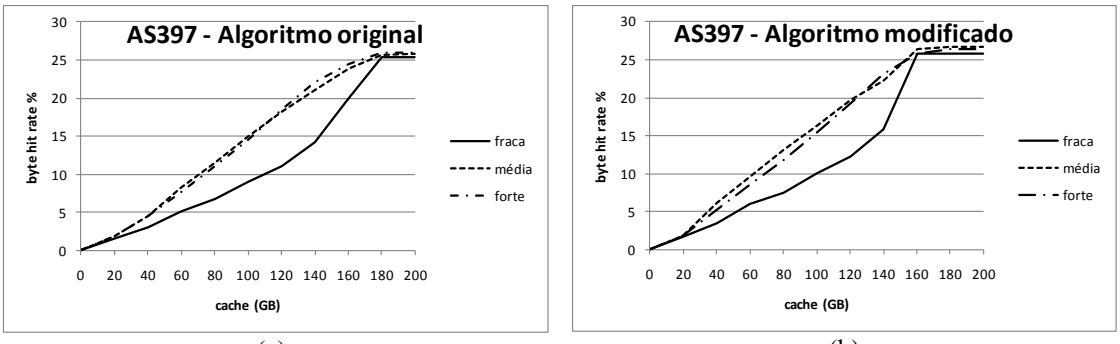


Figura 6. Impacto da correlação temporal

Observa-se que o desempenho do algoritmo praticamente não é afetado pela correlação temporal das requisições e a taxa de acertos da *cache* estabiliza com valores

máximos muito próximos. O algoritmo modificado obtém o desempenho máximo para uma *cache* menor, de 160 GB, enquanto que para o algoritmo original o desempenho máximo é obtido com uma *cache* de 180 GB.

4.5.O Impacto do Tamanho do Segmento

Nesta seção analisamos como o tamanho do segmento, unidade de inserção e remoção de dados na *cache,* pode afetar o seu desempenho. As requisições são feitas em blocos com 50 segmentos de tamanho variável de 1, 2, 5 e 10 MB. Utiliza-se o AS397 (Figura 7).

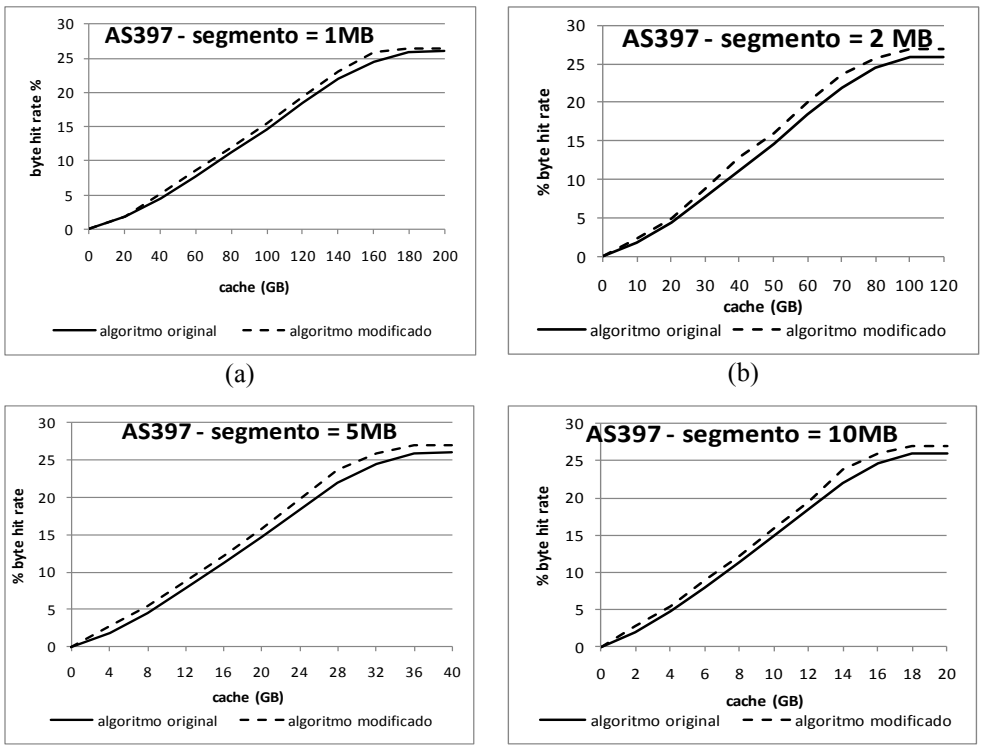


Figura 7. Impacto do tamanho do segmento

Primeiramente verifica-se que o desempenho máximo apresenta apenas uma pequena variação quando o tamanho do segmento é modificado. Ainda assim, o algoritmo modificado alcança uma melhor taxa de acertos em todos os casos. Uma diferença de 1,32% na taxa de acertos, como no caso do segmento de tamanho 5 MB, representa uma melhora no desempenho de 5%, o que pode corresponder a uma fatia considerável de tráfego. Observa-se ainda que o tamanho da *cache* necessário para atingir o desempenho máximo é inversamente proporcional ao tamanho do segmento. Para um tamanho de segmento dez vezes maior, verifica-se que o tamanho de *cache* necessário para atingir o desempenho máximo é reduzido para 10%, como visto nas Figuras 7(a) e 7(d). Quando o tamanho do segmento é muito pequeno, apenas pequenos pedaços do objeto são inseridos na *cache* a cada acerto, o que faz com que seja necessário um tamanho de *cache* maior para que o algoritmo mostre a sua eficiência.

4.6.*Cache*s cooperativas

Nessa seção são apresentados os resultados dos experimentos que envolvem a cooperação entre as *caches* dos sistemas envolvidos, conforme o modelo descrito na

Seção 3.4. Os ISPs que participam da cooperação são modelados como dois sistemas autônomos com as características do AS397 e do AS95 (Tabela 1). De acordo com a literatura, a cooperação faz sentido para ASs de tamanhos similares [Dán 2009] [Hefeeda and Noorizadeh 2008] (Figura 8).

Para avaliar a taxa de acertos nesse cenário, inicialmente foram gerados dois conjuntos de requisições distintos, mas para os mesmos objetos de vídeo. O *ranking* de popularidade dos objetos foi o mesmo nos dois ISPs, sendo mantidas as demais características dos ASs. As requisições foram processadas simultaneamente.

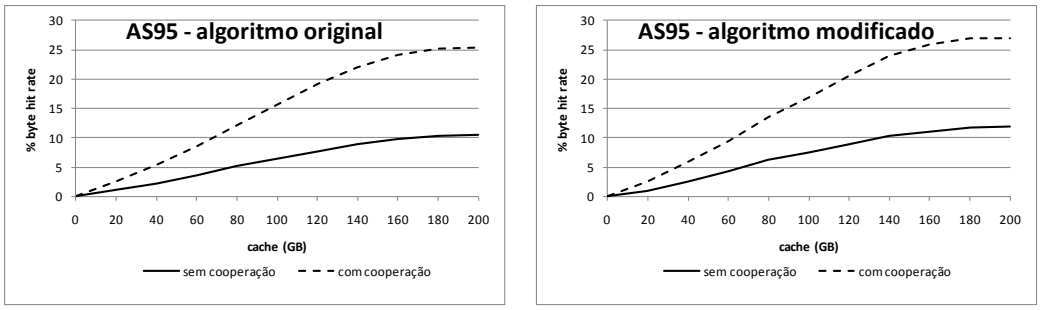


Figura 8. Comparações para os cenários com cooperação e sem cooperação

Percebe-se que, tanto para o algoritmo original como para o modificado, a taxa de acertos servidos pela cooperação no AS95 aumenta em mais de 100% (lembrando que parte do tráfego provém de requisições que chegam através do *link* de cooperação).

Entretanto, embora dois ISPs vizinhos possam tender a apresentar as mesmas características de popularidade, essa popularidade pode não ser exatamente a mesma. Para avaliar a influência deste aspecto sobre a cooperação, a correlação entre a popularidade dos objetos foi variada através de três cenários. No cenário 1, o *ranking* de popularidades é o mesmo para os dois ISPs. No cenário 2, o objeto mais popular do AS397 é o quarto mais popular do AS95 e, no cenário 3, o objeto mais popular do AS397 é o décimo primeiro mais popular do AS95 (Figura 9).

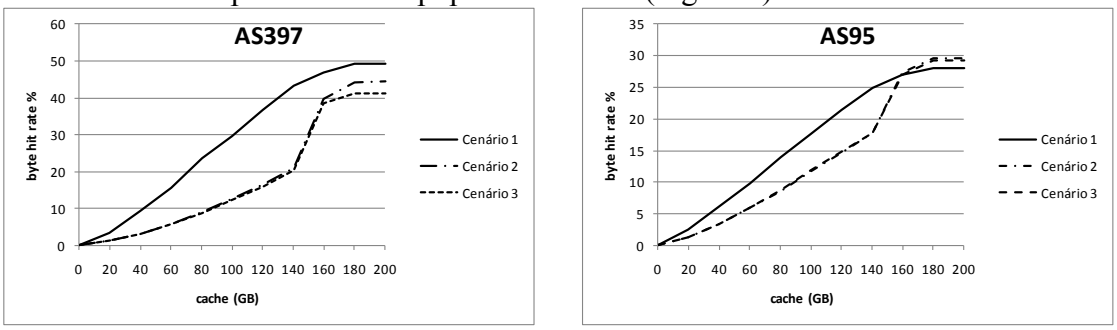


Figura 9. Impacto da cooperação para diferentes graus de correlação

Primeiramente, analisemos o desempenho da cooperação tomando como referência o AS397. O AS397 mantém na *cache* local os objetos mais populares e atende tanto as suas requisições como aquelas recebidas do AS95. Todas as requisições para objetos feitas a partir do AS95 e não encontrados na sua *cache* são encaminhadas para o AS397. À medida que os dois ASs vão emitindo as suas requisições, o AS397 começa a receber as requisições locais para os objetos mais populares e os armazena na *cache.* Após alguns instantes, o AS397 começa a receber as requisições do AS95, que foram redirecionadas, para estes mesmos objetos que provavelmente já estão na sua

cache, o que faz com que as requisições do AS95 sejam atendidas através da cooperação, aumentando sua taxa de acerto. Dessa forma, a cooperação faz com que o AS95 obtenha um desempenho ainda maior do que sem cooperação.

Observando o AS95, uma menor correlação significa que as requisições redirecionadas do AS397 nem sempre serão atendidas, uma vez que os objetos mais populares do AS397 não são tão populares no AS95. Entretanto, a cooperação faz com que a *cache* do AS95 sirva as requisições encaminhadas do AS vizinho, embora a contribuição fornecida seja menor que a recebida. Ao serem encaminhadas, as requisições não afetam a popularidade dos objetos na *cache* local, e provavelmente os objetos mais populares do sistema vizinho não estarão na *cache* do AS95. Isso acontece porque os objetos mais populares no AS397 não serão tão requisitados no AS95, causando uma menor cooperação do segundo em relação ao primeiro.

Quando o esquema de cooperação entre *caches* é utilizado, o principal efeito é o aumento da quantidade de tráfego servida pelo enlace de *peering*. A Tabela 2 mostra o ganho em termos de localidade de tráfego obtido pelos dois ISPs (AS397, AS95).

Tabela 2. Redução de tráfego através da cooperação (GB)

AS397	***Cache* local**	**Enlace de *peering***	**Enlace de trânsito**	**Total**
Sem cooperação	1180	0	8135	9315
Com cooperação	1180	1023	7112	9315
AS95	***Cache* local**	**Enlace de *peering***	**Enlace de trânsito**	**Total**
Sem cooperação	595	0	8721	9316
Com cooperação	595	762	7959	9316

Percebe-se que o esquema de cooperação reduz significativamente o volume de tráfego que usa o enlace de trânsito, permitindo que o tráfego seja servido pelo enlace de *peering*. Para o AS397, o tráfego servido através do enlace de *peering* é de 86,7% do tráfego local, enquanto que para o AS95 é de 128%. Para o AS95, o tráfego que transita pelo enlace de *peering* chega a ser maior que o tráfego servido pela *cache* local. Para o AS397 há uma redução de 1023 GB no uso do enlace de trânsito, representando 12,6% a menos de tráfego. A redução para o AS95 é de 762 GB (8,74%).

5. Conclusões e Trabalhos Futuros

Os ISPs consideram o tráfego P2P como indesejado, pois geralmente a falta de localidade na escolha dos nós para troca de dados provoca um aumento do tráfego de trânsito e, conseqüentemente, do custo de tráfego para os ISPs. Uma das soluções para este problema é o uso de *caches* com o objetivo de manter o tráfego P2P local ao ISP.

Este trabalho analisa o impacto do uso de *caches* para o armazenamento de tráfego decorrente de aplicações P2P de Vídeo sob-Demanda, onde geralmente os objetos são grandes e impactam fortemente no perfil de tráfego do ISP. Através da otimização de um algoritmo de *cache* parcial que armazena os segmentos dos objetos mais populares, verifica-se que o algoritmo de *cache* é capaz de obter um desempenho bem melhor que o algoritmo original em alguns cenários. Observa-se ainda que a distribuição de popularidade dos objetos, o percentual de tráfego *cacheable* e o tamanho dos segmentos impactam na eficiência do algoritmo, enquanto que se observa muito pequeno impacto da correlação temporal.

A maior contribuição deste trabalho é o estudo da cooperação entre as *caches* de

diferentes ISPs. Através da modelagem de requisições entre *caches* de ISPs vizinhos é possível verificar que a cooperação pode proporcionar uma redução de tráfego de trânsito bastante significativa, de até 12,6%. Ao mesmo tempo, o tráfego servido pelo enlace de *peering* chega a ser até 128% a mais do que o tráfego servido pela *cache* local.

Como trabalho futuro pretende-se usar a Teoria dos Jogos para modelar um maior número de caches cooperativas, tanto dentro do mesmo ISP como em ISPs vizinhos, a fim de analisar os ganhos e buscar um equilíbrio na cooperação.

Referências

Blond, S. L., Legout, A. and Dabbous, W. (2008) "Pushing BitTorrent Locality to the Limit", INRIA, Tech. Rep. 0034382.

Busari, M. and Wiliamson, C. (2002) "ProWGen: A Synthetic Workload Generation Tool for Simulation evaluation of Web Proxy *Cache*s", Journal of Comp. Networks.

Choffnes, D. R. and Bustamante, F. E. (2008) "Taming the Torrent - A practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems", In SIGCOMM'08.

Dán, G. (2009) "Cooperative Caching and Relaying Strategies for Peer-to-Peer Content Delivery" at *7th International Workshop on P2P Systems (IPTPS '08)*.

Feitosa, E., Souto, E., Sadok, D. (2008) "Tráfego Internet Não Desejado: Conceitos, Caracterização e Soluções". Minicurso VIII SBSeg, Gramado, RS, Brasil.

Gummadi, K. P., Dunn, R. J., Saroiu, S., D.Gribble, S., Levy, H. M. and Zahorjan, J. (2003) "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload" in Proc. SOSP'03, p. 314-329.

Hefeeda, M. and Noorizadeh, B. (2008) "Cooperative Caching: The Case for P2P Traffic", In *Local Computer Networks, 33rd IEEE Conference,* p. 12-19.

Hefeeda, M. and Saleh, O. (2008) "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems", In *IEEE/ACM Transactions on Networking.*

Huang, Y., Fu, T. Z., Chiu, D., Lui, J. C. and Huang, C. (2008) "Challenges, Design and Analysis of a Large-Scale P2P-VoD System", In *SIGCOMM Comp. Com. Rev.* 38.

Karagiannis, T., Rodriguez, P. and Papagiannaki, K. (2005) "Should Internet Service Providers Fear Peer-Assisted Content Distribution?", In *Internet Measurement Conference 2005*.

Leibowitz, N., Bergman, A., Ben-Shaul, R.and Shavit, A. (2002) "Are File Swapping Networks *Cacheable*? Characterizing P2P Traffic", Proc. of 7th Int. WWW Caching Workshop.

Shen, G., Wang, Y., Xiong, Y., Zhao, B. Y. and Zhang, Z. (2007) "HPTP: Relieving the Tension between ISPs and P2P", In *IPTPS*.

Yu, J.; Chou, C. T.; Yang, Z.; Du, X. and Wang, T., "A dynamic caching algorithm based on internal popularity distribution on streaming media" in *Multimedia Systems,* 2006.