Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2009.2

O Impacto do Uso de Caches Cooperativos no Tráfego Inter-ISP para Distribuição de Vídeo sob-Demanda

Trabalho de Graduação

Aluno	Petrônio Gomes Lopes Júnior	{pglj@cin.ufpe.br}
Orientador	Djamel Fawzi Hadj Sadok	{jamel@cin.ufpe.br}
Co-Orientador	Josilene Aires Moreira	{jam2@cin.ufpe.br}

30 de Novembro de 2009

Universidade Federal de Pernambuco

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2009.2

O Impacto do Uso de Caches Cooperativos no Tráfego Inter-ISP para Distribuição de Vídeo sob-Demanda

Trabalho de Graduação

Trabalho de graduação apresentado no Centro de Informática da Universidade Federal de Pernambuco por Petrônio Júnior, orientado por Djamel Sadok, como requisito para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador Djamel Fawzi Hadj Sadok {jamel@cin.ufpe.br}
Co-Orientador Josilene Aires Moreira {jam2@cin.ufpe.br}

30 de Novembro de 2009

Folha de Aprovação

O Impacto do Uso de Caches Cooperativos no Tráfego Inter-ISP para Distribuição de Vídeo sob-Demanda

Petrônio Gomes Lopes Júnior

Aprovado em 30 de novembro de 2009

Banca examinadora:

_____

Profº. Djamel Fawzi Hadj Sadok, PhD – UFPE (Orientador)

_____

Profª. Judith Kelner, PhD – UFPE (Avaliadora)

"A mente que se abre a uma nova idéia jamais volta ao seu tamanho original."

Albert Einstein

À minha família.

Agradecimentos

Enfim, começo fazendo a parte deste trabalho que certamente será a mais complicada e me tomará mais tempo. São muitos agradecimentos, são muitas pessoas que fazem parte dessa conquista, dessa vitória que não é somente minha. Não posso, e até prefiro, não citar todos porque são muitos e seria injustiça da minha parte esquecer alguém.

Em primeiro lugar, agradeço aos meus pais, Petrônio Gomes Lopes, o meu pai e exemplo, a pessoa que sempre me orientou e com quem sempre quis me parecer, e Ivana de Fátima Silva Lopes, minha querida mãe, que sempre me dedicou os melhores e maiores cuidados independente da situação em que estivéssemos com todo amor possível, e à minha madrinha, Isley Maria da Silva, para mim, também minha mãe. Agradeço também aos meus irmãos, Priscillinha e Guto, que, apesar de crianças, já me ensinaram bastante sobre a vida. Essa família, que tanto amo, com certeza é composta pelos maiores professores que já tive em minha vida, pessoas em quem confio e que me apóiam sempre.

Com toda certeza não menos importante, agradeço à minha namorada, Mirella, que, além de me ajudar, me entender e estar presente nos momentos mais difíceis, é essencial para fazer os momentos alegres ou torná-los ainda melhores. Isso sem considerar que ela trabalhou como revisora deste trabalho, corrigindo alguns de meus muitos erros de português!! Apesar do curto espaço de tempo que estamos juntos, ela é, junto à minha família, fundamental na minha vida.

Agradeço, com todo carinho do mundo, aos meus avôs que já se foram, Plínio, Eunice e Vandete, e ao meu avô Estácio, o único com quem posso dividir essa vitória. Gostaria muito de compartilhar esse momento de felicidade com eles, que foram direta e indiretamente responsáveis por uma grande parcela dessa conquista.

Agradeço aos meus demais familiares, que de alguma forma contribuíram para o meu sucesso.

Agradeço também a todos os professores e, principalmente, aos amigos que fiz no Colégio Militar do Recife. Pessoas muito importantes na minha formação, que me ensinaram bastante e, certamente, não serão esquecidas. Não tenho mais tanto contato quanto gostaria, cada um seguiu sua vida, mas os laços de amizade criados continuam existindo.

Agradeço aos amigos que fiz no curso, amigos importantes com os quais convivi tanto tempo nesses quase quatro anos quanto com minha família. Amigos que fizeram menos complicada a minha permanência no Centro de Informática. Tanto os mais espertos (que escolheram ciência da computação) quanto os coitados de engenharia (menos espertos, mas não menos importantes).

Certamente preciso agradecer aos que fazem parte do GPRT, grupo do qual faço parte há quase dois anos e onde aprendi muito. Todos sempre estão dispostos a ajudar e a ensinar, a resolver os mais diversos problemas e a compartilhar conhecimento sem impor qualquer tipo de dificuldade. Lá, encontrei muitas das diretrizes que não tinha em relação a questões profissionais, descobrindo um caminho para seguir.

Finalmente, agradeço a Deus pelas possibilidades, pela saúde e pela força que tive para concluir mais essa etapa. Sem Ele, com certeza nada teria sido possível, Ele me deu forças, além de todas as ferramentas que precisei para prosseguir.

Resumo

Sistemas *peer-to-peer* (P2P) são cada vez mais presentes nos contextos das redes de distribuição de conteúdo. O volume de tráfego gerado por esse tipo de rede *overlay* é muito grande, em especial VoD, e é bastante custoso para os provedores de serviço. É fundamental para os ISPs, que de alguma forma o tráfego *inter*-ISP seja reduzido.

Existem diversas propostas para diminuir o tráfego *inter*-ISP e a utilização de *caches* aparece como uma solução em potencial. No entanto, *caches* tradicionais não estão preparadas para o tráfego oriundo de uma rede P2P devido a suas características.

É nesse contexto que esse trabalho visa avaliar o uso de *caches* preparadas para o tipo de tráfego em questão e capazes de cooperar entre si, analisando o impacto dessa abordagem no tráfego que passa por ISPs externos e que gera muitos custos para os provedores de serviço.

Sumário

Índice de figuras

Índice de tabelas

Siglas

Sigla	**Significado**
AS	Sistema autônomo
ISP	Provedor de serviço da Internet
P2P	*Peer-to-Peer*
VoD	Vídeo sob demanda
CDN	Rede de distribuição de conteúdo
HTTP	Protocolo de transferência de hipertexto
HPTP	HTTP-*based Peer-to-Peer*
PROP	*Collaborating and coordinating* ***pro****xy and its* ***P****2P clients*

1. Introdução

Nos dias de hoje, a popularidade de sistemas *Peer-to-Peer* (P2P) para distribuição de conteúdo (compartilhamento de arquivos) tem aumentado rapidamente de uma maneira geral. A distribuição de conteúdo através de redes *Peer-to-Peer* é responsável pela maior parte do tráfego gerado entre ISPs (*Internet Service Providers*) e o crescente uso desse tipo de rede sugere um aumento ainda maior do tráfego no futuro [3]. Além disso, de acordo com [13] e [14], a maior parte dos *bytes* transferidos em aplicações P2P pertence a grandes objetos (geralmente arquivos de vídeo), ou seja, é possível identificar que a maior parcela do tráfego nessas aplicações é gerada por vídeos sob demanda (VoD). Dessa forma, distribuição de conteúdo através de redes P2P, em especial VoD já que é o tipo de mídia mais usado quando se trata de consumo de banda, é o principal problema a ser atacado para redução desse tráfego.

Em redes P2P, a distribuição do conteúdo é otimizada, já que não são necessários, relativamente, grandes investimentos nos servidores mesmo para um grande número de usuários. Os custos da distribuição, nesse caso, são divididos entre os provadores de conteúdo, os *peers* e seus ISPs [7], ou seja, é, do ponto de vista econômico, interessante para os provedores de conteúdo e custoso para os ISPs [9] já que os *peers* 'assumem' parte da distribuição, facilitando a tarefa do provedor de conteúdo, e os ISPs lidam com o tráfego adicional gerado.

De acordo com [1], para reduzir custos, alguns ISPs passam a bloquear esse tipo de tráfego, impondo limitações de banda para aplicações P2P já que elas são responsáveis, em alguns casos, por cerca de 50 a 70 por cento do tráfego da rede. As aplicações, por sua vez, passaram a utilizar técnicas de encriptação para burlar essas limitações.

O aumento indiscriminado do tráfego P2P pode impactar negativamente na rede de duas formas: a primeira remete ao aumento de carga circulando no

backbone da Internet, possivelmente congestionando a rede, e; a segunda trata do aumento dos custos para um ISP [3].

Existem várias propostas para redução da carga gerada em ISPs por diversas aplicações P2P. Essas propostas surgem como consequência de estudos que mostram a ineficiência dos protocolos P2P do ponto de vista de custo de tráfego para os ISPs. Na literatura, as principais alternativas para soluções são a noção de localidade dos *peers* e a implementação de *caches* nos sistemas em questão para atender as requisições dos *peers* [1] [7].

Na primeira alternativa, algumas abordagens prezam pela seleção dos *peers* considerando a localidade que reduzem o tráfego *inter*-ISP, entretanto essas soluções não sejam amplamente utilizadas em sistemas P2P comerciais [1].

A segunda alternativa remete ao contexto em que aparecem os algoritmos de *cache* para conteúdo P2P como alternativa viável, apesar da necessidade de uma infra-estrutura e de suporte, para redução dos custos do tráfego entre ISPs e da carga nos *backbones* da Internet [7]. Para o caso específico de distribuição de vídeo nesse tipo de rede, diversos autores defendem um esquema de cache parcial [3] devido ao tamanho dos objetos [13] e aos diferentes objetivos a serem alcançados.

Em [1], uma arquitetura que permite um uso mais eficiente de caches para P2P é proposta. Nela, as caches em diferentes ISPs se comunicam e cooperam entre si, além de executarem seus algoritmos normalmente, a fim de diminuir o tráfego IP. A arquitetura é apresentada no contexto de sistemas P2P de *video streaming*. Essa cooperação entre *caches* de diferentes ISPs só é possível porque ISPs de tamanhos próximos e com cobertura geográfica similares estabelecem acordos (*settlement-free peering agreements*) que possibilitam a troca de tráfego IP livremente para o benefício dos sistemas envolvidos.

Além disso, diversos algoritmos de cache parcial são modelados segundo o tipo de conteúdo a ser compartilhado na rede. Em [2], por exemplo, o esquema de cache se baseia na popularidade dos objetos, enquanto em [5], os autores estendem esse modelo acrescentando o *bit rate* e a largura de banda entre servidores e

clientes. Nessa proposta, o objetivo é minimizar o atraso inicial (*start-up delay)* médio e melhorar a qualidade do vídeo (*playback continuity)*.

Diante do problema causado pelo tráfego P2P, um esquema de *caches* cooperativos surge como uma potencial solução para redução de custos gerados nos ISPs sem comprometer o tráfego P2P, além de, em alguns casos, alcançar melhorias específicas para a qualidade e disponibilidade do conteúdo compartilhado. Para a realização de estudos detalhados, é necessária ou a utilização de dados reais de redes P2P, como em [14], que são extremamente difíceis de serem obtidos, ou a reprodução desse cenário através de *traces* sintéticos como em [4] e [6], por exemplo.

O objetivo principal deste trabalho é esclarecer e avaliar o impacto da cooperação de *caches* para o tráfego *inter*-ISP. Na seção 2, são apresentados alguns trabalhos relacionados com o objetivo deste. Na seção 3, é apresentada a metodologia que será utilizada, bem como o processo de geração dos *workloads* e, em seguida, na seção 4, o algoritmo de *cache* é descrito. Na seção 5, as características da cooperação são expostas. Na seção 6, os experimentos e os resultados são apresentados. Por fim, na seção 7, é realizada a conclusão além de serem listadas algumas possibilidades para a sequência deste trabalho no futuro.

2. Estado da arte

Nessa seção, serão descritos alguns trabalhos que, de alguma forma, se relacionam com o problema que será abordado neste trabalho.

Em [7], o impacto de redes P2P para distribuição de conteúdo é abordado sob três perspectivas (dos *peers,* dos servidores de conteúdo e dos ISPs) e a utilização dessas redes é introduzido como fonte de uma parcela considerável do tráfego *inter*-ISP (ISP-*unfriendly*). Nesse trabalho, apesar dos benefícios gerados para os *peers* e para os servidores de conteúdo, o impacto negativo nos ISPs é destacado a fim de encontrar uma solução razoável para o problema. O estudo foi realizado no contexto do BitTorrent [16], que aumenta os custos de um ISP significativamente. Por fim, algumas diferentes abordagens são discutidas e comparadas com a solução proposta, a inserção da idéia de localidade para ambientes P2P, obtendo uma performance que se aproxima da abordagem ótima utilizando *caches*.

Em [15], é proposta uma abordagem para reduzir o custo do tráfego *inter*-ISP sem que o desempenho do sistema P2P seja sacrificado. A seleção dos *peers* é feita para inserir o conceito de localidade. Nela, os *peers* se comunicam com vizinhos que teoricamente estão próximos (no mesmo ISP), 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 da 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. Por fim, esse trabalho mostra que com essa abordagem é possível encontrar 'caminhos entre *peers*' que, comparados a sistemas P2P sem essa seleção, possibilitam reduzir o tráfego *inter*-ISP.

Em [9], o conceito de localidade em redes P2P também é abordado devido ao aumento do tráfego nos *links inter*-ISPs, ou seja, a idéia de manter uma parcela do tráfego P2P dentro de um mesmo ISP é estudada como uma solução possível. Nesse trabalho, os autores realizam diversos experimentos em um ambiente com

clientes BitTorrent para avaliar os impactos da localidade no tráfego *inter*-ISP e no tempo de *download* do objeto requisitado. Dessa forma, dois mecanismos são propostos a fim de tornar possível a redução dos custos dos ISPs sem afetar o tempo de *download*. Segundo os autores, nos experimentos realizados, o tráfego *inter*-ISP foi reduzido em 40%.

Devido aos problemas entre aplicações P2P e ISPs, os autores em [11] propuseram um *framework* para tratar essa questão. Dessa forma, surgiu o HPTP (HTTP-*based Peer-to-Peer*) *framework,* que propõe a utilização de *caches* já existentes nos ISPs para 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, foram descritas ainda importantes ferramentas para a construção do *framework* proposto e, por meio de simulações, ganhos significativos foram identificados, como a redução da carga de tráfego entre ISPs e no *backbone* da Internet sem comprometer o rendimento da aplicação P2P envolvida.

Em [10], um sistema denominado PROP (*collaborating and coordinating **pro**xy and its **P**2P clients*) é proposto para distribuir *streaming* de vídeo, conciliando os benefícios de um sistema P2P, como escalabilidade por exemplo, com a garantia de qualidade da distribuição do conteúdo. Nesse caso, um *proxy* é posto junto ao sistema P2P em uma mesma *intranet* para garantir a qualidade do serviço, os objetos são divididos em segmentos que são armazenados tanto pelos *peers* quanto pelo *proxy* (segundo alguma política). Dessa forma, os componentes desse sistema são complementares, *peers* provendo escalabilidade e o *proxy* provendo armazenamento dedicado e confiabilidade. Os resultados apresentam melhorias significativas na qualidade do serviço e na escalabilidade do sistema.

Em [3] [8], um estudo sobre características relevantes sobre tráfego P2P (popularidade dos objetos, tamanho do objeto e variação da popularidade) foi realizado. Dessa forma, o tráfego P2P foi caracterizado, a popularidade dos objetos

foi modelada através da distribuição Mandelbrot-Zipf e vários *workloads* foram obtidos. Posteriormente, um algoritmo de *cache* parcial foi proposto, baseado na modelagem realizada anteriormente, considerando as estratégias de *cache*, a segmentação dos objetos e a forma de armazenamento sem existir qualquer tipo de cooperação entre as *caches*. Seguindo a idéia de *cache* de um *web proxy,* o algoritmo é utilizado para minimizar os principais problemas, apontados no artigo, provenientes do tráfego P2P, ou seja, reduzir a carga no *backbone* da Internet e os custos para os ISPs. Por fim, através das simulações, altos índices de acerto foram obtidos (superior aos algoritmos mais populares), sendo possível constatar a importância do algoritmo de *cache* parcial para melhorar o desempenho de sistemas P2P e reduzir o tráfego *inter*-ISP.

Em [1], o autor, assumindo que uma forma de reduzir custos para um ISP é a construção de *caches* para o tráfego P2P, propõe um esquema de caches cooperativas compatível com as relações de negócio existentes entre ISPs. Nesse trabalho, o problema de *caches* cooperativas é formulado como um problema de alocação de recursos e modelado segundo conceitos da teoria dos grafos. O autor propõe uma arquitetura que permite que os ISPs utilizem suas *caches* de maneira mais eficiente através de *peering* ISPs. Os resultados apresentados demonstram a capacidade do esquema de *caches* cooperativas de combater os custos para os ISPs oriundos do tráfego P2P e aumentar significantemente a eficiência das *caches* envolvidas.

Em [12], é analisado o potencial de *caches* cooperativas reduzirem os custos gerados em ISPs por conta do tráfego P2P. Os autores propõem dois modelos de cooperação de *caches*. O primeiro modelo pretende estabelecer cooperação entre *caches* de diferentes sistemas autônomos (ASes) e o segundo modelo trata das *caches* em um mesmo AS. Para obter resultados satisfatórios e coerentes, um *trace* com oito meses de duração foi obtido através de um popular cliente do Gnutella [17] e diversas simulações foram executadas, alternando entre os dois modelos propostos. Além disso, diversas variações puderam ser impostas à simulação,

como tamanho da *cache* e estratégia de replicação, por exemplo. Por fim, foram apresentados os resultados que destacam a relevância da cooperação entre *caches*, o *overhead* insignificante que é gerado comparado ao volume de trafego total e a influência das políticas utilizadas pelas *caches* para armazenar um arquivo.

De acordo com os trabalhos existentes na área, fica evidente que o problema gerado pelo tráfego *inter*-ISP na distribuição de VoD é bastante estudado e diversas abordagens que visam a localidade, seja na seleção dos *peers* ou com a inserção de *caches*, são propostas. Entretanto as soluções descritas na literatura são em sua maioria propostas isoladas e não uma combinação de possíveis abordagens. Este trabalho pretende utilizar soluções existentes combinadas além de realizar alguma melhoria de acordo com as características do tráfego em questão.

3. Metodologia

Inicialmente, fez-se necessário a criação de *workloads* sintéticos capazes de reproduzir as características de ASes reais já que a utilização de *workloads* reais é bastante complicada. Dessa forma, os ASes foram reproduzidos considerando as informações contidas em [3]. Em seguida, os objetos são descritos e os cenários concebidos a partir do *workload* e da descrição dos vídeos.

A partir dos cenários criados, visando à obtenção de resultados conclusivos a cerca do impacto da cooperação de *caches* no tráfego *inter*-ISP, uma série de experimentos foi executada com base em algoritmos e modelos escolhidos. Os resultados foram avaliados posteriormente segundo o *byte hit rate* (taxa de acerto), que métrica escolhida para mensurar a eficiência das *caches.*

Nas subseções seguintes, serão apresentados o processo de geração do *workload*, a descrição dos objetos e a descrição dos cenários.

Com o intuito de restringir o número de cenários, foram escolhidos inicialmente dois ASes cujas características principais são especificadas nas subseções a seguir.

3.1. Geração do *workload*

Para modelar cenários similares ao que pode ser observado em sistemas reais, foi necessário um estudo detalhado das características do *workload* de sistemas P2P e das características de VoD, como descrito em [13] [14] por exemplo.

Na primeira etapa para concepção de um cenário realista, o gerador de *workloads* sintéticos ProWGen [6] foi modificado para suportar o uso da distribuição Mandelbrot-Zipf, que é uma variação da distribuição Zipf e é apontada por muitos autores na literatura como a melhor distribuição para modelar a popularidade de objetos em uma rede P2P [3].

A distribuição Mandelbrot-Zipf recebe dois valores como parâmetros: o primeiro, o *skewness,* remete à inclinação da curva gerada pela distribuição (quanto maior esse parâmetro, maior será a popularidade dos primeiros elementos do *rank*) e; o segundo, o fator de *plateau,* trata do achatamento do topo da curva, tornando os primeiros elementos do *rank* menos populares em relação aos demais à medida que seu valor cresce, suavizando a inclinação da curva, como pode ser observado na Figura 1. Na distribuição Zipf, apenas o primeiro parâmetro aparece, nesse caso, pode-se afirmar que a distribuição Zipf é um caso específico da Mandelbrot-Zipf (*skewness* qualquer e fator de *plateau* igual a zero).

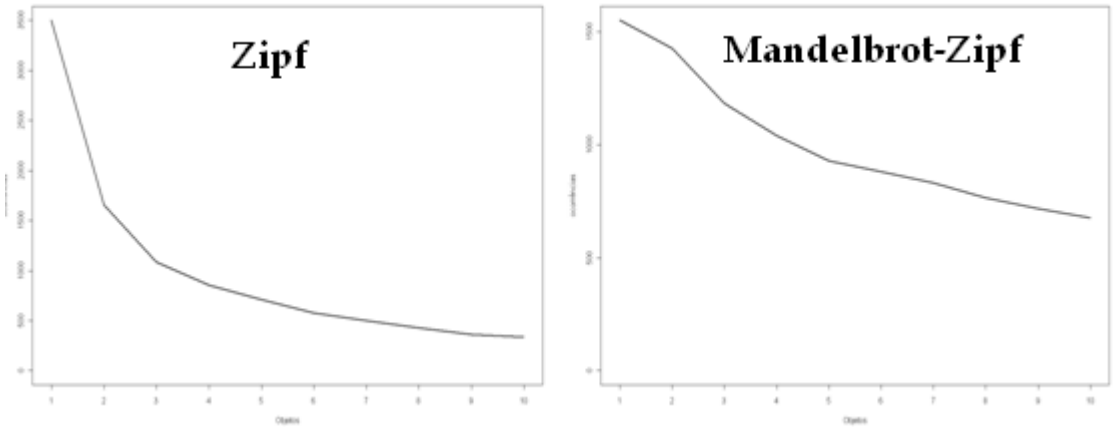

Figura 1 - Comparação das curvas entre as distribuições

O ProWGen gera uma lista de objetos a serem requisitados, já dispostos na ordem em que esses eventos ocorreriam, em outras palavras, a lista gerada contém o identificador de cada vídeo a ser requisitado ordenados segundo o momento em que ocorrem. Essa ferramenta permite ainda uma série de variações de características para o *workload* que será criado: número de requisições, número de objetos ou mesmo grau de correlação temporal. Na prática, as informações geradas identificam qual vídeo é desejado pelo usuário e o momento do início da requisição de cada vídeo em relação às demais requisições.

Em redes P2P, os objetos de vídeo costumam ser grandes, variando entre 100 KB e 1 GB em média. Dessa forma, na concepção do cenário, um tamanho fixo para

todos os objetos foi estabelecido em 1 GB, que remete ao fato dos maiores objetos de uma rede P2P serem objetos de vídeo [13].

A menor unidade de armazenamento para as *caches* foi fixada em um segmento de 1 MB de tamanho. Para avaliar a efetividade do algoritmo proposto em [3], cada vídeo é requisitado parcialmente, i.e, o objeto é divido em várias partes, denominadas blocos, que são compostos por segmentos (menor unidade de armazenamento na *cache*) [20]. O objeto é dividido em *N* blocos de tamanho constante cada um contendo um número *M* de segmentos. Naturalmente, o tamanho do objeto de vídeo é dado pelo produto *N***M* (número de blocos multiplicado pelo tamanho dos blocos).

Na Figura 2, o formato da composição de um objeto de vídeo é exposto. O número de blocos e o número de segmentos por bloco podem ser variados, embora recebam valores fixos nos cenários propostos a fim de prezar pela simplicidade dos experimentos (restrição de algumas características para obter o menor número de variantes do cenário).

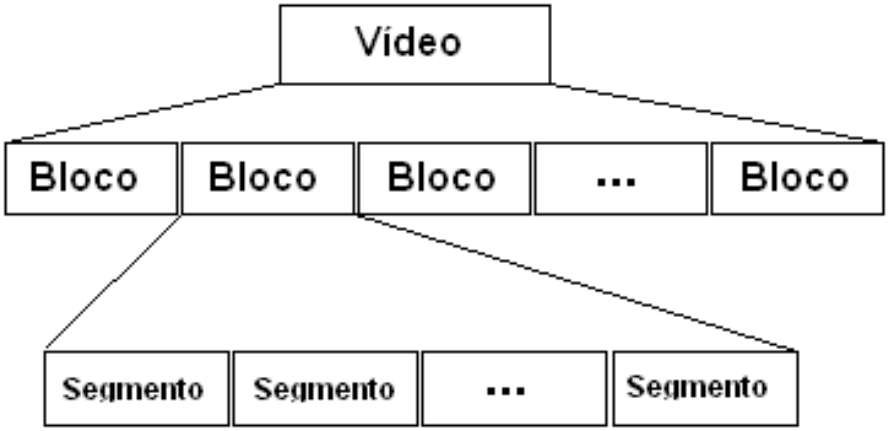

Figura 2 – Composição do vídeo

Depois de gerados os *requests*, é necessária a definição dos instantes de tempo em que cada usuário começa a solicitação do primeiro dos blocos que compõe o objeto, isto é, o instante de tempo em que os eventos gerados pela ferramenta

ocorrem é estabelecido. A fim de modelar os instantes das requisições, uma distribuição exponencial foi utilizada a partir da ferramenta R [21], gerando os intervalos entre *requests* do primeiro bloco de cada vídeo feita pelos usuários. O processo que determina instantes e intervalos acontece da seguinte maneira: a simulação é iniciada no instante zero e o primeiro valor gerado pela exponencial é somado ao momento de início da simulação, produzindo o instante do início do primeiro *request.* O segundo *request* tem seu instante determinado somando o segundo valor gerado pela exponencial ao instante em que ocorre o primeiro *request,* e os demais *requests* têm seus tempos iniciais gerados de maneira análoga. Esse procedimento, na verdade, está simulando a distribuição de Poisson, ideal para determinar a ocorrência de eventos como chegada de requisições em uma rede [18]. As requisições para os blocos subsequentes, já que os tempos foram gerados para o primeiro bloco de cada objeto, têm seus tempos definidos somando ao instante da requisição do bloco anterior um tempo constante previamente definido e atribuído ao tempo de *download* de um bloco. Essas informações, os instantes de tempo, são utilizadas na concepção das requisições, no entanto, não aparecem no arquivo de *requests.*

Uma característica relevante a ser considerada nesse formato de produção de um *workload* sintético é o fato de que todos os objetos pedidos são completamente requisitados, i.e, nenhum usuário interrompe um *download,* uma vez iniciado o *download* não é abortado ou interrompido.

```
 1  790 1 50
 2  1262 1 50
 3  2293 1 50
 4  161 1 50
 5  790 51 100
 6  1313 1 50
 7  1262 51 100
 8  2293 51 100
 9  161 51 100
10  790 101 150
11  1313 51 100
12  2427 1 50
13  1631 1 50
14  1262 101 150
15  2293 101 150
16  161 101 150
17  777 1 50
18  790 151 200
19  1313 101 150
20  2427 51 100
21  1631 51 100
22  1262 151 200
23  2293 151 200
24  161 151 200
25  777 51 100
```

Figura 3 – Exemplo de workload gerado

Na Figura 3, um trecho de um *workload* gerado pelo processo descrito é ilustrado. Nele, a primeira coluna identifica o request, a segunda coluna representa o identificador do vídeo, na terceira coluna, identifica-se o primeiro segmento da faixa requisitada e, por fim, a quarta coluna traz o identificador do último segmento da faixa desejada. Nesse *workload,* para cada objeto são realizadas vinte requisições e, na figura ilustrada, nenhum objeto foi completamente requisitado.

3.2. Descrição dos objetos

Além da geração do *workload*, também é necessário um arquivo capaz de fornecer informações básicas referentes aos objetos que serão compartilhados na rede *overlay*.

```
 1  3000
 2  1 1000
 3  2 1000
 4  3 1000
 5  4 1000
 6  5 1000
 7  6 1000
 8  7 1000
 9  8 1000
10  9 1000
11  10 1000
12  11 1000
13  12 1000
14  13 1000
15  14 1000
16  15 1000
17  16 1000
18  17 1000
19  18 1000
20  19 1000
21  20 1000
22  21 1000
23  22 1000
24  23 1000
25  24 1000
```

Figura 4 – Descrição dos objetos

Na Figura 4, é apresentado um pequeno trecho do arquivo descritor dos objetos de vídeo, que é composto da seguinte maneira: A primeira coluna apenas enumera as linhas do arquivo. Na segunda coluna da primeira linha o número total de vídeo a serem descritos é especificado. A partir da segunda linha, a segunda coluna representa o identificador do vídeo e a terceira coluna informa o número de segmentos que compõe o objeto (tamanho do objeto).

A seguir, é dada uma definição formal para o termo segmento e explica-se porque o arquivo de objetos os define com um único tamanho.

3.3. Descrição dos cenários

Como mencionado na seção anterior, os cenários reproduzidos são baseados em ASes reais, considerando várias de suas principais características, tais como os parâmetros da distribuição de probabilidade que modela a popularidade dos objetos, por exemplo.

O primeiro sistema autônomo, denominado AS397 e especificado na Tabela 1, tem 9315 usuários cada um requisitando apenas um vídeo e permanecendo na rede até o fim de seu *download*. Nesse AS, três mil objetos estão disponíveis, cada um deles tem 1 GB e 48% deles é requisitado mais de uma vez, ou seja, compõe o volume de tráfego considerado *cacheable*, sendo muito interessante para o estudo das *caches*.

Segundo [3], a distribuição que melhor modela a distribuição da popularidade dos vídeos é a Mandelbrot-Zipf, no caso do AS397, essa distribuição recebe como parâmetros o *skewness* 0,62 e o fator de *plateau* 8.

Tabela 1 – AS397

Número de usuários	9315
Número de objetos	3000
Número de blocos por vídeo	20
Número de requests	186300
Tráfego *cacheable*	48%
Popularidade (~AS397)	Mandelbrot-Zipf (*skewness* = 0.62, fator de *plateau* = 8)
Tamanho dos vídeos	1 GB

Nesse cenário, o volume de tráfego total é de aproximadamente 9,3 *terabytes* o que é um volume razoável, permitindo uma avaliação bastante realista. Nesse caso, são aproximadamente 4,5 *terabytes* que potencialmente podem ser armazenados pela *cache* (tráfego *cacheable*).

O número de *requests* é obtido através da multiplicação entre o número de usuários e o número de blocos em que o vídeo é dividido, são 20 blocos de 50 segmentos de um *megabyte*.

O segundo sistema autônomo que será descrito é identificado como AS95 e tem suas principais características expostas na Tabela 2.

Tabela 2 – AS95

Número de usuários	9316
Número de objetos	3000
Número de blocos por vídeo	20
Número de requests	186320
Tráfego *cacheable*	54%
Popularidade (~AS397)	Mandelbrot-Zipf (*skewness* = 0.6, fator de *plateau* = 50)
Tamanho dos vídeos	1 GB

Esse sistema autônomo tem 9316 usuários que requisitam objetos de vídeo. O número de vídeos é o mesmo do AS397 e, para facilitar qualquer interação entre as *caches,* os objetos são os mesmos, além de serem identificados através de referências análogas nos dois sistemas autônomos e segmentados da mesma forma, em 20 de blocos de 50 segmentos.

Nesse AS95, 54% do tráfego é *cacheable,* o que representa cerca de 5 *terabytes* de tráfego. Como dito anteriormente, essa porcentagem representa os objetos que são requisitados mais de uma vez e são passíveis de serem postos na *cache*. Nesse sistema autônomo, o volume de tráfego é, assim como no AS397, aproximadamente 9,3 *terabytes.* O cálculo para obter o número de *requests* também é feito com a multiplicação do número de usuários e o número de blocos (9316*20).

A distribuição que melhor modela a popularidade dos objetos nesse sistema é a Mandelbrot-Zipf com *skewness* 0,6 e fator de *plateau* 50.

A opção de construir dois ASes de tamanhos similares é feita para facilitar comparações e a colaboração entre as *caches* já que a cooperação faz sentido quando os sistemas têm tamanhos similares [1] [12].

Os cenários utilizados neste trabalho são basicamente os descritos nesta seção ou compostos a partir deles. No entanto, com base nos *workloads* especificados, algumas variações podem ser propostas a fim de uma melhor análise ou generalização.

O aumento do fator de *plateau,* a variação do *skewness*, a variação da porcentagem do tráfego *cacheable* dentre outros fatores afetam o desempenho das *caches* de diferentes formas. Entretanto, a avaliação do desempenho de uma *cache* variando exclusivamente tais características está fora do escopo deste trabalho.

4. Algoritmo de *cache* parcial

Como dito anteriormente na seção 1, esquemas de redes com *caches* na Internet são capazes de prover redução do tráfego da rede. Entretanto, *caches* são normalmente projetadas para atender as necessidades de arquivos tradicionais, como arquivos de texto e imagens. Essas *caches* não são adequadas para mídias contínuas, que tem tamanho grande comparado aos objetos tradicionais [19]. É importante considerar também o tipo de rede *overlay* envolvida com a *cache* que é, nesse caso, uma rede P2P. Além disso, no caso deste trabalho, o principal tipo de conteúdo apontado como gerador de tráfego *inter*-ISP é o VoD, que tem um tamanho relativamente grande.

Diante das características do tipo de mídia a ser tratado, um esquema especial de *caches* é necessário. É nesse contexto que um algoritmo de *cache* parcial, que leva em consideração as características específicas de P2P e VoD, foi escolhido como a melhor opção. Nas seções seguintes, será exposto o algoritmo escolhido, uma comparação com um algoritmo tradicional será realizada, um melhoria no algoritmo será proposta e, por fim, o desempenho das duas abordagens (original e modificada) será avaliado.

O algoritmo foi completamente implementado em C/C++, utilizando algumas estruturas de dados de eficiência comprovada a fim de obter um bom desempenho, além de gerar aproximadamente 700 linhas de código.

4.1. Descrição do algoritmo escolhido

A escolha do algoritmo de cache foi realizada diante das considerações feitas anteriormente. O algoritmo é descrito em [3] e trata das características relevantes de uma rede P2P e de VoD, tais como popularidade ponderada do objeto, popularidade dinâmica e tamanho dos objetos.

Na Figura 5, o pseudocódigo do algoritmo de *cache* parcial proposto em [3] é exposto.

Algoritmo de *cache* parcial

```
1.  se objeto i não estiver na cache
2.    insira um segmento de i na cache, removendo segmentos caso necessário
3.  então
4.    hit = interseção(segmentos de i na cache, segmentos requisitados)
5.    Yi = Yi + (hit/nº de segmentos de i na cache)
6.    cache miss = (faixa requisitada - hit)/(tamanho do segmento)
7.    x = (Yi/Y)*(tamanho médio dos objetos)
8.    k = mínimo(cache miss, máximo(1, x))
9.    se não tem espaço na cache para k segmentos
10.     remova k segmentos do objeto menos popular
11.   adicione k segmentos do objeto i na cache
12.   fim se
```

Figura 5 – Pseudocódigo do algoritmo

Os segmentos de um determinado objeto são postos na cache à medida que a popularidade do objeto e da faixa de segmentos requisitada cresce. Esse esquema também é benéfico quando se considera que segmentos iniciais são potencialmente mais populares que os finais, já que a cache não precisa armazenar o objeto completo.

Objetos em sistemas P2P são tipicamente requisitados em blocos e observando o pseudocódigo, é possível perceber que o algoritmo é preparado para suportar essa característica.

O tamanho médio dos objetos é considerado pelo algoritmo por conta da diversidade de mídias compartilhadas em sistemas P2P. Entretanto, neste trabalho, os tamanhos dos objetos foram mantidos constantes, a fim de simplificar os experimentos. A popularidade do objeto i é denotada por Yi e a popularidade do objeto mais popular é armazenada em Y. As variáveis *hit* e *cache miss* representam, respectivamente, o número de segmentos requisitados servidos pela cache e o número de segmentos requisitados não encontrados na cache.

Esse algoritmo prevê os seguintes cenários: (1) caso o bloco de segmentos requisitado seja totalmente encontrado na cache, o usuário será servido diretamente por ela; (2) caso o bloco não seja encontrado na *cache*, ele terá um segmento armazenado na *cache* segundo o algoritmo e o *request* é intermediado ou redirecionado, e; (3) caso o bloco seja parcialmente encontrado, a parte encontrada é servida pela *cache*, ocorre a decisão de se armazenar localmente os segmentos não encontrados de acordo com a execução do algoritmo escolhido e o *request* é redirecionado ou intermediado.

O redirecionamento ou intermediação do *request* durante o algoritmo pela *cache* são estratégias distintas e importantes, mas que fogem ao escopo proposto nesse trabalho.

4.2. Cache parcial x LFU x LRU

Para validar a escolha do algoritmo de *cache* parcial, foram realizados experimentos comparando o seu desempenho com conhecidos algoritmo de *cache* que fazem suas manipulações com o objeto inteiro, o LFU (*Least Frequently Used*) e o LRU (*Least Recently Used*). Nesse caso, o objetivo é certificar a melhor performance do algoritmo escolhido diante das características específicas do tráfego P2P na distribuição de VoD.

Nesse experimento que visa comprovar a efetividade do algoritmo parcial, o AS397 foi escolhido como cenário.

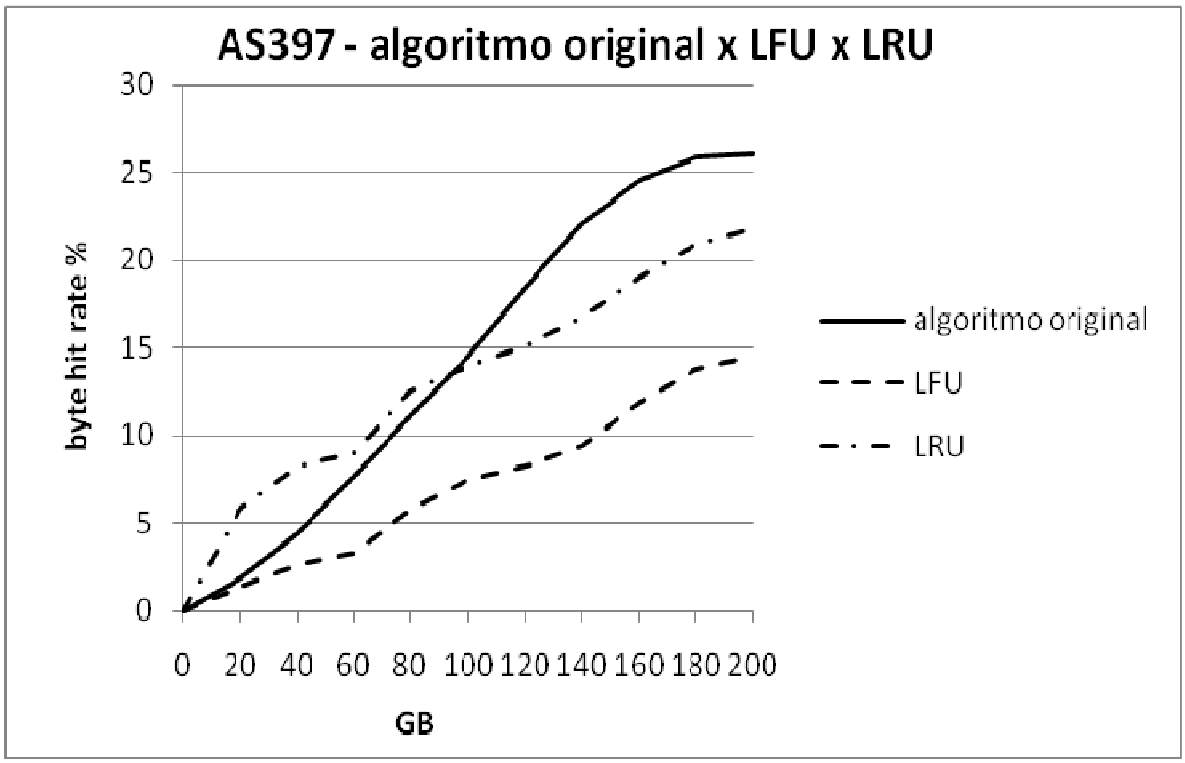

Figura 6 – Comparação entre o desempenho do algoritmo de *cache* parcial, LFU e LRU

Observando a Figura 6, fica claro que para o contexto em questão, o algoritmo parcial se mostra muito mais efetivo, já que ele chega a atingir pouco mais de 25% de taxa de acerto máximo enquanto o LFU alcança cerca de 15% de taxa de acerto para o mesmo tamanho de cache. Dessa forma, diante das características de *workload* que este trabalho aborda, o algoritmo escolhido é mais eficiente que o algoritmo tradicional em questão.

4.3. Modificação proposta

O primeiro passo para propor a modificação foi compreender o algoritmo, estudando detalhadamente cada uma de suas etapas. Dessa forma, foi possível perceber que, em certos casos, o algoritmo original demora um pouco para responder a mudanças de popularidade dos objetos. Com isso, o objetivo da modificação, é obter um algoritmo mais "agressivo".

A alteração acontece na linha 5 da Figura 5, a variável *hit* deixa de ser dividida pelo número de segmentos do objeto na *cache*. Essa modificação foi realizada visando obter uma resposta mais rápida do algoritmo diante das mudanças de popularidade dos objetos, que acontecem de acordo com o *workload* gerado.

4.4. Comparação: original x modificado

A fim de facilitar o entendimento das comparações, o algoritmo sem alterações será referenciado como algoritmo original ou básico e o proposto nesse trabalho de graduação como algoritmo modificado ou hit.

Para validar a efetividade dos algoritmos diante do *workload* gerado, uma série de experimentos foi realizada para os dois sistemas autônomos escolhidos para esse trabalho. Na Figura 7 e na Figura 8, os resultados de alguns dos experimentos para validação são expostos.

Na Figura 7, a relação entre performance e tamanho da *cache* é exibida. Os algoritmos têm seus desempenhos comparados no contexto do AS397, descrito na seção 3.

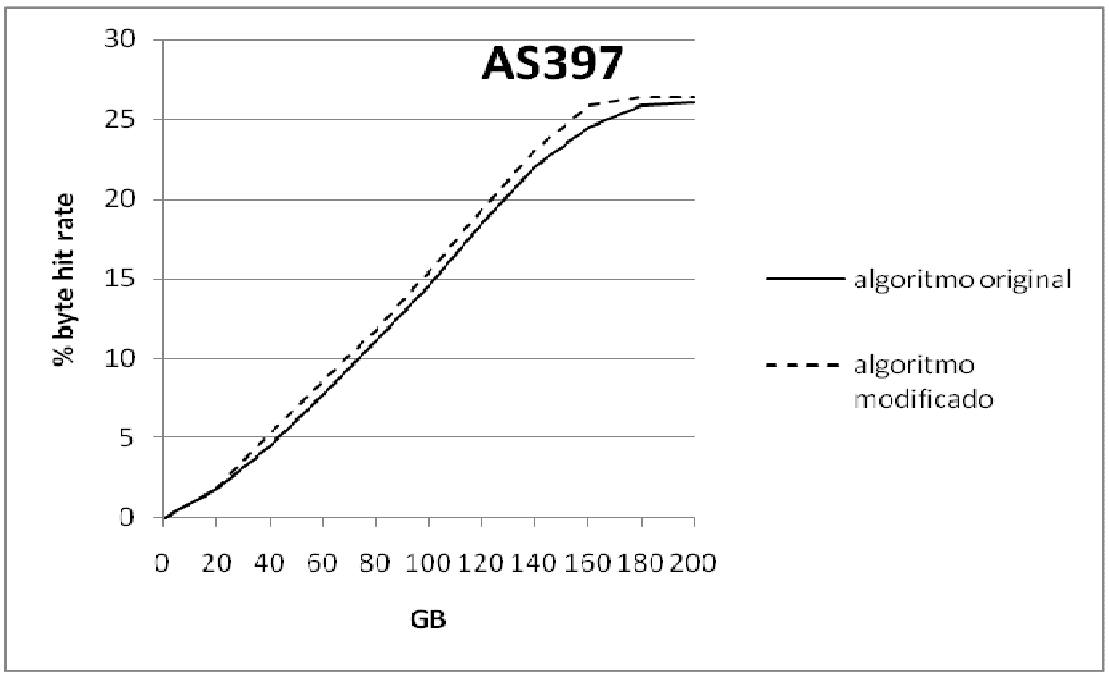

Figura 7 – Desempenho das variações no AS397

Ambos os algoritmos se estabilizam e atingem seu desempenho máximo com um tamanho de cache pequeno se comparado ao volume de tráfego do AS.

Tabela 3 – Dados da comparação dos algoritmos no AS397

Abordagem / Tamanho da *cache* (GB)	Algoritmo original	Algoritmo modificado
20	1,83%	1,88%
40	4,46%	5,27%
60	7,67%	8,5%
80	11,13%	11,85%
100	14,56%	15,44%
120	18,46%	19,29%
140	22,06%	23,15%
160	24,5%	25,83%
180	25,9%	26,4%
200	26,02%	26,4%
Média	15,66%	16,4%

Os dados que resultam no gráfico da Figura 7, estão presentes na tabela 3, a seguir. Observando a tabela 3, é perceptível que também na média de desempenho o algoritmo modificado supera o original. Embora a diferença entre as abordagens seja uma porcentagem pequena, isso representa muitos *gigabytes* nesse cenário e talvez signifique *terabytes* para *workloads* maiores.

Na Figura 8, o *workload* referente ao AS95 foi utilizado nos experimentos. Assim como no cenário anterior, o algoritmo modificado tem um desempenho levemente superior. Ainda analisando o gráfico da Figura 8, nota-se que a diferença entre as curvas cresce um pouco à medida que o tamanho da *cache* aumenta. Nesse caso, com um espaço de armazenamento razoável, o algoritmo modificado passa a responder mais rapidamente às alterações da rede, sendo mais efetivo.

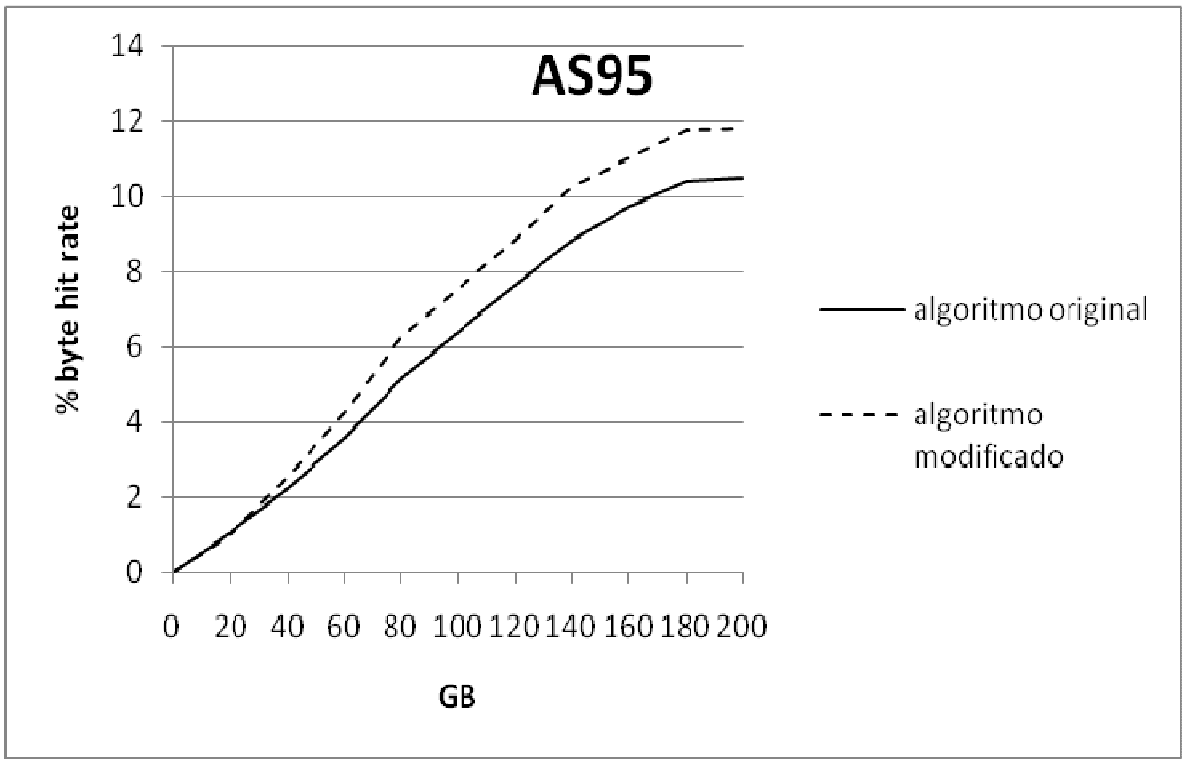

Figura 8 – Desempenho das variações no AS95

Os dados resultantes dos experimentos no AS95 e utilizados na composição do gráfico da Figura 8 são descritos na tabela 4. A considerável diferença entre os desempenhos para os diferentes sistemas autônomos é explicada pelas diferentes características por eles apresentada, como a variação do fator de *plateau* e as diferentes porcentagens de tráfego *cacheable*.

Tabela 4 – Dados da comparação dos algoritmos no AS95

Abordagem / Tamanho da *cache* (GB)	Algoritmo original	Algoritmo modificado
20	1,07%	1,02%
40	2,22%	2,52%
60	3,57%	4,24%
80	5,15%	6,26%
100	6,41%	7,57%
120	7,63%	8,85%
140	8,83%	10,26%
160	9,76%	11,04%

180	10,39%	11,78%
200	10,50%	11,83%
Média	6,55%	7,54%

O algoritmo modificado, de acordo com os experimentos executados, tem um desempenho levemente superior ao algoritmo original. Na tabela 4, por exemplo, o algoritmo modificado é aproximadamente 1% mais efetivo em média e mais de um por cento superior ao original no máximo desempenho obtido. Entretanto, ratifica-se que, apesar de se tratar de uma pequena porcentagem, o volume de tráfego relacionado é bastante significativo.

Nas situações apresentadas, os algoritmos alcançam seus desempenhos máximos com aproximadamente 200 *gigabytes* de *cache*, que é um tamanho bem pequeno considerando um *workload* de mais de nove terabytes. Além disso, o máximo *byte hit rate* atingido é, de maneira geral, bastante aceitável tendo em vista o desempenho dos algoritmos tradicionais como o avaliado neste trabalho e os que são especificados nos trabalhos envolvendo algoritmos de *cache* [3].

Diante do resultado desses experimentos, é possível observar que a performance dos algoritmos é razoável para o tipo de tráfego e de rede em questão, VoD e P2P respectivamente, e esse desempenho seria aceitável dependendo do objetivo a ser atingido.

A despeito disso, é importante perceber que a cooperação ainda pode prover uma melhora significativa para o sistema já que, até esse ponto, não foi implementado qualquer tipo de cooperação entre diferentes *caches* nos experimentos.

5. *Caches* cooperativas

Diversas soluções combinadas podem ser utilizadas para minimizar os problemas decorrentes do grande volume gerado pelo tráfego P2P.

Uma das opções para redução de tráfego *inter*-ISP é a construção de *caches.* No entanto, *caches* isoladas em provedores de serviço sem qualquer cooperação podem, nem sempre, ser capazes de atender às necessidades do tráfego, tendo em vista o grande volume. Outra possibilidade é que, mesmo isoladas, as *caches* atendam satisfatoriamente as necessidades da rede. Nos dois casos, a inserção de cooperação pode prover um serviço prestado pelas *caches* ainda melhor.

Apesar de ter um grande potencial, esquemas de *caches* cooperativas envolvendo tráfego P2P não têm sido muito abordadas na literatura. Em [12], é defendida a idéia de que o esquema de *caches* cooperativas é mais útil ao tráfego P2P que ao tráfego mais comum da *web*. Essa idéia se baseia, principalmente, nas características do tráfego P2P, como ser extremamente repetitivo [13] e ser, geralmente, imutável [14].

Essas características são bastante interessantes para *caches* cooperativas quando se considera a necessidade das *caches* de conhecerem os objetos de sua rede e das demais que com elas cooperam. São necessários identificadores únicos, além da necessidade das redes *overlays* comunicantes proverem os mesmos objetos ou a maior parte deles e terem aproximadamente o mesmo tamanho e volume de tráfego envolvido.

Existem, de maneira geral, dois modelos de cooperação entre *caches* apresentados em [12]: (1) cooperação entre *caches* de sistemas autônomos diferentes, e; (2) cooperação entre *caches* de um mesmo AS.

Neste trabalho, o foco será em *caches* situadas em diferentes sistemas autônomos, como é exposto no modelo da Figura 9.

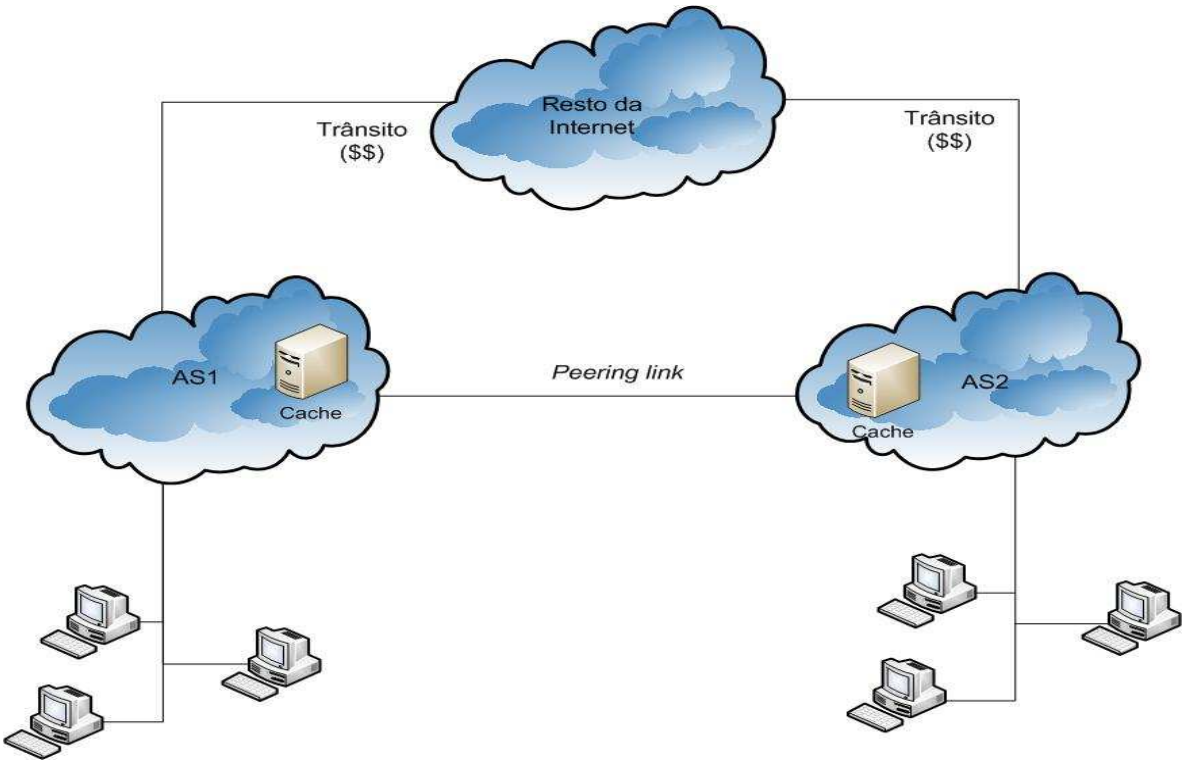

Figura 9 - Modelo de cooperação entre caches em diferentes ASes

O esquema ilustrado na Figura 9 é minimalista, explorando um caso simples, porém capaz de esclarecer diversas questões envolvendo cooperação de *caches.* Dessa forma, nada impediria a concepção de um cenário com um número maior de sistemas autônomos comunicantes.

Nesse modelo, as *caches* cooperam para servir os clientes das redes *overlay*. Os ASes que cooperam devem estabelecer relações de *peering* para carregar o tráfego entre os sistemas comunicantes. É possível também que essas *caches* estejam em uma mesma região geográfica, como uma cidade, por exemplo, que costuma ter *links* "internos" com largura de banda muito superior aos *links* entre regiões diferentes. Esses ASes normalmente estão conectados ao resto da Internet através de ISPs comerciais, com menos largura de banda e sendo muito mais custosos para os sistemas autônomos envolvidos.

Esquemas de *caches* cooperativas podem oferecer uma série de incentivos aos sistemas autônomos comunicantes. O tráfego gerado para o "resto da Internet" pode ser minimizado através do compartilhamento entre *peers* e também pelo

tráfego que transita pelo *peering link* entre os sistemas. Outro benefício desse esquema é encontrado analisando o ponto de vista da aplicação, é melhor para ela um esquema de cooperação que limitar o volume do tráfego P2P.

No caso de *caches* cooperativas, com a chegada de uma requisição no AS1, a *cache* responsável por atender o pedido, o analisa e provê os segmentos que ela contém e é capaz de servir localmente e redireciona para as *caches* que cooperam com ela, no caso da figura 5, a *cache* do AS2, a requisição dos segmentos restantes, que ainda não foram servidos. Nesse caso, os segmentos só não são obtidos através de uma *cache* quando nenhuma *cache* do grupo que coopera pode atender a requisição.

5.1. Modelo utilizado na cooperação

A fim de avaliar o desempenho da cooperação, um modelo simples foi proposto. Inicialmente, os cenários que representam o AS397 e o AS95 passaram a se comunicar e trocar requisições.

Os objetos de vídeos nos dois sistemas são os mesmos e identificados da mesma forma, como dito na descrição dos cenários. Essa característica viabiliza o redirecionamento simples de um *request*, sem a necessidade de uma prévia adaptação ou um posterior *overhead* desnecessário na *cache* que recebe o redirecionamento.

O cenário, da maneira que foi concebido, é próximo do ideal já que os sistemas autônomos têm características similares, como aproximadamente o mesmo tamanho, por exemplo, além de proverem os mesmos objetos. Esse tipo de cooperação é denominado cooperação perfeita segundo [12].

Para obter um único *workload* capaz de representar as requisições de dois ASes, os *workloads* gerados anteriormente foram combinados, adicionando um campo a mais para identificar o sistema autônomo de origem do pedido.

```
1  1 790 1 50
2  1 1262 1 50
3  1 2293 1 50
4  1 161 1 50
5  1 1313 1 50
6  1 2427 1 50
7  1 1631 1 50
8  1 777 1 50
9  0 790 1 50
10 1 2404 1 50
11 1 2735 1 50
12 1 2740 1 50
13 0 1262 1 50
14 1 790 51 100
15 1 1262 51 100
16 0 2293 1 50
17 0 161 1 50
18 1 2293 51 100
19 1 161 51 100
20 1 1313 51 100
21 1 2427 51 100
22 1 222 1 50
23 1 209 1 50
24 1 899 1 50
25 1 1631 51 100
```

Figura 10 – *Workload* combinado dos ASes

Na Figura 10, um trecho do workload que combina as requisições do AS397 e AS95 é exibido. Nele, uma coluna é adicionada, sendo composta por zeros e uns e identificando o AS de origem da requisição. Nesse caso, zero corresponde ao AS397 e um representa o AS95.

A interação entre os sistemas autônomos pode gerar diferentes destinos para um determinado *request*. Inicialmente, com a chegada de um *request* em uma *cache* localmente, as características do sistema podem ser alteradas e os objetos armazenados podem ser alterados. Além disso, as seguintes situações podem ser observadas: (1) a requisição é completamente atendida localmente, pela *cache* do sistema autônomo; (2) a requisição é parcialmente atendida pela *cache* local, e; (3) nenhuma parte do pedido é encontrada na *cache* do sistema autônomo. No primeiro caso, em um cenário de cooperação, nada precisa ser feito visto que a

requisição foi satisfeita. No entanto, no segundo e no terceiro casos, o procedimento é diferente. Em ambos, a parte do *request* que não foi atendida, seja a requisição completa ou uma parcela dela, será redirecionada para a *cache* do sistema autônomo vizinho, que coopera com a local nesse modelo.

Com o redirecionamento desse 'novo' *request* para uma *cache* vizinha, as mesmas situações percebidas na *cache* local podem ser observadas, embora as características do sistema, como popularidade dos objetos e vídeos armazenados, não são alteradas. A diferença é que, quando não pode servir a requisição, a *cache* informa qual parcela do *request* não pôde ser atendida e, a partir dessa informação, a *cache* inicial, para a qual o *request* foi realizado inicialmente, produz tráfego *inter*-ISP para servir a parte da requisição que não foi provida pelo sistema de *caches*.

6. Resultados e experimentos

Nessa seção, os resultados dos experimentos realizados para o cenário de cooperação entre *caches* é apresentado. Diante do *workload* gerado (descrito na seção anterior), será avaliado o desempenho de cada uma das abordagens do algoritmo, tanto o original e o modificado, no contexto de cooperação entre *caches.* Os dois ASes utilizados nesse trabalho têm seus desempenhos avaliados com e sem cooperação.

Em ambos os casos, são apresentadas tabelas com os dados obtidos para diversos tamanhos de *caches* e gráficos expondo curvas comparativas. Além disso, a análise do volume de tráfego que utiliza o *peering link* é apresentada.

6.1. Cooperação – algoritmo original

Nessa seção, foram consideradas duas situações distintas para serem analisadas. Na primeira, os sistemas autônomos AS95 e AS397 realizam suas operações normalmente com base no algoritmo original, sem que suas *caches* se comuniquem. No segundo caso, a "conversa" entre as *caches* é estabelecida, ainda com base no algoritmo básico.

Tabela 5 – Dados da comparação da cooperação no AS397 com o algoritmo original

Abordagem / Tamanho da *cache* (GB)	Sem cooperação	Com cooperação
20	1,83%	3,35%
40	4,46%	7,77%
60	7,67%	13,1%
80	11,13%	19,35%
100	14,56%	25,4%
120	18,46%	31,9%
140	22,06%	38,1%
160	24,5%	42,38%

180	25,9%	44,98%
200	26,02%	45,25%
Média	15,66%	27,16%

Nas tabelas 5 e 6, os *byte hit rate* obtidos após a simulação do algoritmo para as duas situações citadas são apresentados. Observando a Figura 11 e a Figura 12, é possível perceber que o comportamento das curvas é similar, entretanto o resultado conseguido utilizando a cooperação entres os ASes é notavelmente mais satisfatório em cada um dos gráficos.

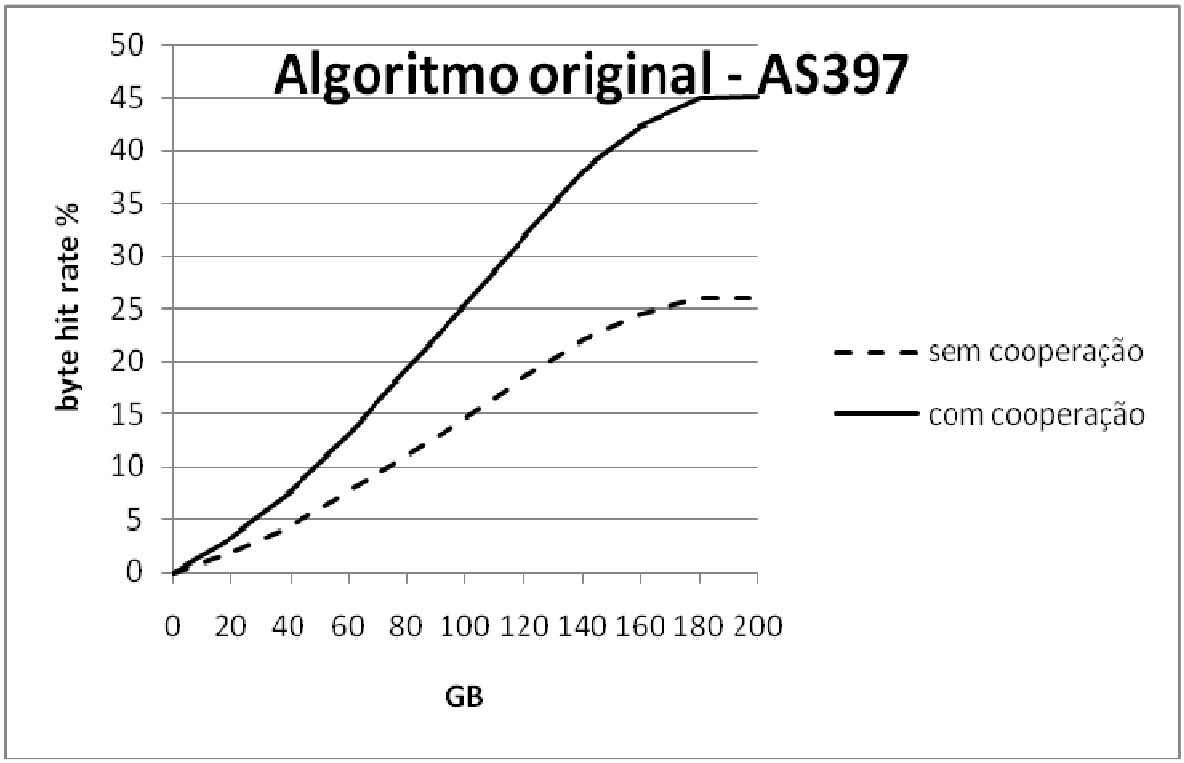

Figura 11 - Algoritmo original AS397 com e sem cooperação

No caso do AS397 utilizando o algoritmo original, aproximadamente 1163 *gigabytes* de volume de tráfego são servidos pela *cache* local no cenário sem cooperação. Para o cenário cooperativo, o mesmo volume de tráfego é servido localmente e cerca de 860 *gigabytes* são provenientes da cooperação, via *peering links*, que são menos custosos para o AS que a comunicação normalmente efetuada. Dessa forma, esse volume de tráfego oriundo do sistema autônomo vizinho deixa

de se transformar em tráfego externo (*inter*-ISP) e é considerado, portanto, uma economia financeira já que se paga menos pela utilização do *peering link* que pela utilização dos *links* de ISPs externos.

Tabela 6 – Dados da comparação da cooperação no AS95 com o algoritmo original

Abordagem / Tamanho da *cache* (GB)	Sem cooperação	Com cooperação
20	1,07%	2,52%
40	2,22%	5,28%
60	3,57%	8,52%
80	5,15%	12,15%
100	6,41%	15,57%
120	7,63%	19,15%
140	8,83%	22,07%
160	9,76%	24,17%
180	10,39%	25,26%
200	10,50%	25,41%
Média	6,55%	16,01%

Considerando o desempenho do AS95 para o mesmo cenário, pode-se perceber que a melhora no *byte hit rate* também é considerável. A *cache* local provê para os *peers* internos ao sistema autônomo, um volume aproximado de 528 GB de conteúdo. Quando a cooperação é inserida, o total de tráfego servido por intermédio de *caches* chega a aproximadamente 1278 GB, o que representa uma melhora significativa para esse caso.

Em ambos os sistemas, a cooperação é benéfica do ponto de vista de redução de tráfego *inter*-ISP e conduz o *byte hit rate* a se tornar aproximadamente o dobro do que seria obtido com sistemas isolados.

Fica claro também que mesmo o algoritmo original, levemente menos eficaz que o algoritmo modificado, tem uma performance aceitável e melhora consideravelmente o seu desempenho quando coopera.

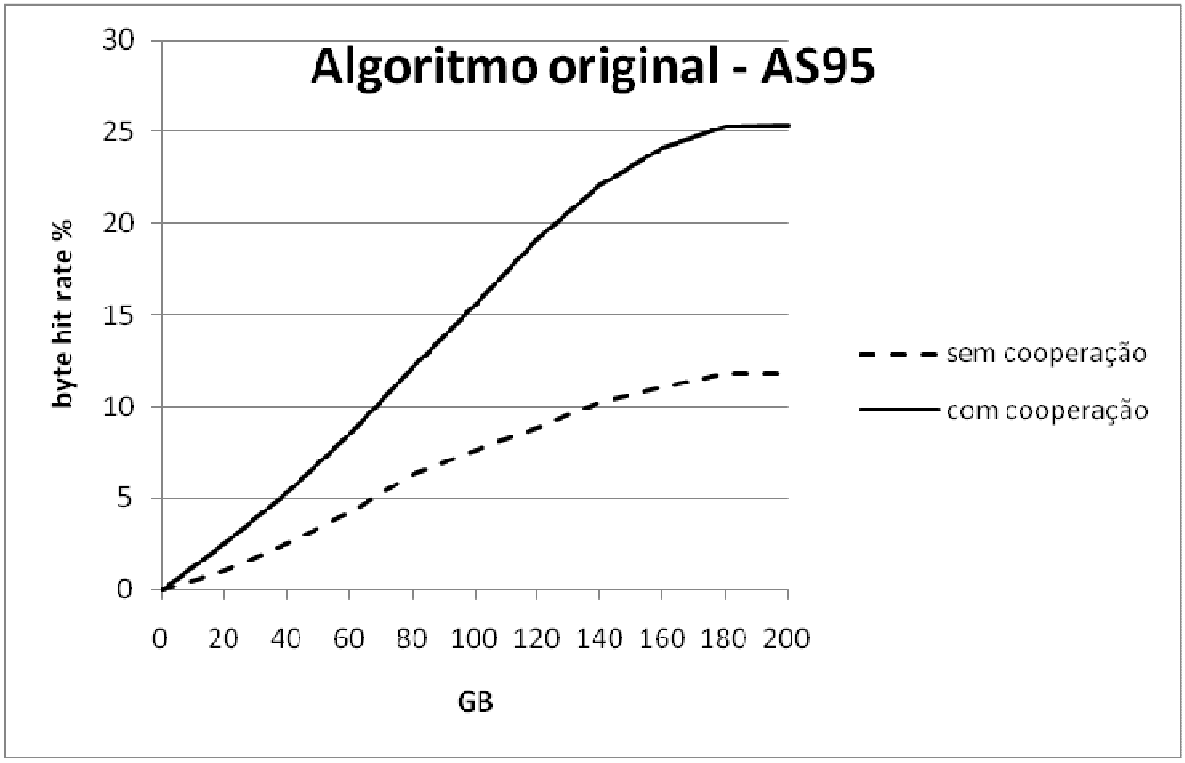

Figura 12 - Algoritmo original AS95 com e sem cooperação

Esse melhor desempenho é ainda mais evidenciado quando se observa as tabelas 7 e 8. Nelas, está quantificado em *gigabytes* o volume em que cada tipo de tráfego é identificado e o total de tráfego que circula internamente em cada sistema autônomo. Nesse caso, a cooperação implica em uma utilização razoável do *peering link*, tanto no partindo do AS397 para o AS95 como no sentido inverso. Além disso, é perceptível também que o tráfego no *link* de cooperação é equivalente nos dois sentidos.

Tabela 7 – Resultados da cooperação entre *caches* no AS397 (algoritmo original)

Tipo de tráfego / **Esquema**	**Local**	***Peering link***	***Inter*-ISP**	**Total**
Sem cooperação	1163 GB	0 GB	8152 GB	9315 GB
Com cooperação	1163 GB	860 GB	7292 GB	9315 GB

Tabela 8 – Resultados da cooperação entre *caches* no AS95 (algoritmo original)

Tipo de tráfego / **Esquema**	**Local**	***Peering link***	***Inter*-ISP**	**Total**
Sem cooperação	528 GB	0 GB	8788 GB	9316 GB
Com cooperação	528 GB	750 GB	8038 GB	9316 GB

6.2. Cooperação – algoritmo modificado

Assim como na seção anterior, foram propostos dois cenários que estariam sujeitos a análises nessa seção: o primeiro leva em consideração a situação com *caches* isoladas, sem comunicação entre elas, e; o segundo, trata do caso em que as *caches* passam a cooperar. Entretanto, a análise realizada será baseada na performance do algoritmo modificado, que já havia apresentado melhores resultados que o algoritmo original.

Tabela 9 – Dados da comparação da cooperação no AS397 com o algoritmo modificado

Abordagem / Tamanho da *cache* (GB)	Sem cooperação	Com cooperação
20	1,88%	3,27%
40	5,27%	9,44%
60	8,5%	15,71%
80	11,85%	23,69%
100	15,44%	29,73%
120	19,29%	36,65%
140	23,15%	43,19%
160	25,83%	46,71%
180	26,4%	49,17%
200	26,4%	49,27%
Média	16,4%	30,68%

A tabela 9 apresenta as porcentagens de acerto obtidas para os diferentes tamanho de *cache* no AS397 com e sem cooperação. Nesse caso, é possível observar a considerável melhora, tanto no máximo *byte hit rate* quanto na média, do sistema cooperativo. Na Figura 13, fica evidente que, apesar de um comportamento similar das curvas, a redução do tráfego gerado para ISPs externos com cooperação é muito maior.

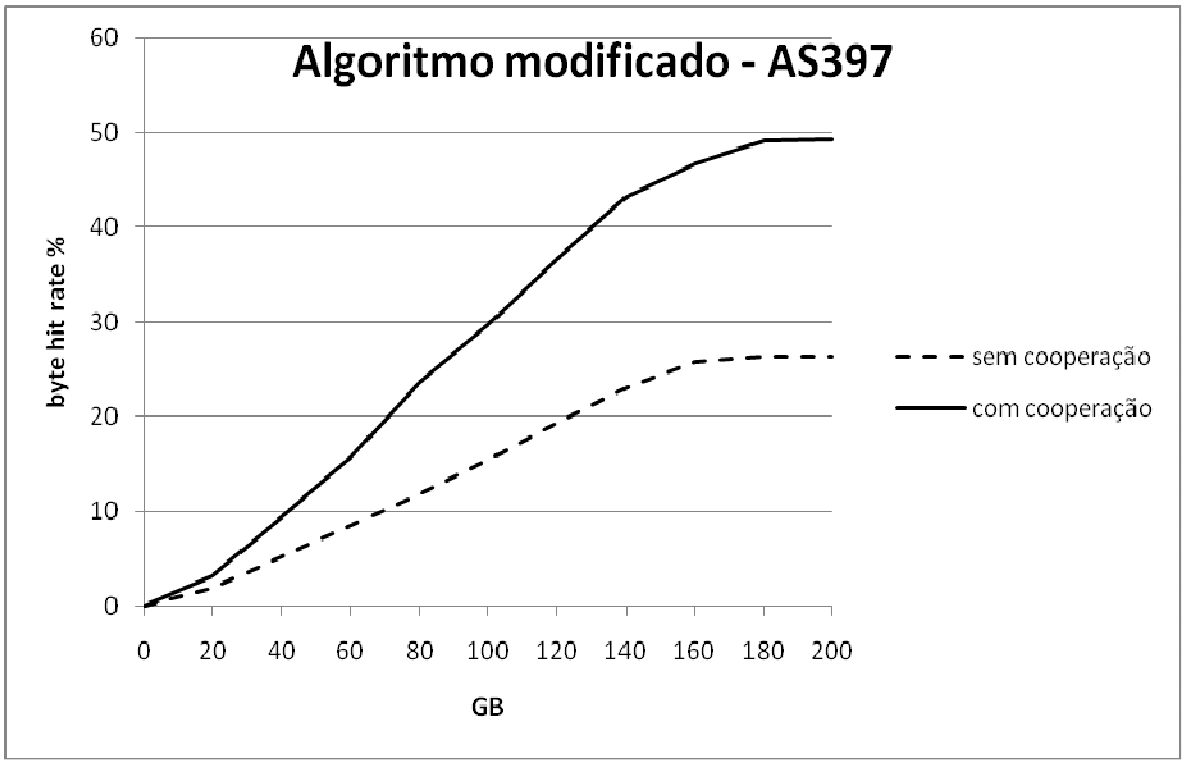

Figura 13 – Algoritmo modificado AS397 com e sem cooperação

No AS397 utilizando o algoritmo modificado, diante de um tráfego de cerca 4,5 *terabytes cacheable,* mais de um *terabyte,* aproximadamente 1180 GB, é servido localmente pela *cache* sem cooperação. Quando cooperando, o sistema consegue obter, proveniente do conjunto de *caches* envolvidas, 2203 GB de conteúdo, ou seja, mais de dois *terabytes*.

Na tabela 10, os valores conseguidos da simulação do algoritmo modificado para o AS95 com e sem cooperação estão disponíveis. Nesse sistema autônomo, a

melhora obtida com cooperação representa um *byte hit rate* igual a mais de duas vezes o valor obtido sem cooperação.

Tabela 10 – Dados da comparação da cooperação no AS95 com o algoritmo modificado

Abordagem / Tamanho da *cache* (GB)	Sem cooperação	Com cooperação
20	1,02%	2,46%
40	2,52%	5,89%
60	4,24%	9,44%
80	6,26%	13,43%
100	7,57%	16,93%
120	8,85%	20,67%
140	10,26%	23,98%
160	11,04%	25,93%
180	11,78%	26,91%
200	11,83%	26,98%
Média	7,54%	17,26%

A diferença entre os desempenhos fica mais clara quando se observa a Figura 14. Nela, o máximo desempenho é atingido com uma *cache* de 200 GB e nesse caso, a diferença entre as curvas é bastante grande.

No cenário do AS95 representado pela Figura 14, tem-se 595 GB servidos pela *cache* do próprio sistema. Já considerando a cooperação, mais 762 GB são obtidos além do que já era servido localmente, ou seja, são 1357 GB obtidos sem que seja gerado tráfego *inter*-ISP. Esse valor é bastante alto quando é levada em consideração a parcela de tráfego *cacheable,* que está em torno de 5 *terabytes*.

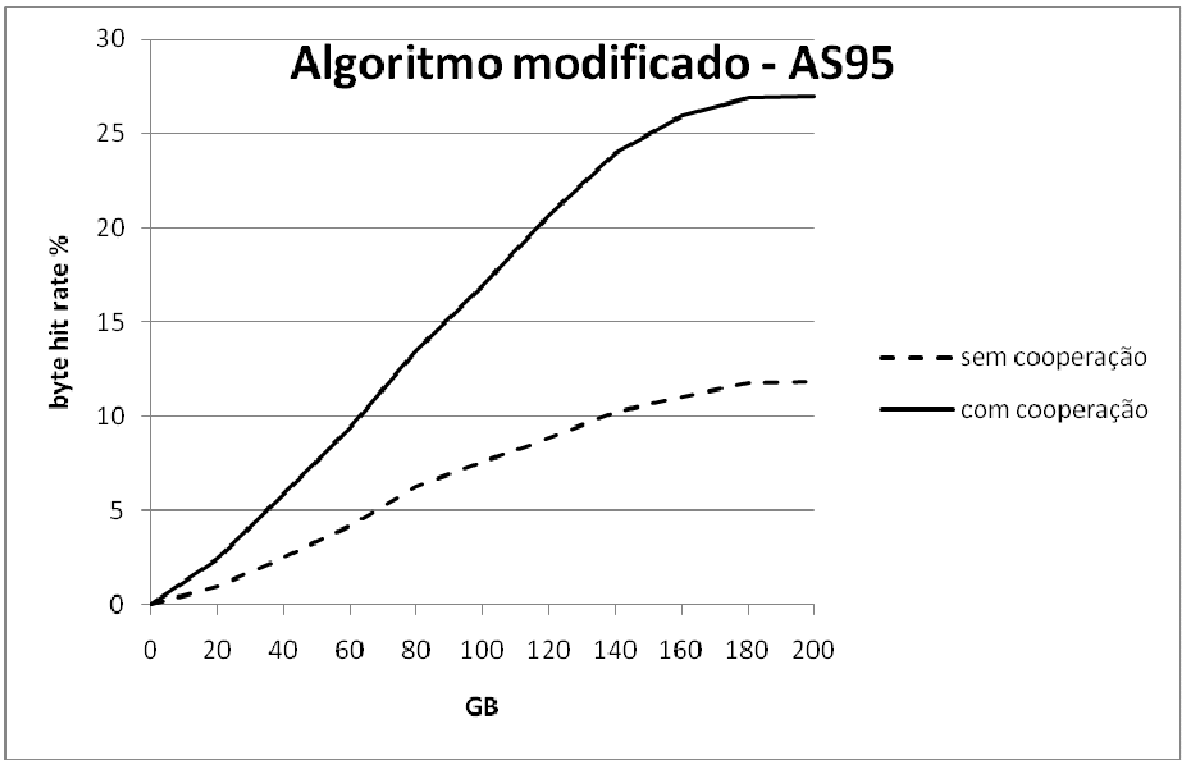

Figura 14 - Algoritmo modificado AS95 com e sem cooperação

Por fim, observando as tabelas 11 e 12, fica evidente que o *peering link* carrega uma parcela considerável do tráfego, tanto do AS397 para o AS95 como no sentido inverso, reduzindo, dessa forma, o tráfego *inter*-ISP. Isso significa uma redução de muitos *gigabytes* no volume de tráfego que o sistema autônomo gera para ISPs externos. Em contrapartida, o volume servido localmente não é alterado, devido ao esquema utilizado e que foi descrito anteriormente

Tabela 11 – Resultados da cooperação entre *caches* no AS397 (algoritmo modificado)

Tipo de tráfego / **Esquema**	**Local**	***Peering link***	***Inter*-ISP**	**Total**
Sem cooperação	1180 GB	0 GB	8135 GB	9315 GB
Com cooperação	1180 GB	1023 GB	7112 GB	9315 GB

Tabela 12 – Resultados da cooperação entre *caches* no AS95 (algoritmo modificado)

Tipo de tráfego / Esquema	Local	*Peering link*	*Inter*-ISP	Total
Sem cooperação	595 GB	0 GB	8721 GB	9316 GB
Com cooperação	595 GB	762 GB	7959 GB	9316 GB

6.3. Considerações finais

Os experimentos realizados permitiram a exploração de todo o potencial da cooperação entre *caches* visto que o cenário é perfeito. Os resultados demonstraram, de maneira geral, um melhor desempenho das *caches* que não deve ser desprezado.

As *caches* analisadas do ponto de vista do esquema cooperativo mostraram uma taxa de acerto, em ambos os sistemas autônomos expostos, de aproximadamente o dobro do *byte hit rate* obtido em esquemas sem cooperação, evolução essa bastante considerável.

É muito importante nesse trabalho frisar que, como já havia sido dito anteriormente, mesmo uma melhora equivalente a pequenas porcentagens representam consideráveis volumes de tráfego em uma rede P2P com objetos grandes, que é o caso da distribuição de VoD. Nos experimentos presentes neste trabalho, são muitos *gigabytes* de tráfego *inter*-ISP reduzidos em decorrência da cooperação e, em um *workload* real, a redução do volume do tráfego provida poderia chegar a ordem de *terabytes*. Esse fato é ratificado nas tabelas que são expostas no fim das duas subseções anteriores, o tráfego nos *peering links* em todos os casos saiu do zero, para o caso sem cooperação, para muitos *gigabytes* com cooperação. Em alguns casos, esse volume supera a quantidade de tráfego servido pela *cache* local.

Dessa forma, fica evidente que o tráfego *inter*-ISP reduzido a partir da cooperação é muito grande, não se tratando de um volume de tráfego a ser desconsiderado e sendo um incentivo suficiente para os ISPs efetuarem a construção de *caches* e permitirem a cooperação entre elas. Além disso, o fato dos ASes terem aproximadamente o mesmo tamanho e fornecerem uma quantidade de conteúdo próxima através do *peering link* é um incentivo adicional para a realização do acordo entre os sistemas analisados neste trabalho.

7. Conclusões e trabalhos futuros

A finalidade deste trabalho era apresentar, de maneira clara, o impacto da cooperação de *caches* para o tráfego *inter*-ISP. Nesse processo, foram estudadas as características dos possíveis *workloads,* desde a distribuição que modela o cenário até o tamanho dos objetos, e a técnica de *cache* parcial a ser aplicada, considerando a rede e o tipo de conteúdo estudados, além das formas de cooperação existentes. Dessa forma, para atingir os objetivos propostos, algumas etapas foram primordiais tais como: (1) geração do *workload* segundo as características do cenário obtido de dois sistemas autônomos a fim de reproduzir a realidade; (2) a escolha, a análise e a proposta de modificação do algoritmo de *cache* parcial escolhido como base para o comportamento das *caches*, e; (3) a proposta de um modelo de cooperação e a avaliação dos resultados obtidos.

Os resultados obtidos foram, de maneira geral, bastante satisfatórios. Eles comprovam o bom desempenho de esquemas de cooperação entre *caches,* sendo uma alternativa viável para redes P2P na distribuição de VoD. Essa alternativa não compromete o tráfego da rede *overlay* (não o limita) e mantém, até certo ponto, o tráfego dentro de um sistema autônomo, provendo localidade e reduzindo o volume de tráfego *inter*-ISP, seja com a *cache* local ou com a cooperação através do *peering link.* Embora um esquema de cooperação traga algumas questões relevantes que não foram abordadas nesse trabalho, como o *overhead* gerado e a realização dos acordos entre os ISPs, a melhora conseguida através da cooperação se configura para os ISPs como um incentivo suficiente para construção de esquemas cooperativos , já que existe o tráfego de bastante conteúdo através de *links* menos custosos. A comprovação da eficiência da cooperação entre *caches* do ponto de vista do tráfego *inter*-ISP é, provavelmente, o principal resultado deste trabalho.

O principal problema relacionado ao desenvolvimento deste trabalho encontrado foi a necessidade de comprovação da corretude do algoritmo utilizado, bem como sua implementação, tanto isolado, quanto cooperando. Diversos

experimentos foram realizados nesse processo e o curto espaço de tempo para o desenvolvimento se configurou como um grande obstáculo no decorrer do trabalho.

Embora tenha atingido os objetivos propostos, este trabalho apresenta a possibilidade de realizar uma série de possíveis trabalhos futuros. Uma razoável extensão dos experimentos realizados é a variação do tamanho dos objetos compartilhados e do tamanho dos segmentos, avaliando um cenário ainda mais realístico [3] [6].

Outra alternativa é a análise da cooperação segundo um conjunto com mais duas *caches* como em [1] [12], considerando buscas pelo objeto e com conteúdo maior a ser compartilhado, avaliando o comportamento de um grupo de *caches*. Além da criação de um cenário não-perfeito.

Por fim, uma extensão bastante interessante e importante desse trabalho é a análise do *overhead* gerado pela cooperação, bem como questões pertinentes à realização dos acordos entre diferentes ISPs, para que os benefícios e prejuízos decorrentes do esquema proposto sejam avaliados e a melhor abordagem para diferentes situações seja determinada.

8. Referências

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

2. 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),* 2006, 135-149

3. Hefeeda, M. and Saleh, O., "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems" *in IEEE/ACM Transactions on Networking,* 2008, *16*

4. Tang, W.; Fu, Y.; Cherkasova, L. and Vahdat, A., "MediSyn: A Synthetic Streaming Media Service Workload Generator" *in Proc. NOSSDAV'03,* 2003

5. S. Jin, A. Bestavros, and A. Iyengar, "Network-aware partial caching for internet streaming media" *in ACM Multimedia Syst. J.*, vol. 9, no. 4, pp. 386–396, Oct. 2003.

6. Busari, M. & Wiliamson, C., "ProWGen: A Synthetic Workload Generation Tool for Simulation evaluation of Web Proxy Caches" *at Journal of Computer Networks,* 2002, 779-794

7. Karagiannis, T.; Rodriguez, P. and Papagiannaki, K., "Should Internet Service Providers Fear Peer-Assisted Content Distribution?" *Internet Measurement Conference 2005,* 2005, 63-76

8. Saleh, O., "Modeling and Caching of Peer-to-Peer Traffic" *at School of Computing Science, SIMON FRASER UNIVERSITY,* 2006

9. Blond, S. L.; Legout, A. and Dabbous, W., "Pushing BitTorrent Locality to the Limit" 2009*, version 2*

10. Guo, L.; Chen, S.; Ren, S.; Chen, X. and Jiang, S., "PROP: a Scalable and Reliable P2P Assisted Proxy Streaming System" *in Proceedings of the 24th International Conference on Distributed Computing Systems,* 2004

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

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

13. Leibowitz, N.; Bergman, A.; Ben-Shaul, R.and Shavit, A., "Are File Swapping Networks Cacheable? Characterizing P2P Traffic" in *Proceedings of the 7th International WWW Caching Workshop,* 2002

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

15. Choffnes, D. R. & Bustamante, F. E., "Taming the Torrent - A practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems" in *SIGCOMM'08,* 2008, 17-22

16. BitTorrent, inc. http://www.bittorrent.com

17. Gnutella. http://www.gnutella.com

18. R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley-Interscience, New York, NY, April 1991, ISBN:0471503361.

19. Park, S. H.; Lim, E. J. & Chung, K. D.; "Popularity-based Partial Caching for VOD Systems using a Proxy Server" in *Proceedings of the 15th international Parallel &Amp; Distributed Processing Symposium,* 2001, p. 115

20. Wu, K.-L.; Yu, P. S. and Wolf, J. L., "Segmentation of Multimedia Streams for Proxy Caching" in *IEEE Transactions on multimedia,* October 2004, vol. 6

21. The R Project for Statistical Computing. http://www.r-project.org/