Projeto e análise de um sistema de nano caches residenciais para distribuição de vídeo*

Gabriel Mendonça[1], **Rosa M. M. Leão**[1]

[1]COPPE/Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro

{gabriel,rosam}@land.ufrj.br

Abstract. *In this paper, we present a P2P video-on-demand (VoD) system that utilizes set-top boxes (STBs) and home gateways for content distribution; we propose a distributed load balancing algorithm that considers the resource constraints in these devices; and describe a probabilistic model that allows a significant reduction of the system's state space: from exponential to polynomial on the number of peers. We evaluate our platform's performance through simulations, using data from a real VoD system in which content popularity seems to follow a Zipf distribution with exponential cutoff. The results show that the proposed algorithm significantly reduces video download time in swarms with low and medium popularity.*

Resumo. *Neste artigo, apresentamos um sistema P2P para transmissão de vídeo sob demanda (VoD) usando* set-top boxes *(STBs) e roteadores domésticos, propomos um algoritmo distribuído para balanceamento de carga que considera a limitação de recursos desses dispositivos e descrevemos um modelo probabilístico que permite a redução do espaço de estados: de exponencial para polinomial no número de* peers*. Avaliamos o desempenho da plataforma através de simulações com dados de um sistema real de VoD, em que a popularidade dos conteúdos é bem modelada por uma distribuição Zipf com decaimento exponencial. Os resultados mostram que, para os* swarms *de baixa e média popularidade, nosso algoritmo reduziu consideravelmente o tempo médio de* download*.*

1. Introdução

As aplicações multimídia têm se tornado cada vez mais populares nos últimos anos. Esse cenário tem motivado diversas propostas para transmissão de vídeo usando arquiteturas *peer-to-peer* (P2P), em que os *peers* participam do processo de distribuição fornecendo recursos para o sistema como banda e capacidade de armazenamento. Aplicações de sucesso como PPLive, QQLive e PPS.tv (PPStream) [1] utilizam essa metodologia para distribuir grandes volumes de tráfego de vídeo na Internet.

Provedores de serviço de Internet (ISP) também podem se beneficiar usando redes P2P. Encontramos na literatura diversas propostas que utilizam os recursos disponíveis em *set-top boxes* (STBs) e/ou roteadores domésticos para a descentralização da distribuição de conteúdo em redes de banda larga [Laoutaris et al. 2008, Cha et al. 2008,

*Este trabalho foi parcialmente financiado pelo CNPq, CAPES e FAPERJ.

[1]http://www.pplive.com/en/index.html, http://live.qq.com/ e http://www.pps.tv/

Whiteaker et al. 2012]. Esse é um cenário favorável para arquiteturas P2P, já que esses dispositivos encontram-se sob controle do provedor de serviço e, em geral, estão ligados durante a maior parte do tempo.

Entretanto, muitos trabalhos [Wu and Lui 2012, Tan and Massoulié 2013] se restringem ao estudo do problema de alocação de conteúdos nos STBs. E, com frequência, as limitações impostas por um dispositivo embarcado não são modeladas com o devido detalhe. Em geral [Suh et al. 2007, Munoz Gea et al. 2012], considera-se apenas um limite no número de conexões de *upload* e *download* que um *peer* é capaz de estabelecer. Além disso, não temos conhecimento de trabalhos sobre redes P2P formadas por STBs que assumiram que a popularidade dos contéudos segue uma distribuição Zipf com decaimento exponencial, como observado em sistemas reais como YouTube e Netflix [Cha et al. 2007].

O objetivo deste trabalho é propor e analisar uma plataforma P2P para transmissão de vídeo sob demanda (VoD) usando nano caches. Os nano caches são dispositivos sob controle de um provedor de serviço de Internet (ISP) presentes na casa dos usuários, como set-top boxes (STBs) e roteadores domésticos. Esses dispositivos formam uma rede P2P, utilizando a banda dos clientes para distribuição de vídeos armazenados localmente. Para o projeto e avaliação dessa plataforma, levamos em consideração circunstâncias reais como uma distribuição de popularidade com decaimento exponencial e uma limitação do número de conteúdos que um dispositivo é capaz de servir simultaneamente.

Nossas principais contribuições incluem:

- Proposta e avaliação de um algoritmo distribuído para balanceamento de carga através da escolha dos conteúdos servidos por cada nano cache.
- Modelo probabilístico para o processo de escolha dos conteúdos servidos, que permite redução significativa do espaço de estados na representação do sistema.
- Análise do desempenho do sistema através de simulações usando parâmetros extraídos de um sistema real de vídeo sob demanda.

Inicialmente, discutimos os trabalhos relacionados na seção 2. Na seção 3, descrevemos a arquitetura do nosso sistema, nosso algoritmo para escolha dos conteúdos a serem servidos e nosso protótipo embarcado desenvolvido para prova de conceito. Em seguida, na seção 4, apresentamos modelos para o limite de capacidade do sistema e para a política de troca de *swarms*. Por fim, apresentamos na seção 5 os resultados de nossas simulações e na seção 6 nossas conclusões.

2. Trabalhos relacionados

Suh *et al.* [Suh et al. 2007] propõem um sistema *peer-to-peer* de VoD usando STBs baseado em operações *push*, em que o conteúdo é "empurrado" pelo provedor de conteúdo para os *peers* em uma etapa anterior à transmissão dos vídeos. Essa proposta, portanto, depende de uma gerência centralizada dos conteúdos armazenados pelos *peers*. Em nosso trabalho, assumimos um algoritmo descentralizado baseado em uma política LRU de substituição de cache, em que um *peer* armazena localmente os conteúdos por ele assistidos.

Em [Munoz Gea et al. 2012], os autores propõem um modelo analítico em que cada STB é modelado como uma fila. Nesse trabalho, o desempenho do sistema é avaliado

a partir da probabilidade de um cliente que assiste a um conteúdo (*leecher*) não encontrar nenhum STB com capacidade disponível para serví-lo. Assim como em [Suh et al. 2007], o número de conexões de *upload* que um STB estabelece é limitado, mas assume-se que um *peer* é capaz de servir qualquer número de conteúdos dentre os que estão armazenados localmente. Em nosso trabalho, assumimos que, devido às limitações de memória e processamento de um dispositivo embarcado, este não é capaz de servir, ao mesmo tempo, todos os conteúdos que armazena.

O problema de colaboração entre *swarms* na distribuição de arquivos com BitTorrent foi estudado em [Chen and Yan 2009, Meng et al. 2013]. Já em [Wang et al. 2010, He et al. 2012, Huang et al. 2008], o mesmo problema foi estudado no contexto de aplicações de VoD P2P. Assim como em nossa proposta, esses autores propõem que a escolha dos conteúdos que um *peer* serve seja feita com base em uma métrica de carga de um *swarm*. A métrica que definimos neste trabalho possui como vantagem o fato de ser facilmente calculada usando informações de um *tracker* ou DHT. As estratégias propostas em [Wang et al. 2010, He et al. 2012], por exemplo, dependem do conhecimento da banda de *upload* dedicada pelos servidores de conteúdo a cada *swarm*, uma informação que não pode ser obtida tão facilmente.

Quanto à popularidade dos conteúdos, grande parte dos trabalhos [Munoz Gea et al. 2012, Wu and Lui 2012, He et al. 2012, Tan and Massoulié 2013, Zhou et al. 2012] assume uma distribuição Zipf. Poucos [Liu et al. 2013] fornecem modelos independentes de distribuição ou usam dados de sistemas reais [Cha et al. 2008, Huang et al. 2008]. Não temos conhecimento de outros trabalhos além do nosso com uma comparação entre o desempenho de um sistema P2P de VoD com uma distribuição Zipf pura e uma distribuição Zipf com decaimento exponencial.

Outros trabalhos [Laoutaris et al. 2008, Whiteaker et al. 2012] sugerem o uso de STBs e *gateways* domésticos num contexto mais geral de aplicações, como VPNs, servidores *web* e jogos *online*.

3. Arquitetura do sistema

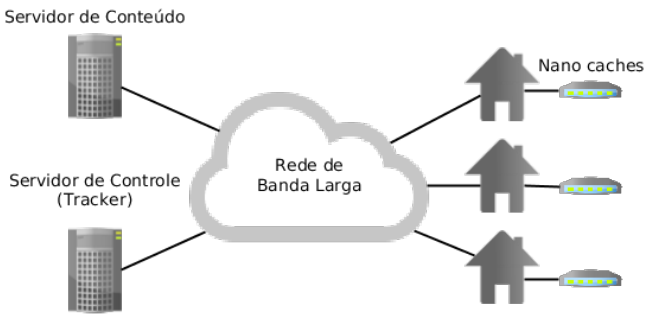

Figura 1. Arquitetura do sistema.

Nesta seção, descrevemos a arquitetura e o funcionamento de nosso sistema, assim como nossa política de troca de *swarms*. Apresentamos também detalhes de nosso protótipo embarcado desenvolvido para prova de conceito.

Propomos um sistema formado por um conjunto de nano caches, um servidor de controle e um servidor de conteúdo, como ilustrado pela Figura 1. Os nano caches são dispositivos embarcados (STBs ou roteadores domésticos) com capacidade de armazenamento local suficiente para um conjunto pequeno de vídeos. Assumimos que estes dispositivos encontram-se em uma mesma rede de banda larga sob controle de um provedor de serviço de Internet (ISP). O servidor de controle é responsável pela indexação dos conteúdos disponíveis, pela autenticação dos clientes e pela monitoração dos nano caches conectados (*tracker*). Já o servidor de conteúdo tem como função garantir a disponibilidade dos vídeos distribuídos no sistema, armazenando cópias de todos os conteúdos e, se necessário, participando da transmissão de vídeo. Mais servidores de conteúdo podem ser adicionados ao sistema para balanceamento de carga e redundância.

Quando um usuário assiste a um vídeo, seu nano cache faz o *download* a partir de outros nano caches (*peers*) e, se preciso, do servidor de contéudo[2]. Em vez de fazer o *download* a uma taxa igual à taxa de codificação do vídeo, o *peer* que recebe o conteúdo (*leecher*) tenta sempre saturar sua banda, minimizando o tempo de *download*. O conteúdo assistido é armazenado localmente e, se não há espaço disponível, uma política LRU (*Least Recently Used*) é adotada. A escolha de uma estratégia ótima para replicação de conteúdos está além do escopo deste trabalho. Já foi mostrado que a adoção de uma política de substituição LRU leva a uma replicação aproximadamente proporcional à popularidade dos conteúdos, fornecendo um desempenho próximo ao ótimo [Zhou et al. 2012].

Os nano caches utilizam a banda de *upload* dos clientes para servir conteúdos que já foram baixados. Como o canal é compartilhado com outros dispositivos da rede doméstica, o nano cache deve minimizar sua interferência com o tráfego local. Se ele atua como roteador doméstico, sua taxa de *upload* pode ser adaptada de acordo com o tráfego medido em suas interfaces de entrada. Caso não seja possível para o nano cache medir diretamente o tráfego local (por exemplo, se for um STB), ele pode utilizar um algoritmo de controle de congestionamento baseado em latência como o LEDBAT [Shalunov et al. 2012]. Ainda assim, esperamos que haja bastante capacidade disponível para o sistema, já que usuários de banda larga raramente saturam sua banda de *upload* e *download* [Grover et al. 2013]. Apresentamos na Tabela 1 a notação utilizada.

3.1. Troca de *swarms*

Dos C conteúdos disponíveis no sistema, um nano cache é capaz de armazenar localmente até B conteúdos, sendo $B \ll C$. Entretanto, devido às suas limitações de processamento e memória, um nano cache serve apenas m dentre os B vídeos armazenados. Precisamos portanto definir uma política para escolha desses m conteúdos que devem ser servidos de modo a fazer uma boa alocação dos recursos disponíveis.

A cada vídeo distribuído no sistema corresponde um *swarm*, do qual participam os *peers* que assistem ao vídeo (*leechers*) e os que o servem (*seeds*). A escolha dos m *swarms* que um *peer* deve servir leva em consideração uma métrica de carga, definida para o *swarm* i como

$$\ell_i = \frac{L_i}{S_i + 1}, \tag{1}$$

[2]A partir deste ponto, adotaremos a terminologia usada em sistemas P2P. Os termos nano cache e *peer* serão utilizados indistintamente.

Tabela 1. Descrição dos principais parâmetros do sistema

Parâmetro	Definição
N	Número de nano caches (*peers*) no sistema
C	Número de conteúdos disponíveis no sistema
B	Número de conteúdos que cabem em um cache
m	Número de conteúdos que um nano cache serve
λ	Taxa de chegada de *leechers*
τ	Intervalo de troca de *swarms*
D_{nc}	Banda disponível para *download* nos nano caches
U_{nc}	Banda disponível para *upload* nos nano caches
U_s	Banda disponível para *upload* no servidor de conteúdo
L_i	Número de *leechers* no *swarm* i
S_i	Número de *seeds* no *swarm* i
p_i	Popularidade relativa do conteúdo i ($\sum_{i=1}^{C} p_i = 1$)
d_i	Duração do vídeo i
b_i	Taxa de codificação (*bitrate*) do vídeo i

onde L_i e S_i são, respectivamente, o número de *leechers* e *seeds* no *swarm* i. Um valor alto de carga pode indicar um número grande de clientes assistindo ao vídeo e/ou um número pequeno de nano caches servindo-o. A carga de um *swarm* pode ser facilmente calculada por um nano cache a partir dos dados recebidos do servidor de controle, que atua como *tracker*. Mesmo sem a presença de um servidor *tracker*, essa métrica poderia ser estimada de maneira descentralizada utilizando, por exemplo, uma Tabela hash distribuída (DHT).

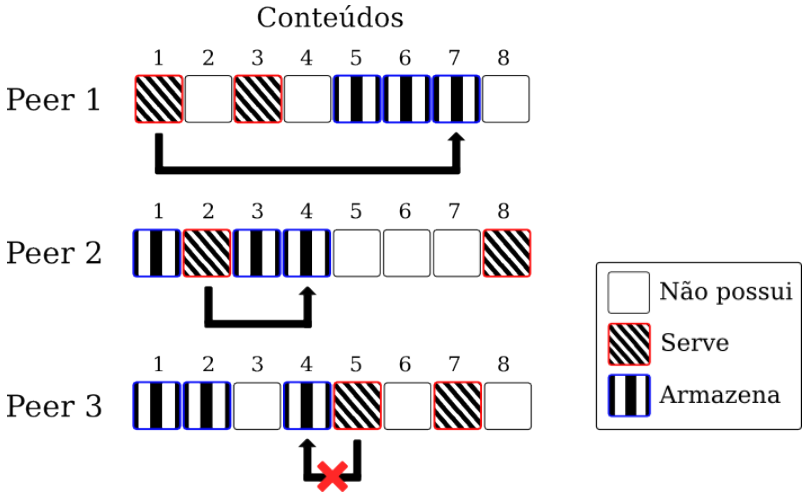

Figura 2. Ilustração da política de troca de *swarms*. Os conteúdos estão ordenados em ordem crescente de carga ℓ_i. Um *peer* deve trocar o *swarm* de menor carga dentre aqueles que serve pelo de maior carga dentre os que armazena (*Peer* 1 troca *swarm* 1 pelo 7 e *Peer* 2 troca *swarm* 2 pelo 4). Caso a carga do primeiro seja maior ou igual à do segundo, a troca não ocorre (*Peer* 3).

A qualidade de experiência do usuário que assiste a um vídeo está relacionada à carga do *swarm* no qual ele é distribuído. Por isso, propomos uma política simples que reduz as diferenças de carga entre os *swarms* introduzindo um *overhead* muito pequeno. Nossa proposta consiste na troca de dois *swarms* em intervalos de tempo τ. No momento da troca, o *peer* busca dentre os m conteúdos que serve aquele que possui o *swarm* de menor carga (*swarm* i). Depois, dentre os $(B - m)$ conteúdos que armazena sem servir,

escolhe aquele com *swarm* de maior carga (*swarm* j). Se $\ell_j > \ell_i$, o *peer* deixa de ser *seed* em i e passa a ser *seed* em j. Caso contrário, ele continua a servir os mesmos m conteúdos. O processo é ilustrado pela Figura 2.

3.2. Protótipo

Desenvolvemos um protótipo de nano cache usando a *libbtstream* [Mendonça and Leão 2012], biblioteca que permite a distribuição de vídeo usando como base o protocolo BitTorrent. O *software* foi adaptado e compilado para o OpenWrt [3], uma distribuição Linux para dispositivos embarcados.

O cliente BitTorrent foi conFigurado para fazer o *download* de pedaços em ordem (sem utilizar a política *Rarest-first* padrão) e fazer *upload* apenas de *torrents* finalizados (como *seed*).

A *libbtstream* faz uso do protocolo uTP [Norberg 2009] de controle de congestionamento no nível da aplicação, que implementa o LEDBAT e minimiza a interferência do tráfego BitTorrent no tráfego doméstico. Isso permite que nosso protótipo de nano cache seja utilizado como roteador *wireless* sem impactar a experiência dos usuários.

Em nossa prova de conceito, usamos como plataforma o roteador doméstico TL-WDR3600 da TP-Link. Esse roteador possui um *System-on-Chip* (SoC) AR9344 da Qualcomm Atheros com processador MIPS operando a 560 MHz, 128 MB de memória RAM, 8 MB de memória flash para armazenamento interno e 2 portas USB 2.0. Para armazenamento dos vídeos, usamos um HD externo com capacidade de 500 GB. Atualmente, possuímos uma rede com nano caches distribuídos entre 15 voluntários de um mesmo provedor de banda larga. Resultados preliminares de experimentos encontram-se em [Mendonça 2015].

4. Modelo

4.1. Limite de capacidade do sistema

Nós propomos um modelo simples que nos permite estimar a taxa máxima de chegada de *leechers* suportada pelo sistema. Com esse modelo, podemos dimensionar os recursos do sistema, como a banda de *upload* dos nano caches e do servidor de conteúdo. Dada uma taxa de chegada, o modelo também nos possibilita o cálculo da taxa de codificação limite para os vídeos, parâmetro de grande impacto na qualidade percebida pelos usuários.

Primeiro, calculamos o número máximo de *leechers* que o sistema pode atender de modo eficiente. Assumiremos que todos os vídeos possuem mesma duração d e taxa de codificação (*bitrate*) b. Para que os clientes sejam capazes de baixar o vídeo a uma taxa maior ou igual a b, queremos que, no pior caso, o valor esperado do número de *leechers* no sistema seja

$$E[L] = \frac{N \cdot U_{nc} + U_s}{b}, \tag{2}$$

onde $N \cdot U_{nc} + U_s$ é a soma da capacidade de *upload* dos nano caches e do servidor de conteúdo. Simultaneamente, queremos que o valor esperado do tempo de *download*, T_D,

[3] https://openwrt.org/

seja no pior caso igual a d. Segundo a Lei de Little,

$$E[L] = \lambda \cdot E[T_D] \iff \lambda = \frac{E[L]}{E[T_D]}. \tag{3}$$

Aplicando (2) em (3) e substituindo $E[T_D]$ pela duração do vídeo, temos nosso limite para a taxa de chegada:

$$\lambda_{max} = \frac{N \cdot U_{nc} + U_s}{b \cdot d}. \tag{4}$$

4.2. Troca de *swarms*

Segundo nossa política de troca de *swarms*, um *peer* deve sair de um dos *swarms* em que é *seed* para entrar em outro *swarm* de carga mais alta. A princípio, para modelar esse processo seria necessário conhecer todos os vídeos que cada nano cache armazena e serve. Entretanto, propomos uma aproximação que nos permite reduzir drasticamente o espaço de estados de um modelo utilizando apenas dois vetores de variáveis de estado de tamanho C, que indicam o número de *seeds* (S_i) e *leechers* (L_i) em cada *swarm*.

Primeiro, modelaremos o processo de saída de um *peer* de um *swarm* de carga mais baixa. Assumimos sem perda de generalidade que os C *swarms* estão ordenados de acordo com sua carga, de modo que $\ell_i \leq \ell_j$ (ver equação (1)) para todo $i < j$, $1 \leq i, j \leq C$. Definimos então a variável aleatória R, que indica o *swarm* de menor carga que um nano cache escolhido ao acaso está servindo. $\Pr(R = k)$ é, portanto, a probabilidade de um *peer* aleatório optar por sair do *swarm* k. Essa probabilidade equivale à fração de nano caches que escolheriam esse *swarm*.

Para que k seja escolhido, é preciso que um *peer* seja *seed* em k e não seja *seed* nos *swarms* anteriores a k (com carga menor ou igual a ℓ_k). Logo, se $\Pr(\omega_i)$ é a probabilidade de um *peer* estar servindo o conteúdo i e $\Pr(\bar{\omega}_i) = 1 - \Pr(\omega_i)$,

$$\Pr(R = k) = \Pr(\omega_k \cap \bar{\omega}_{k-1} \cap \bar{\omega}_{k-2} \cap \cdots \cap \bar{\omega}_1). \tag{5}$$

Propomos uma aproximação para a equação (5) com base na hipótese de que os eventos são mutuamente independentes:

$$\Pr(\hat{R} = k) = c_1 \cdot \Pr(\omega_k) \cdot \prod_{i=1}^{k-1} \Pr(\bar{\omega}_i), \tag{6}$$

onde $c_1 = \left\{ \sum_{j=1}^{C} \Pr(\omega_j) \cdot \prod_{i=1}^{j-1} \Pr(\bar{\omega}_i) \right\}^{-1}$ é uma constante de normalização.

A probabilidade de um nano cache ser *seed* em k é dada por

$$\Pr(\omega_k) = \frac{S_k}{N}, \tag{7}$$

já que S_k dentre os N *peers* do sistema estão servindo o vídeo k.

Aplicando (7) em (6), temos

$$\Pr(\hat{R} = k) = c_1 \cdot \frac{S_k}{N} \cdot \prod_{i=1}^{k-1} \left[1 - \frac{S_i}{N} \right]. \tag{8}$$

Agora iremos modelar o processo de entrada de um *peer* em um *swarm*. Definimos a variável aleatória A, que indica o *swarm* no qual um *peer* aleatório decide entrar. Deste modo, $\Pr(A = k)$ é igual à probabilidade de um nano cache escolhido ao acaso optar por servir o conteúdo k no momento da troca. Vamos assumir que um *peer* escolhe k se e somente se possui k armazenado e não possui os conteúdos posteriores a k (de carga maior ou igual a ℓ_k) [4]. Se $\Pr(\sigma_i)$ é a probabilidade de um nano cache ter o conteúdo i e $\Pr(\bar{\sigma}_i) = 1 - \Pr(\sigma_i)$, temos que

$$\Pr(A = k) = \Pr(\sigma_k \cap \bar{\sigma}_{k+1} \cap \bar{\sigma}_{k+2} \cap \cdots \cap \bar{\sigma}_C) \tag{9}$$

Assumimos novamente que os eventos são mutuamente independentes. Deste modo, temos a aproximação

$$\Pr(\hat{A} = k) = c_2 \cdot \Pr(\sigma_k) \cdot \prod_{i=k+1}^{C} \Pr(\bar{\sigma}_i) \tag{10}$$

onde $c_2 = \left\{ \sum_{j=1}^{C} \Pr(\sigma_j) \cdot \prod_{i=j+1}^{C} \Pr(\bar{\sigma}_i) \right\}^{-1}$ é uma constante de normalização.

Aproximamos a probabilidade de um nano cache possuir o conteúdo k por uma distribuição binomial. Para isso, assumimos que os B conteúdos em um nano cache são amostrados de maneira independente (com reposição) e que o conteúdo k é escolhido com probabilidade p_k:

$$\Pr(\sigma_k) \approx 1 - (1 - p_k)^B. \tag{11}$$

Aplicando (11) em (10), temos:

$$\Pr(\hat{A} = k) = c_2 \cdot \left[1 - (1 - p_k)^B\right] \cdot \prod_{i=k+1}^{C} (1 - p_i)^B. \tag{12}$$

Tendo escolhido os *swarms* r e a para sair e entrar, respectivamente, o *peer* efetua a troca de *swarms* se e somente se $\ell_d < \ell_a$. Caso contrário, ele permanece como *seed* nos mesmos *swarms* por pelo menos mais τ unidades de tempo.

As aproximações $\hat{R}$ e $\hat{A}$ (equações (8) e (12)) foram avaliadas através do método Monte Carlo. Obtivemos um *Root Mean Squared Error* (RMSE) normalizado de 1,38% $\pm$ 0,02 p.p. para $\hat{R}$ e 2,77% $\pm$ 0,05 p.p. para $\hat{A}$.

Com essas duas variáveis aleatórias, podemos modelar a decisão tomada por um *peer* sem conhecer o seu estado, ou seja, os m *swarms* que ele serve e os B conteúdos que possui. Se representamos o estado de cada *peer*, o espaço de estados cresce exponencialmente com o número de nano caches. Com nossa aproximação, o crescimento é polinomial [Mendonça 2015]. Portanto, a aproximação é essencial para resolver modelos com números de *leechers* e *seeds* iguais aos de sistemas reais.

[4] Não consideramos aqui os conteúdos que o *peer* está servindo no momento. Se, por exemplo, já estivesse servindo k, este não poderia ser escolhido.

5. Avaliação de desempenho

Nesta seção, apresentamos uma análise do desempenho de nosso sistema com o objetivo de: (i) validar o modelo para o limite de capacidade do sistema; (ii) observar o impacto da política de troca de *swarms*; (iii) avaliar o funcionamento com e sem a participação do servidor de conteúdo; e (iv) medir a influência de diferentes distribuições de popularidade.

Para tal, desenvolvemos um modelo usando a ferramenta Tangram-II [de Souza e Silva et al. 2009], que possui como variáveis de estado o número de *leechers* e *seeds* em cada *swarm* e usa nossas aproximações $\hat{R}$ e $\hat{A}$ para representar o processo de troca de *swarms*. Assumimos que a taxa de chegada de *leechers* no sistema segue um processo de Poisson homogêneo e que a probabilidade de um novo *leecher* escolher o conteúdo k é igual a p_k. Além disso, assumimos que todos os *peers* saturam sua banda de *upload*, que é dividida igualmente entre os m conteúdos que servem, e que todos os vídeos possuem a mesma duração (5400 s) e taxa de codificação (5 Mbps). Esta análise pode ser facilmente estendida para o caso mais geral em que há diferença de duração e *bitrate* entre os conteúdos. E, a fim de simplificar nossa análise, modelamos o tempo entre trocas de *swarms* e o tempo de *download* dos conteúdos como variáveis aleatórias exponenciais. Essa é uma das principais fontes de aproximação do modelo. Os valores adotados por padrão para os parâmetros do modelo podem ser vistos na Tabela 2.

A distribuição de popularidade dos conteúdos foi extraída de dados de um sistema de VoD de um grande provedor brasileiro de banda larga. Obtivemos acesso a *logs* que cobrem 38299 usuários durante um período de 3 meses. Testes de razão de verossimilhança [Vuong 1989] indicaram, com p-valor próximo a zero, que uma distribuição Zipf com decaimento exponencial é mais adequada à distribuição empírica do que uma Zipf pura, que segue uma lei de potência e é usada com muita frequência para modelar popularidade de conteúdos na Internet (ver seção 2). O decaimento exponencial da cauda da distribuição de popularidade foi observado em outros sistemas reais, como o Netflix, o Youtube e o chinês PowerInfo [Cheng et al. 2007, Cha et al. 2007]. A PMF de uma distribuição Zipf com decaimento exponencial é dada por $\Pr(Z = k) = c \cdot k^{-\alpha} \cdot e^{-\mu \cdot k}$, onde c é uma constante de normalização. Utilizando MLE, estimamos os parâmetros $\hat{\alpha} = 0,63$ e $\hat{\mu} = 0,02$.

Em nossa análise, usamos as seguintes métricas de desempenho: $\bar{D}$, a taxa média de *download* dos *peers*; $\bar{T}_D$, o tempo médio de *download*; e $\bar{G}$, a fração média de tempo com taxa de *download* maior do que a taxa de codificação. Todos os resultados são apresentados com um intervalo de confiança de 95% para 20 rodadas de simulação.

Tabela 2. Parâmetros padrão adotados nas simulações

Parâmetro	Valor	Parâmetro	Valor
N	40000	D_{nc}	10 Mbps
C	250	U_{nc}	1 Mbps
B	20 ($\approx$ 66 GB de armazenamento)	U_s	1 Gbps
m	4	p_i	$\propto i^{-0,63} \cdot e^{-0,02 \cdot i}$
λ	4800 *leechers* / h	d_i	5400 s (90 min) para todo i
τ	1800 s (30 min)	b_i	5 Mbps para todo i

5.1. Taxa de chegada

A fim de validar nosso modelo para o limite de capacidade e avaliar a escalabilidade do sistema, realizamos simulações mantendo a taxa de *upload* do servidor de conteúdo igual

a 1 Gbps e variando o valor de λ. Nesse cenário, temos um limite para a taxa de chegada de aproximadamente 5467 *leechers* por hora. Os resultados são apresentados na Tabela 3.

Vemos que o sistema apresenta um excelente desempenho para valores abaixo do limite superior de escalabilidade. Mesmo para uma taxa de chegada próxima ao limite, de 5200 *leechers* por hora, observamos, por exemplo, que em média 99% dos usuários experimentam tempo de *download* inferior a 5400 segundos. Esses resultados sugerem uma boa qualidade de experiência para os usuários, com pouco ou nenhum tempo de *buffering*.

Para taxas de chegada acima do limite, notamos que a qualidade se degrada rapidamente, já que os nano caches e o servidor de conteúdo não possuem banda suficiente para atender à demanda. Isso se reflete na taxa média de *upload* menor do que 5 Mbps, no alto tempo de *download* e na probabilidade próxima a zero de um *leecher* possuir uma taxa de *download* maior do que a taxa de codificação por mais de 95% do tempo.

Tabela 3. Avaliação do valor máximo de λ. Limite superior: 5467 *leechers* / h

λ (*leechers* / h)	$\bar{D}$ (Mbps)	$\Pr(\bar{T}_D \leq 5400s)$	$\Pr(\bar{G} > 95\%)$
4800	$9,91 \pm 0,04$	$0,9988 \pm 2,78 \cdot 10^{-4}$	$0,97 \pm 6,83 \cdot 10^{-4}$
5200	$8,64 \pm 0,26$	$0,9949 \pm 1,14 \cdot 10^{-3}$	$0,98 \pm 6,79 \cdot 10^{-4}$
5600	$3,73 \pm 0,15$	$0,0022 \pm 1,31 \cdot 10^{-4}$	$0,02 \pm 0,04$
6000	$2,60 \pm 0,05$	$0,0013 \pm 1,11 \cdot 10^{-4}$	$0,00 \pm 0,00$

5.2. Troca de *swarms*

Mostramos agora os resultados de nossa avaliação da política de troca de *swarms*. Comparamos nossas métricas de desempenho para diferentes valores de τ. No cenário em que não há uma política de troca de *swarms*, quando um *leecher* termina o *download* de um conteúdo ele passa a serví-lo imediatamente e deixa de servir um dos m conteúdos que está servindo. Nesse caso, o *swarm* a ser deixado é escolhido aleatoriamente de acordo com a probabilidade do *peer* ser *seed* naquele *swarm* (ver equação (7)).

Os resultados apresentados na Tabela 4 sugerem uma piora de todas as métricas de desempenho quando não há troca de *swarms*. Observamos também uma pequena degradação da qualidade com o aumento do valor de τ (tempo entre trocas).

Na Figura 3, vemos um gráfico com o tempo médio de *download* ($\bar{T}_D$) por *swarm*. Notamos uma piora no desempenho quando não há troca de *swarms*. Sem nossa política, os *swarms* de baixa e média popularidade apresentam tempo médio de *download* até 3 vezes maior do que a duração do vídeo. Entretanto, para os 40 conteúdos mais populares, não vemos uma diferença significativa. Vemos também que o valor de $\bar{T}_D$ cresce juntamente com τ nos *swarms* de menor popularidade.

Tabela 4. Avaliação do intervalo entre trocas de *swarms*.

Política	$\bar{D}(Mbps)$	$\Pr(\bar{T}_D \leq 5400s)$	$\Pr(\bar{G} > 95\%)$
Sem troca	$8,31 \pm 0,09$	$0,9372 \pm 5,46 \cdot 10^{-3}$	$0,83 \pm 0,02$
30 min	$9,91 \pm 0,04$	$0,9988 \pm 2,78 \cdot 10^{-4}$	$0,97 \pm 6,83 \cdot 10^{-4}$
120 min	$9,79 \pm 0,04$	$0,9976 \pm 3,43 \cdot 10^{-4}$	$0,97 \pm 6,01 \cdot 10^{-4}$
480 min	$9,45 \pm 0,05$	$0,9933 \pm 5,04 \cdot 10^{-4}$	$0,97 \pm 5,06 \cdot 10^{-3}$

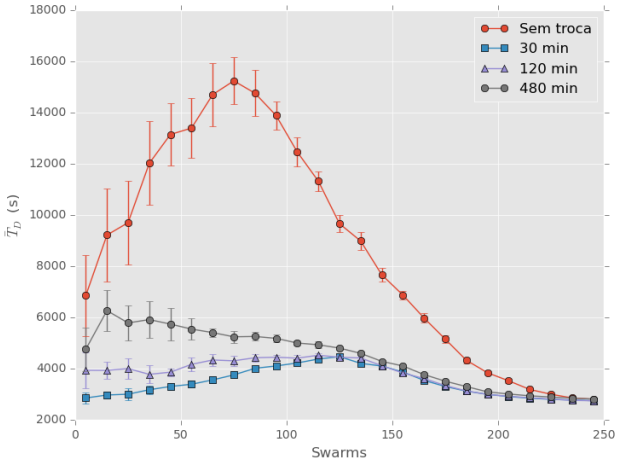

Figura 3. Comparação do tempo médio de *download* por *swarm* para diferentes valores de τ. *Swarms* em ordem crescente de popularidade.

Concluímos que nossa política de troca de *swarms* causa grande impacto na qualidade percebida pelos usuários. Acreditamos que o valor do parâmetro τ pode ser ainda mais relevante em cenários com variação súbita de popularidade (*flash crowd*).

5.3. Servidor de conteúdo

Nos voltamos agora para a análise do impacto da participação do servidor de conteúdo na transmissão dos arquivos de vídeo. Para isso, executamos simulações com U_s igual a 0 Gbps. Na comparação com os resultados para U_s igual a 1 Gbps, vemos uma diferença pouco significativa na taxa média de *download* dos *leechers*: $9,83 \pm 0,08$ Mbps sem servidor e $9,91 \pm 0,04$ Mbps com servidor.

A comparação do tempo médio de *download* por *swarm* também indica pouco impacto do servidor de conteúdo nesse cenário. Como pode ser visto na Figura 4, o cenário sem servidor apresentou valores de $\bar{T}_D$ maiores para os conteúdos menos populares. Os vídeos para os quais observamos um tempo médio de *download* maior representam cerca de 2% das visualizações do sistema. Ainda assim, o tempo médio de *download* manteve-se menor do que a duração dos vídeos em todos os *swarms*.

5.4. Distribuição de popularidade

Por fim, analisamos o desempenho de nosso sistema em função da distribuição de popularidade dos conteúdos. Comparamos a distribuição Zipf com decaimento exponencial com uma distribuição Zipf pura com parâmetro $\alpha = 0,8$.

Na Figura 5, vemos o tempo médio de *download* nos *swarms* medido para as duas distribuições. Vemos uma grande diferença no formato das curvas. O sistema com uma distribuição Zipf pura apresenta pior desempenho para os *swarms* menos populares. Já no caso da distribuição com decaimento exponencial, os usuários tendem a experimentar um tempo de *download* maior ao assistirem aos conteúdos de popularidade média. Lembramos que, com uma distribuição Zipf com truncamento exponencial, os 100 vídeos menos populares somam cerca de 1% dos acessos ao sistema. Com a distribuição Zipf, os 100 conteúdos de menor popularidade recebem aproximadamente 14% das visualizações.

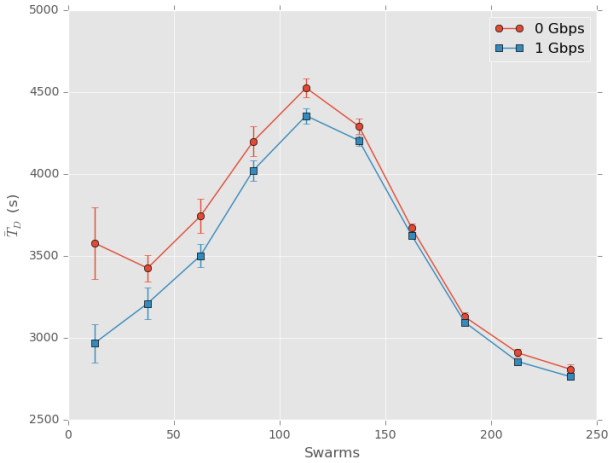

Figura 4. Comparação do tempo médio de *download* por *swarm* com e sem a participação do servidor de conteúdo. *Swarms* em ordem crescente de popularidade.

Concluímos que a aproximação da distribuição real de popularidade do sistema por uma distribuição Zipf pode não ser boa, já que há uma variação significativa no desempenho quando comparada a uma distribuição com decaimento exponencial, mais próxima da realidade.

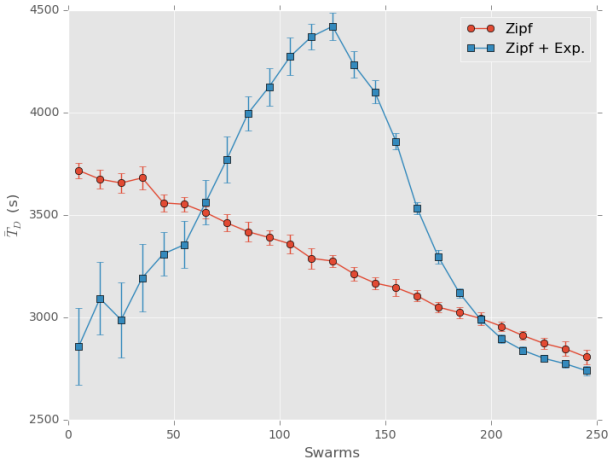

Figura 5. Comparação do tempo médio de *download* por *swarm* para as duas distribuições de popularidade. *Swarms* em ordem crescente de popularidade.

6. Conclusão

Neste trabalho, apresentamos uma plataforma P2P para transmissão de vídeo usando dispositivos embarcados projetada e avaliada com base em circunstâncias reais. Definimos uma política para escolha dos conteúdos a serem servidos por um nano cache que permite o balanceamento da carga do sistema de forma distribuída sem a adição de *overheads*, usando uma métrica simples de carga de um *swarm*.

Através de simulações, avaliamos o desempenho do sistema em diversos cenários e pudemos constatar o impacto de nosso algoritmo de troca de *swarms* na distribuição dos conteúdos menos populares. Ressaltamos que, embora tenhamos assumido nano caches com mesma banda de *upload* e *download* e vídeos de mesma duração e *bitrate*, nosso estudo pode ser facilmente estendido para o caso mais geral.

Nosso modelo probabilístico para a política de troca de *swarms* permite uma redução expressiva no espaço de estados do sistema que, a princípio, deveria crescer exponencialmente com o número de nano caches. Desse modo, torna-se viável a análise do desempenho do sistema usando parâmetros próximos aos reais. No futuro, pretendemos usar esse modelo para a dedução de novos resultados analíticos.

Acreditamos ser este o primeiro trabalho a apresentar uma análise do desempenho de uma plataforma P2P de VoD em que a popularidade de seus conteúdos segue uma distribuição Zipf com decaimento exponencial, com parâmetros extraídos de um sistema real. Concluímos que ainda há muito a ser investigado em cenários como esse. Em especial, observamos que, devido à brusca queda na cauda da distribuição de popularidade, novas medidas podem ser necessárias para atender adequadamente aos usuários interessados em conteúdos de nicho.

Esperamos no futuro poder utilizar nosso protótipo embarcado de nano cache para montar um *testbed* e realizar experimentos com o objetivo de confirmar os resultados de nossas simulações e validar nossas premissas.

Referências

Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2007). I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proc. of ACM IMC*, pages 1–14. ACM.

Cha, M., Rodriguez, P., Moon, S. B., and Crowcroft, J. (2008). On next-generation telco-managed P2P TV architectures. In *Proc. of IPTPS*, page 5.

Chen, Z. and Yan, J. (2009). Optimizing resource scheduling in BitTorrent file distribution network. In *Proc. of MASS'09*, pages 1–4. IEEE.

Cheng, X., Dale, C., and Liu, J. (2007). Understanding the Characteristics of Internet Short Video Sharing: YouTube as a Case Study. *CoRR*, abs/0707.3670.

de Souza e Silva, E., Figueiredo, D. R., and Leão, R. M. (2009). The TANGRAM-II integrated modeling environment for computer systems and networks. *ACM SIGMETRICS PER*, 36(4):64–69.

Grover, S., Park, M. S., Sundaresan, S., Burnett, S., Kim, H., Ravi, B., and Feamster, N. (2013). Peeking behind the NAT: an empirical study of home networks. In *Proc. of ACM IMC*, pages 377–390.

He, Y., Xiong, Z., Zhang, Y., Tan, X., and Li, Z. (2012). Modeling and analysis of multi-channel P2P VoD systems. *Journal of Network and Computer Applications*, 35(5):1568–1578.

Huang, Y., Fu, T. Z., Chiu, D.-M., Lui, J. C., and Huang, h. (2008). Challenges, Design and Analysis of a Large-scale P2P-vod System. *ACM SIGCOMM CCR*, 38(4):375–388.

Laoutaris, N., Rodriguez, P., and Massoulie, L. (2008). ECHOS: edge capacity hosting overlays of nano data centers. *ACM SIGCOMM CCR*, 38(1):51–54.

Liu, F., Li, B., Li, B., and Jin, H. (2013). Peer-assisted on-demand streaming: Characterizing demands and optimizing supplies. *IEEE TC*, 62(2):351–361.

Mendonça, G. and Leão, R. (2012). BTStream – Um ambiente para desenvolvimento e teste de aplicações de streaming P2P. In *Proc. of SBRC*.

Mendonça, G. (2015). Sistema de nano caches residenciais para distribuição de vídeo. Master's thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

Meng, X., Tsang, P.-S., and Lui, K.-S. (2013). Analysis of distribution time of multiple files in a P2P network. *Computer Networks*, 57(15):2900–2915.

Munoz Gea, J. P., Traverso, S., and Leonardi, E. (2012). Modeling and evaluation of multisource streaming strategies in P2P VoD systems. *IEEE T-CE*, 58(4):1202–1210.

Norberg, A. (2009). uTorrent Transport Protocol (BEP 29). `http://www.bittorrent.org/beps/bep_0029.html`. Acessado em: 29/11/2014.

Shalunov, S., Hazel, G., Iyengar, J., and Kuehlewind, M. (2012). Low Extra Delay Background Transport (LEDBAT). RFC 6817 (Experimental).

Suh, K., Diot, C., Kurose, J., Massoulie, L., Neumann, C., Towsley, D., and Varvello, M. (2007). Push-to-peer video-on-demand system: Design and evaluation. *IEEE J-SAC*, 25(9):1706–1716.

Tan, B. and Massoulié, L. (2013). Optimal content placement for peer-to-peer video-on-demand systems. *IEEE TON*, 21(2):566–579.

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, pages 307–333.

Wang, Z., Wu, C., Sun, L., and Yang, S. (2010). Strategies of collaboration in multi-channel P2P VoD streaming. In *Proc. of IEEE GLOBECOM*, pages 1–5. IEEE.

Whiteaker, J., Schneider, F., Teixeira, R., Diot, C., Soule, A., Picconi, F., and May, M. (2012). Expanding home services with advanced gateways. *ACM SIGCOMM CCR*, 42(5):37–43.

Wu, W. and Lui, J. (2012). Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. *IEEE TPDS*, 23(8):1492–1503.

Zhou, Y., Fu, T. Z., and Chiu, D. M. (2012). A unifying model and analysis of P2P VoD replication and scheduling. In *Proc. of IEEE INFOCOM*, pages 1530–1538. IEEE.