Evaluating the 2PC Algorithm for the Maintenance of P2P Live Streaming

Adriel C. dos Santos, Iago A. Carvalho, Cristiano M. Silva, and Eliseu C. Miguel

***Abstract*—Peer-to-peer (P2P) Networks for live streaming face challenges such as ensuring low latency and low discontinuity in media transmission among peers. Algorithms for constructing and maintaining the overlay are often proposed to address several of these challenges. However, it is common to find works that present positive results from the execution of these algorithms without showing the overlay structure constructed by them. In this article, we analyze the overlay constructed and maintained by the Peer Classification for Partnership Constraints (2PC) algorithm. 2PC proved to be efficient in dealing with a large number of free-riders on the network, imposing constraints on partnerships between peers according to their contributions to media transmission. To understand the 2PC execution effects supported by the K-Shortest Path Yen's algorithm, we evaluated the application of the 2PC and identified that the partnership relationships between peers imposed by the algorithm organize the overlay attracting high-contribution peers close to the server while pushing low-contribution peers to the edge of the overlay.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/8534

***Index Terms*—P2P, 2PC, live streaming, overlay construction, graphs, algorithms.**

I. Introdução

Redes *Peer-to-Peer* (P2P) são estruturas de distribuição de conteúdo baseadas na cooperação entre seus usuários, denominados *peers*. Estas redes são utilizadas para transmitir qualquer tipo de conteúdo. Neste trabalho, estamos interessados em redes P2P *live-streaming* (LS-P2P), redes utilizadas para transmissões de audio ou vídeo ao vivo. Nestas redes, os *peers* transmitem fragmentos de dados, denominados *chunks*, sendo estes distribuídos pelo servidor.

A qualidade de serviço de redes LS-P2P depende de diversos fatores: a entrada e saída constante de *peers*, conhecido como *churn* [1], [2]; a chegada repentina de um grande número de *peers* em um curto espaço de tempo, conhecido como *flash crowd* [3]; e a presença e quantidade de *peers* que se negam a retransmitir os *chunks*, conhecidos como *free-riders* [4]. A ocorrência destes fatores é muitas vezes inevitável durante o tempo de vida de uma LS-P2P [5].

O *overlay* é a rede sobreposta formada pelos *peers* sobre a rede física. Pode-se preservar a qualidade de transmissão em redes LS-P2P utilizando algoritmos de construção e gerenciamento de *overlay*. Dentre vários algoritmos, o *Peer Classification for Partnership Constraints* (2PC) [6] apresentou bom desempenho quanto à estabilidade da rede P2P nos experimentos realizados. Ele proporcionou baixa descontinuidade de mídia e latência entre os *peers* mesmo em condições extremamente desfavoráveis. Nos experimentos do 2PC, os autores incluíram o *flash crowd* com excessivo número de *free-riders* na rede. O 2PC oferece estabilidade impondo restrições de parcerias entre os *peers*. Essas restrições são atualizadas para cada *peer* periodicamente pelo algoritmo, que se baseia na contribuição individual de cada *peer* ao distribuir a mídia. Os autores do 2PC sugerem que o sucesso do algoritmo é alcançado por atrair os *peers* que apresentam alta contribuição de mídia para perto do servidor, enquanto os *peers* de baixa contribuição são empurrados para a periferia da rede. Contudo, essa sugestão é baseada nos resultados das métricas *descontinuidade* e *latência*, ao contrário de estudos da topologia da *overlay* durante as iterações do algoritmo.

A primeira avaliação dos efeitos do 2PC na topologia da *overlay* foi apresentada em [7]. Neste caso, para modelar os grafos da *overlay*, foi considerado apenas o caminho mínimo entre os *peers* e o servidor como métrica de distância. Desta forma, gera-se uma simplificação da *overlay*, permitindo posicionar os *peer* por camadas da rede em que cada camada é a distância até o servidor. Porém, como cada *peer* pode ter, pelo menos, vinte parceiros nas configurações do 2PC, considerar apenas o caminho mínimo é um fator limitante.

O objetivo deste trabalho é avaliar os efeitos da execução de 2PC sobre a *overlay* da rede P2P. Para tanto, este trabalho estende a primeira avaliação do 2PC [7] de forma mais realística. Para posicionar os *peers* nos modelos abstratos da topologia, consideramos agrupamentos de até quinze caminhos mínimos entre os *peers* como métrica de distância até o servidor. Com isso, geramos grafos mais densos para modelar a sobreposição e, como consequência, refinamos a compreensão da organização dos *peers* na *overlay*.

Podemos apontar três principais contribuições deste trabalho: i) Nós identificamos como o 2PC promove a movimentação dos *peers* na *overlay*, o que pode ser alvo de novos algoritmos para este fim; ii) identificamos a abordagem adotada para representar a *overlay* em grafos sem a definição formal para o posicionamento dos *peers* por camadas pode ser uma alternativa a outros trabalhos; e iii) apresentamos uma avaliação na taxa natural de *churn* dos *peers* no PlanetLab, nosso ambiente de experimentos.

O restante do artigo está organizado da seguinte forma: a Seção II apresenta o 2PC e os trabalhos relacionados a LS-P2P, enquanto a Seção III descreve a metodologia utilizada

Adriel C. dos Santos, Iago A. Carvalho, and Eliseu C. Miguel are with Universidade Federal de Alfenas, Alfenas, Brasil (e-mails: adriel.santos@sou.unifal-mg.edu.br, iago.carvalho@unifal-mg.edu.br and eliseu.miguel@unifal-mg.edu.br).

Cristiano M. Silva is with Universidade Federal de São João del Rei, Ouro Branco, Brasil (e-mail: cristiano@ufsj.edu.br).

neste trabalho para analisar os *logs* gerados na aplicação do algoritmo 2PC em [6]. A Seção IV apresenta os resultados obtidos neste trabalho e a Seção V apresenta as conclusões.

II. Background e Trabalhos Relacionados

A. Questões Gerais sobre Redes P2P para Vídeo ao Vivo

Várias abordagens podem ser adotadas para se construir a *overlay* de redes P2P [8], sendo as baseadas em árvores [9] e em malha [10] as amplamente utilizadas. As últimas, tratadas neste trabalho, permitem alta robustez à rede [11]. Diferente dos sistemas cliente/servidor, as redes P2P baseam-se na cooperação entre os *peers* para se distribuir o conteúdo. Com essa cooperação, as redes tornam-se escaláveis [12].

Existem duas principais estratégias para lidar com o problema de falta de cooperação dos *peers*: i); uso de mecanismos de incentivo à cooperação; ii) técnicas que visam a identificação e punição de *peers* não-cooperativos [13]. Pode-se observar que mecanismos de incentivo à cooperação foram propostos em [14] e [15]. O primeiro apresenta o uso de anúncios como método de incentivo, enquanto o segundo propõe um mecanismo capaz de fornecer ao *peer* uma qualidade de vídeo proporcional à sua contribuição à rede.

Há também o *Tit-for-Tat*, um mecanismo presente no *software* de compartilhamento de arquivos *BitTorrent* [16]. Alguns trabalhos usam o *Tit-for-Tat* em redes LS-P2P, onde os *peers* pouco cooperativos são penalizados com uma qualidade de serviço inferior [17]. Em [13], os autores citam as desvantagens de mecanismos de incentivo, como o aumento do *overhead* na rede e da complexidade do sistema, o que impõe restrições que limitam o desempenho das redes. Essas desvantagens tornam-se ainda mais significativas em redes LS-P2P, que demandam baixa latência e baixa descontinuidade de mídia.

O comportamento *freeriding* ocorre naturalmente em redes P2P e é irrealístico demandar cooperação de todos os *peers* conectados a uma rede P2P de forma homogênea [13]. No mesmo trabalho, os autores mostram que, sob certas configurações, uma rede P2P ao vivo pode tolerar até 50% de *free-riders conscientes*, i.e., aqueles *peers* que notificam seus parceiros que não contribuirão com a rede P2P [13], sem sofrer uma degradação significativa na qualidade da transmissão. Em outras abordagens, foram propostos outros mecanismos que toleram *peers* não-cooperativos e que visam aumentar a eficiência de distribuição de mídia por meio da construção e gerenciamento do *overlay*. Em grande maioria, tais algoritmos tentam empurrar os *peers* não-cooperativos para a periferia da rede. Como exemplo podemos citar o *framework* de gerenciamento autônomo proposto em [12], que possui um processo de auto-organização que continuamente empurra os *peers* não-cooperativos em direção às folhas da árvore de distribuição.

Outro algoritmo para construção e gerenciamento da *overlay* é o 2PC, que diferentemente dos algoritmos citados, não visa identificar e punir os *free-riders*. Ao contrário, 2PC propõe que os *free-riders* declarem que não desejam contribuir na transmissão de mídia e, assim, organiza a *overlay* para atender ao máximo de usuários de acordo com os recursos de transmissão oferecidos pelos *peers* cooperativos.

B. Síntese do Algoritmo 2PC

O algoritmo 2PC realiza a construção e manutenção da *overlay* definindo classes de *peers* e os classificando em tais classes dinamicamente de acordo com a contribuição por distribuição de mídia de cada *peer* na rede. Cada uma dessas classes estabelece restrições de parceria entre seus *peers* e os *peers* de outras classes. As restrições têm o objetivo de organizar os *peers* na *overlay* de forma a aumentar a eficiência do uso da banda de *upload* individual de cada *peer*.

Nos experimentos realizados em [6] foram propostas quatro classes: i)*hot*; ii) *warm*; iii) *cold*; e iv) *free-rider*. A classe *hot* é reservada para os *peers* que mais contribuem; a classe *warm* acomoda *peers* que contribuem, porém não o suficiente para ser classificados como *hot*; e a classe *cold* contém os *peers* de baixa contribuição. Finalmente, a classe *free-rider* é reservada apenas para *peers* que declaram que não contribuirão com a transmissão de mídia, os *free-riders*.

Uma implementação comum em redes P2P faz com que cada *peer* p possua um conjunto Z de *peers* vizinhos, referente aos *peers* que p conhece na rede, e um conjunto N de *peers* parceiros (p só recebe e envia *chunks* aos *peers* que estejam em N). N é dividido em dois conjuntos: o conjunto de parceiros que fornecem *chunks* a p, $N_{in}(p)$; e o conjunto de parceiros aos quais p envia *chunks*, $N_{out}(p)$. Para a aplicação do algoritmo 2PC, o $N_{out}(p)$ foi dividido em outros dois conjuntos: $N_{out}^{high}(p)$ reservado para os *peers* parceiros de p cuja classe seja considerada de alta contribuição (*hot* e *warm*), e $N_{out}^{low}(p)$, reservado para *peers* parceiros de p cuja classe seja considerada de baixa contribuição (*cold* e *free-rider*). Esta separação do $N_{out}(p)$ garante que *peers* de alta contribuição não disputem parcerias com *peers* de baixa contribuição, o que promove recursos para *peers* a diferentes classes.

As restrições de parcerias definidas pelo algoritmo 2PC entre os *peers* de diferentes classes são aplicadas quando um *peer* q deseja que um *peer* p o forneça *chunks* da mídia distribuída. Caso p aceite o pedido de q, estabelece-se uma parceria em que q é um parceiro que pode receber *chunks* de p, ou seja $q \in N_{out}(p)$. Como definição, o 2PC determina que: *peers* da classe *hot* somente aceitam em seu conjunto N_{out} *peers* das classes *hot* e *warm*; *peers* da classe *warm* aceitam em seu conjunto N_{out} *peers* de qualquer classe; *peers* da classe *cold* aceitam em seu conjunto N_{out} apenas *peers* da classe *cold* e *free-riders*. A Fig. 1 mostra os possíveis parceiros N_{out} de um *peer* para cada classe, considerando a separação do N_{out} em N_{out}^{high} e N_{out}^{low}.

Periodicamente o 2PC analisa a contribuição de cada *peer* e o reclassifica, promovendo-o ou rebaixando-o entre as classes. Assim, a reclassificação de um *peer* p pode levar à desconexão entre ele e alguns parceiros $q \in N_{out}(p)$, caso q esteja em desacordo com as restrições impostas pela nova classe de p. O algoritmo 2PC, na configuração estudada neste trabalho, possui as seguintes características: a) quando um *peer* requisita ingresso na rede, 2PC o classifica na classe *warm*, classe que aceita parceria com *peers* de todas as outras classes; b) os *peers* são reclassificados durante todo o tempo de execução da rede; c) *peers* classificados como *free-rider* ou *cold* não podem se conectar diretamente ao servidor de mídia; e d) o

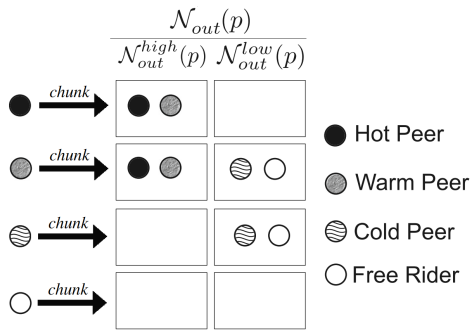


Fig. 1. Restrições de parcerias entre classes. À esquerda, vemos os *peers* de cada classe. Eles aceitam parceiros de acordo com os *peers* da tabela (ao centro da figura). A primeira coluna da tabela ($N_{out}^{high}(p)$) é reservada aos parceiros *peers* de alta contribuição, enquanto a segunda ($N_{out}^{low}(p)$) aos parceiros para *peers* de baixa contribuição. Essa distinção na lista de parceiros elimina a competição entre *peers* de alta e baixa contribuição.

2PC limita as populações das classes *hot* e *cold* em até 15% e 40% dos peers cooperativos, respectivamente.

1) Ambiente de Execução e Configurações: O algoritmo 2PC foi implementando no TVPP [18], um sistema *open-source* P2P de transmissão de vídeo ao vivo. Dois tipos de *logs* são gerados periodicamente pelas implementações no TVPP: *logs* de *overlay* e de *performance*. Os *logs* de *overlay*, base de informação para esse trabalho, reportam em cada uma de suas entradas um registro que identifica um *peer* e seus parceiros. Este *log* é composto por intervalos $T(n)$ de aproximadamente dez segundos. Em um intervalo $T(n)$, cada *peer* ativo na rede gera uma entrada no *log*. Para nossas análises, definimos $T(n)$ como um intervalo de tempo iniciando pelas entradas do *peer* servidor de mídia S. Em outras palavras, cada $T(n)$ é composto por todas as ocorrências de entradas de *peers* da rede em espaços de tempo com início em uma aparição da entrada E_n do servidor S até a entrada $E_{(n+1)}$ de S. Com os *logs* de *performance*, podemos medir a qualidade da transmissão da mídia, como a *latência* e a *descontinuidade* entre os *peers*.

Os experimentos foram realizados no *PlanetLab* com cerca de 1000 instâncias em execução. Uma rede com aproximadamente 110 *peers* é construída no instante 0s e, no tempo 400s, os *peers* que aguardam para ingressar na rede já construída chegam concomitantemente formando o *flash crowd*. Seguindo a metodologia de [6], foram definidos $N_{in}(s) = 0$ e $N_{out}(s) = 20$ para os *peers* do servidor e $N_{in}(p) = 20$ para os *peers* de todas as outras classes. Além disso, a distribuição de banda de *upload* dos *peers* presentes nos experimentos, a proporção de *peers* por banda e as configurações para $N_{out}(p)$ dos *peers* são apresentados na Tabela I. Pode-se observer que 50% dos *peers* foram definidos como *free-riders* (última linha). Os demais *peers* receberam bandas entre 1.0Mbps e 3.5Mbps (distribuição normal). As colunas 2PC e *Classic* apresentam os N_{out} dos *peers* para experimentos com e sem 2PC.

2) Resultados Apresentados para o 2PC: A Fig. 2 apresenta os resultados para dois tipos de experimentos: um com a aplicação do algoritmo 2PC e outro sem o algoritmo (versão *Classic*). A Fig. 2a apresenta o número de *peers* conectados à *overlay*. Os pontos roxos (ao topo) indicam o total de *peers*

TABELA I
CONFIGURAÇÃO DE *peers*

Qualidade (Banda)	*Upload* (Mbps)	x/100 %	2PC $N_{out}^{high}(p)$	2PC $N_{out}^{low}(p)$	*Classic* $N_{out}(p)$
Alta	3.0 - 3.5	$\approx$ 17%	46	0	46
Média	2.0 - 2.5	$\approx$ 17%	18	22	40
Baixa	1.0 - 1.5	$\approx$ 17%	0	38	38
-	0.0	50%	0	0	0

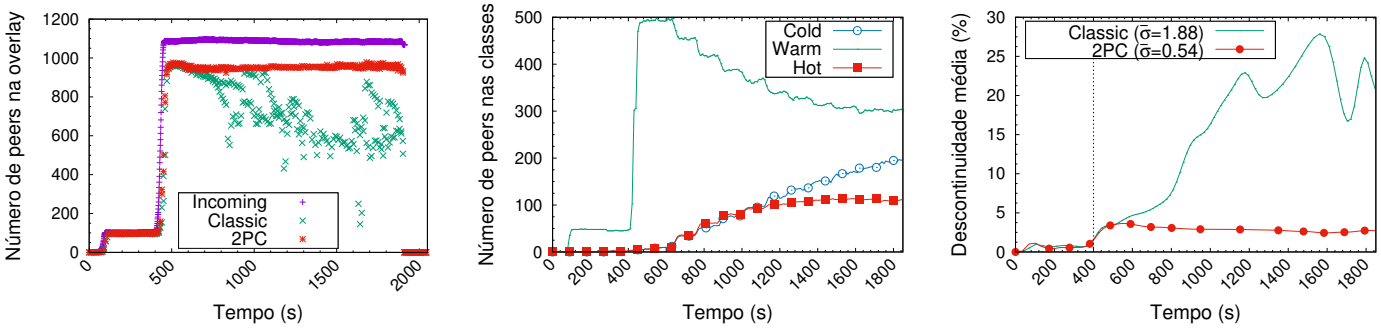
executados no experimento. É visível que no experimento *Classic* o número de *peers* conectados é instável ao longo do tempo (pontos verdes). Já o número *peers* conectados com o 2PC permanece estável ao longo do experimento (pontos vermelhos). A Fig. 2b apresenta a evolução na classificação dos *peers* por classe ao logo do experimento. Nela pode-se observar no instante 400s a ocorrência do *flash crowd* com todos os *peers* ingressando na classe *warm*.

A Fig. 2c descreve a média da descontinuidade de mídia, métrica que demonstra a qualidade da transmissão da mídia na rede. Observa-se que a métrica permanece baixa e estável para a rede com o 2PC, o que não acontece no caso *Classic*. Embora os bons resultados apresentados, não é possível observar a evolução na organização da *overlay*. Neste trabalho, por meio da análise de parcerias proveniente dos *logs* de *overlay*, estudamos como o 2PC organiza *overlay* da rede P2P.

III. METODOLOGIA

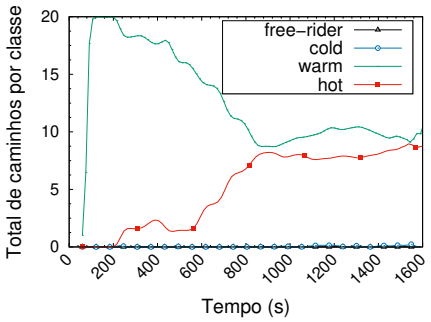
Reconstruímos graficamente a *overlay* dos experimentos observando as classes definidas pelo 2PC para cada *peer*. Para todo intervalo $T(n)$ no *log* de *overlay*, é possível gerar um grafo $G = (V, E)$ que representa a rede P2P, sendo V o conjunto de *peers* na rede cuja mídia é recebida com baixa ou nenhuma descontinuidade e E o conjunto de arestas, onde $\forall(p, q) \in E \iff q \in N_{out}(p)$. Devido ao potencial número de N_{out} de um *peer*, o grafo G é considerado um grafo denso, i.e., um grafo onde $|E| = \Theta(|V|^2)$ [19, p. 47], e há um número exponencial de caminhos em que um *chunk* pode ser transmitido de S até um *peer* p. Com base em *K-Shortest Path Yen's algorithm* [20] definimos os grupos $k_j |\ j \in \{1, 7, 15\}$ de caminhos mínimos partindo de S até p para avaliar a composição das camadas da *overlay*. Desta forma, não temos exatamente a distância de S a p. Ao contrário, consideramos uma camada da *overlay* Q_i como sendo composta por todos caminhos *i-distantes* de S aos *peers*. Com isso, devemos observar que p pode pertencer a mais de uma camada Q_i, uma vez que não há restrições que impeçam caminhos mínimos de tamanhos distintos entre S e p para k_j quando $j > 1$.

Os valores de j foram definidos em experimentos anteriores com os algoritmos 2PC e Yen. Em [6], empregamos $j = 1$, e este trabalho amplia a análise de [6] avaliando diferentes valores de j. Escolhemos esses valores porque eles poderiam demonstrar efetivamente as mudanças na sobreposição em relação ao aumento do número de caminhos mais curtos. Além disso, restringimos j a apenas três valores porque um grande conjunto de valores poluirá visualmente o artigo e não alterará as conclusões.

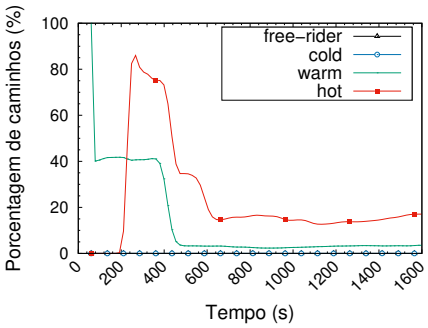


(a) *Overlay* (b) Tamanho das classes (c) Descontinuidade média da mídia

Fig. 2. Redes P2P sem e com a aplicação do algoritmo 2PC (*Classic* e 2PC). A Fig. 2a mostra o ingresso nas redes, ou seja, a quantidade de *peers* que foram executados (linha roxa no topo) e a quantidade de *peers* que participam da transmissão (linhas vermelha e verde). A Fig. 2b apresenta a quantidade de *peers* em cada classe ao longo do tempo de execução do 2PC. Observe que a classe *warm*, que recebe os *peers* ingressantes, perde *peers* para as outras classes ao longo do experimento pelo efeito da reclassificação do 2PC, com tendencia a estabilizar. A Fig. 2c exibe a descontinuidade média da mídia para os experimentos, apresentando os desvios padrões de cada um, no topo.(Fonte: [6])



(a) Q_1: Quantidade de caminhos



(b) Q_1: Porcentagem de caminhos

Fig. 3. Distribuição de caminhos mínimos para a camada Q_1. A Fig. 3a apresenta a quantidade de caminhos de tamanho 1 por classe de *peer*, enquanto a Fig. 3b apresenta a porcentagem de caminhos por classe presente em Q_1.

Cada experimento tem duração de $2000s$. Como $T(n)$=$10s$, obtemos $0 \leq n \leq 200$, i.e., 200 intervalos de análise por experimento. Para cada árvore R gerada por $T(n)$, associamos a contagem dos caminhos para *peers* em cada camada com as classes definidas pelo algoritmo 2PC. A convergência da *overlay* é observada pela comparação das árvores entre os intervalos $T(i)$ e $T(i+1)$ ao longo dos experimentos. Os códigos fonte de avaliação do 2PC[1], geração dos grafos[2] e *logs*[3] podem ser acessados.

IV. Resultados

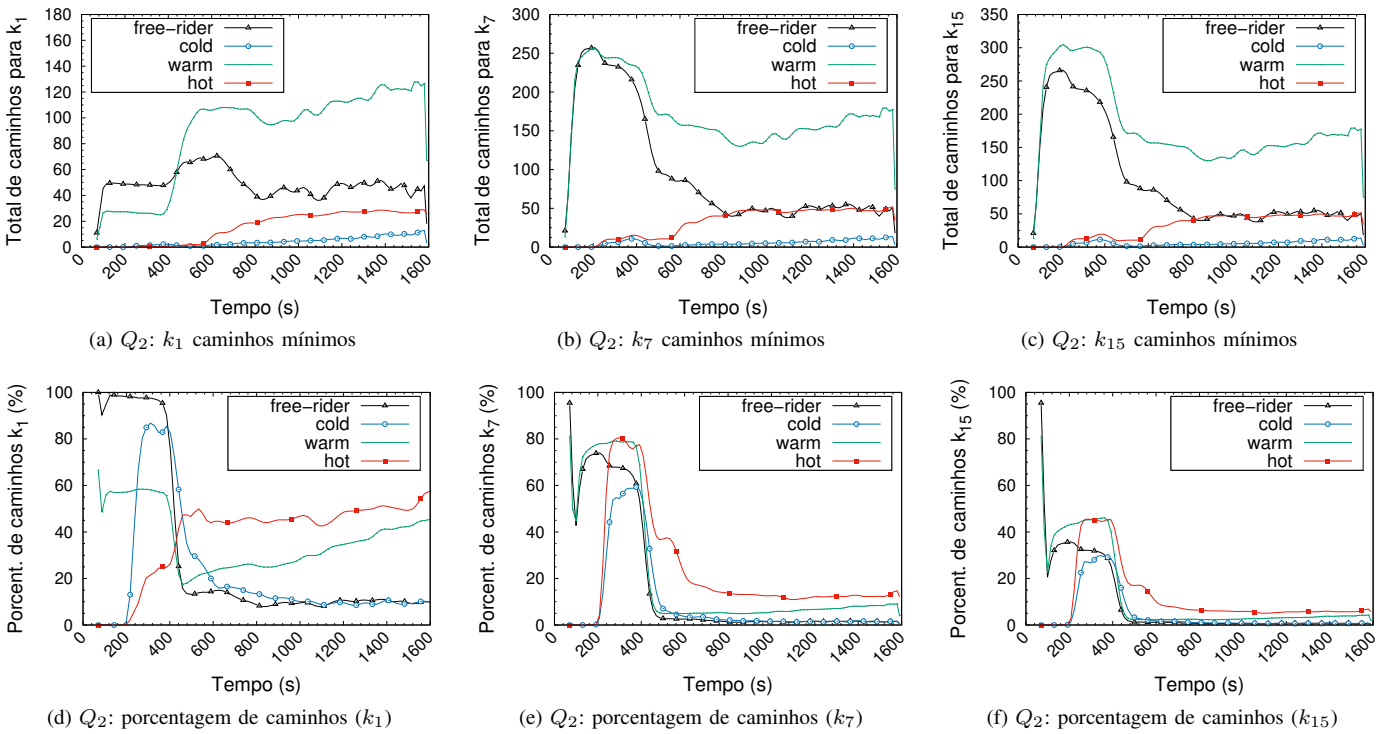
Nesta seção, discutimos os resultados das avaliações de convergência da *overlay*. Apresentamos: o *total* e a *porcentagem* de caminhos mínimos por camada Q_i. Definimos k_1, k_7 e k_{15} como conjuntos de k_j caminhos mínimos para cada classe de *peers* definida por 2PC. Lembramos que $N_{in}(p) = 20$ para $p \neq S$, então p pode receber *chunks* de no máximo 20 parceiros. Em especial, a camada Q_1 só pode ser avaliada para k_1, já que em Q_1 só existe, no máximo, 20 caminhos mínimos entre S e seus parceiros.

A camada Q_1, Fig. 3a, é composta basicamente por *peers* classificados como *hot* e *warm*, com visível decrescimento na população *warm* em favor da *hot* até o instante 1000s, quando estabiliza. Observamos na Fig. 3b que aproximadamente 20% dos *peers hot* presentes no experimento mantiveram-se em Q_1, enquanto a porcentagem de *peers* da classe *warm* tem variação insignificante entre 2% e 4%.

Em relação às figuras de porcentagem de *peers* (como a Fig. 3b e as que seguem), salientamos que antes do instante 400s havia apenas ≈110 *peers* na rede. Como o *flash crowd* acontece no instante 400s, as porcentagem de *peers* são muito altas nas camadas Q_1 e Q_2, que comportam quase todos os *peers* antes do *flash crowd*. Assim, nossa análise por porcentagem de caminhos discute apenas os resultados a partir do instante 400s.

A Fig. 4 representa a camada Q_2. A Fig. 4a mostra quantidade absoluta de caminhos mínimos para *peers warm*, seguida pelos caminhos para *free-rider*, que são 50% dos *peers* da *overlay*, como pode ser visto na Tabela I, superior ao total de caminhos para os *peers hot* (população limitada pelo 2PC a 15% de *peers* cooperativos). Contudo, ao comparar a Fig. 4a com a Fig. 4d, observamos que a maior porcentagem de caminhos na camada Q_2 é para *peers hot*, seguido por *warm*. Destacamos, com proporções menos relevantes, as porcentagem para *free-riders* e *cold*. Finalmente, quando comparamos

[1]https://github.com/ADCDS/tvpp-log-parser
[2]https://github.com/ADCDS/TccPlots
[3]https://github.com/ADCDS/tvpp-log-parser/tree/master/logs



(a) Q_2: k_1 caminhos mínimos (b) Q_2: k_7 caminhos mínimos (c) Q_2: k_{15} caminhos mínimos

(d) Q_2: porcentagem de caminhos (k_1) (e) Q_2: porcentagem de caminhos (k_7) (f) Q_2: porcentagem de caminhos (k_{15})

Fig. 4. Camada Q_2. As Figs. 4a-c apresentam o total de caminhos mínimos por classe de *peers* para grupos de caminhos mínimos k_1, k_7 e k_{15}, respectivamente. As Figs. 4d-f apresentam a porcentagem presentes na camada Q_2 de todos os caminhos mínimos por classe para k_1, k_7 e k_{15}, respectivamente.

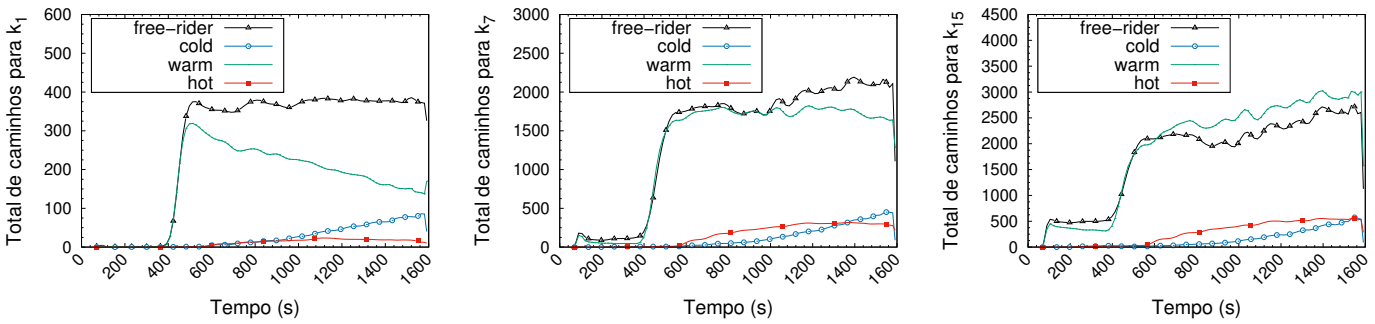
a Fig. 4a com as Figs. 4b-c vemos que, ao considerar mais caminhos mínimos (k_7 e k_{15}), esta camada torna os *peers hot* mais fortemente conectados, alcançando os *free-riders* (que são maioria na rede). Consideramos, assim, que Q_2 é composta em sua maioria por caminhos aos *peers hot* e *warm*.

A Fig. 5 apresenta a camada Q_3. Comparando as três Figs. 5a-5c observamos uma grande presença de *free-riders* nesta camada. Contudo, como em Q_2, à medida em que consideramos mais caminhos mínimos (k_7 e k_{15}) observamos que os *peers hot* e *warm* tornam-se mais fortemente conectados a Q_3, chegando à quantidade de caminhos para *warm* superior à de *free-riders* para k_{15}, o que não acontece com a classe *cold*. Este efeito pode ser observado se compararmos as Figs. 5e-5f com a Fig. 5d. Neste caso, por mais que a população de *peers warm* seja reduzida ao longo do tempo (Fig. 2b), a porcentagem de *peers hot* e *warm* são superiores aos *peers cold* e *free-rider*. Consideramos, assim, que Q_3 atende à demanda de *free-riders* e *cold* na *overlay*, ao tempo que privilegia os *peers* mais cooperativos na proporção de caminhos estabelecidos.

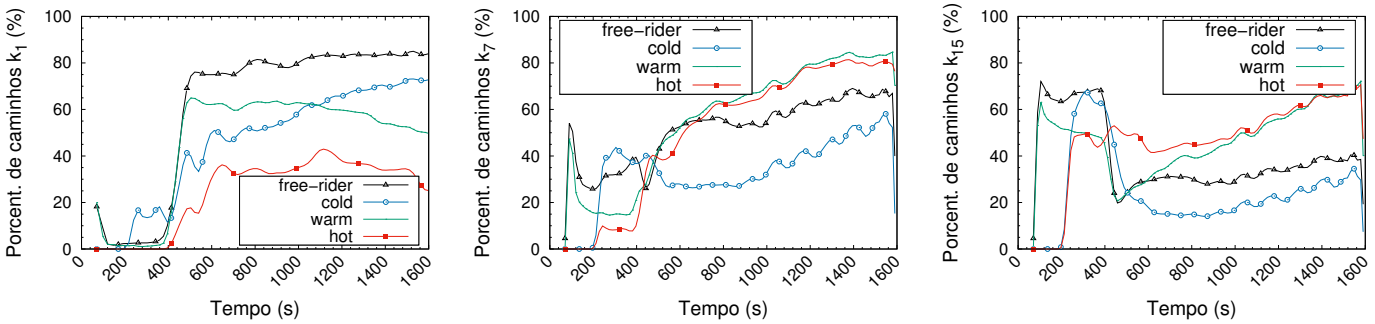
A Fig. 6 representa a camada Q_4. Comparando a quantidade de caminhos mínimos para k_1, k_7 e k_{15} nas Figs. 6(a)-6(c), observamos a presença e manutenção dos caminhos para as classes *free-riders* e *cold*. Já a presença de caminhos para *warm* decresce de forma agressiva ao longo dos experimentos. Os *peers warm*, neste caso, tanto migram para as camadas Q_2 e Q_3, como são classificados para as classes *hot* e *cold*, definido pelo Algoritmo 2PC. Já a quantidade de caminhos para *peers hot* é insignificante e estável. Contudo, como vemos nas Figs. 6a-6c, à medida em que as parcerias vão se estabelecendo ao longo do experimento, a porcentagem de *peers cold* diminui na camada Q_4, migrando, principalmente, para a camada Q_3.

As camadas Q_{5-9} (de Q_5 a Q_9) foram observadas em nossos experimentos (não apresentadas), principalmente para k_{15}. Como observado, o pico de caminhos para *peers warm* e *free-rider* no momento do *flash crowd* na camada Q_4 (Fig. 6a) acontece para as camadas Q_5 à Q_9, mas é momentâneo até que a *overlay* acomode os *peers* recém chegados. Além disso, Q_5 estabiliza no instante 1200s com uma média de 150 caminhos para *free-rider* e *cold*, e próximo a zero para as demais classes, enquanto Q_{6-9} aproximam-se de zero no instante $1000s$.

Análise de *churn*: Observamos a ocorrência de *churn* pelos *peers* do PlanetLab. Contudo, não houve configuração explícita em nossa metodologia para induzir o *churn*, se desconsiderarmos o *flash crowd*. As linhas de *Incoming* e *2PC* na Fig. 2a sinalizam a presença de *churn*, com uma média de aproximadamente 150 *peers* no total desconectados durante a transmissão. Como a execução de cada *peer* é mantida durante todo o experimento, os *peers* que não assistem ao vídeo com qualidade abandonam a transmissão (*churn-out*) para realizar um novo pedido de ingresso (*churn-in*) ao *bootstrap* (característica implementada no *peer*). Esse efeito ocorre quando a *overlay* não suporta a transmissão da mídia com qualidade. Contudo, problemas como sobrecarga de tarefas compartilhadas pelos computadores do PlanetLab podem levar ao *churn* inesperado do *peer*, denominado *livre-arbítrio*. A Fig. 7 representa a média do *churn-out* por livre-arbítrio.

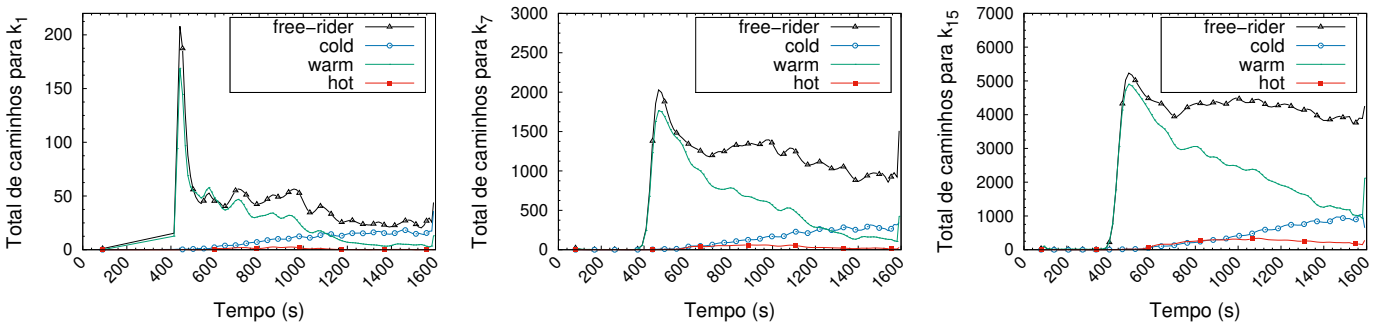


(a) Q_3: k_1 caminhos mínimos (b) Q_3: k_7 caminhos mínimos (c) Q_3: k_{15} caminhos mínimos



(d) Q_3: porcentagem de caminhos (k_1) (e) Q_3: porcentagem de caminhos (k_7) (f) Q_3: porcentagem de caminhos (k_{15})

Fig. 5. Camada Q_3. As Figs. 5a-c apresentam o total de caminhos mínimos por classe de *peers* para grupos de caminhos mínimos k_1, k_7 e k_{15}, respectivamente. As Figs. 5d-f apresentam a porcentagem presentes na camada Q_3 de todos os caminhos mínimos por classe para k_1, k_7 e k_{15}, respectivamente.



(a) Q_4: k_1 caminhos mínimos (b) Q_4: k_7 caminhos mínimos (c) Q_4: k_{15} caminhos mínimos

Fig. 6. Camada Q_4. As Figs. 6a-c apresentam o total de caminhos mínimos por classe de *peers* para grupos de caminhos mínimos k_1, k_7 e k_{15}, respectivamente.

V. Conclusão

A maior contribuição deste trabalho é estudar de forma completa como a organização da rede de sobreposição *overlay* é realizada pelo algoritmo 2PC. Para isso, analisamos os *logs* de *overlay* dos experimentos realizados por [6]. Consideramos combinações entre as quantidades de caminhos mínimos para definir as camadas da rede. Ao agrupar os caminhos por classes de *peers*, mostramos que o algoritmo 2PC promove a migração dos *peers* de alta contribuição para perto do servidor, enquanto os *peers* de baixa ou nenhuma contribuição foram empurrados para as bordas da rede, o que garante o sucesso do algoritmo. Além disso, este trabalho também estudou a construção e manutenção da *overlay* em redes P2P ao vivo e avaliou as taxas de *churn* não induzidas em nossos experimentos, mas inerentes ao PlanetLab. Concluímos que estas taxas podem influenciar negativamente experimentos que configuram a indução do *churn* em proporções preestabelecidas, visto que o *churn* por *livre-arbítrio* no PlanetLab pode não ser considerado. Como trabalhos futuros, pretendemos avaliar a forma em que o 2PC constrói e mantém a *overlay* por meio de registros dos caminhos em que os *chunks* são transmitidos. Com isso, será possível relacionar os resultados deste trabalho e saber quantos caminhos mínimos precisam ser avaliados em uma transmissão P2P para se inferir o real caminho dos *chunks*. Também propomos uma nova experimentação do 2PC considerando diferentes classes de *peers*, além aqui estudadas.

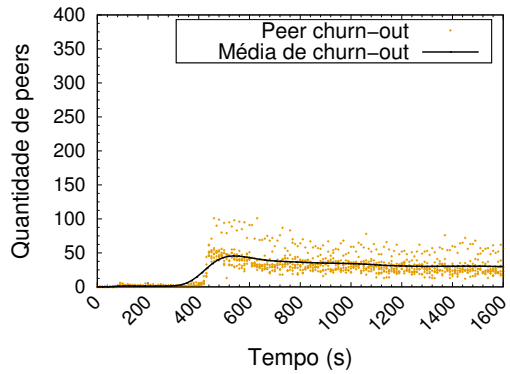


Fig. 7. *Churn* por livre-árbítrio para os experimentos do 2PC.

AGRADECIMENTOS

Os autores agradecem o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), a Fundação Araucária e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelos suportes financeiros.

REFERENCES

[1] Y. Cui, Y. Cao, L. Dai, and Y. Xue, "Optimizing P2P Streaming Throughput Under Peer Churning," *Multimedia systems*, vol. 15, no. 2, pp. 83–99, 2009. DOI: 10.1007/s00530-008-0148-7

[2] H. Terelius and K. H. Johansson, "Peer-to-Peer Gradient Topologies in Networks With Churn," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 2085–2095, 2018. DOI: 10.1109/TCNS.2018.2795704

[3] T. Y. Chung and O. Lin, "A Batch Join Scheme for Flash Crowd Reduction in IPTV Systems," in *Proc. of 2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE, 2011, pp. 823–828. DOI: 10.1109/ICPADS.2011.2

[4] I. Shahriar, D. Qiu, and B. Jaumard, "Modeling of Free Riders in P2P Live Streaming Systems," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 729–734.

[5] Z. Ou, E. Harjula, O. Kassinen, and M. Ylianttila, "Performance Evaluation of a Kademlia-Based Communication-Oriented P2P System Under Churn," *Computer Networks*, vol. 54, no. 5, pp. 689–705, 2010. DOI: 10.1016/j.comnet.2009.09.022

[6] E. C. Miguel, C. M. Silva, F. C. Coelho, Í. F. S. Cunha, and S. V. A. Campos, "Construction and Maintenance of P2P Overlays for Live Streaming," *Multimedia Tools and Applications*, Mar 2021. DOI: 10.1007/s11042-021-10604-w

[7] A. C. dos Santos, C. M. Silva, and E. C. Miguel, "Overlay Convergence Analysis in P2P Networks: An Assessment of the 2PC Algorithm," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6. DOI: 10.1109/3ICT51146.2020.9311950

[8] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-peer Overlay Network Schemes," *Commun. Surveys Tuts.*, vol. 7, no. 2, pp. 72–93, Apr. 2005. DOI: 10.1109/COMST.2005.1610546

[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *Proceedings of the Symposium on Operating Systems Principles*. ACM, 2003, pp. 298–313. DOI: 10.1145/1165389.945474

[10] B. Li, S. Xie, Y. Qu, G. Y. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, April 2008. DOI: 10.1109/INFOCOM.2008.157

[11] F. Picconi and L. Massoulié, "Is There a Future for Mesh-Based live Video Streaming?" in *2008 Eighth International Conference on Peer-to-Peer Computing*, Sept 2008, pp. 289–298. DOI: 10.1109/P2P.2008.18

[12] I. Ullah, G. Doyen, and D. Gaïti, "Towards User-Aware Peer-to-Peer Live Video Streaming Systems," in *Proc. of 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 920–926.

[13] J. Oliveira, Í. Cunha, E. C. Miguel, M. V. Rocha, A. B. Vieira, and S. Campos, "Can Peer-to-Peer Live Streaming Systems Coexist With Free Riders?" in *Proc. of IEEE P2P 2013*. IEEE, 2013, pp. 1–5. DOI: 10.1109/P2P.2013.6688712

[14] B.-C. Wang, A. Chow, and L. Golubchik, "P2P Streaming: Use of Advertisements as Incentives," in *Proc. of the 3rd Multimedia Systems Conference*, 2012, pp. 77–82. DOI: 10.1145/2155555.2155567

[15] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "LayerP2P: Using Layered Video Chunks in P2P Live Streaming," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1340–1352, 2009. DOI: 10.1109/TMM.2009.2030656

[16] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Proc. of Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[17] T. Locher, R. Meier, R. Wattenhofer, and S. Schmid, "Robust Live Media Streaming in Swarms," in *Proc. of the 18th international workshop on Network and operating systems support for digital audio and video*, 2009, pp. 121–126. DOI: 10.1145/1542245.1542273

[18] J. Oliveira, R. Viana, A. B. Vieira, M. Rocha, and S. Campos, "TVPP: A Research Oriented P2P Live Streaming System," *SBRC 2013-Salão de Ferramentas*, 2013.

[19] B. R. Preiss, *Data Structures and Algorithms With Object-Oriented Design Patterns in C++*. John Wiley & Sons, 2008.

[20] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science INFORMS*, vol. 17(11), pp. 712–716, July 1971. DOI: 10.1287/mnsc.17.11.712

Adriel C. dos Santos obtained his Bachelor of Science degree in Computer Science at the Universidade Federal de Alfenas, Brazil.

Iago A. Carvalho obtained his BS in Computer Science (UFSJ, 2014), MS in Computer Science (UFMG, 2016), and PhD in Computer Science (UFMG, 2020). He is an Adjunct Professor at the Computer Science Department of Universidade Federal de Alfenas (DCC / UNIFAL-MG) and Head of the Computational Intelligence Lab of this same institution.

Cristiano M. Silva obtained his BS in Computer Science (UFMG, 2000), MS in Computer Science (UFMG, 2004), MBA (IBMEC, 2008), Specialization in Finances (UCAM, 2010), and PhD in Computer Science (UFMG, 2014). He's an Associate Professor and the Head of the Department of Technology (DTECH/UFSJ), the Director of the Graduate Program in Technological Innovation, and researcher level 2 at the CNPq. Published 120+ refereed papers in flagship journals and international conferences.

Eliseu C. Miguel obtained his BS in Computer Science (UFV, 1998), MS in Computer Science (UFMG, 2004) and PhD in Computer Science (UFMG, 2017). He is an Assistant Professor at the Computer Science Department of Universidade Federal de Alfenas (DCC / UNIFAL-MG) and his research area of interest involves computer networks and distributed systems.