Simulação de Atraso em Rede Local para um Sistema P2P de Streaming de Vídeo ao Vivo

Bruno de Mello Morgan[1], **Murilo Cesar Calado Junior**[1]

[1]Instituto de Ciências Exatas
Universidade Federal de Alfenas (UNIFAL-MG)
Alfenas, Brasil

{bruno.morgan,murilo.cesar}@bcc.unifal-mg.edu.br

Abstract. *The P2P (peer-to-peer) network architecture has more scalability because data exchange is performed in a way that all users act as clients (receive data) and servers (send data). This is why P2P networks have been widely used in live media streaming systems. In this architecture, videos are initially offered through a server to users, who watch the content and share chunks of media with each other. One problem that arises in these networks is the abrupt arrival of a very large number of new users (flash crowd), which can destabilize the transmission to the participants that compose the network and prevent new participants from joining the network and receiving data. In this work, we simulate this effect by replicating real tests, done on the internet, in an local network environment of the Federal University of Alfenas. For this, we implemented changes in a P2P search-oriented live streaming system, the TVPP. Our tests made possible a better understanding of the difference in behavior of users' joining both types of networks. And the changes in TVPP make it possible to approximate actual test simulations in a local area network environment.*

Resumo. *A arquitetura de redes P2P (peer-to-peer) possui maior escalabilidade, pois a troca de dados é realizada de forma que todos os usuários atuam como clientes (recebem dados) e servidores (enviam dados). Por isso as redes P2P têm sido amplamente utilizadas em sistemas de transmissão de mídia ao vivo. Nesta arquitetura, os vídeos são oferecidos inicialmente por meio de um servidor aos usuários, que assistem ao conteúdo e compartilham trechos de mídia entre si. Um problema que surge nestas redes é a chegada abrupta de um número muito grande de novos usuários (flash crowd), podendo desestabilizar a transmissão aos participantes que compõem a rede e impedir que novos participantes se juntem à rede e recebam dados. Neste trabalho, simulamos este efeito, replicando testes reais, feitos na internet, em um ambiente de rede local interna da Universidade Federal de Alfenas. Para isso foram implementadas mudanças em um sistema P2P de transmissão ao vivo orientado a pesquisa, o TVPP. Nossos testes possibilitaram um melhor entendimento da diferença de comportamento no ingresso de usuários nos dois tipos de rede. E as mudanças no TVPP tornam possível aproximar simulações de testes reais em um ambiente de rede local.*

1. Introdução

A arquitetura *peer-to-peer* (par-a-par ou P2P) têm sido muito utilizada em sistemas de *streaming* [Liu et al. 2009a]. Em uma rede P2P, utiliza-se a comunicação direta entre os

usuários conectados alternadamente, chamados pares. Isso permite aumentar a escalabilidade da rede, pois a troca de dados é realizada de forma que todos os participantes são clientes (recebem dados) e servidores (enviam dados) [KUROSE 2010].

Devido à escalabilidade que as redes P2P proporcionam, aplicações de transmissão de vídeo ao vivo, foco deste trabalho, têm usado esta arquitetura, permitindo que uma grande quantidade de usuários assistam simultaneamente ao conteúdo. Nestes sistemas, a mídia deve ser distribuída em um curto prazo de tempo para a visualização sem interrupções no conteúdo transmitido. [Chen et al. 2014, Liu et al. 2012].

Mesmo tendo maior escalabilidade que redes cliente-servidor, sistemas P2P de transmissão de mídia ao vivo possuem problemas em relação à estabilidade, dinamicidade dos usuários e a confiabilidade dos serviços oferecidos. Um problema que podemos citar são os *free-riders*, usuários que não compartilham o conteúdo recebido. Este comportamento interfere na transmissão do vídeo entre os usuários [Wu et al. 2011]. Outro problema é o *churn*, isto é, a dinamicidade de entrada e saída de usuários a qualquer momento. Isso prejudica a qualidade da transmissão, pois com a saída de usuários, novas parcerias devem ser feitas, e com a chegada de novos usuários aumentam as requisições por dados [Wu et al. 2012].

Quando um usuário ingressa na rede ele não tem dados para compartilhar com os outros durante um tempo, se isso ocorre poucas vezes a rede pode facilmente se estabilizar. Porém, quando novos usuários ingressam em grandes proporções, a rede pode não ser capaz de fornecer dados a todos os usuários. Isso pode desestabilizar a transmissão aos participantes que compunham a rede e impedir que novos participantes recebam dados. Esta chegada abrupta de um número muito grande de novos usuários é conhecida como *flash crowd* [Chen et al. 2011, Li et al. 2008, Liu et al. 2009b, Oliveira et al. 2013b].

Para minimizar os efeitos do *flash crowd*, são propostas algumas soluções para promover melhorias no processo de ingresso de grandes grupos de novos usuários, a fim de manter a estabilidade na distribuição de mídia aos participantes que compunham a rede, sem prejudicar os novos participantes [Liu et al. 2009b, Chen 2009]. Uma solução proposta para tratar o *flash crowd* baseia-se em isolar os novos usuários em redes paralelas, tendo como servidor um par estável já existente na rede [Miguel et al. 2016].

Neste trabalho, simulamos o efeito do *flash crowd* em um ambiente de rede local, replicando alguns testes reais, feitos na internet, que buscam melhorar a taxa de ingresso de novos pares em um sistema sob *flash crowd*. Para isso foram implementadas mudanças no TVPP, um sistema P2P de transmissão ao vivo orientado a pesquisa. Nossos testes possibilitaram um melhor entendimento da diferença de comportamento no ingresso de usuários nos dois tipos de rede. E as alterações no TVPP tornam possíveis uma simulação aproximada de testes reais em ambiente de rede local.

2. Trabalhos Relacionados

Em [KUROSE 2010] os autores explicam brevemente as arquiteturas de redes cliente-servidor e *peer-to-peer*. É demonstrado que redes P2P (baseadas em UDP) são auto escaláveis, o que pode ser vantajoso em relação a redes clientes-servidor (baseados em TCP). Contudo, os autores também apontam problemas de estabilidade, dinamicidade de usuários e confiabilidade nas redes P2P.

No artigo [Liu et al. 2009a] os autores desenvolveram um modelo matemático para o relacionamento entre tempo e escalabilidade em sistemas P2P de transmissão ao vivo sob *flash crowd*. Esta análise trouxe também uma compreensão no efeito do protocolo *Gossip* e do *churn*.

No trabalho [Chen et al. 2014] foi estudado a capacidade do sistema, a latência de início dos pares, e o tempo de recuperação do sistema com e sem controle de admissão de *flash crowd*. O estudo demonstra que, sem controle de admissão, o sistema tem capacidade limitada para lidar com *flash crowd*. Essa capacidade é independente do estado inicial do sistema.

Nos trabalhos [Joao F. Oliveira 2013, Miguel et al. 2016, Oliveira et al. 2013b] são apresentados experimentos relacionados ao ingresso de pares e ao *flash crowd* em um sistema específico, o TVPP. Nestes trabalhos foram definidos alguns conceitos importantes ao entendimento destes sistemas. Mais especificamente, em [Joao F. Oliveira 2013] é apresentado um problema comum desta aplicação, os usuários chamados *free riders*, que os autores defendem que podem conviver naturalmente nestas redes.

3. Sistemas P2P de Transmissão de Vídeo ao vivo

Definimos um sistema P2P de transmissão de vídeo ao vivo como um sistema com um conjunto de pares que colaboram um com o outro trocando dados a fim de assistir uma transmissão de mídia ao vivo. O servidor S é um par especial que codifica o vídeo, divide o vídeo em *chunks* (trechos) ou *substreams*, e inicia a transmissão.

Cada par p possui um conjunto de pares parceiros $\mathcal{N}(p)$ em que se conecta e troca *chunks* de vídeo. Para que haja um maior controle entre as parcerias, $\mathcal{N}(p)$ é dividido em dois subconjuntos de pares, $\mathcal{N}_i(p)$ e $\mathcal{N}_o(p)$, que possuem, respectivamente: parcerias *in*, parceiros que enviam *chunks* a p; e parcerias *out*, parceiros que recebem dados de p. O número máximo de parceiros de p é dado por $N_i(p) = |\mathcal{N}_i(p)|$ e $N_o(p) = |\mathcal{N}_o(p)|$. Para S, $\mathcal{N}_i(p) = \emptyset$. Para ingressar em um canal de transmissão ao vivo, um par p registra-se em um servidor B centralizado chamado de *bootstrap*, que retorna a p um subconjunto de pares atualmente ativos no sistema como potenciais parceiros. O par p, então, seleciona pares deste subconjunto e tenta estabelecer parcerias com ele.

Todo relacionamento $p \in \mathcal{N}_o(p')$ requer $p' \in \mathcal{N}_i(p)$. As parcerias bem-sucedidas estabelecem $\mathcal{N}(p)$. Quando p detecta que um de seus parceiros $p' \in \mathcal{N}(p)$ está em silêncio por um período de tempo pré-definido, p remove p' de $\mathcal{N}(p)$. Isto não é um problema pois o par p periodicamente entra em contato com o servidor *bootstrap* para obter uma nova lista de potenciais parceiros para repor a parceria perdida.

Várias obras não impõem um valor fixo de N_o para os pares na rede. Isto garante que se p é escolhido por $\mathcal{N}_i(p')$, implica que p deve aceitar p' em $\mathcal{N}_o(p)$ [Lobb et al. 2009, Traverso et al. 2015]. Por outro lado, nós fixamos tanto N_i quanto N_o, com os pares escolhendo aleatoriamente suas parcerias *in*. Para organizar a topologia da rede, pares estão aptos a aceitar uma nova parceria *out* mesmo quando $\mathcal{N}_o$ está cheio. Neste caso, p analisa a largura de banda da nova requisição de parceria *out* e desconecta aleatoriamente um $q \in \mathcal{N}_o(p)$, que possui a pior largura de banda, e inicia a nova parceria. Depois disso, p permanece incapaz de atendar a novas requisições de parcerias *out* pelos próximos 60 segundos.

Cada par possui um *buffer* local para armazenar seus *chunks* de vídeo. Periodicamente, os pares trocam seus mapas de *buffer* com seus parceiros $\mathcal{N}_o$ para que eles sejam informados sobre quais *chunks* possuem. Cada par, também periodicamente, checa quais *chunks* necessita, identifica quais parceiros $\mathcal{N}_i(p)$ o possuem e envia uma solicitação de que precisa do *chunk* faltante. Neste trabalho, nós agendamos solicitações de *chunks* usando a política do primeiro prazo mais antigo com *Simple Unanswered Request Eliminator* (SURE) [Oliveira et al. 2013a]. SURE permite que cada par p tenha uma lista negra contendo os pares que não respondem para evitar novas falhas de solicitações. Nós não limitamos o número de solicitações pendentes simultâneas. Entretanto, nós limitamos o número de tentativas de solicitação em seis, para controlar as oportunidades de cada parceiro de realizar o *download* de um *chunk* e criar oportunidades de *download* independentemente do tamanho do *buffer*. Um par considera que uma requisição estourou seu tempo se ela não for respondida dentro de 500 milissegundos. Finalmente, pares cooperativos servem imediatamente suas requisições recebidas em ordem de chegada.

Os pares enviam relatórios de monitoramento ao servidor *bootstrap* a cada dez segundos. Estes relatórios, na verdade, incluem o número de *chunks* gerados (apenas relatados pelo servidor de vídeo), enviados, recebidos e aqueles que não cumpriram o prazo de reprodução; o número de solicitações enviadas, respondidas e repetidas; o comprimento médio do caminho de encaminhamento, a contagem de repetições e o tempo de chegada dos *chunks* recebidos; tamanho da vizinhança; e o número de *chunks* duplicados recebidos. Assim, usamos estes relatórios de pares para calcular as métricas de desempenho avaliadas: latência de *chunks* e taxa de falhas de reprodução dos *chunks*.

3.1. *Flash crowd* em Sistemas de vídeo ao vivo

Ao ingressar na rede, o usuário fica um tempo sem possuir dados suficientes para compartilhar com outros pares. Se a chegada de novos pares ocorre regularmente, a rede pode facilmente se estabilizar. Porém, quando novos usuários chegam em grandes proporções, a rede pode não ser capaz de prover dados a todos. Isso pode desestabilizar a transmissão aos participantes que compunham a rede e impedir que novos participantes recebam dados. Esta chegada abrupta de um número muito grande de novos usuários é conhecida como *flash crowd* [Chen et al. 2011, Chen et al. 2014, Liu et al. 2009a, Joao F. Oliveira 2013].

Alguns sistemas controlam a quantidade de usuários que ingressam em um curto tempo e, portanto, possuem excelente capacidade de lidar com *flash crowd* a uma escala logarítmica com base no tamanho do *flash crowd*.

Sistemas que não possuem controle de admissão não conseguem lidar com *flash crowd*. Enquanto sua capacidade máxima não é excedida, o sistema pode se recuperar sozinho. Porém quando a capacidade é excedida, nenhum usuário recém-chegado consegue obter *chunks* suficientes para iniciar a reprodução. Esse fenômeno é chamado de colapso [Chen et al. 2011].

A capacidade máxima de entrada de usuários no sistema pode ser representada pelo nível de choque do maior *flash crowd* que o sistema pode sobreviver, isto é, a relação entre a taxa de chegada de pares durante e antes do *flash crowd* [Chen et al. 2014].

3.2. TVPP

O TVPP é um sistema P2P de transmissão de vídeo ao vivo, que possui código aberto e é voltado para pesquisas em redes e sistemas distribuídos, ele permite organizar os usuários de forma sobreposta à estrutura física da rede. Com isso, é possível basear-se na topologia em malha, com participantes conectados alternadamente entre si [Oliveira et al. 2013b].

Nós utilizamos o TVPP [Oliveira et al. 2013b] para executar os experimentos deste trabalho. Com ele, buscamos replicar alguns testes realizados em uma rede real na internet formada por nós do PlanetLab[1] [PlanetLab 2009].

3.3. Técnicas para tratamento do *flash crowd*

Para minimizar os efeitos do *flash crowd*, são propostas algumas soluções para promover melhorias no processo de ingresso de grandes grupos de novos usuários, a fim de manter a estabilidade na distribuição de mídia aos participantes que compunham a rede, sem prejudicar os novos participantes [Chen 2009, Miguel et al. 2016].

Neste trabalho, consideramos três cenários: (1) o primeiro, onde não existe técnica de tratamento do *flash crowd*, (2) o segundo, onde o *flash crowd* é tratado por meio de uma técnica de ingressos por slot, e (3) o terceiro onde o *flash crowd* é tratado com uma técnica de redes paralelas.

No primeiro cenário, como não existe tratamento do *flash crowd*, quando ele ocorrer será possível observar as consequências deste efeito e como a rede se comportará.

A segunda, chamada Técnica de Ingresso por Slot, é uma técnica para tratamento de *flash crowd* encontrada em [Liu et al. 2012] que segura a entrada dos pares recém-chegados em um grupo $\mathcal{P}$ para controlar o processo de entrada na rede. Basicamente, em cada iteração i, o sistema avalia os recursos de rede e estabelece $\mathcal{R}_i \subseteq \mathcal{P}$ e τ_i. Significa que todos os pares $p \in \mathcal{R}$ possuem permissão para entrar e a iteração $(i + 1)$ acontece após o tempo de estabilização da rede τ_i. O processo se repete até que $\mathcal{P} = \emptyset$.

Por fim, a terceira, chamada de Técnica de Redes Paralelas, usa uma técnica de rede paralela encontrada em [Miguel et al. 2016]. Seja M a rede antes do *flash crowd* com a malha da rede P2P construída em torno do servidor de mídia S. Nesta técnica, o servidor *bootstrap* B seleciona um conjunto de pares $S_{aux} \subset M$. Cada par $p \in S_{aux}$ se torna um servidor auxiliar. A rede M garante a transmissão de mídia aos servidores auxiliares, mas eles não fornecem *chunks* aos pares em M. Esta etapa é chamada de isolamento dos servidores e prepara a rede para o evento do *flash crowd*. Quando os novos pares começam a chegar, eles são distribuídos em torno de cada servidor auxiliar, estabelecendo redes paralelas. As novas relações são estabelecidas apenas entre pares da mesma rede paralela. A última etapa é a fusão das redes paralelas. Nesta etapa, permitimos que os pares das redes paralelas possam trocar dados com pares de M.

4. Ambiente de Trabalho

Para replicar os experimentos do PlanetLab, configuramos nosso ambiente da mesma maneira em que estes experimentos foram executados. Para isso, utilizamos 55 computadores conectados em uma mesma rede local, utilizando o sistema P2P TVPP. Foram

[1] PlanetLab é uma rede de pesquisa global, com cerca de 1100 nós espalhados em mais de 500 lugares.

executadas cinco repetições para cada experimento e nossos resultados são baseados na média destes testes. Configuramos o servidor de vídeo e o *bootstrap* em um computador da rede e usamos outros 54 computadores executando os clientes (pares comuns). O servidor de mídia transmite um vídeo de 420 kbps (cerca de 40 *chunks* por segundo).

Nossos experimentos foram realizados em 1350 segundos. Os primeiros 70 segundos são usados para inicialização do *bootstrap* e do servidor de mídia. Então construímos uma rede inicial com um grupo de 108 pares (2 em cada máquina) para apoiar a chegada repentina de mais de 1080 pares (20 em cada máquina), quando acontece o *flash crowd*, aos 350 segundos após o início de cada experimento.

Definimos 4 tipos de largura de banda de *upload* e tamanho de parceria para cada par. A Tabela 1 mostra a configuração dos pares. Quando $N_o = 0$ na classe 0, é definido um *free rider* que não informa a ninguém seu mapa de *buffer*. Dessa forma, todos os parceiros dos *free riders* sabem que eles não estão dispostos ou não conseguem realizar *upload* de dados [Oliveira et al. 2013a]. Finalmente, todos os pares ingressantes possuem a mesma configuração de recursos.

Tabela 1. Configuração de rede dos pares

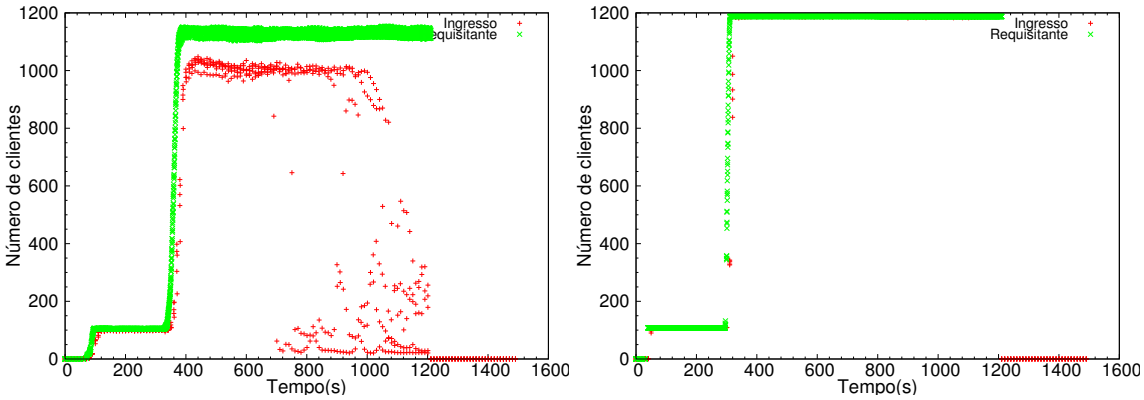
Classe do par	Mb/s	Proporção	N_i	N_o
classe 0	0.0	40%	10	00
classe 1	1.5	27%	10	09
classe 2	2.5	22%	10	20
classe 3	4.0	11%	10	23

Para estudo do *flash crowd* neste artigo, definimos: (1) rede principal, composta pelo conjunto de pares ingressos antes da ocorrência do *flash crowd*; e (2) a rede inteira composta pelo conjunto de todos os pares. Assim, antes do *flash crowd*, a rede principal e toda a rede são iguais. Finalmente, temos um servidor de mídia com $N_o(s) = 20$ e sem limite de largura de banda de *upload*.

4.1. Testes Iniciais

Configurado o ambiente e implantado o sistema TVPP nas máquinas, executamos o primeiro experimento com base nas configurações apresentadas. A Figura 1 mostra o gráfico da rede gerado a partir do experimento. Nele podemos observar o número de requisições (pares que desejam ingressar na rede) e o número de ingressos (pares que conseguiram ingressar na rede). Pode-se notar, no experimento realizado em nossa rede local, que o número de pares que ingressam na rede quando ocorre o *flash crowd* é exatamente igual ao número de requisições, algo que no experimento executado no PlanetLab não acontece. No PlanetLab, quando ocorre o *flash crowd*, o número de ingressos nunca alcança o número de requisições, o que esclareceu a dúvida de que isso poderia ocorrer devido à implementação do TVPP ou configuração da rede, mas é visto que esta ocorrência se dá devido às limitações de performance em uma rede real, tais como atrasos e perdas de pacote.

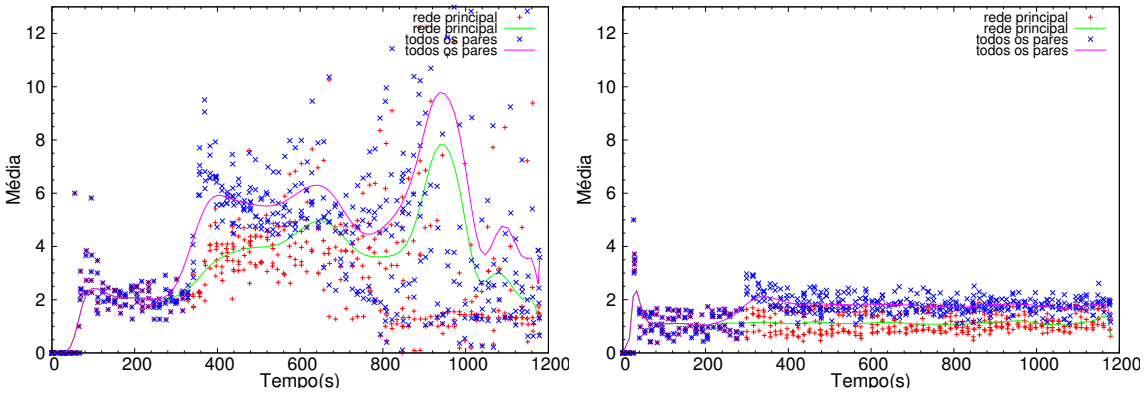
A partir deste primeiro teste foi possível esclarecer e enxergar que em uma rede local tudo funciona melhor, e faz com que os testes lá executados não se aproximem de maneira fácil dos resultados de uma rede real. A Figura 2 apresenta o gráfico da latência



(a) Flash Crowd no PlanetLab. (b) Flash Crowd na Rede Local

Figura 1. Gráfico de rede

dos experimentos, com a latência dos pares da rede principal (rede antes do *flash crowd*) e a latência de todos os pares, e mostra a diferença entre o experimento do PlanetLab, que possui latência maior que o experimento realizado em nossa rede local. Para que fosse possível replicar os testes e aproximar os resultados, seria necessário que nossa rede local contasse com uma maior latência.



(a) Flash Crowd no PlanetLab. (b) Flash Crowd na Rede Local

Figura 2. Gráfico de Latência

4.2. Alterações do TVPP

Precisávamos simular latência na troca de pacotes entre as máquinas da rede, então, como primeira alternativa nos deparamos com o comando de controle de tráfego de rede do Linux, *tc* [Brown 2006], porém este comando necessita de permissão de *root* para execução, o que não possuíamos. Sendo assim, aproveitamos o fato de que o TVPP é um sistema de código aberto, aplicado em diversas pesquisas na área de redes e sistemas distribuídos, e então, demos início à implementação da latência criando um atraso no envio de pacotes, tentando simular atraso na fila de envio de dados entre os pares. Dessa maneira, além de ser útil no desenvolvimento deste trabalho, nossas alterações podem servir a trabalhos futuros envolvendo pesquisa em redes com baixa latência.

Para a implementação da latência, alteramos partes do código do TVPP, mais especificamente criamos variações do método UDPSend da classe *client*. Essa função é

executada em uma *thread* que verifica a existência de pacotes em uma fila e os envia. Com a chegada de novos pacotes, a função verifica a idade da mensagem e, caso ela seja menor que 500 ms, ela é enviada e retirada da fila. Do contrário ocorre perda por *timeout*.

Foram implementadas duas soluções diferentes: o Método 1 coloca atraso em todas as mensagens que são enviadas pelo *client* e o Método 2 atrasa apenas as mensagens de dados, isolando as mensagens de controle que são menores e, portanto, são enviadas mais rapidamente.

Foram adicionados novos parâmetros para a execução do *client*. São eles, minimumDelay e maximumDelay, que respectivamente indicam a latência mínima e máxima em milissegundos, delayMode, que indica qual método de atraso será usado, e groupsToSend, que indica em quantos grupos a fila de mensagens será divida e enviada aos poucos.

No Método 1, criamos uma função baseada na UDPSend, chamada UDPSendWithDelay, que sorteia uma latência em milissegundos entre o mínimo e o máximo passados como parâmetro, então a função espera o tempo e envia todas as mensagens contidas na fila. Este método se mostrou muito agressivo em relação à latência média, descontinuidade (perda de pacote de dados) e ingresso na rede.

Sendo assim, decidimos manter esta implementação visto que funciona para alguns casos com pouca latência. E criamos outro método menos agressivo que aproximou os resultados obtidos aos resultados dos mesmos experimentos realizados no PlanetLab.

Como é de nosso interesse simular latência entre os pares sem prejudicar o ingresso e a reprodução da mídia, pensamos em isolar as mensagens de controle, que são menores em bytes e portanto são transmitidas com pouco atraso, a fim de garantir uma boa comunicação entre os pares para formação de parcerias.

Desta maneira, no Método 2, criamos duas funções, UDPSendWithCyclicTimer, baseada na UDPSend, e uma outra chamada CyclicTimerSend. Cada uma das funções é executada por *threads* separadas. A função UDPSendWithCyclicTimer verifica se as mensagens da fila são de dados ou de controle. Se forem de controle, elas são enviadas imediatamente. Se forem de dados, elas são enfileiradas em outra fila específica para mensagens de dados. Esta fila de mensagens de dados é esvaziada a cada vez que uma variável sendChunks (tipo *boolean*), controlada pela função CyclicTimerSend, for verdadeira, e então, ao final, atribui *false* a ela.

A função CyclicTimerSend controla o tempo de atraso. Caso o parâmetro groupsToSend seja indicado, a fila de dados é dividida em grupos conforme a quantidade configurada, e então cada grupo é enviado com atrasos diferentes. Essa divisão em grupos auxilia na aleatoriedade de atraso das mensagens, fato comum em redes reais. Caso groupsToSend não for configurado, ou for igual a 1, a função CyclicTimerSend controla o atraso sorteando um tempo entre o mínimo e o máximo passados por parâmetro, e então atribui verdadeiro a variável sendChunks, que dispara o envio da fila de *chunks*.

4.3. Configurações de atraso para os experimentos

Em uma rede de computadores existem vários tipos de atraso: atraso de processamento, que está relacionado ao tempo requerido para se examinar o cabeçalho de um pacote e decidir sua rota; atraso de fila, relacionado a espera de um pacote para ser enviado; atraso de transmissão, que está relacionado à velocidade do enlace e ao tamanho do pacote e o

atraso de propagação, que representa o tempo que um bit demora para se propagar de A para B [KUROSE 2010].

Com base nisso, e sabendo que a velocidade da luz limita a propagação de sinais eletromagneticos, podemos pensar que, em uma rede hipotética ideal, a latência mínima *Lmin* de propagação é dada por $L_{min} = S/C_m$, onde S é a distância e C_m é a velocidade da luz no meio (fibra óptica).

Sendo assim, nesta rede hipotética, pensando no pior caso como a maior distância S entre dois pontos do planeta sendo aproximadamente 20000 km, e consideramos a velocidade da luz na fibra óptica seja de 200.000.000 m/s (considerando refração de 1,5). Então, temos que a latência mínima para este pior caso é de 100 ms [O'Reilly 2013]. Sendo assim, assumimos nossa latência mínima como 100 ms e limitamos a latência em 350 ms. Os experimentos foram executados com as configurações presentes na Tabela 2.

Tabela 2. Configuração dos experimentos

Método	Latência Mínima	Latência Máxima	Grupos
método 1	100 ms	350 ms	N/A
método 2	100 ms	350 ms	3

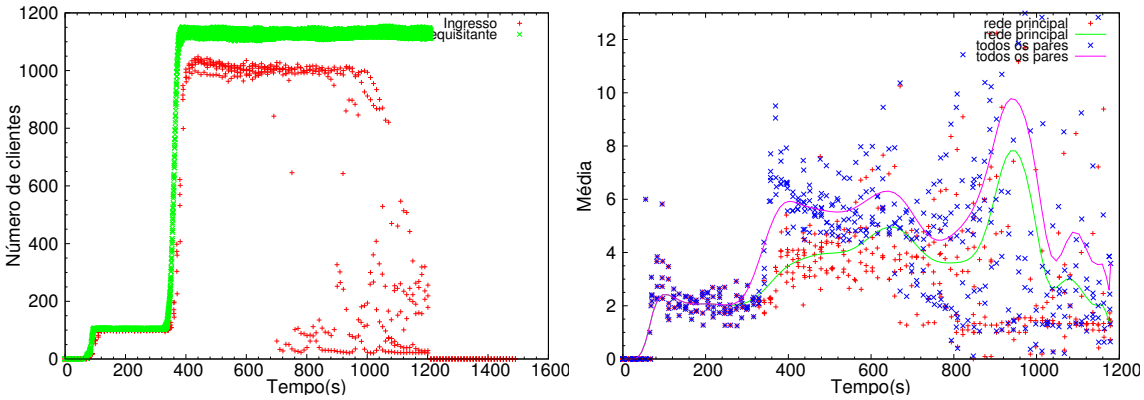
5. Resultados

Para garantir a qualidade dos resultados e observar as consequências de nossas implementações, trabalhamos com três técnicas em um ambiente em que ocorre o *flash crowd*, sendo que, no mundo real, duas delas conseguem tratar este evento e manter os pares na rede e uma delas não consegue manter a rede estável e os pares conectados começam a cair. O uso de diferentes técnicas é para garantir que, com nossos métodos de atraso implementados, a rede reaja como deve reagir, mesmo em diferentes cenários. Para comparação de nossos resultados, obtemos os mesmos experimentos executados na rede do PlanetLab, o que garante um comparativo real e confiável para os resultados obtidos em nossos experimentos.

A primeira técnica não possui tratamento de *flash crowd* e foi rodada apenas com as configurações iniciais, com o *flash crowd* acontecendo aos 350 segundos após o início de cada experimento.

A segunda, chamada Técnica de Ingresso por Slot, é uma técnica para tratamento de *flash crowd* que segura a entrada dos pares recém-chegados para controlar o processo de entrada na rede. Neste caso os pares do *flash crowd* foram separados em 6 grupos e a cada 100 segundos, um grupo de pares é colocado na rede.

Por fim, a terceira, chamada de Técnica de Redes Paralelas. Nesta técnica, alguns pares são selecionados para servirem como servidores auxiliares após a ocorrência do *flash crowd*. Quando os pares chegam, são distribuídos em torno de cada servidor auxiliar, estabelecendo redes paralelas. Com o tempo, as redes paralelas vão se mesclando aos pares da rede principal. Em nosso experimento, foram utilizados 6 servidores auxiliares. Após o *flash crowd*, esperamos 200 segundos para construir a topologia de redes paralelas, e, em seguida, começamos a mesclar cada uma das redes a cada 100 segundos.



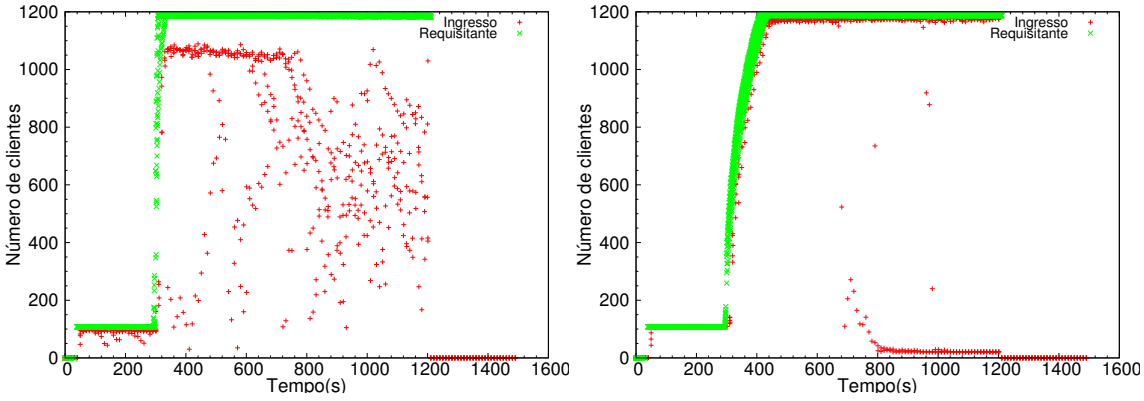
(a) Gráfico de Rede (b) Gráfico de Latência

Figura 3. Experimento no PlanetLab (Técnica sem tratamento de Flash Crowd)

5.1. Técnica sem tratamento de Flash Crowd

A Figura 3 possui os gráficos de rede e latência referentes ao experimento da Técnica sem tratamento de Flash Crowd realizado no PlanetLab. É possível observar o comportamento da rede, percebendo que os pares ingressos nela, após o acontecimento do *flash crowd*, começam a cair, a partir, aproximadamente, dos 700 segundos de experimento. É possível também, nesta figura, observar o gráfico de latência da rede. Os resultados apresentados nestes gráficos são os resultados esperados e que tentamos simular em nossos experimentos com nossa simulação do atraso.

Na Figura 4, encontramos os gráficos da rede dos experimentos realizados em nossa rede local aplicando os métodos de atraso implementados, e podemos observar que o Método 1 fez com que os pares ingressos na rede começassem a cair rapidamente a partir, aproximadamente, dos 500 segundos de experimento. No Método 2 os pares começam a cair mais próximos do fim do experimento, em torno dos 700 segundos.

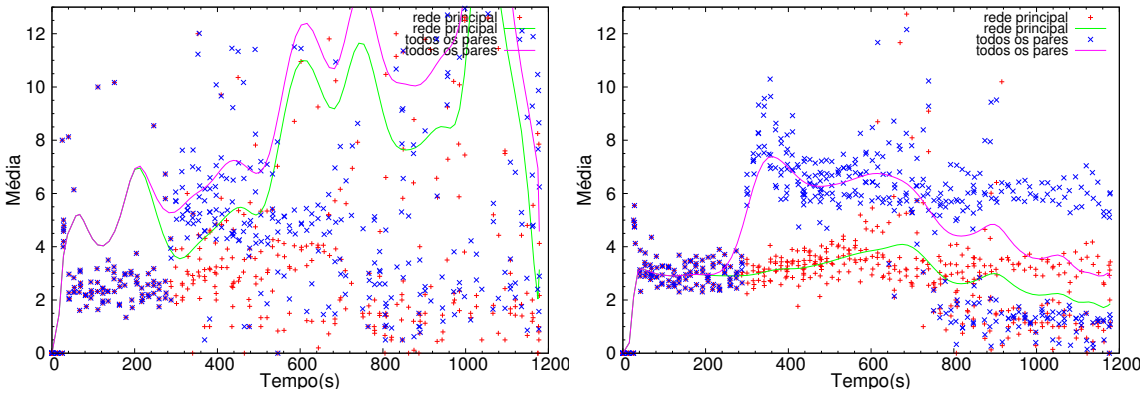


(a) Método 1 (b) Método 2

Figura 4. Gráfico de Rede (Técnica sem tratamento de Flash Crowd)

Na Figura 5, temos os gráficos de latência dos experimentos realizados em nossa rede local utilizando os métodos de atraso implementandos e podemos comparar a latência obtida nos experimentos de cada método, podendo observar que a latência no Método 1 é

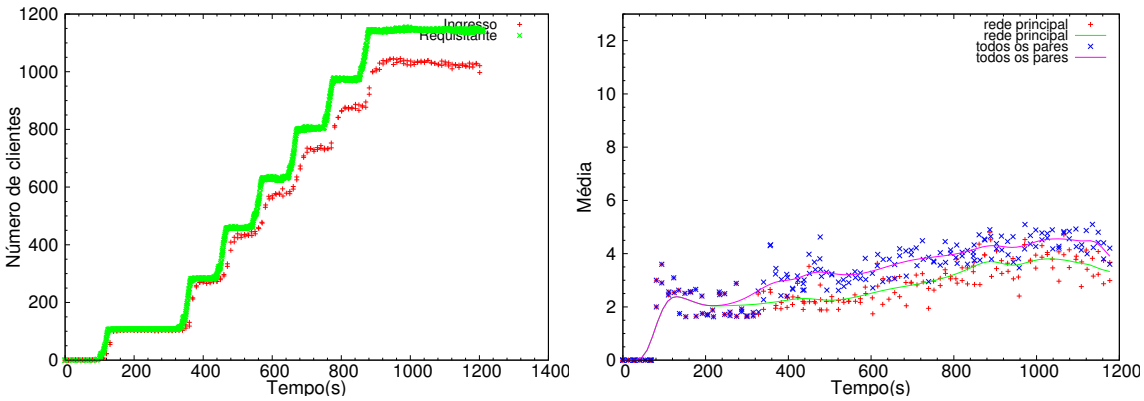
muito grande e difere do esperado, enquanto que no gráfico de latência do Método 2, há uma maior aproximação ao comportamento dos resultados obtidos no PlanetLab.



(a) Método 1 (b) Método 2

Figura 5. Gráfico de Latência (Técnica sem tratamento de Flash Crowd)

5.2. Técnica de Ingresso por Slot

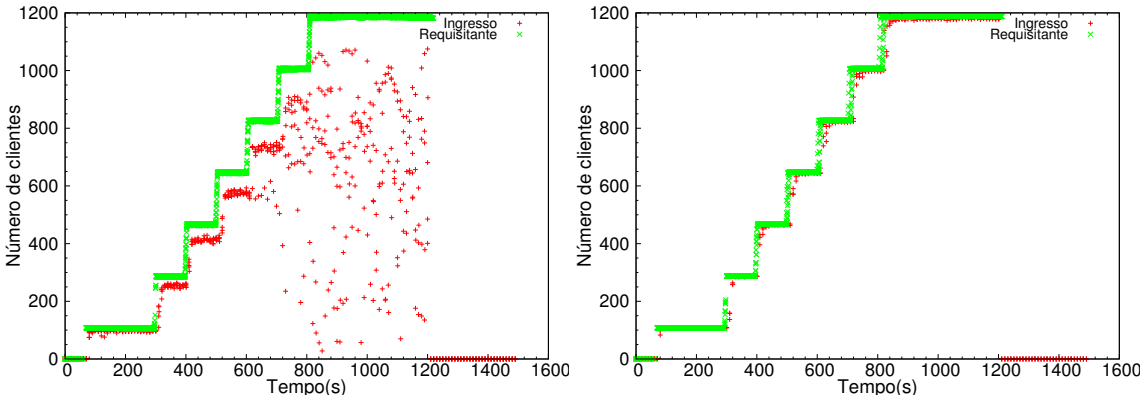


(a) Gráfico de Rede (b) Gráfico de Latência

Figura 6. Experimento no PlanetLab (Técnica de Ingresso por Slot)

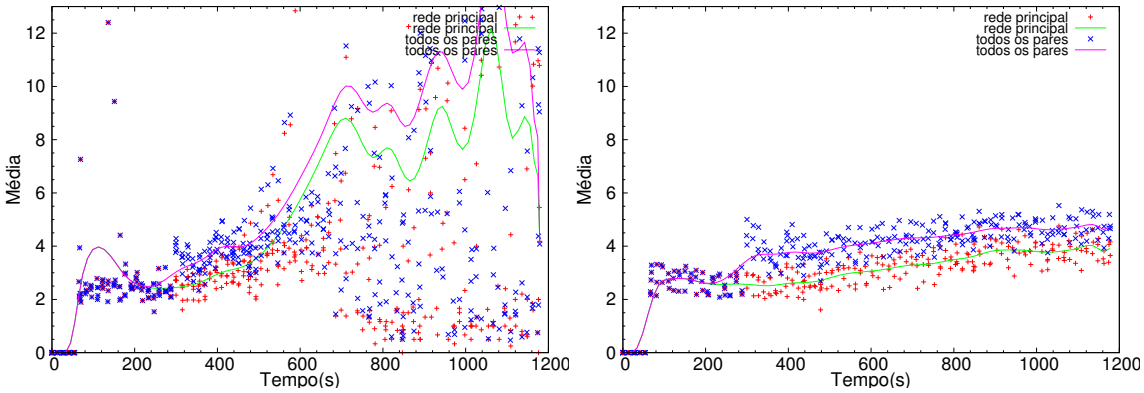
A Figura 6 possui os gráficos de rede e latência referentes ao experimento da Técnica de Ingresso por Slot realizado no PlanetLab. É possível observar o comportamento da rede, percebendo que a técnica, que vai permitindo o ingresso dos pares a cada 100 segundos, consegue tratar o *flash crowd* e evitar a queda da rede. É possível também, nesta figura, observar o gráfico de latência da rede, que cresce aos poucos conforme novos pares ingressam na rede. Os resultados apresentados nestes gráficos são os resultados esperados e que tentamos simular em nossos experimentos com nossa simulação do atraso.

Na Figura 7, temos os gráficos da rede dos experimentos realizados em nossa rede local aplicando os métodos de atraso implementados. Com essa técnica, podemos observar que no experimento do Método 1, pares começaram a cair em torno dos 800 segundos, próximo ao quarto ”degrau”, e não atinge o resultado esperado desta técnica, que no PlanetLab consegue evitar a queda dos pares. Com o Método 2, a os pares vão ingressando aos poucos, sem que haja queda dos pares, como previsto na técnica.



(a) Método 1 (b) Método 2

Figura 7. Gráfico de Rede (Técnica de Ingresso por Slot)



(a) Método 1 (b) Método 2

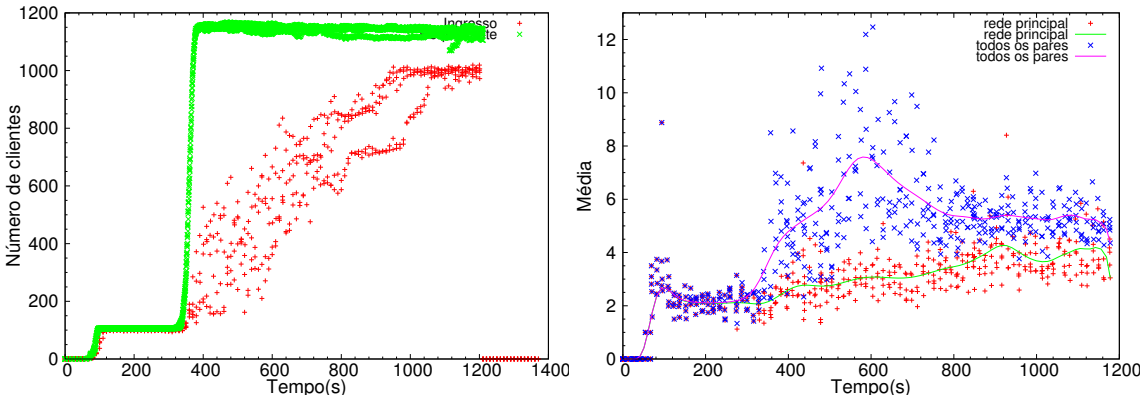
Figura 8. Gráfico de Latência (Técnica de Ingresso por Slot)

Na Figura 8, temos os gráficos de latência dos experimentos realizados em nossa rede local utilizando os métodos de atraso implementados e podemos notar uma latência muito alta no gráfico do Método 1, devido à queda dos pares no decorrer do experimento. O Método 2 apresenta um comportamento da latência muito parecido com o comportamento esperado, que pode ser observado no gráfico de latência do PlanetLab encontrado na Figura 6, em que a latência cresce aos poucos conforme os pares vão ingressando na rede.

5.3. Técnica de Redes Paralelas

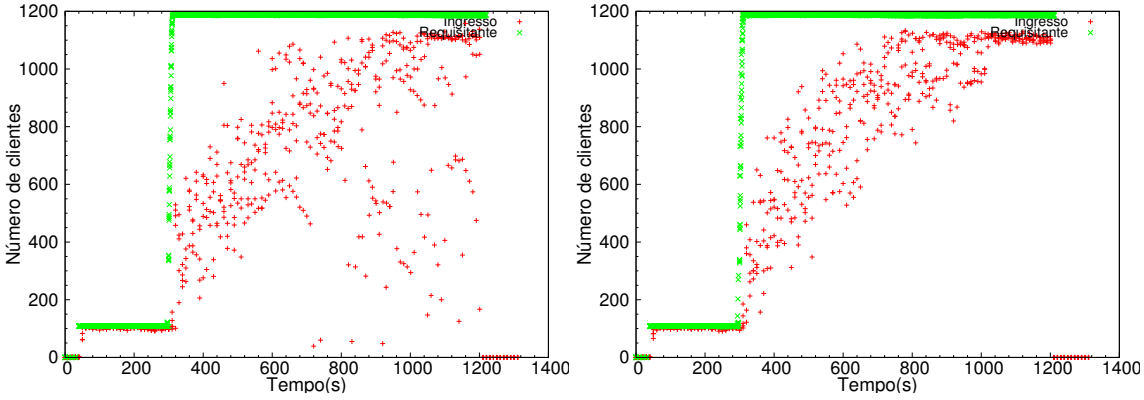
A Figura 9 possui os gráficos de rede e latência referentes ao experimento da Técnica de Redes Paralelas realizado no PlanetLab. Podemos observar, no gráfico de rede, o comportamento da rede, com os pares ingressando conforme ocorre a mesclagem das redes paralelas. Com esta técnica, foi possível tratar o *flash crowd*, evitando a queda dos pares. É possível também, nesta figura, observar o gráfico de latência da rede, que possui um pico no meio do experimento. Os resultados apresentados nestes gráficos são os resultados esperados e que tentamos simular em nossos experimentos usando nossa simulação do atraso.

Na Figura 10, temos os gráficos da rede dos experimentos realizados em nossa



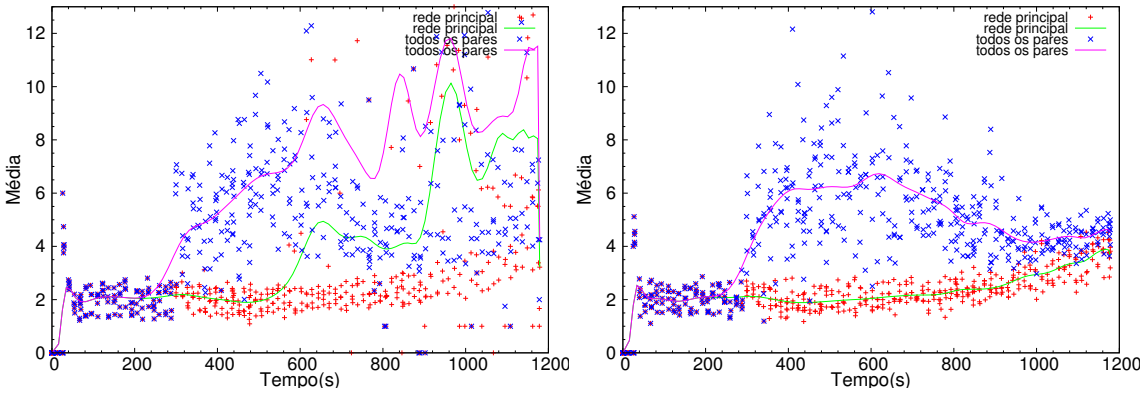
(a) Gráfico de Rede (b) PlanetLab

Figura 9. Experimento no PlanetLab (Técnica de Redes Paralelas)



(a) Método 1 (b) Método 2

Figura 10. Gráfico de Rede (Técnica de Redes Paralelas)



(a) Método 1 (b) Método 2

Figura 11. Gráfico de Latência (Técnica de Redes Paralelas)

rede local aplicando os métodos de atraso implementados. Observamos, no gráfico do Método 1, que conforme as redes paralelas foram se mesclando à rede principal e os pares ingressando, alguns pares começaram a cair, o que não deveria acontecer, já que esta técnica consegue tratar o *flash crowd* em um ambiente real. No Método 2 a rede

reage como esperado, os pares conseguem ingressar na rede, e a técnica consegue evitar a queda de pares.

Na Figura 11, temos os gráficos de latência dos experimentos realizados em nossa rede local utilizando os métodos de atraso implementados. O Método 1, devido à queda de pares no decorrer do experimento, obteve uma latência acima do esperado neste experimento. O Método 2 consegue uma maior aproximação do comportamento esperado, trazendo um pico de latência no meio do experimento.

6. Conclusão

Comparando ambos os métodos implementados em nosso trabalho, acreditamos que o Método 2, que atrasa apenas as mensagens de dados, trouxe uma maior aproximação dos resultados obtidos nos experimentos realizados em rede local em comparação com os experimentos realizados no PlanetLab.

Com os resultados dos nossos experimentos, é possível observar que o trabalho converge para uma simulação realista da latência em redes locais, mas ainda não é conclusivo. Os gráficos gerados em nossos experimentos possuíram similaridades ao que se espera de uma rede real, tal como o PlanetLab, mas ainda não há uma estimativa estatística do erro que nossa solução atinge.

Acreditamos que em trabalhos futuros, uma melhor configuração dos parâmetros, estudando e entendendo melhor cada um deles, ou a criação de novos parâmetros, pode retornar resultados mais convincentes e a utilização de um método estatístico para a avaliação dos resultados pode ajudar no julgamento dos mesmos.

Referências

Brown, M. A. (2006). Traffic control howto. *Guide to IP Layer Network*.

Chen, Z.; B. L.; Keung, G. Y. Y. H. L. C. . W. Y. (2009). How scalable could p2p live media streaming system be with the stringent time constraint? In *International Conferences on Communication, 1–5*. IEEE.

Chen, Y., Zhang, B., and Chen, C. (2011). Modeling and performance analysis of p2p live streaming systems under flash crowds. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE.

Chen, Y., Zhang, B., Chen, C., and Chiu, D. M. (2014). Performance modeling and evaluation of peer-to-peer live streaming systems under flash crowds. *IEEE/ACM Transactions on Networking (TON)*, 22(4):1106–1120.

Joao F. Oliveira, Italo Cunha, E. C. M. M. V. R. A. B. V. S. V. C. (2013). Can peer-to-peer live streaming systems coexist with free riders? *International Conference on Peer-to-Peer Computing*.

KUROSE, James F.; ROSS, K. W. (2010). *Redes de computadores e a internet: uma abordagem top-down*. Addison Wesley, Sao Paulo, 5 edition.

Li, B., Keung, G. Y., Xie, S., Liu, F., Sun, Y., and Yin, H. (2008). An empirical study of flash crowd dynamics in a p2p-based live video streaming system. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE.

Liu, F., Li, B., Zhong, L., and Li, B. (2009a). Understanding the flash crowd in p2p live video streaming systems. In *Packet Video Workshop, 2009. PV 2009. 17th International*, pages 1–10. IEEE.

Liu, F., Li, B., Zhong, L., Li, B., Jin, H., and Liao, X. (2012). Flash crowd in p2p live streaming systems: Fundamental characteristics and design implications. *IEEE Transactions on Parallel and Distributed Systems*, 23(7):1227–1239.

Liu, F., Li, B., Zhong, L., Li, B., and Niu, D. (2009b). How p2p streaming systems scale over time under a flash crowd? In *IPTPS*, volume 1, page 2.

Lobb, R. J., Couto da Silva, A. P., Leonardi, E., Mellia, M., and Meo, M. (2009). Adaptive Overlay Topology for Mesh-based P2P-TV Systems. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '09, pages 31–36, New York, NY, USA. ACM.

Miguel, E., Cunha, I., and Campos, S. (2016). Ingressos em Redes P2P para Video ao Vivo. In *SBRC 2016 - WP2P+ ()*.

Oliveira, J. F., Cunha, Í., Miguel, E. C., Rocha, M. V., Vieira, A. B., and Campos, S. V. (2013a). Can peer-to-peer live streaming systems coexist with free riders? In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–5. IEEE.

Oliveira, J. F., Viana, R. S., Vieira, A. B., Rocha, M. V., and Campos, S. V. (2013b). Tvpp: A research oriented p2p live streaming system. SBRC.

O'Reilly (2013). Primer on latency and bandwidth. `https://hpbn.co/primer-on-latency-and-bandwidth/`. Accessado em: 15 fevereiro 2017.

PlanetLab (2009). An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services. `http://www.planet-lab.org/`.

Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Lo Cigno, R., and Mellia, M. (2015). Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison. *IEEE/ACM Transactions on Networking (TON)*, 23(3):741–754.

Wu, H., Jiang, H., Liu, J., Sun, Y., Li, J., and Li, Z. (2011). How p2p live streaming systems scale quickly under a flash crowd? In *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, pages 1–8. IEEE.

Wu, H., Liu, J., Jiang, H., Sun, Y., Li, J., and Li, Z. (2012). Bandwidth-aware peer selection for p2p live streaming systems under flash crowds. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 360–367. IEEE.