Detecção de streamers em redes BitTorrent

Daniel V. C. da Silva, Antonio A. de A. Rocha

Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil

{dvasconcelos,arocha}@ic.uff.br

Abstract. *Many BitTorrent (BT) clients start use these networks as a video on demand service, using of the popularity and the enormity of the collection of media available. However, transforming the swarms into an on-demand media services can cause serious damage to the overall network performance, all types of users may experience degradation in quality of service, because those streamers clients modify the way to download determined in the protocol. In this paper, a spy Bittorrent client, developed to monitor exchanges of messages in the network is presented. Using the concepts of entropy, is defined a calculation to determine and classify the peers within a swarm. Experiments were made to detect the presence of streamers in public swarms and was also evaluated the impact of this type of user in swarms.*

Resumo. *Muitos clientes BitTorrent (BT) passaram permitir o uso da rede como serviço de mídia sob demanda, fazendo uso da sua popularidade e da enormidade do seu acervo. No entanto, esse uso das redes BT pode trazer sérios prejuízos ao desempenho geral da rede, todos os tipos de usuários podem experimentar queda na qualidade de serviço, isso porque para adaptar essas redes, os clientes modificam a forma de* download *determinada no protocolo. Nesse artigo é apresentado um cliente BitTorrent espião, desenvolvido para monitorar trocas de mensagens na rede. Usando os conceitos de entropia, é definida uma forma de cálculo para determinar e classificar os* peers *dentro de um enxame. Foram feitos experimentos para detectar a presença de* streamers *e também foi avaliado o impacto desse tipo de usuário nos enxames.*

1. Introdução

A melhora na velocidade de acesso à Internet, disponível para os usuários finais nos dias atuais, têm permitido a materialização de novos tipos de aplicação, bem como novas formas de interação com aplicações antigas. Uma prova disso é que, em um passado recente, o tráfego gerado por sistemas P2P (*peer-to-peer*) era responsável por uma fração significativa (mais da metade) de todo o trafego da Internet, sendo que desse montante estimava-se que em torno de 30-50% era proveniente de compartilhamentos de arquivos através do BitTorrent [Menasche et al. 2009]. De acordo com [BitTorrent Inc. 2014], em 2012 a soma mensal de usuários ativos dos *softwares* clientes uTorrent e BitTorrent ultrapassava 150 milhões. Um estudo sobre as diferentes categorias de conteúdo disponibilizados pelos usuários do BitTorrent demonstra que a grande maioria dos conteúdos (aproximadamente 80%) são de arquivos multimídia de filmes, músicas, séries de tv e outros tipos de vídeo ou áudio [Zhang et al. 2010].

Mais recentemente, o aumento da largura de banda da rede de acesso dos usuários na Internet contribuiu também para intensificar o uso das aplicações de mídias contínuas (de áudio e vídeo), também chamadas de aplicações *streaming* multimídia. As taxas mais altas de *download* passaram a permitir que os usuários iniciem a reprodução da mídia com a transferência do arquivo ainda em andamento, isso sem que hajam interrupções ou que pelo menos essas interrupções sejam toleráveis. Assim, dentre os usuários interessados em conteúdos multimídia, essas aplicações passaram a se tornar cada vez mais populares, em detrimento das soluções de compartilhamento de arquivo P2P, como o BitTorrent. Usuários, cada vez mais exigentes, agora desejam (e já podem) iniciar a reprodução da mídia (quase que) instantaneamente e não mais esperar pela conclusão do *download*.

Atualmente, o volume de tráfego gerado pelas aplicações de *streaming* (que incluem NetFlix, Hulu, Youtube, Deezer, Spotify, dentre outras) já é superior ao das aplicações P2P ou de qualquer outro tipo de sistema em uso na Internet. Estudos apresentados em [Cisco 2014] apontam que aproximadamente 66% de todo o tráfego da Internet hoje é oriundo das aplicações de *streaming* de vídeo. Nesse mesmo estudo, os autores prevêem que em 2018 essa fração possa alcançar valores entre 80-90%.

Apesar da crescente popularidade das aplicações de *streaming*, os seus usuários sofrem de dois problemas fundamentias: (i) a falta de escalabilidade inerente à arquitetura cliente/servidor; e, (ii) o acervo limitado de algumas dessas aplicações.

Para lidar com a falta de escalabilidade e, ao mesmo tempo, atender a demanda dos usuários de não ter que esperar a conclusão do *download* de todo o conteúdo para iniciar a reprodução da mídia, sistemas P2PTV podem ser utilizadas. Assim como os sistemas de compratilhamento de arquivo, as aplicações P2PTV surgiram como uma solução escalável para transmissão de vídeo, onde participantes não só assistem a mídia enquanto recuperam o conteúdo, mas também ajudam na transmissão do dados para os demais participantes da rede. Para isso, canais são criados para transmitir o conteúdos de mídia previamente armazenados ou capturados em tempo real (a partir de um sinal de televisão, por exemplo) sem sobrecarregar a máquina do publicador.

Nos últimos anos, uma gama de aplicações P2PTV foram criadas (PPLive, SopCast, CoolStreaming e StreamerP2P são apenas algumas delas). Embora todas elas apresentem soluções para o problema de escalabilidade inerente às arquiteturas com um único servidor central, de alguma forma elas não se tornaram tão populares. Com isso, assim como o NetFlix, Youtube e Hulu, as aplicações P2PTV sofrem com o problema do acervo limitado. Estudos mostram que todas as redes P2PTV somam juntas "apenas" poucas dezenas de milhões de usuários [Hei et al. 2007], sendo a maioria deles originários da China. Por outro lado, apesar da redução significativa do volume de tráfego gerado pelas aplicações BitTorrent nos últimos anos, é indiscutível que ela ainda oferece aos seus usuários um rico acervo de conteúdos de áudio e vídeo.

Nesse cenário, uma abordagem que tem se mostrado promissora é fazer com que os programas clientes BitTorrent explorem a grande popularidade e alta disponibilidade, adaptando essas aplicações para conseguirem operar em modo *streaming*. Para isso, os aplicativos clientes devem alterar as suas política de requisição dos blocos, passando a solicitar as partes do conteúdo de forma sequencial, ao invés de obedecer a política definida pelo protocolo (no caso, a prioridade para o bloco mais raro).

A discussão sobre a possibilidade de co-existência de *peers*, com diferentes políticas de seleção dos blocos em uma mesma rede, não é uma grande novidade [Shah and Paris 2007, Vlavianos et al. 2006, Mendonça and Leao 2012]. Algumas ferramentas já há algum tempo fazem uso desse conceito (conhecidos como *view as you download*). Porém, apenas mais recentemente essa função tem se tornado comum entre as aplicações clientes e mais popular entre os usuários.

Os estudos que tratam da possibilidade de explorar essa capacidade das redes BitTorrent de atender a usuários *streamers*, de modo geral, consideram a ótica dos *peers* interessados na reprodução antecipada e do carregamento sequencial, avaliando apenas fatores como atraso, tempo de reprodução, número e/ou tamanho das interrupções. No entanto, a mudança no mecanismo de seleção dos blocos pode interferir no desempenho da rede como um todo [Li and Zhang 2013]. Estudos apresentados em [Vlavianos et al. 2006, Parvez et al. 2008, Cohen 2003] demonstram que a política de seleção por trechos mais raro levam ao aumento da disponibilidade do conteúdo na rede.

Pensando no funcionamento e desempenho da rede BitTorrent, de uma forma mais ampla, existem portanto algumas questões ainda em aberto, são elas: (i) ainda não há um mecanismo que permita identificar de forma rápida e eficiente usuários operando na função *view as you download* dentro das redes BitTorrent; (ii) não há evidências reais que demonstrem o impacto da presença desse tipo de usuário no desempenho global do sistema; e, (iii) não se sabe ao certo se há hoje um número significativo de usuários conectados às redes convencionais do BitTorrent operando em modo *streaming*. Para responder tais questões, esse trabalho tem como contribuições específicas:

- O desenvolvimento de uma arquitetura de monitoramento da rede BitTorrent, aliada a um método para classificar os usuários conectados a ela como regulares (requisição de blocos mais raros) ou *streamers* (requisição sequencial);
- A análise dos dados coletados em um longo período de monitoramento pela a arquitatura desenvolvida, com o objetivo identificar se há, de fato, uma fração significativa de usuários *streamers* nas redes públicas BitTorrent;
- A realização de experimentos reais, em redes privadas BitTorrent, para mensurar o potencial impacto no desempenho geral do sistema com a presença de clientes operando no modo "*view as you download*".

A organização deste trabalho é feita da seguinte forma. Na Seção 2 é apresentada uma breve revisão teórica, que abordará os conceitos de P2P, *streaming* e entropia. A Seção 3 descreve a arquitetura de monitoramento desenvolvida e os resultados obtidos com a metodologia adotada para detectar a presença de *streamers* em redes BitTorrent. O impacto no desempenho do sistema com a presença de usuários "*view as you download*" é analisada na Seção 4. Por fim, a Seção 5 apresenta as conclusões deste trabalho.

2. Revisão Teórica

Nessa seção serão apresentados os conceitos fundamentais usados na proposta da arquitetura de monitoramento e análises realizadas nesse artigo.

2.1. BitTorrent: do compartilhamento de arquivos ao streaming

As redes P2P se tornaram a principal alternativa à arquitetura cliente-servidor na distribuição de dados. Isso por terem como característica fundamental o fato de que, cada

usuário (nó, *peer* ou par) se comporta simultâneamente tanto como cliente (recebendo os dados) quanto como servidor (servindo os dados já recebidos). Embora o conceito desse tipo de sistema seja bem antigo e usado para outras finalidades que vão muito além do simples compartilhamento de arquivo, nenhuma outra aplicação se tornou tão popular quanto a que foi criada por Bram Cohen [Cohen 2014].

As redes de compartilhamento BitTorrent são usualmente chamadas de "torrent". Nessas redes, os *peers* são denominados *leechers*, se ainda estão baixando conteúdo, ou *seeders*, se já concluíram o *download* por completo. Juntos, os *peers* formam um enxame (do termo em inglês, *swarms*) e contam com uma espécie de coordenador central (conhecido como *Tracker*). A existência do coordenador possibilita que os *peers* conheçam uns aos outros e se conectem para, então, realizar a troca de dados. Para isso, os *peers* devem trocar entre eles os seus respecitvos mapas de *bits* (*bitmaps*) informando aos seus vizinhos os trechos do conteúdo já recuperados.

A disseminação da informação é feita obedecendo uma série de mecanismos definidos no protocolo BitTorrent. Algumas delas são: (i) a política de prioridade, geralmente é implementada de forma que se escolhe o bloco mais raro entre os vizinhos; (ii) a política de compartilhamento "toma-lá-dá-cá", que tenta fomentar a reciprocidade direta entre os *peers* no sistema; e, (iii) a política de compartilhamento otimista, que possibilita a novos usuários receberem inicialmente alguma parte do conteúdo, mesmo sem terem compartilhado nada, para adquirirem algum poder de barganha com os seus vizinhos.

Em [Vlavianos et al. 2006], mostrou-se que dois desses mecanismos do BitTorrent se tornam inadequados quando a aplicação possui a necessidade de uma ordem cronológica no recebimento das partes do conteúdo e/ou uma restrição de tempo na recepção desses dados, como é o caso das aplicações de streaming. A política de prioridade aos pedaços mais raros impõe um modo, de certa forma, aleatório ao recebimento das partes da mídia. Essa mecânica é um problema para aplicações de streaming, uma vez que as partes do conteúdo serão executadas (e assim deveriam também ser recebidas) em sequência. Já a política de compartilhamento "toma-lá-dá-cá" pode impedir que, *peers* com pouca capacidade para contribuir com o sistema (ou seja, taxa de *upload* limitada), recebam os dados a uma taxa satisfatória para a reprodução contínua da mídia.

A Figura 1 ajuda a compreender a necessidade de alteração do protocolo no cliente BitTorrent, para que os usuários possam usufruir da popularidade dos enxames e assistirem enquanto ainda carregam o conteúdo. Essa figura representa a abstração de um enxame BitTorrent com um tracker e três *peers*. Na ilustração, é apresentada uma representação do *bitmap* associado a cada *peer*, onde os blocos de cor cinza indicam os que já foram recuperados e os de cor branca os blocos que ainda estão por receber. Nota-se ainda que, o *peer* com todos os blocos em cinza é um *seeder*, enquanto o *peer* cujo os blocos cinzas são os iniciais é um *leecher streamer* e o outro um *leecher* regular. De acordo com a abstração da figura, o *peer* regular deve escolher aleatoriamente entre o sexto e o sétimo blocos, uma vez que esses blocos são igualmente raros no enxame. No entanto, o *peer streamer* deve necessariamente requisitar o quinto bloco.

Para viabilizar a reprodução de um streaming de vídeo transmitido através do sistema BitTorrent, [Shah and Paris 2007] propõe que sejam alteradas as políticas de seleção dos pedaços e de reciprocidade de transmissão. No entanto, o que tem se visto na rea-

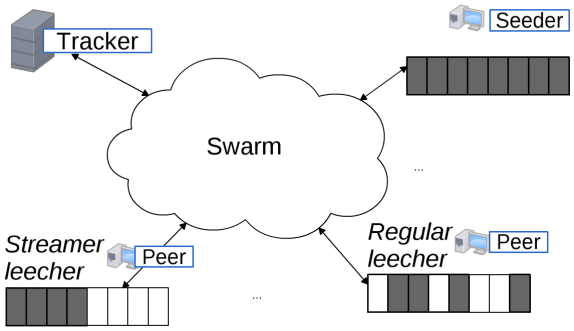

Figura 1: Comportamento dos *peers* aleatórios e sequenciais em um enxame

lidade é que cada aplicação implementa as alterações que seus desenvolvedores acham mais adequadas para a realização do streaming. Nos dias atuais, uma série de clientes BitTorrent têm disponibilizado, integrado à ferramenta, algum tocador de mídia. Isso permite aos usuários assistirem (por completo ou parcialmente) o conteúdo, enquanto este está sendo recuperado. Alguns dos clientes que oferecem essa função são: Vuze, Transmission, Thunder (Xunlei), qBitTorrent, KTorrent, Deluge, Bitcomet e Utorrent. Porém, algumas aplicações vão muito além da simples disponibilização da função "*view as you download*". PopCorn Time e Joker, por exemplo, são dois sistemas que, não só permitem a realização do streaming e execução do conteúdo em um tocador de vídeo, mas tornam todo processo absolutamente simples e totalmente transparente para os seus usuários.

2.2. Entropia Clássica e de Permutação

O conceito de entropia foi adaptado e generalizado pelo matemático americano Claude Shannon em [Shannon 2001] e tem encontrado aplicações nas mais diversas áreas do conhecimento. No contexto de teoria da comunicação (informação), essa teoria é muitas vezes chamada de entropia de Shannon. Em seu trabalho, Shannon define entropia como uma medida de incerteza associada a uma variável aleatória.

Para compreender o conceito, considere um conjunto finito de elementos de uma variável aleatória (digamos um conjunto M, de ordem $n = |M|$), cujo as probabilidades de ocorrência de cada elemento são conhecidas e definidas pela função distribuição de probabilidade discreta $P = \{p_i : i = 1, ..., n\}$. A medida proposta por Shanon, denominada entropia, pode ser expressada da forma:

$$H(M) = -\sum_{1}^{n} p_i \, log \, p_i, \tag{1}$$

onde, sendo considerada a base 2 no logaritmo, a unidade da medida é em bits. Além disso, o limite inferior dessa medida é dado pela máxima certeza sobre a ocorrência do evento, enquanto que o limite superior ocorre em caso de máxima incerteza, ou seja, quando os eventos são equiprováveis. Assim, $H(M) = [0, log_2 \, n]$.

Embora encontre inumeras aplicações, a entropia de Shannon falha por não capturar a relação cronológia de ocorrência dos eventos. Por exemplo, sejam $M_1 = (0, 0, 1, 1)$ e $M_2 = (0, 1, 0, 1)$ dois conjuntos finitos de elementos. Pela definição de Shannon, temos

que $H(M_1) = H(M_2)$. Portanto, a estimativa falha na captura da relação cronologica existente entre os elementos de M. O cálculo da entropia de Shannon também se torna mais complexo à medida que o número de elementos de M aumenta muito.

As limitações impostas ao modelo clássico de entropia são, de certa forma, tratados com uma medida apresentada em [Bandt and Pompe 2002] e revisada por [Riedl et al. 2013]. Denominada Entropia de Permutação, esta medida leva em consideração a ocorrência cronológica dos elementos de um conjunto, comparando os valores adjacentes em uma série temporal.

O método pode ser definido da seguinte forma. Seja s uma determinada ocorrência de uma série de elementos $M = (m_i : i = 1, ..., n)$. Seja v um vetor formado pelos k-ésimos subsequentes a s dessa série. Assim, $v \rightarrow (m_s, m_{s+1}, ..., m_{s+r(k-2)}, m_{s+r(k-1)})$, onde k é chamada a ordem da permutação e determina o tamanho do vetor a ser considerado. O vetor v é associado a um padrão ordinal, denominado permutação $\pi = (r0r1...r(k-1))$ de $(01...k)$, que satisfaz $m_{s+r0} \leq m_{s+r1} \leq ... \leq m_{s+r(k-2)} \leq m_{s+r(k-1)}$. Portanto, cada valor de v é ordenado de forma crescente e o padrão ordinal π é determinado com os deslocamentos dos valores permutados.

Para uma melhor compreensão, considere a sequência $M = (9, 32, 50, 13, 99, 5)$. Para k=3, o vetor correspondente à ocorrência de $s = 1$ é $v = (9, 32, 50)$, sendo ele próprio a sua versão ordenada e, consequentemente, a permutação $\pi = (012)$. Já para a ocorrência de $s = 2$, sendo o vetor $v = (32, 50, 13)$, a ordenação crescente (13,32,50), e resultando em uma permutação $\pi = (201)$.

Note que são $k!$ possíveis combinações de permutações. Seja, então, π_j as frequencias relativas associadas aos j possíveis padrões de permutações, logo $j = 1, ..., k!$. A entropia de permutação de uma sequência M é dada por:

$$H_P(M) = -\sum_{1}^{k!} \pi_j \; log_2 \; p_j, \tag{2}$$

onde $H_P(M) \in [0, log_2 \; k!]$. Logo, a entropia de permutação normalizada (h_P) pode ser obtida por:

$$h_P(M) = -\frac{1}{log_2 \; k!} \sum_{1}^{k!} \pi_j \; log_2 \; p_j. \tag{3}$$

3. Eye of the Swarm

O desenvolvimento de meios que permitam a detecção de aplicações "burlando" o modo padrão de operação do sistema é de suma importância, não só para ajudar a compreender a realidade atual da rede BitTorrent, mas também para auxiliar na criação de mecanismo que permitam inibir tais usuários, se assim for desejado. A questão que se coloca é, será que é possível desenvolver uma técnica não intrusiva (i.e., sem interferir no funcionamento regular do enxame) que permita identificar a existência deste tipo de usuários em um enxame BitTorrent?

Nesta seção, será apresentada uma arquitetura de monitoramento desenvolvida e a metodologia proposta que responde positivamente a questão colocada acima. Dessa forma, é possível analisar o funcionamento e detectar a existência de aplicações BitTorrent operando no modo *streaming* em exames reais.

3.1. Ferramenta Espiã e Metodologia Desenvolvida

Para possibilitar o desenvolvimento da arquitetura de monitoramento e a implementação da metodologia para identificação de *streamers*, uma versão da ferramenta Transmission [TransmissionBT 2014] foi adaptada. Da aplicação foram removidas todas as funções de troca de dados de conteúdo, mantendo apenas as o envio e recebimento dos dados de controle previstos originalmente no protocolo.

O "espião" BitTorrent adaptado funciona como um cliente regular, é visto pelo enxame como tal, porém ele não faz qualquer requisição de conteúdo e registra toda a evolução do *bitmap* (ou *bitfield*) dos *peers* conectados ao enxame. Enquanto se conectar ao maior número possível de *peers* da rede, a ferramenta requisita o mapa de bits de todos os vizinhos e, em seguida, acompanha a evolução dos *bitmaps* através do registro das mensagens de *have* (que são mensagens enviadas por todos os *peers*, para todos os seus vizinhos, informando a conclusão da recuperação de mais um pedaço do conteúdo). Essas informações serão a base da metodologia apresentada a seguir para a classificação dos *peers*.

A partir da evolução do *bitfield* registrado para cada cliente, a técnica proposta usa o conceito de entropia de permutação para determinar se tal aplicação recuperou os trechos de forma sequencial (*streaming*) ou aleatória (a política do bloco mais raro tende a gerar uma recuperação mais uniformemente distribuída). Para isso, a cada cliente registrado com um número suficientemente grande de mensagens de *have* (o experimento descartou dados dos peers com menos de 20 mensagens), uma série M é formada a partir da ordem que foi recebida a informação sobre a recuperação dos pedaços. A partir da sequencia M e para um determinado valor de k, $h_P(M)$ é computado. Valores altos $h_P(M)$ indicam a aleatoriedade da recuperação dos pacotes, enquanto que valores baixos sugerem um modo sequencial de recuperação.

3.2. Validação e Refinamento do Método Proposto

Para validar o método proposto, foram testados os clientes BitTorrent mais populares na Internet. Para cada um desses clientes (um total de nove), foram feitas baterias considerando os modos de *download* aleatório e sequencial (se existisse). Os cenários experimentados foram tanto em ambientes controlados como em ambientes públicos. No primeiro, todas as máquinas encontravam-se em uma rede local e eram controladas . Já no teste em ambiente real, a máquina espiã se conectava a um enxame público, assim como a máquina a ser identificada. O registro da evolução do mapa de bits era feito para a máquina alvo, mas também para todas das demais máquinas encontradas no exame público utilizado.

De posse dos registros de troca de mensagem realizados nas baterias do experimento, considerando ambiente controlado e ambiente real público, foram calculados os valores de entropia por permutação dos programas clientes BitTorrent funcionando em modo aleatório e em modo sequencial. Inicialmente, para esse cálculo foram considerados os valores de k = 2, 3 e 4. No entanto, os resultados demonstram que o valor de k = 2 já é suficiente para a estimativa, além de exigir menos complexidade computacional para ser estimado.

Um dos resultados obtidos de experimentos realizados com o cliente Bitcomet estão ilustrados nas Figuras 2 (a) e (b). Os gráficos indicam a evolução dos valores computados para $h_P(M)$ para $k = 2,\ 3\ e\ 4$ para cada nova mensagem de *have* recebida pelo

espião. Cabe ressaltar que no gráfico é ilustrado uma curva com diversas estimativas de $h(M)$, mas a metodologia prevê a estimativa de apenas um valor para todas as mensagens de "*have*". Neste caso, o objetivo era mostrar a evolução da medida. Pelo gráfico é possível notar que, a partir de um certo tamanho da sequência de *have*, existe uma clara diferença entre os valores computados para $h_P(M)$ sequencial e aleatório, independente do valor de k. (Note que por uma questão de simplicidade, as curvas são identificadas como aleatória ou *streaming* $H(k)$, para $k = 2,\ 3\ e\ 4$). O trecho inicial apresenta a estratégia desse cliente de priorizar o inicio e o fim do arquivo, independente do modo de download.

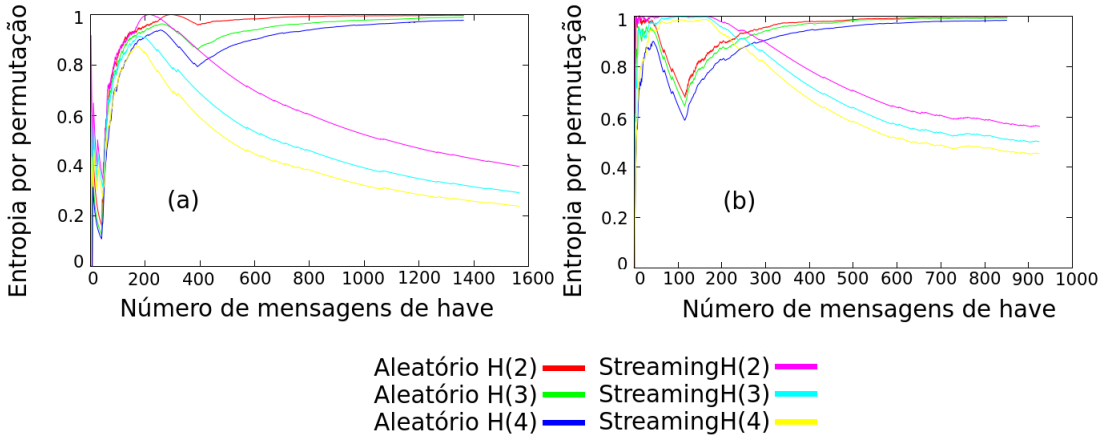

Figura 2: Evolução da entropia por permutação do cliente BitComet em (a) ambiente controlado e (b) em enxame público

A Tabela 1 apresenta todos os valores finais computados. Nela é possível verificar que existe uma grande diferença entre os valores, em ambas estratégias de *download* e entre os clientes. Isso reflete as diferentes implementações adotadas pelos desenvolvedores das aplicações.

Tabela 1: Entropia dos clientes BitTorrent em ambiente controlado

Cliente	Modo	Nº Msg	H(2)	H(3)	H(4)
Bitcomet	Aleatório	163	0.9982	0.9953	0.9856
Bitcomet	Sequencial	1569	0.3966	0.2924	0.2385
BitLord	Aleatório	730	0.9991	0.9969	0.9956
BitLord	Sequencial	604	0	0	0
KTorrent	Aleatório	490	0.9998	0.9985	0.9964
KTorrent	Sequencial	973	0.0465	0.0374	0.0320
qBitTorrent	Aleatório	1550	0.9999	0.9996	0.9988
qBitTorrent	Sequencial	726	0.2422	0.2005	0.1705
Vuze	Aleatório	1570	0.5897	0.5447	0.5039
Vuze	Sequencial	1570	0.5539	0.5045	0.4619
Xunlei	Aleatório	3139	0.4338	0.3986	0.3749
Xunlei	Sequencial	1568	0.1975	0.1645	0.1483

Muito embora seja clara a existência da diferença entre a entropia calculada para

o caso sequencial e o caso aleatório, não há um *threshold* comum entre todas as diferentes versão que possa ser usado para determinar a classificação dos *peers*. Por isso, a solução adotada passa por atribuir a cada tipo de aplicação estudada um limiar diferente. Esse adequação pode ser feita, pois nas mensagens de controle enviadas pelos clientes é indicado nome da aplicação remota. Certamente, a razão pela qual diferentes aplicações possuem variações significativas no valor computado da entropia de permutação está relacionado às opções feitas pelo grupo de desenvolvedores na implementação das ferramentas.

3.3. Arquitetura de Monitoramento e Experimentos em Larga Escala

Com o objetivo de identificar a existencia (e qual a fração) de usuários operando em modo *streaming* em enxames públicos, experimentos em larga escala foram realizados. A arquitetura de monitoramento, resumida pela Figura 3, operou por noventa dias consecutivos, da seguinte forma: (i) a cada duas horas eram obtidos os links dos 100 enxames de vídeo mais populares do PirateBay; (ii) para cada um desses 100 enxames, um processo espião era disparado para realizar o monitoramento (coleta de dados) de todos os *peers* conectados ao enxame; (iii) Ao final do período de duas horas, os logs de dados de controle eram armazenados em uma base de dados, os demais registros e metadados apagados do monitor, e o processo de monitoramento voltava para a etapa (i).

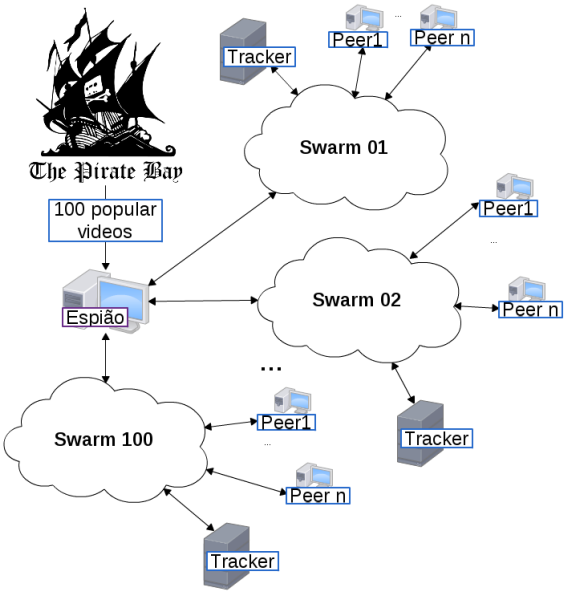

Figura 3: Esquema do experimento em enxames populares do PirateBay

O espião se conectou a mais de 1 milhão e meio de *peers*, de um total de mais de 100 mil enxames. A coleta resultou em uma base com aproximadamente 40 gigabytes de registro de mensagens.

Para todos os *peers* monitorados por uma fração de tempo suficientemente grande, foi calculado a entropia de permutação. A Tabela 2 apresenta um resumo das coletas e classificações realizadas pela arquitetura de monitoramento por ferramenta cliente. Dos números mostrados, nota-se que dos mais de 1.5 milhões de *peers*, apenas aproximadamente 7 mil deles (cerca de 0,53%) foram classificados como *streamers*.

Tabela 2: Avaliação da existência de *streamers* em duas etapas

Cliente	Nº Peers	Streamers	%
Vuze	145285	657	0,45%
Transmission	202439	1	0,00%
Thunder (Xunlei)	2346	376	16,03%
qBitTorrent	7834	9	0,11%
KTorrent	62	0	0,00%
Deluge	28246	0	0,00%
BitTorrent	97761	687	0,70%
BitLord	181	0	0,00%
Bitcomet	7110	29	0,41%
Utorrent	764185	5212	0,68%
IL50	5791	0	0,00%
JS09	45	0	0,00%

3.4. Monitoramento de Enxames Específicos

Não é incoerente imaginar que usuários sequenciais são pessoas que não desejam esperar para desfrutar do conteúdo. Então, uma hipótese é que esse tipo de indivíduo acessa às redes assim que uma determinada mídia de interesse é disponibilizada na Internet, e possivelmente faria uso de um modo de operação "*view as you download*". Para verificar essa hipótese, a ferramenta espiã foi também configurada para monitorar por 24 horas, os enxames dos últimos 5 episódios da quarta temporada da série Walking Dead e todos os episódios da quarta temporada de Game of Thrones, assim que esses vídeos aparecessem na lista de mais populares do PirateBay.

No caso do Game of Thrones, o espião se conectou a 110 mil *peers*, e dentre esses 3600 foram classificados como *streamers*, o que representa 3,28% do total de usuários monitorados. Embora não seja um número muito alto, é um aumento considerável, quando comparado ao valor inicial de 1,10% do experimento realizado com todos os enxames de vídeo populares do PirateBay. Já o experimento com a série Walking Dead, o espião se conectou a 13143 *peers*, e dentre esses apenas 57 possuem entropia por permutação que os classifiquem como *streamers*, o que representa 0,43% dos *peers*. Um valor muito próximo ao do experimento em larga escala.

Um resultado que merece destaque é ilustrado na Figura 4. Nela são mostrados os valores ordenados da entropia de permutação dos *peers*, isso para um tipo específico de cliente BitTorrent (Xunlei Thunder). O comportamento do gráfico ilustra claramente a separação de usuários *streamers* dos usuários em modo aleatório. Nota-se que essa ferramenta em especial, para essa série, é muito comum encontrarmos usuários operando no modo *"view as you donwload"*. Neste caso são mais de 1000 *streamers*, de um total aproximado de 1600 *peers* (63%).

4. Análise do Impacto de *Streamer*s no Desempenho dos Enxames

Bram Cohen defende argumenta em [Cohen 2003] que a robustez das redes BitTorrent é uma consequencia direta da política apropriada de seleção de blocos mais raros. Porém, as discussões são superficiais e, nem tão pouco, o autor apresenta evidências reais que

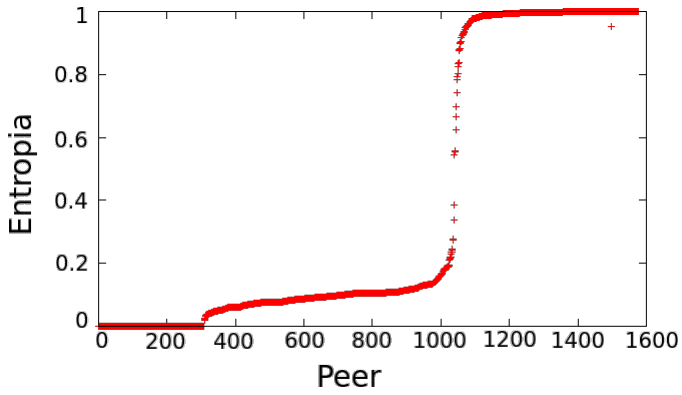

Figura 4: Entropia por permutação de clientes Xunlei (Game of Thrones)

garantam essa afirmação. Nessa seção serão apresentados experimentos feitos em laboratório, considerando diversos cenários. O objetivo é entender se de fato a existência de usuários *streamers* interferem no desempenho global dos enxames. E, se sim, a partir de que fração de usuários na rede isso se torna mais significativo. Para isso, foi considerado o tempo de *download* como principal medida de desempenho do enxame.

Foram definidos três diferentes cenários, todos com 0%, 10%, 20%, 30%, 40% e 50% de usuários em modo sequencial e *peers* com velocidade limitada a 300kB. Todos os *peers* saem do enxame assim que concluírem o seus *download*. Esses três conjuntos de experimentos diferem fundalmentalmente entre eles pelo padrão de entrada considerado para os *peers* e o tamanho da vizinhança, como descritos a seguir:

- Entrada dos *peers leechers* em rajada, sem restrições quanto ao número de conexões simultâneas que os *peers* podem estabelecer.
- Entrada dos *peers leechers* em rajada, sendo que o tamanho da vizinhança de cada *peer* foi limitada a 10 vizinhos.
- Entrada dos *peers leechers* a cada 10 segundos, com o tamanho da vizinhança de cada *peer* limitada a 10 vizinhos.

A Figura 5 ilustra a média dos valores da sessão com vizinhança ilimitada e entrada simultânea. Ela apresenta, de forma direta, o comportamento do tempo de *download* durante as seis baterias. No gráfico é possível identificar uma redução da média do tempo de download para a bateria 10%, porém existe um aumento no tempo total. A bateria de 20% tem um melhor desempenho geral, pois embora sua média e moda sejam um pouco maiores que a da bateria anterior, o tempo total do experimento é menor que a metade de todos os outros. E por fim, nas baterias a com 30% ou mais, existe um clara piora nas métricas de tempo.

A Figura 6 apresenta a média dos valores da sessão com vizinhança limitada e entrada simultânea. Embora as curvas sejam mais suaves, fica claro que o comportamento observado na configuração anterior e apresentado na Figura 5 se repete nessa configuração. É perceptível um aumento do tempo total na bateria a 10%, e embora não exista uma redução da média do tempo a 10%, ela também não aumenta muito, comparativamente. Também percebe-se que a bateria de 20% tem um melhor desempenho geral. Já nas baterias a com 30% ou mais, existe um clara piora nas métricas de tempo, embora não muito direta em 30%.

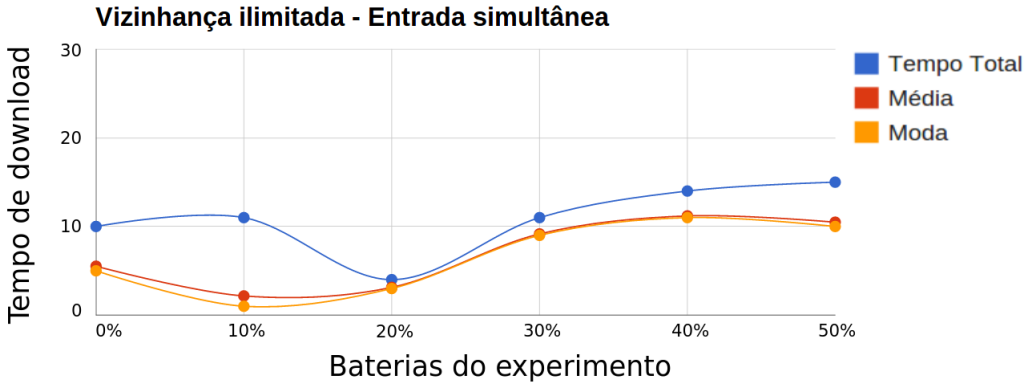

Figura 5: Desempenho do enxame com vizinhança ilimitada e entrada simultânea

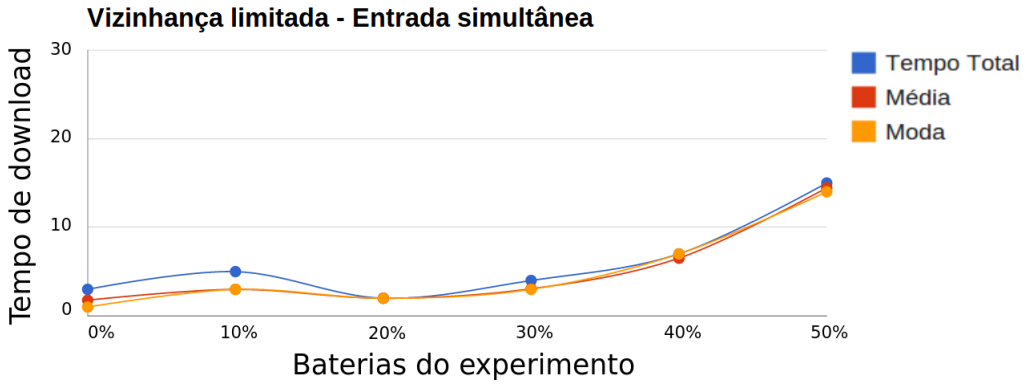

Figura 6: Desempenho do enxame testado em laboratório com vizinhança limitada e entrada simultânea

No exemplo de sessão com vizinhança limitada e entrada dos *peers* a cada 10 segundos (apresentada pela Figura 7) é possível identificar um comportamento um pouco diferente dos resultados anteriores. Nessa configuração, a bateria de 10% tem um resultado pior que as de 0%, 20%. 30% e 40%, e a bateria a 50% continua apresentando o pior desempenho geral.

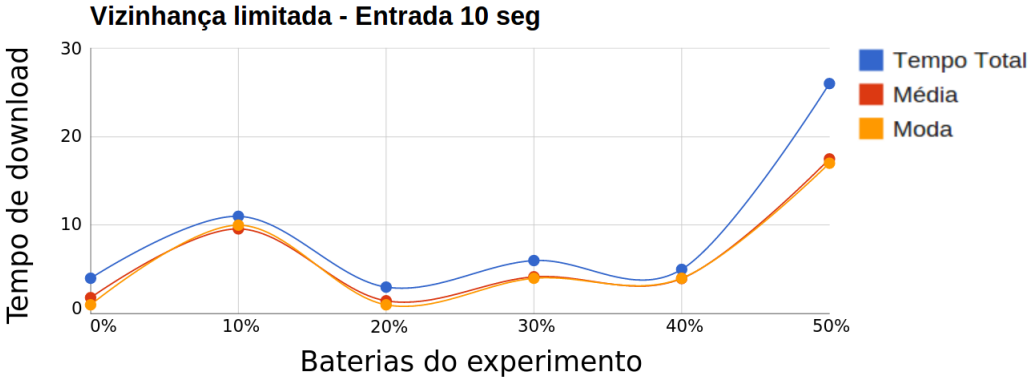

Figura 7: Desempenho do enxame testado em laboratório com vizinhança ilimitada e entrada a cada 10 segundos

Considerando as diferentes configurações de experimento, é possível afirmar que existe sim uma piora no tempo de *download* no enxame, seja ele individual ou total, à medida que se cresce o número de usuários operando em modo sequencial nessas redes. Em casos extremos, a piora no desempenho visto em uma bateria sem a presença de *streamers* se comparado a uma com metade dos *peers* operando em modo sequencial chega a ser 17 vezes maior, e o tempo total da bateria chegou a ser 6 vezes pior com 50% dos *peers* em modo *streaming*. Esse trabalho não apresenta intervalo de confiança pois os dados mostram pequena variância.

Por outro lado, os resultados apresentados também sugerem que, para alguns parâmetros específicos, a presença desses usuários pode ser irrelevante ou mesmo até bem vinda, desde que não ultrapasse um limite definido pela configuração do enxame em questão.

5. Conclusões

A adaptação das aplicações BitTorrent para uso de mídia sob demanda tem se apresentado cada vez mais frequente. Só no último ano duas novas ferramentas foram apresentadas na Internet com grande repercursão. Sem dúvida, esse movimento foi motivado pelo vasto acervo de conteúdo existente no BitTorrent, somado ao alto número de usuários desse modelo de aplicação. Além, é claro, de um aumento da demanda de usuários por soluções de vídeo sobre demanda.

A questão fundamental que este trabalho tenta responder são: Será que existe hoje um número significativo de usuários *streamers*? Será que é possível identificar em tempo real a existência desse tipo de usuário em enxames reais? A presença desses usuários burlando o sistema pode afetar o desempenho global da aplicação?

Esse trabalho, usando como base os conceitos de entropia por permutação, propos uma metodologia que permita clasisficar os usuários como *streamers* ou aleatórias. Além disso, uma arquitetura de monitoramento desenvolvida permitiu realizar experimentos em larga escala e identificar que, embora não muito alta, já há uma fração relevante de usuários *streamers* usando as redes BitTorrent para baixar conteúdos de forma sequencial. Por fim, experimentos realizados em laboratório permitiram verificar que existe valores aceitáveis da fração de usuários sequenciais dentro dos enxames, sem que haja degradação da qualidade experimentada pelos usuários. Porém, para um número elevado (acima de 30%), em geral há uma perda significativa do desempenho para todos no sistema.

Esse trabalho aponta, como possibilidades de estudos futuros, a ampliação da análise do impacto de usuários sequenciais nos enxames. Mesmo com os experimentos realizados para avaliar o impacto não se foi capaz de determinar alguns comportamentos da rede, como por exemplo a degradação após os 20%.

Referências

Bandt, C. and Pompe, B. (2002). Permutation entropy: a natural complexity measure for time series. *Physical Review Letters*, 88(17):174102.

BitTorrent Inc. ((acessado em dezembro de 2014)). Bittorrent and utorrent software surpass 150 million user milestone; announce new consumer electronics partnerships. http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users.

Cisco ((acessado em dezembro de 2014)). Cisco visual networking index: Forecast and methodology, 20132018. "http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf".

Cohen, B. (2003). Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72.

Cohen, B. ((acessado em agosto de 2014)). The bittorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html.

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W. (2007). A measurement study of a large-scale p2p iptv system. *Multimedia, IEEE Transactions on*, 9(8):1672–1687.

Li, Z. and Zhang, T. (2013). Understanding the quality-of-service of ubiquitous bittorrent-like media streaming. *Greenorbs*.

Menasche, D., Rocha, A., Li, B., Towsley, D., and Venkataramani, A. (2009). Content availability and bundling in swarming systems. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 121–132. ACM.

Mendonça, G. G. and Leao, R. M. (2012). Btstream–um ambiente para desenvolvimento e teste de aplicaç oes de streaming p2p. *XXX SBRC, Salao de Ferramentas, Ouro Preto, MG, Brazil*.

Parvez, N., Williamson, C., Mahanti, A., and Carlsson, N. (2008). Analysis of bittorrent-like protocols for on-demand stored media streaming. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):301–312.

Riedl, M., Müller, A., and Wessel, N. (2013). Practical considerations of permutation entropy. *The European Physical Journal Special Topics*, 222(2):249–262.

Shah, P. and Paris, J.-F. (2007). Peer-to-peer multimedia streaming using bittorrent. In *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pages 340–347. IEEE.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.

TransmissionBT ((acessado em dezembro de 2014)). Transmission. http://www.transmissionbt.com/.

Vlavianos, A., Iliofotou, M., and Faloutsos, M. (2006). Bitos: Enhancing bittorrent for supporting streaming applications. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE.

Zhang, C., Dhungel, P., Wu, D., and Ross, K. W. (2010). Unraveling the BitTorrent ecosystem. *IEEE Transactions on Parallel Distributed Systems*, 22(7):1164–1177.