Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Informática

SELEÇÃO DE PARES BASEADA EM QOE PARA TRANSMISSÃO DE VÍDEO EM REDES P2P BITTORRENT

Peron Rezende de Sousa

Orientadores
Sidney Cunha de Lucena
Morganna Carmen Diniz

Rio de Janeiro, RJ – Brasil
Setembro de 2013

Sousa, Peron Rezende de.
S729 Seleção de pares baseada em QoE para transmissão de vídeo em redes P2P BitTorrent / Peron Rezende de Sousa, 2013.
xv, 107 f. ; 30 cm + DVD

Orientador: Sidney Cunha de Lucena.
Coorientadora: Morganna Carmen Diniz.
Dissertação (Mestrado em Informática) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

1. P2P (Protocolo de rede de computador). 2. Internet. 3. Demanda em vídeo - Transferência. 4. BitTorrent. 5. QoE. I. Lucena, Sidney Cunha de. II. Diniz, Morganna Carmen. III. Universidade Federal do Estado do Rio de Janeiro. Centro de Ciências Exatas e Tecnológicas. Curso de Mestrado em Informática. IV. Título.

CDD - 004.67

SELEÇÃO DE PARES BASEADA EM QOE PARA TRANSMISSÃO DE VÍDEO EM REDES P2P BITTORRENT

Peron Rezende de Sousa

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Sidney Cunha de Lucena, D.Sc. – UNIRIO

Morganna Carmen Diniz, D.Sc. – UNIRIO

Carlos Alberto Vieira Campos, D.Sc. – UNIRIO

Antonio Augusto de Aragão Rocha, D.Sc. – UFF

Daniel Sadoc Menasché, Ph.D. – UFRJ

Rio de Janeiro, RJ – Brasil
Setembro de 2013

Dedico esse trabalho à minha adorável esposa por sua paciência, compreensão e amor; À minha mãe por me orientar e motivar desde a infância; E à minha tia por ter possibilitado, há 21 anos atrás, o inicio dos meus estudos na área da informática.

O melhor resultado é obtido
quando todos no grupo
fazem o que é melhor pra si
e para o grupo.

John Nash

Agradecimentos

Agradeço a Deus por me proteger em todos os meus caminhos e a todos que direta ou indiretamente me ajudaram durante essa jornada, em especial ao meu orientador Sidney Lucena e a minha co-orientadora Morganna Diniz, agradeço a ambos pela compreensão e tempo dedicado nos finais de semana e altas horas da noite. A paciência e bom humor dos professores Carlos Alberto, Vânia Dias e Márcio Barros. Aos colegas do mestrado Max Faria, Rodrigo Rocha, Anna Cruz, Nelson Machado Junior, Thiago Martorelli, Carlos Quintella, José Carlos de Albuquerque, Leonardo Anversi, Luiz Farias, Elenilson Gomes, Alessandra Nascimento e Douglas Brito. Aos fãs do Linux Diego Von Sohsten, Carlos Sari e Marco Gutierrez. Ao especialista em Java e professor Rafael Castaneda. Aos colegas de trabalho Eduardo Freitas, Cláudio Constantino e Vanessa Patrício. A Pablo Rodríguez-Bocca e Pablo Romero pelo suporte com GoalBit. Agradeço também aos integrantes da comissão examinadora, pois suas observações, dicas e críticas ajudaram a melhorar a versão final deste trabalho, ainda que a falta de tempo não tenha permitido aproveitar todas as orientações.

Resumo

A transmissão de vídeo representa grande parte do tráfego na Internet atualmente e as redes P2P são vistas como uma alternativa escalável e de baixo custo para atender a essa demanda, porém não oferecem garantias de QoS. Entre as áreas de pesquisas que procuram uma solução para esse problema estão os algoritmos para seleção de pares. Este trabalho avalia a QoE dos participantes de uma rede BitTorrent com base em uma nova metodologia, aqui proposta e denominada Avaliação Áurea de Qualidade (A^2Q). Ela estabelece uma métrica que quantifica e qualifica o QoE através da ausência de pedaços em um vídeo no momento da reprodução, o resultado recebe o nome "nível de estresse". Além de avaliar o QoE, também verificamos, em cada par, com quais outros participantes foi mantida uma comunicação contínua ao longo do tempo, formando com isso uma Lista de Pares Estáveis (LPE). Com isso o QoE obtido é atribuído a LPE. Nossa implementação, batizada de Seleção por Qualidade (S4Q), promove a troca de LPEs qualificadas por QoE com o objetivo de obter um menor tempo de *download* e uma reprodução contínua do vídeo com o mínimo de interrupções, durante sua transmissão. Comparamos o S4Q com o BitTorrent tradicional e com outras três soluções (Ono, P4P e Yukka). Os resultados mostram que nossa proposta é vantajosa quando comparada aos outros algoritmos do estado da arte e isso sem depender de fontes externas.

Palavras-chave: P2P; BitTorrent; QoE

Abstract

Video streaming is responsible for a significant share of the traffic on the Internet today and P2P networks are seen as an alternative scalable and low cost to meet this demand, but they do not offer QoS guarantees. Among the studies that find a solution to this problem are algorithms for peer selection. This work assesses QoE of peers of a BitTorrent network based on a new methodology, proposed here and called Avaliation Aurea of Quality (A^2Q). It establishes a new metric that quantifies and qualifies the QoE by absence of pieces in video at the time of playback, the result is called "stress level". Besides assessing the QoE, we also observed, in each peer, with which other peers was maintained a communication continuous over time, forming with it a List Peers Stable (LPE). With this the QoE obtained is assigned to LPE. Our implementation, named Selection for Quality (S4Q), promotes the exchange of LPEs qualified for QoE with the objective of obtaining a less download time and a continuous video playback with minimal interruptions, during its transmission. We compare S4Q with BitTorrent traditional implementation and other three solutions (Ono, P4P and Yukka). The results show that our proposal is advantageous when compared to other state of the art algorithms and that without relying on external sources.

Keywords: P2P; BitTorrent; QoE

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A evolução da Internet possibilitou que milhões pessoas passassem a compartilhar conteúdo que é totalmente (ou parcialmente) gerado e mantido por elas mesmas (Peer-to-Peer, Twitter, You Tube, Redes Sociais, são alguns exemplos de aplicações utilizadas para essa finalidade). Atualmente, apenas a banda cosumida por aplicações *peer-to-peer* e de vídeo juntas somam aproximadamente 80% de todo o tráfego da Internet atual [Rocha e Menasché 2013].

A distribuição de vídeo é uma das aplicações de maior sucesso atualmente na Internet. Serviços de vídeo sob demanda (*Video On Demand - VoD*) são acessados por milhões de usuários diariamente. Em 2008 o YouTube, site mais famoso desse gênero, já atingia a marca de 20 milhões de usuários por dia. Os serviços de vídeo ao vivo também vêm crescendo. Já em 2006 uma TV estatal chinesa usou um sistema P2P, o GridMedia, para transmitir ao vivo o Festival da Primavera Chinês. Estima-se que aproximadamente dois milhões de usuários assistiram a essa transmissão e que o sistema suportou mais de 200 mil usuários simultâneos [Moraes et al 2008, Li, Zhang e Yuan 2011].

Além do P2P, podemos encontrar outras soluções para fornecimento de vídeo, como o modelo cliente-servidor, o *multicast* e as redes de distribuição de conteúdo (*Content Delivery Network - CDN*). Porém, nenhum desses modelos se compara ao P2P em custo e escalabilidade. Por outro lado, o P2P não consegue oferecer determinados níveis de qualidade de serviço (*Quality of Service - QoS*). Isso abriu

um vasto campo de pesquisa e começaram a aparecer diversas propostas para vencer essas limitações.

Há muitos desafios nas redes P2P a serem estudados, entre eles estão os algoritmos para seleção de pares. Esses algoritmos têm o objetivo de otimizar a distribuição do conteúdo, considerando o cenário dinâmico dessas redes.

O processo de escolha dos pares começa com a solicitação de uma lista de participantes a um ou mais controladores (*trackers*) e depois, esses pedidos passam a ser feitos periodicamente para atualização. Normalmente, na sequência temos uma seleção feita com base em métricas de desempenho relacionadas à capacidade de transmissão, à continuidade da reprodução e à manutenção da qualidade do vídeo [Cui, Dai e Xue 2007].

Existem muitos trabalhos que abordam o problema da seleção de pares. Entre as propostas podemos citar:

- orientação por taxa de *upload* [Wen et al 2011];
- sobreposição de árvores [Awiphan, Zhou Su e Katto 2010] e sobreposição de malhas [Payberah, Dowling e Haridi 2011] com auxilio da taxa de *upload*;
- distância em saltos [Pereira, Vazão e Rodrigues 2012];
- controle do *buffer* [Lan et al 2011];
- por camadas de refinamento na distribuição de vídeo com formato de codificação em escalas [Kulatunga et al 2010, Kulatunga et al 2012];
- *supernodes* com cálculos de centralidade [Oliveira 2010] e
- informações do usuário [Bonti, Li e Shi 2011].

Embora as expressões *supernode* e *super-seeding* pareçam sinônimos, elas têm definições diferentes. Um *supernode*, numa rede par-a-par, é um nó que, por alguma característica, quebra a homogeneidade dos papéis dos pares na rede. Ele deve ter algum destaque ou uma função especial, como por exemplo ser responsável por

indexar os arquivos do seu grupo de vizinhos. No trabalho de [Oliveira 2010], são considerados *supernodes* aqueles que se destacam pela taxa elevada de *upload* entre todos os participantes da *overlay*.

Super-seeding é um algoritmo desenvolvido por John Hoffman para o BitTorrent. Esse algoritmo aplica-se a uma situação em que existe apenas um *seeder* no enxame. Ao permitir que cada par baixe apenas partes específicas dos arquivos listados em um *torrent*, ele possibilita que os pares se tornem *seeders* mais cedo. Em 2003, o BitTornado tornou-se o primeiro cliente de BitTorrent a implementar esse algoritmo [Theory 2013].

1.1 Motivação

Até o limite do nosso conhecimento, não foram encontradas na literatura propostas que façam uso de métricas associadas à qualidade da experiência do usuário (*Quality of Experience - QoE*) para seleção de pares. No entanto, o uso desse tipo de métrica para orientar escolhas não é algo novo. Em [Abboud et al 2010] implementou-se um algoritmo para seleção de camadas com base na QoE e em [Savas, Gürler e Tekalp 2011] foi elaborada uma arquitetura P2P que se adapta à qualidade do vídeo para melhorar a QoE (neste caso, as decisões do sistema se baseiam em um levantamento prévio sobre as percepções de um pequeno grupo de pessoas em relação a diferentes qualidades de vídeo).

1.2 Descrição do problema

Existem opiniões diversas sobre os benefícios da seleção de pares. Trabalhos como os de [Rejaie e Stafford 2004] e [Zhang et al 2005b] alegam que a amostra da rede conhecida já é suficiente para a escolha de pares que atendam os requisitos desejados e um refinamento nesse aspecto seria uma preocupação desnecessária. Outros pesquisadores, como [Liang e Nahrstedt 2006], propõem a separação dos membros da rede em grupos de acordo com as características de desempenho. [Silva et al 2010]

elaboraram um modelo que permite comparar o desempenho de diferentes estratégias, eles conseguiram mostrar que um sistema pode obter uma melhora significativa quando informações sobre os pares são utilizadas para organizar a rede.

Validar uma estratégia para seleção de pares não é tão simples. O problema já foi classificado com NP-Difícil, por ser uma variação do problema "Programação de Máquinas Paralelas", segundo [Liu et al 2008], e por ser mais complexo que "Definição de Domínio e P-Centros" da teoria dos grafos, quando envolve *supernodes*, afirmaram [Lo et al 2005]. Na verdade, mesmo que tenhamos conhecimento global de uma rede P2P, encontrar uma alocação ótima de recursos ainda é NP-Difícil [Liao et al 2006]. [Moraes et al 2008] destacam que encontrar mecanismos eficientes de escolha de pares e escaláveis em redes altamente dinâmicas permanece um desafio em aberto.

1.3 Objetivos do trabalho

Este trabalho avalia a QoE dos participantes de uma rede BitTorrent com base em uma nova métrica, aqui proposta. Nossa métrica quantifica e qualifica o QoE através da ausência de pedaços do vídeo no momento da reprodução e o resultado denominamos "nível de estresse". Além de avaliar o QoE, também verificamos, em cada um, com quais outros participantes foi mantida uma comunicação contínua ao longo do tempo, formando com isso uma Lista de Pares Estáveis (LPE). Com isso o QoE obtido é atribuído a LPE. Ao promover a troca de LPEs com boa QoE esperamos realizar uma transmissão mais rápida e uma reprodução contínua do vídeo com o mínimo de interrupções.

A eficiência da nossa implementação é comparada com o BitTorrent tradicional e com outros três algoritmos do estado da arte.

1.4 Contribuições

Como resultado deste trabalho realizamos contribuições na forma de novos conhecimentos científicos e produtos, que vão além da nova e simplificada métrica em QoE e do primeiro algoritmo para seleção de pares por QoE.

1.4.1 Conhecimentos

Considerando como conhecimento as novas informações produzidas por este trabalho, listamos a baixo as identificadas até o momento:

- Primeiro algoritmo para seleção de pares por QoE;
- Nova métrica em QoE, que se revela uma alternativa de fácil implementação em contraponto aos complexos métodos do estado da arte;
- Extenso comparativo entre soluções em variados cenários e;
- Método que induz a uma chegada sequencial de pedaços sem interferir no algoritmo para seleção de pedaços.

1.4.2 Produtos

Como produto consideramos toda a implementação necessária a viabilidade deste estudo e os dados produzidos durante o mesmo. Veja abaixo os principais:

- S4Q PlugIn, implementação do algoritmo para seleção de pares;
- S4Q Video Player, em conjunto com o S4Q PlugIn, exibi vídeo com *download* em andamento, mostrando também a evolução do nível de estresse nos últimos 30 minutos;
- Pedaços Ausentes PlugIn, gera *traces* sobre cortes e pausas a nível de pedaços;
- Conjunto de *Shell Scripts* que permitem experimentos simultâneos em ambiente real controlado (PlanetLab) e;

- *Traces* e bancos de dados consolidados, para comparativos entre diversas soluções em variados cenários.

1.5 Estrutura do texto

Este trabalho possui a seguinte organização: o Capítulo 2 discute os termos e as teorias que formam a base dessa pesquisa; o Capítulo 3 explica a nova métrica e o processo de escolha dos pares; o Capítulo 4 descreve o ambiente em que foram realizados os experimentos, a coleta, o processamento dos dados e as estratégias utilizadas; o Capítulo 5 apresenta os resultados e uma análise dos mesmos; por último, o Capítulo 6 descreve as conclusões e aponta possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Os sistemas P2P podem ser construídos sobre uma arquitetura em árvore ou em malha. A primeira tem uma melhor performance, mas apresenta uma lenta recuperação a saída de pares "galhos" e não aproveita a banda de *upload* da maioria dos pares, por serem "folhas". Na arquitetura em malha ocorre o inverso, menor desempenho, rápida recuperação a saídas e aproveitamento da capacidade de *upload* [Moraes et al 2008].

Também existem arquiteturas hibridas, mas para este trabalho escolhemos uma das mais famosas implementações do protocolo P2P, o BitTorrent, que foi construído com a arquitetura em malha.

2.1 BitTorrent

O BitTorrent é um protocolo P2P, criado por Bram Cohen, que tem o objetivo de favorecer o compartilhamento conteúdo entre os usuários (*peer*), que fazem parte de um grupo (*swarm*). Esses usuários são qualificados em fornecedor do conteúdo (*seeder*), possuidor da lista de participates (*tracker*) e interessados pelo material (*leecher*) [Rocha e Menasché 2013, Moraes et al 2008].

Embora o BitTorrent tenha sido desenvolvido para outra finalidade, Bram Cohen tem trabalhado para adaptar a sua criação à transmissão de vídeo ao vivo e já entrou com o pedido de patente [BitTorrent Live 2013].

Na distribuição, o que será compartilhado é dividido em pedaços (*chunks*), de

normalmente 512 KBytes. Para cada um deles é calculado um código *hash* SHA1 de 20 Bytes correspondente ao seu conteúdo, esse código serve para verificar a integridade do pedaço, após seu recebimento.

O código *hash* é uma sequência de bits gerados por um algoritmo de dispersão, em geral é representado em base hexadecimal. Esses *hashes* são concatenados para formar uma *string* que irá formar o valor *pieces* no arquivo de metadados (.torrent). Note que isso não é uma lista, mas sim uma única sequência e seu comprimento deve ser um múltiplo de 20.

A sequência de códigos *hash* e o endereço do *tracker* são informações obtidas no arquivo de metadados. Com essas informações um *leecher* pode entrar em um *swarm*. Esse arquivo de metadados é escrito no formato BEncode, que é uma forma de especificar e organizar dados em formato conciso [Theory 2013].

No BitTorrent o vídeo é dividido em pedaços e cada um deles recebe um número de identificação único e distribuído aleatoriamente no VoD. No GoalBit [Barrios et al 2011], uma adaptação do BitTorrent para vídeo ao vivo, como não existe o vídeo inteiro a ser dividido, os pedaços recebem um número indentificador cíclico que vai de zero à 2^{31} no momento em que são gerados. Para adequar a necessidade de pedaços sequenciais do vídeo ao vivo foi feita a proposta de dividir o *buffer* em faixas classificadas como urgente, usual e futura, para orientar a seleção de pedaços [Bertinat et al 2009]. É uma boa prática gerar os *chunks* com 512 KBytes ou menos. Programas como o Vuze [Vuze 2012] utilizam esse tamanho por padrão [Theory 2013].

Entre as diversas implementações do BitTorrent escolhemos para este trabalho o Vuze, por ser *Open Source*, multiplataforma e por aceitar *PlugIns*.

2.1.1 Vuze (atual Azureus)

O Vuze é um programa desenvolvido em Java, que implementa o protocolo BitTorrent [Vuze 2013a]. Existem dois tipos de Vuze, um com código aberto (*Open Source*) e outro com código fechado, proprietário.

Anteriormente chamado de Azureus, o Vuze foi elaborado de tal forma que é possivel adicionar funcionalidades e/ou modificações através de *PlugIns* [Vuze 2013b].

Dentre as funcionalidades do pacote comercial (Vuze Plus) podemos encontrar o Play Now [Vuze 2013c] que possibilita assistir ao vídeo enquanto este ainda está sendo baixado.

O vídeo inicia a reprodução com base na Fórmula 2.1 [Vuze 2013d]:

$$f(z) = \begin{cases} t\left(\frac{y}{z} - 1\right) & \text{para} \quad 0 \leq z \leq y \\ 0 & \text{para} \quad z > y \end{cases} \tag{2.1}$$

Na Fórmula 2.1, $\boldsymbol{t}$ representa o tempo total do vídeo em segundos, $\boldsymbol{y}$ a taxa de *download* necessária para uma reprodução contínua em Kbits/s e $\boldsymbol{z}$ a taxa de *download* do usuário, também em Kbits/s. Com isso podemos obter $\boldsymbol{x}$, que é o tempo de espera necessário ao ínicio da reprodução para que não ocorram interrupções.

Por exemplo, um filme de duas horas que exija uma taxa de 512 Kbits/s só poderá ser visto pelo usuário após uma espera de 8 minutos, caso sua velocidade de *download* esteja em 480 Kbits/s.

Caso a taxa do usuário diminua após o início da reprodução o vídeo pode vir a ser interrompido e o tempo de espera recalculado. Neste caso $\boldsymbol{t}$ é o tempo remanescente.

2.1.2 Outros aplicativos

Na Tabela 2.1 podemos ver uma lista de programas para transmissão de vídeo com licença *GNU General Public License* (GPL), ou seja, são *Open Source*, a coluna Mídias indica os conjuntos de bibliotecas para formatos de vídeo suportados pelos programas [SCVI.NET 2013a]. A Tabela 2.2 apresenta uma relação de programas proprietários para transmissão de vídeo [SCVI.NET 2013b]. Além desses aplicativos podemos citar (1) a Java BitTorrent API [Sourceforge 2013]; (2) o AnySee [Liao et al 2006], que é um sistema P2P para *Live Streaming* desenvolvido em Java; (3) o BitStream [BitStream 2013] que está sendo desenvolvido em Java com código aberto (licença GPL) que utiliza o *Freedom for Media in Java* (FMJ) para suporte

ao vídeo [FMJ 2013], esse recurso é uma alternativa ao *Java Media Framework API* (JMF) [Oracle 2013]; e (4) o Open-Source NextSharePC [Eberhard et al 2012].

O NodeZilla (Tabela 2.1) utiliza o *package* VLCj da VideoLAN [VideoLAN 2013a], a parte Java desse recurso apenas faz uma ponte para as bibliotecas de vínculo dinâmico (*Dynamic-link library - DLL*) da VLC, que são escritas em C [VideoLAN 2013b].

Tabela 2.1: Programas com código aberto para transmissão de vídeo em P2P [SCVI.NET 2013a]

Programa	Linguagem	S. O.			Mídias			
		Windows	Linux	Mac	Windows Media	Theora	Dirac	NullSoft
GoalBit [Bertinat et al 2009]	C++	X	X	X	X	X	X	X
Tribler [Pouwelse et al 2008]	Python	X	X	X	X	X	X	
PeerStreamer [Li 2004]	C	X	X	X	X	X	X	
Freecast [Freecast 2013]	Java	X	X	X		X		
Stream 2 Stream [Stream 2 Stream 2013]	Java	X	X	X		X	X	X
P2P-Radio [P2P-Radio 2013]	Java	X	X	X		X	X	X
NodeZilla [Nodezilla 2013]	Java	X	X	X	X	X	X	X
Trevbus [Trevbus 2013]	C++		X		X	X	X	X
VideoTyrant [VideoTyrant 2013]	C++	X			X	X	X	X
FlightPath [FlightPath 2013]	C++			X	X	X	X	X

2.2 Vídeo em camadas

Com a transmissão de vídeo pela Internet foram criados formatos que permitem aos usuários, com diferentes capacidades, receberem conteúdos com qualidades diferentes (camadas de refinamento). Um exemplo é a codificação do vídeo em escala (*Scalable Video Coding - SVC*) que é uma extensão do formato H.264 (MPEG-4 AVC). Outro exemplo é a codificação em multiplas descrições (*Multiple Description Coding - MDC*) [Abboud et al 2011].

No domínio da transmissão de vídeo, o SVC vem como uma solução que se adapta à variação da largura de banda de rede e de terminais heterogêneos. O critério de múltiplas camadas de codificação do SVC aumentou sua correlação com

Tabela 2.2: Programas com código proprietário para transmissão de vídeo em P2P [SCVI.NET 2013b]

Programa	País	Idioma	Site
Allcast	EUA	Inglês	http://www.allcast.com/
AnyVue	China	Inglês	http://mwnet.cse.ust.hk/p2pstream/
Grid Networks	EUA	Inglês	http://www.gridnetworks.com/
Haihaisoft	China	Inglês	http://www.haihaisoft.com/Live_P2P_Broadcasting.aspx
KeyHole TV	Japão	Japonês	http://www.v2p.jp/
Mermaid Multisource	India	Inglês	http://mermaid.metaaso.com/mermaid/multisource/overview.html
VJLive	China	Inglês	http://www.nagashare.com/vjlive.html
Network Foundation Technologies	EUA	Inglês	http://www.nft-tv.com/
Octoshape	Dinamarca	Inglês	http://www.octoshape.com/
PeerpApp	EUA	Inglês	http://www.peerapp.com/
Peer Live	Reino Unido	Inglês	http://www.peerlive.org/
PPLive	China	Inglês	http://www.pplive.com/en/ index.html
PPStream	China	Inglês	http://www.pps.tv/en/
QQLive	China	Chinês	http://tv.qq.com/download.htm
RAYV	EUA	Inglês	http://www.rayv.com/
Roxbeam	China	Chinês	http://www.roxbeam.com/
Sharecast 2	Japão	Japonês	http://scast.tv/sc2/
Sopcast	China	Inglês	http://www.sopcast.org/
StreamerOne	Italia	Inglês	http://www.streamerone.com/
Streamer P2P	Reino Unido	Inglês	http://www.streamerp2p.com/
Swarmcast	EUA	Inglês	http://www.swarmcast.com/
Synacast	China	Inglês	http://www.synacast.com/en/
Tvkoo	China	Inglês	http://www.tvkoo.com/en/aboutus.htm
TVU Networks	China	Inglês	http://www.tvunetworks.com/
Uusee	China	Chinês	http://www.uusee.com/download/
Vatata	China	Inglês	http://www.vatata.com/en/
Vgo	China	Chinês	http://vgo.21cn.com/
XunBoo	China	Inglês	http://www.xunboo.com/en/
Zatto	EUA	Inglês	http://www.zattoo.com/

a transmissão através de vários caminhos. No entanto, um importante desafio a enfrentar é como maximizar a qualidade de vídeo em geral de uma forma que satisfaça o usuário final, sem avidamente consumir recursos da rede. Para isso, o trabalho de [Ghareeb, Ksentini e Viho 2011] visa proporcionar um sistema que se baseia em avaliações da qualidade da experiência do usuário (*Quality of Experience - QoE*), para ajustar a transmissão de fluxos de vídeo SVC sobre vários caminhos. O método seleciona dinamicamente os melhores caminhos na sobreposição usando as estimativas de largura de banda disponível. Manter / atualizar os caminhos selecionados é, então, feito automaticamente com base no *feedback* da qualidade como ela é percebida pelo usuário final. Para avaliar a QoE no destinatário, os autores usaram um módulo compatível com SVC e uma ferramenta híbrida para Avaliação da Qualidade Pseudo-Subjetiva (*Pseudo Subjective Quality Assessment - PSQA*).

O PSQA também foi utilizando por [Silva et al 2008b] em uma rede P2P destinada a distribuição dos fluxos de vídeo em tempo real. Os autores utilizaram uma abordagem *multi-source*, onde o vídeo é decomposto em vários fluxos e enviados por diferentes pares para cada cliente. O objetivo era verificar a resistência dessa abordagem à frequente entrada e saída de pares. Os resultados mostraram é possível compensar de forma eficiente as perdas de quadros.

No fluxo dinâmico, um usuário pode dinamicamente escolher entre diferentes versões do mesmo vídeo. Na transmissão dinâmica com P2P, há um enxame para cada versão e, dentro de um enxame, os pares podem compartilhar pedaços de vídeo uns com os outros, reduzindo assim o custo da largura de banda do servidor. Com foco na cooperação entre os pares [Tian et al 2013] utilizaram a teoria dos jogos cooperativos para atribuir dinamicamente a cada *peer* uma versão. Para incentivar ao máximo a cooperação, os autores utilizaram um mecanismo que avalia a comunicação considerando o conteúdo e a largura de banda. Com esta abordagem, cada *peer* é atribuído de um enxame que é proporcional à sua contribuição em *upload*.

2.3 Algoritmos para seleção de pares

Ao entramos em um *swarm*, devemos solicitar ao *tracker* uma lista com os endereços dos outros participantes. Por padrão, ele envia uma lista com, no máximo, 50 pares, que são selecionados aleatoriamente [Theory 2013].

O desafio está em trocar a lista selecionada de forma aleatória e uniforme por um agrupamento dos pares que faça o sistema funcionar melhor. O que dificulta a solução dessa questão é a estrutura dinâmica da rede, isto é, os pares podem deixar o *swarm* a qualquer momento (*churn*), pelas mais variadas razões.

A seguir veremos o que já foi proposto até o momento para solucionar esse problema.

Entre os projetos baseados na arquitetura em malha encontramos o trabalho de [Lobb et al 2009] que constrói e mantém uma topologia de sobreposição P2PTV por meio de um mecanismo simples e totalmente distribuído. Esse algoritmo move os *supernodes* para perto da fonte de vídeo, isso melhora o atraso na entrega para todos os pares da rede. A propriedade chave desse esquema é a capacidade de estimar indiretamente a banda de *upload* dos pares sem ter que conhecer explicitamente ou medir. Os resultados das simulações mostram uma melhoria de 50% no desempenho.

Em [Awiphan, Zhou Su e Katto 2010] são utilizadas árvores sobrepostas para evitar o problema da lenta recuperação da estrutura quando pares saem da rede com frequência, e o agrupamento dos pares é realizado com base na capacidade de *upload* de cada um. Dessa forma eles obtiveram uma melhora no balanceamento da banda, ou seja, pares com grande capacidade de *upload* eram colocados na parte superior da árvore para evitar sua subutilização.

No trabalho de [Wen et al 2011], o objetivo é maximizar o uso da banda de *upload* dos pares recém-chegados ao enxame (*swarm*). Os autores levaram em consideração alguns aspectos, como por exemplo: quando um novo usuário entra na rede e começa a receber pedaços, é muito pouco provável que esses pedaços interessem a quem está há mais tempo no grupo, logo a capacidade de *upload* desse novo membro não é aproveitada pelo sistema. Em uma transmissão de vídeo sobre demanda (*Video On*

Demand - VoD), utilizou-se a troca do ponto de reprodução com os vizinhos (uso do *buffer*) para se obter uma redução de 60% no *upload* do servidor.

Live Streaming e *Video-on-Demand* (VoD) são dois campos da transmissão de vídeo (*video streaming*) que têm sido tratados como completamente diferentes. Porém, eles são, do ponto de vista do usuário, muito semelhantes. Usuários de um sistema P2P colaboram com sucesso na distribuição de fluxos de vídeo, permitindo assim que uma colaboração integrada tenha pares com estoque de vídeos assistidos no passado, para permitir a sua distribuição no futuro. As principais propriedades do P2P, como escalabilidade e economia de custos, dependerão da eficácia do mecanismo de incentivo subjacente. Em [Hecht e Stiller 2010] é proposto um relatório e um mecanismos de incentivo baseado em reciprocidade.

Entre os algoritmos de incentivo ao *upload* e combate aos *free riders* (pares pouco ou nada colaborativos), podemos citar o iPASS [Liang et al 2010] que previamente atrai pares com com maior contribuição a velocidades maiores em sistemas assíncronos (VoD P2P), elevando com isso o nível de QoE de todos os participantes.

Para evitar o problema de acúmulo de reputação, em sistemas que consideram esse fator, e a limitação do tráfego, em sistemas baseados na reciprocidade, [Li, Liang e Wu 2012] propôs, como mecanismo de incentivo, uma linha de crédito com base em camadas de tributação. Esse modelo é totalmente distribuído e não confia em qualquer servidor central para gestão do crédito. A topologia formada em camadas e a estratégia de tributação podem inspirar pares à contribuir com a sua banda, e maximizar a utilidade individual e do sistema. Os resultados da simulação indicam que, com o mecanismo de incentivo proposto, pares cooperativos podem ganhar um desempenho muito melhor, enquanto aproveitadores seriam removidos do sistema.

[Birke et al 2011] aborda o problema da regulação das taxas de *upload* entre os pares, a fim de coincidir com a demanda do sistema e evitar sobrecarregar. Os autores propõem o *Hose Rate Control* (HRC), um sistema que controla a velocidade na qual os pares oferecem pedaços aos outros pares, isto é, controla a utilização da

capacidade de *uplink* dos pares. Isso tem sua importância em cenários heterogêneos como na Internet, onde a capacidade de upload dos pares é desconhecida e varia muito. HRC se adapta bem as banda disponíveis de *upload*, com isso a qualidade da experiência dos usuários é muito maior. Ambas as simulações e as experiências reais que envolvem até 1000 pares são apresentadas para avaliar o desempenho em cenários reais. Os resultados mostram que a HRC é superior aos sistemas não-adaptativos.

Trabalhando com vídeo ao vivo (*Live Streaming*) temos o GLive. Neste método para seleção de pares, seus autores idealizaram um sistema com malhas sobrepostas, onde os pares com maior poder de *upload* são "colocados" próximos à fonte (*broadcaster*) e os pares com taxa de *upload* semelhantes são agrupados [Payberah, Dowling e Haridi 2011]. Dessa forma, o problema com *free riders* fica resolvido ou, pelo menos, atenuado, pois o sistema prioriza o fornecimento de dados aos bons fornecedores.

Com os serviços de internet para os usuários finais se tornando mais homogêneos, proporcionando alta largura de banda para todos, serviços de multimídia, tais como IPTV, vão se tornando uma realidade. Mesmo tendo em conta os recursos mais abundantes, arquitetura IPTV está longe de ser altamente disponível devido a limitações técnicas. O trabalho [Bonti, Li e Shi 2011] visa proporcionar uma otimização significativa no modelo de distribuição P2P, que atualmente está baseado em uma estrutura aleatória limitada pelos altos atrasos e baixo desempenho. Os autores usaram probabilidade de canal, hábitos de estudos de usuários e similaridade dos usuários para otimizar um dos aspectos-chave da IPTV, que é a gestão de seus pares, o que reflete diretamente sobre os recursos e a qualidade da experiência do usuário. Seguindo essa linha, da tomada de decição com base nas informações dos usuários, encontramos o trabalho de [Ioannidis, Chaintreau e Messoulié 2009], nele os autores mostram uma distribuição de conteúdo dinâmico sobre uma rede social móvel.

Os trabalhos [Kulatunga et al 2010, Kulatunga et al 2012] avaliaram o método atualmente utilizado para troca de pares fornecedores com problemas no *upload*

durante uma transmissão com SVC. O atual método consiste em trocar todos os pares, então fizeram a proposta de trocar apenas o fornecedor que esta apresentando problemas. O novo modelo foi comparado com um programa para transmissão de TV em par-a-par (*Peer-to-Peer Television - P2PTV*), o CoolStreaming [Zhang et al 2005a]. Os resultados mostraram uma melhora considerável na qualidade percebida pelo usuário (*Quality of Experience - QoE*).

A fim de oferecer boa qualidade de serviço, dois aspectos fundamentais devem orientar a construção de uma arquitetura em árvore para difusão de vídeo: baixo grau dos *peers* e distâncias curtas para o servidor. Com base nesse princípio, [Miranda e Figueiredo 2009] proporam um processo muito simples que constrói uma árvore para difusão de vídeo. O modelo de gerador é baseado no princípio de ligação preferencial, onde a preferência é dada em termos de qualidade do *peer*. Os resultados indicam que o modelo proposto é capaz de se auto-organizar em boas árvores sob várias avaliações da qualidade do *peer*.

Sistemas Peer-to-Peer (P2P) para compartilhamento de arquivos consomem uma parte muito significativa do tráfego da Internet, afetando o desempenho de outros aplicativos e traduzindo-se em custos significativos para os ISPs (*Internet Service Providers*) dos pares participantes. Tem sido notado que, assim como o tráfego WWW, o tráfego P2P de compartilhamento de arquivos mostra as propriedades de localidade, que não são exploradas pelos atuais protocolos P2P de compartilhamento de arquivos. [Pereira, Vazão e Rodrigues 2012] proporam um algoritmo de seleção de pares, *Adaptive Search Radius* (ASR), onde os pares exploram a localidade, baixando apenas dos outros colegas que estão mais próximos (em saltos de rede).

O ASR garante robustez ao enxame adaptando dinamicamente a distância de acordo com a disponibilidade apresentada dos pedaços. Ele visa reduzir o tráfego P2P e diminuir o tempo de *download*.

Os autores acreditam que o ASR é o primeiro sistema de compartilhamento de arquivos P2P por localidade, ciente de que ele não necessita da assistência dos ISPs ou de terceiros, nem de modificação da infraestrutura dos servidores. A avaliação desta

proposta foi apoiada com estudos de simulação abrangentes, utilizando o protocolo eDonkey / eMule em SSFNet. Estes mostram uma diminuição de 19% a 29% no tempo de *download* e uma redução de 27% para 70% no tráfego transportado entre ISPs. ASR também é comparado (favoravelmente) com *Biased Neighbour Selection* (BNS) e *traffic shaping*. Concluiu-se que o ASR e BNS são soluções complementares que proporcionam o mais alto desempenho quando combinados.

Foi avaliado o impacto do tráfego P2P em compartilhamento de arquivos com HTTP, mostrando os benefícios do desempenho HTTP na reduzição do tráfego P2P. Um plano para a introdução do ASR em clientes do eMule também é discutido. Isso permitirá uma migração progressiva das versões ativas do software cliente eMule para o ASR. O ASR também foi utilizado com sucesso no *download* de vídeo ao vivo, proporcionando economias significativas no tráfego.

O gerenciamento do *buffer* em malhas P2P para transmissão de vídeo ao vivo tem efeito direto e significativo na QoE.

Devido à instabilidade e heterogeneidade das redes P2P, o *buffer* não está sempre pronto para a reprodução. Esse fato não está associado apenas ao "ponto" do tempo no vídeo, mas também com a disponibilidade de dados em seus vizinhos. Gerir a sincronização da posição do *buffer* é crítico para prover a continuidade da reprodução.

O *Buffer Mapa Matrix* (BMM) de [Lan et al 2011] é uma estrutura de dados para descrever a disponibilidade de pedaços, de cada ponto e seus vizinhos. Além disso foi proposto um algoritmo de gestão dinâmica assíncrona do *buffer*, *MaxCount*, no qual os pares sincronizam a sua janela do tempo para a posição onde a quantidade máxima de blocos poderá ser recebida. A posição é prevista de acordo com a tabela BMM e a capacidade de *upload* dos vizinhos. A janela desloca-se para a posição onde se encontra a quantidade máxima de blocos disponível no *buffer* dos vizinhos. Os autores realizaram experiências extensivas com simulação no NS-2 [NS-2 2013]. Os resultados experimentais mostraram que o algoritmo *MaxCount* tem um melhor desempenho do que as abordagens existentes na continuidade de reprodução e adaptabilidade em redes com *churn*, especialmente quando a largura

de banda agregada aos pares é limitada.

Entre as soluções propostas para vencer o desafio da seleção de pares está o **P4P** [Xie et al 2008], ele usa "dicas" fornecidas pelos ISPs sobre suas redes para identificar pares melhores. Nem todos os ISPs tem suporte para P4P, portanto seu uso não representa vantagens atualmente, mas quando existe esse suporte ocorre uma melhora no desempenho para o usuário final e também há uma redução no tráfego entre ISPs, o que representa um custo menor para os provedores.

Outra solução para a seleção de pares é o **Ono** [Choffnes e Bustamantel 2008], sua estratégia é fazer uso de informações sobre o redirecionamento das redes de distribuição de conteúdo (*Content Delivery Network - CDN*) para identificar colegas próximos e potencialmente acelerar *downloads*, além de reduzir o trafégo entre ISPs. Esse método consegue aumentar a taxa média de *download* em 31%.

Em outra proposta encontramos o método **Yukka** [Polaczyk e Cholda 2010]. Essa solução realiza consultas aos *Regional Internet Registries* (RIRs), mas especificamente ao europeu RIPE NCC [RIPE 2012] e ao norte americano ARIN [ARIN 2012], para promover agrupamentos geográficos. Através do IP, ele obtém *Country*, *Netname*, *Description* e *Maintained_by*, com essas informações o algoritmo atribui uma nota de similaridade entre os pares, para assim, agrupa-los objetivando obter um menor tempo de *download*. Os resultados mostram um redução de até 25% no tempo de *download*.

2.4 *Quality of Experience* (QoE)

Atualmente, o termo QoE é utilizado em diversas áreas e representa as métricas utilizadas para qualificar a percepção do usuário no uso de determinado produto/serviço. Não se trata de uma medida objetiva, mas subjetiva pois depende da opnião do usuário. No entanto, qualificada essa impressão podemos relaciona-la às medidas objetivas que orientam a QoS.

Entre os estudos sobre métricas para QoE encontramos a Avaliação de Qualidade Pseudo-Subjetiva (*Pseudo-Subjective Quality Assessment - PSQA*), que constrói um

mapeamento entre certos fatores de qualidade e o que é recebido pelos usuários finais. Essa relação pode ser aprendida através de uma ferramenta estatística, a Rede Neural Aleatória (*Random Neural Network - RNN*). O resultado final é uma função capaz de imitar, de alguma forma, a maneira que um ser humano médio avalia a qualidade de um fluxo de vídeo [Rodríguez-Bocca 2008].

Em [Michel et al 2010] é apresentado um estudo que correlaciona cache, largura de banda e QoE, com isso os autores elaboraram uma estrutura para calcular um ponto de operação ideal.

[Fu et al 2010] faz uma comparação entre a taxa necessária a reprodução do vídeo e a descontinuidade, para ajudar na seleção de um fluxo adequado para o aumento da QoE.

Uma metodologia para a avaliação da QoE (usando métricas objectivas padrões para qualidade de vídeo) foi proposta por [Kiraly, Abeni e Lo Cigno 2010]. Eles mostraram que diferentes formas de agrupar os quadros em blocos para a distribuição podem levar a uma qualidade muito diferente, quando o sistema é sobrecarregado.

Em [Menkovski, Exarchakos e Liotta 2010] são apresentadas técnicas de inteligência artificial para modelar as dependências de diferentes redes e de parâmetros QoS para QoE na camada de aplicação, usando a resposta do usuário (*feedback*).

No trabalho de [Juluri, Plissonneau e Medhi 2011] uma ferramenta de tomografia (Pytomo) foi projetada para medir a qualidade da reprodução do vídeo, como se ele estivesse sendo visto por um usuário. Diversos testes foram feitos no YouTube.

O YouTube é um site que permite que seus usuários carreguem e compartilhem vídeos em formato digital, através de uma CDN. Em 22 de Novembro de 2008, foi lançado o YouTube Live para transmissões de vídeo ao vivo [YouTube 2013].

No artigo de [Ibekwe, Klima e Soucek 2011], os autores adotam métricas objetivas, em conjunto com a avaliação subjetiva de um usuário, para estudar a percepção da qualidade das imagens geradas por serviços de segurança baseados em IP submetidos à erros de comunicação. Para servir de referencial eles utilizaram o nível mínimo de identificação visual de objetos (limite de QoE) sobre o conteúdo degradado.

Em [Espina et al 2011] os pesquisadores utilizaram *Frame Starvation Ratio* para avaliar a qualidade do vídeo e melhorar a recepção do vídeo quando o caminho da rede apresenta perdas em rajadas. Os resultados mostraram que a classificação de quadros no destino e um seriviço para identificar o tipo de rede podem melhorar a qualidade da reprodução substancialmente.

Em outro trabalho [Agboma, Smy e Liotta 2008] foi proposto um método de avaliação para QoE em TV P2P. Nesse artigo foram apresentados resultados preliminares com o uso Joost (sistema de transmissão de vídeo sob P2P). Ele sugere que longos atrasos para o início da reprodução e falta de continuidade suave afetam a experiência de visualização. Os autores reconhecem as dificuldades por trás de avaliações subjetivas em P2P devido a cobertura de grandes áreas geográficas e numerosos pares. Nesse trabalho os autores relatam a questão da performance ser afetada pelo comportamento imprevisível do usuário e pelo *status* das sub-redes. Eles afirmam que experimentos em pequena escala não refletem adequadamente o comportamento desses sistemas na vida real. Também afirmam que sistemas P2P são muito difíceis de compreender. Eles avaliaram os quatro fatores que contribuem para o QoE em TV P2P e os relacionaram ao atraso, a largura de banda e a perda de pacotes. Nesse estudo são listadas as mais diversas técnicas, inclusive algumas que não devem ser utilizadas, seus riscos, explicações e exemplos com boas ou falsas conclusões. O artigo apresenta duas arquiteturas e métodos para correção de erros, além de citar algumas ferramentas e técnicas. Além disso, essa pesquisa demonstrou que a variável mais sensível é a perda de pacotes e que é necessário um estudo com maior número de indivíduos com idades diversas, utilizando uma granularidade de perda entre 0 a 14%, estender o trabalho a um ambiente móvel, verificar questões sobre gerenciamento de direitos digitais, proteção de direitos autorais e segurança de rede.

O Joost é um sistema para transmissão de vídeo sob P2P baseado na arquitetura em malha. Octoshape, PPLive, Zattoo, PPStream, SopCast e TVants são outros sistemas que utilizam a arquitetura em malha. PeerCast e Conviva são exemplo de

sistemas P2P com arquitetura em árvore [Gu et al 2011].

No trabalho de [Rodríguez-Bocca 2008] podemos observar que o autor idealiza a troca da métrica perda de pacotes por perda de quadros do vídeo. Segundo o autor, estudar o impacto das perdas a nível de quadros permite a independência do tipo de falha de distribuição e os resultados podem ser aplicados aos erros de transmissão em rede (isto é, congestionamento) ou as falhas do servidor (por exemplo, em nosso contexto, a desconexão de um par). Além disso, o estudo mostra que o processo de perda de pacotes não se correlaciona bem com qualidade num contexto geral devido à dependência do protocolo utilizado. Já os fatores sobre perda de quadros tem aplicabilidade geral.

Rodríguez-Bocca também buscou uma melhora da QoE utilizando a técnica de múltiplas-fontes, onde o fluxo de vídeo é decomposto em diferentes fluxos redundantes que viajam de forma independente através da rede. A idéia é controlar o número de fluxos, suas taxas e a quantidade da redundância transportada por cada um deles, possibilitando a escolha de uma configuração tão robusta quanto se desejar em termos de qualidade percebida. Além disso, essa técnica conduz a um custo muito baixo de sinalização (sobrecarga), em contraste as abordagens BitTorrent tradicionais.

Um sistema chamado MOAM (*Monitoring of Operation, Administration and Management System*) é proposto em [Huang et al 2011]. Ele serve para monitorar operações, administrar e gerenciar sistemas P2P. Os autores consideram que a engenharia de tráfego pode melhorar o QoE/QoS.

Uma arquitetra também é proposta em [Cruvinel et al 2011], ela recebeu o nome de Distribuidora Dinâmica de QoS (DDQoS). Ela utiliza dados dos *IETF's Differentiated Services* para suprir a fronteira da rede com o conhecimento necessário a melhor gestão do tráfego. *Internet Engineering Task Force* (IETF) é uma organização que definiu como seu objetivo, "fazer a Internet funcionar melhor" [IETF 2013]. O DDQoS foi testado com as *Video Quality Metric* (VQM). No *Survey* de [Wang 2006] podemos encontrar mais informações sobre VQM.

No estudo de [Huo et al 2010] sobre o protocolo de TV para Internet (*Internet*

Protocol Television - IPTV) são demonstrados os principais parâmetros que impactam a QoE.

O que domina a percepção do usuário, no contexto da transmissão de vídeo, é o adiamento da reprodução segundo [Schatz, Hobfeld e Casas 2012]. Nesse trabalho é realizada uma análise sobre o tempo de *download*, taxa de transferência e *buffer* do vídeo.

A tecnologia P2P provou ser mais escalável nos serviços de vídeo sob demanda e ao vivo e é amplamente utilizada hoje. Como a maioria dos fluxos de vídeo apresentados em sistemas P2P são de canais de televisão, temos, portanto, feito referência a eles como sistemas P2PTV. O comportamento de um único usuário é apresentado em termos de uma exponencial para filas fechadas de rede. Com base nisso foi construido um modelo comportamental com vários usuários de redes P2PTV com múltiplos canais [Aminu, Gaidamaka e Samuylov 2010]. A distribuição estacionária de probabilidade do número de usuários por canal é obtida numa forma de produto (*product form*), em termos de número de canais e o número de utilizadores presentes na rede. Dada a popularidade dos canais de televisão, os autores obteram fórmulas para a análise de alguns parâmetros em QoE para P2PTV com número finito e infinito de espectadores.

Em [Khiem, Ravindra e Ooi 2011] foi realizado um estudo com 35 participantes vendo 5 *clips* de vídeo para entender a tolerância a latência da rede quando ha *zoom* e visão panorâmica em fluxos de vídeo com *zoom*. Com *zoom* ou visão panorâmica, regiões espaciais invisíveis em um quadro são reveladas e momentaneamente em um estado desconhecido até que os dados chegam do servidor. Para lidar com esse estado desconhecido, dois esquemas de ocultação comuns são usados, ou seja, sistema de preto e esquema Low-Res. Esquema Preto torna a região recém-revelada como pixels pretos, enquanto Low-Res cobre a parte desconhecida com dados de um fluxo de vídeo de baixa resolução, que é adicionalmente transmitido pelo servidor. No âmbito destes programas, o estudo com base na simulação de atrasos mostra que os usuários são mais tolerantes ao atraso no esquema Low-Res. Até 94% dos participantes

podem tolerar um segundo de atraso e 80% pode tolerar um atraso de 2 segundos no esquema Low-Res, enquanto apenas 77% dos participantes podem tolerar um segundo de atraso no esquema Preto. O atraso tolerável em transmissão de vídeo com *zoom* é maior do que os limiares encontrados em algumas aplicações multimídia de alta interatividade.

O impacto da qualidade na transmissão de vídeo já foi estudado utilizando-se extensivos *traces* da rede Akamai, que incluem 23 milhões de visualizações feitas por 6,7 milhões de visitantes. O estudo mostra que os espectadores abandonam o vídeo se este levar mais do que dois segundos para iniciar, para cada segundo adicional há um aumento de 8% na taxa de abandono. A pesquisa também mostra que uma quantidade moderada de interrupções pode diminuir significativamente o número de espectadores [Krishnan e Sitaraman 2012].

2.4.1 *Mean Opinion Score* (MOS)

Uma forma muito comum de avaliar a percepção do usuário é através de um escore médio de opinião (*Mean Opinion Score* – MOS).

MOS é um método subjetivo para teste de qualidade [Vorren 2006]. Nele o usuário qualifica o que percebe atribuindo uma das notas da Tabela 2.3.

Tabela 2.3: Qualificação subjetiva da qualidade com MOS [Vorren 2006]

MOS	Qualidade
5	Excelente
4	Bom
3	Razoável
2	Pobre
1	Ruim

Os pesquisadores [Mok, Chan e Chang 2011] testaram variações de QoS em transmissões de vídeo com HTTP e obtiveram valores diferentes de MOS quando optaram por separar os vídeos em 4 tipos de programas: Esportivos, Noticiários, Comédias e Musicais. O mesmo foi feito em [Boulos et al 2009] com o formato H.264/AVC, focando perda de pacotes e considerando vários tipos de programas.

No trabalho de [Kanumuri et al 2006] estudou-se dois métodos (*Classification and Regression Trees - CART* e *Generalized Linear Model - GLM*) que avaliam se uma perda de pacotes é ou não visível em vídeos MPEG-2.

A pesquisa de [Staelens et al 2009] consistiu em um estudo sobre a perda de pacotes e o congelamento de quadros na transmissão de vídeo. Ao final os autores propõem uma metodologia que avalia o efeito de um sobre o outro.

Os autores [Carreira et al 2010] trabalharam na melhoria da QoE disfarçando quadros perdidos em transmissões de vídeo 3D. Eles também mostraram que o MOS muda de acordo com o tipo de vídeo.

Em sua dissertação, [Vorren 2006] estudou a perda de pacotes em vídeos de alta definição e analisou a relação entre o MOS e as VQM com o formato H.264/AVC. Os resultados foram diferentes para cada VQM avaliada.

O trabalho [Mu et al 2009] também avalia o formato H.264/AVC, considerando os efeitos da perda de pacotes na qualidade percebida do vídeo. Os autores também separam os vídeos em 4 tipos: *Football*, *Betes*, *Autumn* e *Susie*. A Tabela 2.4 mostra a relação de cada vídeo com os quesitos complexidade e movimento de cena.

Tabela 2.4: Tipos de vídeos segundo a imagem [Mu et al 2009]

Complexidade / Movimento	Alto	Baixo
Alto	*Football*	*Betes*
Baixo	*Autumn*	*Susie*

Esses quatro tipos de vídeos (chamados *Susie*, *Betes*, *Football* e *Autumn*) foram selecionados a partir de uma sequência de testes publicados por um grupo de especialista em qualidade de vídeo. Eles refletem duas características de conteúdo (movimento e complexidade). *Football* possui um elevado nível de complexidade em textura e sequências como vários indivíduos se movimentando rapidamente para seguir a bola. O estudo assume que em vídeos de *Football* a tela como um todo é a região de interesse do usuário. Em contraste, *Susie* tem um baixo nível de movimento e textura, porque há apenas alguns movimentos de cabeça em primeiro plano. Além disso, tanto o primeiro plano e o fundo são simples. Neste caso, assume-se que a face

é a região de interesse. A sequência dos desenhos animados (*Betes*) e o *Autumn* estão no meio do sistema de classificação das características de conteúdo. Para exemplificar, *Betes* tem um nível elevado de movimento e um baixo nível de textura, sendo o centro da tela a região de interesse. Em contraste, o *Autumn* tem um baixo nível de movimento, pois possui apenas com a cachoeira no meio em movimento, enquanto a grande quantidade de árvores de *Autumn* constitui um alto nível de textura.

[Mwela e Adebomi 2010] pesquisou QoE atribuindo MOS à sistemas operando com 3 tipos de programas, 3 tamanhos de pacotes e com perdas de até 10%.

Tabela 2.5: Qualificação subjetiva da qualidade com MOS [Mwela e Adebomi 2010]

Vídeo	Perda de pacotes em percentagem									
Tipo	0%	0.1%	0.3%	0.5%	0.7%	1%	3%	5%	7%	10%
FM1500	3.93	3.78	3.33	2.52	3.22	2.78	2.19	2.78	2.07	1.70
FM1024	3.96	3.67	2.81	2.37	3.19	2.19	1.96	1.67	1.30	1.44
FM512	4.04	3.33	3.78	2.22	2.74	2.04	1.85	1.78	1.74	1.26
FT1500	3.70	3.41	3.52	2.70	3.33	2.33	1.96	2.11	1.44	1.26
FT1024	3.85	3.52	3.30	2.11	2.15	2.15	1.37	1.52	1.44	1.19
FT512	3.52	2.59	2.44	2.07	2.00	1.78	1.26	1.22	1.41	1.15
NE1500	3.85	3.65	3.78	3.15	3.67	3.30	2.67	2.19	3.04	1.96
NE1024	3.85	3.63	3.33	2.85	3.26	2.96	2.19	2.56	2.19	1.56
NE512	3.70	3.52	2.93	2.59	3.41	2.48	2.04	2.26	1.44	1.30

O experimento dessa pesquisa foi realizado em Blekinge Tekniska Högskola (BTH), na cidade de Karlskrona, Suécia, onde vinte e sete pessoas foram convidadas a participar. A Tabela 2.5 apresenta as opiniões dos telespectadores sobre a qualidade de diferentes videoclipes. As variáveis utilizadas neste trabalho foram a “perda de pacotes” e o “tamanho dos pacotes”. Também são utilizados vídeos diferentes, um dos vídeos tem movimentos rápidos e mudaças de cena. Ele é chamado de Futebol. As cenas no segundo vídeo, chamado de Capataz, não mudam rapidamente como o Futebol, mas suas cenas variam e mudam mais rápido quando comparado com o terceiro vídeo chamado Noticiário. A primeira coluna da tabela contém três tipos de abreviaturas. FM1500 é uma abreviação do vídeo *Foreman* (Capataz) e transmissão utilizando pacotes com tamanho de 1500 Bytes, enquanto FT e NE representam os vídeos *Football* e *News*, respectivamente. O tamanho dos pacotes para FT e NE são descritos na mesma forma dos vídeos FM.

O autor também observa que o aumento da perda de pacotes faz com que o valor do MOS decai exponencialmente.

[Ribadeneira 2007] estudou a influência do atraso, *jitter* e perda de pacotes em VoIP. Comparando com o trabalho de [Mwela e Adebomi 2010] vemos que existe uma maior tolerância para transmissões, exclusivamente, de áudio.

Não encontramos trabalhos sobre perda / ausência de quadros ou pedaços.

Capítulo 3

Proposta

Nossa proposta é desenvolver um novo algoritmo para seleção de pares que seja orientado por QoE, ele será referenciado neste trabalho como Seleção por Qualidade (S4Q). Para avaliar a QoE criamos uma nova metodologia, denominada Avaliação Áurea de Qualidade (A^2Q), que estima a percepção dos usuários com base na ausência a nível de pedaços. Neste trabalho vamos chamar o número produzido por esta métrica de "nível de estresse".

Além do PSQA, citado na Seção 2.4, podemos encontrar outras métricas de qualidade em vídeo no *survey* de [Wang 2006], veja Tabela 3.1.

O PSNR (*Peak-Signal-to-Noise-Ratio*) foi desenvolvido inicialmente para avaliação de fotografias, somente algum tempo depois ocorreu a adaptação para vídeos. Apesar de simples, o PSNR é considerado pouco assertivo na correlação com métodos subjetivos.

Wang destaca o MPQM (*Moving Pictures Quality Metric*) como um método que apresenta uma grande margem de erro.

A performace da avaliação do SSIM (*Structural Similarity Index*) no momento da transmissão de um vídeo é desconhecida e tanto ele quanto o VQM (*Video Quality Metric*) precisam conferir o vídeo recebido com o original.

Por fim, Wang afirma que o NQM (*Noise Quality Measure*) não possui nenhum estudo que relacione seus resultados com a percepção dos usuários.

O PSQA e outras métricas de qualidade apresentadas na Tabela 3.1 não foram

Tabela 3.1: *Métricas de Qualidade em Vídeo* [Wang 2006]

Métrica de Qualidade	Complexidade Matemática	Correlação com Métodos Subjetivos	Acessibilidade
PSNR	Simples	Ruim	Fácil
MPQM	Complexo	Instável	Não Disponível
VQM	Muito Complexo	Bom	Não Disponível
SSIM	Complexo	Razoável	Disponível (MATLAB)
NQM	Complexo	Desconhecido	Não Disponível

aqui adotadas devido a complexa implementação exigida. Diante desse cenário, decidimos seguir o trabalho de [Rodríguez-Bocca 2008] com o objetivo de criar um novo método para avaliar a qualidade através das perdas em nível de quadros. Buscamos em outras áreas do conhecimento uma forma mais simples de avaliar a percepção dos usuários e encontramos no mercado de capitais a Teoria de Elliott, que prevê o comportamento humano utilizando a Sequência Fibonacci [Matsura 2006]. Nossa proposta se baseia no uso da Teoria de Elliott para quantificar a QoE dos usuários.

3.1 Associando quadros à pedaços

Na transmissão de vídeo, entre as métricas tradicionais de QoS (*jitter*, latência e perda de pacotes) a que mais afeta a percepção do usuário é a perda de pacotes [Agboma, Smy e Liotta 2008]. No entanto, o número de fatores que normalmente afetam uma transmissão é mais elevado e seu efeito conjunto sobre a qualidade é complexo. Em [Silva et al 2008a] foi feito um estudo sobre os efeitos combinados da taxa de bits, correção de falhas, taxa de perda, distribuição da perda, atraso e *jitter* na qualidade de conversação percebida em um sistema VoIP. Para alcançar isso, os autores utilizaram o PSQA. Como resultado eles apresentaram um estudo detalhado dos efeitos de cada parâmetro e mostram que o PSQA pode ser usado para fornecer uma estimativa precisa da qualidade da conversação. Com relação a transmissão de vídeo, a tese de [Rodríguez-Bocca 2008] mostra que o processo de perda de pacotes não se correlaciona bem com qualidade num contexto geral devido

à dependência do protocolo utilizado. Por exemplo, o protocolo UDP sofre um efeito diferente do TCP, pois o TCP conta com um método de recuperação quando pacotes são perdidos. Para contornar esse problema, Rodrígues-Bocca sugere o estudo da perda em nível de quadros.

Um vídeo, por padrão, trabalha a uma taxa de 23 quadros por segundo. Como estamos usando o protocolo BitTorrent, que normalmente separa o conteúdo em pedaços de 512 KBytes, podemos concluir que cada pedaço, em um vídeo com taxa de 512 Kbits/s, contém alguns segundos e que cada segundo contém alguns quadros. Logo, decidimos implementar uma verificação sobre ausência de pedaços, ao invés de perda a nível de quadros. Isso simplificou consideravelmente o algoritmo. Para reduzir ainda mais a complexidade nos cálculos e comparações, decidimos estabelecer uma quantidade específica de segundos para cada pedaço. Essa decisão envolvia a escolha da frequência do vídeo a ser utilizada nos experimentos.

Em pesquisa ao *site* NetFlix.com, uma famosa locadora de vídeos da Internet, vimos que eles utilizam a taxa de 512 Kbits/s para vídeos de baixa resolução, e 1.536 Kbits/s para formatos em alta definição [NetFlix 2013]. Optamos pela taxa de 512 Kbits/s. Dessa forma, o vídeo foi dividido em pedaços iguais com 8 segundos cada, conforme Fórmula 3.1, onde $\boldsymbol{y}$ é o tamanho do pedaço em Kbits, $\boldsymbol{z}$ a taxa de transmissão do vídeo em Kbits/s e $\boldsymbol{x}$ o tamanho do pedaço em segundos.

$$x = \frac{y}{z} = \frac{512 \times 1024 \times 8 \times 1000^{-1}}{512} \approx 8s \tag{3.1}$$

3.2 Escala de Estresse Percebido (ESP)

Não encontramos estudos que façam qualquer relação entre perdas de pedaços e QoE, nem estudos que pontuassem a percepção desse evento fazendo uso de MOS. Decidimos estabelecer pontuações distintas para reproduções bem sucedidas e ausência de pedaços, dessa forma montamos uma escala de qualidade ao longo do tempo, o valor obtido no tempo vamos chamar de “nível de estresse”.

Durante este trabalho elaboramos e observamos duas formas de pontuar o nível

de estresse, uma que pontua as ausências e presenças de pedaços com relação 1:1 e outra com 1:1/2.

A primeira forma de traçar esse valor é pontuando cada falha com 1 e cada reprodução bem sucedida com -1 (relação 1:1). Por exemplo, a Figura 3.1 ilustra esse método, nele o usuário se depara com uma sequência de perdas e reproduções, porém o nível de estresse ficou estacionado, falhando ao representar a percepção do usuário.

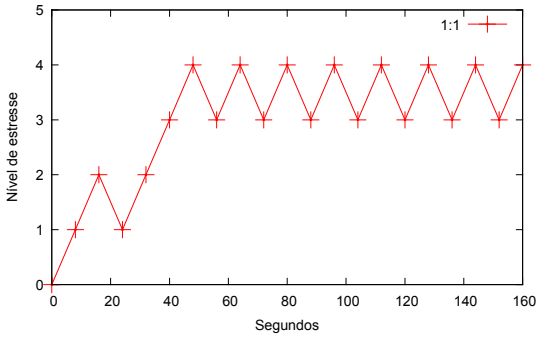

Figura 3.1: Exemplo de falhas e acertos com mesmo peso

Outra forma é considerar que o usuário não esquece as falhas com a mesma intensidade que percebe os acertos. Continuamos a pontuar as falhas com 1, mas os acertos passaram para menos meio ponto (veja um exemplo na Figura 3.2). Aqui temos uma melhor representação da percepção do usuário, pois o nível de estresse sobe diante uma sequência de erros e acertos, porém, há pesos desiguais, outro problema é a possibilidade de comportamentos lineares e o fato de não ter sido considerado o intervalo em que os erros ocorrem, pois é natural que 20 erros em 10 minutos provoquem um estresse maior que 20 erros em 2 horas. Logo, podemos esperar que a pontuação das falhas seja mais acentuada sempre que os erros ocorram próximos uns dos outros. O efeito cumulativo pode não resolver a questão naturalmente, conforme pode ser visto na Figura 3.1, pois erros próximos não aumentam nível de estresse.

Comportamentos lineares na avaliação da percepção do usuário não combinam com o que foi observado no trabalho de [Mwela e Adebomi 2010], onde o MOS seguiu o que parece ser uma exponencial.

Buscando uma solução para esse problema, pesquisamos formas para pontuar a

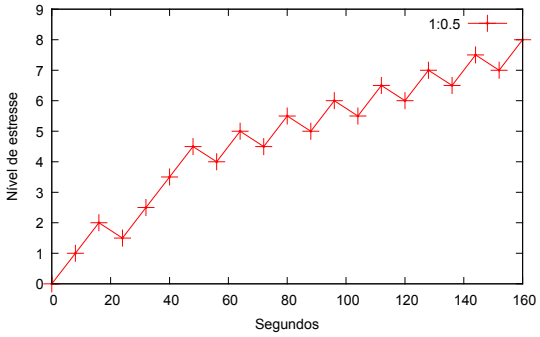

Figura 3.2: Exemplo de acertos com metade do peso das falhas

percepção humana. Na psicologia encontramos estudos sobre a Escala de Estresse Percebido (*Perceived Stress Scale - PSS*), mas não conseguimos estabelecer uma relação com transmissão de vídeo, pois os estudos envolviam uma avalição do estresse em um contexto geral no dia-a-dia das pessoas. Porém, notamos que nela é comum a prática de enquadrar os escores em categorias, como baixo, médio e alto. No entanto, os autores da PSS não recomendam esta prática. Eles alegam que, ao se agrupar diferentes escores em uma mesma categoria, perde-se precisão nas análises estatísticas [Luft et al 2007].

Muitos fatores precisariam ser considerados para refinar essa escala, como por exemplo, idade, sexo, escolaridade, situação econômica e estado civil. Ainda que essas informações possam ser obtidas em redes sociais, decidimos aproveitar apenas a questão dos três níveis e continuar a busca por uma forma mais simples e menos segmentada de avaliar a percepção das pessoas.

Durante as pesquisas encontramos uma teoria afirmando que o público age de forma emocional, subjetiva e impulsivamente, tomando decisões em condições de ignorância e incerteza, na maioria das vezes assumindo a chamada “atitude manada” [Ferreira 2007]. Vimos que o efeito manada ocorre entre investidores da bolsa de valores e quais os eventos que o provocam. Observando esses eventos acabamos identificando algumas semelhanças com a transmissão de vídeo P2P, como mostrado na Tabela 3.2. Por exemplo, quando um investidor observa a ação que comprou cair, juntamente com o aumento das ordens de venda, ou seja, com o aumento de investidores saindo daquela ação, ele tende a realizar uma operação de venda

para sair também, deixando aquele investimento. Na transmissão de vídeo, quando um espectador percebe uma queda na qualidade, juntamente com o aumento de espectadores saindo daquele canal, ele tende a realizar uma operação de desconexão para sair também, deixando aquele canal de vídeo. Faltava saber como associar essa teoria a ausência de pedaços.

Tabela 3.2: Efeito manada na bolsa de valores e na transmissão de vídeo

Home Broker	Canal de Vídeo P2P
Alteração no valor das ações	Alteração na qualidade do vídeo
Alteração na quantidade de ordens	Alteração na quantidade de pares
Investidor realiza operação	Espectador realiza operação

3.3 A Teoria de Elliott

Ralph Nelson Elliott foi um pesquisador que, entre as décadas de 30 e 40 do século XX, conseguiu demonstrar e provar graficamente que o movimento dos preços se comportava de forma cíclica formando padrões geométricos, os quais eram gerados pelo comportamento do emocional das massas no mercado financeiro.

Sua base de raciocínio é a de que a emoção surge primeiro que a ação. Por isso, a representação gráfica de uma série histórica de cotações nada mais é do que a oscilação de humor coletivo numa tentativa desesperada de encontrar sua precificação. Segundo Elliott, o público age de forma emocional, subjetiva e impulsiva, tomando decisões em condições de ignorância e incerteza, e, na maioria das vezes, assumindo a chamada “atitude manada”.

Com o propósito de quantificar a psicologia humana associada às oscilações dos preços, Elliott catalogou diversos padrões gráficos criando regras específicas, originando assim o Princípio das Ondas de Elliott ou, simplesmente, Teoria de Elliott [Elliott 1938, Mendonça 2013].

A identificação dos padrões gráficos permitiu que Elliott conseguisse encontrar as “formas” existentes no mercado de capitais, porém faltava uma ferramenta que lhe auxiliasse na medição dessas formas. Então, Elliott recorreu à matemática e

lá encontrou uma importante proporção chamada de Proporção Áurea ou Número de Ouro (aproximadamente 1,618) que foi extraída da sequência de números de Fibonacci, a qual possui fortes correlações com o seu princípio [Matsura 2006].

Para Elliott, os ciclos têm características definidas e a compreensão de sua Teoria permite que o investidor possa antecipar com certa precisão grandes reversões de tendência tanto de índices, ações, derivativos, *commodities* ou moedas, e assim, aproveitar os bons movimentos de alta ou de baixa do mercado.

A sequência Fibonacci tem o nome do matemático pisano do século XIII, Leonardo de Pisa, conhecido como Leonardo Fibonacci, e os termos da sequência são chamados números de Fibonacci. Ela é uma sucessão de números que aparece em muitos fenômenos da natureza. É infinita e começa com 0 e 1. Os números seguintes são sempre a soma dos dois números anteriores. Portanto, depois de 0 e 1, vêm 1, 2, 3, 5, 8, 13, 21, 34 e assim por diante.

Quanto mais você avança na sequência de Fibonacci, a divisão entre um termo e seu antecessor se aproxima de 1,618. Outra curiosidade é que ao transformar os números da sequência em quadrados e dispô-los de maneira geométrica, é possível traçar uma espiral perfeita, que também aparece em diversos organismos vivos [Ferreira 2007]. Ela tem aplicações na análise de mercados financeiros, na ciência da computação, na teoria dos jogos e nas artes. A chamada "proporção áurea" é muito usada na arte, na arquitetura e no *design* por ser considerada agradável aos olhos. Também aparece em configurações biológicas, como, por exemplo, na disposição dos galhos das árvores, das folhas em uma haste, no arranjo do cone da alcachofra, do abacaxi e no desenrolar da samambaia, só para citar alguns exemplos.

3.4 O uso da Sequência Fibonacci

A sequência de Fibonacci possui uma característica altamente interessante para a solução pretendida, a simplicidade. Logo, resolvemos estabelecer uma relação entre a Sequência Fibonacci e a ausência de pedaços implementando duas sequências distintas e inversas com ponteiros que andam sempre juntos. Uma sequência é

responsável por aumentar o “estresse” a cada ausência de pedaço, enquanto a outra reduz a cada pedaço reproduzido com sucesso, formando uma idéia semelhante a de um fluxo de caixa. Veja um exemplo na Figura 3.3, nela podemos ver uma verificação a cada 8 segundos, onde nas 5 primeiras checagens os pedaços estavam ausentes e nas duas últimas os pedaços foram reproduzidos com sucesso, alcançando um nível de estresse igual a 10 (1+1+2+3+5-1-1). Diante de uma nova sequência de erros não voltamos a iniciar a sequência das ausências do zero, mas do ponto em que o ponteiro parou.

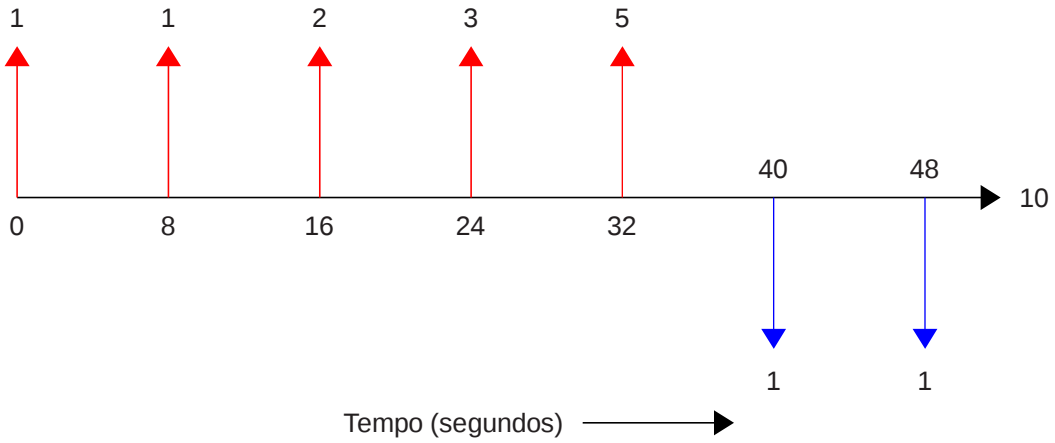

Figura 3.3: Uso da sequência Fibonacci no tempo

Para entender melhor veja o exemplo hipotético da Figura 3.4. Começamos com a Sequência Fibonacci Direta (SFD) para ausências de pedaços marcando 3 e com a Sequência Fibonacci Inversa (SFI) para reproduções bem sucedidas marcando 1. No momento seguinte, verificamos se o pedaço a reproduzir está presente, ao constatarmos a presença do mesmo, os ponteiros se movem no sentido da SFI e é subtraído no nível de estresse o valor marcado nela, ou seja, menos 1. No momento seguinte, verificamos se o pedaço à reproduzir está presente, ao constatarmos sua ausência, nesse instante os ponteiros se movem no sentido da SFD e é acrescentado ao nível de estresse o valor marcado nela, ou seja, mais 3.

O que aconteceria com a SFI se ocorressem mais duas ausências? A SFI continua

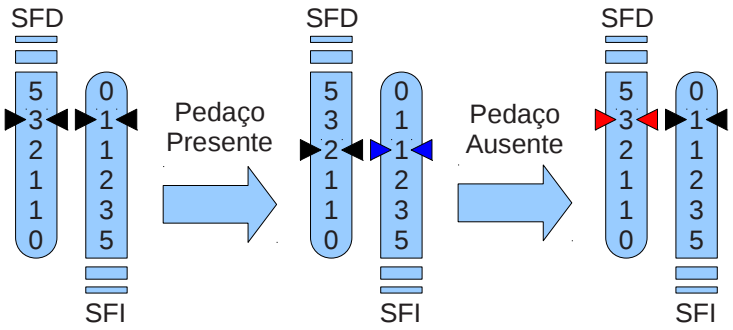

Figura 3.4: Uma sequência Fibonacci direta e outra inversa

acompanhando o ponteiro, veja a Figura 3.5. Seguindo essa lógica apresentamos na Figura 3.6 um gráfico que mostra uma série de dez ausências, seguida por uma série de dez reproduções. Desta vez podemos ver valores progressivamente maiores quando os erros estão próximos. Outra vantagem é que, partindo do princípio que as pessoas muito estressadas não se acalmam imediatamente, os valores caem lentamente quando a constatação das ausências ainda está recentes. Logo, a sequência Fibonacci representa melhor a percepção dos usuários, quando comparamos com os dois exemplos anteriores.

No trabalho de [Krishnan e Sitaraman 2012] podemos ver que muitos usuários desistem da visualização se o vídeo demorar apenas 2 segundos para iniciar. Porém, nos perguntamos se esses mesmos usuários desistiriam do vídeo se este apresentasse um corte ou pausa de 2 segundos, após uma hora de perfeita visualização. Acreditamos que eles não desistiriam, pois a percepção sobre a QoE já teria mudado em função do bom serviço prestado ao longo do tempo e como resultado, para este exemplo, temos uma pessoa mais tolerante a falhas.

Acreditamos que a variação da QoE ocorre nas mais váriadas situações envolvendo qualquer tipo de produto ou serviço, não apenas com a visualização de vídeo. Um exemplo pode ser visto no livro [Welch e Welch 2005], onde Jack Welch compara o tratamento dado a uma pessoa recém contratada com uma funcionária que está a mais tempo na empresa e que “acumulou créditos” ao longo de sua carreira na compania.

Vimos, por meio da Figura 3.6, que a A^2Q representa melhor a percepção dos usuários e agora podemos ver na Figura 3.7 que ela também considera o "acumulo créditos", ou seja, a variação da QoE ao longo do tempo. A Figura 3.7 ilustra uma série de vinte reproduções, seguida por uma série de dez ausências e, logo depois, dez reproduções. Repare que, em comparação a Figura 3.6, o nível de estresse baixa mais rápidamente na Figura 3.7, ou seja, o usuário está mais tolerante a falhas. Isso ocorreu porque, durante as vinte primeiras reproduções, os ponteiros contínuaram se deslocando na direção SFI, mesmo com o nível de estresse marcando zero. Após a série de dez ausências o SFI já estava em sua décima primeira casa, marcando 55 na Sequência Fibonacci, e com a reprodução seguinte o ponteiro avançou em SFI, subtraindo 89 do nível de estresse.

Em seu livro *The Wave Principle*, Elliott mostra que as "ondas" precedem tanto as altas quanto as quedas, por isso utilizamos duas sequências inversas [Elliott 1938]. Não utilizamos a proporção áurea para evitar valores fracionados.

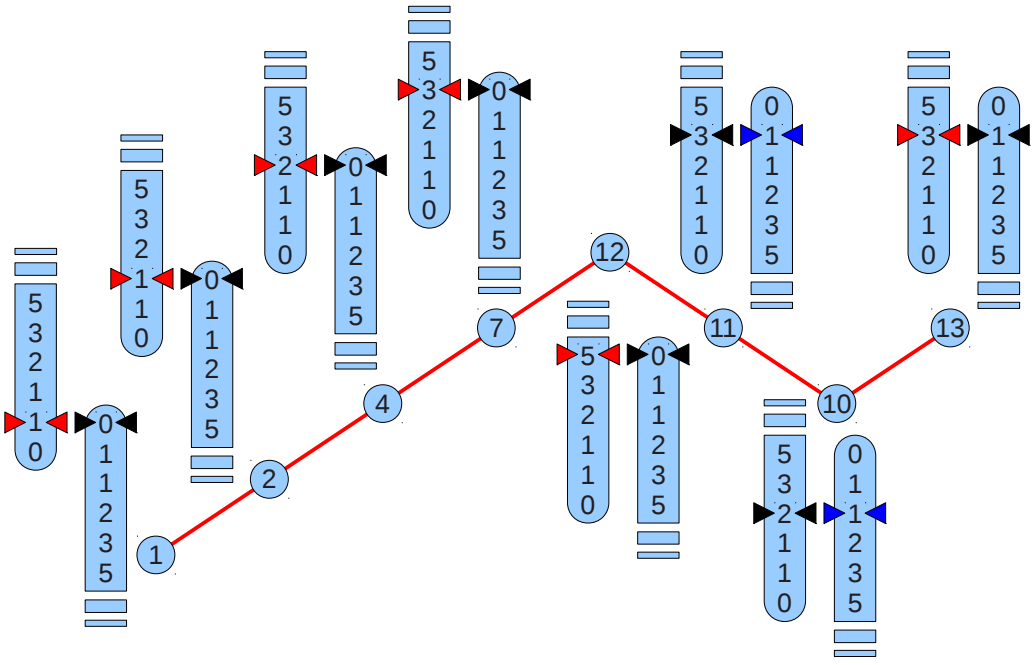

Figura 3.5: A movimentação das sequências e dos ponteiros

Outro fato interessante é que [Mwela e Adebomi 2010] chega a afirmar que a

avaliação da qualidade feita por um grupo de usuários decresce seguindo uma exponencial, conforme a qualidade piora. Em seu trabalho os usuários qualificavam a qualidade por meio da Tabela 2.3, mas os resultados tanto lembram uma exponencial quanto um decréscimo pela proporção áurea, ou seja, uma sequência Fibonacci.

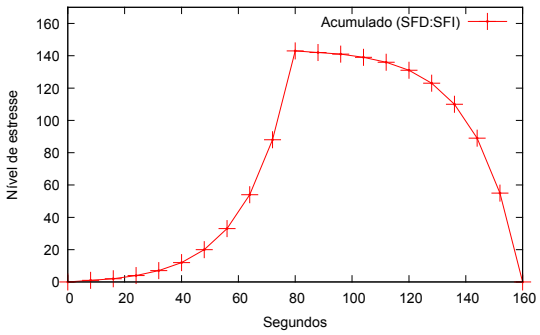

Figura 3.6: Exemplo de aumento do peso em função do tempo

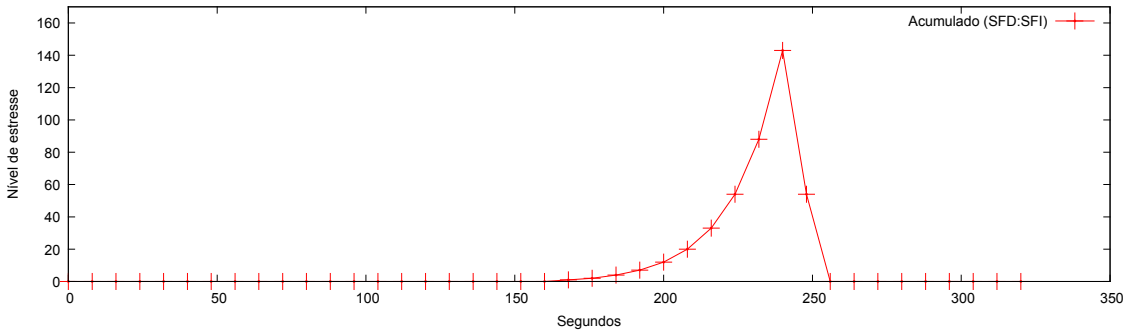

Figura 3.7: Exemplo de variação da QoE no tempo

Com a implementação do algoritmo decidimos que a estimativa do nível de estresse deve iniciar 40 segundos após a solicitação de um vídeo. Esse tempo foi escolhido por ser o mesmo necessário à chegada dos primeiros dados e formação de *buffer* no AnySee [Liao et al 2006], também por ser equivalente a 5 pedaços de vídeo, quando o mesmo possui uma taxa total de 512 Kbits/s e está dividido em pedaços de 512 KBytes, Fórmula 3.1. Escolhemos o AnySee por ele exigir apenas 40 segundos, enquanto que o famoso CoolStreaming necessita de 120 segundos.

Após o tempo de recebimento dos primeiros dados e *buffer* do vídeo o algoritmo começa a verificar, a cada 8 segundos, a presença do pedaço necessário à reprodução segundo dois critérios: cortes e pausas. Escolhemos esses dois critérios por representarem os cenários que encontramos tanto nos sistemas de TV convencionais (ausências

de sinal causam cortes na transmissão), quanto na maioria das transmissões pela Internet (vídeos do YouTube pausam diante da ausência de dados).

No corte, os pedaços são verificados em sequência ao longo do tempo, um após o outro. Diante de uma ausência a reprodução do vídeo para, espera 8 segundos (tempo de cada pedaço) e verifica a presença do próximo pedaço. A reprodução só continua quando o pedaço seguinte estiver presente, ou seja, o trecho do vídeo referente aos pedaços ausentes não é exibido.

Na pausa, em caso de ausência, a reprodução do vídeo para e espera 8 segundos. Depois, volta a verificar a presença do mesmo pedaço. Dessa forma o vídeo só continua após a chegada do pedaço ausente, prosseguindo do ponto onde parou. Cada critério, cortes e pausas, alimenta um nível de estresse.

Outro motivo para adotarmos essa distinção está nos algoritmos para seleção de pedaços, pois observamos que eles podem provocar níveis diferentes. Quando utilizamos o algoritmo tradicional do BitTorrent, Algoritmo para Seleção Aleatória de Pedaços (ASAP), o Nível de Estresse por Cortes (NEC) tende a ser baixo, enquanto que o Nível de Estresse por Pausas (NEP) tente a ser alto.

Vale lembrar que o BitTorrent não utiliza aleatoriedade total. Ao entrar em um *swarm*, o *leecher* solicita um pedaço escolhido aleatóriamente (modo *Random First Piece*). Depois de receber esse pedaço ele passa a solicitar o mais raro (modo *Rarest First*). Por último, quando todos os pedaços já foram solicitados, o *leecher* dispara requisições do que falta a todos os outros *peers* (modo *Endgame*). Apenas do ponto de vista do usuário a chegada dos pedaços segue uma ordem aleatória.

Para entender o que causa essa distorção considere um serviço de VoD, por exemplo: o NEP para diante de um pedaço ausente que pode vir a chegar somente no fim do *download*. Isso elevaria consideravelmente o seu valor, enquanto que o NEC continuaria no pedaço seguinte, podendo manter um valor menor devido aos pedaços presentes que encontra no caminho.

Com o Algoritmo para Seleção Sequencial de Pedaços (ASSP), situação que se assemelha a transmissão de vídeo ao vivo, o NEC pode ficar um pouco à frente do

ponto ideal para reprodução, nunca encontrando um pedaço presente. Isso eleva seu valor, enquanto que o NEP aguardaria o ponto ideial para reprodução, isto é, a chegada do próximo pedaço, mantendo com isso um valor menor, conforme Figura 3.8.

Neste trabalho, durante o uso do ASSP de [Harjoc 2013], não limitamos a capacidade de *upload* do *seeder* a 512 Kbits/s, nem consideramos a visualização do vídeo a partir do instante que o *peer* entra no *swarm*. Logo, os dados obtidos nessa etapa não refletem uma transmissão de vídeo ao vivo, mas uma transmissão em rajada.

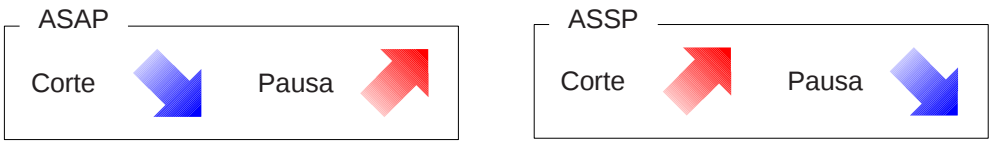

Figura 3.8: Seleção de pedaços e níveis de estresse

3.5 Utilizando a nova métrica de QoE para seleção de pares

A idéia inicial era a implementação de um *Super Tracker*, que seria responsável por receber e armazenar o nível de estresse de cada par da rede. O nível de estresse chegaria ao *Tracker* embarcado na solicitação da lista de pares e o nível de estresse seria atribuído à lista anteriormente enviada ao par. Com isso, diante de uma nova requisição, ele poderia optar por uma lista aleatória ou uma com nível de estresse menor que a do par solicitante. Mas, haviam alguns problemas: (1) a especificação do BitTorrent considera a requisição de listas ao *Tracker* uma operação onerosa; (2) a maioria dos programas, por padrão, solicita uma nova lista ao *Tracker* somente após 30 minutos; (3) os pares atualizam as próprias listas a partir da comunicação entre eles, assim o *Tracker* não poderia ter certeza se a lista enviada anteriormente foi responsável pelo nível de estresse apresentado pelo par. Seguir essa estratégia inicial envolveria muitas mudanças no protocolo BitTorrent e a intenção era reaproveitar o

máximo possível.

Durante a observação de uma lista altamente dinâmica, referente a um *Torrent* muito popular na época, foi possível perceber que muitos pares eram adicionados, mas excluídos momentos depois pelos mais diversos motivos. Entre os poucos pares que sempre apareciam na lista verificamos que se tratavam de máquinas geograficamente próximas (um agrupamento ocasional) e/ou com boa capacidade de transmissão (*supernodes*).

Em [Oliveira 2010] vemos a extensa influência dos *supernodes* sobre a segurança e qualidade de serviço da rede. Experimentos em ambiente real mostraram que um único *supernode* pode, através somente dos dados que envia, influenciar mais de 30% da banda de *download* da rede e mais de 50% dos pares.

Diante desses fatos, resolvemos unir o calculo do nível de estresse à dinâmica da lista de pares. O algoritmo começa guardando, a cada 8 segundos, a lista atual de pares. Logo depois de guardar a 10º lista é feita uma consolidação, onde apenas os pares que aparecem em todas permanecem, formando uma única lista que chamaremos de Lista de Pares Estáveis (LPE). Observe a Figura 3.9, nesse exemplo podemos ver que **A**, **E** e **F** aparecem em todas as listas. Utilizamos 10 listas porque uma quantidade menor poderia não ser representativa e uma maior faria com que o algoritmo demorasse para atuar. A quantidade ideal ainda precisa ser estudada.

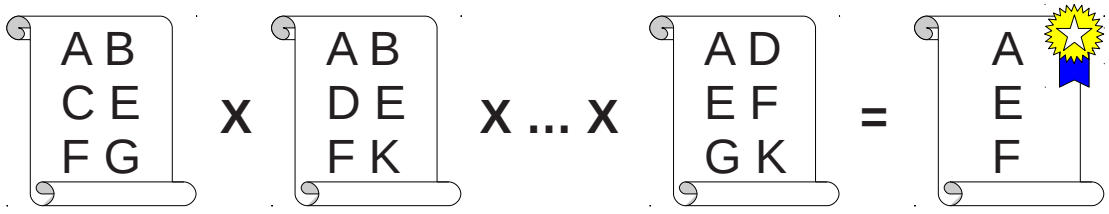

Figura 3.9: Geração da lista de pares estavéis

Esse processo é cíclico e se mantém durante toda a transmissão, ou seja, a cada 80 segundos uma nova LPE é elaborada. Nesse instante o algoritmo verifica o NEC

e o NEP, caso ambos estejam baixos nada será feito. Do contrario, uma mensagem perguntando o NEC e o NEP é enviada a cada *Leecher* da LPE. Os *Seeders* (no VoD) e os *Broadcasters* (na *live streaming*) não são considerados porque seus níveis de estresse tendem a zero e não há garantia que eles tenham uma LPE com bons fornecedores. Na Figura 3.10 vemos um exemplo em que o par **D** solicita os níveis de estresses aos integrantes da sua LPE (os pares **A**, **E** e **F**), por estar com nível de estresse médio em NEP, conforme Tabela 4.1.

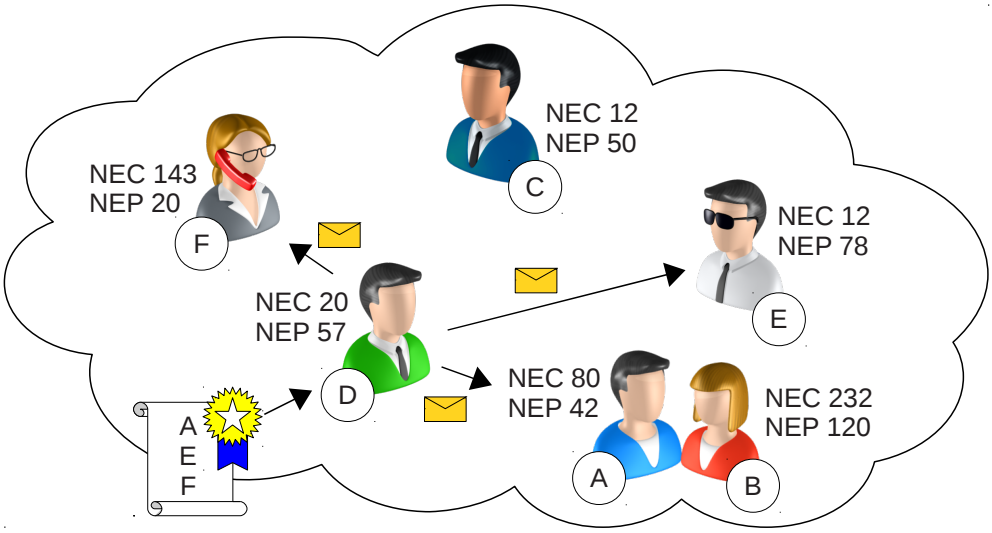

Figura 3.10: Solicitando os níveis de estresse aos pares da LPE

Depois de receber as respostas, o algoritmo compara seu maior valor entre NEC e NEP, com o correspondente de cada resposta, isto é, NEC se compara com NEC e NEP com NEP, apenas. Não existindo valor menor que o seu nada será feito, em caso contrário uma mensagem solicitando a LPE é enviada ao *Leecher* que apresentou o menor nível de estresse. Na Figura 3.11 vemos a continuação do nosso exemplo, onde os pares **A**, **E** e **F** respondem a solicitação do par **D** com seus níveis de estresse (**A** com NEP 42, **E** com NEP 78 e **F** com NEP 20). Na Figura 3.12 o par **D** verifica que o menor nível de estresse é o NEP 20 do par **F** e que este NEP é menor que seu próprio NEP, que está em 57. Diante disso o par **D** solicita ao par **F** a LPE dele.

Com a resposta de **F**, a LPE recebida é acrescentada a lista de pares, vide

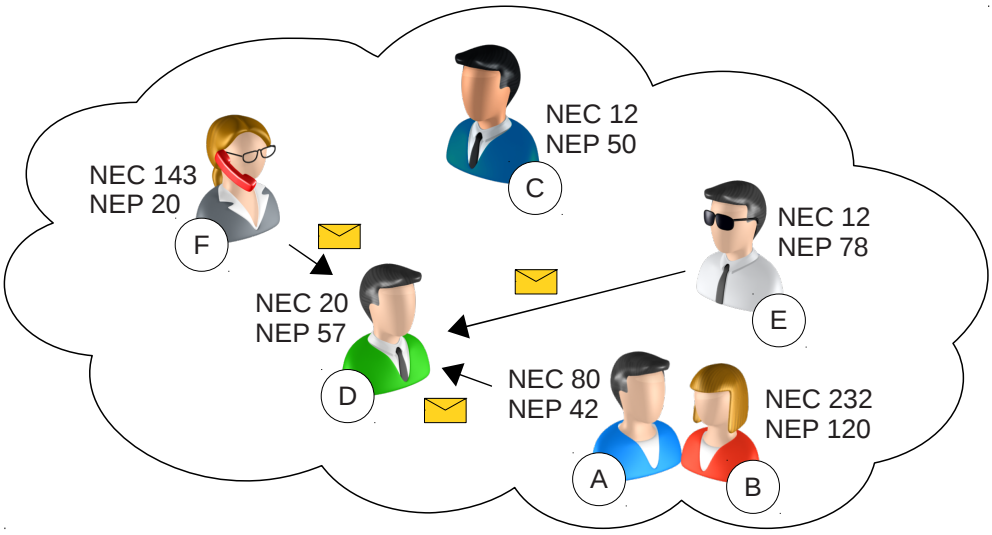

Figura 3.11: Coletando os NEC/NEP e verificando o menor

Figura 3.13, passando o solicitante (par **D**) a se comunicar com os novos pares adicionados, neste caso o par **C**, pois **D** é o próprio solicitante e **E** já faz parte da LPE de **D**.

Dessa forma, esperamos (1) reduzir o tempo de *download*, (2) reduzir as interrupções na reprodução do vídeo durante sua transmissão, (3) promover a chegada sequencial dos pedaços sem interferir no ASAP e sem provocar os problemas do ASSP, (4) produzir uma melhor e mais homogênea QoE entre os pares, (5) promover agrupamentos sem uso da taxa de *upload*, saltos, posição geográfica ou informações de fontes externas, (6) acelerar a formação de *supernodes* (pares colaborativos com grande capacidade de *upload*) e (7) desestimular *free riders* (pares pouco ou nada colaborativos).

A redução no tempo de *download* pode ser alcançada porque o algoritmo possibilita o encontro de bons fornecedores (pela troca das LPEs) indicados por quem está tendo bons resultados (baixo NEC/NEP).

A redução no tempo de *download* pode ser alcançada porque o algoritmo possibilita o encontro de bons fornecedores (pela troca da LPE) indicados por quem está tendo bons resultados (baixo NEC/NEP).

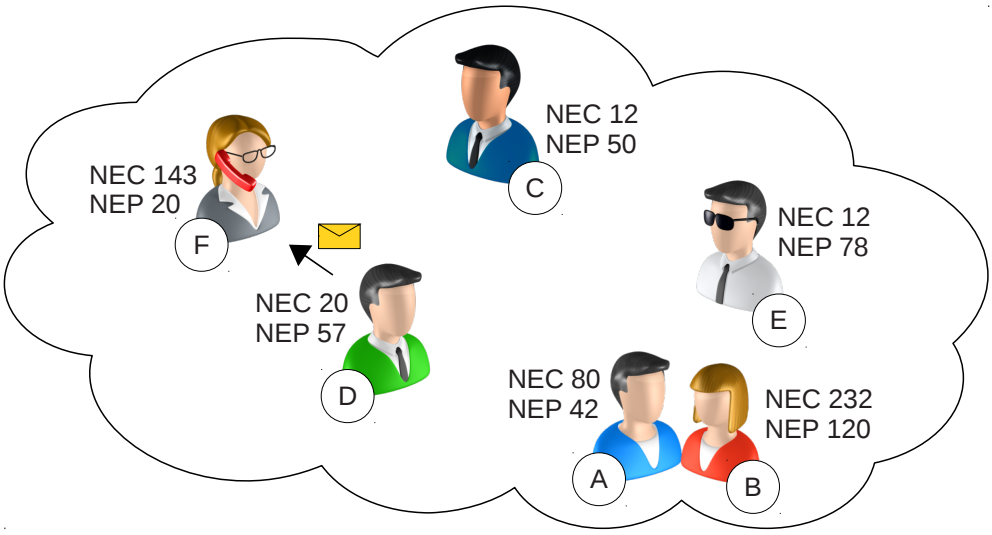

Figura 3.12: Solicitando a LPE do par com menor NEC/NEP

Uma reprodução contínua do vídeo com o mínimo de interrupções, durante sua transmissão, pode ser obtida porque os valores de NEC e NEP tem relação com a chegada do pedaço certo no momento adequado, isto é, quem tem um baixo NEC/NEP está recebendo os pedaços em uma sequência que permite uma reprodução contínua do vídeo. A organização dos grupos por NEC/NEP induz a uma seleção sequencial de pedaços, mas sem interferir no algoritmo do BitTorrent que trata esse assunto, por isso ele funciona tanto com o ASSP, quanto com o ASAP.

O algoritmo produz uma melhor e mais homogênea QoE entre os pares, porque a troca das LPEs tende a nivelar o NEC/NEP dos pares e induz a chegada sequencial dos pedaços.

A solução promove agrupamentos sem uso da taxa de *upload*, saltos, posição geográfica ou informações de fontes externas. O agrupamento ocorre pela troca das LPEs, pois os pares tendem a ter uma boa comunicação com os integrantes da LPE dos integrantes de sua própria LPE, desde que colhidos de um par com baixo NEC/NEP. Mesmo que isso não se revele uma verdade o algoritmo tradicional do BitTorrent trata o caso com um *choke*, ou seja, um par BitTorrent pode "afogar" seu vizinho, no momento que este não coopera com ele, interrompendo todos os *uploads*

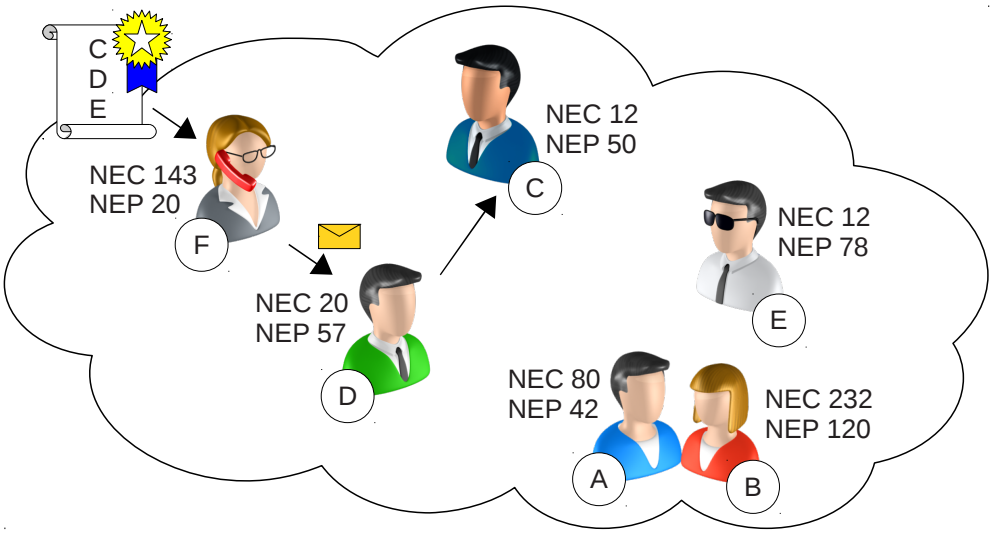

Figura 3.13: Adicionando os novos pares da LEP recebida

a este. Um exemplo seria o recebimento de uma LPE que contenha pares fora do perímetro de qualidade mínima para comunicação, para entender o que isso significa veja a Figura 3.14. Os círculos envolta dos pares **A** e **B** representam os perímetros de qualidade mínima para comunicação (que podem ter seus limites estabelecidos por distâncias geográficas, infra-estruturas das redes, capacidade dos pares, etc). Neste exemplo, o par **B** está respondendo a uma solicitação de LPE do par **A**. Repare que na lista consta o par **E** que se encontra fora do círculo do par **A**, ou seja, **E** está fora do alcance de **A** e por isso não irá ocorrer uma boa comunicação entre eles.

Acelerar a formação de *supernodes* (pares com grande capacidade de *upload*) é uma tarefa que pode ser alcançada partindo do princípio que esses pares são bons fornecedores e naturalmente aparecem nas LPEs que são trocadas pela rede. Por outro lado não descartamos a possibilidade que ocorra uma convergência a um pequeno grupo de pares (efeito manada) ocasionando uma sobrecarga. Por exemplo, quando um par fica saturado, e há excessiva concentração, o NEC/NEP podem aumentar e provocar uma migração para um mesmo novo par (sincronismo). Isso é apenas uma hipótse, esse caso não foi observado durante os experimentos.

Vale lembrar que o recebimento de uma nova LPE não provoca a substituição

da atual, por isso é pouco provável que ocorra um “efeito manada” no abandono de pares. A manutenção dos pares segue por conta da política tradicional do BitTorrent.

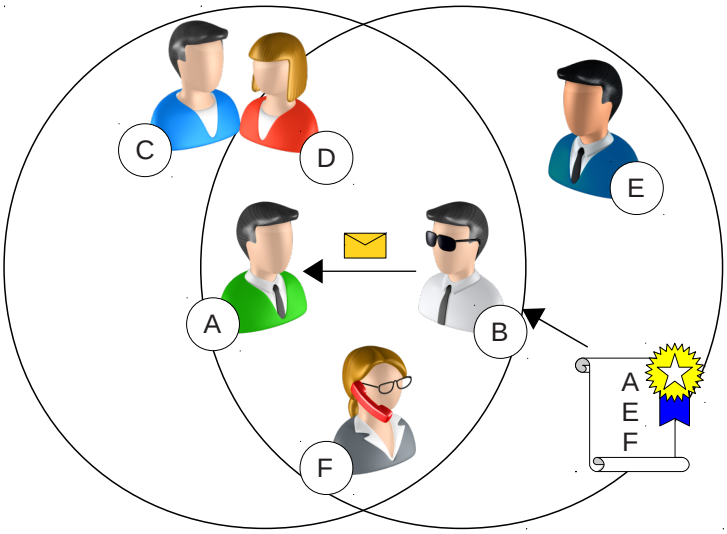

Figura 3.14: LEP com par fora do perímetro de qualidade mínima para comunicação

3.6 Induzindo à uma chegada sequencial dos pedaços

Um dos benefícios da nossa proposta é que ela pode induzir a chegada sequencial de pedaços sem interferir no ASAP e sem provocar os problemas do ASSP.

Para explicar como isso ocorre vamos usar um exemplo extremo. Veja a Figura 3.15 e considere que o par **A** solicitou o NEC/NEP dos pares **B**, **C** e **D**. Considere também que **B** está se comunicando com **E** e **F** e que eles possuem início do vídeo, que **C** está se comunicando com **G** e **I** e que eles possuem a parte central e que **D** está se comunicando com **H**, **J** e **K** e que eles possuem o final do vídeo. Nessa condições podemos deduzir que o par **B** vai apresentar níveis menores de NEC/NEP, por ter registrado presenças de pedaços em contra ponto as ausências registradas por **C** e **D**, pois o conteúdo recebido por eles não é o necessário ao momento da reprodução.

Logo, o par **A** solicita a LPE de **B** (Figura 3.16) para em seguida começar a receber pedaços referentes ao início do vídeo.

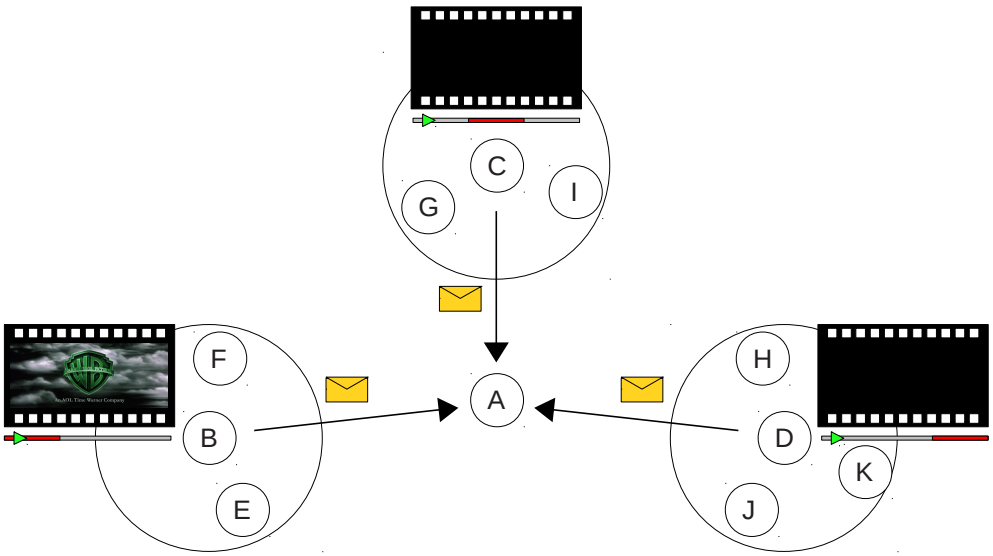

Figura 3.15: O NEC/NEP é menor no grupo que possui o início do vídeo

3.7 Resumo

Criamos as métricas Nível de Estresse por Cortes (NEC) e Nível de Estresse por Pausas (NEP), criamos uma Lista de Pares Estáveis (LPE) e atribuímos a ela os valores de NEC/NEP. Também criamos um algoritmo para seleção de pares que pesquisa entre os "amigos" (pares integrantes da LPE do par interessado em diminuir seus próprios níveis de NEC/NEP) aquele que apresenta menor NEC/NEP, para solicitar a LPE deste.

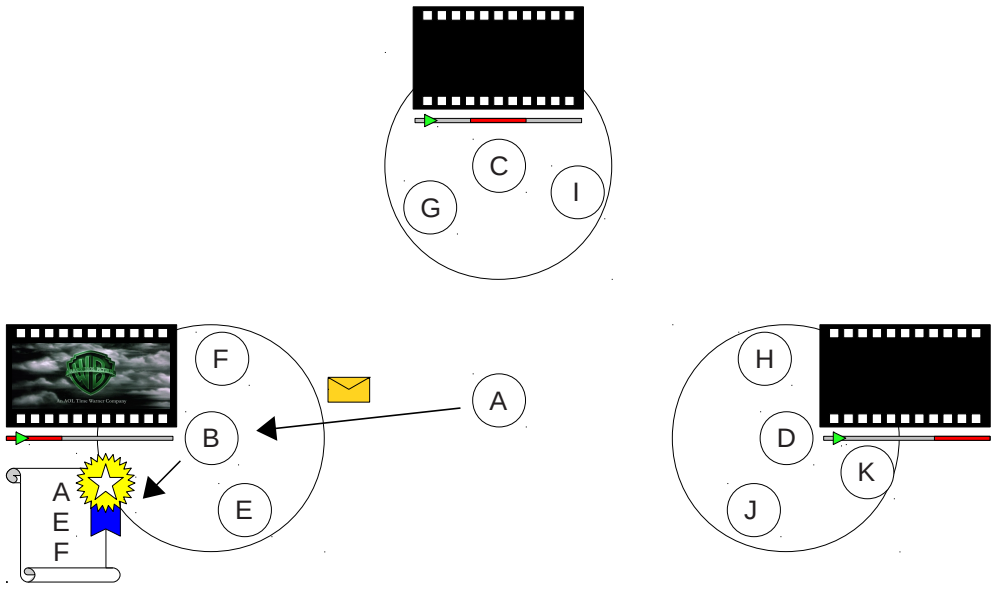

Figura 3.16: O algoritmo busca, indiretamente, a sequência que interessa a reprodução

Capítulo 4

Avaliação de desempenho da proposta

Para avaliar o desempenhoda nossa proposta realizamos um experimento no PlanetLab que emulou uma distribuição de vídeo com BitTorrent e a partir dessa emulação obtivemos métricas referentes à quailidade da distribuição do vídeo para cada solução alterando a quantidade de pares no enxame e o algoritmo de seleção de pedaços. Uma vez que a avaliação da qualidade da transmissão envolve apenas a verificação da presença ou ausência de determinado pedaço, a transmissão foi emulada de tal forma que o vídeo possuísse um tamanho compatível com a avaliação pretendida.

Comparamos nossa proposta com o modelo tradicional de BitTorrent e mais três propostas que implementam algoritmos para seleção de pares (Ono, P4P e Yukka). Cada uma dessas propostas foi desenvolvida para rodar como *PlugIn* no Vuze, o mesmo foi feito com o S4Q. O Ono utiliza informações de CDNs, o P4P coleta informações junto aos ISPs e o Yukka usa os IPs para consultar o banco de dados das RIRs e com esses dados avaliar a proximidade dos pares.

4.1 Ambiente

O ambiente escolhido foi o PlanetLab (ver Seção 4.1.1) e o critério para seleção das máquinas foi por *nodes* da autoridade PLC (PlanetLab norte-americano), que

apresentaram *status boot* por mais de 20 dias, arquitetura i386, IP listado e sistema operacional Fedora 8. Com isso, reunimos 368 MVs (máquinas virtuais) no *slice*. A preferência pelo Fedora 8 está relacionada ao número de máquinas, pois escolhemos o Sistema Operacional (SO) que estava presente no maior número delas.

4.1.1 PlanetLab

O PlanetLab é uma rede de pesquisa mundial que permite o desenvolvimento de novos serviços de rede. Atualmente conta com mais de 1100 nós abrigados em mais de 500 locais, proporcionando recursos de processamento e armazenamento distribuídos para o desenvolvimento de projetos em redes de grande alcance, formando um ambiente que dificilmente poderia ser montado por uma única instituição. Essa estrutura permite desenvolver e testar experimentalmente, sob condições reais de rede, aplicações e serviços de grande escala [PlanetLab 2012, RNP 2013].

4.1.2 Limitando a capacidade de transmissão

Começamos com a verificação da latência (*Round Trip Time - RTT*), usando o comando *ping*, com uma sonda de 64 *Bytes*, e saltos, com *traceroute*, de um par para cada um dos outros.

Colhido o resultado, verificamos conexões onde (1) a distância em saltos na ida e na volta eram igual a 1, e (2) a MV (máquina virtual) de origem possuia o mesmo número IP da MV de destino. Esses fatores podem indicar máquinas virtuais diferentes em uma mesma máquina física. Logo, essas MVs foram consideradas iguais e mantivemos apenas uma delas na relação de *nodes* utilizada no experimento, evitando com isso casos com baixíssima e irreal latência.

Porém, seguindo esse critério, 207 *nodes* precisariam ser excluídos.

Considerando que 512 Kbits/s é suficiente para receber vídeo, optamos por manter todas as MVs e limitar a velocidade à 1 Mbits/s, mas depois de alguns testes julgamos razoável mudar o limite para 8 Mbits/s, ou seja, 1 MByte/s. Essa velocidade foi escolhida por ser suportada pelos maiores provedores de Internet no Brasil e por ser

suficiente para atender a taxa mínima de 512 Kbits/s, mesmo com o controle de tráfego do TCP. Escolhemos a taxa de 512 Kbits/s por ser a mesma requisitada pela NetFlix para filmes com baixa resolução [NetFlix 2013].

4.2 Experimento em ambiente real controlado

Com o GoalBit, sistema P2P para vídeo ao vivo, existem trabalhos envolvendo simulações com 300 pares [Barrios et al 2011] e com 60 pares em ambiente real [Bertinat et al 2009].

Nosso experimento foi executado em grupos de 25, 50, 75, 100 e 125 *peers*, com um *tracker* e um *seeder*.

No *tracker* o tráfego foi verificado com o *sniffer* TCPDump [Garcia 2012]. *Sniffing*, em rede de computadores, é o procedimento realizado por um *software* ou *hardware* com o propósito de interceptar e registrar o tráfego de dados em uma rede de computadores. Conforme o fluxo de dados trafega na rede, o *sniffer* captura cada pacote e eventualmente decodifica e analisa o seu conteúdo.

Já no *seeder* e nos *leechers* os dados estatíticos foram extraídos do arquivo "Azureus_stats.xml". O Vuze, por padrão, gera esse arquivo a cada 30 segundos, quando as estatísticas estão ativadas. Desse arquivo foram extraídos o *downloaded* e o *uploaded* de cada *peer* com o respectivo IP. Essa coleta aconteceu durante 2.000 segundos, mesmo tempo utilizado no trabalho com o GoalBit [Bertinat et al 2009]. Também foi coletado diretamente do sistema operacional os níveis de consumo da memória em cada par.

Nos experimentos realizados, os pares que se tornaram *seeders* foram mantidos até o término dos 2.000 segundos.

4.3 A coleta e processamento dos dados

Houve um maior trabalho na redução do tamanho dos *traces* gerados por cada par, pois esses arquivos foram produzidos em cada uma das soluções, para cada grupo,

para cada algoritmo de seleção de pedaços e cada um desses testes foi repetido 30 vezes para respeitar a Lei dos Grandes Números [Alzina, Sarriera e Martinez 2004].

Cada *trace* tem em média 70 KBytes, com isso chega-se a uma coleta de pouco mais de 7,61 GBytes distribuídos em cerca de 194 mil arquivos.

Após a coleta, os dados foram inseridos em um banco de dados MySQL e na sequência realizamos a extração das informações em tabelas estatísticas [Oracle 2012]. Essa operação levou cerca de 750 horas para processar meio bilhão de registros, 30% do tempo total do experimento. O tempo dedicado a produção dos *traces* foi reduzido com a execução de rodadas simultâneas.

Para inserção dos *traces* no banco de dados e extração das informações, construímos um programa em Java que realizava essas tarefas e gerava arquivos adequados ao GNUPlot.

4.4 Unidade de controle

Para controlar o experimento utilizamos o serviço Linode 512 [Linode 2012] por fornecer uma máquina virtual com bom espaço em disco, boa capacidade de *download*, disponibilidade 24/7 e a possibilidade de acesso remoto de qualquer lugar sem a necessidade de SSH. O controlador contou com o sistema operacional Ubuntu 12.04 LTS 64 Bits e Xampp 1.8, em um servidor localizado em Atlanta, GA, USA.

Para envio dos arquivos necessários a cada um dos *nodes* optamos pelo DropBox [DropBox 2012], por ser um serviço de nuvem gratuito com boa capacidade de *upload* e que não faz restrições ao comando *wget*.

4.5 O *PlugIn* para nova métrica

Também coletamos, através de um *PlugIn* (desenvolvido para esse fim específico), o número de ausências de pedaços tanto no critério cortes quanto no pausas.

Seguindo o trabalho sobre o AnySee [Liao et al 2006], definimos que somente após 40 segundos o *PlugIn* inicia a métrica.

Considerando que é uma boa prática utilizar pedaços de 512 KBytes, no protocolo BitTorrent [BitTorrent 2012, Theory 2013], podemos concluir que cada pedaço irá conter 8 segundos de vídeo e que, em tese, 5 pedaços serão recebidos durante a chegada dos primeiros dados e a formação do *buffer* (Fórmulas 3.1).

No critério corte, quando uma ausência de pedaço é identificada, o contador aguarda 8 segundos (vide Fórmula 3.1) e tenta continuar com o próximo pedaço, ou seja, o pedaço que não estava pronto no momento adequado é pulado na reprodução.

No critério pausa, ao encontrar uma ausência o contador aguarda 8 segundos e verifica novamente o mesmo pedaço, com o propósito de continuar o vídeo do ponto onde parou, comportamento semelhante ao adotado no YouTube.

No trabalho com o AnySee vemos que o tempo de vida médio dos pares é exponencialmente distribuído e no estudo de [Faria 2010] podemos ver que a entrada dos pares também segue uma exponencial. Considerando essas informações a entrada dos pares no enxame foi projetada para seguir uma exponencial, disbribuindo a entrada nos primeiros 200 segundos com taxa média de 0,05 (λ).

4.6 Os Níveis de Estresse e a QoE

Estabelecemos uma escala de estresse com três níveis (baixo, médio e alto). Como mostrado na Tabela 4.1. Assumimos que estresse baixo equivale uma espera de pouco mais de 1 minuto, estresse médio fica entre 1 e 2 minutos e estresse alto fica com valores maiores que 2 minutos.

Tabela 4.1: Escala de estresse com três níveis

Níveis de Estresse	Ausências de Pedaços	Acumulado de Fibonacci
Baixo	0 ⊣ 8	0 ⊣ 54
Médio	8 ⊣ 16	54 ⊣ 2.583
Alto	16 ⊣ 24	2.583 ⊣ 121.392

O nível alto foi limitado para que o acumulado da sequência Fibonacci não gerasse números excessivamente grandes e difíceis de trabalhar nos gráficos. Outro fato que ajuda a justificar o limite em 24 pedaços ausentes são os percentuais utilizados

nos trabalhos de [Agboma, Smy e Liotta 2008, Mwela e Adebomi 2010], onde vemos que 10% em perda de pacotes produz níveis baixíssimos de MOS.

O tamanho mínimo do vídeo para o experimento é de 127 MBytes. Esse tamanho foi definido usando a Fórmula 4.1, onde multiplicamos $\boldsymbol{t}$, o tempo de um único experimento (2.000 segundos), por $\boldsymbol{y}$, a taxa do vídeo (512 Kbits/s), depois dividimos por $\boldsymbol{z}$, 1 MByte devidamente convertido para Kbits, para obter $\boldsymbol{x}$, tamanho do vídeo.

$$x = \frac{t \times y}{z} = \frac{2000 \times 512}{8000} \approx 127 MBytes \tag{4.1}$$

Considerando o arquivo do nosso experimento (127 MBytes) e o tempo de duração do experimento (2000 segundos), podemos concluir que 10% de perda de pacotes na transmissão desse vídeo equivale, respectivamente, a 12,7 MBytes e a 200 segundos.

Sabemos que o vídeo é dividido em pedaços de 512 KBytes e que cada pedaço abrange 8 segundos. Dividindo 12,7 MBytes por 512 KBytes ou 200 segundos por 8, chegamos ao mesmo resultado, 25 pedaços. Logo, podemos perceber que existe um relação entre o limite estabelecido para o maior nível de estresse e a percepção dos usuários (QoE), segundo o levantamento feito no trabalho de [Mwela e Adebomi 2010].

Capítulo 5

Análise dos resultados

Com a intenção de provar que o algoritmo para seleção de pares implementado neste trabalho possibilita uma transmissão de vídeo mais rápida (menor tempo de *download*), menos interrupções (menos ausência de pedaços e nível de estresse baixo) e com menos esforço (menos *upload* do *seeder* e consumo de memória), realizamos a extração das medidas apresentadas nas próximas seções.

5.1 *Status* das máquinas virtuais

Vimos anteriormente (Seção 4.1) como foi o processo de escolha das MVs utilizadas durante o experimento. Apesar de todo o cuidado, não é possível prever se uma MV vai estará disponível durante todo o processo. O *script* responsável pelo experimento tenta reduzir esse risco, porém algumas máquinas ficaram *off-line* em algum momento, seja já no inicio (*start* do Vuze), durante ou no final (coleta dos *traces*) do experimento.

As máquinas em que não foi possível obter os *traces* foram consideradas "desligadas". As que geraram *traces*, mas não estabeleceram conexão com outras máquinas foram classificadas como "*download* zero". As que não completaram o *download* (consequentemente, apresentaram uma taxa de *download* inferior a 512 Kbits/s) como "*download* parcial" e as demais como "*download* completo".

Para verificar a igualdade de condições em que ocorreram os experimentos, avaliamos a existência de possíveis distorções comparando os quatro conjuntos de

MVs, para cada uma das soluções, com cada um dos algoritmos para seleção de pedaços.

Nas Figuras 5.1, 5.2, 5.3, 5.4 e 5.5 podemos visualizar a existência de uma igualdade de condições entre as soluções com seleção aleatória de pedaços (ASAP) e sequencial de pedaços (ASSP). Note que, no ASSP as máquinas ficaram mais lentas. Isto pode ser visto pelo aumento das máquinas com *download* parcial. Isso justifica o argumento de alguns especialistas que afirmam ser ruim para o BitTorrent os ASSP [Vuze 2013e].

No experimento com 125 máquinas houve um aumento generalizado da lentidão, tanto com ASAP, quanto no ASSP, conforme pode ser visto na Figura 5.5. Esse aumento não tem relação com o método estatístico, pois esses dados são parâmetros e não estatísticas.

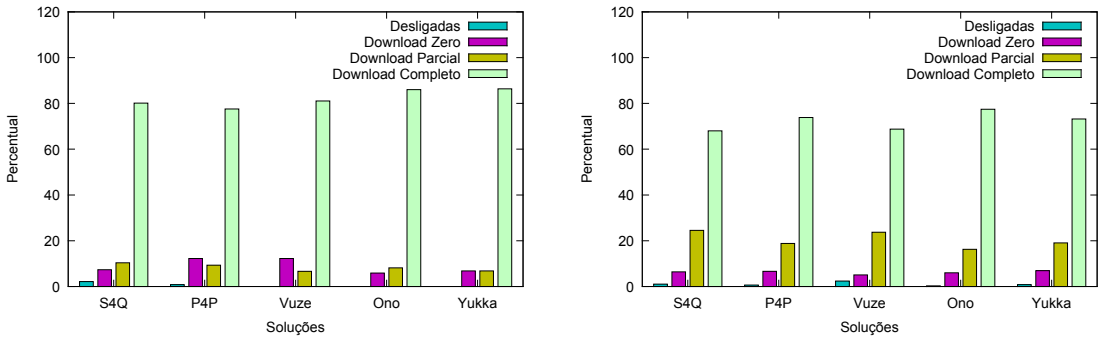

Figura 5.1: Situação das máquinas do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

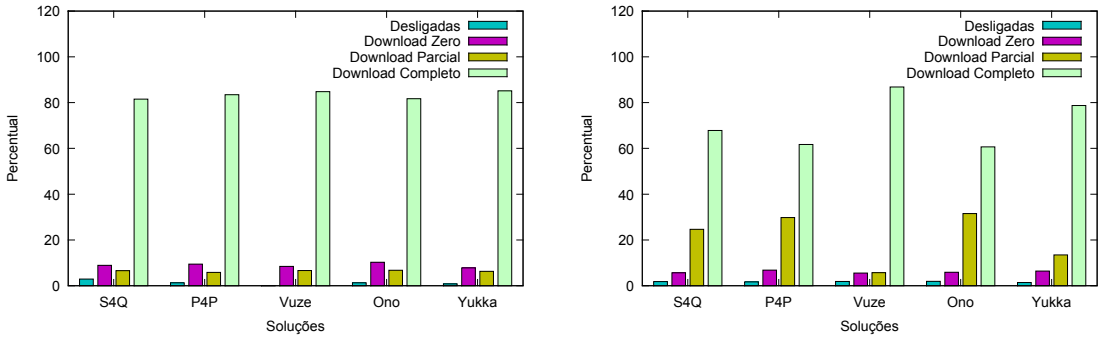

Figura 5.2: Situação das máquinas do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

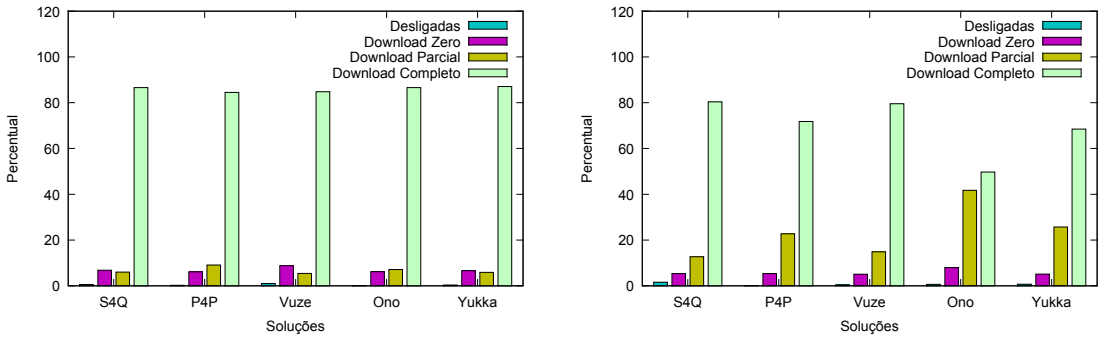

Figura 5.3: Situação das máquinas do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

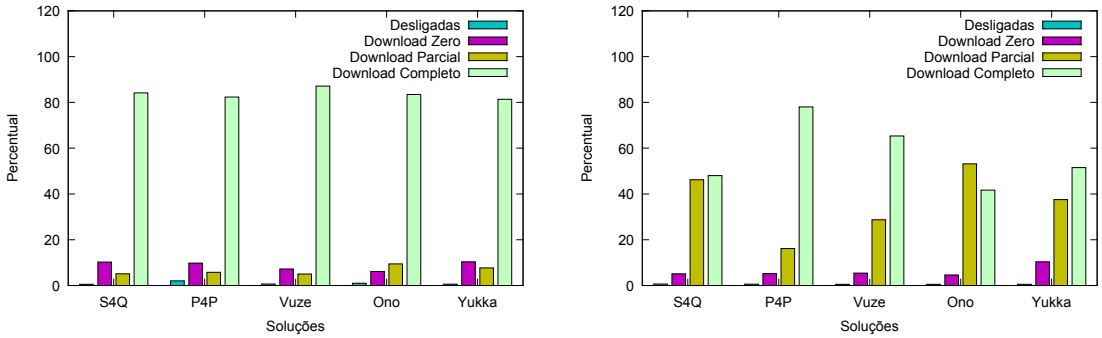

Figura 5.4: Situação das máquinas do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

5.2 Tempo médio de *download*

No levantamento do tempo médio de *download* podemos ver que, entre as máquinas que completaram o *download*, o tempo médio chega a dobrar em pequenos grupos com a utilização do ASSP, Figura 5.6. Também vemos que as soluções alcançam resultados muito próximos.

O tempo médio de *download* engloba o tempo médio para início do *download*, mas não podemos afirmar que há efeito de um sobre o outro (veja Seção 5.3). Uma das razões pode ser o fato desta métrica ter considerado apenas as máquinas do conjunto *download* completo, enquanto que a outra métrica avaliou os conjuntos *download* parcial e completo.

O intervalo de confiança, com nível de confiança em 95%, para o tempo médio de *download* com ASAP oscilou entre 2,13% (Vuze) e 5,49% (S4Q). Com ASSP os valores foram de 2,31% (Vuze) até 6,00% (Vuze e S4Q). Para o tempo médio de

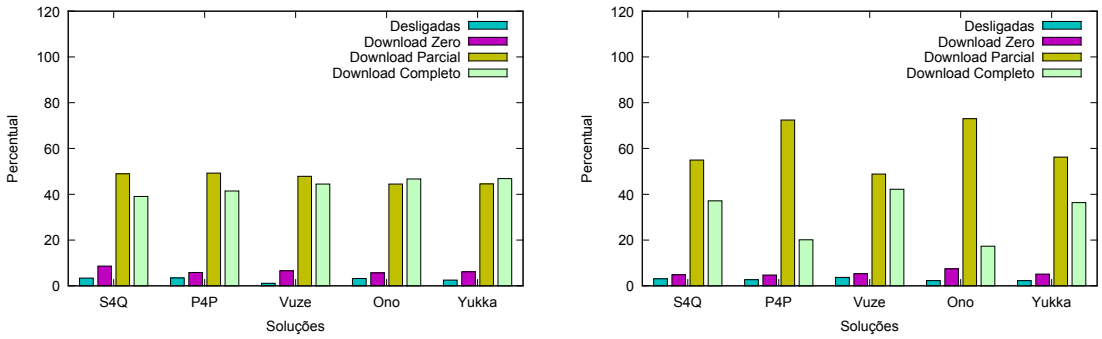

Figura 5.5: Situação das máquinas do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

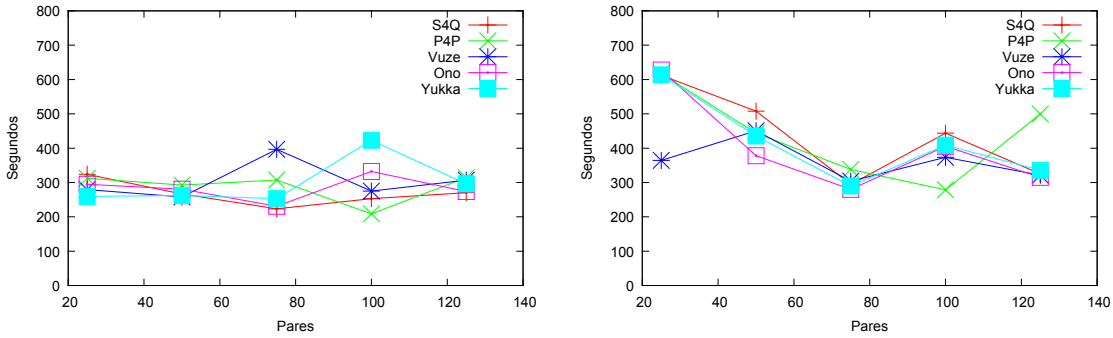

Figura 5.6: Tempo médio de *download* com ASAP (esquerda) e ASSP (direita)

Tabela 5.1: Tempo médio de *download* e intervalo de confiança (nível de confiança em 95%) com ASAP

Total	S4Q		P4P		Vuze		Ono		Yukka	
Pares	μ	IC	μ	IC	μ	IC	μ	IC	μ	IC
25	324,13	17,78	312,27	14,71	279,24	12,23	295,02	10,93	259,68	10,95
50	266,53	9,69	292,41	11,00	257,96	8,83	280,47	9,69	262,36	8,66
75	223,52	7,54	307,17	15,49	396,71	8,46	230,95	9,43	252,77	7,90
100	253,03	8,17	208,79	5,49	275,02	7,77	332,32	15,15	423,16	16,54
125	270,47	8,23	314,45	9,28	306,92	8,27	273,81	8,31	296,24	7,00

Tabela 5.2: Tempo médio de *download* e intervalo de confiança (nível de confiança em 95%) com ASSP

Total	S4Q		P4P		Vuze		Ono		Yukka	
Pares	μ	IC	μ	IC	μ	IC	μ	IC	μ	IC
25	612,34	36,75	617,50	30,73	364,51	21,88	627,36	34,76	613,34	30,72
50	507,76	18,46	443,89	17,71	450,59	16,04	378,18	17,16	435,52	14,83
75	294,48	8,45	338,06	11,38	303,88	10,38	279,96	10,37	289,71	10,85
100	443,75	18,26	278,53	7,89	373,46	13,75	405,51	12,63	408,57	11,96
125	325,18	9,39	499,61	29,08	321,41	7,43	315,19	14,68	335,69	8,98

início do *download* com ASAP houve uma variação entre 3,95% (Vuze) e 11,52% (Ono). Com ASSP os valores foram de 4,28% (P4P) até 11,81% (Ono). Os valores foram omitidos dos gráficos para maior clareza na leitura.

5.3 Tempo médio para iniciar o *download*

Na coleta do tempo médio para início do *download* foi possível ver que tanto no uso do ASAP, quanto no ASSP, as soluções trabalham na faixa entre 50 e 150 segundos (Figura 5.7). Para esta métrica consideramos as máquinas dos conjuntos *download* parcial e completo.

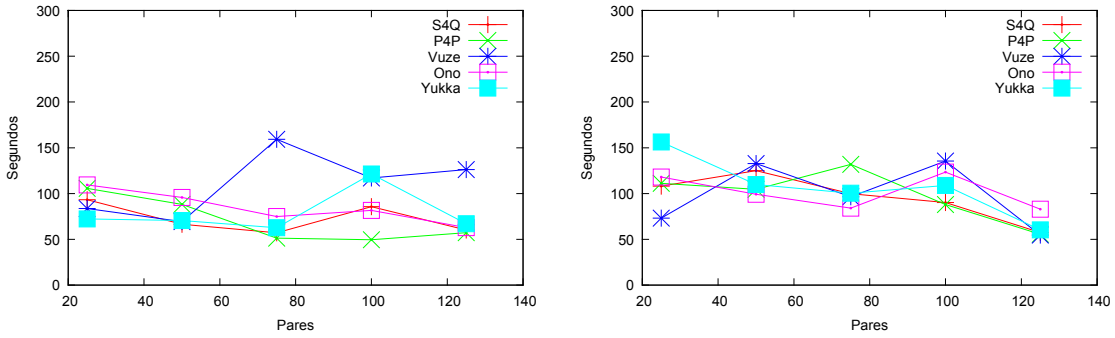

Figura 5.7: Tempo médio para início do *download* com ASAP (esquerda) e ASSP (direita)

Tabela 5.3: Tempo médio para iniciar o *download* e intervalo de confiança (nível de confiança em 95%) com ASAP

Total	S4Q		P4P		Vuze		Ono		Yukka	
Pares	μ	IC	μ	IC	μ	IC	μ	IC	μ	IC
25	93,18	9,74	105,58	10,23	83,67	6,51	109,38	12,60	72,30	7,28
50	66,5	4,94	88,13	8,29	69,29	5,23	95,65	8,97	70,46	6,50
75	57,12	3,49	51,22	3,61	159,19	6,70	74,88	8,52	62,63	3,86
100	85,60	5,77	49,55	3,22	116,94	6,28	81,58	6,86	121,30	7,14
125	60,21	4,75	57,22	2,50	126,17	4,99	62,74	4,88	67,03	3,15

Tabela 5.4: Tempo médio para iniciar o *download* e intervalo de confiança (nível de confiança em 95%) com ASSP

Total	S4Q		P4P		Vuze		Ono		Yukka	
Pares	μ	IC	μ	IC	μ	IC	μ	IC	μ	IC
25	108,15	11,07	111,23	10,11	73,09	6,64	117,99	13,93	156,33	10,38
50	124,98	8,00	104,61	8,60	132,79	8,84	99,26	9,99	109,92	8,16
75	99,78	6,48	131,92	8,35	96,42	6,76	84,02	7,99	100,42	6,98
100	90,13	5,30	87,96	5,45	135,53	9,61	123,35	8,37	108,77	7,33
125	57,47	2,58	55,58	2,38	54,61	2,43	82,86	5,35	60,28	2,85

5.4 Máquinas que concluíram o *download* ao longo do tempo

As Figuras 5.8, 5.9, 5.10, 5.11 e 5.12 mostram o percentual de máquinas que completaram o *download* durante o experimento usando ASAP e ASSP, para 25, 50, 75, 100 e 125 pares. Podemos notar dois pontos, o primeiro é que as máquinas levam mais tempo para concluir o *download* com ASSP (fato já mostrado na Seção 5.2), isso pode ser visto pela demora no nivelamento do gráfico. O segundo ponto, com ASSP, esta na diferença quantitativa de máquinas que completam o *download* dentro do tempo do experimento, entre cada solução (fato também apontado na Seção 5.9), com ASAP as soluções trabalham de forma muito semelhante.

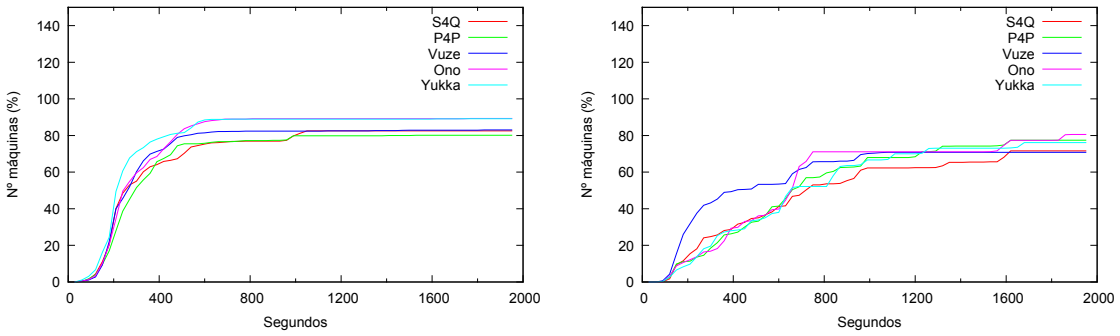

Figura 5.8: *Download* completo ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

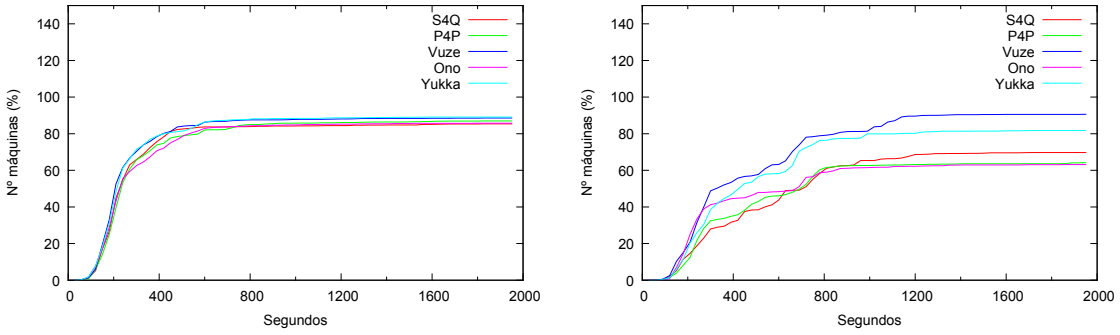

Figura 5.9: *Download* completo ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

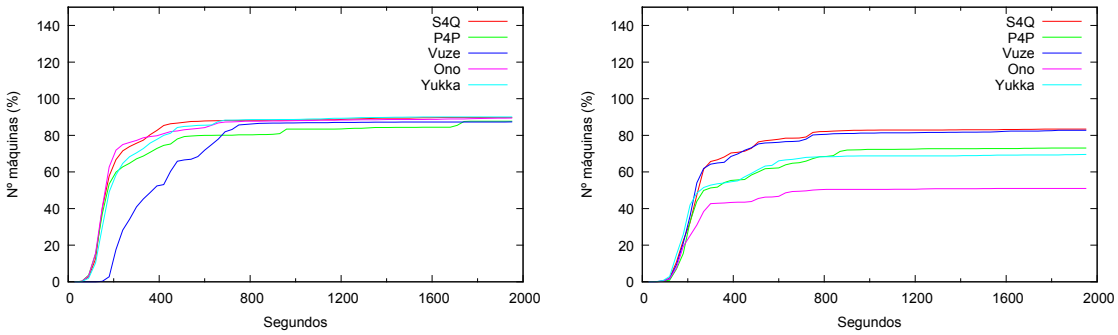

Figura 5.10: *Download* completo ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

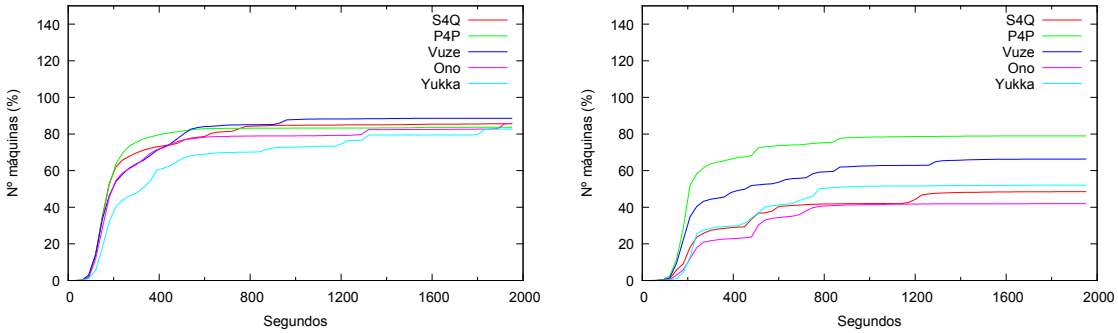

Figura 5.11: *Download* completo ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

5.5 *Upload* do *seeder*

O *upload* exigido do *seeder* pode ser visto nas Figuras 5.13, 5.14, 5.15, 5.16 e 5.17. Comparando a utilização do ASAP com ASSP, podemos ver que o *seeder* envia mais dados quando utilizamos ASSP. Outro fato é que o aumento no número de pares não produz um aumento de transmissão do *seeder*, chega a ocorrer o inverso na Figura 5.17. Porém, como a Figura 5.5 pode estar indicando lentidão na rede ou outro tipo de problema, não é possivel afirmar que um aumento do número de pares diminui a transmissão a partir do *seeder*.

5.6 Consumo de memória

As soluções usam em média 50 MBytes de memória, conforme pode ser visto nas Tabelas 5.15, 5.16 e 5.17, e 100 MBytes de memória virtual (*Virtual Memory Size - VSZ*), veja as Tabelas 5.18, 5.19 e 5.20. O Ono apresentou um grande consumo

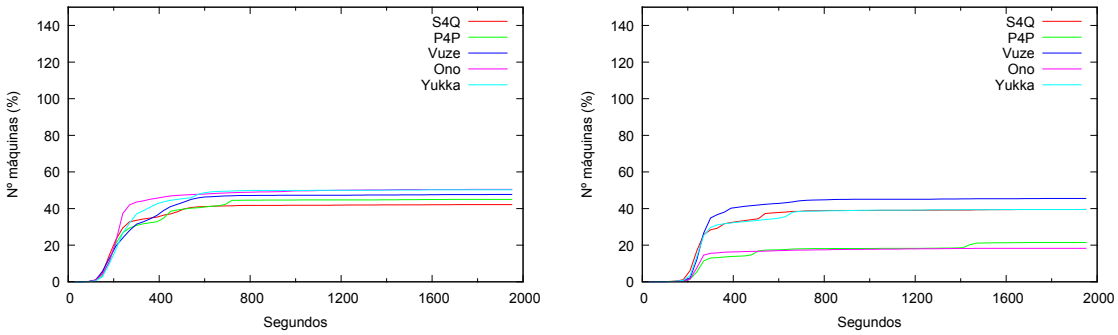

Figura 5.12: *Download* completo ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

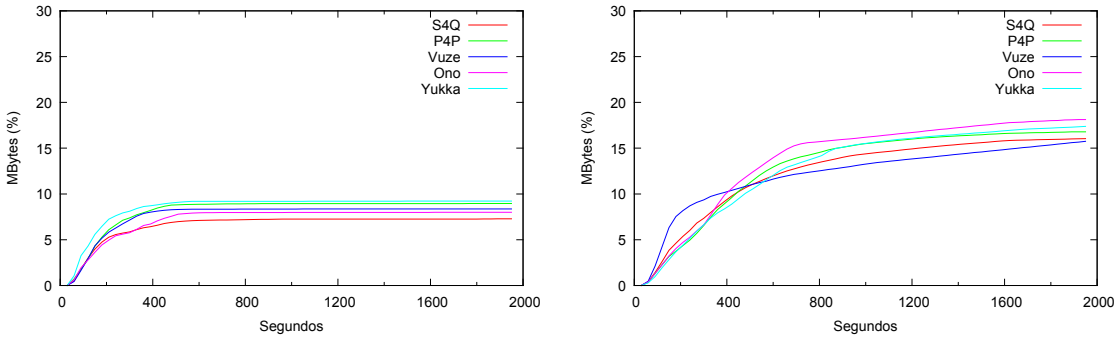

Figura 5.13: *Upload* do *seeder* ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

médio de memória física não *swapped* (*Resident Set Size - RSS*) durante a execução de suas tarefas, tanto nos *trackers* (Figura 5.18 e Tabela 5.21), quanto nos *seeders* (Figura 5.19 e Tabela 5.22) e nos *leechers* (Figura 5.20 e Tabela 5.23). O consumo de memória foi maior nos *seeders* e *leechers* do grupo com 125 pares.

Não foi possível precisar as razões que levam a um maior consumo de memória, pois elas pode variar do gerenciamento das listas de pares à erros na implementação das soluções.

5.7 Ausências de pedaços ao longo do tempo

O acumulado dos pedaços ausentes ao longo do experimento, no critério corte, pode ser visto nas Figuras 5.21, 5.23, 5.25, 5.27 e 5.29. Observando esses gráficos vemos que o S4Q (implementação do algoritmo para seleção apresentado neste trabalho) tende a ter menos ausências. Com ASSP houve uma redução geral na ausência de

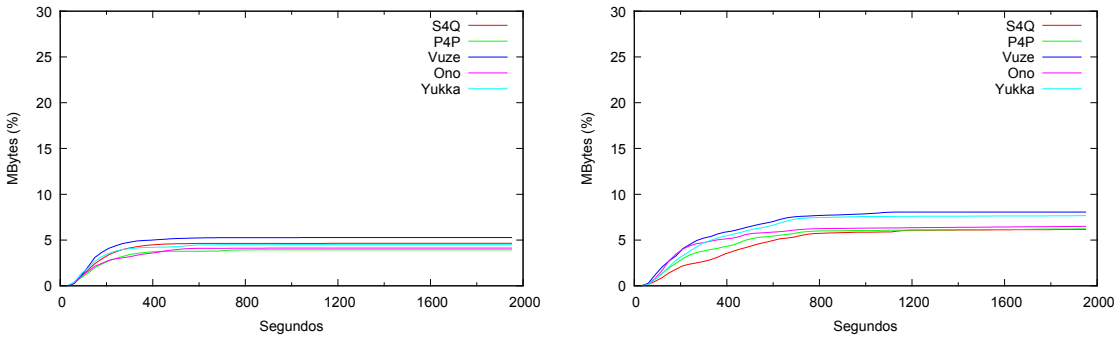

Figura 5.14: *Upload* do *seeder* ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

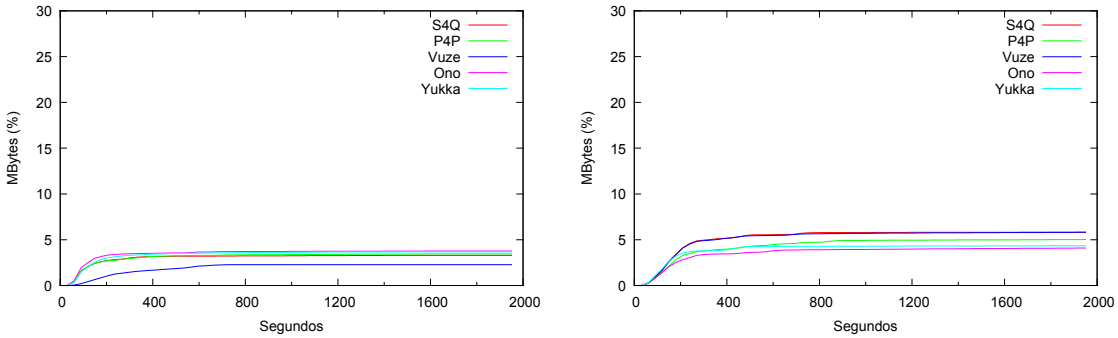

Figura 5.15: *Upload* do *seeder* ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

pedaços.

Os resultados referentes a ausência de pedaços e nível de estresse representam uma amostragem, onde apenas os dados da última máquina a entrar no grupo, de cada repetição, foram processados. Com ASSP os dois critérios (corte e pausa) apresentaram ausências acumuladas de pedaços muito parecidas, gerando níveis de estresse com a mesma aparência (veja na Seção 5.8)

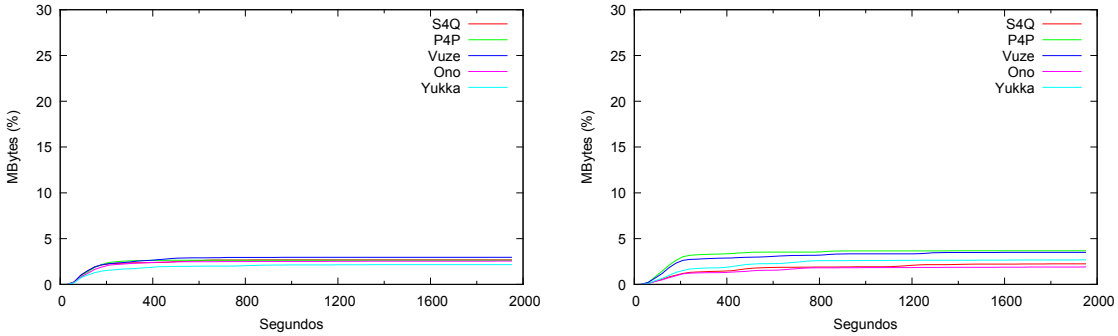

Figura 5.16: *Upload* do *seeder* ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

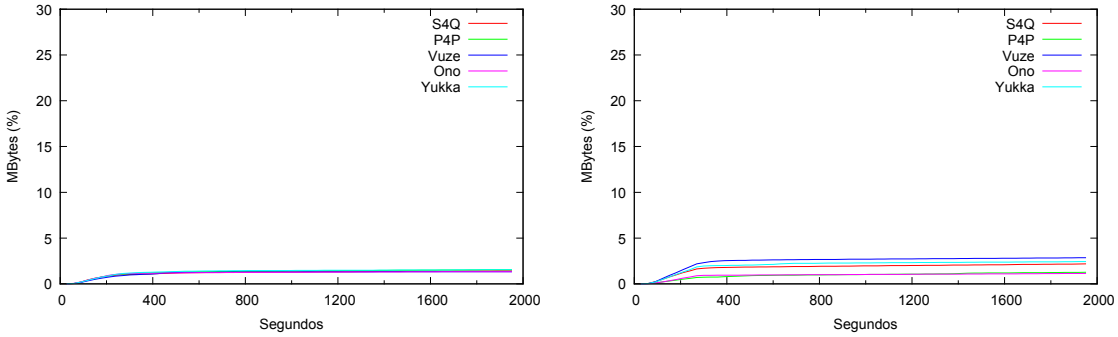

Figura 5.17: *Upload* do *seeder* ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

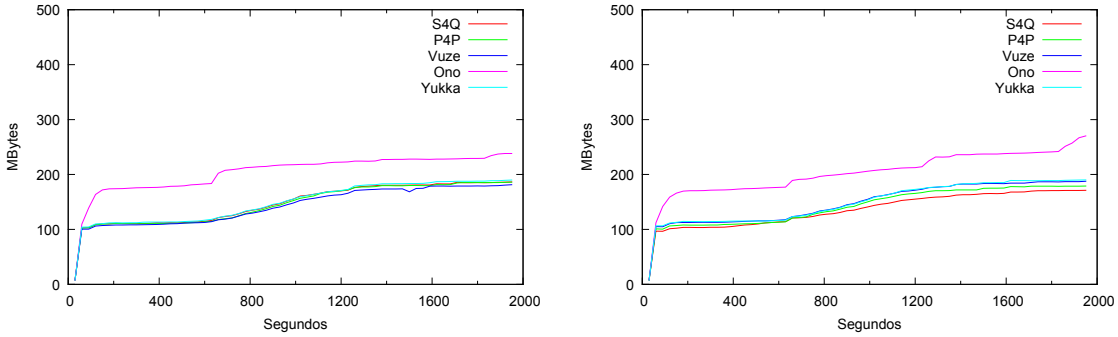

Figura 5.18: Consumo de memória RSS no *tracker* ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

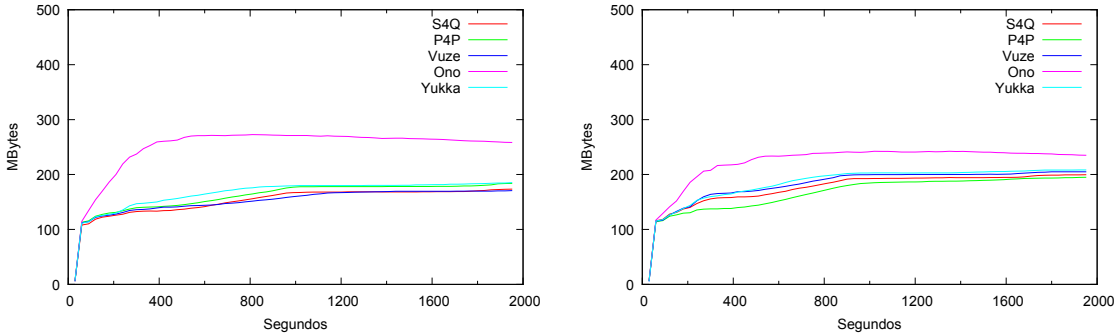

Figura 5.19: Consumo de memória RSS no *seeder* ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

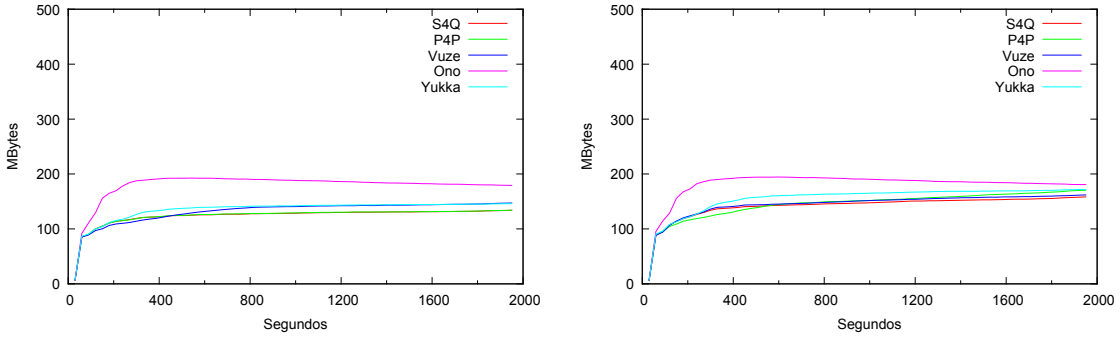

Figura 5.20: Consumo de memória RSS no *leecher* ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

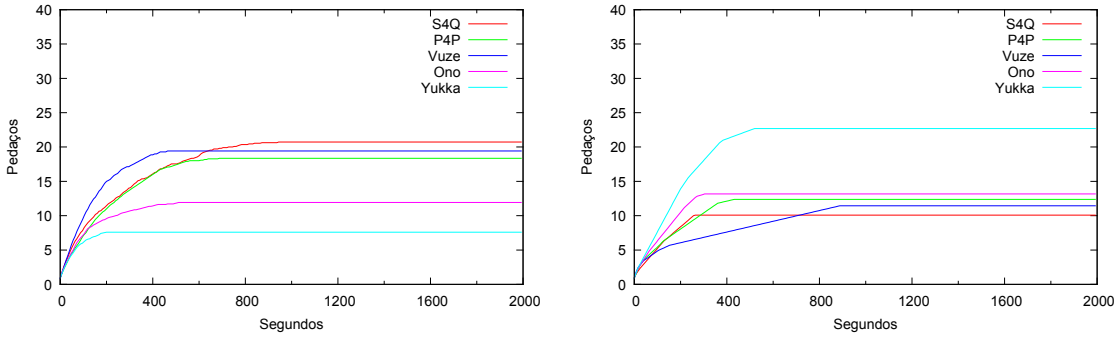

Figura 5.21: Média amostral da ausência de pedaços no critério corte ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

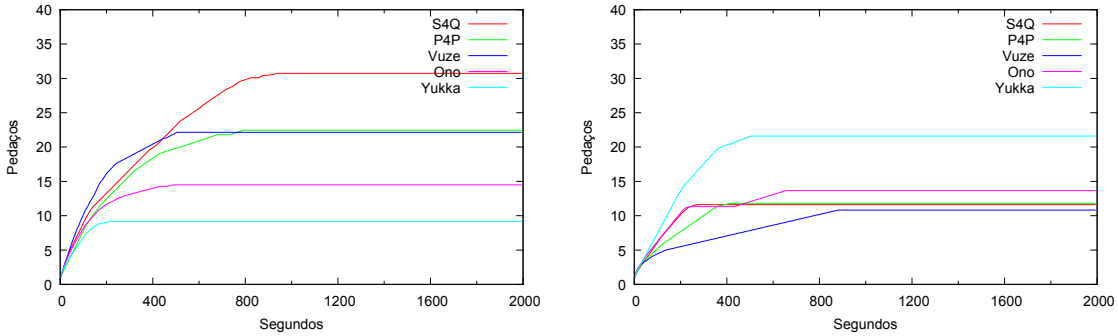

Figura 5.22: Média amostral da ausência de pedaços no critério pausa ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

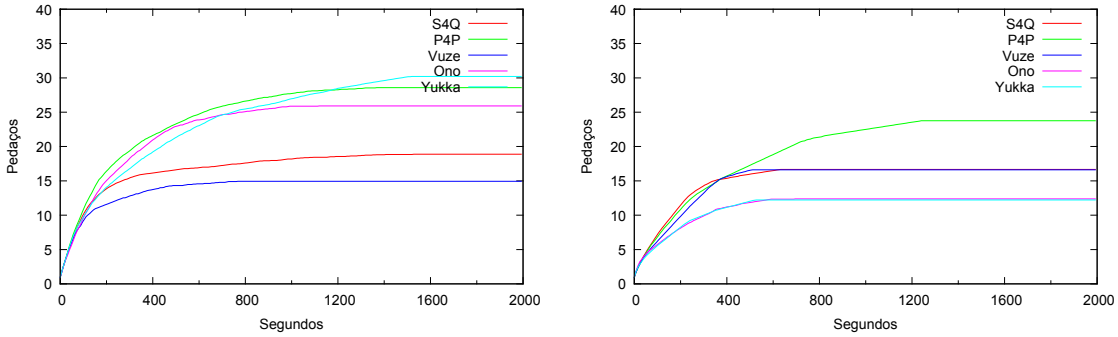

Figura 5.23: Média amostral da ausência de pedaços no critério corte ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

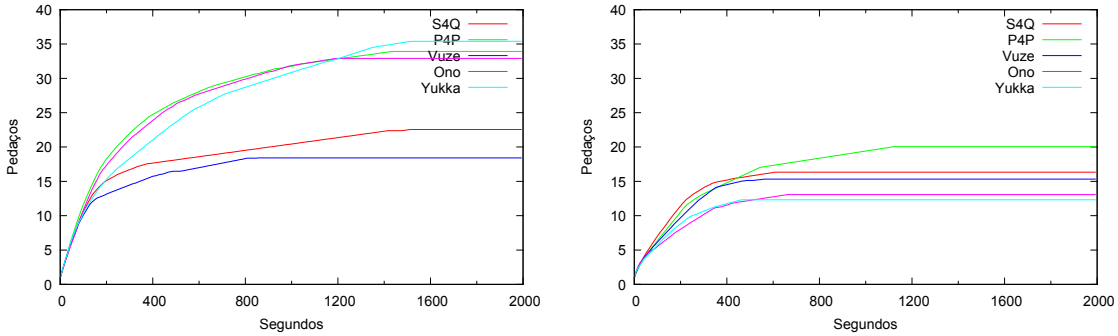

Figura 5.24: Média amostral da ausência de pedaços no critério pausa ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

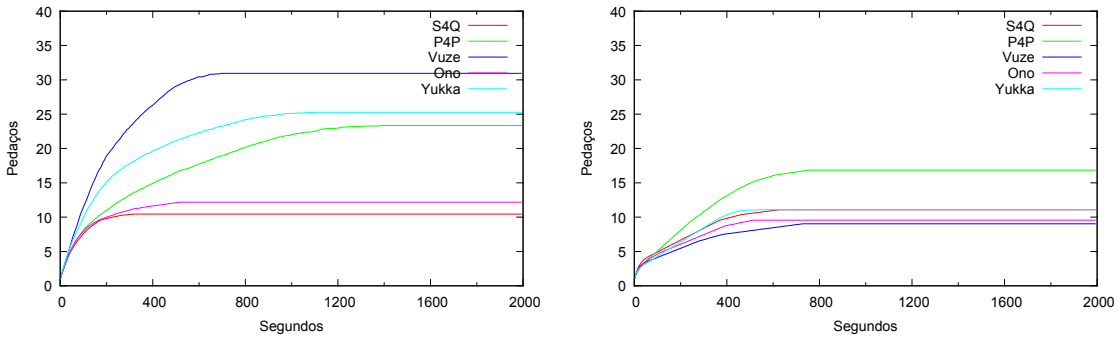

Figura 5.25: Média amostral da ausência de pedaços no critério corte ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

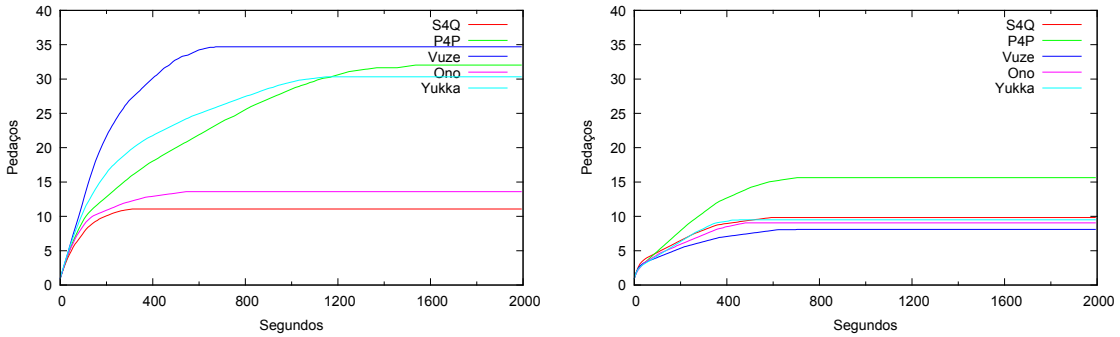

Figura 5.26: Média amostral da ausência de pedaços no critério pausa ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

5.8 Nível de estresse ao longo do tempo

O nível de estresse ao longo do experimento, no critério corte, pode ser visto nas Figuras 5.31, 5.33, 5.35, 5.37 e 5.39. Nesses gráficos vemos o mesmo que foi observado no acumulado de pedaços ausentes, isto é, o S4Q tende a ter um dos melhores resultados, exceto no grupo com 25 pares (no BitTorrent, por padrão, o *tracker* distribui listas com até 50 pares, isso pode fazer com que o último a entrar em um grupo com até 50 pares não tenha o que fazer com um algoritmo para seleção de pares). Com ASSP houve uma melhora do S4Q com o grupo de 25 pares e com os outros grupos ele mantevem um baixo nível de estresse, mas as outras soluções também alcançaram bons números. Essa melhora com ASSP mostra como os níveis de NEC/NEP estão relacionados com a chegada sequencial de pedaços.

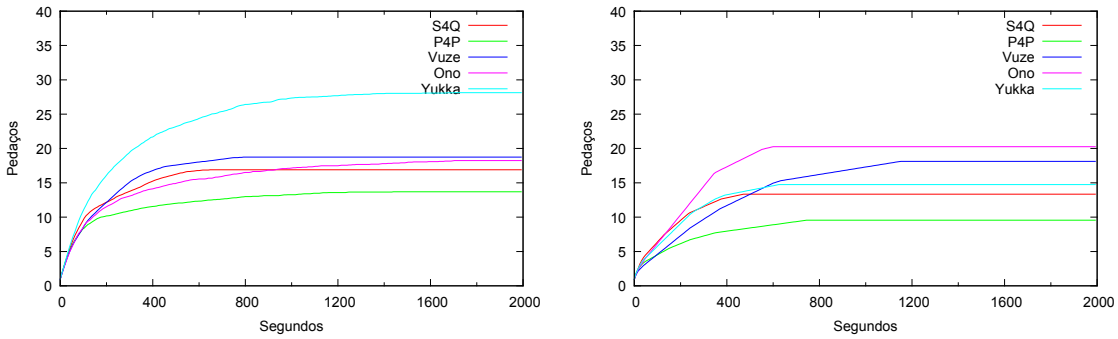

Figura 5.27: Média amostral da ausência de pedaços no critério corte ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

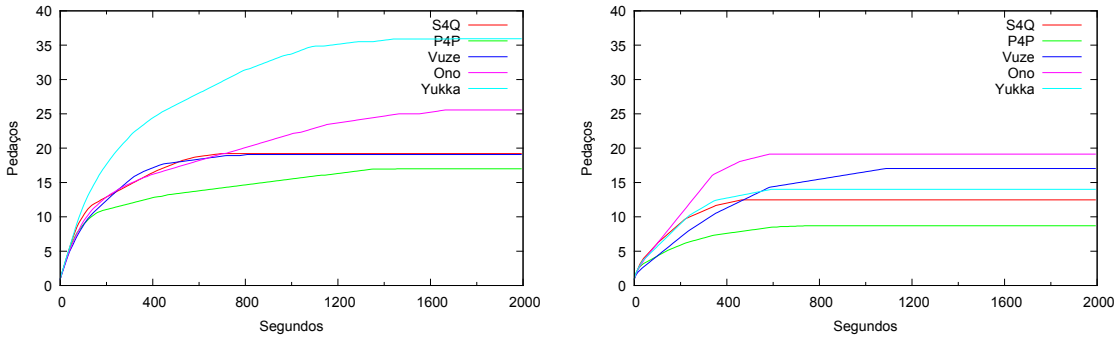

Figura 5.28: Média amostral da ausência de pedaços no critério pausa ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

Em todos os casos vemos uma aumento acentuado do nível de estresse nos primeiros 200 segundos. Isso pode ser explicado observando as Seções 3.4 e 5.3, sobre a espera de 40 segundos para iniciar a métrica e o tempo para início do *download* que vai de 50 à 150 segundos, respectivamente. Logo, o tempo perdido para iniciar o *download* já produz uma elevação do nível de estresse. A queda no nível de estresse que surge em seguida revela o fim do *download* que ocorre entre 200 e 600 segundos, conforme pode ser visto na Figura 5.6.

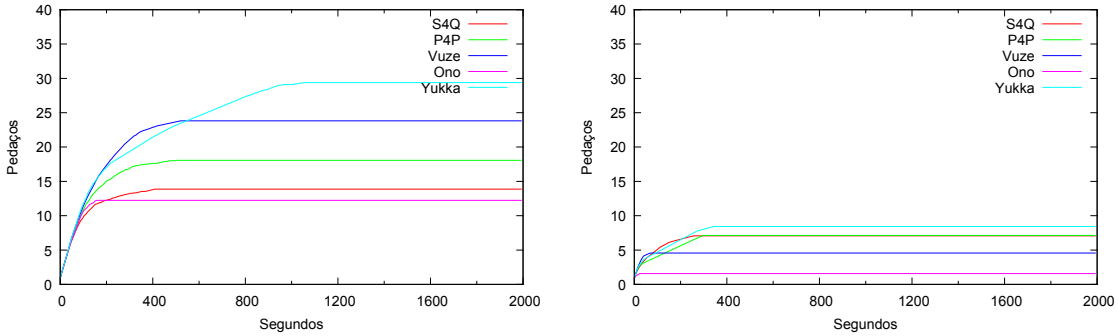

Figura 5.29: Média amostral da ausência de pedaços no critério corte ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

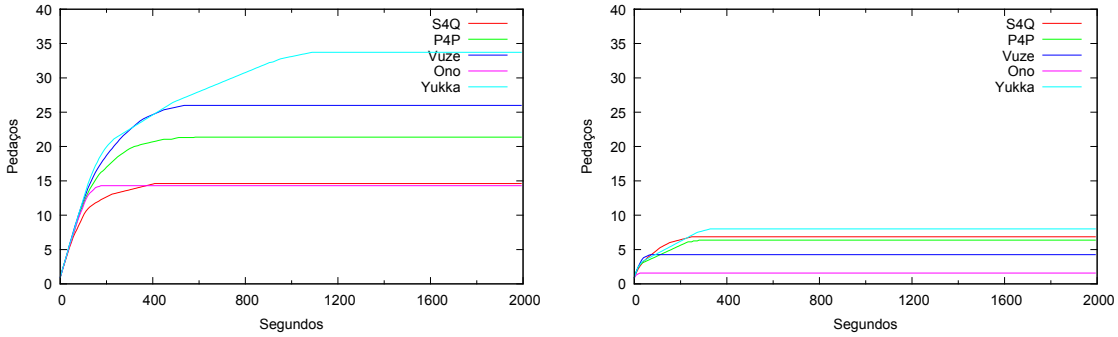

Figura 5.30: Média amostral da ausência de pedaços no critério pausa ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

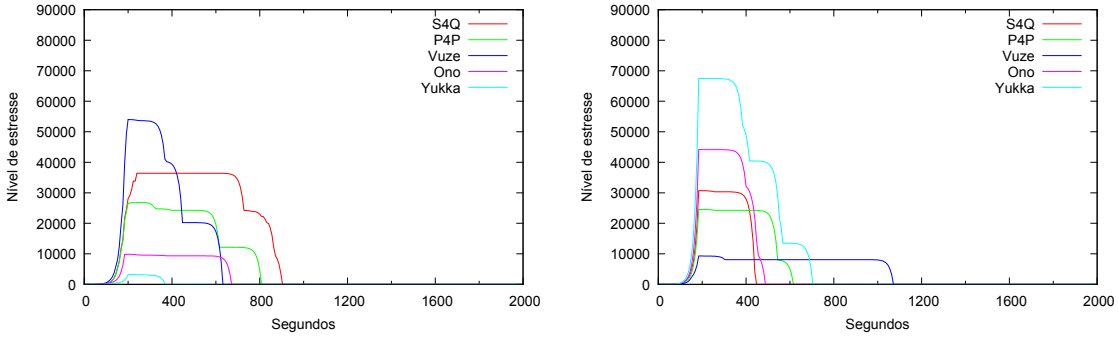

Figura 5.31: Média amostral do nível de estresse no critério corte ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

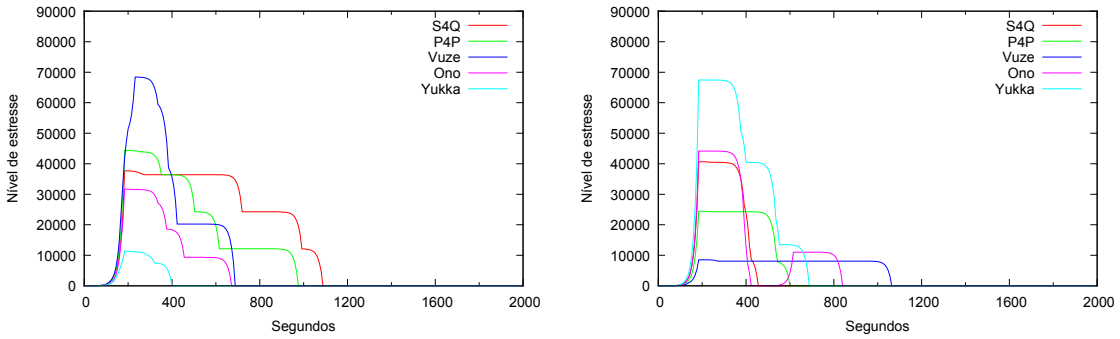

Figura 5.32: Média amostral do nível de estresse no critério pausa ao longo do experimento com 25 pares utilizando ASAP (esquerda) e ASSP (direita)

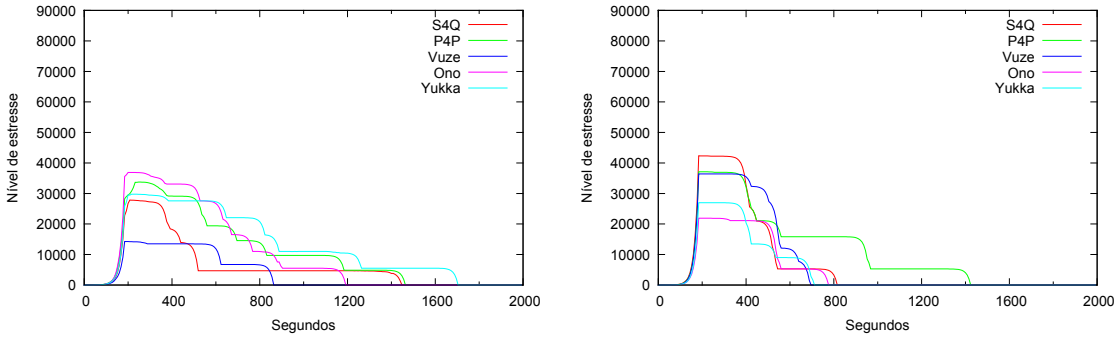

Figura 5.33: Média amostral do nível de estresse no critério corte ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

5.9 O novo algoritmo e seu tempo de reação

Analisando os dados verificamos que o S4Q ainda tem chances de apresentar um resultado melhor. Como podemos ver nas Tabelas 5.5, 5.7, 5.9, 5.11 e 5.13, o experimento teve uma grande concentração de máquinas com velocidade de 1 MByte/s, exceto no grupo com 125 pares, onde houve um aumento dos que apresentaram velocidades iguais ou inferiores a 512 Kbits/s.

As máquinas que atingem 1Mbyte/s conseguem baixar o vídeo em apenas 127 segundos, não dando "tempo de reação" ao S4Q, uma vez que a LPE é formada a cada 80 segundos (Figura 5.41).

Reduzindo a velocidade máxima das máquinas, aumentando o tamanho do vídeo, aumentando a taxa do vídeo e/ou o tempo do experimento, podemos exigir dos pares

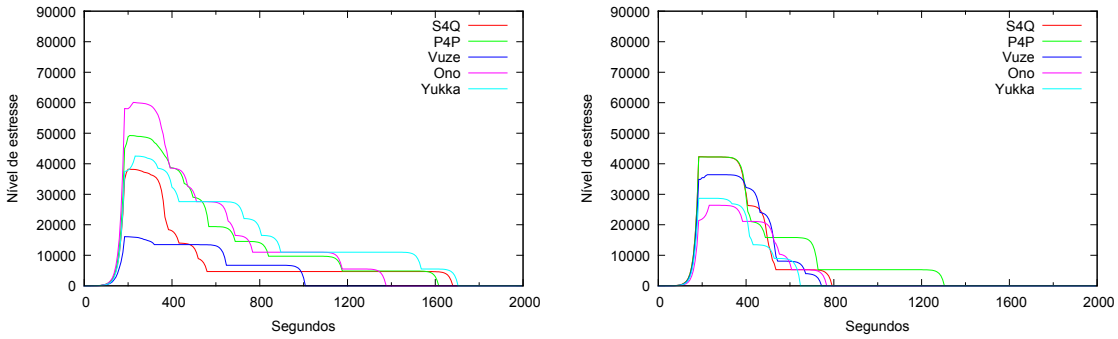

Figura 5.34: Média amostral do nível de estresse no critério pausa ao longo do experimento com 50 pares utilizando ASAP (esquerda) e ASSP (direita)

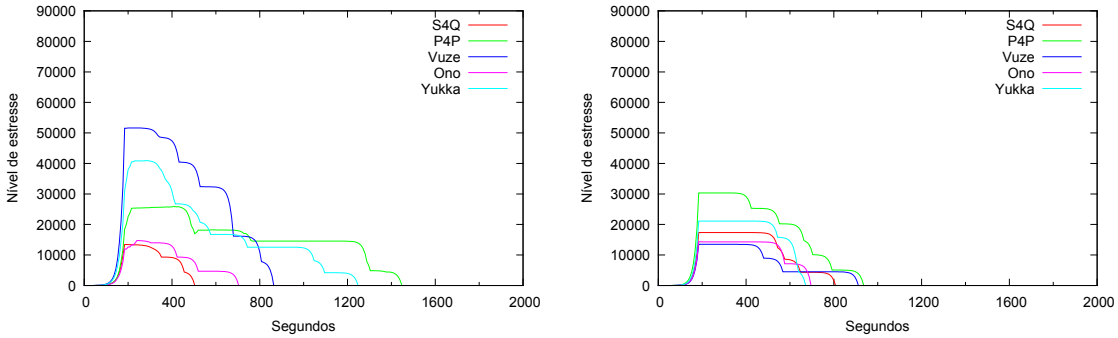

Figura 5.35: Média amostral do nível de estresse no critério corte ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

uma maior troca de LPEs. Logo, isso pode fazer com que o S4Q apresente resultados melhores.

Nas Tabelas 5.6, 5.8, 5.10, 5.12 e 5.14, vemos uma distribução um pouco melhor, quando o assunto é ASSP. Porém, precisamos observar a concentração de máquinas com velocidade igual ou inferior a 512 Kbits/s, pois elas não são consideradas para as métricas de estresse.

Quando a distribuição é mais homogênea o S4Q tem um resultado melhor, basta comparar, no experimento com 25 pares, a Figura 5.31 (ASAP) e a Tabela 5.5 com a Figura 5.31 (ASSP) e a Tabela 5.6.

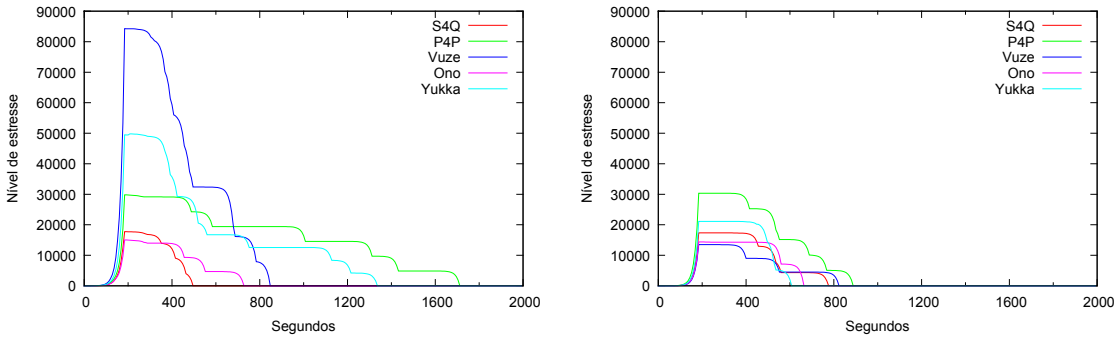

Figura 5.36: Média amostral do nível de estresse no critério pausa ao longo do experimento com 75 pares utilizando ASAP (esquerda) e ASSP (direita)

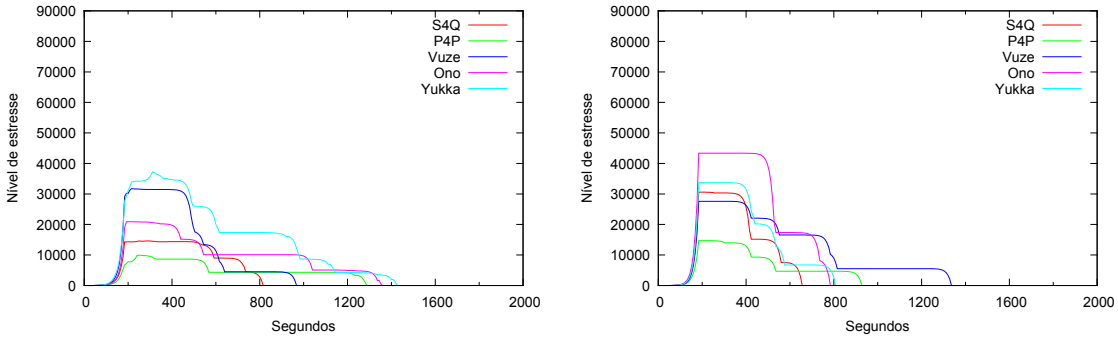

Figura 5.37: Média amostral do nível de estresse no critério corte ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

5.10 Consumo (*download*) e contribuição (*upload*) dos pares

Deixamos o sistema livre sem provocar nenhuma interferência como, por exemplo, incluir *free riders*. Podemos notar na Tabela 5.24 que houve um equilíbrio entre o consumo e a contribuição dos pares em todos os casos, com uma pequena distorção quando utilizamos ASSP em pequenos grupos.

5.11 Comparativo

Por fim, na Tabela 5.25 destacamos as soluções que alcançaram os dois melhores resultados. Note que o S4Q é mais constante em seu desempenho, ou seja, ele é mais confiável para o propósito de se obter uma boa transmissão de vídeo.

Tabela 5.5: Percentual de máquinas em cada faixa de velocidade, de *download*, com 25 pares e ASAP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	15.67	19.22	16.93	10.67	10.80
512 ⊣ 1024	0.00	0.27	0.53	0.13	0.27
1024 ⊣ 2048	6.68	3.63	0.67	1.06	0.13
2048 ⊣ 4096	21.39	24.87	21.60	28.53	17.60
4096 ⊣ 8192	56.27	52.02	60.26	59.60	71.20

Tabela 5.6: Percentual de máquinas em cada faixa de velocidade, de *download*, com 25 pares e ASSP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	27.63	22.01	27.46	19.25	23.25
512 ⊣ 1024	9.16	6.04	0.0	9.49	3.09
1024 ⊣ 2048	21.16	26.04	17.49	26.47	26.61
2048 ⊣ 4096	16.98	27.25	10.79	27.94	27.29
4096 ⊣ 8192	25.07	18.66	44.26	16.85	19.76

Tabela 5.7: Percentual de máquinas em cada faixa de velocidade, de *download*, com 50 pares e ASAP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	12.02	11.89	11.34	13.17	10.22
512 ⊣ 1024	0.76	0.74	0.33	0.34	0.34
1024 ⊣ 2048	1.03	4.12	1.67	2.09	1.81
2048 ⊣ 4096	18.27	16.68	16.21	21.00	15.59
4096 ⊣ 8192	67.93	66.58	70.45	63.40	72.04

Tabela 5.8: Percentual de máquinas em cada faixa de velocidade, de *download*, com 50 pares e ASSP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	28.99	34.71	7.68	35.62	17.04
512 ⊣ 1024	0.61	0.88	0.54	0.68	0.68
1024 ⊣ 2048	20.57	16.81	25.34	14.07	21.97
2048 ⊣ 4096	21.32	14.64	16.71	7.61	21.16
4096 ⊣ 8192	28.51	32.95	49.73	42.01	39.15

Tabela 5.9: Percentual de máquinas em cada faixa de velocidade, de *download*, com 75 pares e ASAP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	9.75	12.07	11.80	10.41	9.63
512 ⊣ 1024	0.72	3.92	0.22	0.98	0.36
1024 ⊣ 2048	1.12	3.83	12.34	3.07	4.06
2048 ⊣ 4096	12.29	12.83	34.29	8.19	15.25
4096 ⊣ 8192	76.13	67.35	41.34	77.35	70.70

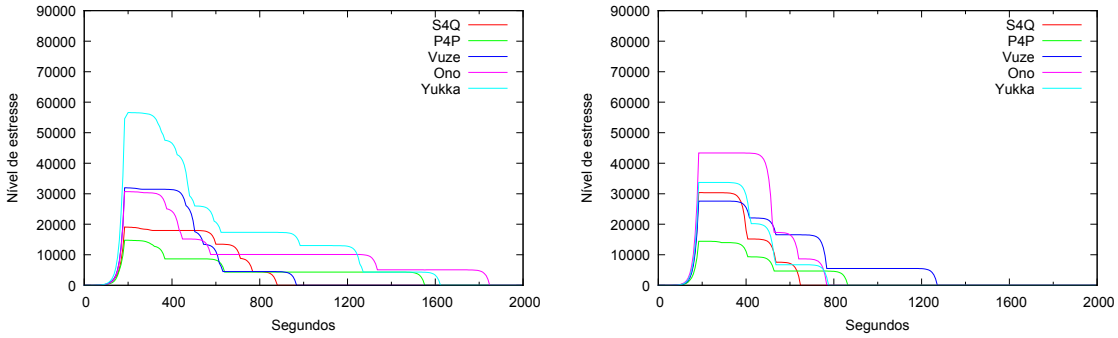

Figura 5.38: Média amostral do nível de estresse no critério pausa ao longo do experimento com 100 pares utilizando ASAP (esquerda) e ASSP (direita)

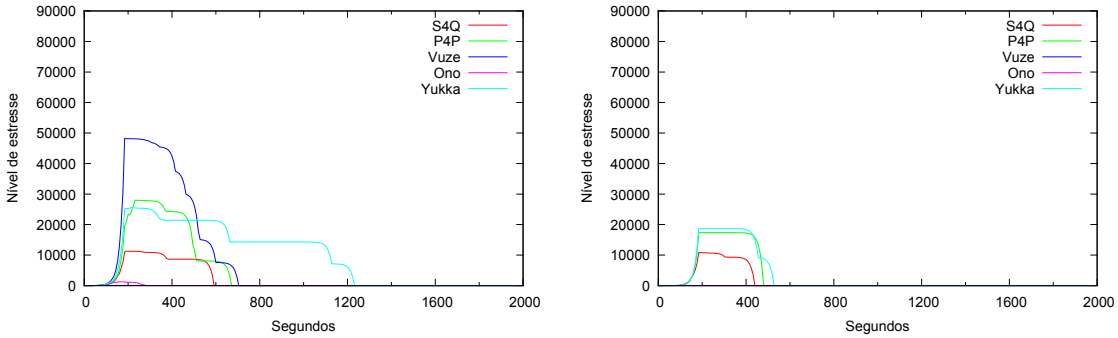

Figura 5.39: Média amostral do nível de estresse no critério corte ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

Tabela 5.10: Percentual de máquinas em cada faixa de velocidade, de *download*, com 75 pares e ASSP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	15.26	26.87	16.84	48.70	29.90
512 ⊣ 1024	0.54	0.40	1.21	0.13	0.81
1024 ⊣ 2048	4.51	8.76	5.00	2.28	2.24
2048 ⊣ 4096	12.96	12.63	12.37	5.95	13.88
4096 ⊣ 8192	66.73	51.33	64.58	42.93	53.18

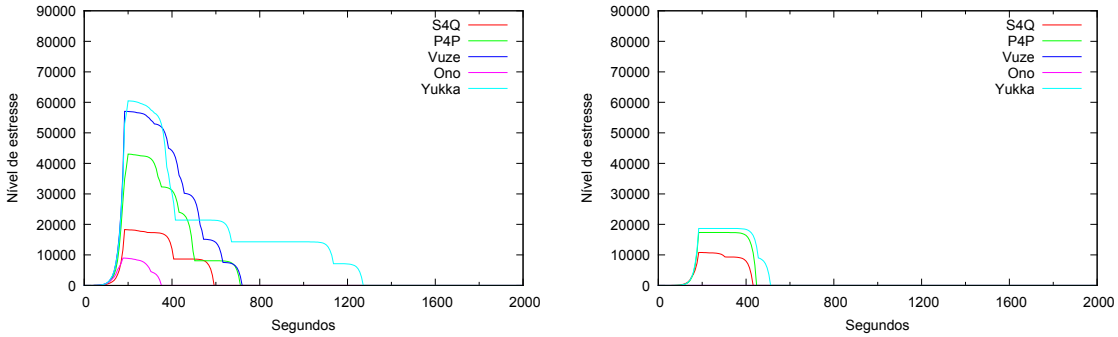

Figura 5.40: Média amostral do nível de estresse no critério pausa ao longo do experimento com 125 pares utilizando ASAP (esquerda) e ASSP (direita)

Tabela 5.11: Percentual de máquinas em cada faixa de velocidade, de *download*, com 100 pares e ASAP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	13.94	14.50	10.81	13.47	16.82
512 ⊣ 1024	0.64	0.48	0.30	5.96	6.27
1024 ⊣ 2048	4.62	0.27	4.13	1.28	7.073
2048 ⊣ 4096	10.75	7.56	20.57	15.25	21.79
4096 ⊣ 8192	70.05	77.20	64.20	64.04	48.04

Tabela 5.12: Percentual de máquinas em cada faixa de velocidade, de *download*, com 100 pares e ASSP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	51.19	20.58	33.34	57.81	47.62
512 ⊣ 1024	0.84	0.30	1.41	0.10	0.34
1024 ⊣ 2048	6.98	4.93	9.88	7.14	10.19
2048 ⊣ 4096	13.42	10.06	10.82	13.14	13.30
4096 ⊣ 8192	27.58	64.13	44.56	21.82	28.55

Tabela 5.13: Percentual de máquinas em cada faixa de velocidade, de *download*, com 125 pares e ASAP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	56.35	53.34	51.75	47.84	48.48
512 ⊣ 1024	0.25	0.33	0.35	0.36	0.25
1024 ⊣ 2048	0.72	3.65	0.89	2.07	0.85
2048 ⊣ 4096	7.92	10.75	15.16	4.74	12.39
4096 ⊣ 8192	34.77	31.94	31.85	45.00	38.04

Tabela 5.14: Percentual de máquinas em cada faixa de velocidade, de *download*, com 125 pares e ASSP

Kbits/s	S4Q	P4P	Vuze	Ono	Yukka
0 ⊣ 512	59.20	77.91	52.78	81.31	59.62
512 ⊣ 1024	0.41	3.23	0.33	0.19	0.16
1024 ⊣ 2048	1.07	0.74	2.05	1.01	3.74
2048 ⊣ 4096	10.01	4.69	8.66	1.53	6.06
4096 ⊣ 8192	29.30	13.43	36.17	15.96	30.42

Tabela 5.15: Consumo de memória (MBytes) no *tracker*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	3162	3976	3134	3912	3209	3207	3166	4004	3076	3613
510	3184	4045	3186	3998	3241	3233	3117	4135	3080	3664
1020	3182	4062	3205	4022	3233	3239	3125	4155	2911	3675
1500	3181	4060	3208	4035	3216	3144	3129	4158	2949	3680
1950	3182	3942	3225	4064	3218	3146	3139	4025	2969	3539
50 Pares										
30	2534	2451	2545	2527	2603	2488	2505	2481	2503	2469
510	2579	2504	2592	2578	2658	2545	2576	2562	2557	2523
1020	2588	2512	2608	2587	2663	2555	2579	2559	2574	2534
1500	2587	2522	2612	2599	2675	2566	2578	2573	2581	2544
1950	2587	2529	2608	2605	2676	2558	2571	2577	2579	2550
75 Pares										
30	3849	3586	3815	3695	3785	3831	3800	3765	3851	3851
510	3936	3682	3911	3796	3859	3930	3922	3914	3923	3944
1020	3958	3730	3942	3825	3875	3954	3921	3923	3941	3967
1500	3959	3756	3947	3842	3883	3948	3925	3927	3940	3967
1950	3955	3732	3946	3798	3879	3951	3927	3931	3938	3964
100 Pares										
30	3474	3642	3643	3677	3526	3610	3551	3530	3582	3553
510	3649	3724	3696	3751	3663	3719	3633	3703	3640	3652
1020	3690	3737	3725	3833	3741	3700	3654	3753	3691	3688
1500	3687	3721	3714	3789	3734	3682	3669	3767	3719	3642
1950	3667	3740	3746	3757	3783	3685	3694	3768	3711	3700
125 Pares										
30	3483	2882	3379	2698	3101	3570	2747	3357	3143	3508
510	3586	2987	3480	2805	3180	3668	2923	3526	3238	3611
1020	3627	3010	3524	2839	3202	3718	2950	3558	3303	3645
1500	3649	3038	3547	2867	3254	3732	2953	3582	3306	3666
1950	3650	3050	3553	2871	3250	3730	2972	3621	3321	3668

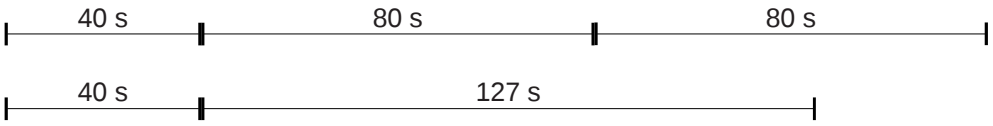

Figura 5.41: Montagem da LPE x *download* a 1 MByte/s

Tabela 5.16: Consumo de memória (MBytes) no *seeder*

Tempo	Ono		P4P		Vuze		S4Q		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	3166	3494	3453	3487	3814	3152	3811	3555	3764	3430
510	3166	3566	3486	3577	3826	3230	3922	3652	3815	3519
1020	3167	3573	3525	3571	3821	3240	3924	3652	3806	3525
1500	3163	3570	3516	3555	3825	3233	3928	3645	3828	3496
1950	3168	3583	3535	3557	3839	3232	3932	3652	3829	3507
50 Pares										
30	3689	3622	3570	3564	3485	3434	3596	3421	3478	3492
510	3765	3742	3675	3693	3604	3570	3761	3605	3607	3644
1020	3767	3757	3673	3704	3609	3584	3764	3602	3614	3658
1500	3774	3762	3677	3710	3609	3579	3762	3600	3610	3664
1950	3778	3742	3688	3709	3612	3579	3760	3598	3616	3660
75 Pares										
30	3887	4288	3995	3996	2123	3891	3975	3947	3787	3893
510	3935	4370	4037	4090	2164	3980	4115	4061	3829	3963
1020	3939	4378	4031	4112	2150	3986	4124	4070	3828	3982
1500	3934	4378	4039	4113	2153	3986	4145	4072	3832	3977
1950	3945	4396	4046	4109	2154	3974	4137	4071	3836	3980
100 Pares										
30	3360	3439	3538	3519	3377	3483	3559	3350	3541	3397
510	3572	3557	3616	3724	3550	3651	3755	3556	3634	3534
1020	3558	3592	3658	3717	3588	3631	3717	3636	3661	3536
1500	3542	3614	3652	3671	3607	3559	3733	3674	3666	3570
1950	3515	3619	3662	3592	3644	3555	3739	3701	3699	3633
125 Pares										
30	3427	3632	3757	3700	3806	3526	3745	3078	3797	3556
510	3546	3780	3914	3836	3942	3705	3992	3338	3936	3657
1020	3561	3812	3913	3900	3921	3771	3929	3249	3950	3679
1500	3568	3784	3968	3865	3933	3758	3958	3263	3945	3686
1950	3552	3791	3952	3865	3970	3775	3928	3280	3966	3711

Tabela 5.17: Consumo de memória (MBytes) no *leecher*

Tempo	Ono		P4P		Vuze		S4Q		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	3674	3421	3634	3369	3625	3600	3568	3373	3511	3468
510	3831	3603	3798	3567	3776	3782	3819	3628	3726	3624
1020	3833	3619	3811	3597	3761	3798	3823	3646	3726	3646
1500	3826	3621	3813	3602	3764	3793	3824	3645	3729	3648
1950	3834	3623	3817	3601	3765	3797	3827	3646	3743	3648
50 Pares										
30	3396	3278	3402	3316	3412	3316	3352	3315	3383	3315
510	3612	3492	3623	3521	3632	3547	3635	3581	3606	3555
1020	3612	3516	3628	3552	3630	3573	3636	3596	3620	3582
1500	3612	3523	3630	3550	3631	3575	3633	3597	3616	3587
1950	3605	3528	3631	3553	3628	3563	3628	3594	3616	3579
75 Pares										
30	3331	3357	3413	3325	3476	3282	3420	3397	3437	3344
510	3569	3584	3646	3553	3681	3536	3703	3649	3645	3582
1020	3565	3599	3657	3580	3699	3550	3704	3653	3646	3598
1500	3564	3607	3666	3578	3696	3554	3706	3649	3645	3603
1950	3566	3607	3666	3583	3695	3547	3703	3652	3651	3606
100 Pares										
30	3471	3488	3532	3475	3463	3480	3484	3423	3572	3440
510	3730	3668	3732	3708	3686	3687	3756	3655	3779	3652
1020	3742	3678	3751	3713	3698	3694	3755	3685	3788	3663
1500	3724	3696	3750	3700	3708	3683	3759	3703	3782	3658
1950	3714	3699	3748	3688	3736	3681	3754	3694	3786	3698
125 Pares										
30	3470	3526	3480	3539	3486	3523	3459	3405	3435	3528
510	3646	3677	3659	3665	3638	3691	3672	3608	3617	3698
1020	3657	3679	3666	3687	3642	3721	3661	3577	3622	3705
1500	3656	3692	3693	3686	3647	3721	3660	3567	3626	3700
1950	3658	3694	3693	3694	3649	3720	3673	3576	3646	3718

Tabela 5.18: Consumo de memória (MBytes) VSZ no *tracker*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	387	828	485	830	444	415	528	879	551	470
510	826	1027	855	1028	867	814	880	1036	856	955
1020	828	1031	858	1032	869	830	880	1038	857	958
1500	830	1030	857	1032	869	830	881	1037	858	959
1950	829	1024	859	992	869	831	881	988	859	948
50 Pares										
30	720	771	771	771	628	695	734	771	758	772
510	832	833	831	836	829	791	803	845	834	835
1020	834	836	835	836	830	792	804	846	834	836
1500	834	835	834	838	832	793	805	847	834	836
1950	833	836	835	838	832	793	806	848	835	837
75 Pares										
30	1075	1001	1075	1039	965	968	1039	1075	1008	1074
510	1135	1138	1139	1101	1174	1137	1113	1153	1143	1140
1020	1139	1144	1142	1107	1180	1142	1115	1156	1146	1145
1500	1140	1144	1142	1108	1180	1143	1116	1159	1149	1147
1950	1140	1145	1142	1108	1179	1143	1116	1159	1149	1147
100 Pares										
30	1065	1102	738	993	993	1030	1102	1028	1067	1028
510	1125	1166	1169	1169	1165	1167	1180	1139	1130	1132
1020	1129	1176	1174	1177	1170	1173	1181	1143	1134	1138
1500	1131	1176	1173	1178	1171	1174	1185	1144	1134	1138
1950	1137	1178	1179	1184	1178	1178	1187	1150	1143	1142
125 Pares										
30	1101	1101	1101	990	1031	1064	1105	1101	1064	1100
510	1163	1163	1164	1127	1164	1164	1184	1179	1169	1165
1020	1166	1172	1168	1135	1167	1170	1185	1182	1172	1172
1500	1172	1176	1175	1141	1135	1176	1194	1189	1180	1179
1950	1176	1180	1180	1144	1179	1180	1193	1189	1185	1183

Tabela 5.19: Consumo de memória (MBytes) VSZ no *seeder*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	572	605	582	666	469	422	756	627	779	336
510	911	846	898	848	967	929	990	854	982	870
1020	911	847	899	849	968	932	995	859	984	874
1500	912	832	899	849	970	931	995	859	984	874
1950	912	847	898	848	971	932	995	860	984	873
50 Pares										
30	859	943	892	927	877	790	884	922	957	883
510	998	1022	1005	1010	989	1042	1009	1051	1036	1002
1020	999	1025	1006	1011	989	1044	1013	1053	1037	1004
1500	999	1025	1007	1011	988	1044	1013	1054	1036	1004
1950	998	1025	1007	1011	990	1044	1014	1055	1037	1004
75 Pares										
30	684	648	645	864	368	749	774	814	747	812
510	972	980	972	970	696	979	987	944	939	977
1020	973	981	972	971	696	980	989	947	939	977
1500	973	981	972	972	697	981	991	948	939	978
1950	974	980	973	973	697	980	990	948	938	978
100 Pares										
30	1002	1112	896	1041	962	1038	1041	1075	1111	1073
510	1155	1195	1154	1201	1196	1195	1204	1203	1190	1152
1020	1155	1196	1155	1203	1196	1198	1208	1209	1192	1155
1500	1156	1199	1157	1202	1197	1198	1209	1210	1192	1156
1950	1157	1199	1157	1203	1197	1199	1210	1209	1193	1157
125 Pares										
30	1062	1140	1138	1064	988	987	1097	1027	1134	989
510	1175	1220	1217	1219	1210	1220	1226	1190	1212	1223
1020	1177	1223	1217	1221	1211	1222	1228	1193	1215	1225
1500	1178	1222	1219	1220	1211	1221	1229	1192	1214	1225
1950	1177	1222	1219	1221	1213	1222	1229	1194	1215	1224

Tabela 5.20: Consumo de memória (MBytes) VSZ no *leecher*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	779	827	737	823	600	617	836	850	851	561
510	1007	1006	1006	1019	1022	1019	1029	1025	1020	1014
1020	1009	1007	1006	1020	1022	1026	1030	1028	1018	1015
1500	1011	1009	1005	1020	1020	1025	1032	1029	1018	1015
1950	1014	1008	1007	1020	1022	1028	1032	1027	1019	1016
50 Pares										
30	772	861	836	859	712	746	844	843	831	856
510	996	1018	997	1021	1002	1008	1014	1019	1004	1022
1020	997	1019	1000	1023	1003	1011	1015	1022	1005	1024
1500	995	1022	1000	1023	1004	1011	1017	1023	1005	1025
1950	995	1020	998	1018	1001	1008	1014	1019	1006	1025
75 Pares										
30	840	779	840	865	756	790	843	868	747	859
510	996	1024	999	1012	994	1020	1010	1024	998	1013
1020	997	1025	1001	1018	996	1024	1012	1028	999	1016
1500	997	1023	1000	1017	994	1021	1013	1027	999	1015
1950	996	1027	1000	1019	997	1022	1013	1028	999	1015
100 Pares										
30	860	903	792	878	805	831	867	899	870	901
510	1040	1051	1037	1057	1038	1048	1049	1062	1041	1056
1020	1039	1055	1039	1060	1037	1052	1051	1067	1041	1059
1500	1041	1056	1039	1061	1038	1054	1050	1067	1041	1061
1950	1040	1057	1038	1059	1039	1055	1053	1067	1042	1057
125 Pares										
30	845	826	836	863	798	753	822	861	842	847
510	993	1009	994	1008	990	1008	1002	1010	993	1013
1020	996	1009	996	1010	991	1011	1004	1014	994	1015
1500	994	1010	996	1011	993	1011	1005	1013	994	1016
1950	997	1010	996	1011	991	1012	1004	1016	994	1015

Tabela 5.21: Consumo de memória (MBytes) RSS no *tracker*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	5	6	6	8	4	6	6	9	7	5
510	69	105	75	104	76	74	121	160	76	94
1020	84	126	89	127	87	90	135	172	88	112
1500	88	141	96	143	90	95	136	175	97	122
1950	90	149	101	144	91	96	144	172	100	129
50 Pares										
30	8	8	8	8	7	8	8	8	8	9
510	80	79	80	86	78	77	101	110	81	84
1020	90	90	93	96	88	88	105	115	90	94
1500	100	98	109	113	94	100	105	120	104	109
1950	102	102	111	117	96	105	106	123	106	112
75 Pares										
30	8	8	7	8	8	6	7	8	7	9
510	109	113	111	110	113	114	161	180	111	116
1020	142	148	150	140	157	149	176	193	149	151
1500	169	175	172	169	177	175	185	198	172	177
1950	171	177	175	171	179	175	194	199	174	178
100 Pares										
30	7	8	5	8	8	7	8	8	8	8
510	111	117	113	119	115	117	173	173	112	118
1020	162	174	156	174	170	171	180	184	163	170
1500	172	182	161	183	175	180	190	190	169	177
1950	181	192	172	195	188	190	205	207	180	188
125 Pares										
30	8	8	9	8	7	9	8	8	7	8
510	113	110	111	112	112	115	180	175	114	115
1020	160	144	156	153	152	160	218	208	160	158
1500	180	166	179	174	168	184	228	237	183	186
1950	186	172	185	180	181	187	239	270	189	191

Tabela 5.22: Consumo de memória (MBytes) RSS no *seeder*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	5	4	5	5	5	4	6	6	6	5
510	103	113	116	113	120	115	179	144	123	111
1020	107	120	117	118	123	125	180	148	126	125
1500	107	120	118	119	125	129	180	144	128	130
1950	109	123	123	118	129	130	185	143	130	132
50 Pares										
30	7	8	7	8	7	7	8	8	8	7
510	129	137	126	140	141	153	181	201	137	147
1020	129	160	129	156	141	162	187	202	141	158
1500	131	161	131	158	143	162	183	195	142	157
1950	134	164	135	159	144	164	189	193	145	160
75 Pares										
30	4	4	4	5	4	5	6	6	6	5
510	118	143	120	136	86	146	200	166	126	138
1020	121	146	127	150	89	151	207	174	128	151
1500	121	148	128	155	89	154	217	177	128	151
1950	122	149	131	157	91	154	224	176	130	154
100 Pares										
30	7	7	5	7	6	7	6	9	7	7
510	147	150	146	172	153	162	229	201	153	164
1020	152	183	148	184	156	184	229	199	164	188
1500	154	186	147	184	157	187	226	196	166	188
1950	156	189	152	186	157	188	226	191	170	191
125 Pares										
30	7	7	8	8	7	6	8	8	7	7
510	136	160	146	145	143	171	267	231	158	175
1020	168	192	178	185	162	199	272	242	180	202
1500	169	194	179	189	170	200	266	240	181	204
1950	173	199	184	196	170	204	259	235	185	208

Tabela 5.23: Consumo de memória (MBytes) RSS no *leecher*

Tempo	S4Q		P4P		Vuze		Ono		Yukka	
(s)	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
25 Pares										
30	5	7	6	5	4	5	5	7	5	4
510	127	131	129	134	133	139	188	187	135	137
1020	131	135	132	137	135	142	193	192	138	141
1500	131	137	132	136	134	142	193	191	138	142
1950	132	138	132	138	137	145	194	190	140	143
50 Pares										
30	5	6	6	7	6	6	7	7	5	7
510	129	143	132	140	131	142	196	208	135	147
1020	131	150	135	148	133	149	195	212	137	155
1500	131	153	136	150	133	150	191	208	138	155
1950	135	157	137	154	135	152	189	209	139	158
75 Pares										
30	5	5	7	7	5	7	7	7	5	7
510	139	155	135	151	134	156	206	211	145	158
1020	141	161	138	163	140	161	206	206	148	166
1500	141	162	138	165	141	163	202	200	148	169
1950	143	167	140	170	140	168	199	197	150	174
100 Pares										
30	6	8	6	7	6	6	6	7	6	7
510	141	153	142	160	144	154	215	213	151	170
1020	145	164	144	169	148	169	212	210	156	180
1500	145	174	144	169	148	172	205	204	157	185
1950	147	178	147	173	150	179	204	198	160	186
125 Pares										
30	7	6	7	8	7	7	7	8	7	6
510	124	141	124	139	128	144	192	194	138	158
1020	128	147	130	153	140	151	188	191	143	165
1500	131	153	132	161	143	158	184	185	144	169
1950	133	158	135	171	148	162	180	180	146	171

Tabela 5.24: *Downloaded* x *Uploaded*

	S4Q		P4P		Vuze		Ono		Yukka	
Pares	Down	Up	Down	Up	Down	Up	Down	Up	Down	Up
ASAP										
25	52.0	48.0	52.0	48.0	52.0	48.0	51.0	49.0	52.0	48.0
50	51.0	49.0	51.0	49.0	51.0	49.0	50.0	50.0	51.0	49.0
75	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	51.0	49.0
100	51.0	49.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
125	50.0	50.0	50.0	50.0	49.0	51.0	49.0	51.0	50.0	50.0
ASSP										
25	55.0	45.0	55.0	45.0	55.0	45.0	55.0	45.0	55.0	45.0
50	52.0	48.0	52.0	48.0	52.0	48.0	51.0	49.0	52.0	48.0
75	51.0	49.0	51.0	49.0	51.0	49.0	51.0	49.0	51.0	49.0
100	51.0	49.0	51.0	49.0	51.0	49.0	51.0	49.0	51.0	49.0
125	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0

Tabela 5.25: Comparativo

		S4Q		P4P		Vuze		Ono		Yukka	
		ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP	ASAP	ASSP
Menor tempo de *download*	25		2			2	1			1	
	50					1			1	2	2
	75	1						2	1		2
	100	2		1	1		2				
	125	1					2	2	1		
Menor tempo para início de *download*	25		2			2	1			1	
	50	1			2	2			1		
	75	2		1			2		1		
	100		2	1	1			2			
	125	2		1	2		1				
Menor *uploaded* do *seeder*	25	1	2				1	2			
	50		1	1	2			2			
	75	2				1			1		2
	100		2					2	1	1	
	125				2	1		2	1		
Menor número de ausências no critério corte	25		1				2	2		1	
	50	2				1			2		1
	75	1					1	2	2		
	100	2	2	1	1						
	125	2					2	1	1		
Menor número de ausências no critério pausa	25						1	2	2	1	
	50	2				1			2		1
	75	1					1	2	2		
	100		2	1	1	2					
	125	2					2	1	1		
Sub-total (1 Lugar)		6	2	7	4	5	7	2	10	5	2
Total (1 Lugar)		8		11		12		12		7	
Sub-total (2 Lugar)		9	7	0	4	4	6	11	5	1	3
Total (2 Lugar)		16		4		10		16		4	
Total Geral		24		15		22		28		11	

Capítulo 6

Conclusão

Este trabalho apresentou uma nova métrica em QoE para transmissão de vídeo, partindo da verificação de pedaços e utilizando a sequência Fibonacci sobre dois critérios: cortes e pausas. Também foi apresentado um novo algoritmo para seleção de pares que atribui o valor obtido com a nova métrica a uma lista de pares estáveis. Dessa forma as melhores listas são trocadas entre os pares, com o propósito promover agrupamentos e acelerar a formação de *supernodes*, para com isso obter um melhor tempo de *download* e uma reprodução contínua do vídeo com o mínimo de interrupções, durante sua transmissão, para o sistema de um modo geral.

A partir dos resultados obtidos podemos concluir que:

- É possível construir algoritmos simples que alcançam resultados tão bons quanto os que buscam informações em fontes externas, uma vez que o S4Q apresentou números similares aos demais;
- Podemos realizar agrupamentos sem estabelecer limitações, como taxa de *upload* ou localização geográfica, nossa solução realiza essa tarefa com a troca das LPEs;
- Não precisamos nos preocupar em identificar os *supernodes* para aproveitar seus recursos, por serem bons fornecedores eles acabam fazendo parte da LPEs;
- É viável construir soluções que são orientadas pela QoE, pois conseguimos

produzir avaliações dinâmicas dos NEC/NEP e utilizá-los na seleção de pares.

6.1 Trabalhos futuros

Nossa pesquisa mostrou pontos que precisam ser explorados com mais detalhes e que podem produzir resultados ainda melhores, otimizando o algoritmo para seleção de pares e refinando a nova métrica.

São propostas para trabalhos futuros:

- Estender o algoritmo para o uso de SVC, pois o uso desse formato pode produzir uma melhor significativa da QoE em cenários heterogêneos e requer um método diferenciado para seleção de pares devido a forma de distribuição de cada camada de refinamento do vídeo;

- Investigar a alteração de atributos do algoritmo como segundos por pedaço, tempo de *buffer*, tempo para formação da lista de pares estáveis e número de listas que devem compor a LPE. Essas variáveis podem indicar um ponto de operação ideal entre performace e sobrecarga de controle;

- Investigar a alteração de atributos do experimento como velocidade máxima das máquinas, tamanho do vídeo, taxa do vídeo e tempo. A intenção é avaliar até que ponto a falta de tempo para reagir ao meio afeta o novo algoritmo para seleção de pares;

- Avaliar ausência de pedaços em termos percentuais, sobre cada critério (cortes ou pausas), por meio de MOS. Isso vai permitir uma melhor qualificação das faixas que definem os níveis de estresse em baixo, médio e alto. Também devemos considerar a variação da QoE no tempo;

- Acreditamos que nossa solução acelera a formação dos *supernodes*, mas é preciso verificar se a troca das LPEs não leva a uma "clusterização". Sendo caracterizado o evento, deve-se estudar critérios para adição dos pares da LPE recebida, buscar alguma diversidade e evitar sincronismo;

- Relizar experimentos em um cenário real de vídeo e considerar a saída dos pares (*churn*) assim que eles se tornam *seeders*, refinar a avaliação dos níveis de estresse com base em informações das redes sociais, avaliar o impacto do algoritmo sobre *free riders*, expandir os experimentos para cenários moveis, implementar um mecanismo que reduza o tempo de ínicio de *download* e comparar a A^2Q com o PSQA.

Referências Bibliográficas

[Abboud et al 2009] Abboud, Osama; Pussep, Konstantin; Kovacevic, Aleksandra e Steinmetz, Ralf (2009). *Quality Adaptive Peer-to-Peer Streaming Using Scalable Video Coding*. Em 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS), Páginas 41-54.

[Abboud et al 2010] Abboud, Osama; Zinner, Thomas; Pussep, Konstantin; Oechsner, Simon; Steinmetz, Ralf e Hossfeld, Tobias (2010). *A QoE-Aware P2P Streaming System based on Scalable Video Coding*. Em IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Páginas 1-2, Delft.

[Abboud et al 2011] Abboud, Osama; Zinner, Thomas; Pussep, Konstantin; Al-Sabea, Sabah e Steinmetz, Ralf (2011). *On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems*. Em ACM Multimedia Systems.

[Abeni et al 2010] Abeni, Luca; Kiraly, Csaba; Russo, Alessandro; Biazzini, Marco e Cigno, Renato Lo (2010). *Design and implementation of a generic library for P2P streaming*. Em ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking (AVSTP2P), Páginas 43-48, New York, NY, USA.

[Agboma, Smy e Liotta 2008] Agboma, Florence; Smy, Malcolm e Liotta, Antonio (2008). *QoE analysis of a peer-to-peer television system*. Em IADIS International Telecommunications, Networks and Systems, Páginas 114-119.

[Alhaisoni, Ghanbari e Liotta 2010a] Alhaisoni, Majed; Ghanbari, Mohammed e Liotta, Antonio (2010a). *Scalable P2P Video Streaming*. Em International Journal

of Business Data Communications and Networking (IJBDCN), Volume 6, Edição 3.

[Alhaisoni, Ghanbari e Liotta 2010b] Alhaisoni, Majed; Ghanbari, Mohammed e Liotta, Antonio (2010b). *Resilient P2P Streaming.* Em International Journal On Advances in Networks and Services, Volume 3, Números 1 e 2, Páginas 216-226.

[Allani 2010] Allani, Mouna (2010). *Tree-based message diffusion for managing replicated data.* Em Universite de Lausanne, Tese de D.Sc.

[Alzina, Sarriera e Martinez 2004] Alzina, Rafael Bisquerra; Sarriera, Jorge Castellá e Martinez, Francese (2004). *Introdução a Estatística: Enfoque informático com o pacote estatístico SPSS.* Em Editora Artmed, Páginas 68 e 145.

[Aminu, Gaidamaka e Samuylov 2010] Aminu, A.; Gaidamaka, Y. e Samuylov, A. (2010). *Analytical modeling of P2PTV network.* Em International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Páginas 1115-1120, Moscow.

[Angelides et al 2010] Angelides, Marios C.; Agius, Harry; Iqbal, Razib e Shirmohammadi, Shervin (2010). *Spatiotemporal H.264/AVC Video Adaptation with MPEG-21.* Em The Handbook of MPEG Applications: Standards in Practice, Páginas 205-220.

[ARIN 2012] ARIN (2012). *American Registry for Internet Numbers (ARIN).* Em: <https://www.arin.net/>. Acesso em: 31 dezembro 2012 às 10:26.

[Avramova et al 2011] Avramova, Z.; Vleeschauwer, D. De; Wittevrongel, S. e Bruneel, H. (2011). *Performance analysis of a caching algorithm for a catch-up television service.* Em Multimedia Systems, Volume 17, Número 1, Páginas 5-18.

[Awiphan, Zhou Su e Katto 2010] Awiphan, S.; Zhou Su e Katto, J. (2010). *A contribution-aware multiple parent overlay network for P2P media streaming.* Em 18th International Packet Video Workshop (PV), Páginas 118-125, Hong Kong.

[Barbetta, Reis e Bornia 2009] Barbetta, Pedro Alberto; Reis, Marcelo Menezes e Bornia, Antonio Cezar (2009). *Estatística para cursos de engenharia e informática*. Em Editora Atlas, Edição 2, São Paulo, SP.

[Barrios et al 2011] Barrios, Andrés; Barrios, Matías; Vera, Daniel De; Rodríguez-Bocca, Pablo e Rostagnol, Claudia (2011). *GoalBit: A Free and Open Source Peer-to-Peer Streaming Network*. Em 19th ACM international conference on Multimedia, New York, NY, USA.

[Bertinat et al 2009] Bertinat, María Elisa; Vera, Daniel De; Padula, Darío; Amoza, Franco Robledo; Rodríguez-Bocca, Pablo; Romero, Pablo e Rubino, Gerardo (2009). *GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network*. Em 5th International Latin American Networking Conference, New York, NY, USA.

[Birke et al 2011] Birke, R.; Kiraly, C.; Leonardi, E.; Mellia, M.; Meo, M. e Traverso, S. (2011). *Hose rate control for P2P-TV streaming systems*. Em IEEE International Conference on Peer-to-Peer Computing (P2P), Páginas 202-205, Kyoto.

[BitStream 2013] BitStream (2013). *BitStream*. Em: <http://code.google.com/p/bitstream/>. Acesso em: 2 janeiro 2013 às 14:12.

[BitTorrent 2012] BitTorrent (2012). *BitTorrent - Delivering the World's Content*. Em: <http://www.bittorrent.com/intl/pt/>. Acesso em: 31 dezembro 2012 às 11:46.

[BitTorrent Live 2013] BitTorrent Live (2013). *BitTorrent Live: A New Way to Live Broadcast*. Em: <http://live.bittorrent.com/>. Acesso em: 4 janeiro 2013 às 12:45.

[Bonti, Li e Shi 2011] Bonti, A.; Li, Ming e Shi, Wen (2011). *Improving P2P IPTV random peers search through user similarity*. Em 5th International Conference on Network and System Security (NSS), Páginas 137-144, Milan.

[Boulos et al 2009] Boulos, Fadi; Parrein, Benoit; Callet, Patrick Le e Hands, David S. (2009). *Perceptual Effects of Packet Loss on H.264/AVC Encoded Videos.* Em Fourth International Workshop on Video Processing and Quality Metrics for Consumer Electronics VPQM-09, Scottsdale, Arizona : États-Unis.

[Brandenburg et al 2011] Brandenburg, R. van; Stokking, H. M.; Deventer, M. O. van; Niamut, O. A.; Walraven, F. A.; Netherlands, TNO; Vaishnavi, I.; Netherlands, CWI; Boronat, F. e Montagud, M. (2011). *RTCP for inter-destination media synchronization.* Em Universidad Politecnica de Valencia.

[Carreira et al 2010] Carreira, J.; Pinto, L.; Rodrigues, N.; Faria, S. e Assuncao, P. (2010). *Subjective assessment of frame loss concealment methods in 3D video.* Em 28th Picture Coding Symposium (PCS 2010), Páginas 182-185, 8-10 Dezembro 2010, Nagoya, Japan.

[Chen e Du 2010] Chen, Jun e Du, Xu (2010). *A Layered Semi-Structured Overlay for Supporting VCR-Like Interactions in P2P VoD System.* Em IEEE International Conference on Communications (ICC), Páginas 1-5, Cape Town.

[Choffnes e Bustamantel 2008] Choffnes, David R. e Bustamante, Fabián E. (2008). *Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems.* Em ACM SIGCOMM conference on Data communication, Páginas 363-374, Seattle, Washington, USA.

[Choi, Reaz e Mukherjee 2012] Choi, Joonho; Reaz, Abu (Sayeem) e Mukherjee, Biswanath (2012). *A Survey of User Behavior in VoD Service and Bandwidth-Saving Multicast Streaming Schemes.* Em IEEE Communications Surveys & Tutorials, Volume 14, Número 1, Páginas 156-169.

[Coelho et al 2010] Coelho, Rafael Vieira; Barcellos, Marinho Pilla; Jansch-Pôrto, Ingrid e Gaspary, Luciano (2010). *P2P Streaming de Alta Definição: Análise Quantitativa de Esquemas de Assinatura Digital para Autenticação de Conteúdo.*

Em XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Páginas 3-16, Gramado/RS.

[Courcoubetis et al 2011] Courcoubetis, C.; Dramitinos, M.; Stamoulis, G. D.; Blocq, G.; Miron, A. e Orda, A. (2011). *Inter-carrier interconnection services: QoS, economics and business issues*. Em IEEE Symposium on Computers and Communications (ISCC), Páginas 779-784, Kerkyra.

[Cruvinel et al 2011] Cruvinel, Laercio e Vazão, Teresa (2011). *Improving performance for multimedia traffic with distributed dynamic QoS adaptation*. Em Computer Communications, Volume 34, Edição 10, Páginas 1222-1234.

[Cui, Dai e Xue 2007] Cui, Y.; Dai, L. e Xue, Y. (2007). *Optimizing P2P streaming throughput under peer churning*. Em IEEE Global Telecommunications Conference (GLOBECOM), Páginas 231-235.

[Deshpande, Bawa e Garcia-Molinal 2001] Deshpande, H.; Bawa, M. e Garcia-Molinal, H. (2001). *Streaming Live Media over a Peer-to-peer Network*. Em Stanford InfoLab, Technical Report.

[DropBox 2012] DropBox (2012). *DropBox: Simplify your life*. Em: <https://www.dropbox.com/>. Acesso em: 31 dezembro 2012 às 10:57.

[Eberhard et al 2012] Eberhard, Michael; Palo, Andi; Kumar, Amit; Petrocco, Riccardo; Mapelli, Licio e Uitto, Mikko (2012). *NextSharePC: an open-source BitTorrent-based P2P client supporting SVC*. Em 3rd Multimedia Systems Conference (MMSys), Páginas 101-106, New York, NY, USA.

[Ebrahimi 2010] Ebrahimi, Touradj (2010). *QoE issues in P2P video streaming*. Em ACM workshop on Social, adaptive and personalized multimedia interaction and access (SAPMIA), Florence, Italy.

[Ekmekcioglu et al 2010] Ekmekcioglu, Erhan; Günel, Banu; Dissanayake, Maheshi; Worrall, Stewart T. e Kondoz, Ahmet M. (2010). *A scalable multi-view audiovisual entertainment framework with content-aware distribution*. Em 17th IEEE

International Conference on Image Processing (ICIP), Páginas 2401-2404, 26-29 Setembro 2010, Hong Kong.

[Elliott 1938] Elliott, Ralph Nelson (1938). *The Wave Principle*. Em Republicação (2012), Editora Snowball Publishing, New York, NY, USA.

[Espina et al 2011] Espina, F.; Morato, D.; Izal, M. e Magana, E. (2011). *Improving Video Quality in Network Paths with Bursty Losses*. Em IEEE Global Telecommunications Conference (GLOBECOM), Páginas 1-6, Houston, TX, USA.

[Faria 2010] Faria, Maximiliano Martins de (2010). *Lidando com tráfego egoísta em redes BitTorrent*. Em Centro de Ciências Exatas e Tecnológicas, Universidade Federal do Estado do Rio de Janeiro - UNIRIO, Dissertação de M.Sc.

[Ferreira 2007] Ferreira, Rogério Augusto (2007). *Seqüência de Fibonacci*. Em UNIFIEO, Osasco, SP.

[FlightPath 2013] FlightPath (2013). *FlightPath*. Em: <http://flightpath.austin.googlepages.com/>. Acesso em: 2 janeiro 2013 às 14:11.

[FMJ 2013] FMJ Project (2013). *Freedom for Media in Java*. Em: <http://fmj-sf.net/>. Acesso em: 11 janeiro 2013 às 14:02.

[Fortuna et al 2010] Fortuna, R.; Leonardi, E.; Mellia, M.; Meo, M. e Traverso, S. (2010). *QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs*. Em IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Páginas 1-10, Delft.

[Freecast 2013] Freecast (2013). *Freecast*. Em: <http://www.freecast.org/>. Acesso em: 2 janeiro 2013 às 13:55.

[Fu et al 2010] Fu, T. Z. J.; Leung, Wai-tung; Lam, Pak-yin; Chiu, Dah Ming e Lei, Zhibin (2010). *Perceptual quality assessment of P2P assisted streaming video*

for chunk-level playback controller design. Em 18th International Packet Video Workshop (PV), Páginas 102-109, Hong Kong.

[Garcia 2012] Garcia, Luis Martin (2012). *TCPDump & LibPCap.* Em: <http://www.tcpdump.org/>. Acesso em: 31 dezembro 2012 às 09:31.

[Ghareeb, Ksentini e Viho 2011] Ghareeb, M.; Ksentini, A. e Viho, C. (2011). *An adaptive QoE-based multipath video streaming algorithm for Scalable Video Coding (SVC).* Em IEEE Symposium on Computers and Communications (ISCC), Páginas 824-829, Kerkyra.

[Gitman 2002] Gitman, Lawrence J. (2002). *Princípios de Administração Financeira.* Em Editora Harbra, Edição 7, Páginas 182; 310-311; 329.

[Gonçalves et al 2010] Gonçalves, Glauber Dias; Guimarães, Anna; Vieira, Alex Borges e Almeida, Jussara (2010). *Predição do Nível de Cooperação em Sistemas Par-a-Par de Vídeo ao Vivo a partir de Métricas de Centralidade.* Em XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Ouro Preto, MG.

[Grafl et al 2011] Grafl, Michael; Timmerer, Christian; Hellwagner, Hermann; Negru, Daniel; Borcoci, Eugen; Renzi, Daniele; Mevel, Anne-Lore e Chernilov, Alex (2011). *Scalable Video Coding in Content-Aware Networks: Research Challenges and Open Issues.* Em Trustworthy Internet, Parte 6, Páginas 349-358.

[Gu et al 2011] Gu, Y.; Zong, N.; Zhang, Hui; Zhang, Yunfei; Lei, J.; Camarillo, Gonzalo e Liu, Yong (2011). *Survey of P2P Streaming Applications.* Em Internet-Draft in Internet Engineering Task Force (IETF), Páginas 1-19.

[Guo, Liu e Wang 2010] Guo, Hongfang; Liu, Jiangchuan e Wang, Zongmin (2010). *Frequency-Aware Indexing for Peer-to-Peer On-Demand Video Streaming.* Em IEEE International Conference on Communications (ICC), Páginas 1-5, Cape Town.

[Guo, Lin e Wang 2011] Guo, Hongfang; Lin, Yusong e Wang, Zongmin (2011). *Exploring User-Based Scheduling Strategy for VoD with Frequent Seeks.* Em International Conference on Control, Automation and Systems Engineering (CASE), Páginas 1-4, Singapore.

[Harjoc 2013] Harjoc, Bogdan (2013). *Vuze/Azureus Ordered Piece Downloads.* Em <http://patraulea.com/azureus-ordered-download.patch/#_downloads>. Acesso em: 22 agosto 2013 às 00:59.

[Hecht e Stiller 2010] Hecht, Fabio Victora e Stiller, Burkhard (2010). *Report- and Reciprocity-Based Incentive Mechanisms for Live and On-Demand P2P Video Streaming.* Em The Future Internet Lecture Notes in Computer Science, Volume 6155/2010, Páginas 81-84.

[Hu, Guo e Liu 2011] Hu, Hao; Guo, Yang e Liu, Yong (2011). *Peer-to-Peer Streaming of Layered Video: Efficiency, Fairness and Incentive.* Em IEEE Transactions on Circuits and Systems for Video Technology, Volume 21, Edição 8, Páginas 1013-1026.

[Huang et al 2011] Huang, Nen-Fu; Wang, Ming-Hung; Wang, Tzu-Chien e Peng, Shiu-Shun (2011). *Measuring QoE/QoS of large scale P2P IPTV service.* Em 13th Asia-Pacific Network Operations and Management Symposium (APNOMS), Taipei.

[Huo et al 2010] Huo, Yusong; Su, Yujie; Wang, Zhenhua; Wu, Jun e Ma, Yan (2010). *Measurement based modeling and assessment of QoE index in P2P IPTV services.* Em 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Páginas 1020-1024, Beijing.

[Ibekwe, Klima e Soucek 2011] Ibekwe, M.; Klima, M. e Soucek, P. (2011). *The objective image quality criteria and QoE in the security technology.* Em IEEE International Carnahan Conference on Security Technology (ICCST), Páginas 304-305, Barcelona.

[IETF 2013] IETF (2013). *The Internet Engineering Task Force*. Em: <http://www.ietf.org/>. Acesso em: 16 janeiro 2013 às 21:17.

[Ioannidis, Chaintreau e Messoulié 2009] Ioannidis, Stratis; Chaintreau, Augustin e Massoulié, Laurent (2009). *Optimal and Scalable Distribution of Content Updates over a Mobile Social Network*. Em IEEE INFOCOM, Páginas 1422-1430.

[Juluri, Plissonneau e Medhi 2011] Juluri, Parikshit; Plissonneau, Louis e Medhi, Deep (2011). *Pytomo: a tool for analyzing playback quality of YouTube videos*. Em 23rd International Teletraffic Congress (ITC), Páginas 304-305.

[Kanumuri et al 2006] Kanumuri, Sandeep; Cosman, Pamela C.; Reibman, Amy R. e Vaishampayan, Vinay A. (2006). *Modeling Packet-Loss Visibility in MPEG-2 Video*. Em IEEE Transactions on Multimedia, Volume 8, Edição 2, Páginas 341-355, Abril 2006.

[Khiem, Ravindra e Ooi 2011] Khiem, Ngo Quang Minh; Ravindra e Ooi, Guntur e Wei Tsang (2011). *Towards understanding user tolerance to network latency in zoomable video streaming*. Em 19th ACM international conference on Multimedia (MM), Páginas 977-980, New York, NY, USA.

[Kiraly, Abeni e Lo Cigno 2010] Kiraly, C.; Abeni, L. e Lo Cigno, R. (2010). *Effects of P2P Streaming on Video Quality*. Em IEEE International Conference on Communications (ICC), Páginas 1-5, Cape Town.

[Krishnan e Sitaraman 2012] Krishnan, Shunmuga e Sitaraman, Ramesh (2012). *Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs*. Em ACM Internet Measurement Conference (IMC), Boston, MA.

[Kulatunga et al 2012] Kulatunga, Chamil; Botvich, Dmitri; Balasubramaniam, Sasitharan e Donnelly, William (2012). *Analysis of Block-aware Peer Adaptations in Substream-based P2P*. Em Lecture Notes of the Institute for Computer Scien-

ces, Social Informatics and Telecommunications Engineering, Volume 66, Páginas 14-27.

[Kulatunga et al 2010] Kulatunga, Chamil; Botvich, Dmitri; Balasubramaniam, Sasitharan e Donnelly, William (2010). *Analysis of Block-aware Peer Adaptations in Substream-based P2P.* Em International ICST Conference on Broadband Communications, Networks and Systems (BroadNets), Athens, Greece.

[Lan et al 2011] Lan, Shanzhen; Zhang, Qi; Zhang, Xinggong e Guo, Zongming (2011). *Dynamic asynchronous buffer management to improve data continuity in P2P live streaming.* Em 3rd International Conference on Computer Research and Development (ICCRD), Volume 2, Páginas 65-69, Shanghai.

[Li, Liang e Wu 2012] Li, Aikun; Liang, Yi e Wu, Di (2012). *Utilizing Layered Taxation to provide incentives in P2P streaming systems.* Em Journal of Systems and Software.

[Li 2004] Li, Jin (2004). *PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System.* Em Microsoft Research (MSR), Technical Report.

[Li, Zhang e Yuan 2011] Li, Y.; Zhang, Y. e Yuan, R. (2011). *Measurement and analysis of a large scale commercial mobile internet tv system.* Em ACM SIGCOMM conference on Internet measurement conference (IMC), Páginas 209-224, New York, NY, USA.

[Liang et al 2010] Liang, Chao; Fu, Zhenghua; Liu, Yong e Wu, Chai Wah (2010). *Incentivized Peer-Assisted Streaming for On-Demand Services.* Em IEEE Transactions on Parallel and Distributed Systems, Volume 21, Edição 9, Páginas 1354-1367.

[Liang e Nahrstedt 2006] Liang, J. e Nahrstedt, K. (2006). *RandPeer: Membership management for QoS sensitive peer-to-peer applications.* Em IEEE INFOCOM, Páginas 1-10.

[Liao et al 2006] Liao, Xiaofei; Jin, Hai; Liu, Yunhao; Ni, Lionel M. e Deng, Dafu (2006). *AnySee: Peer-to-Peer Live Streamingm.* Em IEEE INFOCOM.

[Linode 2012] Linode (2012). *Linode.com: Deploy and Manage Linux Virtual Servers in the Linode Cloud.* Em: <http://www.linode.com/>. Acesso em: 31 dezembro 2012 às 10:49.

[Liu et al 2008] Liu, Jiangchuan; Rao, S.G.; Li, Bo e Zhang, Hui (2008). *Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast.* Em IEEE Journals & Magazines, Volume 96, Edição 1, Páginas 11-24.

[Lo et al 2005] Lo, Virginia; Zhou, Dayi; Liu, Yuhong; GauthierDickey, C. e Li, Jun (2005). *Scalable supernode selection in peer-to-peer overlay networks.* Em Second International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P), Páginas 18-25.

[Lobb et al 2009] Lobb, Richard; Silva, Ana Paula Couto da; Leonardi, Emilio; Mellia, Marco e Meo, Michela (2009). *Adaptive Overlay Topology for Mesh-Based P2P-TV Systems.* Em The 19th International Workshop on Network and Operating Systems Support for Digital Audio and Video (ACM NOSSDAV 2009).

[Luft et al 2007] Luft, Caroline Di Bernardi; Sanches, Sabrina de Oliveira; Mazo, Giovana Zarpellon e Andrade, Alexandro (2007). *Versão brasileira da Escala de Estresse Percebido: tradução e validação para idosos.* Em Revista Saúde Pública, Páginas 606-615.

[Macedo et al 2010] Macedo, José; Nunes, Ivan de Oliveira; Silva, Ebenezer Nogueira da; Floriano, Alan Silva da Paz; Gomes, Roberta Lima e Martinello, Magnos (2010). *Análise Quantitativa baseada em Medições de Sistemas P2P para Video Streaming.* Em XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Ouro Preto, MG.

[Mathieu et al 2010] Mathieu, B.; Paris, P.; Guelvouit, G. Le e Rouibia, S. (2010). *A Secure and Legal Network-Aware P2P VoD System.* Em Fifth International

Conference on Internet and Web Applications and Services (ICIW), Páginas 194-199, Barcelona.

[Matsura 2006] Matsura, Eduardo (2006). *Comprar ou Vender? Como Investir na Bolsa Utilizando Análise Gráfica*. Em Editora Saraiva, Edição 4, São Paulo, SP.

[Mendonça 2013] Mendonça, André (2013). *O que é Teoria das Ondas de Elliott?*. Em: <http://www.elliottbrasil.com/teoriadeelliott.php>. Acesso em: 22 agosto 2013 às 01:06.

[Menkovski, Exarchakos e Liotta 2010] Menkovski, V.; Exarchakos, G. e Liotta, A. (2010). *Machine Learning Approach for Quality of Experience Aware Networks*. Em 2nd International Conference on Intelligent Networking and Collaborative Systems (INCOS), Páginas 461-466, Thessaloniki.

[Merani e Saladino 2010] Merani, Maria Luisa e Saladino, Daniela (2010). *Live Video and IP-TV*. Em Handbook of Peer-to-Peer Networking, Parte 8, Páginas 985-1024.

[Michel et al 2010] Michel, M.; Agarwal, S.; Kellerer, W. e Feldmann, A. (2010). *Toward QoE-Aware Optimum Peer Cache Sizes for P2P Video-on-Demand Systems*. Em IEEE International Conference on Communications (ICC), Páginas 1-5.

[Miranda e Figueiredo 2009] Miranda, Marcio N. e Figueiredo, Daniel R. (2009). *A Preferential Attachment Model for Tree Construction in P2P Video Streaming*. Em WPerformance.

[Mok, Chan e Chang 2011] Mok, Ricky K. P.; Chan, Edmond W. W. e Chang, Rocky K. C. (2011). *Measuring the Quality of Experience of HTTP Video Streaming*. Em IFIP/IEEE International Symposium on Integrated Network Management (IM), Páginas 485-492, 23-27 Maio 2011, Dublin.

[Mokarzel et al 2010] Mokarzel, Marcos P.; Rossi, Sandro M.; Sassi, André B. e Rocha, Mônica L. (2010). *Redução do tempo de zapping em serviços IPTV sobre*

redes GPON utilizando vídeos escaláveis. Em XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), Páginas 699-712, Gramado/RS.

[Moraes et al 2008] Moraes, Igor M.; Campista, Miguel Elias M.; Moreira, Marcelo D. D.; Rubinstein, Marcelo G.; Costa, Luís Henrique M. K. e Duarte, Otto Carlos M. B. (2008). *Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios*. Em Minicursos do XXVI Simpósio Brasileiro de Redes de Computadores (SBRC), Páginas 115-171, Rio de Janeiro.

[Mu et al 2009] Mu, Mu; Gostner, Roswitha; Mauthe, Andreas; Tyson, Gareth e Garcia, Francisco (2009). *Visibility of individual packet loss on H.264 encoded video stream – A user study on the impact of packet loss on perceived video quality*. Em Multimedia Computing and Networking, Volume 7253, 18 Janeiro 2009, San Jose, CA.

[Mwela e Adebomi 2010] Mwela, John Samson e Adebomi, Oyekanlu Emmanuel (2010). *Impact of Packet Loss on the Quality of Video Stream Transmission*. Em School of Computing at Blekinge Institute of Technology, Maio 2010, Dissertação de M.Sc.

[NetFlix 2013] NetFlix (2013). *NetFlix: Que velocidade de Internet banda larga preciso ter para assistir online?*. Em: <https://signup.netflix.com/HowItWorks>. Acesso em: 2 janeiro 2013 às 19:12.

[Nguyen 2011] Nguyen, The Tung (2011). *An environment for peer-to-peer high performance computing*. Em University of Toulouse, Tese de D.Sc.

[Nodezilla 2013] Nodezilla (2013). *Nodezilla Grid Network*. Em: <http://www.nodezilla.net/rtp_guide.html>. Acesso em: 2 janeiro 2013 às 14:06.

[NS-2 2013] NS-2 (2013). *NS-2: Wiki Main Page*. Em: <http://nsnam.isi.edu/nsnam/index.php/Main_Page>. Acesso em: 15 janeiro 2013 às 12:08.

[Oliveira 2010] Oliveira, João Ferreira D'Araújo e (2010). *Super nós em sistemas P2P de distribuição de mídia ao vivo.* Em Universidade Federal de Minas Gerais. Departamento de Ciência da Computação, Dissertação de M.Sc, Belo Horizonte/MG.

[Oracle 2012] Oracle (2012). *MySQL.* Em: <http://www.mysql.com/>. Acesso em: 31 dezembro 2012 às 09:45.

[Oracle 2013] Oracle (2013). *Java Media Framework API.* Em: <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html>. Acesso em: 11 janeiro 2013 às 14:07.

[P2P-Radio 2013] P2P-Radio (2013). *P2P-Radio: Peer to Peer Streaming.* Em: <http://p2p-radio.sourceforge.net/>. Acesso em: 2 janeiro 2013 às 14:03.

[Payberah, Dowling e Haridi 2011] Payberah, A.H.; Dowling, J. e Haridi, S. (2011). *GLive: The Gradient Overlay as a Market Maker for Mesh-Based P2P Live Streaming.* Em 10th International Symposium on Parallel and Distributed Computing (ISPDC), Páginas 153-162, Cluj Napoca.

[Pereira, Vazão e Rodrigues 2012] Pereira, Ricardo Lopes; Vazão, Teresa e Rodrigues, Rodrigo (2012). *Adaptive Search Radius – Using hop count to reduce P2P traffic.* Em Computer Networks, Volume 56, Edição 2, Páginas 642-660, 2 Fevereiro 2012.

[Petrocco et al 2011] Petrocco, R.; Eberhard, M.; Pouwelse, J. e Epema, D. (2011). *Deftpack: A Robust Piece-Picking Algorithm for Scalable Video Coding in P2P Systems.* Em IEEE International Symposium on Multimedia (ISM), Páginas 285-292, Dana Point, CA.

[PlanetLab 2012] PlanetLab (2012). *PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services.* Em: <http://www.planet-lab.org/>. Acesso em: 31 dezembro 2012 às 09:56.

[Polaczyk e Cholda 2010] Polaczyk, Bartosz e Cholda, Piotr (2010). *BitTorrent Traffic Localization via Operator-related Information.* Em IEEE International Conference on Communications (ICC), Cape Town, South Africa.

[Pouwelse et al 2008] Pouwelse, J. A.; Garbacki, P.; Wang, J.; Bakker, A.; Yang, J.; Iosup, A.; Epema, D. H. J.; Reinders, M.; Steen, M. R. Van e Sips, H. J. (2008). *TRIBLER: a social-based peer-to-peer system.* Em Concurrency and Computation: Practice and Experience, Volume 20, Edição 2, Páginas 127-138.

[Pussep et al 2010] Pussep, K.; Abboud, O.; Gerlach, F.; Steinmetz, R. e Strufe, T. (2010). *Adaptive server allocation for peer-assisted Video-on-Demand.* Em IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), Páginas 1-8, Atlanta, GA.

[Rejaie e Stafford 2004] Rejaie, R. e Stafford, S. (2004). *A framework for architecting peer-to-peer receiver-driven overlays.* Em International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Páginas 42-47.

[Ribadeneira 2007] Ribadeneira, Alexander F. (2007). *An Analysis of the MOS under Conditions of Delay, Jitter and Packet Loss and an Analysis of the Impact of Introducing Piggybacking and Reed Solomon FEC for VOIP.* Em School of Computing at Blekinge Institute of Technology, Maio 2007, Dissertação de M.Sc.

[RIPE 2012] RIPE (2012). *RIPE Network Coordination Centre.* Em: <http://www.ripe.net/>. Acesso em: 31 dezembro 2012 às 10:20.

[RNP 2013] RNP - Rede Nacional de Ensino e Pesquisa (2013). *PlanetLab: Um laboratório virtual para pesquisa em redes.* Em: <http://www.rnp.br/pd/planetlab/>. Acesso em: 18 janeiro 2013 às 11:17.

[Rodríguez-Bocca 2008] Rodríguez-Bocca, Pablo (2008). *Quality-centric design of Peer-to-Peer systems for live-video broadcasting.* Em l'Université de Rennes, Tese de D.Sc.

[Rocha e Menasché 2013] Rocha, Antonio Augusto de Aragão e Menasché, Daniel Sadoc (2013). *Mensagem dos Coordenadores do WP2P+*. Em XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos: Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo, 6 a 10 Maio 2013, Brasília, DF, BR.

[Rossi e Veglia 2011] Rossi, D. e Veglia, P. (2011). *Assessing the Impact of Signaling on the QoE of Push-Based P2P-TV Diffusion Algorithms*. Em 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Páginas 1-5, Paris.

[Roverso et al 2011] Roverso, Roberto (2011). *Design and Implementation of Centrally-Coordinated Peer-to-Peer Live-streaming*. Em KTH School of Information and Communication Technology, Sweden.

[Savas et al 2012] Savas, Saadet Sedef; Gürler, Cihat Göktuğ; Tekalp, Ahmet Murat; Ekmekcioglu, Erhan; Worrall, Stewart e Kondoz, Ahmet (2012). *Adaptive streaming of multi-view video over P2P networks*. Em Signal Processing: Image Communication.

[Savas, Gürler e Tekalp 2011] Savas, Saadet Sedef; Gurler, Cihat Goktug e Tekalp, Ahmet Murat (2011). *Adaptive Multi-view Video Streaming over P2P Networks Considering Quality of Experience*. Em ACM Workshop on Social and Behavioural Networked Media Access (SBNMA), Páginas 53-58, New York, NY, USA.

[Schatz, Hobfeld e Casas 2012] Schatz, Raimund; Hobfeld, Tobias e Casas, Pedro (2012). *Passive YouTube QoE Monitoring for ISPs*. Em Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Páginas 358-364, Palermo.

[SCVI.NET 2013a] SCVI.NET (2013a). *Open Source P2P Video Streaming Software*. Em: <http://www.scvi.net/stream/soft.htm>. Acesso em: 4 janeiro 2013 às 11:16.

[SCVI.NET 2013b] SCVI.NET (2013b). *Proprietary Software for P2P Video Streaming Software*. Em: <http://www.scvi.net/stream/soft2.htm>. Acesso em: 4 janeiro 2013 às 19:30.

[Shahriar, Qiu e Jaumard 2012] Shahriar, I,; Qiu, D. e Jaumard, B. (2012). *Analysis of HnH Model for Live Streaming Channels with a Small Number of Viewers*. Em Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Páginas 140-147, 12 a 14 Novembro 2012, Victoria, BC.

[Silva et al 2010] Silva, Ana Paula Couto da; Leonardi, Emilio; Mellia, Marco e Meo, Michela (2010). *Chunk distribution in Mesh-Based Large Scale P2P Streaming Systems: a Fluid Approach*. Em IEEE Transactions on Parallel and Distributed Systems "To appear", ISSN: 1045-9219, 2010.

[Silva et al 2008a] Silva, Ana Paula Couto da; Varela, Martin; Silva, Edmundo de Souza e; Leão, Rosa Maria Meri e Rubino, Gerardo (2008). *Quality Assessment of Interactive Real Time Voice Applications*. Em Computer Networks Journal, Volume 52, Edição 6, Abril 2008, Páginas 1179-1192.

[Silva et al 2008b] Silva, Ana Paula Couto da; Rodriguez-Bocca, Pablo e Rubino, Gerardo (2008). *Optimal Quality-of-Experience Design for a P2P Multi-source Video Streaming*. Em International Conferenceon Communications (ICC), Maio 2008.

[Sourceforge 2013] Sourceforge (2013). *Java Bittorrent API*. Em: <http://bitext.sourceforge.net/>. Acesso em: 4 janeiro 2013 às 12:37.

[Staelens et al 2009] Staelens, Nicolas; Vermeulen, Brecht; Moens, Stefaan; Macq, Jean-François; Lambert, Peter; Walle, Rik Van de e Demeester, Piet (2009). *Assessing the influence of packet loss and frame freezes on the perceptual quality of full length movies*. Em 4th International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM 2009), Scottsdale, AZ, USA.

[Statovci-Halimi e Franzl 2011] Statovci-Halimi , Brikena e Franzl, Gerald (2011). *QoS differentiation and Internet neutrality: A controversial issue within the future Internet challenge*. Em Telecommunication Systems.

[Stream 2 Stream 2013] Stream 2 Stream (2013). *Stream 2 Stream*. Em: <http://s2s.sourceforge.net/>. Acesso em: 2 janeiro 2013 às 14:00.

[Theory 2013] Theory (2013). *BitTorrent Protocol Specification v1.0*. Em: <http://wiki.theory.org/BitTorrentSpecification>. Acesso em: 8 janeiro 2013 às 14:33.

[Tian et al 2013] Tian, Guibin; Xu, Yang; Liu, Yong e Ross, Keith (2013). *Mechanism Design for Dynamic P2P Streaming*. Em 13th IEEE International Conference on Peer-to-Peer Computing, Setembro 2013.

[Trevbus 2013] Trevbus (2013). *Trevbus*. Em: <http://www.trevbus.org/>. Acesso em: 2 janeiro 2013 às 14:07.

[VideoLAN 2013a] VideoLAN Organization (2013a). *VideoLAN Organization: Um projeto e uma organização sem fins lucrativos, composta de voluntários, desenvolvendo e promovendo soluções multimídia livres e gratuitas*. Em: <http://www.videolan.org/>. Acesso em: 2 janeiro 2013 às 14:14.

[VideoLAN 2013b] VideoLAN Organization (2013b). *VideoLAN Wiki: Developers Corner*. Em: <http://wiki.videolan.org/Developers_Corner>. Acesso em: 10 janeiro 2013 às 22:36.

[VideoTyrant 2013] VideoTyrant (2013). *VideoTyrant*. Em: <http://sourceforge.net/projects/videotyrant/>. Acesso em: 2 janeiro 2013 às 14:09.

[Vorren 2006] Vorren, Sander Sunde (2006). *Subjective quality evaluation of the effect of packet loss in High-Definition Video*. Em Norwegian University of Science and Technology, Junho 2006, Dissertação de M.Sc.

[Vuze 2012] Vuze (2012). *Vuze: O melhor aplicativo de bittorrent do mundo*. Em: <http://www.vuze.com/?lang=pt_BR>. Acesso em: 31 dezembro 2012 às 11:46.

[Vuze 2013a] Vuze (2013a). *Vuze: Using Eclipse*. Em: <http://wiki.vuze.com/w/Using_Eclipse>. Acesso em: 8 janeiro 2013 às 17:36.

[Vuze 2013b] Vuze (2013b). *Vuze: Plugin Development Guide*. Em: <http://wiki.vuze.com/w/Plugin_Development_Guide>. Acesso em: 8 janeiro 2013 às 17:37.

[Vuze 2013c] Vuze (2013c). *Vuze Plus: Play Now*. Em: <http://www.vuze.com/features/playnow>. Acesso em: 8 janeiro 2013 às 17:46.

[Vuze 2013d] Vuze (2013d). *Vuze Plus: FAQ Play Now*. Em: <http://wiki.vuze.com/w/FAQ_Play_Now>. Acesso em: 8 janeiro 2013 às 18:09.

[Vuze 2013e] Vuze (2013e). *Sequential downloading is bad*. Em: <http://wiki.vuze.com/w/Sequential_downloading_is_bad>. Acesso em: 13 setembro 2013 às 19:50.

[Wang et al 2011] Wang, Hui; Sun, Zhigang; Wang, Baosheng; Zhang, Hua e Liu, Li (2011). *A Point-to-Multipoint Distribution Mechanism for IPTV Video Network*. Em 6th IEEE International Conference on Networking, Architecture and Storage (NAS), Páginas 214-219, Dalian, Liaoning.

[Wang 2006] Wang, Yubing (2006). *Survey of Objective Video Quality Measurements*. Em EMC Corporation Hopkinton, USA.

[Welch e Welch 2005] Welch, Suzy e Welch, Jack (2005). *Paixão por Vencer*. Em Editora Elsevier, Edição 1.

[Wen et al 2011] Wen, Zheng; Liu, Nianwang; Yeung, K. L. e Lei, Zhibin (2011). *Closest Playback-Point First: A New Peer Selection Algorithm for P2P VoD Systems*. Em IEEE Global Telecommunications Conference (GLOBECOM), Páginas 1-5, Houston, TX, USA.

[Xie et al 2008] Xie, Haiyong; Yang, Y. Richard; Krishnamurthy, Arvind; Liu, Yanbin e Silberschatz, Avi (2008). *P4P: Provider Portal for Applications*. Em ACM SIGCOMM conference on Data communication, Páginas 351-362, New York, NY, USA.

[Xu et al 2010] Xu, Tianyin; Ye, Baoliu; Wang, Qinhui; Li, Wenzhong; Lu, Sanglu e Fu, Xiaoming (2010). *APEX: A personalization framework to improve quality of experience for DVD-like functions in P2P VoD applications*. Em 18th International Workshop on Quality of Service (IWQoS), Páginas 1-9, Beijing.

[YouTube 2013] YouTube (2013). *YouTube: Broadcast Yourself*. Em: <http://br.youtube.com/>. Acesso em: 18 janeiro 2013 às 12:04.

[Zhang et al 2005a] Zhang, X.; Liu, J.; Li, B. e Yum, T. (2005a). *CoolStreaming/DONet: A Datadriven Overlay Network for Efficient Live Media Streaming*. Em IEEE INFOCOM.

[Zhang et al 2005b] Zhang, X.; Liu, J.; Li, B. e Yum, T.-S. P. (2005b). *CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming*. Em IEEE INFOCOM, Páginas 2102-2111.

[Zhu et al 2011] Zhu, Xun; Deng, Hongtao; Chen, Zheng e Yang, Hongyun (2011). *Design of Large-Scale Video Surveillance System Based on P2P Streaming*. Em 3rd International Workshop on Intelligent Systems and Applications (ISA), Páginas 1-4, Wuhan.