# S4Q: Um Algoritmo para Seleção de Vizinhos Baseada em QoE para Sistemas P2P de Transmissão de Vídeo

**Peron Rezende de Sousa**[1], **Sidney Cunha de Lucena**[2], **Morganna Carmem Diniz**[2], **Antonio Augusto de Aragão Rocha**[1], **Daniel Sadoc Menasché**[3]

[1]Instituto de Computação
Universidade Federal Fluminense (UFF)
[2]Centro de Ciências Exatas e Tecnológicas
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
[3]Departamento de Ciência da Computação
Universidade Federal do Rio de Janeiro (UFRJ)

{sidney, morganna}@uniriotec.br,
{prezende, arocha}@ic.uff.br, sadoc@dcc.ufrj.br

***Abstract.*** *Much research has been conducted in order to provide certain levels of QoS to P2P networks, for they represent a scalable and cost-effective alternative. Among these studies met the challenge of selecting peers. In this work we present a new algorithm in this area. It makes decisions with the aid of a new QoE metric, called "stress level" and also presented here. Despite the new metric has not yet been validated with users, it was possible to apply it and get positive results. Our proposal comes to have, at least, 19% fewer pieces of video absences at the time of reproduction other algorithms of the state of the art and a more rapid decrease in "stress level" 32%.*

***Resumo.*** *Muitas pesquisas tem sido realizadas com o intuito de prover certos níveis de QoS as redes P2P, por elas representarem uma alternativa escalável e de baixo custo. Entre esses estudos encontramos o desafio da seleção de pares. Neste trabalho apresentamos um novo algoritmo nessa área. Ele toma suas decisões com o auxílio de uma nova métrica em QoE, denominada "nível de estresse" e também apresentada aqui. Apesar da nova métrica ainda não ter sido validada com usuários, já foi possível aplicá-la e obter resultados positivos. Nossa proposta chega a ter, pelo menos, 19% menos ausências de pedaços de vídeo no momento da reprodução que outros algoritmos do estado da arte e uma redução no "nível de estresse" 32% mais rápida.*

## 1. Introdução

As aplicaccões de distribuição de vídeo, sem dúvida, estão entre as de maior sucesso na Internet atual. Além dos sistemas Par-a-Par (*Peer-to-Peer* - P2P), existem 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.

Assim como as demais soluções, o P2P nem sempre consegue oferecer determinados níveis de qualidade de serviço (*Quality of Service - QoS*). A transmissão de vídeo requer uma grande largura de banda passante e é sensível ao atraso, *jitter* e perda de pacotes. Garantir valores adequados a esses indicadores de QoS e escalabilidade são grandes

desafios para qualquer arquitetura que venha ser empregada [Moraes et al. 2008]. Quando se fala de sistemas P2P para transmissão de vídeo, um mecanismo exerce um papel fundamental na eficiência do seu funcionamento: a seleção de vizinhos dos pares da rede. Selecionar estrategicamente os melhores pares pode ser essencial para se alcançar o nível de QoS requerido pela aplicação.

Diferentes mecanismos já foram propostos na literatura com o objetivo de tornar mais eficiênte a seleção de pares [Xie et al. 2008, Choffnes and Bustamante 2008, Polaczyk and Cholda 2010]. No entanto, até o limite do nosso conhecimento, não foram encontradas na literatura propostas para seleção de pares que façam uso de métricas explícitamente associadas à qualidade da experiência do usuário (*Quality of Experience - QoE*). Podemos encontrar uma idéia semelhante em [Ghareeb, Ksentini e Viho 2011], onde o sistema se baseia em avaliações de 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. A manutenção dos caminhos selecionados é, então, feita 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*). Outro parecido é o de [Rosário et al 2013] que molda o roteamento de uma rede *wireless* por meio de QoE.

Nossa implementação, intitulada Seleção por Qualidade (S4Q), verifica em cada participante de uma rede BitTorrent (um tipo de implementação P2P), com quais outros participantes foi mantida uma comunicação contínua ao longo do tempo, formando com isso o que será conhecido ao longo deste texto como Lista de Pares Estáveis (LPE). Dessa forma cada par da rede poderá ter uma LPE e cada uma delas pode ser útil aos outros pares se partirmos do pressuposto que, caso os integrantes da LPE de determinado par estejam tendo uma boa "experiência" com suas respectivas LPEs, existe uma chance desse par também ter caso ele utilize a LPE dos integrantes de sua LPE. Logo, mecanismos de seleção de pares podem tirar proveito dessa informação. Restava apenas saber como definir a qualidade da "experiência" e atribuir essa impressão as LPEs. Com esse propósito desenvolvemos uma nova metodologia, denominada Avaliação Áurea de Qualidade ($A^2Q$). Nossa metodologia produz uma métrica em QoE, que está relacionada a ausência de pedaços do vídeo no momento da reprodução, cujo resultado denominamos "nível de estresse". Com isso o QoE obtido é atribuído à LPE. Ao promover a troca de LPEs com boa QoE esperamos realizar transmissões mais rápidas e com mínimo de interrupções.

A eficiência da nossa implementação é comparada com o BitTorrent tradicional e com outros três algoritmos do estado da arte (Ono, P4P e Yukka). Os resultados mostram que nossa proposta chega a ter, pelo menos, 19% menos ausências de pedaços de vídeo no momento da reprodução que os outros algoritmos e uma redução no "nível de estresse" 32% mais rápida. Com a vantagem do S4Q não depender de informações obtidas de fontes externas à rede P2P.

Este trabalho possui a seguinte organização: a Seção 2 discute os termos e as teorias que formam a base dessa pesquisa; a Seção 3 explica a nova métrica e o processo de escolha dos pares; a Seção 4 descreve o ambiente de experimentação, apresenta os resultados e uma análise dos mesmos; por último, a Seção 5 descreve as conclusões.

## 2. Fundamentação Teórica

O BitTorrent é um protocolo P2P, criado por Bram Cohen, que tem o objetivo de favorecer o compartilhamento de conteúdo entre os usuários (*peers*), 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*) [Moraes et al. 2008].

Na distribuição, o arquivo (ou *bundle* de arquivos) 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. A sequência de códigos *hash* e o endereço do *tracker* são informações obtidas no arquivo de metadados (.torrent). Com essas informações um *leecher* pode entrar em um *swarm*.

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*).

Repare que seguindo a estratégia acima os pedaços tendem a chegar de forma desordenada, prejudicando o interesse de quem tem a intenção de assistir um vídeo com BitTorrent durante o *download*. No decorrer deste trabalho vamos nos refererir a essa estratégia como Algoritmo de Seleção Aleatória de Pedaços (ASAP).

Não demorou muito para surgir propostas à requisição ordenada de pedaços. Essa idéia consegue melhorar a reprodução de um vídeo durante sua transmissão, mas provoca uma maior latência e outros tipos de problemas. Neste trabalho vamos nos referir a essa estratégia como Algoritmo de Seleção Sequencial de Pedaços (ASSP).

Sobre o modelo tradicional de seleção de pares do BitTorrent, podemos dizer que ele não utiliza uma central de alocação de recursos, isto é, cada par é responsável pela maximização do seu *download* e escolha dos pares para os quais realizará *upload* seguindo a estratégia *tit-for-tat*. Nessa estratégia os pares não colaborativos são "afogados" (*choke*), ou seja, deixam de receber *upload*. De tempos em tempos um par afogado é escolhido aleatoriamente para voltar a receber dados.

Entre as diversas implementações do BitTorrent escolhemos para este trabalho o Vuze (atual Azureus), por ser *open source*, multiplataforma, por aceitar acréscimos de funcionalidades por meio de *PlugIns*, por ter sido a escolha dos outros algoritmos aqui avaliados e por possuir uma versão chamada Vuze Play Now, que permite a visualização de um vídeo durante seu *download*.

### 2.1. Algoritmos para Seleção de Pares

Nesta seção apresentamos 3 algoritmos de seleção de pares propostos na literatura. Iremos comparar nossa proposta contra estas 3 outras soluções e com o método tradicional do BitTorrent.

Entre as soluções com os melhores resultados e mais referenciadas, sobre o desafio da seleção de pares, está o **P4P**, que usa dados fornecidos pelos ISPs (*Internet Service Providers*) sobre suas redes para identificar pares melhores. Nem todos os ISPs têm

suporte para P4P, portanto seu uso nem sempre 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 [Xie et al. 2008].

Também encontramos o **Ono** entre as mais difundidas. Sua estratégia é fazer uso de informações sobre o redirecionamento das CDNs para identificar colegas próximos e potencialmente acelerar *downloads*, além de reduzir o trafégo entre ISPs. Segundo [Choffnes and Bustamante 2008], esse método consegue aumentar a taxa média de *download* em até 31%.

Outra proposta é o método **Yukka**. Essa solução realiza consultas aos *Regional Internet Registries* (RIRs), para promover agrupamentos geográficos a partir da atribuição de uma nota de similaridade entre os pares. Os resultados em [Polaczyk and Cholda 2010] mostram uma redução de até 25% no tempo de *download*.

### 2.2. *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. Diferentemente das métricas objetivas essas medidas trabalham com o subjetivo, pois dependem da opinião do usuário. No entanto, podemos relacioná-las às medidas objetivas de QoS.

Entre os estudos sobre os métodos usados como métrica para QoE encontramos a 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].

Outras métricas de qualidade são encontradas em [Wang 2006], conforme Tabela 1. Em seu trabalho, Wang descreve o PSNR (*Peak-Signal-to-Noise-Ratio*), desenvolvido inicialmente para avaliação de fotografias e depois adaptado para avaliação de vídeos. Apesar de simples, o PSNR é considerado pouco assertivo na correlação com métodos subjetivos. Sobre o MPQM (*Moving Pictures Quality Metric*), o autor ressalta que o mesmo apresenta uma grande margem de erro. Wang também destaca que a performace da avaliação do SSIM (*Structural Similarity Index*), no momento da transmissão de um vídeo, é desconhecida e, tanto esta métrica quanto a VQM (*Video Quality Metric*), precisam conferir o vídeo recebido com o original. O VQM mede ofuscamento, movimento não natural, ruído global, distorção de bloco e distorção da cor, e os combina em uma única métrica que tem forte correlação com a avaliação subjetiva da qualidade de vídeo. Por fim, temos o NQM (*Noise Quality Measure*) que não possui nenhum estudo que relacione seus resultados com a percepção dos usuários.

As métricas da Tabela 1 e a PSQA não foram aqui adotadas devido a complexa implementação exigida. 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 [Elliott 1938]. Nossa proposta se baseia na Teoria de Elliott para medir a QoE.

**Tabela 1. *Métricas de Qualidade em Vídeo* [Wang 2006]**

| Métrica de Qualidade | Complexidade Matemática | Correlação com Métodos Subjetivos |
|---|---|---|
| PSNR | Simples | Ruim |
| MPQM | Complexo | Instável |
| VQM | Muito Complexo | Bom |
| SSIM | Complexo | Razoável |
| NQM | Complexo | Desconhecido |

## 3. Proposta

Nesta seção descrevemos o funcionamento do algortimo, mas antes vamos explicar a razão de algumas escolhas. Para avaliarmos a qualidade de um vídeo de forma mais simples, decidimos nos apoiar na ausência de pedaços e não na ausência de quadros, como foi sugerido por [Rodríguez-Bocca 2008]. Sobre a taxa de reproção do vídeo, consideramos as taxas comuns utilizadas, por exemplo pelo NetFlix, que são 512 Kbits/s (baixa resolução) e 1.536 Kbits/s (alta definição). Por uma questão prática, neste trabalho optamos pela primeira. Considerando que o protocolo BitTorrent utiliza pedaços de 512 KBytes, podemos concluir que, em vídeos de baixa resolução, cada pedaço conterá cerca de 8 segundos. Como um vídeo, por padrão, trabalha a uma taxa de 23 quadros por segundo, cada um destes pedaços conterá cerca de $8 \times 23 = 184$ quadros.

A perda de um único pedaço, contendo 8 segundos de vídeo, pode parecer muito se comparamos esse tempo com o que foi avaliado em [Krishnan and Sitaraman 2012], nesse trabalho os autores estudaram o impacto da qualidade na transmissão de vídeo utilizando extensivos *traces* da rede Akamai, que incluem 23 milhões de visualizações feitas por 6,7 milhões de visitantes. O artigo mostra que os espectadores abandonam o vídeo se este levar mais do que dois segundos para iniciar e 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. Nossa proposta não ataca o tempo de inicialização, nesse ponto é preciso atentar para certas diferenças entre as redes P2P e as CDNs. No entanto, o trabalho sobre a rede Akamai serviu de inspiração para a metodologia de inferência de QoE proposta a seguir, pois nos ajudou a perceber um fato relacionado a variação da QoE no tempo e que torna plausível o tempo de 8 segundos.

Também existe a questão dos *Coders-Decoders (CODEC)* de vídeo, que podem resultar em variações na quantidade de quadros, logo segundos, por pedaços e efeitos diversos na visualização que tem relação com a estratégia utilizada pelos formatos como, por exemplo, perdas de *key-frames* predecessores de *intra-frames*. Com o objetivo de simplificar a análise não utilizamos variados tipos de formatos.

A literatura sobre relações entre perdas de pedaços e QoE é escassa, o mesmo vale para perdas de quadros. Resolvemos estabelecer um paralelo com os trabalhos [Agboma, Smy e Liotta 2008, Mwela and Adebomi 2010]. Esses artigos pontuaram a percepção dos usuários sobre perdas de pacotes, fazendo uso de avaliações subjetivas baseadas em *Mean Opinion Score (MOS)*. Um MOS específico para perdas de pedaços será elaborado em trabalhos futuros.

A Tabela 2 foi elaborada empiricamente considerando que 10% de ausências é o máximo tolerado. Ela foi dividida em 3 níveis por ser o padrão adotado na elaboração das Escalas de Estresse Percebido (*Perceived Stress Scale - PSS*). Cada rodada do nosso experimento leva 2.000 segundos, mesmo tempo utilizado no trabalho com o GoalBit [Bertinat et al. 2009], onde 10% equilave a 25 pedaços. Reduzimos o teto para 24 pedaços por ser um valor divisível por 3 e para não produzir um acumulado de Fibonacci muito alto. Nos trabalhos de [Agboma, Smy e Liotta 2008, Mwela and Adebomi 2010] podemos ver que o MOS é muito baixo quando o sistema sofre 10% de perda de pacotes, por isso decidimos fazer esse vínculo entre o percentual de perdas de pacotes e o percentual de ausência de pedaços.

**Tabela 2. Escala de estresse com três níveis**

| Níveis de Estresse | Ausências de Pedaços | Acumulado de Fibonacci |
|---|---|---|
| Baixo | $0 \dashv 8$ | $0 \dashv 54$ |
| Médio | $8 \dashv 16$ | $54 \dashv 2.583$ |
| Alto | $16 \dashv 24$ | $2.583 \dashv 121.392$ |

Nosso objetivo é estabelecer uma relação entre o histórico de perda de pedaços e a QoE. Para tal, estabelecemos pontuações distintas para reproduções bem sucedidas e ausências de pedaços. Vamos chamar de "nível de estresse" o valor obtido, com esse processo de pontuação, em determinado instante.

A $A^2Q$ foi inspirada no trabalho de Elliott. Com o propósito de quantificar a psicologia humana associada às oscilações dos preços, ele catalogou diversos padrões gráficos criando regras específicas, originando assim o Princípio das Ondas de Elliott ou, simplesmente, Teoria de Elliott. A identificação dos padrões gráficos permitiu encontrar as "formas" existentes no mercado de capitais. Então, Elliott usou a Proporção Áurea ou Número de Ouro (aproximadamente 1,618) que foi extraída da sequência de números de Fibonacci, na caracterização das "formas" encontradas [Elliott 1938].

A Sequência Fibonacci é uma sucessão de números que aparece em muitos fenômenos da natureza. Ela 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 e assim por diante. Por sua inerente simplicidade, resolvemos estabelecer uma relação entre a Sequência Fibonacci e a ausência de pedaços implementando duas sequências distintas. Uma sequência é responsável por aumentar o "estresse" a cada ausência de pedaço, enquanto a outra reduz o "estresse" a cada pedaço reproduzido com sucesso.

Para entender melhor, veja o exemplo hipotético da Figura 1a. 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 e, ao constatarmos a presença do mesmo, os ponteiros se movem no sentido da SFI e é subtraído no nível de estresse o valor nela marcado, ou seja, 1. No momento seguinte, verificamos se o pedaço a reproduzir está presente e ao constatarmos sua ausência, os ponteiros se movem no sentido da SFD e é acrescentado ao nível de estresse o valor marcado nela, ou seja, 3.

Na Figura 1b podemos ver o exemplo de um gráfico que mostra uma série de 10 ausências, seguida por uma série de 10 reproduções. É possível notar valores pro-

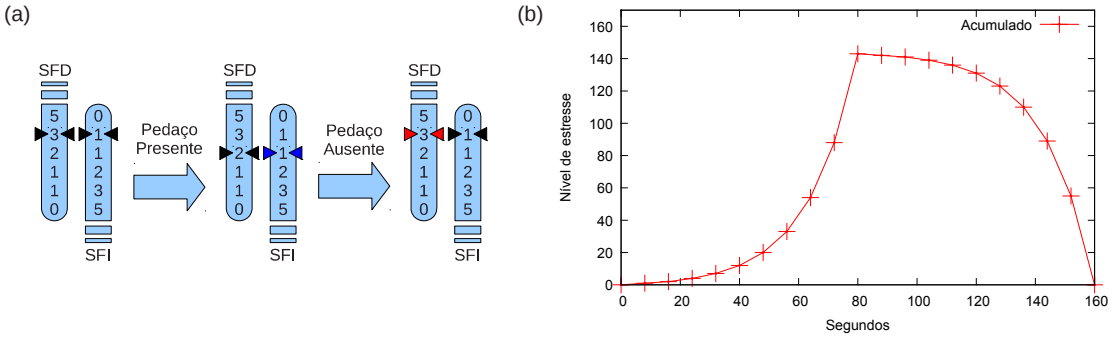


**Figura 1. Elementos básicos da avaliação de qualidade: (a) as duas Sequências Fibonacci e ilustração da variação de SFD e SFI em função do tempo, e (b) variação do nível de estresse em função do tempo**

gressivamente maiores quando os erros estão próximos. Note que, partindo do princípio de que pessoas muito estressadas não se acalmam imediatamente, os valores caem lentamente quando a constatação das ausências ainda está recente. Vale reparar que a forma do gráfico lembra uma exponencial. Na avaliação da percepção do usuário [Mwela and Adebomi 2010] observou que o MOS parece seguir uma exponencial, conforme o percentual das perdas de pacotes aumenta.

Com a implementação do algoritmo (S4Q), 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, 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. Escolhemos o AnySee por ele exigir apenas 40 segundos, enquanto que outros, como o CoolStreaming, necessitam de até 120 segundos [Liao et al. 2006].

Após o tempo de recebimento dos primeiros dados e *buffering* do vídeo, o algoritmo começa a verificar, a cada 8 segundos, a presença do pedaço necessário à reprodução segundo dois critérios: corte e pausa. Estes dois critérios representam os cenários encontrados 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). Ele também começa a guardar, a cada 8 segundos, a lista atual de pares. Após a décima lista é feita uma consolidação, onde apenas os pares que aparecem em todas as listas farão parte de uma única lista (neste trabalho esse resultado recebe o nome de Lista de Pares Estáveis - LPE). Utilizamos 10 listas porque uma quantidade menor poderia não ser representativa e uma maior faria com que o algoritmo demorasse para atuar. A determinação da quantidade ideal é assunto para trabalhos futuros.

A formação da LPE é um processo cíclico que se mantém durante toda a transmissão. Ao gerar uma LPE, o algoritmo verifica o Nível de Estresse por Corte (NEC) e/ou o Nível de Estresse por Pausa (NEP). Caso ambos estejam baixos, (ver Tabela 2) 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 2a vemos um exemplo em que o par **D** solicita os

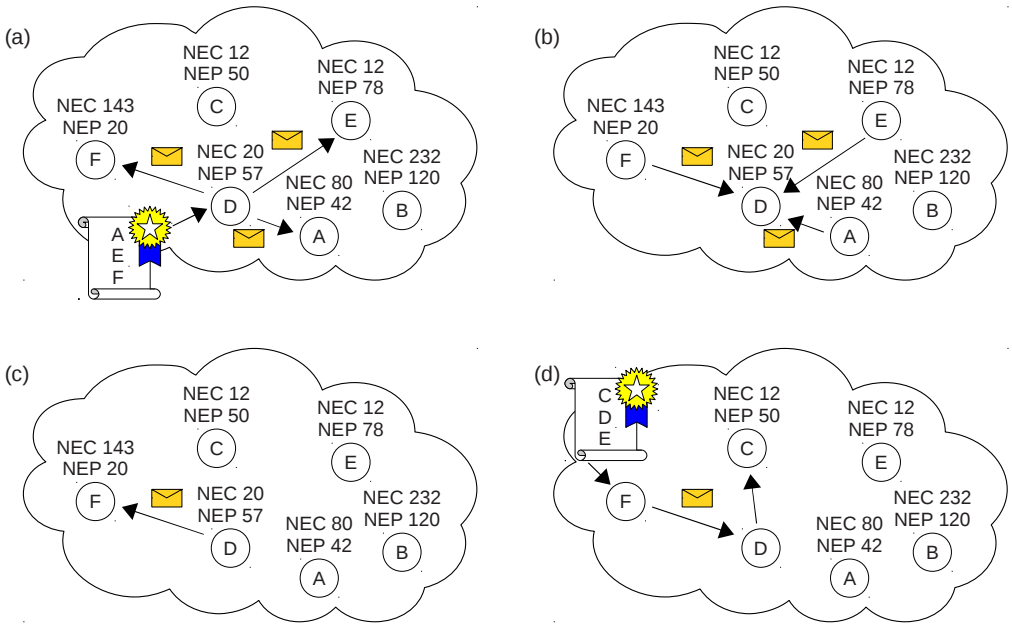


**Figura 2. O funcionamento do algoritmo para seleção de pares**

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 2.

Depois de receber as respostas, o par **D** 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 2b vemos a continuação do nosso exemplo, onde os pares **A**, **E** e **F** respondem à solicitação do par **D** com seus níveis de estresse (**A** com NEP 42, **E** com NEP 78 e **F** com NEP 20). Na Figura 2c, 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 (Figura 2d), 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**.

O recebimento de uma nova LPE não provoca a substituição da atual, apenas provoca o acréscimo de pares. A manutenção dos pares segue por conta da política tradicional do BitTorrent. A LPE recebida pode conter pares fora do perímetro de qualidade mínima para comunicação. Por exemplo, um par **A** pode ter uma boa comunicação com um par **B** e indicá-lo ao par **C**, por meio de sua LPE. Porém, o par **C** pode não conseguir uma boa comunicação com **B** devido a distância geográfica, questões de infra estrutura ou por motivos diversos.

Dessa forma, com esta proposta, esperamos: (i) reduzir o tempo de *download*, (ii) reduzir as interrupções na reprodução do vídeo durante sua transmissão, (iii) promover a chegada sequencial dos pedaços sem interferir no ASAP e sem provocar os problemas do ASSP, (iv) produzir uma melhor e mais homogênea QoE entre os pares, (v) promover

agrupamentos sem uso da taxa de *upload*, posição geográfica ou informações de fontes externas, (vi) acelerar a formação de *supernodes* (pares colaborativos com grande capacidade de *upload*) e (vii) 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). Uma reprodução do vídeo com menos interrupções, durante sua transmissão, pode ser obtida porque os valores de NEC e NEP têm relação com a chegada do pedaço antes da necessidade de sua reprodução pelo *player* de vídeo. Quem tem um baixo NEC/NEP está recebendo 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, reduzindo tanto interrupções no ASAP quanto no ASSP. 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.

A solução promove agrupamentos sem uso de (a) taxa de *upload*, que pode apresentar problemas com *jitter*, (b) distância em saltos que exige maior trafego de controle ou (c) informações de fontes externas que deixam o algoritmo dependente da boa performace e disponibilidade do serviço consumido. O agrupamento ocorre pela troca das LPEs, pois os pares tendem a ter uma boa comunicação com os integrantes das LPEs 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 as exceções com um *choke*, isso significa que, um par BitTorrent pode "afogar" seu vizinho, no momento que este não coopera com ele, interrompendo todos os *uploads* ao mesmo.

Acelerar a formação de *supernodes* 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 de que ocorra uma convergência a um pequeno grupo de pares, podendo ocasionar sobrecarga, clusterização ou que as LPEs causem um sincronismo. Por exemplo, quando um par fica saturado e há excessiva concentração, o NEC/NEP podem aumentar e, hipoteticamente, provocar uma migração para um mesmo novo par (sincronismo). Porém, este caso não foi observado durante os experimentos.

Finalmente, nota-se que o efeito oposto do que ocorre com os *supernodes* ocorrerá com os *free riders*, já que eles não conseguem montar uma LPE e não serão selecionados pelos outros pares para compor uma LPE. Logo, os *free riders* não tiram proveito dos benefícios do algoritmo.

## 4. Resultados Experimentais

Nesta seção, apresentamos os resultados experimentais e análises decorrentes dos mesmos. Nossos objetivos são (a) mostrar que o S4Q consegue uma transmissão com poucas ausências de pedaços, (b) que nosso algoritmo obtem bons resultados mesmo utilizando o ASAP e (c) que ele reduz rapidamente os níveis de estresse, decorrentes do atraso na inicialização, proporcionando uma melhorar na QoE geral do sistema. Por uma questão de síntese vamos apresentar com maiores detalhes apenas os dados provenientes dos experimentos com ASAP e com o critério pausa.

### 4.1. Ambiente de Experimentação

Os experimentos foram realizados utilizando o ambiente do Planetlab, onde diversas máquinas foram pré-selecionadas de acordo com as suas respectivas disponibilidades e características e escolhidas a cada rodada do experimento (detalhes em [de Sousa 2013]). Cada cenário foi repetido 30 vezes, eles emulavam uma distribuição de vídeo com o objetivo de comparar a QoE de cada solução para cada algoritmo de seleção de pedaços e quantidades diferentes de *leechers* no *swarm*.

Nosso experimento foi executado em grupos de 25, 50, 75, 100 e 125 *peers*, com um *tracker* e um *seeder*. Os dados estatíticos foram extraídos do arquivo "Azureus_stats.xml', que é gerado, por padrão, a cada 30 segundos, quando as estatísticas estão ativadas. Desse xml coletamos o *downloaded* e o *uploaded* de cada *peer* com o respectivo IP. O registro dos dados aconteceu durante 2.000 segundos. Também foi coletado diretamente do sistema operacional os níveis de consumo da memória em cada par.

Durante as rodadas recebemos vários alertas do PlanetLab sobre o consumo excessivo de memória e em análise verificamos que se tratava do Ono. Essa solução apresentou um grande consumo médio de memória física não *swapped* (*Resident Set Size - RSS*) durante a execução de suas tarefas.

### 4.2. Avaliação dos Resultados Experimentais

A execução do experimento levou mais de 3 meses e desde o início nos preocupamos em garantir uma igualdade de condições entre as soluções, por isso algumas informações foram colhidas e verificadas a fim de identificar possíveis distorções. Não julgamos suficiente apenas selecionar aleatoriamente as máquinas que iriam compor cada grupo em cada rodada, queriamos nos certificar de que nada fora do normal alterasse os dados e beneficiasse uma solução em detrimento das outras, por isso analisamos e classificamos 4 tipos de situações: (1) máquinas em que não foi possível obter os *traces* (dados) foram consideradas "desligadas"; (2) que geraram *traces*, mas não estabeleceram conexão com outras máquinas foram classificadas como "*download* zero'; (3) que não completaram o *download* como "*download* parcial" e, por último, (4) as demais como "*download* completo". Os dois primeiros casos (desligadas e *download* zero) têm relação direta com igualdade de condições em que cada solução foi submetida e esses valores, entre as soluções, foram muito próximos em todos os casos. Os dois últimos tiveram maiores variações, mas julgamos que essas diferenças são provenientes da estratégia usada por cada algoritmo.

As variações apontadas para o caso *download* completo foram estudadas por meio do intervalo de confiança (IC), com nível de confiança em 95%, para o tempo médio de *download*. Com ASAP o IC indica um empate técnico entre as soluções para pequenos *swarms*, ao aumentar o número de *leechers* o IC cai para aproximadamente 2% em todas as soluções. Também avaliamos o *download* parcial junto com o *download* completo, por meio do IC, para o tempo médio de início do *download*. Neste caso o uso do ASAP gerou o mesmo efeito já relatado, chegando a 11,52% (Ono com 25 pares) e reduzindo com o aumento no número de *leechers*. Resolvemos não mostrar os ICs nos gráficos para maior clareza na leitura.

O empate a pouco citado pode ter relação com o comportamento do protocolo. No BitTorrent, por padrão, os *leechers* solicitam um lista de pares ao *tracker* a cada 30

minutos e a responda pode conter até 50. Isso faz com que, em um *swarm* com até 50 *peers*, um algoritmo para seleção de pares não tenha muito o que decidir.

Na avaliação do *upload* a partir do *seeder* com ASAP constatamos uma variação entre 5% e 10% da transmissão total do sistema e com ASSP uma oscilação de 15% a 20%, em ambos os casos os valores tendem a 2% conforme o número de pares aumenta. Logo, podemos concluir que o *seeder* é mais requisitado em pequenos grupos e mais ainda quando utilizamos o ASSP. Esse esforço extra do *seeder*, com ASSP, pode prejudicar o sistema como um todo. Foi possível notar que o ASSP aumenta a latência reduzindo a velocidade do *download*, mas também que ele proporciona uma melhor QoE devido a chegada sequencial dos pedaços, que, por sua vez, mantém os níveis de NEC/NEP mais baixos se o compararmos com o ASAP.

Sobre os dados que têm relação direta com a QoE, podemos verificar que o S4Q tende a ter menos ausências de pedaços. Repare nas Figuras 3 e 4, nossa proposta conseguiu se destacar (ficando em primeiro ou segundo lugar) em quase todos os grupos, exceto no grupo com 100 pares por uma diferença mínima de 0,2 pedaços. O Ono também conseguiu uma boa colocação, mas note que os outros valores indicam grandes oscilações, enquanto que o S4Q é mais constante em seus resultados.

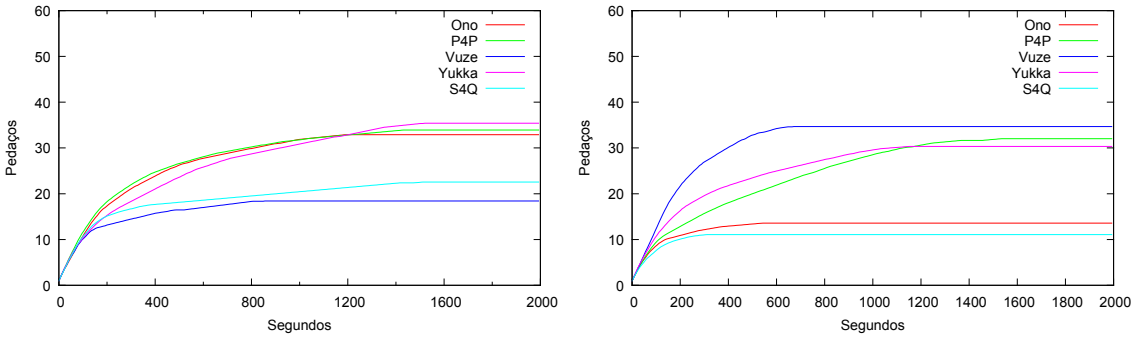


**Figura 3. Média amostral da ausências de pedaços no critério pausa com ASAP ao longo do experimento com 50 (esquerda) e 75 pares (direita)**

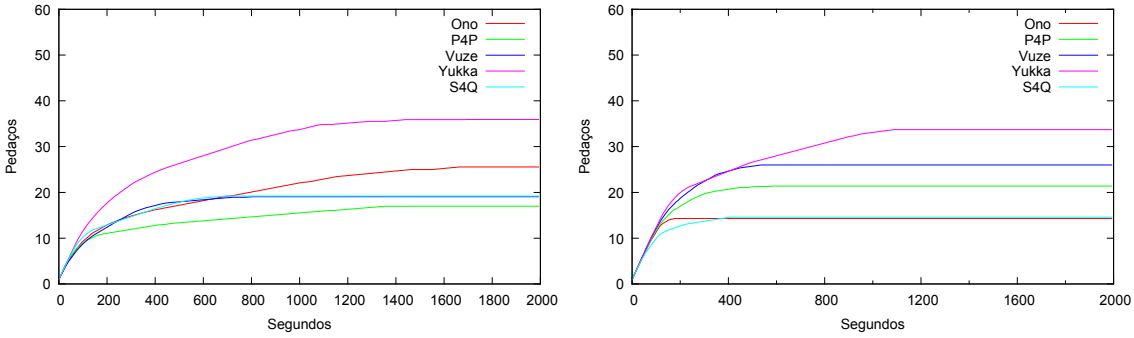


**Figura 4. Média amostral da ausências de pedaços no critério pausa com ASAP ao longo do experimento com 100 (esquerda) e 125 pares (direita)**

Nas Figuras 5 e 6 podemos ver que o tempo que o S4Q leva para reduzir o nível de estresse. Elas mostram que o aumento no número de pares faz com que o S4Q melhore a QoE cada vez mais rápido e que as outras soluções oscilam muito em seus resultados.

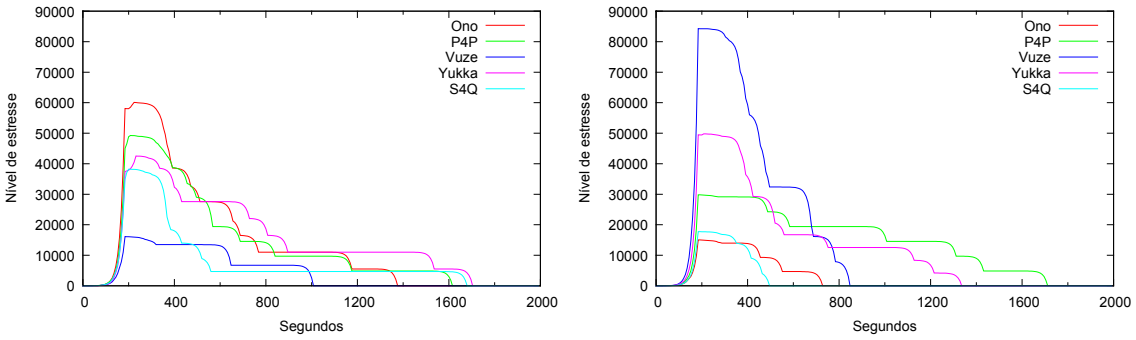


**Figura 5. Média amostral do nível de estresse no critério pausa com ASAP ao longo do experimento com 50 (esquerda) e 75 pares (direita)**

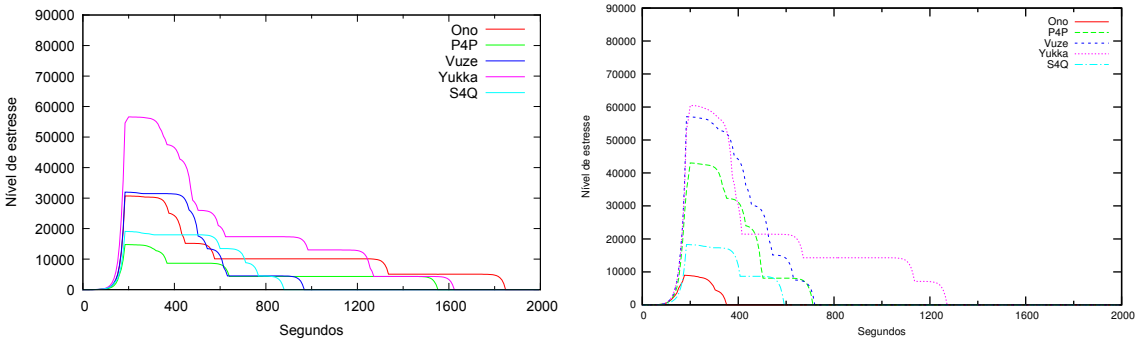


**Figura 6. Média amostral do nível de estresse no critério pausa com ASAP ao longo do experimento com 100 (esquerda) e 125 pares (direita)**

Na avaliação dos dados produzidos pela $A^2Q$, podemos ver que, em todos os casos (observe as Figuras 5 e 6), há uma aumento acentuado do nível de estresse nos primeiros 200 segundos (mesma faixa de tempo escolhida para entrada dos *leechers* no *swarm*). Isso pode ser explicado observando na Figura 7, nela podemos ver que o tempo para início do *download* varia de 50 à 150 segundos. Repare que até o menor valor ficou acima de 40 segundos, ou seja, após o tempo de espera definido para início das verificações de pedaços e constituição da nova métrica. 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, entre os *leechers* bem sucedidos. Ao avaliar a duração das ausências podemos notar que o S4Q consegue reduzir esse tempo em relação as demais soluções, veja na Figuras 8 a distribuição da duração das ausências em escala logaritmica.

Analisando os dados verificamos que o S4Q ainda tem chances de apresentar um resultado melhor. Ao avaliar a velocidade alcançada pelos *peers*, notamos que o experimento teve uma grande concentração de máquinas com velocidade de 1 MByte/s, com uma pequena redução no grupo com 125 pares. As máquinas que atingem 1Mbyte/s conseguem baixar um vídeo de 127 MBytes em apenas 127 segundos, não dando "tempo de reação" ao S4Q, uma vez que a LPE é formada a cada 80 segundos. 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 uma maior troca de LPEs. Logo, isso poderá fazer com que o S4Q alcance resultados melhores.

No geral as soluções apresentaram um tempo de *download* maior e um nível de estresse menor quanto utilizando o ASSP, porém foi possível observar que o S4Q consegue níveis de estresse próximos utilizando o ASAP. Isso ocorre em função da indução a chegada sequencial de pedaços.

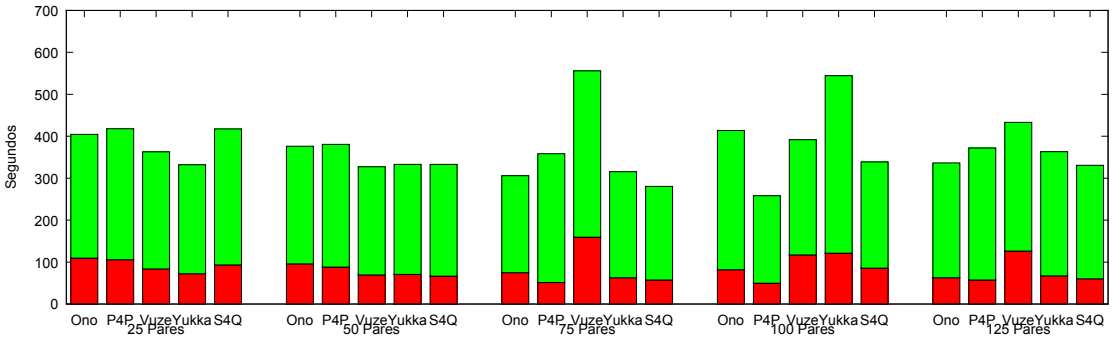


**Figura 7. Tempo de início (barra inferior) e *download* (barra superior) com ASAP**

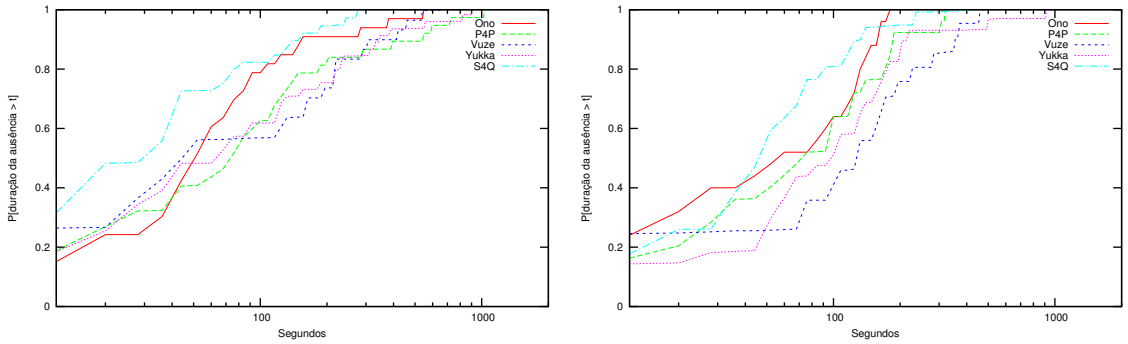


**Figura 8. Função de Distribuição Cumulativa da duração das ausências de pedaços no critério pausa com ASAP ao longo do experimento com 75 (esquerda) e 125 pares (direita)**

## 5. Conclusão

Este trabalho apresentou uma nova métrica em QoE, ainda não validada pela percepção de usuários reais por meio de MOS, para transmissão de vídeo, partindo da verificação de pedaços e utilizando a Sequência Fibonacci sobre dois critérios: corte e pausa. Também foi apresentado um novo algoritmo para seleção de vizinhos que atribui a uma lista de pares estáveis o valor obtido com a nova métrica. Dessa forma, as melhores listas são trocadas entre os pares, com o propósito de promover agrupamentos e acelerar a formação de *supernodes*, para com isso obter um menor tempo de *download* e uma reprodução de vídeo com menos interrupções, durante sua transmissão pela rede.

A partir dos resultados obtidos podemos concluir que (1) é possível tratar a seleção de pares com algoritmos simples, eficazes e que não dependem de informações contidas em fontes externas, uma vez que o S4Q chega a ter, pelo menos, 19% menos ausências de pedaços de vídeo no momento da reprodução que outros algoritmos do estado da arte e uma redução no "nível de estresse" 32% mais rápida; (2) podemos realizar agrupamentos sem estabelecer limitações, como taxa de *upload* ou localização geográfica, pois nossa

solução realiza essa tarefa com a troca das LPEs; (3) não precisamos nos preocupar em identificar os *supernodes* para aproveitar seus recursos, por serem bons fornecedores eles acabam fazendo parte das LPEs; (4) é viável construir soluções que são orientadas pela QoE, sem utilizar recursos complexos, pois conseguimos produzir avaliações dinâmicas dos NEC/NEP e utilizá-los na seleção de pares.

## Referências


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.

Bertinat, M. E., Vera, D. D., Padula, D., Amoza, F. R., Rodríguez-Bocca, P., Romero, P., and Rubino, G. (2009). *GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network*. IEEE LANC.

Choffnes, D. R. and Bustamante, F. E. (2008). *Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems*. ACM SIGCOMM.

de Sousa, Peron Rezende (2013). *Seleção de Pares Baseada em QoE para Transmissão de Vídeo em Redes P2P BitTorrent*. CCET - UNIRIO. Dissertação de M.Sc.

Elliott, R. N. (1938). *The Wave Principle*. Republicado (2012), Editora Snowball Publishing, New York, NY, USA.

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.

Krishnan, S. and Sitaraman, R. (2012). *Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs*. ACM IMC.

Liao, X., Jin, H., Liu, Y., Ni, L. M., and Deng, D. (2006). *AnySee: Peer-to-Peer Live Streaming*. IEEE INFOCOM.

Moraes, I. M., Campista, M. E. M., Moreira, M. D. D., Rubinstein, M. G., Costa, L. H. M. K., and Duarte, O. C. M. B. (2008). *Distribuição de Video sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios*. Minicursos do XXVI SBRC.

Mwela, J. S. and Adebomi, O. E. (2010). *Impact of Packet Loss on the Quality of Video Stream Transmission*. School of Computing at Blekinge Institute of Technology. Dissertação de M.Sc.

Polaczyk, B. and Cholda, P. (2010). *BitTorrent Traffic Localization via Operator-related Information*. IEEE ICC.

Rodríguez-Bocca, P. (2008). *Quality-Centric Design of Peer-to-Peer Systems for Live-Video Broadcasting*. l'Université de Rennes. Tese de D.Sc.

Rosário, D.; C., R.; S., A.; B., T. and C., E. (2013). *QoE-aware Multiple Path Video Transmission for Wireless Multimedia Sensor Networks*. XXXI SBRC, 2013.

Wang, Y. (2006). *Survey of Objective Video Quality Measurements*. EMC Corporation Hopkinton.

Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y., and Silberschatz, A. (2008). *P4P: Provider Portal for Applications*. ACM SIGCOMM.