Universidade Federal de Juiz de Fora
Programa de Pós-Graduação em Ciência da Computação
Mestrado em Ciência da Computação

Rafael Barra de Almeida

ANÁLISE E CONTRA-ATAQUE À POLUIÇÃO E WHITEWASHING EM SISTEMAS P2P DE VÍDEO AO VIVO

Juiz de Fora
2013

Rafael Barra de Almeida

Análise e Contra-Ataque à Poluição e Whitewashing em Sistemas P2P de Vídeo ao Vivo

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. Alex Borges Vieira

Coorientadora: Profa. D.Sc. Ana Paula Couto da Silva

Juiz de Fora

2013

Almeida, Rafael Barra de.
Análise e contra-ataque à poluição e whitewashing em sistemas P2P de vídeo ao vivo / Rafael Barra de Almeida. – 2013.
79 f. : il.

Dissertação (Mestrado em Ciência da Computação)–Universidade Federal de Juiz de Fora, Juiz de Fora, 2013.

1. Ciência da computação. 2. Redes de computadores. I. Título.

CDU 681.3

Rafael Barra de Almeida

Análise e Contra-Ataque à Poluição e Whitewashing em Sistemas P2P de Vídeo ao Vivo

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 25 de Janeiro de 2013.

BANCA EXAMINADORA

Prof. D.Sc. Alex Borges Vieira - Orientador
Universidade Federal de Juiz de Fora

Profa. D.Sc. Ana Paula Couto da Silva
Universidade Federal de Juiz de Fora

Prof. Phd. Daniel Sadoc Menasché
Universidade Federal do Rio de Janeiro

Prof. D.Sc. Antonio Augusto de Aragão Rocha
Universidade Federal Fluminense

A Deus em primeiro lugar. Aos meus pais, namorada e amigos pelo apoio incondicional.

AGRADECIMENTOS

Primeiramente a Deus por me permitir chegar até aqui e me dar força e saúde.

Aos meus familiares, principalmente meus pais, Antonio e Maria, e irmãos, pelo apoio incondicional em todos os sentidos. Muito obrigado por tudo que me ensinaram e por serem sempre a base firme de tudo na minha vida.

A minha namorada Carla, pela compreensão e apoio durante a realização desse trabalho. Por ter paciência nos momentos em que o trabalho não permitiu estarmos juntos. Obrigado por tornar tudo mais fácil para mim.

Aos meus amigos pela amizade e companheirismo que ajudaram a passar por mais essa fase da vida. Aos meus colegas do grupo de redes pela ajuda no desenvolvimento desse trabalho e pelos momentos de distração que amenizaram a dificuldade do trabalho.

Aos meus orientadores, Alex e Ana pela oportunidade, confiança, dedicação e paciência.

Agradeço também à Capes, CNPq e FAPEMIG pelo auxílio financeiro e pelos equipamentos disponibilizados.

"Nada no mundo consegue tomar o lugar da persistência. O talento não consegue; a genialidade não consegue; a educação não consegue. A persistência e determinação sozinhas são onipotentes"

Calvin Coolidge

RESUMO

Aplicações de transmissão de vídeo ao vivo na Internet têm ganhado bastante popularidade nos últimos anos. A facilidade para se publicar e ter acesso a esse tipo de conteúdo tem atraído grande atenção. A arquitetura P2P ganhou lugar de destaque neste contexto, principalmente, por ser mais tolerante a falhas e superar os problemas de escalabilidade presentes no modelo cliente-servidor. No entanto, devido às suas características, como ausência de controle centralizado, as redes P2P podem ser suscetíveis a ataques e comportamentos maliciosos.

Este trabalho analisa o impacto causado por ataques de poluição e *whitewashing* em sistemas P2P de transmissão de vídeo ao vivo. Para combater estes tipos de ataques, mecanismos simples e descentralizados de reputação são propostos e implementados através de um protótipo de aplicação executado em um ambiente de rede real, configurado no PlanetLab.

Os resultados mostram que ataques de poluição são bastante prejudiciais a sistemas P2P de transmissão de vídeo ao vivo. Quando apenas 10% dos *peers* agem de maneira maliciosa, os demais participantes do sistema recebem mais de 90% de dados poluídos. Além disso, a sobrecarga média na banda de rede chega a 230% em momentos de pico, forçando os participantes a utilizar 3 vezes mais banda de rede do que seria necessário em um sistema sem ataques de poluição. Os mecanismos de reputação propostos, testados no ambiente do PlanetLab, bloqueiam ataques de poluição rapidamente, reduzindo a porcentagem de dados poluídos recebidos a 6% e a sobrecarga a 5%. Para o caso de ataque combinado de poluição e *whitewashing*, o mecanismo de reputação proposto diminui a sobrecarga no sistema de 112% para 20% e a porcentagem de dados poluídos de 70% para 19%.

Palavras-chave: P2P. Segurança. Ataques de Poluição. *Whitewashing*.

ABSTRACT

Live streaming video applications on the Internet are becoming more popular in the last years. The ease of publishing and accessing content through such applications has attracted great attention from users and researchers. The P2P architecture has obtained a prominent place in this context, mainly because it is more fault tolerant and overcome the scalability issues present in the client-server model. However, due to its distributed characteristics, the P2P networks can be more susceptible to attacks and malicious behavior.

In this work, we analyze the impact of pollution attacks and whitewashing in P2P live streaming systems. To combat these attacks, simple mechanisms and decentralized reputation are proposed and implemented by a prototype application running on a real network environment, set in PlanetLab.

Our results show that pollution attacks are harmful to P2P live streaming systems. When 10% of peers are malicious, the remaining peers in the system receive more than 90% of data polluted. Furthermore, the average overhead in network bandwidth reaches 230% at peak times, forcing the peers to use three times the bandwidth required in a system without pollution attacks. The proposed reputation mechanism, tested on PlanetLab testbed, quickly blocks isolated pollution attacks, reducing the amount of polluted data received to 6% and the overhead to 5%. In the case of pollution attack combined with whitewashing, the proposed reputation mechanism reduces the overhead in the system from 112% to 20% and the amount of polluted data from 70% to 19%.

Keywords: P2P. Security. Pollution Attacks. *Whitewashing.*

SUMÁRIO

1 Introdução

Nos últimos anos, aplicações de vídeo ao vivo na Internet tem atraído a atenção de muitos usuários e pesquisadores na área de redes. Devido a quantidade de tráfego gerado por esse tipo de aplicação, os diversos sistemas existentes e a facilidade para se publicar conteúdo de áudio e vídeo na Internet, esse assunto tem ganhado grande destaque. Uma característica que torna esse tipo de aplicação cada vez mais popular é a possibilidade de as utilizar para transmitir conteúdo tal como uma TV, porém com uma maior flexibilidade, permitindo às "emissoras" gerar conteúdo personalizado para cada tipo de público.

As primeiras versões destas aplicações eram baseadas na arquitetura cliente-servidor. Atualmente, estas aplicações utilizam a arquitetura P2P (*peer-to-peer*) que tende a ser escalável e mais resiliente a falhas. Por não se basear em uma infraestrutura dedicada, a arquitetura P2P oferece uma rápida implementação a um baixo custo, sem requerer altos recursos centralizados [Hei, 2008a]. Ao se utilizar a arquitetura P2P, a capacidade de *upload* de cada participante pode ser utilizada para auxiliar na disseminação do conteúdo de vídeo por toda a rede. Desse modo, a largura de banda, que seria necessária em um único ponto no modelo cliente-servidor, é dividida entre vários participantes do sistema, reduzindo consideravelmente a carga no servidor origem da transmissão.

Como exemplo da importância destas aplicações, a CNN utilizou uma plataforma P2P para auxiliar na distribuição do conteúdo da posse do presidente Obama em janeiro de 2009. Este evento, considerado um dos maiores na história da Internet, atendeu a 1.3 milhões de usuários ao mesmo tempo. Mais da metade dos usuários utilizavam a estrutura P2P para acompanhar a transmissão do evento [1].

Existem duas formas principais de se construir a rede sobreposta (*overlay*) P2P: redes organizadas em árvore e em malha. Na primeira, os *peers* se organizam em forma de árvore, onde o conteúdo de vídeo é empurrado da origem (nó pai) para os demais *peers* (nós filhos) em uma abordagem conhecida como *tree-push*. Na segunda, os *peers* se organizam em forma de malha, e cada *peer* requisita explicitamente o dado que deseja de seus parceiros. Esta abordagem é chamada *mesh-pull*. Vários sistemas baseados na abordagem *tree-push* foram propostos [Castro, 2003, Padmanabhan, 2002], porém, esta abordagem

[1] http:www.nytimes.com/external/gigaom/2009/02/07/07gigaom-cnn-inauguration-p2p-stream-a-success-despite-bac-17849.html

não obteve sucesso comercialmente [Hei, 2008a]. Atualmente, as aplicações mais populares de transmissão de vídeo ao vivo em P2P organizam seus *peers* em uma rede sobreposta em forma de malha [Hei, 2007], como por exemplo CoolStreaming [2], PPLive [3] e o SopCast [4].

Em sistemas P2P de transmissão de vídeo ao vivo, um *peer* especial codifica o vídeo e o divide em pequenos pedaços denominados *chunks*. Os demais requisitam (ou trocam) esses pedaços de informação para (entre) seus parceiros. Um novo participante, ao entrar no sistema, estabelece conexões com um grupo aleatório de participantes que estejam assistindo ao mesmo conteúdo de vídeo desejado. Em curtos intervalos de tempo, os participantes anunciam os *chunks* de vídeo que possuem e que necessitam para seus parceiros. A partir desse momento é possível realizar ou receber pedidos por dados.

Em um cenário ideal, espera-se que os participantes desse sistema assumam um comportamento altruísta e não malicioso, ou seja, compartilhem dados de maneira proporcional aos que recebem e não realizem nenhum tipo de ataque contra a estrutura do sistema. Entretanto, em um sistema real podem existir vários usuários que assumem um comportamento malicioso e oportunista, se aproveitando das características do sistema para provocar algum tipo de dano ao mesmo.

Um dos ataques realizados em sistemas P2P de transmissão de vídeo ao vivo é a poluição de conteúdo [Vieira, 2008]. Nesse ataque, participantes maliciosos alteram ou danificam o conteúdo do dado compartilhado antes de encaminhá-lo a seus parceiros. Esse ataque pode criar um falso fluxo de vídeo ou danificar o conteúdo original, tornando-o inútil para os demais. Caso os participantes que recebem esses dados não percebam que estes estão poluídos, eles podem repassá-los, consumindo recursos de rede e processamento com dados indesejados, aumentando o problema da poluição. Com isso, o conteúdo legítimo fica menos disponível, comprometendo o funcionamento do sistema. Este comportamento pode ter maior impacto quando os participantes maliciosos realizam um ataque combinado, onde participantes se juntam a outros para causar um dano maior ao sistema. A esse ataque combinado é dado o nome de conluio.

Entre os diversos ataques e comportamentos indesejados pode-se citar o ***Free-Riding***, quando um participante malicioso recebe dados de seus parceiros mas não os envia para os outros participantes, não contribuindo com o sistema [Karakaya, 2009]. Outro ataque

[2] www.coolstreaming.us
[3] www.pplive.com
[4] www.sopcast.org

é o ataque de **Descarte de Dados**, que ocorre quando um participante atrai alguns parceiros prometendo servi-los mas, quando um parceiro requisita algum dado, o atacante o nega [Seibert, 2010]. Por fim, a prática de ***Whitewashing***, quando um participante sai e entra no sistema, repetidamente, com uma nova identidade visando evitar penalidades devido a seu mau comportamento [Seibert, 2010].

Os ataques e comportamentos maliciosos listados anteriormente obrigam os *peers* a pedir novamente o dado de vídeo que não foi recebido com sucesso, gerando sobrecarga no sistema. Como consequência os participantes do sistema poderão sofrer com perdas de *chunks* e atraso na exibição do conteúdo de vídeo. Assim, a qualidade da exibição do vídeo nos usuários fica comprometida.

Em aplicações de compartilhamento de arquivo, como o BitTorrent [Cohen, 2003], existe uma série de trabalhos que tratam de ataques e comportamentos oportunistas [Walsh, 2005, Damiani, 2002, Costa, 2007a, Costa, 2007b]. Porém, mesmo havendo semelhanças entre aplicações de compartilhamento de arquivos e aplicações de transmissão de vídeo ao vivo, as soluções dadas para um sistema podem não funcionar em sistemas ao vivo [Coelho, 2010]. Isso porque, enquanto as aplicações de compartilhamento de arquivos se preocupam apenas em distribuir um arquivo completo, as aplicações de transmissão de vídeo ao vivo necessitam distribuir o vídeo em um curto intervalo de tempo. Ou seja, o tempo é um fator crítico nesse tipo de aplicação [Lin, 2010]. Por esse motivo, o desenvolvimento de medidas de combate ao comportamento malicioso e/ou oportunista em sistemas P2P de transmissão de vídeo ao vivo é de grande importância.

Na literatura são encontrados vários esquemas de combate a ataques específicos a sistemas P2P de vídeo ao vivo [Dhungel, 2007, So, 2012, Vieira, 2008, Vieira, 2009]. No entanto, essas soluções podem falhar em situações onde participantes maliciosos mudam frequentemente suas identidades, ou seja, em cenários com *whitewashing*. A facilidade de se obter uma nova identidade e a dificuldade de caracterizar um *whitewasher* fazem com que este comportamento se torne um desafio [Feldman, 2006].

Os trabalhos [Yu, 2011, Seibert, 2010, Oualha, 2009, Chen, 2009] abordam a prática de *whitewashing* em sistemas P2P. Em [Yu, 2011] os autores propõem um esquema baseado na opinião sobre o comportamento de um nó, denominada reputação, que desencoraja a prática de *whitewashing* enquanto incentiva bons comportamentos em sistemas P2P de compartilhamento de arquivos. Em [Seibert, 2010] o problema de whitewashing é tratado

em sistema de transmissão de vídeo. Os autores propõem um mecanismo baseado em reputação para combater ataques de descarte de dados. Ao marcar os *peers* recém-chegados como suspeitos e reduzindo os recursos disponíveis para estes *peers*, os autores afirmam que esse mecanismo também pode combater o ataque de *whitewashing*. Em [Puttaswamy, 2009] é proposta uma entidade centralizada responsável por identificar unicamente participantes recém chegados ao sistema de armazenamento. A desvantagem desta abordagem é a centralização de informações em um sistema de natureza distribuída.

Feldman *et al.* [Feldman, 2006] mostram que abordagens que impõe penalidades aos nós recém-chegados afetam a escalabilidade do sistema. Segundo Feldman *et al.*, como *whitewashing* é um comportamento difícil de ser observado, a penalidade será aplicada a todos os participantes recém-chegados (bons ou maus *peers*). Além disso, o desempenho do sistema também seria reduzido caso aconteça alta dinamicidade dos participantes. Os autores de [Chen, 2009] apresentam um modelo que captura as diferenças de comportamento dos *peers whitewashers* e dos *peers* de comportamento colaborativo. Os autores afirmam que cada *peer* deve manter uma sequência de ações similares mesmo quando saem e entram novamente no sistema (visitar os mesmos nós, requisitar os mesmos serviços de Internet ou tomar ações similares). Assim, estes *peers* poderiam ser identificados e bloqueados.

Dado esse contexto, o objetivo principal desse trabalho é analisar o impacto causado por estes ataques em sistemas P2P de transmissão de vídeo ao vivo, além de propor um mecanismo baseado em reputação para minimizar os efeitos de tal prática. O esquema proposto nesse trabalho para combater *whitewashing* se diferencia dos citados anteriormente por se basear em um mecanismo de reputação simplificado, onde cada participante avalia seus parceiros individualmente, sem considerar o depoimento do restante da rede e sem punir os participantes recém-chegados. Também é implementado um mecanismo que permite a recuperação de participantes que, por algum problema de rede, seja considerado um poluidor. Para realizar esse trabalho, foi desenvolvido um protótipo de aplicação de transmissão de vídeo ao vivo em P2P, capaz de simular ataques de poluição e *whitewashing*, assim como combater esses tipos de ataques.

O protótipo de aplicação desenvolvido foi executado em máquinas do PlanetLab [5] e os resultados experimentais mostram que identificar os dados poluídos e pedir retransmissão

[5] www.planet-lab.org

é ineficiente. Nesse caso, como os *peers* pedem retransmissão, a sobrecarga média na banda de rede chega a 230% em momentos de pico. Nesse cenário, os *peers* também experimentam uma alta taxa de perda no tempo de execução, o que indica que os usuários não assistem um vídeo com qualidade adequada. O primeiro mecanismo de reputação implementado nessa dissertação, proposto, originalmente, por Vieira *et al.* [Vieira, 2009], apresenta valores de sobrecarga abaixo de 5% quando os poluidores não realizam *whitewashing*. Porém, quando há ataque de *whitewashing*, a sobrecarga no sistema alcança 112% e a taxa de perda chega a 50%. Assim, nessa dissertação, foi proposto um segundo mecanismo de reputação que é capaz de reduzir a sobrecarga para cerca de 20% e a taxa de perda de pedaços de vídeo é baixa, com valores em momento de pico por volta de 3%. Finalmente, os resultados mostram que os *peers* com problemas temporários, identificados como poluidores, tem oportunidade de se redimir, e rapidamente, voltar a contribuir com o sistema P2P.

1.1 Contribuições

As principais contribuições dessa dissertação são:

- Desenvolvimento de um protótipo de aplicação de transmissão de vídeo ao vivo utilizando a arquitetura P2P. Esse protótipo é capaz de simular ataques de poluição isolados, assim como, combinados com *whitewashing*.

- Implementação e validação, em um ambiente de rede real, de um mecanismo de reputação simplificado, originalmente proposto e simulado por Vieira *et al.* [Vieira, 2009], capaz de combater ataques de poluição isolados.

- Modificação no mecanismo de reputação simplificado proposto em [Vieira, 2009] para torná-lo efetivo contra ataques de poluição combinado com *whitewashing*. Essa modificação também foi implementada como funcionalidade do protótipo desenvolvido.

- Amplo conjunto de testes em um ambiente real implementado no PlanetLab, com o objetivo de avaliar os mecanismos de reputação propostos. As métricas utilizadas nos testes permitem avaliar a utilização da rede, assim como a qualidade da exibição do vídeo percebida pelos usuários.

1.2 Organização da Dissertação

A dissertação está organizada da seguinte forma: o capítulo 2 apresenta conceitos teóricos importantes para um bom entendimento desse trabalho. O capítulo 3 descreve trabalhos que serviram como base de estudos para a implementação do sistema de reputação proposto nesse trabalho. No capítulo 4, são detalhados os mecanismos de reputação propostos e o protótipo de aplicação P2P para transmissão de vídeo ao vivo. O capítulo 5 apresenta os resultados obtidos através da execução do protótipo de aplicação nas máquinas do PlanetLab. Por fim, no capítulo 6 são apresentadas as conclusões desse trabalho.

2 Conceitos Teóricos

Nesse capítulo são apresentados conceitos importantes para a implementação de sistemas P2P para distribuição de vídeo. Assim, na seção 2.1 são dadas características gerais que definem um modelo básico de redes P2P. A seção 2.2 descreve brevemente a arquitetura das redes P2P. Nesta seção, os sistemas P2P são classificados de acordo com sua topologia e com a maneira como são criados e mantidos. A seção 2.3 apresenta exemplos de sistemas P2P populares. A seção 2.4 apresenta as principais topologias utilizadas para organizar a rede sobreposta (*overlay*) P2P. Essa seção apresentada também as principais estratégias para seleção de pedaços de vídeo. Estas estratégias influenciam diretamente no desempenho do sistema de transmissão de vídeo em P2P. Por fim, a seção 2.5 apresenta alguns dos mecanismos utilizados para combater ataques e comportamentos oportunistas em sistemas P2P.

2.1 Redes P2P

Redes P2P são redes descentralizadas, onde cada máquina, referida como *peer* ou nó, possui funcionalidades de cliente e servidor ao mesmo tempo [Liu, 2008]. Essas redes estão sobrepostas à rede física de computadores existente, que é, na maioria dos casos, uma rede TCP/IP. Nas redes P2P, diferente do modelo cliente-servidor, o conteúdo desejado não fica disponível somente em um nó central, mas distribuído entre todos os participantes do sistema. A arquitetura descentralizada evita um grave problema inerente à arquitetura cliente-servidor: a redução do desempenho do sistema conforme o aumento do número de usuários. Em redes P2P, o desempenho geral do sistema melhora à medida em que novos participantes se juntam ao sistema, dado que cada nó colabora compartilhando sua largura de banda com os demais participantes. Esse comportamento torna a rede escalável [Hei, 2008a].

Uma outra característica importante das redes P2P é a tolerância a falhas. Quando ocorre a queda ou uma simples saída de um *peer* da rede, a aplicação permanece funcionando. Os demais *peer* podem facilmente suprir o trabalho feito pelo *peer* que não está mais respondendo. Por exemplo, em uma aplicação BitTorrent [Cohen, 2003], cada

cliente trabalha também como um servidor enquanto faz *download* de arquivos da rede. Quando algum dos parceiros de um determinado *peer* deixa de responder, os dados desejados podem ser obtidos de algum de seus outros parceiros e, assim, o *download* pode continuar normalmente. No modelo cliente-servidor, a comunicação é interrompida caso o servidor tenha algum problema [Hei, 2008a].

2.2 Arquiteturas P2P

De acordo com a organização da rede sobreposta, as redes P2P podem ser classificadas como não estruturadas ou estruturadas [Androutsellis-Theotokis, 2004, Li, 2008]. A classificação é feita de acordo com a topologia e com a maneira como o sistema é criado e mantido. A organização de uma rede P2P tem reflexos importantes na robustez, desempenho e segurança dos sistemas.

2.2.1 Redes P2P Não Estruturadas

Nas redes P2P não estruturadas, a topologia é determinada de forma aleatória. Após entrar no sistema, os nós estabelecem conexões com participantes sem seguir nenhuma regra fixa. Nas redes P2P não estruturadas a localização dos recursos é independente da topologia.

O maior desafio em sistemas P2P não estruturados é o processo de busca por conteúdo. Esses processos têm grande influência no desempenho do sistema, principalmente na manutenção, disponibilidade e escalabilidade [Androutsellis-Theotokis, 2004]. Inicialmente, o processo de busca era feito através do método por inundação, onde o *peer* requisitava o conteúdo a cada um de seus parceiros, e estes repetiam o processo até encontrar o conteúdo desejado (ou até a busca alcançar um certo raio limite). Embora seja uma solução simples e resistente a falhas, ela não é escalável [Androutsellis-Theotokis, 2004]. Isso motivou o desenvolvimento de novas técnicas que fossem mais eficientes no uso dos recursos da rede. Por exemplo, técnicas como a caminhada aleatória [Gkantsidis, 2004], os índices de roteamento [Tsoumakos, 2003] e as Tabelas *Hash* Distribuídas (DHT). Sistemas não estruturados são bastante adequados para trabalhar com nós que apresentam grande dinamicidade, ou seja, quando entram e abandonam o sistema constantemente.

A arquitetura dos sistemas não estruturados pode ser classificada quanto ao grau de

centralização da rede, como:

- Centralizada: nessa arquitetura existe uma entidade centralizadora que regula a entrada e saída de nós do sistema, podendo também controlar o mapeamento da localização de conteúdo na rede. Desse modo, a construção e gerenciamento da rede sobreposta são simplificados. Por existir uma entidade centralizadora, esse sistema tem a desvantagem de ter um ponto único de falha. O principal exemplo desses sistemas é o Napster [1].

- Descentralizada: nessa arquitetura não há um ponto central que controla as atividades da rede e cada nó possui exatamente o mesmo comportamento e todos os nós realizam as mesmas operações. Todos os nós agem como servidor e cliente ao mesmo tempo. O principal método de busca utilizado por esses sistemas é a *inundação de consultas*. Um exemplo de destaque desse tipo de sistema é o Gnutella [Ripeanu, 2002].

- Híbrida: essa arquitetura tem características semelhantes às características dos sistemas descentralizados, com uma diferença básica: nem todos os nós possuem as mesmas operações ou comportamentos. Os nós que possuem mais recursos (largura de banda, processamento) são tratados como líderes de grupo (super-nós) e possuem maiores responsabilidades. A função desses super-nós é gerenciar a busca por conteúdo. Apesar de facilitar a manutenção da rede e permitir um melhor desempenho na busca por conteúdo, nem sempre essa busca alcança sucesso. Kazaa [Leibowitz, 2003] e BitTorrent [Cohen, 2003] são exemplos desse tipo de sistema.

2.2.2 Redes P2P Estruturadas

As redes P2P estruturadas surgiram como uma solução para o problema de escalabilidade gerado pelos mecanismos de busca por conteúdo utilizados nas arquiteturas não estruturadas [Androutsellis-Theotokis, 2004]. Nesse tipo de rede, a criação e manutenção da rede sobreposta seguem regras bem definidas. A localização de um determinado recurso é associada a um nó de forma determinística e é conhecida por todo o sistema. Essa associação é feita atribuindo, a um determinado nó, a chave que identifica unicamente um recurso. Desse modo, é possível obter um mapeamento entre identificador do recurso e

[1] www.napster.com

sua localização. Essa identificação pode ser feita na forma de uma de tabela roteamento distribuída. Assim, qualquer pedido pode ser eficientemente direcionado para o nó que contém o recurso [Lv, 2002a].

Como as consultas são direcionadas para o local exato do recurso, é possível evitar problemas causados por técnicas como a inundação. A grande desvantagem das redes estruturadas é o alto custo para manter a estrutura necessária para que os pedidos sejam roteados de maneira eficiente [Lv, 2002b]. Os participantes desse sistema podem ter um comportamento bastante dinâmico, entrando e saindo do sistema com frequência, fato que dificulta a manutenção da estrutura da rede. Alguns exemplos de sistemas P2P estruturados são: Chord [Stoica, 2001] e CAN [Ratnasamy, 2001].

2.3 Exemplos de Sistemas P2P

2.3.1 BitTorrent

O BitTorrent [Cohen, 2003] foi desenvolvido com foco na distribuição de conteúdo utilizando a estrutura de uma rede P2P e é uma das aplicações mais populares para compartilhamento de arquivos. Uma entidade centralizada (*tracker*) é responsável por controlar a entrada de nós no sistema. O *tracker* gerencia cada participante através de seus endereços de rede e as partes de arquivos desejadas e presentes em cada um. Ao entrar no sistema, um novo participante contacta o *tracker* para obter uma lista de participantes interessados no mesmo arquivo. Assim, os participantes se conectam, formando grupos e trocando dados entre si.

Para incentivar o bom comportamento dos participantes, o BitTorrent utiliza um mecanismo conhecido como "*tit-for-tat*" [Cohen, 2003]. Nesse mecanismo, os participantes do sistema somente trocam arquivos com outros participantes que também compartilhem dados. Um participante somente receberá dados e conseguirá boas parcerias se for um bom parceiro para os demais participantes. Essa política de incentivo pune usuários que tenham comportamento egoísta, ou seja, que não compartilham dados com seus parceiros (conhecidos como *free-riders*). Um bom participante deve conseguir taxas de transferência de dados mais altas e maior acesso a recursos do sistema devido a essa política.

A busca por conteúdo não é uma funcionalidade implementada por esse sistema. Antes de distribuir um objeto, é necessário criar um arquivo *torrent*. Esse arquivo contem as

informações necessárias para a distribuição do objeto, como endereço de rede e porta do *tracker*, quantidade de blocos em que o arquivo foi dividido e *hash* de cada bloco para validação. Geralmente, o arquivo *torrent* é publicado em alguma página *web*, permitindo sua busca e localização por qualquer mecanismo *web*.

2.3.2 Gnutella

O Gnutella [Ripeanu, 2002] é um sistema P2P não estruturado e descentralizado. Nós que possuem maior largura de banda e processamento são preferidos no momento da escolha das parcerias. Essa característica facilita a busca por conteúdo, aumentando o desempenho da rede. Cada nó na rede Gnutella tem uma visão parcial da rede, ou seja, conhece somente os participantes aos quais está diretamente conectado. Os demais nós são desconhecidos, a menos que se anunciem respondendo alguma requisição.

Para se juntar à rede, um nó precisa somente saber o endereço de rede e porta TCP de algum outro nó que seja um participante do sistema. Após se conectar a este nó, o nó recém-chegado envia mensagens de anúncio na rede para descobrir outros parceiros.

A busca por conteúdo no Gnutella é feita através de inundação de mensagens. Um nó envia uma requisição para seus parceiros, e cada um deles envia a mensagem aos nós a que está conectado. Esse processo, apesar de ser bastante adequado para atender à dinamicidade dos nós, não se mostra escalável e gera sobrecargas na rede [Lua, 2005].

2.3.3 FastTrack

O FastTrack é um protocolo de rede P2P que foi usado por sistemas descentralizados de compartilhamento de arquivos como o Kazaa [Leibowitz, 2003] e o Morpheus (www.morpheus.com). Essa rede faz uso de super-nós para tornar a busca pelo conteúdo mais rápida e eficiente. Cada participante envia as informações sobre os arquivos que possui para os supernós. A busca nesse sistema é feita através de inundação, porém com uma diferença importante em relação ao Gnutella original: a existência dos super-nós torna a busca mais rápida. Assim como no BitTorrent, o FastTrack permite que usuários façam *download* de múltiplas fontes ao mesmo tempo.

2.3.4 Chord

A rede sobreposta do sistema Chord é organizada em forma de um anel lógico onde cada participante mantem partes específicas da tabela de roteamento e do conteúdo disponível na rede [Stoica, 2001]. Essa tabela de roteamento distribuída guarda pares *chave-conteúdo*. Cada chave fica associada a um nó que é responsável por um determinado conteúdo.

O protocolo Chord especifica como uma chave é associada aos nós, e como um nó pode descobrir um conteúdo a partir de uma chave. Basicamente, um participante precisa saber para qual nó dentro do anel ele deve encaminhar a solicitação. As pesquisas são direcionadas para esse nó até que alcance o conteúdo desejado.

Ao se juntar à rede, o nó recém-chegado recebe uma chave que o identifica e partes das chaves/conteúdo que seu sucessor possui. Quando algum participante deixa o sistema, seu sucessor assume as suas chaves e seus conteúdos.

2.3.5 CAN - Rede de Conteúdo Endereçável

A rede CAN é um sistema não estruturado que possui características como escalabilidade e tolerância a falhas [Ratnasamy, 2001]. Um nó recém-chegado no sistema deve primeiro encontrar e se conectar a um nó pertencente à CAN. A seguir, o nó recém-chegado deve receber uma parcela dos recursos que estão sob responsabilidade de algum nó. Por fim, os demais participantes devem ser notificados para modificar suas tabelas de roteamento. Essa característica permite que o sistema seja escalável.

Essa rede funciona como uma tabela *hash*; cada participante do sistema realiza operações como inserção, busca e remoção de pares (chave,valor). Cada nó na rede CAN guarda um pedaço da tabela *hash* completa do sistema e partes das tabelas de seus nós parceiros. As requisições por uma chave em particular são roteadas dos nós intermediários para os nós que possuem o recurso desejado.

2.4 Distribuição de Vídeo Utilizando Arquiteturas P2P

Nessa seção serão discutidas as duas principais abordagens para transmissão de vídeo utilizando arquiteturas P2P: vídeo sob demanda, quando o vídeo está pré-armazenado ou codificado; e vídeo ao vivo, transmitido no momento em que é produzido e visualizado

pelos usuários quando recebem os dados. Adicionalmente serão discutidas as três principais topologias utilizadas para organizar o *overlay* P2P em sistemas de distribuição de vídeo [Moraes, 2008]: topologia em árvore (*tree-push*), em malha (*mesh-pull*) e a híbrida.

Um sistema de vídeo sob demanda (*VoD - Vídeo On Demand*) deve permitir aos usuários assistir aos vídeos armazenados no momento desejado, bem como controlar o fluxo da exibição do vídeo de acordo com suas necessidades. Usualmente, os serviços de vídeo sob demanda apresentam grande interatividade com os usuários. O serviço *VoD* permite aos usuários realizar ações como: executar e parar um vídeo, pausar e retomar a execução, avançar e recuar. Um exemplo de sistema *VoD* é o Youtube [2].

Ao contrário dos sistemas *VoD*, os participantes de um sistema de distribuição de vídeo ao vivo devem estar sincronizados e assistir ao fluxo de vídeo ao mesmo tempo. Essa restrição existe porque o conteúdo de vídeo é transmitido no mesmo momento em que o fluxo é gerado. Cada pedaço de vídeo tem um tempo de vida determinado, após o qual são descartados. Isto é necessário para manter a característica ao vivo da transmissão.

Na literatura existem várias propostas para distribuição de vídeo ao vivo utilizando P2P [Hei, 2007]. Elas se diferenciam, basicamente, pela estrutura utilizada para organizar a rede sobreposta. A estrutura da rede sobreposta pode ser organizada em malha, em árvore ou ser híbrida, combinando as duas abordagens anteriores.

2.4.1 Topologia em Árvore (Tree-push)

Nesse tipo de topologia, os nós formam uma árvore de distribuição de conteúdo em nível de aplicação, sendo a raiz a origem do fluxo. Cada nó que entra nessa árvore conhece seu antecessor (pai) e sucessores (filhos). Ao receber um dado, o nó encaminha uma cópia para seus filhos sem que haja um pedido explícito. A partir do momento em que o fluxo de vídeo é criado, este é difundido de pai para filho dentro da estrutura da árvore, como mostra a Figura 2.1.

No caso da topologia em árvore, quando a entrada e saída de participantes não é frequente, a sobrecarga e a latência são baixas. Com a árvore construída, não há necessidade de envio de mensagens de controle, e somente dados de vídeo são enviados. Essa característica provê uma boa continuidade na exibição dos fluxos de vídeo. Porém, se a dinamicidade de nós é alta, a árvore deverá ser reconstruída com frequência, fato que deve

[2] www.youtube.com

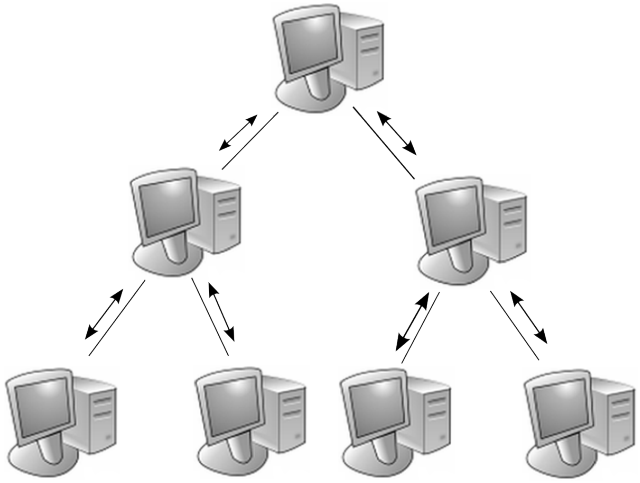


Figura 2.1: Difusão de conteúdo em P2P na arquitetura em árvore. As setas indicam troca de mensagens de controle e dados.

gerar maior sobrecarga de controle. Um problema ainda maior com essa dinamicidade é que toda vez que acontece uma saída de um participante, todos os seus filhos sofrem temporariamente com descontinuidade na exibição do fluxo de vídeo. Por isso, é de grande importância que o tempo de recuperação da árvore, após a saída de nós, seja pequeno.

Nas topologias em árvore, a maioria dos nós do sistema são folhas da árvore e não compartilham seus recursos com os demais nós do sistema. Essa característica afeta negativamente o desempenho do sistema e grande quantidade de recursos é desperdiçada. Outro ponto crítico dessa estrutura é o balanceamento dos nós. Como os pacotes de um determinado fluxo de vídeo vão seguir o mesmo caminho até o receptor, esses fluxos podem gerar congestionamento na rede caso a árvore não esteja bem balanceada. A diferença de largura de banda entre os nós também pode ser um fator negativo. Os filhos de um nó que possui baixa largura de banda podem ser prejudicados, recebendo o fluxo de vídeo com baixa qualidade.

Como exemplo, em [Jannotti, 2000] os autores apresentam um sistema que utiliza a topologia em árvore, chamado *Overcast*, que provê uma solução alternativa para o *multicast* IP utilizando um protocolo simples para construção de árvores de distribuição de dados que se adapta às condições da rede. Os autores mostram que o *Overcast* provê suas funcionalidades de maneira competitiva ao IP *multcast*.

2.4.2 Topologia em Malha (Mesh-pull)

Nos sistemas P2P organizados em malha, o nó origem do conteúdo divide um fluxo de vídeo em pedaços, chamados *chunks*. Estes pedaços ficam então disponíveis no nó origem para serem transferidos pela rede. Nessa topologia não é construída uma estrutura explícita para a distribuição do fluxo de vídeo. Ao invés disso, o que é disseminado na rede é a disponibilidade de vídeos em cada nó, ou seja, os nós somente avisam aos seus parceiros os pedaços de vídeo que possuem ao invés de repassá-los diretamente. Os participantes podem solicitar explicitamente os pedaços de vídeo que desejam. Esta é a principal diferença entre a topologia em malha e a topologia em árvore. A Figura 2.2 exemplifica o funcionamento das redes baseadas na topologia em malha. Todos os participantes do sistema podem trocar dados entre si, sem obedecer uma estrutura fixa.

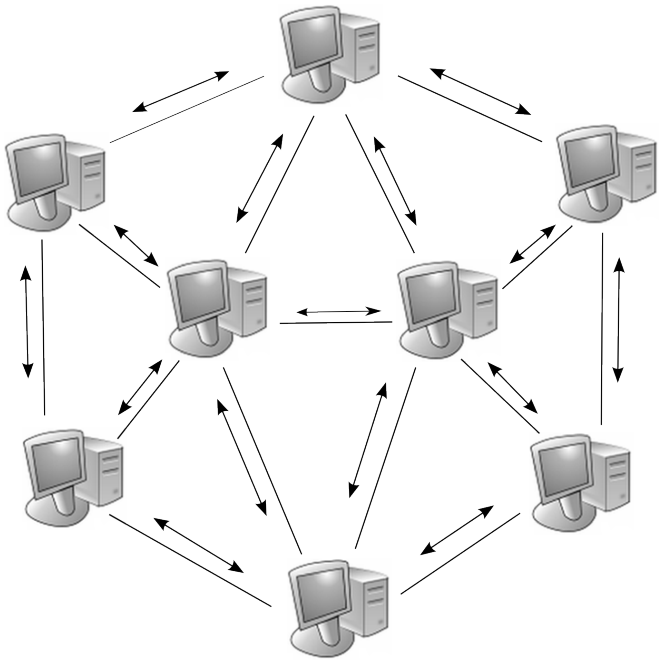


Figura 2.2: Difusão de conteúdo em P2P na arquitetura em malha

Nessa arquitetura um participante especial, chamado *bootstrap*, que é um servidor que mantém uma lista de participantes que estão interessados no mesmo conteúdo, e é responsável por gerenciar a entrada de nós na rede. Quando um novo nó entra na rede, ele recebe do *bootstrap* uma lista de participantes que estão assistindo ao mesmo conteúdo solicitado. Ao receber essa lista o nó recém-chegado se conecta com esses participantes para formar seu grupo de parceiros. Estes nós cooperam entre si para ter acesso ao conteúdo de vídeo. Cada nó pode também se conectar diretamente ao nó origem do

vídeo. A Figura 2.3 ilustra este comportamento. No passo 1, o novo nó se registra no *bootstrap*. A seguir, o *bootstrap* envia a lista de participantes (passo 2). Após ter esta lista, o novo nó forma suas parcerias (passo 3).

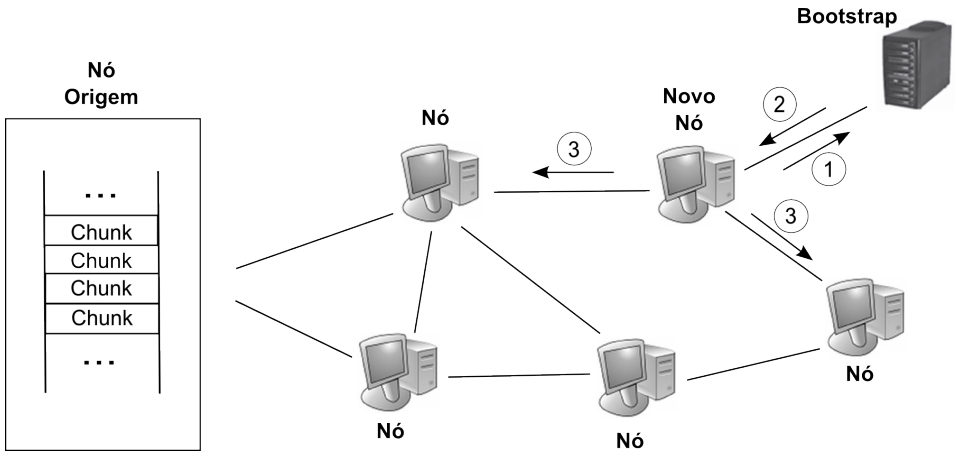


Figura 2.3: Passos para entrada no sistema.

Para que cada nó conheça o conteúdo disponível em seus parceiros, todos os participantes do sistema enviam um mapa de *chunks* para seus parceiros. Esse mapa de *chunks* indica quais pedaços do vídeo um determinado nó possui. Com essa informação, cada nó pode solicitar novos pedaços de vídeo. Em intervalos de tempo, cada nó procura por outros parceiros a partir da lista recebida do *bootstrap* e de seus próprios parceiros.

Cada participante possui um *buffer* que funciona como uma janela deslizante. O tamanho desse *buffer* é reduzido e armazena somente uma pequena quantidade de *chunks* que serão exibidos no próximo intervalo de tempo. Conforme os *chunks* são exibidos, a janela deslizante se move e esses *chunks* são descartados. Os processos de verificação do mapa de *chunks*, pedido, recebimento e exibição de *chunks* são mostrados na Figura 2.4. No passo 1, o participante p_i recebe e envia, ao mesmo tempo, um mapa de *chunks*. A seguir, no passo 2, p_i conhece os parceiros que possuem o *chunk* desejado. Desse modo, p_i é capaz de fazer uma requisição. O *peer* p_i pode receber requisições de um ou mais parceiros. Por fim, no passo 3, o parceiro que recebeu a requisição de p_i disponibiliza o dado pedido para que p_i possa fazer o *download*. Ao mesmo tempo, p_i faz *upload* de algum *chunk* que possui.

Uma das vantagens da topologia em malha, comparada à árvore, é a maior resiliência à alta dinamicidade (entrada e saída do sistema) dos nós. Como o conteúdo está disponível em vários nós, quando um participante deixa o sistema, os *chunks* desejados ainda podem

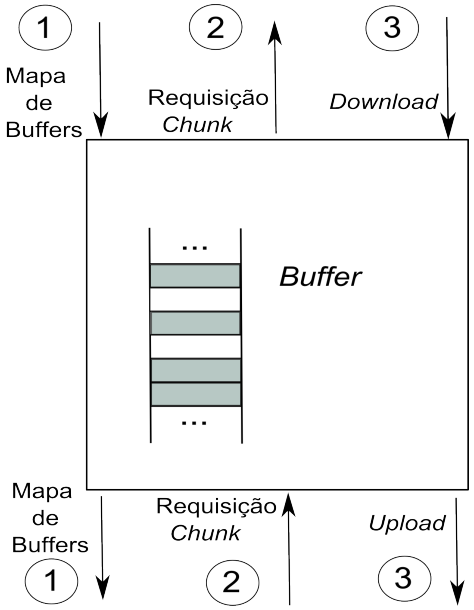


Figura 2.4: Exemplo de Janela Deslizante.

ser encontrados nos parceiros que permanecem. Essa característica afeta diretamente a qualidade da exibição do vídeo. Nos sistemas em malha, a possibilidade de descontinuidade na recepção do vídeo é reduzida. Em contrapartida, a troca de informações sobre a disponibilidades dos *chunks* em cada participante gera uma maior sobrecarga de controle na rede. Além disso, como não há uma estrutura explícita de distribuição de vídeo, o atraso de inicialização do vídeo pode ser maior do que na topologia em árvore.

O desempenho dos sistemas baseados na topologia em malha está fortemente ligado ao tamanho dos *buffers* em cada nó. Os *chunks* podem ser recebidos fora de ordem e terão de ser armazenados para evitar retransmissão. Assim, quanto maior o tamanho dos *buffers*, mais *chunks* eles podem armazenar, aumentando, assim, a disponibilidade do vídeo na rede. Entretanto, o sistema deixaria de ser ao vivo.

Aplicações que utilizam a arquitetura P2P em malha possuem grande sucesso comercial [Hei, 2008b]. Como exemplo dessas aplicações temos PPLive [3], PPStream [4] e o SopCast [5] que apresentam uma grande variedade de canais e suporte para vários sistemas operacionais.

[3] www.pplive.com
[4] www.ppstream.com
[5] www.sopcast.org

2.4.3 Topologia Híbrida

A topologia híbrida busca unir as vantagens das topologias em árvore e em malha para construir uma topologia eficiente e robusta. As topologias em árvore e em malha não são capazes de resolver todos os desafios inerentes aos sistemas P2P de transmissão de vídeo ao vivo. A estrutura em árvore sofre muito com a dinâmica de nós, com a subutilização de banda dos nós folhas e com a escolha de nós pais ineficientes. Por sua vez, a topologia em malha pode ter problemas com sobrecarga de mensagens de controle.

Existem alguns esforços na literatura com a finalidade de construir aplicações baseadas na topologia híbrida. Por exemplo, GridMedia [Zhang, 2005] e Coolstream+ [Xie, 2007]. Em [Wang, 2007], os autores afirmam que mesmo utilizando uma estrutura em malha, a disseminação de um conteúdo tende a formar um pequeno conjunto de árvores de distribuição. Essas árvores são formadas por um subconjunto de nós que geralmente permanecem por longos períodos de tempo no sistema. Os autores propõem uma hierarquia em árvore a partir da origem do vídeo com esses nós estáveis, enquanto os demais nós se organizam em malha. O desafio, nesse caso, é identificar os nós estáveis para formar o núcleo apropriado. Nem sempre os nós mais estáveis são aqueles que possuem maiores recursos (largura de banda, processamento). Assim, se forem colocados nós com poucos recursos na árvore próxima à origem, todo o sistema pode ficar comprometido.

2.4.4 Estratégias de Seleção de Chunks

A estratégia de seleção de *chunks* define as regras que cada participante segue no momento de solicitar um *chunk*. A estratégia escolhida tem influência direta no desempenho do sistema P2P de transmissão de vídeo ao vivo. Essas estratégias visam manter a continuidade de exibição de vídeo nos *peers* e difundir o conteúdo de vídeo o mais rápido possível. As duas principais estratégias de seleção de *chunks* encontradas na literatura são: 1) *Mais Raro Primeiro* [Zhao, 2009], usada por sistemas como BitTorrent [Cohen, 2003] e Coolstreaming [Xie, 2007]; e 2) *Próximo Deadline Primeiro* [Zhao, 2009].

Na estratégia *Mais Raro Primeiro*, um participante solicita o *chunk* mais recente no sistema, ou seja, aquele que acabou de ser criado pelo nó servidor ou o que está menos replicado entre seus parceiros. Nessa estratégia, o *chunk* mais raro a ser buscado é sempre aquele que acaba de ser criado pelo nó origem. Essa estratégia permite que este *chunk* seja disseminado mais rapidamente pelo sistema.

A estratégia *Próximo Deadline Primeiro* tem como objetivo preencher os espaços no *buffer* que estão mais próximos de serem exibidos. Utilizando essa estratégia, os participantes do sistema tendem a armazenar os dados mais antigos produzidos pelo nó origem.

2.5 Combate a Ataques de Poluição de Dados em Sistemas P2P

Em redes P2P, são necessários mecanismos para verificação de conteúdo para permitir que cada participante verifique se os dados recebidos são poluídos ou legítimos. Esse mecanismo também deve tratar um dado corrompido como poluído. Essa característica é importante, pois, tanto dados poluídos, como corrompidos, não devem ser exibidos.

Nas abordagens de verificação de conteúdo, todo participante é obrigado a verificar os dados recebidos antes de consumi-los ou compartilhá-los. Caso essa verificação não seja feita, os participantes podem encaminhar dados poluídos/corrompidos para seus parceiros, sendo considerados poluidores. Na literatura algumas abordagens recebem destaque:

- Uma das maneiras de fazer essa verificação é utilizar uma função *hash* para gerar um resumo de cada dado que será enviado [Haridasan, 2007]. Esse resumo é assinado com a chave privada do nó servidor de vídeo e é enviado pela rede. Juntamente ao conteúdo recebido por um participante, vem o resumo assinado daquele conteúdo. Com essas informações, o participante é capaz de recalcular o valor de *hash* do conteúdo recebido e comparar com o resumo assinado que recebeu. Assim, o participante saberá se o dado recebido foi corrompido ou alterado, além de saber se este dado foi, de fato, criado pelo servidor do vídeo.

  A implementação simplificada do mecanismo de verificação de *hash* consiste em assinar cada *chunk* antes de difundi-lo pela rede. Porém, essa abordagem introduz um alto custo computacional. O servidor de vídeo terá de computar um valor de *hash* para cada *chunk* gerado. Cada participante, ao receber esses *chunks* terá que calcular um valor de *hash* a partir do dado recebido e compará-lo com o resumo recebido do seu parceiro. Essa abordagem gera uma sobrecarga de processamento nos participantes e no servidor [Dhungel, 2007].

- Outra maneira é utilizar uma técnica conhecida como *Linear Digests* [Haridasan,

2008]. Essa técnica visa evitar o problema da sobrecarga de processamento causada pela geração e verificação dos valores de *hash* no servidor e nos nós participantes do sistema. A ideia é criar um valor de *hash* para cada grupo de *n chunks* gerados, sendo esse grupo assinado pela origem. O nó servidor de vídeo deve enviar a mensagem com as assinaturas antes das mensagens de dados para os participantes. Assim, cada participante pode verificar a autenticidade de qualquer *chunk* presente naquele grupo [Coelho, 2010]. Essa característica pode causar atraso na exibição do vídeo pela rede, dado que a origem deve gastar certo tempo para agrupar os *chunks*. Uma vantagem dessa abordagem é a menor sobrecarga causada na rede.

A simples verificação dos dados recebidos pode não ser eficiente para combater totalmente os efeitos de ataques de poluição [Vieira, 2008]. Sempre que um participante identificar um dado recebido como poluído, ele será obrigado a solicitar esse dado novamente. Essa característica ainda pode gerar atrasos na exibição e grande sobrecarga na rede devido às novas requisições [Vieira, 2008].

Devido a esse problema, surgiram propostas para atribuir pontos de reputação aos participantes do sistema, punindo os que não compartilham dados na mesma proporção com que os obtêm. Sendo assim, a reputação consiste em um mecanismo capaz de avaliar o comportamento de um participante da rede P2P através de suas ações passadas, atribuindo-lhe uma pontuação. Quanto menor for a pontuação de um participante, menos acesso aos recursos do sistema ele possui. Esse mecanismo incentiva o comportamento altruísta, ao mesmo tempo em que desencoraja o comportamento malicioso dos participantes em um sistema P2P.

Assim, além de utilizar um tipo de verificação de dados, é de grande importância utilizar outro mecanismo que permita punir participantes maliciosos em um sistema P2P. Existem duas abordagens principais para se desenvolver um mecanismo de reputação.

- **Lista Negra Centralizada**: essa abordagem proposta em [Liang, 2005] utiliza um nó centralizador responsável por manter uma lista que guarda identificadores de nós poluidores. Todos os demais participantes do sistema podem acessar essa lista a qualquer momento. Quando um participante p_i recebe uma requisição para formação de parceria de um participante p_j, p_i pode verificar se o identificador de p_j está nesta lista. Caso estiver, p_i pode evitar parcerias com p_j. Além disso,

p_i pode verificar a lista negra regularmente para verificar se seus parceiros foram caracterizados como maliciosos [Dhungel, 2007, Haridasan, 2006, Haridasan, 2008].

Quando um bom participante reporta um mau comportamento de um determinado parceiro p_j, um identificador de p_j é adicionado à lista negra. Geralmente, o nó centralizador pondera as informações recebidas para evitar distorções na lista negra. Por exemplo, em [Jin, 2006], o nó centralizador ao receber uma reputação sobre algum participante leva em consideração a média das demais reputações recebidas para dar um peso à essa informação. Assim, o sistema atribui mais importância às reputações que tem valor próximo à média. Essa abordagem é suscetível a ataques de difamação feitos em conluio [Vieira, 2009]. As reputações enviadas por um grupo de participantes maliciosos farão com que a reputação recebida pelo nó centralizador se aproxime cada vez mais do valor enviado por esse grupo. Dessa maneira, esse grupo de participantes maliciosos conseguiria isolar bons parceiros no sistema.

Existem casos em que a reputação é ponderada pela credibilidade do participante que a definiu [Wang, 2006, Xiong, 2004]. Nesse caso, informações enviadas por participantes com maior reputação terão pesos maiores. Apesar desta modificação, esta abordagem está suscetível a ataques de difamação. Um participante malicioso pode se comportar de maneira altruísta com todos os seus parceiros para obter uma boa reputação. Porém, sempre envia falsas reputações de seus parceiros ao nó centralizador. Como esse participante tem uma boa reputação, suas informações receberiam um peso maior.

- **Reputação Distribuída:** Nos modelos mais comuns de reputação distribuída, um participante p_i calcula a pontuação de reputação de um parceiro p_j baseado na sua experiência individual e no depoimento da rede sobre ele [Vieira, 2008, Costa, 2007a]. Nesse modelo, não há uma entidade centralizada que controla a pontuação de reputação dos participantes. A decisão sobre manter ou punir um participante é tomada a partir da pontuação local de cada nó.

  Explorar o depoimento da rede pode não ter um custo-benefício satisfatório para aplicações P2P de transmissão de vídeo ao vivo. Um participante nesse tipo de sistema apresenta interações com seus parceiros por curtos intervalos de tempo. Assim, se comparado a taxa de interação entre os nós, o depoimento da rede sobre

um determinado participante pode convergir muito lentamente. Além disso, o depoimento da rede sobre um determinado parceiro p_j pode não ser confiável. Isso acontece quando um participante p_i calcula a reputação de p_j mas não possui muitos parceiros que também sejam parceiros de p_j.

Assim, Vieira *et al.* [Vieira, 2009] apresenta uma abordagem mais simples que desconsidera o depoimento da rede para o cálculo da reputação. Nessa abordagem, um participante calcula a pontuação de seus vizinhos baseado somente em sua experiência individual, desconsiderando o depoimento da rede.

3 Trabalhos Relacionados

Ao se desenvolver um sistema P2P de transmissão de vídeo ao vivo, espera-se que os participantes assumam um comportamento altruísta e não malicioso. Porém, nem sempre essa é a realidade. Vários participantes podem assumir um comportamento egoísta e/ou malicioso, com o intuito de prejudicar a estrutura ou o serviço do sistema.

Por esse motivo, o desenvolvimento de medidas de combate ao comportamento malicioso/oportunista é de grande importância. Além disso, como sistemas P2P de transmissão de vídeo ao vivo têm como fator crítico o tempo de distribuição do conteúdo como fator crítico, essas medidas devem reagir ao mal comportamento em um tempo muito curto para que o desempenho do sistema não seja comprometido [Lin, 2010].

Entre os diversos tipos de ataques encontrados na literatura, destacam-se os ataques de poluição de conteúdo e *whitewashing*. O primeiro pode causar danos consideráveis ao sistema P2P de transmissão de vídeo ao vivo. Nele, um participante malicioso altera ou corrompe o conteúdo antes de encaminhá-lo aos seus parceiros, repassando uma mídia falsa ou inválida. Soluções isoladas para combater ataques de poluição são apresentadas na literatura [Dhungel, 2007,So, 2012,Vieira, 2008,Vieira, 2009]. Porém, todas elas podem falhar caso o participante poluidor deixe o sistema e volte com uma nova identidade para evitar as penalidades impostas devido ao seu mau comportamento. Ou seja, tais soluções podem falhar quando o participante poluidor faz *whitewashing*. Entre outros ataques e comportamento maliciosos, destacam-se [Seibert, 2010]:

- **Descarte de Dados**: atacante atrai alguns parceiros prometendo servi-los. Esses atacantes anunciam-se com muitos recursos e conteúdo. Porém, quando um parceiro requisita o serviço, o atacante não envia os dados. Esse comportamento pode causar atrasos e descontinuidade da exibição do vídeo nos participantes que foram atraídos pelo atacante.

- **Conluio**: ocorre quando vários atacantes realizam ataques combinados para causar danos maiores ao sistema. Por exemplo, os atacantes em conluio podem atribuir uma boa reputação para seus parceiros poluidores para enganar um mecanismo de lista negra.

- **Egoísmo (*Free Riding*)**: participante malicioso recebe dados de seus parceiros mas não os envia para os outros participantes, não contribuindo com o sistema. Esse comportamento pode fazer com que o sistema sofra com escalabilidade no número de usuários e a qualidade do serviço pode cair.

- **Ataques de Eclipse**: ocorre quando vários atacantes combinados controlam grande parte dos parceiros de um participante legítimo. Desse modo, os atacantes podem ocultar o participante legítimo do restante da rede P2P. Os atacantes podem se desfazer de toda mensagem de/para esse participante ou até mesmo, se anunciar como sendo o participante legítimo. Caso esses atacantes isolem o *peer* origem de vídeo, todo o sistema pode ficar comprometido.

- **Ataque de Consumo de Recursos**: nesse tipo de ataque, o atacante inunda seus parceiros com requisições, mesmo sem precisar dos dados. Com isso, o atacante tenta exaurir os recursos de seus parceiros e fazer com que eles não sejam capazes compartilharem mais dados.

Os sistemas P2P de transmissão de vídeo ao vivo podem sofrer com ataques tanto sobre os dados transmitidos, quanto sobre as mensagens de controle utilizadas pelo protocolo do sistema. Em ataques realizados sobre as mensagens de controle, os atacantes podem alterar informações importantes sobre os participantes do sistema ou sobre si mesmo. Por exemplo, mentindo sobre a intensidade de suas parcerias ou enviando um mapa de *chunks* alterado [Seibert, 2010].

Nos ataques realizados sobre os dados, os atacantes podem corromper dados, atrasar a entrega, enviar dados duplicados, ignorarem pedidos ou compartilhar dados falsos. Como consequência pode haver, por exemplo, queda na qualidade do fluxo de vídeo disponível, recepção de informações falsas, diminuição do número de participantes do sistema, dentre outros [Seibert, 2010].

Além disso, alguns ataques dependem diretamente da topologia da rede P2P. Por exemplo, a rede P2P organizada em forma de árvore pode sofrer com nós que mentem sobre os recursos que têm disponível. Nesse caso, um nó p_k que possui recursos limitados pode atrair vários filhos forjando possuir grande capacidade de *upload*, quando na verdade não possui. Nesse caso, os danos causados ao sistema podem ser consideráveis, visto que, todo o restante da árvore abaixo de p_k ficará comprometida, experimentando atrasos na

exibição do vídeo. O mesmo acontece se o atacante negar as solicitações recebidas, ocultar seus vizinhos do resto da rede ou enviar dados corrompidos.

Em grande parte dos casos, os atacantes podem combinar várias técnicas para reforçar o impacto dos ataques. Podem atacar a topologia e o controle da rede ao mesmo tempo, assim como os dados transmitidos. Além disso, os atacantes podem se organizar para gerar ataques combinados, com o intuito de causar mais danos e tornar o combate aos ataques mais difícil.

Os primeiros esforços para combater ataques de poluição de conteúdo em redes P2P de transmissão de vídeo ao vivo apresentavam uma proposta de um sistema resistente a alteração de conteúdo e intrusão de participantes maliciosos [Dhungel, 2007, Haridasan, 2006, Haridasan, 2008]. Em [Haridasan, 2006] os autores mostram que é possível evitar a execução de trechos de mídia poluídos realizando a verificação do conteúdo recebido da rede antes de exibi-lo, reduzindo a chance do conteúdo poluído ser disseminado pela rede. Apesar de essas soluções evitarem a exibição de conteúdo poluído, ainda existe o problema da sobrecarga gerada devido às novas requisições. A simples verificação de dados não é capaz de banir os nós maliciosos do sistema. Devido a esse problema, surgiram propostas baseadas em um mecanismo de reputação para punir participantes maliciosos [Vieira, 2008, Seibert, 2010, Yu, 2011].

Aliado aos problemas de ataques de poluição, os nós podem alterar sua identidade para tentar enganar os sistemas de reputação. Esta prática, conhecida como *whitewashing*, pode tornar o mecanismo de reputação ineficaz. Na literatura duas abordagens têm sido bastante usadas para combater ataques de poluição combinados com *whitewashing* em sistemas P2P de transmissão de vídeo ao vivo.

A primeira abordagem faz uso de uma entidade centralizada de confiança, que é responsável por associar fortes identidades ligadas a informações únicas dos *peers* recém-chegados no sistema. Essa abordagem, além de introduzir um ponto único de falha, que pode comprometer o funcionamento do sistema, vai de encontro à natureza descentralizada dos sistemas P2P [Oualha, 2009].

A segunda abordagem impõe uma penalidade a todos os *peers* recém-chegados no sistema [Seibert, 2010]. No entanto, como mostrado em [Feldman, 2006], essa abordagem pode afetar negativamente a escalabilidade do sistema. Feldman *et al.* afirmam que *whitewashing* é um comportamento difícil de ser observado e, sendo assim, a penalidade é

aplicada a todos os *peers* recém-chegados. Além disso, o desempenho do sistema também seria comprometido caso houvesse alta dinâmica dos *peers*.

3.1 Proposta *Simpler is Better*

A proposta *Simpler is Better* [Vieira, 2009] apresenta um mecanismo simplificado e descentralizado de reputação para combater *ataques de poluição* em redes P2P de transmissão de vídeo ao vivo. O mecanismo de reputação utilizado desconsidera o depoimento da rede. Um participante p_i apenas monitora a troca de dados direta com seu parceiro p_j para computar sua reputação.

Um nó p_i computa sua experiência individual com p_j baseado na fração de *chunks* poluídos recebidos de p_j. Assuma que, durante um intervalo de tempo p_i solicita r *chunks* a p_j, e que este responde com n *chunks* poluídos. Se a razão n/r for menor que um limite estipulado por p_i, então p_i considera p_j como um bom parceiro, aumentando sua reputação. Caso contrário, p_i considera p_j como mau parceiro e diminui sua reputação. Se a reputação de p_j se tornar menor que a reputação mínima aceita por p_i (R_i^{min}), a parceria entre os nós é desfeita.

Além do mecanismo para banir participantes maliciosos, Vieira *et al.* apresentam um esquema que permite a reabilitação de participantes através de um mecanismo de variação da reputação mínima (R_i^{min}). Foram definidos dois estados de sistema: *calmaria* e *tempestade*. Para determinar o estado do sistema, um participante p_i monitora todas as suas parcerias. Caso não receba dados poluídos de mais de um parceiro p_i após consecutivas interações, ele considera que o sistema está em *calmaria*, caso contrário, está em *tempestade*. Enquanto o sistema está em *calmaria*, p_i pode relaxar o limite R_i^{min}. Os nós que voltam a se comportar como bons parceiros podem realizar novas parcerias com os nós que o classificaram como *peer* malicioso. Essa característica é importante para permitir a recuperação de participantes que possam ter sofrido com problemas de rede que os fizeram enviar dados corrompidos a seus parceiros.

Os resultados obtidos pelos autores, através de simulação, mostram que o mecanismo de reputação simples é mais efetivo que os propostos previamente e também identifica atacantes dissimulados, ou seja, que em intervalos de tempo agem como bons nós. A abordagem proposta alcança um desempenho quase duas vezes melhor do que os sistemas

de reputação tradicional.

O mecanismo de reputação proposto por Vieira *et al.* foi incluído no sistema P2P de vídeo ao vivo implementado nessa dissertação, com o intuito de combater ataques de poluição. Os resultados obtidos através de execuções nas máquinas do PlanetLab confirmaram a eficiência do mecanismo no combate a ataques de poluição isolados. Porém, este mecanismo se mostrou ineficiente nos casos onde os poluidores fazem *whitewashing*. Assim, foram implementadas modificações nesse mecanismo de reputação para o tornar resistente também a *whitewashing*.

3.2 Proposta de Seibert

Os autores em [Seibert, 2010] propõem um mecanismo baseado em reputação que leva em conta a experiência individual de cada participante e o depoimento global do sistema para combater *ataques de descarte de dados* em sistemas P2P de transmissão de vídeo ao vivo. Os objetivos dos autores são: 1) limitar o impacto dos ataques, permitindo que os nós encontrem rapidamente parceiros confiáveis para trocar dados, isolando os participantes maliciosos; 2) limitar a sobrecarga do mecanismo de defesa, fazendo com que esse mecanismo não interfira no desempenho do sistema quando este não estiver sofrendo ataques; e 3) proteger a origem e os demais participantes do sistema.

Para proteger os participantes do sistema, os autores propõem um sistema de reputação descentralizado que funciona da seguinte maneira: cada participante p_i do sistema calcula a reputação de cada um dos seus parceiros p_j baseados na quantidade de dados recebida deles. Assim, nós que entregam pouca quantidade de dados terão uma reputação menor. O objetivo é capturar a degradação no desempenho do sistema, bem como ataques de descarte de dados. Essa reputação, por si só, não é suficiente para banir um participante p_j da lista de vizinhos. Quando a reputação de p_j fica abaixo de um limite estipulado, este é marcado como suspeito. Para confirmar essa suspeita, o mecanismo proposto utiliza, adicionalmente, um outro valor, calculado de acordo com o depoimento da rede. Este valor é proporcional à quantidade de vezes que p_j aparece como parceiro dos parceiros de p_i. Caso p_j também seja parceiro de determinado número de parceiros de p_i, ele será marcado como malicioso e poderá ser removido das parcerias de p_i.

Os autores afirmam que o mecanismo de reputação proposto torna o sistema resis-

tente à prática de *whitewashing*. Primeiro, porque todo nó recém-chegado é marcado como suspeito. Assim, um nó não terá acesso a todos os recursos do sistema até que contribua e aumente sua reputação. Segundo porque, de acordo com os autores, os nós são identificados por seu endereço IP e não podem controlar mais do que alguns IPs diferentes.

Apesar das afirmações dos autores, o mecanismo proposto pode ter algumas dificuldades para combater apropriadamente a prática de *whitewashing*. Segundo Douceur [Douceur, 2002], sem uma entidade centralizada de autenticação, um participante da rede pode trocar sua identidade (endereço IP) quantas vezes desejar, exceto quando se assume o uso de recursos, como largura de banda e processamento, e coordenações extremas e irrealistas entre os participantes da rede para validar simultaneamente todas as identidades presentes. Além disso, Feldman *et al.* [Feldman, 2006] mostra que abordagens que impõe restrições aos nós recém-chegados no sistema podem causar degradação do desempenho do mesmo quando há alta dinamicidade dos participantes.

Diferente da proposta de Seibert *et al.* para combater a prática de *whitewashing*, o mecanismo de reputação proposto nesta dissertação não pune todos os *peers* recém-chegados. Ao invés disso, inicia a reputação de todo nó recém-chegado a um valor mínimo que ainda permite a troca de dados com os demais participantes do sistema. Assim, qualquer tentativa de poluição será rapidamente identificada e o poluidor será punido. Porém, se o *peer* recém-chegado se comportar de maneira altruísta, este terá acesso aos recursos do sistema.

4 Combate a Ataques de Poluição e *Whitewashing* em Sistemas P2P de Transmissão de Vídeo ao Vivo

Este capítulo tem como objetivo apresentar de forma detalhada o mecanismo de reputação simplificado adotado nesse trabalho para combater ataques de poluição, bem como as modificações propostas para torná-lo eficaz contra a prática de *whitewashing*.

4.1 Mecanismo de Defesa baseado em Reputação

Este trabalho implementa, em um ambiente real, um modelo de reputação distribuído, simplificado, originalmente proposto e simulado em [Vieira, 2009]. Neste mecanismo de reputação distribuído, cada *peer* monitora a troca de dados com seus parceiros, permitindo aos mesmos associar um valor de reputação a cada um de seus parceiros. O objetivo do mecanismo de reputação proposto por Vieira *et al.* é permitir que cada *peer* identifique e isole os participantes que disseminem conteúdo poluído.

Em uma abordagem de reputação distribuída clássica, cada *peer* p_i associa uma nota a cada parceiro p_j. Essa nota é computada através da experiência individual entre p_i e seu parceiro p_j e pela opinião consensual da rede sobre este parceiro. A experiência individual de p_i com p_j mostra o comportamento de p_j nas trocas de dados com p_i. O depoimento da rede expressa o comportamento de p_j com os demais participantes do sistema. O participante p_i decide se continua sua parceria com p_j baseado no valor calculado da reputação. Caso a reputação de p_j com p_i ($R_i[p_j]$) seja menor que um limite determinado, p_i interrompe suas interações com p_j, A partir desse momento (interrupção da parceria), p_j é considerado poluidor [Vieira, 2008].

Entretanto, não é claro que explorar o depoimento da rede seja sempre eficiente. Um *peer* em um sistema de transmissão ao vivo em P2P tipicamente apresenta muitas interações com seus parceiros em curtos intervalos de tempo. Os interesses de parcerias de p_i podem mudar rapidamente. Assim, a opinião da rede sobre um determinado *peer* pode

convergir muito lentamente, se comparado a taxa de interações entre 2 *peers*. Mais ainda, caso um determinado nó p_i queira calcular a reputação de p_j e não tenha muitos parceiros em intersecção p_j, o depoimento da rede em relação a p_j pode não ser confiável.

O mecanismo de reputação simplificado, proposto em [Vieira, 2009], se baseia somente na experiência individual que cada *peer* tem em relação a seus parceiros. Neste trabalho, esse mecanismo será chamado *mecanismo de reputação simples*.

Por esse mecanismo, cada *peer* p_i periodicamente calcula a reputação de cada parceiro p_j ($R_i[p_j]$) de maneira individual, desconsiderando o depoimento da rede. Mais precisamente, periodicamente um *peer* p_i requisita r_{ij} *chunks* a p_j durante cada intervalo de tempo. O parceiro p_j pode prover n_{ji} respostas ruins para p_i (onde $0 \leq n_{ji} \leq r_{ij}$). Uma resposta é definida como *ruim* quando p_i é forçado a pedir o *chunk* novamente a outro parceiro. Essa resposta pode conter, por exemplo, um *chunk* poluído ou um pacote acidentalmente danificado. A razão n_{ji}/r_{ij} representa a qualidade da experiência de p_i em relação a p_j. Em outras palavras, representa o grau de poluição presente nos dados recebidos de p_j. Se a razão n_{ji}/r_{ij} tem valor acima de um limite máximo T_i^{max} definido em cada *peer*, ou seja, se o grau de poluição percebido por p_i nas trocas de dados com p_j for maior do que o aceitável, p_i diminui a reputação local de p_j. Caso contrário, a reputação local de p_j é aumentada.

Raramente, no contexto da aplicação P2P apresentada, podem ocorrer casos em que seja inadequado utilizar o valor de n_{ji}/r_{ij} para definir a punição ou não de um *peer*. Caso um *peer* p_k envie um único *byte* e este esteja poluído, p_k será considerado poluidor. Porém, um *peer* não será considerado poluidor se enviar 1000 *bytes* dos quais 300 *bytes* são poluídos. Apesar disso, essa abordagem é válida pois ela é bastante simples e, no caso dos experimentos realizados, não foram observadas discrepâncias como nesse exemplo citado.

A reputação que p_i associa a p_j é atualizada através da Equação 4.1. Caso o valor da reputação de p_j com p_i ($R_i[p_j]$) fique abaixo de um determinado limite R_{min}, p_i interrompe as trocas de dados com p_j e o remove de sua lista de parceiros.

$$R_i[p_j] = \begin{cases} max(0, R_i[p_j] - \alpha_{p_i} * (1 + n_{ji}/r_{ij})^{y_i}) & \text{se } n_{ji}/r_{ij} > \mathrm{T}_i^{max} \\ min(1, R_i[p_j] + \alpha_{g_i} * (1 - n_{ji}/r_{ij})) & \text{caso contrário,} \end{cases} \tag{4.1}$$

Os parâmetros α_{p_i} e α_{g_i} são os fatores de penalidade e gratificação, respectivamente. Na Equação 4.1, ao considerar $\alpha_{p_i} > \alpha_{g_i}$ e $1 < y \leq 2$, a pontuação de um parceiro

p_j é reduzida de maneira mais rápida. Enquanto a reputação de um bom parceiro é incrementada de forma linear, a reputação de um parceiro poluidor é reduzida de forma exponencial. Assim, o sistema se torna mais sensível a ataques e é capaz de identificar e penalizar os *peers* poluidores rapidamente.

Todos os *peers* recém-chegados no sistema recebem um valor inicial de reputação ($R_{inicial}$), com $0,7 \leq R_{inicial} \leq 1$. Cada participante p_i tem um limite de reputação mínima R_i^{min}, com $0 \leq R_i^{min} \leq 1$. Caso $R_i[p_j]$ seja menor que R_i^{min}, p_i remove p_j de sua lista de parceiros.

Considerando a Equação 4.1, o *mecanismo de reputação simples* não permite a recuperação de participantes classificados como poluidores. Caso p_j passe por problemas temporários em sua rede e, por isso, seja considerado poluidor (*falso poluidor*), p_j não terá mais oportunidade de trocar dados com p_i, ainda que p_j se torne um excelente parceiro para os demais participantes do sistema. Como consequência, a taxa total de transmissão na rede pode ser prejudicada.

Esse fato pode ser ainda mais agravado caso os parâmetros α_{p_i} e y_i não sejam escolhidos adequadamente. Quanto maior os valores de α_{p_i} e y_i, mais rápido o mecanismo de reputação banirá os poluidores. Porém, valores muito altos podem fazer com que o mecanismo identifique muitos falsos poluidores. Em [Vieira, 2012b] os autores verificam a sensibilidade do mecanismo de reputação simplificado com a variação dos parâmetros α_{p_i}, α_{g_i} e y_i. Veira *et al.* recomendam valores apropriados para que o mecanismo de reputação simplificado não identifique muitos falsos poluidores.

4.1.1 Recuperação de Falsos Poluidores

No modelo de reputação clássico, quando um *peer* p_j é considerado poluidor, o depoimento da rede poderia ajudar p_j a se redimir e, futuramente, trocar dados com p_i novamente. Nesse sentido, a fim de permitir a recuperação de participantes com a reputação simplificada, Vieira *et al.* [Vieira, 2009], também propõem um mecanismo que dinamicamente altera o limite mínimo de reputação R_i^{min}.

Para alterar R_i^{min}, cada *peer* p_i reage às condições da rede, realizando avaliações localmente. Mais detalhadamente, caso p_i infira que a rede está sob ataque, este aumenta o valor de R_i^{min}, penalizando mais rapidamente seus prováveis parceiros poluidores. Caso contrário, este diminui o valor de R_i^{min} para permitir a reabilitação das parcerias punidas

anteriormente.

Para mudança dinâmica do valor de R_i^{min}, são definidos dois estados para o sistema P2P: (1) *calmaria* ou (2) *tempestade*. Nessa dissertação, o mecanismo de recuperação de falsos positivos foi implementado como definido a seguir.

Na visão de p_i, o sistema está em *calmaria* se este percebe um nível de poluição (np) abaixo de um limite pré-definido, chamado *lcalm*. O nível de poluição (np) é definido pela razão entre o número de *chunks* poluídos recebidos ($\sum_{\substack{i,j=1 \\ i \neq j}}^{n} n_{ji}$) e o número total de *chunks* recebidos ($\sum_{\substack{i,j=1 \\ i \neq j}}^{n} r_{ij}$) por p_i, ou seja, $np = (\sum_{\substack{i,j=1 \\ i \neq j}}^{n} n_{ji}) \ / \ (\sum_{\substack{i,j=1 \\ i \neq j}}^{n} r_{ij})$. Quando o sistema está em calmaria ($np \leq lcalm$), p_i reduz o valor da reputação mínima R_i^{min}, permitindo que os parceiros que tenham uma reputação menor que a R_i^{min} inicial não sejam mais considerados atacantes. O sistema se encontra no estado de *tempestade* quando p_i percebe $np > lcalm$. Nesse caso, p_i aumenta o valor de R_i^{min} para bloquear mais rapidamente os parceiros maliciosos.

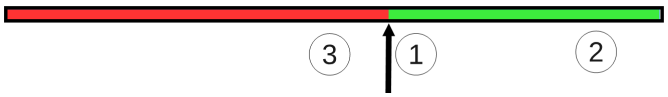
A cada intervalo de tempo, p_i verifica o estado do sistema e atualiza R_i^{min} de acordo com a Equação 4.2. Se o sistema está em *calmaria*, p_i diminui seu limite local R_i^{min} no fator de γ_{g_i}; caso contrário, R_i^{min} é aumentado no fator γ_{p_i}. Para que o sistema identifique rapidamente participantes poluidores foi considerado $\gamma_{p_i} > \gamma_{g_i}$. Assim, os valores de R_{min} são incrementados de maneira mais rápida quando o sistema está em tempestade. Os limites inferior (RT_i^{min}) da reputação mínima e superior (RT_i^{max}) são estabelecidos de tal maneira que $0 \leq RT_i^{min} \leq R_i^{min} \leq RT_i^{max} \leq 1$.

$$R_i^{min} = \begin{cases} max(RT_i^{max}, R_i^{min} + \gamma_{p_i}), & \text{se o sistema está no } \textit{estado de tempestade} \\ min(RT_i^{min}, R_i^{min} - \gamma_{g_i}), & \text{se o sistema está no } \textit{estado de calmaria} \end{cases} \tag{4.2}$$

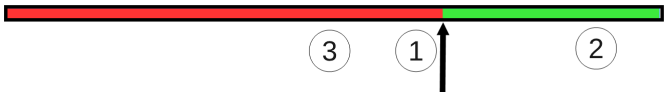
O estado do sistema é definido na perspectiva local de cada *peer*, baseado somente nas experiências obtidas em relação a seus parceiros: caso p_i receba uma quantidade de dados poluídos de seus parceiros, a sua visão local é de que o sistema está sofrendo ataque de poluição. A Figura 4.1 ilustra como p_i reage a mudanças de estado do sistema: a barra representa uma escala de reputação de baixos para altos valores (da esquerda para direita). A seta representa o valor atual de R_i^{min}, sendo os números $1, 2$ e 3 , os valores das reputações atuais de três parceiros p_1, p_2 e p_3, na visão de p_i.

Inicialmente, na Figura 4.1-a), todos os parceiros de p_i estão localizados no lado direito

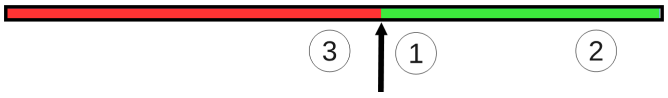
(a) Estado inicial do *peer* p_i



(b) *Tempestade*



(c) *Peers* maliciosos continuam a atacar



(d) *Calmaria*

Figura 4.1: Atualização do limite da reputação mínima de p_i. A seta indica o valor de R_i^{min}. 1, 2 e 3 são parceiros de p_i e suas posições na barra indicam suas reputações atuais na visão de p_i).

da seta, indicando que suas reputações estão acima do limite mínimo. No momento que p_i recebe uma taxa de *chunks* poluídos, o estado do sistema é modificado para *tempestade*, e seu limite mínimo de reputação é aumentado, conforme mostrado na Figura 4.1-b). O valor da reputação de p_3 está abaixo do mínimo e assim, p_i classifica p_3 como poluidor, removendo-o de sua lista de parceiros. Nos intervalos consecutivos, p_i continua recebendo *chunks* poluídos e, novamente, aumenta o valor de R_i^{min}, conforme mostrado na Figura 4.1-c). Após esta nova modificação, p_1 também é classificado como poluidor.

Após duas mudanças consecutivas no valor de R_i^{min}, p_i não recebe *chunks* poluídos, mudando a sua visão de estado do sistema para *calmaria*, diminuindo R_i^{min}. O novo valor de R_i^{min} permite p_i aceitar p_1 como parceiro novamente. Note que as mudanças em R_i^{min}, de acordo com a Equação 4.2, são realizados em paralelo às mudanças na reputação local de cada parceiro, definida na Equação 4.1.

A Tabela 4.1 apresenta um resumo de todos os elementos presentes nos mecanismos de reputação apresentados.

Parâmetro	Descrição
p_i	Participante i do sistema
R_i	Lista de reputações dos parceiros de p_i
$R_i[p_j]$	Reputação do participante p_j com p_i
$R_{inicial}$	Reputação inicial
R_{min}	Reputação Mínima
r_{ij}	Total de *chunks* requisitados em um intervalo de tempo
n_{ji}	Total de *chunks* poluídos recebidos em um intervalo de tempo
T_i^{max}	Limite máximo para a razão n/r
α_p	Fator de Penalidade
α_g	Fator de Gratificação
y	Fator para acelerar a penalidade
γ_{g_i}	Fator de redução da R_i^{min}
γ_{p_i}	Fator de aumento da R_i^{min}
RT_i^{min}	Valor mínimo que R_i^{min} pode assumir
RT_i^{max}	Valor máximo que R_i^{min} pode assumir
$\sum_{\substack{i,j=1 \\ i \neq j}}^{N} n_{ji}$	número *chunks* poluídos recebidos
$\sum_{\substack{i,j=1 \\ i \neq j}}^{N} r_{ji}$	número *chunk* recebidos.
np	nível de poluição percebido no sistema
$lcalm$	Limite máximo de $lcalm$ para manter o estado de calmaria

Tabela 4.1: Resumo dos elementos do modelo de reputação.

4.2 Mecanismo de Reputação Modificado

Os poluidores podem, frequentemente, sair do sistema e voltar novamente com uma nova identidade. Um *peer*, ao realizar essa constante troca de identidade, conhecida como *whitewashing*, consegue enganar o sistema de reputação. Ataques como estes podem causar grandes danos à qualidade do sistema P2P. Assim, nessa dissertação é proposto um *mecanismo de reputação modificado* (assim denominado para critérios de comparação) mais resistente a ataques de poluição combinados com *whitewashing*.

Nesse sentido, pelo *mecanismo de reputação modificado*, os participantes recém-chegados ao sistema recebem um baixo valor de reputação inicial. O valor estabelecido para a reputação inicial é próximo ao valor limite para que troca de *chunks* aconteça, ou seja $R_i[p_j] \tilde{=} R_i^{min}$. Assim, qualquer tentativa de poluição fará com que a reputação de p_j com p_i ($R_i[p_j]$) fique abaixo da reputação mínima (R_i^{min}). Então p_j será removido da lista de parceiros de p_i. Assim, o sistema é capaz de reagir mais rapidamente a esse tipo de

ataque, desencorajando os participantes poluidores a fazer *whitewashing*.

Vale ressaltar que, apesar da diminuição no valor da reputação mínima, os *peers* recém-chegados podem trocar dados normalmente com todos os participantes do sistema. Ou seja, estes *peers* não receberão qualquer tipo de restrição caso tenham um bom comportamento. Essa característica é importante para evitar que o mecanismo de reputação afete o bom desempenho do sistema, mesmo em casos onde ocorre alta dinamicidade de *peers*.

Por outro lado, o problema da identificação de participantes falsos positivos pode ser agravado. Com o valor da reputação reduzido tão próximo à R_i^{min}, os participantes que tiverem problemas de rede, mesmo que por um período de tempo curto, podem ser considerados poluidores. Isto pode gerar uma subutilização de recursos disponíveis na rede. Porém, como o *mecanismo modificado* utiliza o mesmo valor de R_i^{min} que o *mecanismo simples*, quando a reputação de um participante p_k ficar menor que R_i^{min}, ele será tratado da mesma forma pelos 2 mecanismos de reputação, ou seja, será banido do sistema. Porém, se o sistema alcançar o estado de calmaria, os valores de R_i^{min} serão reduzidos e p_k poderá ser reintegrado ao sistema. Assim, o esquema de recuperação de participantes falsos positivos pode ser utilizado no mecanismo modificado para reduzir este problema. Não há certeza se a prática de *whitewashing* permitirá que o sistema alcance o estado de calmaria. Portanto, um melhor estudo desse caso particular deve ser tratado em trabalhos futuros.

Para não prejudicar *bons peers* que deixem o sistema e voltem novamente, é proposto que todos os participantes mantenham um pequeno histórico de suas parcerias. Assim, tão logo um *peer bom*, p_j, que saiu do sistema, volte, este poderá ser pontuado com o seu valor de reputação anterior à saída. Nesse caso, p_j receberá um valor de reputação inicial mais alto que a reputação mínima. Para incentivar o bom comportamento os mecanismos de defesa baseados em reputação, geralmente, permitem que participantes que possuem uma reputação mais alta tenham acesso a mais recursos do sistema. Por isso, manter mais alta a reputação de bons participantes é uma característica importante.

Estas simples modificações tornam o mecanismo de reputação mais robusto, auxiliando no combate ao ataque de poluição combinado com *whitewashing*. Assim, os *peers* não poluidores podem receber vídeo ao vivo de qualidade. A versão modificada do mecanismo de reputação considera todos os *peers* recém-chegados no sistema como bons *peers*, enquanto tenta combater o comportamento malicioso o mais rapidamente possível.

5 Avaliações

Neste capítulo são apresentados detalhes da implementação de um protótipo de aplicação P2P para transmissão de vídeo ao vivo que foi executado no PlanetLab, a configuração do ambiente real de experimentação e a metodologia para captura e análise de dados.

5.1 Protótipo de Aplicação P2P para Transmissão de Vídeo ao Vivo

Para a análise dos mecanismos de reputação propostos nesse trabalho, foi desenvolvido um protótipo de aplicação P2P para transmissão de vídeo ao vivo baseado na topologia em malha. O protótipo de aplicação foi executado no PlanetLab, que disponibiliza um ambiente real de testes para aplicativos executados em redes de computadores.

O protótipo de aplicação desenvolvido nessa dissertação define o comportamento do *peer* origem do vídeo e dos demais participantes do sistema, incluindo os participantes maliciosos que fazem ataques de poluição e *whitewashing*. A construção e manutenção da topologia em malha são feitas por um nó especial central que regula a entrada e saída de participantes no sistema, chamado *bootstrap*. Para cada canal disponível no sistema, o *bootstrap* guarda uma lista de nós conectados no canal, qual participante no sistema é a origem desse canal e o *chunk* produzido mais recentemente.

Quando um novo participante chega no sistema, inicialmente ele deve contactar o *bootstrap* que retorna uma lista contendo alguns participantes que estão interessados no mesmo conteúdo. Ao receber esta lista, o *peer* recém-chegado tenta estabelecer conexões com alguns participantes para formar sua lista de parceiros, chamada de lista de *peers* ativos. Essa lista foi implementada como um vetor associativo onde cada entrada possui um endereço IP e um identificador do *peer*, assim como a porta utilizada por este *peer* para troca de dados. Com a lista de *peers* ativos completa, o *peer* recém-chegado pode começar a receber e enviar dados de vídeo. Periodicamente, cada participante solicita uma nova lista de parceiros para manter sua lista atualizada.

Qualquer participante do sistema tem a liberdade de criar um canal no *bootstrap* e transmitir vídeos pela rede. Para isso, basta que o participante anuncie sua intenção de

criar um canal ao *bootstrap*. O conteúdo de vídeo gerado pelo nó origem é dividido em *chunks* antes de ser difundido pela rede. Cada *chunk* possui um número de sequência que irá identificá-lo pela rede. Para manter atualizadas as listas de participantes que estão assistindo a um canal, cada participante envia periodicamente uma mensagem do tipo *ping* para o *bootstrap*. O nó origem do vídeo também envia mensagens ao *bootstrap* para mantê-lo atualizado sobre o último identificador de *chunk* gerado para determinado canal. Essas mensagens são utilizadas para que um participante possa avisar que ainda está ativo no sistema (*keep-alive*). Sendo assim, caso um participante não envie essa mensagem dentro de 10 segundos, o *bootstrap* entende que tal participante deixou o sistema.

Além de enviar mensagens do tipo *ping* para o *bootstrap*, cada participante também envia essas mensagens para seus parceiros. A diferença é que essas mensagens contêm um mapa de *chunks* onde cada posição determina a presença ou não de um determinado *chunk*, além de indicar o *chunk* mais novo presente no *buffer* do *peer*. Se algum *peer* fica mais que alguns segundos sem receber essa mensagem de um de seus parceiros, o *peer* remove essa parceiro de sua lista.

Cada *peer* p_i do sistema mantém um *buffer* B_i para armazenar *chunks* de vídeo antes de serem executados ou compartilhados. O tamanho do *buffer* é delimitado em cada *peer* p_i. Posições disponíveis no *buffer* são inicializadas como vazias. Os participantes do sistema mapeiam seus respectivos *buffers*, criando um mapa de *chunks*, para posterior sinalização de dados disponíveis e desejados.

O *buffer* B_i é implementado por uma estrutura do tipo lista circular, onde a posição a ser consumida é $B_i[0]$, e a posição mais recente a ser preenchida é $B_i[s]$ (s é a última posição de um *buffer*). A cada intervalo de tempo a aplicação consome o dado menos recente no *buffer* e o descarta. Além disso, a aplicação utiliza um mecanismo de janela deslizante (J_i). Esta janela é utilizada para manter uma visualização sem interrupções. Para tal, a aplicação deve obter alguns dados à frente do momento de exibição. Durante a exibição do vídeo, o nó p_i verifica quais *chunks* devem ser consumidos em um futuro próximo e os obtêm entre os seus parceiros.

A Figura 5.1 exemplifica o funcionamento do *buffer* implementado. A cada intervalo de tempo um participante tenta obter o *chunk* mais recentemente criado ($J_i[k]$). Após consumir o *chunk* mais à esquerda da janela de interesse ($J_i[0]$) o participante desloca a janela para a direita, e, novamente, tenta obter o *chunk* mais recentemente criado.

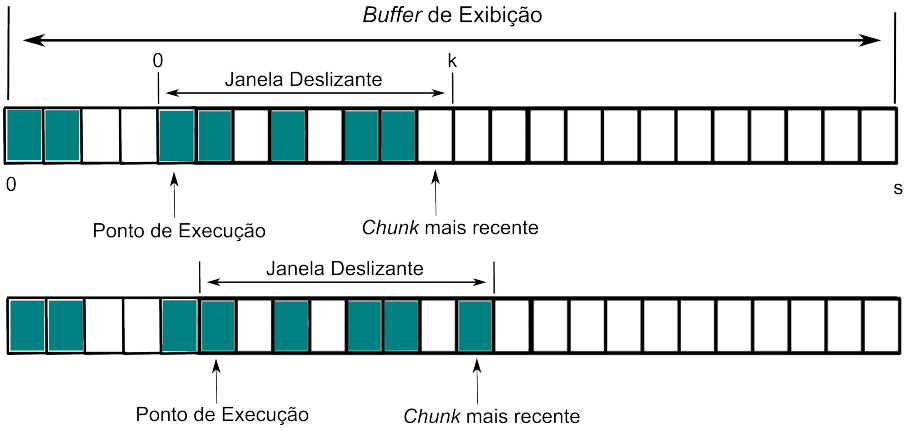


Figura 5.1: Exemplo de Janela Deslizante.

A origem anuncia a disponibilidade de novos *chunks* utilizando as mensagens contendo mapa de *chunks*. As mensagens são reproduzidas por toda a rede e, assim, o novo *chunk* pode ser disseminado. Nesse trabalho, é utilizada a estratégia de requisição de *chunks* *Mais Raro Primeiro* para evitar latência na exibição do conteúdo, mantendo as características ao vivo do sistema. Em poucos milissegundos, cada participante p_i compara seus próprios mapas de *chunks* com os de seus parceiros. Ao verificar a existência de um novo *chunk*, p_i seleciona um parceiro aleatoriamente e o adiciona a uma lista de requisições sinalizando a intenção de solicitar o *chunk* desejado a este parceiro. No momento em que recebe o *chunk* p_i o remove da lista de pedidos.

No protótipo de aplicação P2P de transmissão de vídeo ao vivo são implementados os ataques de poluição e *whitewashing*, bem como os mecanismos de reputação para combater tais ataques. Um *chunk* poluído é identificado por um *bit* no cabeçalho do pacote que indica se aquele *chunk* é poluído ou não. Desta forma, é possível simular um mecanismo de verificação de dados de maneira mais simples, que permita diferenciar os *chunks* enquanto trafegam pela rede. Entretanto, qualquer esquema de verificação de dados (e.g., assinatura baseada em *hash* proposto em [Haridasan, 2007]) poderia ser usado para identificar automaticamente *chunks* poluídos no sistema. Porém, como esse trabalho foca somente na sobrecarga causada na rede devido a retransmissões de *chunks*, esses tipos de técnicas não foram implementadas.

Durante um ataque, os poluidores anunciam um falso mapa de *chunks* completo, forjando possuir todo os *chunks*. Essa característica faz com que o participante poluidor atraia mais requisições, aumentando os efeitos do ataque. Todos os *chunks* enviados por um poluidor são, de fato, alterados.

O mecanismo de reputação foi implementado de acordo com a seção 4.1. A cada 15 segundos, todos os participantes que não são poluidores calculam a reputação de todos os *peers* p_j que estão na sua lista de *peers* ativos de maneira individual. Para isso, são mantidas duas listas, implementadas como vetores associativos, que guardam a quantidade de requisições feitas a cada parceiro e a quantidade de *chunks* poluídos recebidos de cada um. A razão entre estes valores mostra a experiência de p_i com p_j, mais precisamente, esta razão é o parâmetro n_{ji}/r_{ij} definido na seção 4.1. Uma outra lista guarda os valores da reputação de cada parceiro p_j. Caso a reputação de p_j fique abaixo do limite definido, p_j é removido da lista de *peers ativos* e adicionado a uma outra lista que contem os participantes considerados poluidores por p_i, chamada *lista de poluidores*. Ao receber do *bootstrap* uma nova lista de parceiros, p_i compara os *peers* contidos nessa lista com os *peers* na *lista de poluidores*. Assim, p_i é capaz de bloquear os *peers* que compartilharam dados poluídos anteriormente.

A escolha do tempo de cálculo da reputação, assim como dos valores dos parâmetros das Equações 4.1 e 4.2 foi feita através de experimentação, com base nos valores apresentados em [Vieira, 2012b] . Os valores escolhidos permitem que o sistema reaja melhor aos ataques, ao mesmo tempo em que não prejudicam o bom desempenho do sistema durante o cálculo da reputação e registro de *logs* dos *peers*.

Para avaliar o mecanismo de recuperação de participantes falsos poluidores, foi escolhido um *peer* p_k em particular no sistema. Para simular problemas de rede, p_k foi configurado para enviar dados corrompidos com uma probabilidade de 10% a cada requisição recebida. Este valor corresponde ao pior caso no envio de *chunks* corrompidos [Caceres, 1999]. Cada participante p_i do sistema ($p_i \neq p_k$) guarda o momento em que p_k foi considerado poluidor e o momento em que foi recuperado. Assim, é possível verificar qual a porcentagem de participantes que consideraram p_k como poluidor e o tempo médio para que ele fosse recuperado.

A cada minuto, todos os participantes que não são poluidores armazenam localmente informações sobre sua sessão de vídeo, contendo valores de poluição percebidos, sobrecarga, perdas de *chunks* e atrasos de cada *chunk*. Ao final da simulação, o nó origem executa um *script* para buscar todos os arquivos de *log* nos demais participantes e gera os resultados finais de cada simulação. Foram desenvolvidos outros programas auxiliares que analisam cada arquivo de *log* recebido, eliminando entradas inválidas e separando os

dados válidos por minuto. Esses programas auxiliares também realizam os cálculos das médias para cada métrica utilizada.

5.2 Metodologia e Ambiente de Experimentação

Os mecanismos de reputação propostos (*simples* e *modificado*) foram avaliados em um ambiente real, configurado no PlanetLab (www.planet-lab.org).

O *peer* servidor do vídeo foi instalado numa máquina na rede interna do campus da UFJF, transmitindo 30 minutos de vídeo a uma taxa de 120 kbps que é executado continuamente durante os experimentos. Ou seja, quando a exibição do vídeo termina, o *peer* origem continua a exibição do mesmo a partir do início, mantendo a numeração dos *chunks* como se o vídeo não tivesse terminado. Assim, se o último *chunk* criado antes do fim do vídeo tiver um valor de identificação x, ao exibir novamente o vídeo, o primeiro *chunk* receberá identificação $x + 1$.

Foram utilizados 133 nós PlanetLab como *peers* do sistema P2P de transmissão de vídeo ao vivo. Este é um número de usuários comum em canais como os do SopCast de transmissão de vídeo ao vivo em P2P [Vieira, 2013]. Desses *peers*, 120 são classificados como *peers bons* e 13 como poluidores (10% do total), que agem independentemente, ou seja, sem fazer conluio. Vieira *et al.*, em [Vieira, 2012b], mostram, através de simulação, que com apenas 1% de poluidores os efeitos de ataques de poluição são reduzidos, alcançando 20% de sobrecarga nos momentos de pico. Porém, com 10% de poluidores a sobrecarga gerada no sistema ultrapassa 100%. Devido a esse comportamento e para um melhor estudo dos efeitos causados por ataques de poluição, nessa dissertação 10% dos participantes foram considerados poluidores. Não foram impostas quaisquer restrições em relação a processamento e largura de banda tanto ao servidor quanto aos *peers*. Cada máquina foi utilizada com a capacidade total disponível.

Todos os nós PlanetLab são sincronizados utilizando o NTP (www.ntp.org). Como o PlanetLab configura o horário de todas as máquinas de forma idêntica, foi possível utilizar o *cron* [1] do Linux para programar a entrada de todos os nós no sistema ao mesmo tempo. O horário da máquina que hospeda o *peer* origem também foi sincronizado com os horários das máquinas do PlanetLab.

[1] http://manpages.ubuntu.com/manpages/hardy/man8/cron.8.html

No momento em que cria um *chunk*, o nó origem registra o horário de criação desse *chunk* e cria um identificador que o diferencia unicamente em todo sistema. Em cada um dos demais nós, quando um *chunk* é recebido, o nó salva horário de recebimento e o identificador desse *chunk* em um arquivo de *logs*. Assim, após recolher todos os arquivos de *log* da rede, o nó origem pode calcular a diferença entre o horário de geração de cada *chunk* e recebimento em cada *peer* do sistema. Assim, é possível obter o atraso médio entre a geração e recebimento dos *chunks* no sistema.

Durante os experimentos, todos os *peers* permanecem ativos até o fim da transmissão. Cada *peer* se conecta no máximo a 18 parceiros. Este número é próximo a uma observação realizada no SopCast, considerando como parceria a troca de *chunks* de vídeo [Vieira, 2013]. Se um de seus parceiros falha ou deixa o sistema, um *peer* deve requisitar novos parceiros candidatos ao *bootstrap*.

Os poluidores atacam desde o momento em que entram no sistema até o final da transmissão. Durante o ataque, os *peers* poluidores anunciam um mapa de *chunks* completo, forjando ter todos os dados possíveis. Cada participante p_i do sistema, ao receber um *chunk* poluído irá descartá-lo e pedi-lo novamente a algum outro *peer* entre seus vizinhos. Caso esta nova requisição não seja atendida antes de 20 segundos, que é um valor de atraso comumente observado em aplicações do tipo SopCast, p_i considera o *chunk* como perdido e não o solicita novamente. Essa restrição de tempo de espera por um *chunk* existe devido às restrições de tempo que sistemas de transmissão de vídeo ao vivo exigem. Caso um *peer* fique esperando a resposta de uma solicitação por muito tempo, ele pode assistir ao vídeo com um atraso significante que iria de encontro às características ao vivo do sistema.

No sistema proposto, cada participante é identificado pelo seu endereço IP concatenado a um valor gerado aleatoriamente, da forma *IP:valor*. Assim, para simular a prática de *whitewashing*, o participante malicioso simplesmente seleciona outro valor aleatório para formar seu identificador. Desse modo, o *whitewasher* será tratado como um novo participante no sistema. Um participante poluidor faz *whitewashing* a cada 215 segundos.

Para o propósito de análise do sistema, foram considerados dois cenários diferentes. No primeiro cenário, os participantes maliciosos realizam ataques de poluição isolados. Este cenário é estudado com duas abordagens. Na primeira, é utilizado apenas um mecanismo de verificação de dados para combater ataques de poluição. O objetivo é mostrar que a

utilização isolada desse mecanismo não é efetiva no combate a esses ataques. Na segunda abordagem, os poluidores atacam o sistema seguindo o mesmo comportamento anterior, porém, agora é utilizado o mecanismo de reputação apresentado na seção 4.1 para punir os participantes poluidores. O objetivo dessa abordagem é avaliar a eficiência do *mecanismo de reputação simples* ao combater ataques de poluição isolados.

No segundo cenário, os poluidores atacam o sistema desde o início da execução, porém eles saem e entram novamente no sistema, assumindo uma nova identidade (ataque de *whitewashing*). Este cenário é estudado com outras duas abordagens. A primeira utiliza o *mecanismo de reputação simples* para combater ataques de poluição combinados com *whitewashing*. O objetivo dessa primeira abordagem é mostrar que o *mecanismo de reputação simples* não é capaz de reduzir os efeitos desse ataque combinado de maneira satisfatória. Na segunda abordagem, é incluído o *mecanismo de reputação modificado*. Nesse caso o *mecanismo de reputação modificado* é utilizado para combater ataques de poluição combinado com *whitewashing*. O objetivo da segunda abordagem é avaliar a eficiência do *mecanismo de reputação modificado* no combate a ataques de poluição combinado com *whitewashing*.

5.3 Resultados

Nesta seção são apresentados os resultados da análise do protótipo P2P de transmissão de vídeo ao vivo obtidos através dos experimentos realizados no ambiente real configurado no PlanetLab.

Os principais objetivos da análise realizada são os seguintes: com os experimentos espera-se mostrar que utilizar somente um mecanismo de verificação de dados, sem o auxílio de algum outro mecanismo para punição de participantes, não é uma medida efetiva para garantir o bom desempenho do sistema. Sendo assim, é necessário um esquema, como o de reputação, para banir maus participantes. Além disso, é avaliado o impacto causado por ataques de poluição isolados e combinados com a prática de *whitewashing* em um sistema P2P de transmissão de vídeo ao vivo. A ideia é mostrar através de algumas métricas, que esses ataques são prejudiciais ao sistema. Adicionalmente, mostrar que o mecanismo de reputação simples implementado pode ser eficiente ao identificar e banir participantes maliciosos, sem causar uma sobrecarga adicional ao sistema para

isso. E, por fim, mostrar que as alterações propostas no mecanismo de reputação simples para combater *whitewashing* podem reduzir consideravelmente o impacto desse tipo de comportamento nas redes P2P de transmissão de vídeo ao vivo.

5.3.1 Métricas

Para analisar o desempenho do sistema, foram utilizadas quatro métricas: a) taxa de *chunks* poluídos recebidos ou *taxa de erro*; b) a sobrecarga no sistema; c) taxa de *chunks* perdidos (taxa de perda); e d) atraso na exibição de *chunks*. Estas métricas permitem capturar o dano causado ao sistema por ataques de poluição isolados e combinados com *whitewashing* e também inferir a qualidade de experiência dos usuários. Todas as métricas foram calculadas através da média dos valores reportados por cada participante do sistema P2P de transmissão de vídeo ao vivo. Todos os resultados apresentam o valor médio das medidas utilizadas e o intervalo de confiança de 95% da realização de 5 rodadas de difusão de um vídeo através do protótipo da aplicação P2P de vídeo ao vivo. Assim, define-se:

1. **Taxa de Erro:** essa métrica é obtida através da razão entre o número de *chunks* poluídos recebidos ($\sum_{\substack{i,j=1 \\ i \neq j}}^{n} n_{ji}$) pelos participantes e o número total de *chunks* recebidos ($\sum_{\substack{i,j=1 \\ i \neq j}}^{n} r_{ij}$). Ou seja, a taxa de erros é a média entre as razões $\sum_{\substack{i,j=1 \\ i \neq j}}^{n} n_{ji} / \sum_{\substack{i,j=1 \\ i \neq j}}^{n} r_{ij}$ entre todos os participantes do sistema. Nesse trabalho, toda resposta de solicitação que contenha um *chunk* poluído é considerada como um erro na obtenção do *chunk*. Este erro obriga o participante a pedir o *chunk* novamente para outro parceiro de sua lista. Essa métrica demonstra a quantidade de respostas ruins recebidas pelos participantes, ou seja, a taxa de respostas contendo *chunks* poluídos ou corrompidos.

2. **Sobrecarga no sistema:** essa é a principal métrica utilizada nesse trabalho para medir o impacto causado por ataques de poluição e *whitewashing* na rede P2P de transmissão de vídeo ao vivo. A sobrecarga acontece a partir do momento em que um participante p_i recebe um *chunk* poluído de um parceiro p_j. A partir disso, p_i precisa solicitar aquele *chunk* novamente, dessa vez para outro parceiro $p_k \neq p_j$. Além da mensagem de requisição enviada, o próprio *chunk* terá de ser enviado novamente pela rede por p_k. Essas mensagens repetidas provocam subutilização da rede, comprometendo seriamente o desempenho do sistema P2P.

3. **Taxa de Perda:** um *chunk* requisitado é considerado como perdido caso não seja

recebido ou seja recebido após o período de 20 segundos contados a partir do momento em que foi solicitado. Assim, o participante não solicita mais esse *chunk*. Essa métrica permite avaliar a qualidade da exibição do vídeo nos usuários. Quanto maior a taxa de perdas, menor a quantidade de *chunks* bons os participantes estão recebendo. Como consequência, não será possível a exibição contínua do vídeo nos usuários, fazendo com que estes abandonem o sistema.

4. **Atraso na Exibição de *Chunks*:** o atraso percebido por um participante é calculado através da diferença entre o momento de recepção de um *chunk* por esse participante e o momento em que este *chunk* foi criado pelo *peer* origem. Esta métrica também é importante para avaliar a qualidade de exibição dos vídeos nos usuários. Este atraso acontece como consequência das outras métricas, como taxa de erros e sobrecarga. A cada erro de *chunk* o *peer* será obrigado a pedir este *chunk* novamente até obter uma resposta com um *chunk* que possa ser exibido. Cada requisição exige certo tempo entre o pedido e o recebimento do *chunk*. Este tempo gasto com novas requisições gera um atraso considerável na exibição do vídeo ao vivo. Este atraso é ainda mais agravado pela sobrecarga gerada na rede.

Os valores dos parâmetros utilizados em todas as simulações estão resumidos na tabela 5.1. Estes valores foram escolhidos com base no trabalho de Vieira *et al.* [Vieira, 2012b] que apresenta um estudo da variação dos valores de α_p, α_g e y, além de indicar quais valores ideais para esse tipo de sistema.

Parâmetro	Valor
Total de *peers*	133
Total de poluidores	13
Número de parceiros	18
y	2
α_p	0.07
α_g	0.05
T_i^{max}	0.4
$R_{inicial}$	0.7
R_{min}	0.4
lcalm	0.1; 0.2; 0.3

Tabela 5.1: Parâmetros usados nas execuções no PlanetLab.

5.3.2 Eficiência do Mecanismo de Reputação

5.3.2.1 Análise do Primeiro Cenário

Nessa seção são mostrados resultados comparativos entre duas abordagens. Na primeira, os participantes somente implementam um mecanismo de verificação de dados, sem nenhum outro mecanismo adicional e, na segunda, os participantes implementam o *mecanismo de reputação simples* para combater ataques de poluição.

As Figuras 5.2, 5.3 e 5.4 mostram os resultados obtidos nesse cenário, onde os poluidores não realizam o ataque de *whitewashing*. O mecanismo *verificação de dados*, representa o simples descarte de *chunks* reconhecidos como poluídos e o pedido de retransmissão dos mesmos. O mecanismo *reputação simples* refere-se ao mecanismo de reputação descrito na Seção 4.1.

A Figura 5.2 mostra a média da taxa de erros de *chunks* percebida em cada participante do sistema. Por essa figura é possível notar que a taxa de erros de *chunks* é de quase 100% em um sistema onde não há tentativa de isolar poluidores. Assim, no esquema de *verificação de dados*, os *peers* recebem uma grande quantidade de *chunks* poluídos. Apesar de este mecanismo evitar a exibição do *chunk* poluído e/ou corrompido ele não é capaz de garantir a obtenção de um bom *chunk*, ou seja, em condições de ser exibido. Utilizando somente mecanismo de verificação de dados, no pior caso, quase todo o vídeo é perdido. O *mecanismo de reputação simples* proposto diminui a taxa de erros para 6%. Este mecanismo é capaz de banir os participantes poluidores do sistema, permitindo que os *peers* voltem a receber bons *chunks*, bloqueando a difusão de conteúdo poluído no sistema.

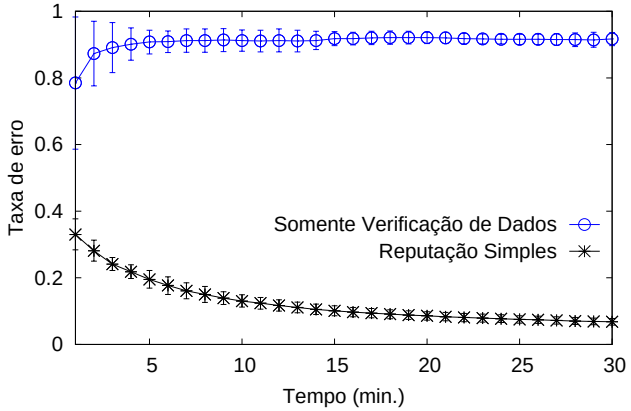


Figura 5.2: Taxa de erros de *chunks* no primeiro cenário, onde não há prática de *whitewashing*.

A taxa de erros tem um reflexo direto na sobrecarga gerada no sistema. Toda nova requisição de *chunk* gerada a partir do recebimento de um *chunk* poluído e/ou corrompido gera sobrecarga. A redução da taxa de erros obtida com o mecanismo de *reputação simples* fará com que a sobrecarga média percebida no sistema também seja reduzida.

A Figura 5.3 apresenta a sobrecarga no sistema. Em um sistema usando somente a abordagem de verificação e retransmissão de dados, a sobrecarga se estabiliza em torno de 230% após 15 minutos. Nesse caso, os *peers* são forçados a utilizar mais de três vezes a largura de banda necessária em um sistema sem poluidores para tentar receber bons *chunks*. Nesse caso, o mecanismo de verificação de dados, por não eliminar os poluidores e forçar os *peers* a pedir os dados novamente, acaba sendo o responsável por gerar toda a sobrecarga no sistema. Em contrapartida, o *mecanismo de reputação simples* reage rapidamente ao ataque de poluição. Este mecanismo é capaz de identificar e punir os poluidores e reduzir a sobrecarga no sistema desde o primeiro minuto de execução. Ao final, este mecanismo reduz a sobrecarga a valores abaixo de 5% no geral, evitando a subutilização da rede e, por consequência, permitindo a recuperação do sistema. Ao banir os poluidores, esse mecanismo de reputação permite que as novas requisições de *chunks*, quando necessárias, sejam respondidas com um *chunk* bom em 95% dos casos.

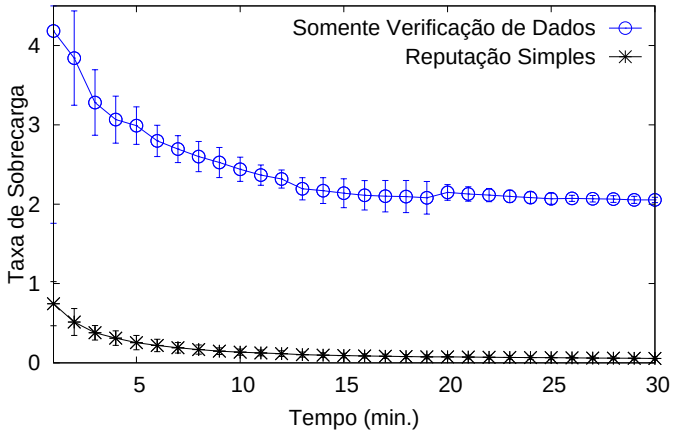


Figura 5.3: Sobrecarga no sistema em ataques de poluição sem a prática de *whitewashing*.

Finalmente, é apresentada a taxa de perda de *chunks* global do sistema. A Figura 5.4 confirma que ataques de poluição geram grandes danos em um sistema que usa somente a abordagem de verificação de dados. Neste caso, os *peers* descartam 82% dos *chunks* referentes às respostas das novas requisições feitas, seja por tê-los recebidos após o prazo de 20 segundos ou simplesmente por não ter obtido resposta para a solicitação feita. Como

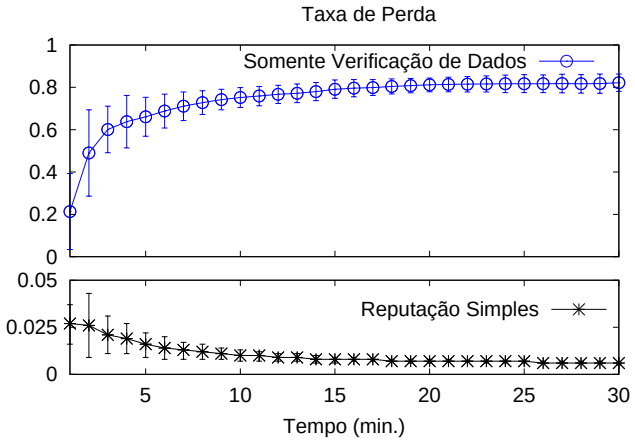


Figura 5.4: Taxa de perda em ataques de poluição sem a prática de *whitewashing*.

consequência, os *peers* não serão capazes de assistir continuamente ao vídeo, diminuindo a qualidade de experiência. Usando o *mecanismo de reputação simples*, a taxa de perda cai para um valor desprezível, abaixo de 0.7%. Assim, os participantes praticamente não perdem mais nenhum *chunk* criado pelo nó origem.

Analisando estas 3 métricas, percebe-se que o mecanismo de verificação de dados isolado não é efetivo no combate a ataques de poluição. Apesar de este mecanismo evitar a exibição de *chunks* poluídos e/ou corrompidos, ele não é capaz de eliminar os participantes poluidores e, consequentemente, não é capaz de eliminar os efeitos causados por ataques de poluição. Assim, os participantes do sistema ainda percebem grande quantidade de dados poluídos e ainda sofrem com a sobrecarga na rede, além de não ter garantias de exibição do vídeo correto em tempo hábil.

Estes resultados mostram que é necessário um mecanismo adicional capaz de punir e eliminar os participantes poluidores. Os resultados obtidos mostraram que o *mecanismo de reputação simples* é capaz de banir os poluidores do sistema, reduzindo consideravelmente os valores das 3 métricas citadas. Como o sistema foi simulado através de um protótipo de aplicação em um ambiente real, o *mecanismo de reputação simples* se mostrou como uma boa alternativa para o uso na prática em sistemas P2P para transmissão de vídeo ao vivo, quando estes sofrem com ataques de poluição isolados.

5.3.2.2 Análise do Segundo Cenário

Nesse cenário, onde todos os poluidores também fazem *whitewashing* para tentar enganar o sistema de reputação, são comparadas outras duas abordagens. Na primeira, o *meca-*

nismo de reputação simples é adotado para combater a prática de *whitewashing*. Assim, é possível ver que este mecanismo de reputação, apesar de conseguir reduzir os efeitos dos ataques de poluição combinados com *whitewashing*, não é efetivo no combate a este comportamento. Na segunda abordagem, os participantes utilizam o *mecanismo de reputação modificado* para combater a prática de *whitewashing*.

O objetivo desse cenário é mostrar o impacto que ataques de poluição combinados com *whitewashing* causam em um sistema P2P de transmissão de vídeo ao vivo, mesmo quando este implementa um mecanismo de reputação e verificação de dados. Mais ainda, é avaliada a efetividade do *mecanismo de reputação modificado* no combate a ataques de poluição combinados com *whitewashing*. Como é mostrado a seguir, esse mecanismo alcança melhores resultados se comparado ao *mecanismo de reputação simples*.

Para mostrar o impacto causado pela prática de *whitewashing* e a efetividade do *mecanismo de reputação modificado*, as Figuras 5.5, 5.6 e 5.7 apresentam os resultados obtidos nos casos onde os poluidores também fazem *whitewashing*. A Figura 5.5 mostra a taxa de erro nesses cenários. Os resultados mostram que, mesmo com o *mecanismo de reputação simples* ativado, a prática de *whitewashing* ainda causa uma alta taxa de erro de *chunks*, alcançando 70%. Assim, os participantes serão obrigados a pedir novamente a grande maioria dos *chunks* recebidos, menos de 1/3 dos *chunks* chegariam em condições de ser exibidos. Porém, o *mecanismo de reputação modificado* reduz a taxa de erro de *chunks* a 19%.

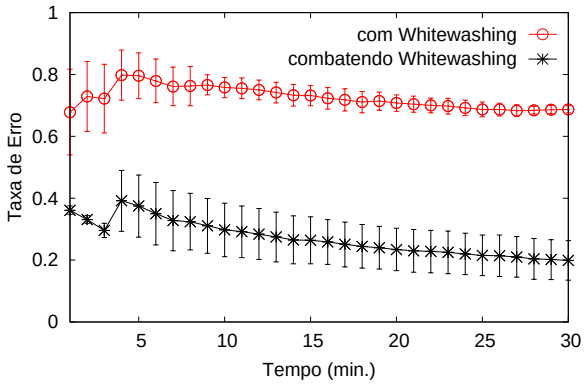


Figura 5.5: Taxa de erros de *chunks* com ataques de poluição combinado com a prática de *whitewashing*.

Apesar de a probabilidade de se obter um *chunk* poluído na primeira requisição não poder ser ignorada, o *mecanismo de reputação modificado* se mostrou mais eficaz que o *mecanismo de reputação simples*. Nesse caso, mais de 80% dos *chunks* recebidos poderão

ser exibidos. A redução nos valores da taxa de erro de *chunks* implica diretamente nos resultados das demais métricas. Essa redução possibilita uma diminuição considerável da sobrecarga na rede e, como consequência, a taxa de perdas também é reduzida.

A sobrecarga também apresenta uma diferença significante de comportamento comparado ao cenário onde os poluidores não fazem *whitewashing*. A Figura 5.6 mostra que, mesmo utilizando o *mecanismo de reputação simples*, os *peers* experimentam 112% de sobrecarga, ou seja, precisam de mais de duas vezes a largura de banda necessária em um sistema sem ataques. Este valor confirma o fato de que *whitewashing* é um comportamento indesejado em sistemas P2P de transmissão de vídeo ao vivo. O *mecanismo de reputação modificado* proposto reduziu a sobrecarga para 20%. Apesar desse valor de sobrecarga não desprezível, nota-se que a sobrecarga permanece baixa.

Quando os participantes poluidores voltam ao sistema com uma nova identidade, eles ainda têm um pequeno tempo para transmitir dados poluídos antes que o mecanismo de reputação seja capaz de identificá-los e bani-los novamente. Este pequeno intervalo de tempo, entre a nova entrada do poluidor e o reconhecimento por parte do mecanismo de reputação, impossibilita uma maior redução nos valores de sobrecarga apresentados pelo *mecanismo de reputação modificado*.

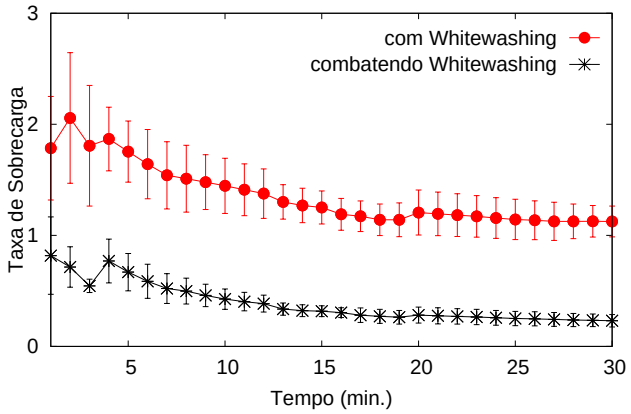


Figura 5.6: Sobrecarga no sistema com ataques de poluição combinado com a prática de *whitewashing*.

De acordo com a Figura 5.7, a taxa de perda confirma, mais uma vez, que a prática de *whitewashing* é um comportamento prejudicial ao sistema. Com o *mecanismo de reputação simples*, a taxa de perda é de cerca de 50%. Nesse caso, metade das solicitações de *chunks* feitas pelos *peers* seriam desperdiçadas, seja por solicitações por novos *chunks* ou por retransmissões devido a recebimento de *chunks* poluídos. Esse comportamento gera

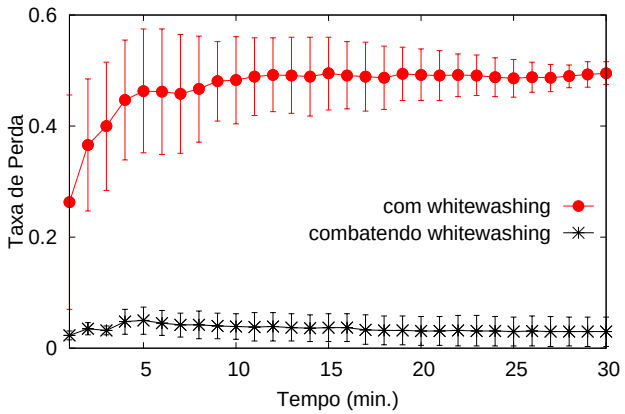


Figura 5.7: Taxa de perda com ataques de poluição combinado com a prática de *whitewashing*.

um grande desperdício dos recursos disponíveis no sistema. O *mecanismo de reputação modificado* se mostrou bastante eficiente, reduzindo a taxa de perdas para valores abaixo de 3%. Nesse caso, o sistema praticamente não sofre com perdas de *chunks*, isto torna este valor totalmente aceitável para sistemas P2P de transmissão de vídeo ao vivo.

Os resultados apresentados mostram que o mecanismo de reputação simples não é efetivo contra poluidores que também fazem *whitewashing*. Em contrapartida, o *mecanismo de reputação modificado* proposto reduziu os danos causados pelos ataques de poluição combinados com *whitewashing*, mostrando ser uma boa alternativa para combater estes ataques combinados em sistemas reais. Mesmo não eliminando totalmente os danos causados, a viabilidade do uso do *mecanismo de reputação modificado* na prática não pode ser subestimada. Primeiro porque as modificações propostas são simples e fáceis de implementar. Segundo, no esquema de reputação proposto não é necessário manter um custoso mecanismo centralizado de identificação de *peers* comumente usado para evitar a prática de *whitewashing*.

5.3.2.3 Recuperação dos Participantes

Uma característica importante do *mecanismo de reputação simples* implementado é a recuperação de bons *peers* que possam ter sido considerados poluidores devido ao funcionamento incorreto da rede. Para avaliar a robustez dos mecanismos propostos, um *peer* no sistema foi configurado para enviar 10% de seus *chunks* propositalmente como corrompidos (simulando problemas de rede). Para definir os estados de *calmaria* e *tempestade* foi usada a razão *chunks* poluídos/não-poluídos recebidos em cada *peer*.

Foram considerados três valores do parâmetro *lcalm*: 0.1, 0.2 e 0.3. Para $lcalm = 0.1$ 30.2% dos *peers* identificaram o falso poluidor, sendo que 82.8% destes recuperam o falso positivo em aproximadamente 110 segundos. Para o caso $lcalm = 0.2$ 28.9% dos *peers* identificaram o falso poluidor, sendo que 73% destes recuperam o falso positivo em um tempo médio de 97 segundos. Por último, para o caso em que $lcalm = 0.3$ 27.3% dos *peers* identificam o falso poluidor e 51% voltam a trocar informações. O tempo para recuperação foi, em média, de 72 segundos.

A seguinte discussão pode ser feita com base nos resultados apresentados. Para valores decrescentes do parâmetro *lcalm*, uma maior percentagem de *peers* identificam o falso positivo, bem como os recuperam. Para valores menores de *lcalm*, o período de *tempestade* é mais longo, assim o valor da reputação mínima R_i^{min} fica maior. Por consequência, os *peers* podem reagir melhor à poluição. Dessa maneira, poluidores reais são removidos mais rapidamente. Com os poluidores removidos, os *peers* vão atingir o estado de calmaria, que deve permanecer até o final da execução. A necessidade de um maior tempo para recuperação de falso positivos está relacionada à dinâmica da mudança de valores da reputação mínima. Por ficar mais tempo em estado de tempestade, o valor de reputação mínima nos *peers* fica maior. Por isso, ao atingir estado de calmaria, esse valor demora mais tempo para ser reduzido a um valor que permita a recuperação dos falsos positivos. Mesmo esse tempo sendo maior, mais falsos positivos serão recuperados, dado que o sistema terá banido praticamente todos os poluidores e assim, o sistema não voltará mais ao estado de tempestade.

Para visualizar o impacto em causado no *peer* p_j considerado como falso positivo, a Figura 5.8, mostra o total de *chunks* recebidos pelo mesmo durante toda a simulação. Em todos os casos, após 10 minutos iniciais, p_j recebe, em média, de 2900 a 3000 *chunks* a cada 30 segundos. Este total é o valor esperado de recebimento de *chunks* pelos *peers* que não agem como poluidores. Assim, mesmo p_j sendo considerado como poluidor por um tempo, este não terá problemas ao exibir o vídeo. Este resultado está intimamente ligado à rápida recuperação do mecanismo de reputação implementado.

Concluindo, a Figura 5.9 mostra o número de *chunks* recebidos pelo falso positivo p_j, no intervalo entre 15 minutos e 25 minutos. Este intervalo representa o melhor caso da recuperação de falso positivo devido ao mecanismo de reputação proposto. Neste intervalo, o sistema se encontra estável, e p_j recebe quase 100% de *chunks* sem poluição

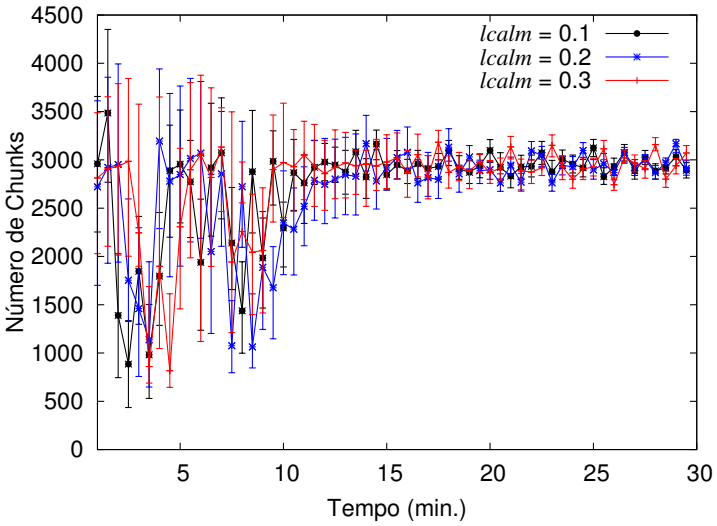


Figura 5.8: Número de *chunks* recebidos pelo falso positivo.

(taxa de *chunks* poluídos não chega a 1%).

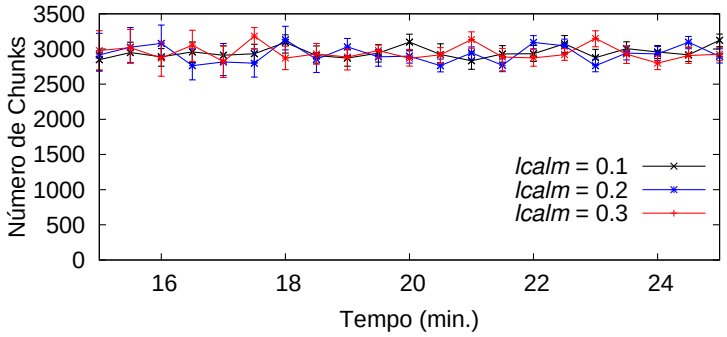


Figura 5.9: Número de *chunks* recebidos pelo falso positivo entre os 15 e 25 minutos da execução.

5.3.3 Impacto do Mecanismo de Reputação na Qualidade Oferecida aos Usuários

Apesar dos mecanismos propostos combaterem com eficiência o ataque de poluição combinado ou não com o ataque de *whitewashing*, os resultados da Seção 5.3.2 mostram que, mesmo com os mecanismos implementados, estes ataques ainda causam danos no sistema P2P. Desta forma, é necessário estudar a viabilidade prática do uso dos mesmos, no que tange a qualidade provida aos usuários do sistema. Para manter as características ao vivo desejadas nesse tipo de sistema é importante que o atraso na exibição do vídeo nos usuários seja pequeno. Além disso, é importante que não exista um atraso na exibição

entre os próprios participantes, ou seja, a exibição de um determinado *chunk* não deve acontecer em momentos muito distantes em cada participante.

Para analisar a qualidade de experiência dos usuários quando os mecanismos de reputação propostos são implementados, considera-se a métrica atraso de exibição do vídeo. A Figura 5.10 mostra o valor médio de 5 execuções dos testes (sob as mesmas condições) na abordagem onde somente a verificação de dados é implementada. O atraso médio, considerando todos os *peers* e todos os *chunks* é de mais de 2 minutos (132 segundos). Somando-se à taxa de erros, perdas e sobrecarga mostradas anteriormente pode-se afirmar que a qualidade de experiência dos usuários é muito ruim quando o sistema utiliza somente um mecanismo de verificação de dados. Quando um participante consegue obter um *chunk* não poluído, fato que é raro nesse caso, este já está com 2 minutos de atraso desde a sua criação.

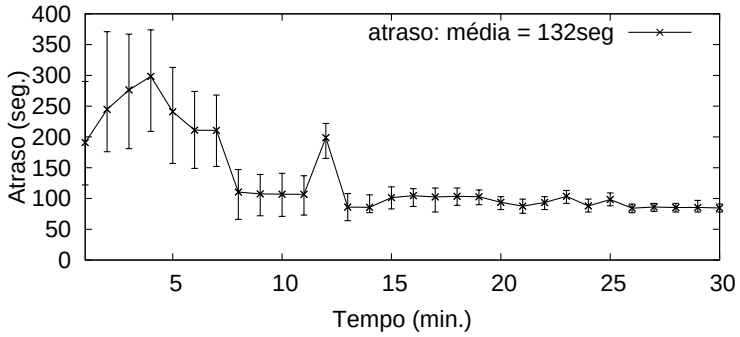


Figura 5.10: Atrasos - Verificação de Dados.

Em contrapartida, a Figura 5.11 mostra que o *mecanismo de reputação simples* reduz o atraso médio a apenas 6 segundos. Este valor desprezível de atraso confirma, mais uma vez, a eficiência do *mecanismo de reputação simples* no combate a ataques de poluição isolados em sistemas reais de transmissão de vídeo ao vivo em P2P.

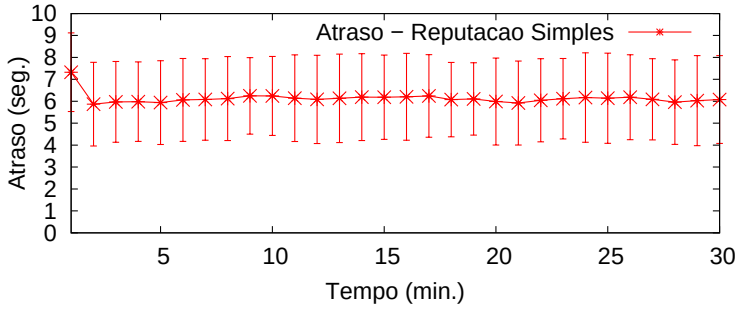


Figura 5.11: Atrasos - Reputação Simples.

Finalizando, a Figura 5.12 mostra o atraso médio quando os poluidores também fazem um ataque de *whitewashing*. Apesar de aplicar o *mecanismo de reputação simples*, o atraso médio é de quase 1 minuto e meio. Este resultado evidência a necessidade de modificação do mecanismo simples para tratar a prática de *whitewashing*. Com o *mecanismo de reputação modificado* proposto, o atraso médio experimentado pelo não poluidores é reduzido a cerca de 13 segundos, valor aceitável para aplicações desta natureza.

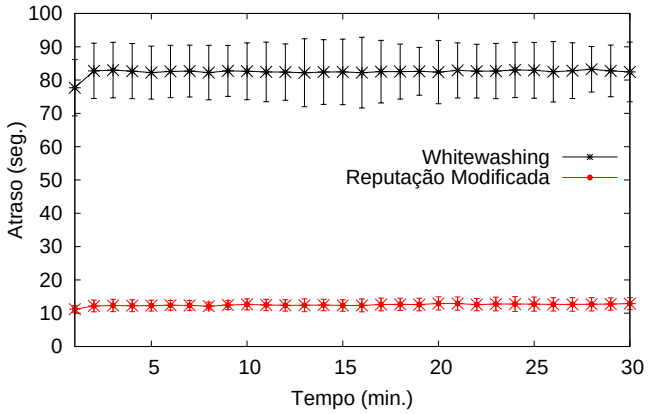


Figura 5.12: Atrasos - Whitewashing e Reputação Modificada.

6 Conclusões

As aplicações de vídeo na Internet têm atraído atenção de pesquisadores e usuários. A arquitetura P2P tem sido amplamente utilizada nesses tipos de aplicações, tomando o lugar que era, tradicionalmente, ocupado pela arquitetura cliente-servidor. Essa preferência pelas redes P2P se deve ao fato destas serem mais escaláveis e resistentes a falhas.

O grande destaque recebido por aplicações P2P para transmissão de vídeo ao vivo atrai também participantes mal intencionados, que tentam se aproveitar de características do sistema para causar danos ao mesmo. Um dos ataques realizados nesse tipo de sistema é o ataque de poluição. Nesse tipo de ataque, o participante malicioso modifica ou corrompe o conteúdo antes de enviá-lo para seus parceiros, podendo alterar a mídia recebida por participantes legítimos do sistema P2P. Para combater esses ataques, um mecanismo de verificação de dados juntamente a um sistema de reputação podem ser empregados. Porém, estes mecanismos podem falhar caso o poluidor também faça *whitewashing*, ou seja, saia e entre novamente no sistema com uma identificação nova.

Tendo em vista os problemas citados acima, neste trabalho, foi desenvolvido um protótipo de aplicação P2P para transmissão de vídeo ao vivo com o objetivo de verificar os efeitos causados por ataques de poluição isolados, bem como combinados com a prática de *whitewashing* nesses sistemas. Além disso, foram propostos mecanismos baseados em reputação para combater tais ataques. Esse protótipo implementa os ataques de poluição e *whitewashing*, assim como os mecanismos de reputação utilizados para combatê-los.

Para a análise dos mecanismos propostos, o protótipo implementado foi testado na plataforma PlanetLab. Através das métricas de interesse definidas, os resultados mostram que utilizar somente um mecanismo de verificação e retransmissão de dados não é uma boa medida para combater ataques de poluição em sistemas P2P de transmissão de vídeo ao vivo. Nesse caso, mesmo impedindo a exibição de *chunks* poluídos e/ou corrompidos, os participantes ainda utilizam três vezes a largura de banda necessária em um sistema sem poluição. Além disso, esse mecanismo de verificação de dados não garante a obtenção de um *chunk* em condições de ser exibido. Isso porque, os participantes ainda perdem mais de 80% das solicitações feitas, dado que continuam recebendo dados poluídos, desperdiçando os recursos disponíveis no sistema e sofrem com mais de 2 minutos atraso na recepção

de *chunks*. Estes resultados motivam propostas de mecanismos de defesa capazes de punir participantes maliciosos, principalmente quando os mesmos tentam se dissimular, mudando sua identificação (*whitewashing*.

Nesta direção foram utilizados dois mecanismos de reputação que combatem ataques de poluição e ataques combinados de poluição e *whitewashing*. Suas principais vantagens são a simplicidade de implementação e a característica distribuída : informações necessárias são coletadas localmente em cada participante da difusão.

Os resultados obtidos com o *mecanismo de reputação simples* implementado, utilizado para combater ataques de poluição isolados, comprovam sua eficácia. O mecanismo possibilitou a redução da sobrecarga para valores abaixo de 5%, ou seja, a reduz em até 46 vezes se comparado ao mecanismo de verificação de dados, praticamente eliminando este problema. Como consequência, as demais métricas também são consideravelmente reduzidas. A taxa de perdas foi reduzida para desprezíveis, 0.7%. Além do atraso na exibição de *chunks*, que foi reduzido para 6 segundos. Assim, o *mecanismo de reputação simples* manteve satisfatória a qualidade de experiência dos usuários, banindo os poluidores e permitindo a rápida recuperação do sistema, quando este sofre com ataques de poluição isolados.

A prática de *whitewashing*, porém, compromete a eficiência desse mecanismo de reputação simples. Os resultados mostraram que o sistema ainda precisaria de o dobro da largura de banda necessária em um sistema sem poluidores, devido à sobrecarga. Além disso, os participantes ainda perdem cerca de metade das solicitações de *chunks* feitas e sofrem com um atraso de quase de 1 minuto e meio na recepção de *chunks*. O *mecanismo de reputação modificado* proposto foi capaz de reduzir, consideravelmente, os efeitos de ataques de poluição combinados com *whitewashing*. Esse mecanismo reduziu a sobrecarga na rede para 20%, a taxa de perdas para um valor menor que 3% e o atraso na exibição de *chunks* para 13 segundos. Estes valores podem manter uma boa exibição do vídeo nos usuários, por isso são aceitáveis para aplicações dessa natureza.

O esquema de recuperação de falsos poluidores implementado também foi avaliado. Este esquema se mostrou capaz de recuperar cerca de 80% dos participantes que, por algum problema de rede, possam ter enviado dados corrompidos por um período de tempo não longo. Assim, apesar de considerar esses participantes como poluidores, o sistema permite que eles se recuperem e voltem a trocar dados com seus parceiros, caso resolvam

seus problemas de rede. Essa característica é importante para que o sistema testado englobe mais situações reais e possa utilizar melhor os recursos disponíveis na rede.

A implementação do protótipo de aplicação P2P para transmissão de vídeo ao vivo no PlanetLab, possibilitou o estudo do comportamento desse tipo de aplicação em um ambiente real, indicando a viabilidade de uso dos mecanismos de combate a ataques de poluição e *whitewashing* na prática.

6.1 Trabalhos Futuros

Algumas linhas de atuação podem estender o trabalho apresentado nessa dissertação, como por exemplo:

- Analisar o comportamento do sistema com mais participantes, incluindo também outros comportamentos presentes em sistemas reais, como a dinâmica dos *peers* e diferentes padrões de chegada dos participantes. Nesse trabalho, por simplificação, os participantes entram ao mesmo tempo no sistema e permanecem até o final dos testes. Na prática, os participantes de um sistema de transmissão de vídeo ao vivo não obedecem regras para entrada e saída do sistema. Assim, o estudo desses comportamentos devem ser considerados para aproximar ainda mais o sistema apresentado nessa dissertação de um sistema real.

- Implementar alterações no protótipo de aplicação desenvolvido para que seja possível analisar quais impactos os ataques de poluição combinados com *whitewashing* causam em um sistema P2P de transmissão de vídeo sob demanda (*VoD*). Foram feitos alguns testes iniciais que indicaram que os resultados neste sistema são semelhantes aos observados nos sistemas P2P de transmissão ao vivo. Porém, ainda é necessário um estudo mais aprofundado dos sistemas de vídeo sob demanda.

- Implementar e verificar, através de testes, a eficiência do mecanismo de recuperação de falsos poluidores quando os poluidores reais fazem *whitewashing*. Ainda não há certeza se a prática de *whitewashing* permitirá que o sistema alcance o estado de calmaria, o que, neste caso, poderia comprometer o mecanismo de recuperação de falsos positivos.

REFERÊNCIAS

[Androutsellis-Theotokis, 2004] Androutsellis-Theotokis, S.; Spinellis, D. A survey of peer-to-peer content distribution technologies. **ACM Computing Surveys (CSUR)**, v.36, n.4, p. 335–371, 2004.

[Caceres, 1999] Caceres, R.; Duffield, N.; Moon, S. B.; Towsley, D. ; others. Inferring link-level performance from end-to-end multicast measurements. **Global Internet, Rio de Janiero**, 1999.

[Castro, 2003] Castro, M.; Druschel, P.; Kermarrec, A.; Nandi, A.; Rowstron, A. ; Singh, A. **Splitstream: high-bandwidth multicast in cooperative environments**. In: ACM SIGOPS Operating Systems Review, volume 37, p. 298–313. ACM, 2003.

[Chen, 2009] Chen, J.; Lu, H. ; Bruda, S. **A solution for whitewashing in p2p systems based on observation preorder**. In: IEEE NSWCTC, 2009.

[Coelho, 2010] Coelho, R.; Barcellos, M.; Jansch-Pôrto, I. ; Gaspary, L. P2p streaming de alta definição: Análise quantitativa de esquemas de assinatura digital para autenticação de conteúdo. **SBRC 2010**, 2010.

[Cohen, 2003] Cohen, B. **Incentives build robustness in bittorrent**. In: Workshop on Economics of Peer-to-Peer systems, volume 6, p. 68–72, 2003.

[Costa, 2007a] Costa, C.; Almeida, J. **Reputation systems for fighting pollution in peer-to-peer file sharing systems**. In: Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on, p. 53–60. IEEE, 2007.

[Costa, 2007b] Costa, C.; Soares, V.; Almeida, J. ; Almeida, V. **Fighting pollution dissemination in peer-to-peer networks**. In: Symposium on Applied Computing: Proceedings of the 2007 ACM symposium on Applied computing, volume 11, p. 1586–1590, 2007.

[Damiani, 2002] Damiani, E.; di Vimercati, D.; Paraboschi, S.; Samarati, P. ; Violante, F. **A reputation-based approach for choosing reliable resources in peer-to-peer networks**. In: Proceedings of the 9th ACM conference on Computer and communications security, p. 207–216. ACM, 2002.

[Dhungel, 2007] Dhungel, P.; Hei, X.; Ross, K. ; Saxena, N. **The pollution attack in p2p live video streaming: measurement results and defenses**. In: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV, p. 323–328. ACM, 2007.

[Douceur, 2002] Douceur, J. The sybil attack. **Peer-to-peer Systems**, p. 251–260, 2002.

[Feldman, 2006] Feldman, M.; Papadimitriou, C.; Chuang, J. ; Stoica, I. Free-riding and whitewashing in peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, v.24, n.5, p. 1010–1019, 2006.

[Gkantsidis, 2004] Gkantsidis, C.; Mihail, M. ; Saberi, A. **Random walks in peer-to-peer networks**. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, volume 1. IEEE, 2004.

[Haridasan, 2006] Haridasan, M.; van Renesse, R. **Defense against intrusion in a live streaming multicast system**. In: Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on, p. 185–192. IEEE, 2006.

[Haridasan, 2007] Haridasan, M.; van Renesse, R. **Securestream: An intrusion-tolerant protocol for live-streaming dissemination.** In: Journal of Computer Communications. Special issue on Foundation of Peer-to-Peer Computing. Elsevier, 2007.

[Haridasan, 2008] Haridasan, M.; van Renesse, R. Securestream: An intrusion-tolerant protocol for live-streaming dissemination. **Computer Communications**, v.31, n.3, p. 563–575, 2008.

[Hei, 2007] Hei, X.; Liang, C.; Liang, J.; Liu, Y. ; Ross, K. A measurement study of a large-scale p2p iptv system. **IEEE Transactions on Multimedia**, v.9, n.8, p. 1672–1687, 2007.

[Hei, 2008a] Hei, X.; Liu, Y. ; Ross, K. Iptv over p2p streaming networks: the mesh-pull approach. **Communications Magazine, IEEE**, v.46, n.2, p. 86–92, 2008.

[Hei, 2008b] Hei, X.; Liu, Y. ; Ross, K. W. Iptv over p2p streaming networks: The mesh-pull approach. **IEEE Communications Magazine**, 2008.

[Jannotti, 2000] Jannotti, J.; Gifford, D.; Johnson, K.; Kaashoek, M. ; others. **Overcast: reliable multicasting with an overlay network**. In: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4, p. 14–14. USENIX Association, 2000.

[Jin, 2006] Jin, X.; Chan, S.; Yiu, W.; Xiong, Y. ; Zhang, Q. **Detecting malicious hosts in the presence of lying hosts in peer-to-peer streaming**. In: Multimedia and Expo, 2006 IEEE International Conference on, p. 1537–1540. IEEE, 2006.

[Karakaya, 2009] Karakaya, M.; Korpeoglu, I. ; Ulusoy, O. Free riding in peer-to-peer networks. **Internet Computing, IEEE**, v.13, n.2, p. 92–98, 2009.

[Leibowitz, 2003] Leibowitz, N.; Ripeanu, M. ; Wierzbicki, A. **Deconstructing the kazaa network**. In: Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on, p. 112–120. IEEE, 2003.

[Li, 2008] Li, J. On peer-to-peer (p2p) content delivery. **Peer-to-Peer Networking and Applications**, v.1, n.1, p. 45–63, 2008.

[Liang, 2005] Liang, J.; Naoumov, N. ; Ross, K. Efficient blacklisting and pollution-level estimation in p2p file-sharing systems. **Technologies for Advanced Heterogeneous Networks**, p. 1–21, 2005.

[Lin, 2010] Lin, E.; de Castro, D.; Wang, M. ; Aycock, J. **Spoim: A close look at pollution attacks in p2p live streaming**. In: Quality of Service (IWQoS), 2010 18th International Workshop on, p. 1–9. IEEE, 2010.

[Liu, 2008] Liu, J.; Rao, S.; Li, B. ; Zhang, H. Opportunities and challenges of peer-to-peer internet video broadcast. **Proceedings of the IEEE**, v.96, n.1, p. 11–24, 2008.

[Lua, 2005] Lua, E.; Crowcroft, J.; Pias, M.; Sharma, R. ; Lim, S. A survey and comparison of peer-to-peer overlay network schemes. **IEEE Communications Surveys and Tutorials**, v.7, n.2, p. 72–93, 2005.

[Lv, 2002a] Lv, Q.; Cao, P.; Cohen, E.; Li, K. ; Shenker, S. **Search and replication in unstructured peer-to-peer networks**. In: Proceedings of the 16th international conference on Supercomputing, p. 84–95. ACM, 2002.

[Lv, 2002b] Lv, Q.; Ratnasamy, S. ; Shenker, S. Can heterogeneity make gnutella scalable? **Peer-to-Peer Systems**, p. 94–103, 2002.

[Moraes, 2008] Moraes, I.; Campista, M.; Moreira, M.; Rubinstein, M.; Costa, L. ; Duarte, O. Distribuiç ao de vıdeo sobre redes par-a-par: Arquiteturas, mecanismos e desafios. 2008.

[Oualha, 2009] Oualha, N.; Roudier, Y. **A game theoretical approach in securing p2p storage against whitewashers**. In: 18th IEEE WETICE'09, 2009.

[Padmanabhan, 2002] Padmanabhan, V.; Wang, H.; Chou, P. ; Sripanidkulchai, K. **Distributing streaming media content using cooperative networking**. In: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, p. 177–186. ACM, 2002.

[Puttaswamy, 2009] Puttaswamy, K.; Zheng, H. ; Zhao, B. Securing structured overlays against identity attacks. **Parallel and Distributed Systems, IEEE Transactions on**, v.20, n.10, p. 1487–1498, 2009.

[Ratnasamy, 2001] Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R. ; Shenker, S. **A scalable content-addressable network**, volume 31. ACM, 2001.

[Ripeanu, 2002] Ripeanu, M.; Foster, I. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. **Peer-to-Peer Systems**, p. 85–93, 2002.

[Seibert, 2010] Seibert, J.; Sun, X.; Nita-Rotaru, C. ; Rao, S. **Towards securing data delivery in peer-to-peer streaming**. In: Communication Systems and Networks (COMSNETS), 2010 Second International Conference on, p. 1–10. IEEE, 2010.

[So, 2012] So, J.; Reeves, D. **Antiliar: Defending against cheating attacks in mesh based streaming**. In: Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on, p. 115–125. IEEE, 2012.

[Stoica, 2001] Stoica, I.; Morris, R.; Karger, D.; Kaashoek, M. ; Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. **ACM SIGCOMM Computer Communication Review**, v.31, n.4, p. 149–160, 2001.

[Tsoumakos, 2003] Tsoumakos, D.; Roussopoulos, N. **A comparison of peer-to-peer search methods**. In: Proceedings of the Sixth International Workshop on the Web and Databases. Citeseer, 2003.

[Vieira, 2008] Borges, A.; Almeida, J. ; Campos, S. **Fighting pollution in p2p live streaming systems**. In: Multimedia and Expo, 2008 IEEE International Conference on, p. 481–484. IEEE, 2008.

[Vieira, 2009] Vieira, A.; Campos, S. ; Almeida, J. **Fighting attacks in p2p live streaming. simpler is better**. In: IEEE INFOCOM Workshops, 2009.

[Vieira, 2012b] Vieira, A.; de Almeida, R.; de Almeida, J. ; Aguiar Campos, S. Simplyrep: A simple and effective reputation system to fight pollution in p2p live streaming. **Computer Networks**, 2012.

[Vieira, 2013] Vieira, A.; Silva, A.; Henrique, F.; Goncalves, G. ; Gomes, P. Sopcast p2p live streaming: Live session traces and analysis. **Multimedia Systems Conference**, 2013.

[Walsh, 2005] Walsh, K.; Sirer, E. **Fighting peer-to-peer spam and decoys with object reputation**. In: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, p. 138–143. ACM, 2005.

[Wang, 2006] Wang, W.; Xiong, Y.; Zhang, Q. ; Jamin, S. **Ripple-stream: Safeguarding p2p streaming against dos attacks**. In: Multimedia and Expo, 2006 IEEE International Conference on, p. 1417–1420. IEEE, 2006.

[Wang, 2007] Wang, F.; Xiong, Y. ; Liu, J. **mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast**. In: Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on, p. 49–49. IEEE, 2007.

[Xie, 2007] Xie, S.; Li, B.; Keung, G. ; Zhang, X. Coolstreaming: Design, theory, and practice. **Multimedia, IEEE Transactions on**, v.9, n.8, p. 1661–1671, 2007.

[Xiong, 2004] Xiong, L.; Liu, L. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. **Knowledge and Data Engineering, IEEE Transactions on**, v.16, n.7, p. 843–857, 2004.

[Yu, 2011] Yu, X.; Fujita, S. Whitewash-aware reputation management in peer-to-peer file sharing systems. **International Conference on Parallel and Distributed Processing Techniques and Applications**, v.6, n.5, p. 9, 2011.

[Zhang, 2005] Zhang, M.; Tang, Y.; Zhao, L.; Luo, J. ; Yang, S. **Gridmedia: A multi-sender based peer-to-peer multicast system for video streaming**. In: Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on, p. 614–617. IEEE, 2005.

[Zhao, 2009] Zhao, B.; Lui, J. ; Chiu, D. **Exploring the optimal chunk selection policy for data-driven p2p streaming systems**. In: Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on, p. 271–280. IEEE, 2009.