Avaliando Simulações de Enxames BitTorrent no TorrentSim*

Matheus B. Lehmann[1], Marinho P. Barcellos[2,3] (orientador)

[1]UNISINOS – Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 – São Leopoldo, RS

[2]PPGC – Programa de Pós-Graduação em Computação
UFRGS – Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Campus do Vale – Bloco IV – Porto Alegre, RS

[3]PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, Prédio 32 – Porto Alegre, RS

matios@gmail.com, marinho@acm.org

Abstract. *This work presents a simulator to investigate BitTorrent networks behavior and, in particular, denial of service attacks and countermeasures in these networks. With the increased use of BitTorrent protocol to distribute contents, protocol vulnerabilities become more relevant, given the consequencies. The project which includes this work intend to reproduce, through simulations, attacks to this kind of network, making possible to evaluate the attacks potential and each countermeasure efectivity against its attack. The scope of this paper aims at measuring simulations performance, in aspect of swarm size and content size.*

Resumo. *Este trabalho apresenta um simulador para investigar o funcionamento de redes BitTorrent e, em particular, ataques de negação de serviço e contramedidas nessas redes. Com o aumento do uso do protocolo BitTorrent para distribuição de conteúdo, vulnerabilidades no protocolo se tornam cada vez mais relevantes, dadas as consequências. O projeto que engloba esse trabalho busca reproduzir, através de simulações, ataques a este tipo de rede, tornando possível avaliar tanto o potencial do ataque, quanto a efetividade de contramedidas propostas para cada ataque. O escopo desse artigo procura testar o desempenho de simulações, em questão de tamanho de enxame e de conteúdo.*

1. Introdução

BitTorrent é um protocolo baseado no paradigma P2P para distribuição e propagação de conteúdo composto por grande volume de dados. Sua característica fundamental é a descentralização da rede utilizando recursos de largura de banda e armazenamento em disco de cada par presente na mesma, garantindo assim, robustez e escalabilidade.

Uma rede do protocolo BitTorrent é denominado um enxame. Este é composto por um rastreador, que centraliza o gerenciamento dos pares no enxame, e pares, que compartilham um dado conteúdo. Os pares presentes no enxame podem ser de dois tipos: sugadores e semeadores. Os sugadores são os pares que ainda não possuem a cópia completa do conteúdo compartilhado, enquanto que os semeadores são os que já o possuem.

*Aluno com apoio do CNPq – Bolsa ITI do Projeto de Pesquisa P2P-SeC.

Cada conteúdo é dividido em peças de tamanho fixo, exceto a última, que pode ser menor. Cada peça é dividida em blocos de tamanho fixo, geralmente de 16 KB.

Apesar de ser um protocolo que vem sendo amplamente utilizado, apresenta vulnerabilidades que podem ser exploradas para dificultar a propagação de conteúdo. Em outros trabalhos são apresentadas maneiras de um par obter vantagem para si mesmo. Em [Locher et al. 2006], é apresentado o ataque de *free-riding*, em que o par explora o enxame fazendo o *download* do conteúdo sem contribuir em troca e, em [Sirivianos et al. 2007], o ataque de *free-riding* é estendido, aumentando o tamanho do conjunto de pares ativos para ter uma visão mais ampla do enxame e aumentando o potencial do *download*.

Este trabalho se concentra em ataques de negação de serviço que tem como objetivo dificultar a evolução de um enxame da rede BitTorrent. Em [Konrath et al. 2007] são introduzidos os ataques de eclipse e mentira de peças. Em [Barcellos et al. 2008], é especificado o ataque de corrupção de peças e apresentadas as contramedidas para os ataques investigados. O presente trabalho propõe investigar a performance das simulações, estimando e verificando até quais tamanhos de enxame e de pares torna viável a execução das mesmas.

Em função da atividade de bolsista de iniciação científica do projeto P2P-SEC financiado pela CNPq, minhas atividades foram a implementação de uma nova versão proposta do algoritmo Rotação de Pares, além da re-engenharia do código. Nesta última atividade, destaca-se a revisão e correção de todas as classes implementadas, documentação clara do código, re-organização estrutural da hierarquia de classes e avaliação das mudanças realizadas.

O restante do trabalho está organizado como segue. A Seção 2 apresenta a fundamentação para os ataques investigados e as contramedidas propostas. A seção 3 discute o projeto do simulador de rede BitTorrent TorrentSim, enquanto a Seção 4 mostra detalhes práticos de implementação dos ataques e contramedidas. Na Seção 5, avalia-se o desempenho do simulador e, na Seção 6, é apresentada a conclusão do trabalho.

2. Fundamentos

Esta seção apresenta os fundamentos das vulnerabilidades detectadas em redes BitTorrent, assim como os ataques de negação de serviço investigados para explorar tais brechas no protocolo e contramedidas criadas como formas de proteção. No presente trabalho não será apresentada a arquitetura do protocolo BitTorrent, há referências sobre o funcionamento deste em [Konrath et al. 2007] e [Barcellos et al. 2008]. Os ataques investigados de negação de serviço têm como objetivo atrapalhar e dificultar a evolução de um enxame, podendo causar a morte do mesmo.

São abordadas três fragilidades do protocolo: fraco gerenciamento de identidades pelo rastreador, política de requisitar peça mais rara primeiro e função *hash* para integridade da peça completada. A partir destas, são investigadas possibilidades de ataques para cada uma dessas. Os ataques estudados são: Eclipse, Mentira de Peças e Corrupção de Peças.

A primeira vulnerabilidade explorada no protocolo é seu fraco gerenciamento de identidades. Apesar do rastreador exercer uma função de autenticador, não há um controle

rígido de autenticação. Desse modo, é possível que um par mal intencionado crie e ingresse no enxame com múltiplas identidades diferentes. Cada identidade virtual diferente controlada por um mesmo par denomina-se *sybil* e é utilizado para massificar o efeito do ataque. A partir dessa vulnerabilidade, o primeiro ataque identificado é denominado "Eclipse", caracterizado por tentar conectar o maior número de *sybils* a pares honestos no enxame. Quanto maior o número de *sybils* conectados a um par (e, consequentemente, menor o número de pares honestos conectados), melhor o ataque, pois dificulta os pares honestos a se encontrarem para trocar conteúdo, causando a diminuição da taxa de *download* do enxame. Se o conjunto de pares ativos de um par qualquer do enxame estiver composto apenas por *sybils*, diz-se que o par está eclipsado. Estando eclipsado, o par não consegue evoluir seu *download*, porque nenhum par remoto irá contribuir com ele. Portanto, quanto maior for a proporção de *sybils* para pares honestos, maior a efetividade do ataque.

A segunda vulnerabilidade encontrada está justamente numa política do protocolo para garantir maximização do tempo de vida do enxame. Cada par requisita novas peças do conteúdo segundo a política da peça mais rara primeiro, de acordo com a própria visão de peças no enxame. Aliando a possibilidade de controlar múltiplos *sybils* com a de mentir a posse de peças, é identificado o segundo ataque: "Mentira de peças", que caracteriza-se pela mentira em massa das peças que os *sybils* supostamente possuem. Configura-se um conjunto de peças a serem mentidas e, ao iniciar novas conexões, estas peças são informadas para os pares honestos a quem estão se conectando. Em seguida o *sybil* não realiza qualquer outra ação, permanecendo ocioso. O sucesso do ataque é forçar os pares a não requisitar as peças mentidas de forma que, em determinado momento, alguma das peças desapareça do enxame, causando a morte do mesmo.

A última fragilidade do protocolo pesquisada está na forma de garantir a integridade do conteúdo, a função *hash SHA-1*. Uma característica fundamental dessa função é que ela seja não reversível, isto é, a partir do valor obtido seja impossível retornar à entrada da função. No protocolo BitTorrent, ao completar o *download* de uma peça, o par calcula o *hash* dela e compara ao valor armazenado no arquivo de metadados do conteúdo. Se o *hash* não for igual, diz-se que ele falhou e a peça deve ser descartada e requisitada novamente, pois não sabe-se qual bloco está corrompido. Este aspecto fundamenta o último ataque investigado, o de "Corrupção de Peças". O objetivo deste é dificultar a propagação do conteúdo, corrompendo peças ao enviar um ou mais blocos corrompidos para o par remoto. Pelo fato de não ser detectável quem enviou o bloco corrompido, este ataque ganha eficácia e tempo de duração. A maximização do ataque consiste em corromper cada peça com apenas um bloco, tendo em vista que apenas um é suficiente para o *hash* falhar e o segundo não acrescenta efeito no ataque, apenas consumo de recursos.

Partindo destes ataques investigados, foram pesquisadas formas de combatê-los e garantir continuidade e boa evolução do enxame. Foram propostas duas contramedidas, chamadas de "Rotação de Pares", para combater a ociosidade de um par remoto conectado, e "Anti-Corrupção", contra pares contribuindo com conteúdo falso.

A contramedida de Rotação de Pares está baseada no tempo de conexão de um par remoto e na quantidade de dados transferidos por este par. A partir de uma análise periódica desses valores para cada par conectado, é possível detectar quais pares estão ociosos e desconectá-los, abrindo oportunidade para conexão com outros pares remo-

tos que possivelmente possam contribuir para o *download* do conteúdo. Enquanto isso, a Anti-Corrupção está fundamentada na contribuição dos pares remotos. A cada peça completada, é verificada sua integridade. Caso seja íntegra, são acrescentados pontos de reputação para os pares contribuintes, caso contrário, é feita a tentativa de recuperação da peça e o decremento da reputação dos pares envolvidos na corrupção da peça.

3. Projeto

Os ataques e contramedidas fundamentadas na seção anterior são avaliados num ambiente de simulação. Para essa avaliação, foi desenvolvido um simulador de enxames BitTorrent, denominado TorrentSim. O projeto desse simulador é apresentado a seguir.

O TorrentSim é baseado no simulador de protocolos, Simmcast [Barcellos et al. 2001]. O Simmcast é um framework de simulação discreta implementado na linguagem de programação Java. O *framework* implementa um modelo genérico de protocolo, permitindo ao usuário que o especialize para o comportamento desejado. Como consequência, há o grande re-aproveitamento de código, ocasionando menor esforço de desenvolvimento e maior robustez da aplicação. Partindo de uma API com operações básicas de comunicação, o usuário estende as classes do *framework* para modelar o protocolo desejado, garantindo que as funcionalidades do *framework* sejam utilizadas, sem a necessidade de reinventá-las.

O TorrentSim estende o Simmcast para modelagem de enxames BitTorrent. Utilizando a base do Simmcast, são especificados os nodos, que representam os pares do enxame, e a rede, representando o enxame, para desenvolver a arquitetura do protocolo investigado. Na Figura 1, é apresentada a estrutura hierárquica dos nodos presentes no TorrentSim. Todo elemento do enxame, é representado como um nodo BitTorrent. A partir desse nodo, tem-se a especialização para o rastreador e um par do enxame. O par do enxame pode ser extendido tanto para um par honesto, quanto para um par malicioso ou atacante. Além do par honesto básico, implementado segundo o protocolo BitTorrent, ele pode conter os comportamentos de contramedida de Rotação de Pares ou de Anti-Corrupção. Por sua vez, o par malicioso implementa os três ataques descritos anteriormente: Eclipse, Mentira de Peças e Corrupção de Peças.

O TorrentSim tem como elementos básicos o nodo BitTorrent, o rastreador, o par e o par honesto. Partindo dessa base, foi aumentada a complexidade de implementação, acrescentando comportamentos distintos para os pares como forma de investigar ataques e contramedidas no enxame. Essa extensão para modelar os ataques e contramedidas tem a mesma idéia do Simmcast. No TorrentSim, primeiro foi criado um modelo básico do protocolo e, em seguida, especializado para os diferentes comportamentos, permitindo re-aproveitamento de código e maior facilidade na implementação. No TorrentSim, cada elemento presente no enxame é representado por um nodo, contendo as estruturas de dados necessárias para execução, e uma thread, que define o comportamento do mesmo durante a simulação.

Nodo BitTorrent. Baseado numa estrutura hierárquica de classes do paradigma orientado ao objeto, o nodo BitTorrent é o elemento base do simulador. Este nodo implementa a estrutura de dados básicos que qualquer conceito da rede BitTorrent deve ter, como sua identidade real (IP) e virtual, enquanto que a thread possui os comportamentos base para qualquer elemento da rede deve ter, sobrescrita do envio e recebimento de

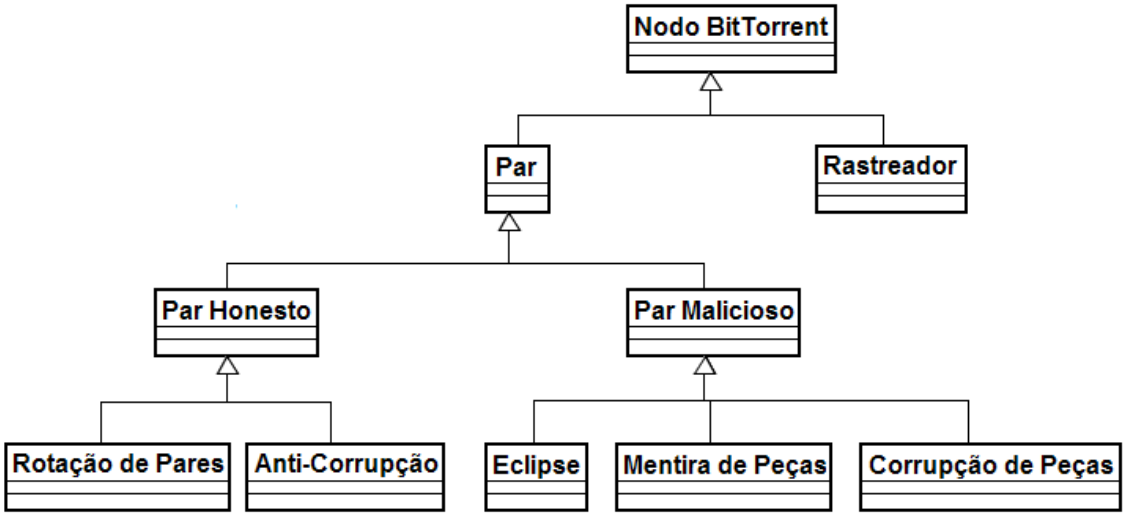

Figura 1. Diagrama da estrutura hierárquica de classes implementadas do TorrentSim.

mensagens, por exemplo. Este nodo e thread derivam da classe Nodo do Simmcast e será estendido para implementar os pares e o rastreador do enxame.

Rastreador. Responsável pelo gerenciamento fraco dos pares no enxame e autenticação dos mesmos durante o tempo de permanência no enxame. Implementado a partir do nodo BitTorrent, ele possui os atributos e métodos necessários para gerar uma lista de pares randômicos. Seu comportamento consiste em gerenciar os pares ao longo do tempo de vida do enxame, permitir a entrada e saída de pares no enxame e enviar conjunto de pares aleatórios quando requisitado.

Par. Outro elemento especializado a partir do nodo BitTorrent, o par do enxame é projetado como uma classe abstrata. Ou seja, é uma classe nunca instanciada, que fornece uma implementação incompleta compartilhada por suas classes derivadas. Este par possui as características básicas tanto de um par honesto, quanto um malicioso. Na prática, ele contém todas as estruturas de dados, tais como conjunto de pares sufocados, interessados, requisições, entre outras, e comportamentos necessários para executar o protocolo, como o envio de todas mensagens do protocolo, conexão e requisição de conjunto de pares ao rastreador, mas não implementa completamente o protocolo. São derivados o par honesto e o malicioso.

Par Honesto. Deriva do par do enxame. Herda sua estrutura de dados da classe pai e implementa o comportamento especificado pelo protocolo. Interage com outros pares e com o rastreador no enxame, faz avaliações de pares esnobes e maiores contribuidores, envia e recebe dados do conteúdo. Possui também alguns atributos de controle para o tempo de permanência no enxame como o semeador, como o tempo mínimo que o par local deve permanecer no enxame semeando. A partir do par honesto, é especializado para a contramedida de Rotação de Pares e de Anti-Corrupção

Rotação de Pares. Especializa o par honesto, agregando o comportamento de rotacionar pares durante o tempo de permanência no enxame. O par com Rotação de Pares procura detectar aqueles pares que estão inativos e apenas ocupando seus espaços

de conexões, através de avaliações periódicas. Essa contramedida tanto pode ser aplicada ao ataque de Eclipse, quanto ao de Mentira de Peças.

Anti-Corrupção. Também estende o par honesto, acrescentando estrutura para controle de reputação e recuperação de peça e métodos para executar a contramedida. Busca detectar os pares maliciosos que estão contribuindo com blocos corrompidos e baní-los do seu conjunto de pares conectados. Isto é feito segundo um sistema de reputação baseado nas contribuições de cada par remoto para as peças recebidas pelo par local.

Par Malicioso. Além do par honesto, o par malicioso também é uma especialização do par do enxame. Seu nodo e seu comportamento permitem múltiplas identidades virtuais para a maximização da eficácia do ataque. Classe abstrata que serve como base para a implementação de todos os ataques. Atualmente, estende-se três ataques: Eclipse, Mentira de Peças e Corrupção de Peças.

Eclipse. Este ataque especializa o par malicioso, executando o comportamento de eclipsar pares remotos com múltiplas identidades. Busca maximizar o número de *sybils* conectados a outros pares a fim de dificultar a distribuição do conteúdo.

Mentira de peças. O segundo ataque estende o par malicioso, acrescentando estrutura de dados e controle sobre quais peças serão mentidas e acrescenta-se o comportamento de mentir peças para o ataque. Procura, com um conjunto de *sybils*, mentir a posse de um conjunto de peças com a finalidade de fazê-la desaparecer do enxame, matando o mesmo.

Corrupção de peças. O terceiro e último ataque tem como base o par malicioso e implementa o comportamento de enviar peças corrompidas para outros pares do enxame. Seu ataque visa atrasar o *download* de outros pares e fazê-los consumir mais recursos por ter que baixar uma mesma peça múltiplas vezes.

4. Implementação

Nesta seção será discutida a implementação dos pares com os diferentes tipos de ataques e contramedidas que foram apresentados na Seção 3. Primeiramente, são estudadas questões de implementação dos pares honestos e, em seguida, dos pares maliciosos.

Um par honesto do enxame BitTorrent possui estruturas de dados para armazenar informações próprias e informações fornecidas pelos outros pares do enxame. As estruturas básicas necessárias são identidade virtual, informações relacionadas ao conteúdo e enxame, incluindo número de pares e número de peças, conjunto de pares conhecidos e ativos, mapa de bits das peças possuídas, conjunto de pares remotos que são interessantes para o par local e de pares remotos que estão interessados no par local, lista de pares remotos que estão sufocados por par local de pares remotos que estão sufocando par local atualmente, peças requisitadas pelo par local e requisitados pelos pares remotos do enxame para o par local. A fim de evitar replicação de informação e gerar sobrecarga, todas as informações comuns compartilhadas pelos pares são armazenadas no enxame, tais como número de blocos, de peças e tamanho dos blocos. Além das estruturas de dados, os comportamentos básicos de inicializar estruturas, iniciar conexões, registrar-se e requisitar lista de pares ao rastreador, receber e tratar diferentes tipos de mensagens, enviar mensagens, avaliar os pares para fazer *upload*, avaliar os pares que estão esnobando e

desconectar também são implementados. Esse par honesto especificado serve como base, permitindo sua extensão para adaptar comportamentos. Desse modo, é possível estudar e avaliar os impactos das contramedidas propostas.

Estendendo o par honesto e acrescentando a contramedida de Rotação de Pares, muda-se o comportamento do par e acrescenta-se dados necessários para suportar tal contramedida. A Rotação de Pares utiliza um mecanismo de quarentena, baseado no tempo de conexão e quantidade de dados trocados com os pares conectados. Para isso, é necessário ter estruturas que guardem o tempo restante dos pares em quarentena, o tempo de conexão dos pares conectados atualmente, a quantidade de dados trocadas na conexão atual e o próximo tempo de quarantena para cada par. Esse último refere-se ao fato que cada nova quarentena sentenciada para um par, seu tempo a cumprir aumenta de acordo com uma taxa definida pelo par local por ser reincidente. Para detectar pares ociosos no enxame, é realizada uma avaliação periódica, atualmente a cada minuto, com todos os pares conectados, em que é calculado a taxa de transferência (dados enviados e recebidos dividido pelo tempo de conexão) de cada par e ordenados de forma crescente por essa taxa de transferência. Em seguida são desconectados os pares de pior índice de transferência, mantendo 75% do número mínimo de pares conectados e conectando novos pares até alcançar o número mínimo de pares conectados, especificado pelo protocolo. Esse número mínimo de pares conectados refere-se ao valor mínimo de pares necessários no conjunto de pares ativos.

Assim como o par com Rotação de Pares, o de Anti-Corrupção também necessita de estrutura complementar de dados para conseguir executar seu comportamento. Orientado a evento e baseado em reputação, a contramedida propõe alterar a reputação de acordo com o sucesso ou fracasso da peça recebida. Além disso, caso a peça venha corrompida, há uma estrutura auxiliar utilizada na recuperação da mesma, que visa completá-la requisitando o menor número de blocos para poupar tempo e consumir menos recursos. A cada peça corrompida, o par replica a estrutura de requisição de peças e armazena informações sobre a peça corrompida com o intuito de recuperá-la, como blocos a requisitar novamente e último par a contribuir para a peça, que será utilizado para tentar recuperá-la.

O par malicioso é uma extensão do par base do enxame BitTorrent. Em questões práticas, ele não implementa completamente o protocolo, apenas as funcionalidades necessárias para efetivar seu ataque de negação de serviço. Também representada por uma classe abstrata, possui elementos comuns para todos os diferentes pares atacantes. Como o atacante não está preocupado em fazer o *download* do conteúdo, algumas estruturas podem ser relevadas, como as de controle para recebimento e envio de dados, enquanto outras devem ser sobrescritas. São armazenadas todas identidades virtuais e a lista de pares sufocados é re-escrita para suportar uma lista exclusiva para cada *sybil*, pois apesar de agirem em conluio, podem estar conectados com diferentes pares. Como os pares atacantes usualmente atacam em bando composto de vários *sybils*, deve haver um controle do recebimento de mensagens e sincronização das mesmas com as diferentes identidades. Isso é feito sobrescrevendo os métodos de tratamento de mensagens implementadas pelo par.

São investigados três ataques no contexto de negação de serviço em enxames BitTorrent: Eclipse, Mentira de Peças e Corrupção de Peças. A seguir, suas respectivas

implementações são apresentadas.

O primeiro ataque analisado é o par malicioso com ataque de Eclipse. Este não possui nenhuma estrutura extra à definida pela classe abstrata do par malicioso. Devido ao fato de seu único comportamento ser iniciar conexão com pares remotos e ocupar espaço no conjunto de pares conectados, seu código é simples e seu comportamento básico. Pela simplicidade apresentada, prova-se ser altamente escalável.

O segundo ataque é o par malicioso com ataque de Mentira de Peças. Além das estruturas básicas, este par possui uma estrutura de armazenamento das peças cuja posse são mentidas para outros pares. Igualmente ao ataque de Eclipse, seu comportamento é bem simples e, além de iniciar conexão com pares remotos, o par deve também informar as peças supostamente possuídas. Esse envio de mensagem do mapa de peças é feito apenas uma vez por conexão. Sua escalabilidade assemelha-se ao Eclipse.

O último ataque, de Corrupção de Peças, possui uma estrutura de armazenamento de requisições solicitados por pares remotos para cada *sybil*. Pelo mesmo motivo da lista de pares sufocados por cada *sybil*, cada *sybil* recebe requisições diferentes dos pares remotos, por isso cada requisição deve ser armazenada separadamente. Seu comportamento é mais complexo, pois este par interage com os pares enviando blocos requisitados a cada intervalo de tempo, geralmente 8 segundos. O par malicioso deve armazenar as requisições solicitadas para cada *sybil*, tratar mensagens de solicitações de blocos e avaliar quais blocos devem ser enviados. Além disso, o par malicioso envia blocos com cada *sybil* em intervalos constantes de tempo. Mesmo sendo mais complexo, esse ataque se prova ser escalável.

5. Avaliação

Para a avaliação foi usada uma grade com 8 computadores. Cada computador tem a seguinte configuração: processador Pentium 4 Xeon 2.4 HT, 2GB de memória RAM, 80 GB de armazenamento em disco, sistema operacional Gentoo Linux 2007.0, java versão "1.5.0_11" Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_11-b03) Java HotSpot(TM) Server VM (build 1.5.0_11-b03, mixed mode).

Foram escolhidos três cenários para a avaliação do simulador. Todos cenários possuem apenas pares honestos, diferindo cada uma na versão do par honesto. Um cenário com par honesto base, outro com o par honesto e contramedida de Rotação de Pares e o último com Anti-Corrupção. A escolha desses cenários leva em conta verificar se alguma contramedida torna sua simulação inviável de ser executada.

Para avaliar o desemepenho do simulador de enxames BitTorrent na questão de tamanho do mesmo, foram executadas simulações dos diferentes cenários variando o número de pares em escala logaritmica desde 4 pares até 1024 pares com conteúdo fixo em 64 peças de 1 MB, sendo o total de 64 MB de dados. Na Figura 2, o eixo x representa o número de pares, enquanto o eixo y significa o tempo de execução de uma simulação em minutos. Pode-se perceber que o aumento do tamanho do enxame, incrementa o tempo de processamento de maneira linear, provando ser escalável inclusive para além de 1024 pares. Não há grande diferença entre as simulações com e sem contramedida, mostrando indícios que a implementação dos mecanismos de defesa propostos não causa uma sobrecarga de processamento.

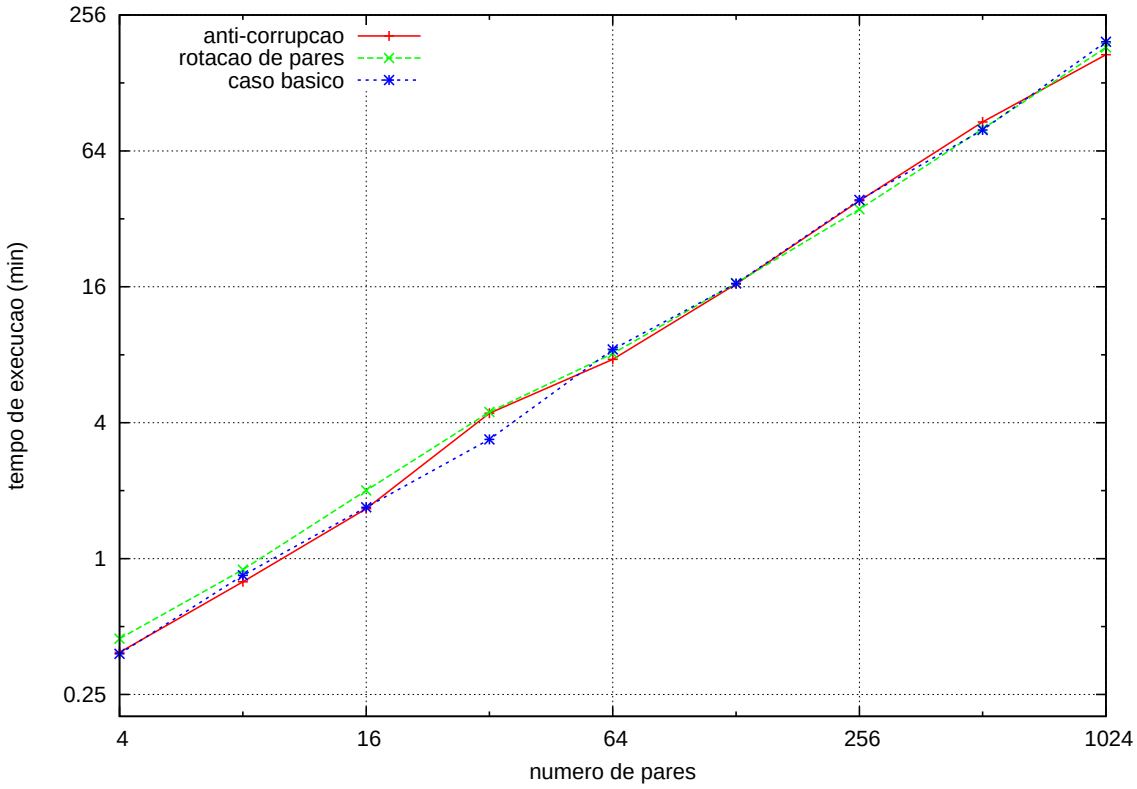

Figura 2. Gráfico do tempo de processamento das simulações em função do número de pares no enxame.

A avaliação do limite do TorrentSim em função do tamanho de conteúdo, foi feita a partir de uma série de simulações executadas dos diferentes cenários, variando o número de peças em escala logaritmica desde 16 peças até 1024 peças com número de pares no enxame fixo em 128 pares. Cada peça foi definida com tamanho de 1 MB, portanto o tamanho do arquivo compartilhado varia de 16 MB até 1024 MB. A partir da Figura 3, cujo eixo x representa o número de pares e o eixo y, tempo de execução da simulação, conclui-se que o aumento do tempo de processamento até o volume de dados compartilhado de 256 MB ocorre de maneira linear. No intervalo de 256 MB a 1024 MB, o crescimento deste tempo se torna maior, demonstrando indícios que simulações de grandes volumes de dados talvez não sejam escaláveis.

6. Conclusão

O TorrentSim é uma ferramenta valiosa na avaliação de enxames BitTorrent, através de simulações. Devido a sua modularidade, o simulador permite extensões de comportamentos e funções de maneira rápida e simples. No contexto do projeto P2P-SEC, essa característica permite implementar os ataques e contramedida investigados, auxiliando no entendimento e avaliação dos mesmos.

Esta ferramenta não está disponível atualmente para outros projetos, devido à reformulação em andamento que visa agregar qualidade necessária para sua futura divulgação, que consiste na minha atividade como bolsista de iniciação científica financiado pelo CNPq, destacado como focos do projeto a reformatação e melhoramento de código nos aspectos de clareza e comentários. Apesar disso, o simulador se provou útil

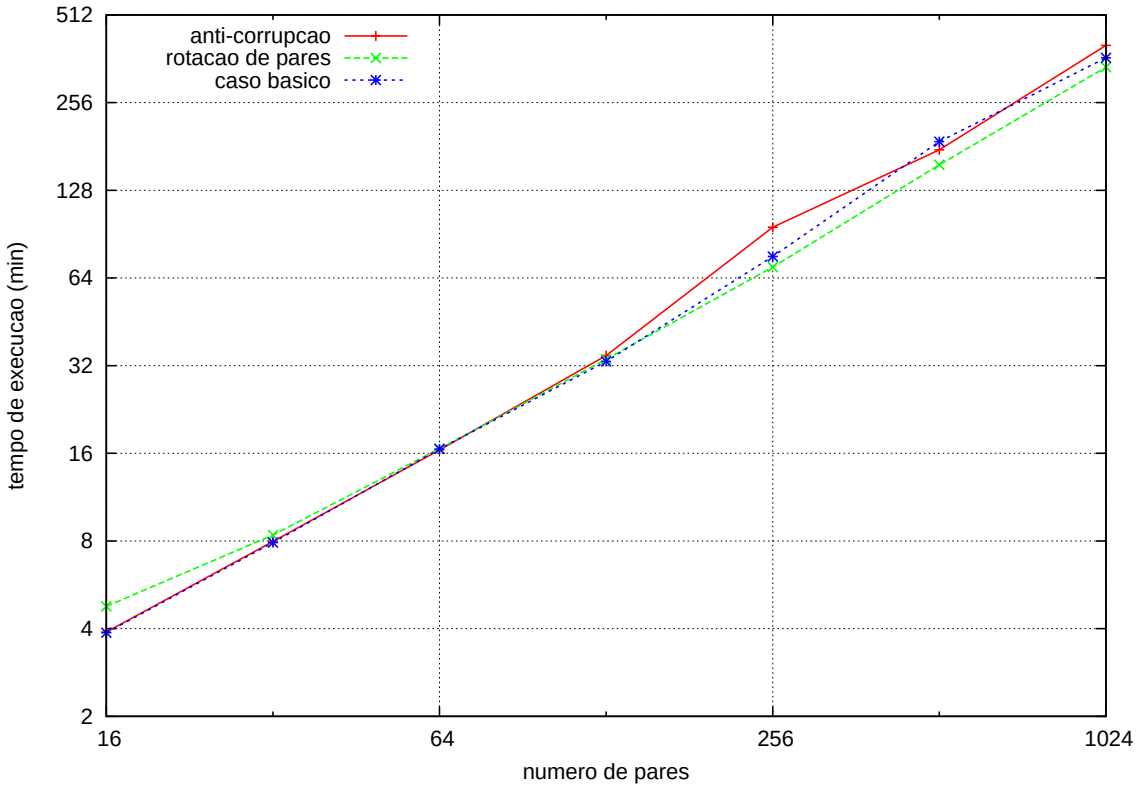

Figura 3. Gráfico do tempo de processamento das simulações em função do tamanho do conteúdo do enxame.

na simulação de enxames, com os resultados do presente artigo indicando que é possível executar simulações que se aproximem de configurações reais de enxame, o que poderá em breve auxiliar outros grupos de trabalho com simulação de redes BitTorrent.

Referências

Barcellos, M., Muhammad, H., and Detsch, A. (2001). Simmcast: a simulation tool for multicast protocol evaluation. In *XIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2001)*, pages 418–433.

Barcellos, M. P., Bauermann, D., Sant'anna, H., Lehmann, M., and Mansilha, R. (2008). Protecting bittorrent: design and evaluation of effective countermeasures against dos attacks. In *27th International Symposium on Reliable Distributed Systems (IEEE SRDS 2008)*.

Konrath, M. A., Barcellos, M. P., and Mansilha, R. B. (2007). Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent. In *The Seventh IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2007)*. IEEE.

Locher, T., Moor, P., Schmid, S., and Wattenhofer, R. (2006). Free riding in bittorrent is cheap. In *Fifth Workshop on Hot Topics in Networks (HotNets-V)*, Irvine, CA, US.

Sirivianos, M., Park, J. H., Chen, R., and Yang, X. (2007). Free-riding in bittorrent with the large view exploit. In *6th International Workshop on Peer-to-Peer Systems (IPTPS 2007)*, Bellevue, WA, US.