TorrentLab: Um Ambiente para Avaliação do Protocolo BitTorrent

Rodrigo B. Mansilha[1], Marinho P. Barcellos[2], Francisco V. Brasileiro[3]

[1] UNISINOS – Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 – 93.022, São Leopoldo, RS

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

[3]UFCG – Universidade Federal de Campina Grande
Av. Aprígio Veloso, 882 Bloco CO – 58.109-970, Campina Grande, PB

rmansilha@gmail.com, marinho@acm.org, fubica@dsc.ufcg.edu.br

Resumo. *BitTorrent é provavelmente o mais popular protocolo para distribuição de conteúdo na atualidade, sendo utilizado por milhões de pessoas para compartilhar conteúdo. Embora goze deste sucesso, BitTorrent foi criado primariamente como um esforço de engenharia. Isso criou uma demanda da comunidade científica por avaliações de BitTorrent, que permitissem melhor entendimento do protocolo. Recentemente, houve uma série de trabalhos com avaliações de BitTorrent que identificaram deficiências e oportunidades para melhorias do protocolo. Este trabalho apresenta o projeto e implementação de um ambiente integrado de avaliação de redes BitTorrent, denominado TorrentLab, que permite tanto a simulação quanto a execução de experimentos utilizando BitTorrent.*

Abstract. *BitTorrent is probably the most popular protocol in content distribution networks, being used by millions of people to share content. Although quite successful, it was created mostly as an engineering effort. This led to a demand of BitTorrent evaluations, to allow a better understanding of the dynamics of the protocol. Recently, there have been attempts to measure and understand the behavior of BitTorrent networks, which led to drawbacks being exposed and opportunities for improvement. Simulation and experimental evaluation are two important ways of evaluating BitTorrent implementations and variations. TorrentLab is a BitTorrent testbed that allows BitTorrent experiments under both controlled and uncontrolled environments.*

1. Introdução

Compartilhamento de arquivos em redes P2P tornou-se uma das mais relevantes aplicações da Internet, permitindo uma rápida disseminação de conteúdo entre usuários. Nesta categoria, BitTorrent é um dos protocolos mais populares, sendo utilizado por milhões de pessoas para compartilhar conteúdo [Das et al. 2006] e consumindo fração substancial do tráfego da Internet [Barbera et al. 2005]. Sistemas de distribuição segura de conteúdo proprietário baseados em BitTorrent estão começando a entrar em operação (ex: Azureus Vuze) e grandes empresas como Warner Brothers, 20^{th} Century Fox e BBC já estão oferecendo conteúdo através desses sistemas.

A literatura é rica em artigos sobre BitTorrent, a maioria dos quais requer algum tipo de avaliação quantitativa, com objetivos tais como elucidar o comportamento de sistemas atuais e avaliar o benefício de novas propostas. Classicamente,

existem três metodologias de avaliação: *analítica*, *simulacional* e *experimental*, sendo cada método mais adequado a uma determinada situação. Idealmente, as técnicas de avaliação não deveriam ser usadas isoladamente, de forma a oferecer uma confiança maior na validade dos resultados [Jain 1991]. Independente da metodologia escolhida, existe sempre um custo associado à sua execução, potencialmente inviabilizando empregar mais de uma metodologia em conjunto. Um dos maiores desafios é o esforço envolvido ao se passar de uma metodologia para outra, como por exemplo entre simulação e avaliação experimental em uma rede real. Tanto a implementação como a plataforma de execução dos experimentos tende a ser completamente distinta. Neste contexto, seria desejável avaliar o BitTorrent através de diferentes metodologias , e que isso pudesse ser feito de maneira integrada e sem-esforço, similarmente ao que já foi empregado em outros contextos [Barcellos et al. 2006, Urbán et al. 2002].

Este trabalho apresenta o projeto, implementação e avaliação de um ambiente integrado de avaliação de redes P2P, com ênfase em BitTorrent e por tal denominado **TorrentLab**. Diferentemente de propostas anteriores, o ambiente comporta de maneira homogênea a execução de avaliações baseadas em simulação e experimentos em rede real. Em termos gerais, a rede BitTorrent é primeiramente descrita pelo usuário em bom grau de detalhe através de um Ambiente de Modelagem. A especificação gerada serve de entrada para ambos os métodos de avaliação considerados. O TorrentLab é composto de mais três ambientes, sendo dois deles denominados **TorrentSim** e **TorrentExp**. Como o próprio nome já indica, o primeiro é responsável pelas simulações, enquanto o segundo pelos experimentos na rede. Ambos valem-se de uma infraestrutura de computação distribuída para viabilizar a execução paralela de simulações ou a instanciação de agentes em máquinas remotas. As saídas produzidas, em ambos os casos, são consolidadas em um Ambiente de Consolidação de Resultados.

O restante do artigo está organizado como segue. Os trabalhos relacionados são discutidos na Seção 2. A arquitetura do TorrentLab, formada por Ambientes e Módulos, é apresentada na Seção 3, enquanto que aspectos de implementação relacionados à materialização de tal arquitetura são alvo da Seção 4. Para validar nossa proposta, na Seção 5 resultados de experimentos com o TorrentLab são analisados e comparados com a literatura. Na Seção 6, são apresentadas as considerações finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados a formas de avaliação do BitTorrent e ambientes de avaliação similares em natureza ao TorrentLab. Primeiramente, são abordadas formas de avaliação do BitTorrent, que ajudam a demonstrar a empregabilidade do ambiente. Em seguida, trabalhos com propostas semelhantes são discutidos, apontando o diferencial do TorrentLab.

Entre as três metodologias de avaliação anteriormente citadas, a modelagem analítica é possivelmente aquela que permite um melhor entendimento sobre as propriedades globais do sistema. Porém, no caso do BitTorrent, seu funcionamento é amplamente influenciado pela escolha de valores para parâmetros básicos como tamanho máximo do conjunto de pares e número de pares a serem beneficiados com upload. Segundo [Bharambe et al. 2006], seria difícil, senão impossível, controlar um espaço tão amplo de possibilidades em uma configuração baseada em modelagem analítica. Por exemplo, em [Pontes et al. 2007] os autores apresentam um modelo

analítico para estimar a taxa de download dos peers, mas fazem isso considerando um instante de tempo na execução do protocolo, não conseguindo portanto analisar a dinâmica do sistema.

A complexidade do problema ou a dificuldade prática de se conduzir experimentos em larga escala leva a abordagem baseada em simulação a ser bastante popular. Exemplos de simulações de BitTorrent são [Purandare and Guha 2006], onde é avaliado um mecanismo de auto-punição para encorajar usuários a cooperarem e [Bindal et al. 2006], que avalia uma proposta de melhoria ao protocolo, baseada em escolha de vizinhos mais próximos . Em [Konrath et al. 2007b, Konrath et al. 2007a, Mansilha et al. 2007], os autores avaliam o impacto de ataques de segurança sobre o BitTorrent.

Por fim, no campo experimental, em trabalhos como [Erman et al. 2005] e [Guo et al. 2007] traços são coletados através de pontos de monitoração e de uma versão instrumentada de um agente de usuário BitTorrent. Nestes trabalhos, sites que publicam arquivos ".torrent" são vasculhados e enxames BitTorrent extensivamente examinados. Em diversos trabalhos, agentes de usuário BitTorrent são modificados (ou completamente reimplementados) de maneira a coletar dados e/ou estudar experimentalmente os mecanismos básicos do BitTorrent, e particularmente comprovar a possibilidade de comportamento injusto na rede por parte de pares carona. Exemplos dessa categoria de trabalhos são [Shneidman et al. 2004, Pouwelse et al. 2005, Locher et al. 2006, Piatek et al. 2007, Sirivianos et al. 2007].

A seguir, são comentados os trabalhos mais próximos a este. Em [Naicken et al. 2007], os autores realizam uma análise superficial sobre simuladores em P2P e encontram uma gama deles, incluindo BitTorrent. Complementarmente à simulação, dois *testbeds* que permitem avaliação experimental de sistemas P2P (incluindo BitTorrent) são o P2PLab e o PlanetLab. Em [Nussbaum and Richard 2006], apresenta-se o P2PLab, um ambiente que permite a execução de experimentos em rede real, valendo-se de uma grade computacional como substrato. No entanto, o P2PLab exige FreeBSD como sistema operacional e demanda virtualização para execução de múltiplas instâncias de agente P2P na mesma máquina. PlanetLab é uma grade computacional intercontinental utilizada no desenvolvimento de protocolos e aplicações de redes. Em termos de funcionalidade, P2PLab e o PlanetLab podem ser comparados estritamente apenas com o lado experimental do TorrentLab (conforme explicado adiante). Além disso, não são orientados a BitTorrent, e por isso não consideram as especificidades deste ambiente. Ao que sabemos, não existe proposta similar na literatura, que combine as abordagens e permita a execução de experimentos de investigação de sistemas BitTorrent.

3. Arquitetura do TorrentLab

Esta seção descreve a arquitetura do TorrentLab. Primeiramente, a Subseção 3.1 faz um levantamento de requisitos funcionais. A Subseção 3.2 apresenta o Ambiente de Modelagem e a Subseção 3.3 mostra como um agregado ou grade computacional funciona como substrato fundamental para computação distribuída no TorrentLab. A Subseção 3.4 define o Ambiente de Simulação (TorrentSim), e a Subseção 3.5 apresenta o Ambiente de Execução de Experimentos (TorrentExp). Por fim, a Subseção 3.6 apresenta o Ambiente de Consolidação de Resultados.

3.1. Requisitos Funcionais

A arquitetura do TorrentLab foi organizada em função de cinco requisitos funcionais. O primeiro requisito é prover uma forma de modelagem unificada de redes BitTorrent: esta deve servir igualmente bem para experimentos e para simulações. Idealmente, o conjunto de cenários modelados deve ser próximo ao conjunto de cenários possíveis na realidade. Ainda, os modelos devem ser persistentes, para permitir reaproveitamento.

O segundo requisito é oferecer uma interface para execução de simulações. A interface deve ser modular e extensível para avaliação de novos algoritmos e comportamentos.

O terceiro requisito é oferecer meios para executar experimentos com implementações de BitTorrent, ou seja, com agentes de usuário (chamados popularmente de "clientes BitTorrent") e rastreadores. Combinando o conjunto de configurações de conteúdo a ser compartilhado, de tipos de agentes de usuário, de condições da rede, o conjunto de possibilidades resultante é infinito. Se não é viável esperar que o ambiente de experimentação satisfaça todas as possibilidades, por outro lado ele deve ser flexível a ponto de não restringir desnecessariamente as possibilidades de experimento.

O quarto requisito é unificar a consolidação de saídas de agentes e simuladores. A consolidação de resultados é feita através de dois modos principais: gráficos e extração de dados quantitativos/estatísticos. Algumas estatísticas básicas devem ser implementadas, como por exemplo, tempo médio de download e número máximo de pares na rede, assim como geração automática de gráficos ilustrando a evolução de pares no enxame. Além de oferecer um conjunto de possibilidades "típicas" de métricas e gráficos, a arquitetura deve possibilitar que os existentes sejam estendidos ou novos sejam criados e integrados ao sistema.

Em teoria, um experimento de simulação ou real pode ser executado em um único hardware. Na prática, no entanto, as limitações impostas pela exiguidade de recursos em uma única máquina requerem que as simulações e experimentos sejam executados valendo-se de um conjunto de máquinas, formando uma infraestrutura distribuída. Portanto, o quinto e último requisito é integrar a utilização de agregados ou grades computacionais de maneira a aumentar a escalabilidade dos experimentos e das simulações.

3.2. Ambiente de Modelagem

O Ambiente de Modelagem oferece, de maneira homogênea, funcionalidade para modelagem de enxames para simulação e experimentos. O desafio é criar um mecanismo que atenda diferentes requisitos e sintaxes, identificando parâmetros comuns e permitindo a extensão para parâmetros específicos. Largura de banda dos agentes é um exemplo de parâmetro comum a qualquer simulação ou experimento real. Por outro lado, um usuário disposto a desenvolver, avaliar, ou testar um novo agente de usuário BitTorrent deve ter meios para variar um determinado parâmetro específico que não foi previsto.

Este ambiente recebe como entrada interações do usuário e oferece como saída um objeto encapsulando todas as informações referentes ao enxame a ser avaliado. Este objeto é passado como parâmetro para o TorrentSim ou TorrentExp de acordo com o tipo de avaliação sendo processada.

3.3. Agregado ou Grade Computacional

Conforme mencionado anteriormente, a utilização de um conjunto de máquinas no TorrentLab aumenta a escalabilidade das simulações e dos experimentos. Do ponto de vista das simulações, a execução em um agregado ou em uma grade computacional permite reduzir o tempo de resposta de uma campanha, à medida que as várias simulações que formam a campanha podem ser facilmente executadas em paralelo. Por outro lado, uma maior quantidade de máquinas faz com que seja possível instanciar mais agentes BitTorrent e executar experimentos reais com mais pares. Dessa forma, a execução de uma campanha de simulação ou de um experimento podem ambas serem vistas como a execução, sobre a infraestrutura multi-processada, de uma "aplicação" paralela composta por várias tarefas.

O TorrentSim e o TorrentExp geram tarefas a serem processadas pela infraestrutura de computação distribuída. Além disso, eles são responsáveis pelo escalonamento dessas tarefas na infraestrutura de execução. Cada tarefa de uma "aplicação" TorrentLab executa três fases, quais sejam: i) preparação do ambiente (*stage in*), que é responsável por transferir para a máquina da infraestrutura onde a tarefa irá executar (máquina remota) os dados de entrada e, se necessário, o executável do agente; ii) iniciação da execução do agente na máquina remota (*spawn*); e, iii) coleta de resultados (*stage out*), que é responsável por transferir os resultados da execução do agente da máquina remota para a máquina onde a aplicação foi iniciada. Após a instanciação e término ou falha das tarefas, os resultados são enviados para o respectivo ambiente.

3.4. Ambiente de Simulação: TorrentSim

O TorrentSim é responsável pela instanciação e controle de simulações. Ele oferece uma interface para um simulador de BitTorrent extensível baseado no arcabouço Simmcast [Barcellos et al. 2001]. A integração com o TorrentLab é relevante sobre três aspectos da avaliação através da simulação: modelagem de enxames, execução na infraestrutura computacional, e comparação de resultados com o TorrentExp. Primeiro, a modelagem de enxames é facilitada porque pode-se reutilizar modelos garantindo equivalência dos parâmetros entre múltiplas execuções. Segundo, a integração com o TorrentLab simplifica execuções em lote através do agregado/grade, abstraindo todo o processo de instanciação e controle. Finalmente, há equivalência entre TorrentExp e TorrentSim no que tange parâmetros de entrada e métodos de consolidação de resultados, permitindo a comparação.

A Figura 1 ilustra a arquitetura do TorrentSim no âmbito do TorrentLab. Primeiramente, o Ambiente de Modelagem fornece o modelo do enxame a ser avaliado. O Ambiente de Simulação traduz o modelo para a sintaxe do simulador e gera uma quantidade de tarefas igual ao número de sementes aleatórias utilizadas. As tarefas, junto do simulador e dos parâmetros, são enviadas para a infraestrutura computacional subjacente, que por sua vez, encarrega-se de distribuí-las entre os nodos que o compõe. Conforme as simulações terminem ou falhem, os resultados são retornados para o TorrentSim, que os formata para o Ambiente de Consolidação. Por fim, os dados são consolidados gerando os resultados finais.

3.5. Ambiente de Experimentos: TorrentExp

O TorrentExp é responsável pela instanciação e controle de experimentos. Um experimento BitTorrent consiste em uma encenação da realidade através da instanciação fracamente sincronizada de agentes de usuário e rastreador reais sobre uma rede de

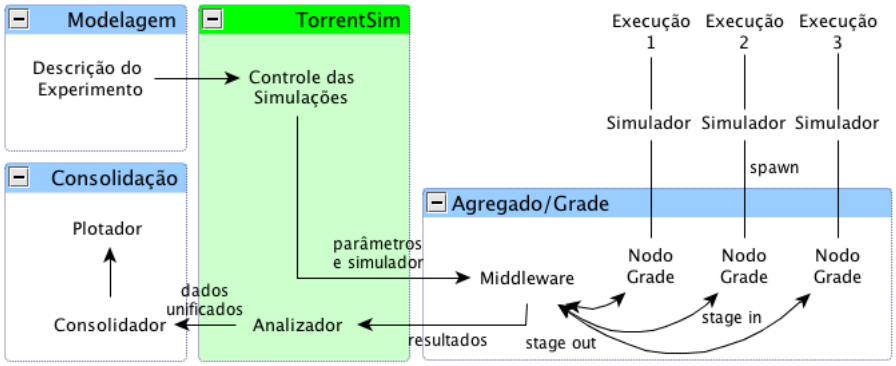


Figura 1. Arquitetura do TorrentSim

computadores. Os traços a serem gerados pelos agentes são realísticos, ao contrário de simulações em que certos fatores ambientais precisam ser abstraídos.

O objetivo do TorrentExp é permitir que experimentos sejam conduzidos com implementações de BitTorrent em "configurações reais". Estas podem ser divididas em três tipos, em ordem decrescente de controle e sincronização: uma rede local controlada, provavelmente parte de um agregado de alto desempenho, uma rede de longo alcance com Qualidade de Serviço, parte de uma infraestrutura de grade, ou este último caso sem quaisquer garantias de QoS. No primeiro caso, o cientista possui o controle do experimento, porém sofre limitações de escala; no outro extremo, perde-se a reprodutibilidade, porém os experimentos são mais realísticos e podem ser efetuados em larga escala.

Para permitir a execução de experimentos de acordo com o modelo de enxame inserido, o sistema deve possuir meios para definir quando executar agentes remotos (e quais agentes) sobre uma determinada rede. O sistema deve garantir, ou mais fracamente presumir (por exemplo através de superprovisionamento), que os nodos selecionados possuam no momento recursos disponíveis suficientes para instanciação do agente (por exemplo, largura de banda).

A Figura 2 ilustra a arquitetura do TorrentExp, como inserido no TorrentLab. Primeiramente, o TorrentExp recebe do Ambiente de Modelagem o modelo do enxame a ser avaliado e, de acordo com o mesmo, gera tarefas e transmite-as para um elemento intermediário chamado *Orquestrador*. Este é responsável pela instanciação e sincronização das tarefas na infraestrutura distribuída. O *middleware* encarrega-se de distribuir as tarefas entre os nodos que o compõe. Conforme os agentes terminem de cumprir seu papel no enxame, a tarefa termina e os resultados são retornados para o TorrentExp. O nodos liberados são aproveitados pelo Orquestrador para instanciar a chegada de novos pares ao enxame. Ao final da participação de todos os pares, as saídas são processadas e formatadas para o Ambiente de Consolidação. Este, por sua vez, consolida os dados, gerando os resultados finais.

Existem dois tipos de agentes em enxames BitTorrent: *de usuário* e *rastreador*. O rastreador é o elemento central que torna possível o encontro dos pares no enxame e por isto seu endereço deve ser pré-determinado antes que as tarefas dos agentes de usuário sejam definidas. Os agentes de usuário podem atuar como semeadores (possuem cópia completa do conteúdo a ser distribuído) ou sugadores (o oposto). O conteúdo distribuído no enxame é gerado através de um algoritmo determinístico. Caso o par entre no enxame como semeador, o arquivo descrito no

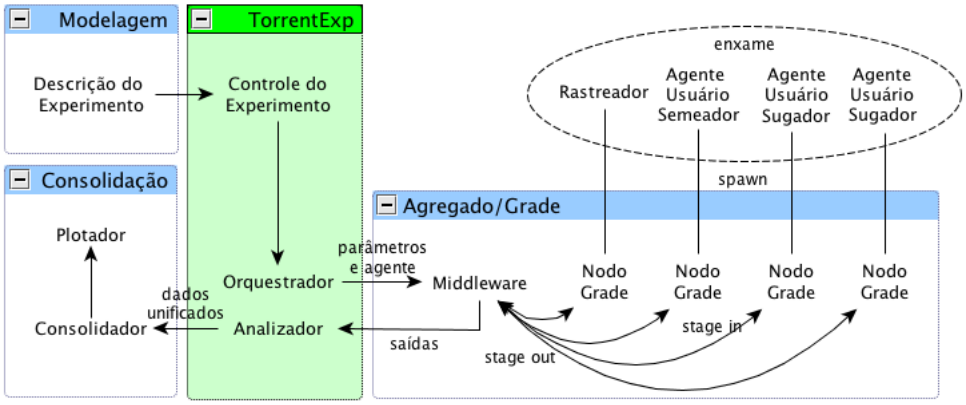


Figura 2. Arquitetura do TorrentExp

.torrent é reproduzido remotamente através do mesmo algoritmo.

3.6. Ambiente de Consolidação de Resultados

O Ambiente de Consolidação de Resultados é responsável por "unificar" os rastros, tanto gerados pelo TorrentSim (e extensões do mesmo) como por diferentes agentes executados no TorrentExp. Em outras palavras, este ambiente permite definir métodos de análise em apenas um local, independentemente da origem dos traços. Cada agente e simulador apresenta sintaxes de saídas diferentes, além de diferir quanto a tipo, unidade e glanuraridade de dados de saída. No entanto, como a proposta é a mesma, encenar enxames BitTorrent, espera-se que a partir de traços de fontes diferentes seja possível identificar dados com semântica em comum.

O ambiente é composto por três módulos: analizador léxico/sintático, consolidador e plotador. Primeiro, o analizador é a interface para o processamento de traços, sendo implementado pelo TorrentSim e TorrentExp. Segundo, consolidador é o objeto que encapsula as informações processadas pelo analizador. Terceiro, o plotador é a interface para o programa externo que, a partir dos objetos consolidadores, gera resultados finais.

Desta forma, cada passo do fluxo de consolidação é atribuído ao elemento participante mais coerente: aquele que gera o traço é o mesmo que o processa; quando um novo método de consolidação é definido, ocorre em apenas um lugar; por fim, a interface para o elemento externo plotador é responsável apenas pela geração do resultado final. Consequentemente, separa-se a etapa de processamento de dados da etapa de geração de resultados, além de centralizar definições de métodos de consolidação.

4. Aspectos de Implementação

A arquitetura apresentada na seção anterior foi materializada através da construção de um protótipo, demonstrando as principais características da proposta. A implementação permitiu averiguar o acerto das decisões principais e, ocasionalmente, enganos ou omissões. Esta seção oferece uma breve descrição da implementação, ressaltando os apectos mais relevantes.

Para a implementação do protótipo, foram assumidas premissas simplificatórias. Dentre elas, destacam-se a escolha pelo ambiente Unix e agregado em rede local com pelo menos 100 Mbps de largura de banda.

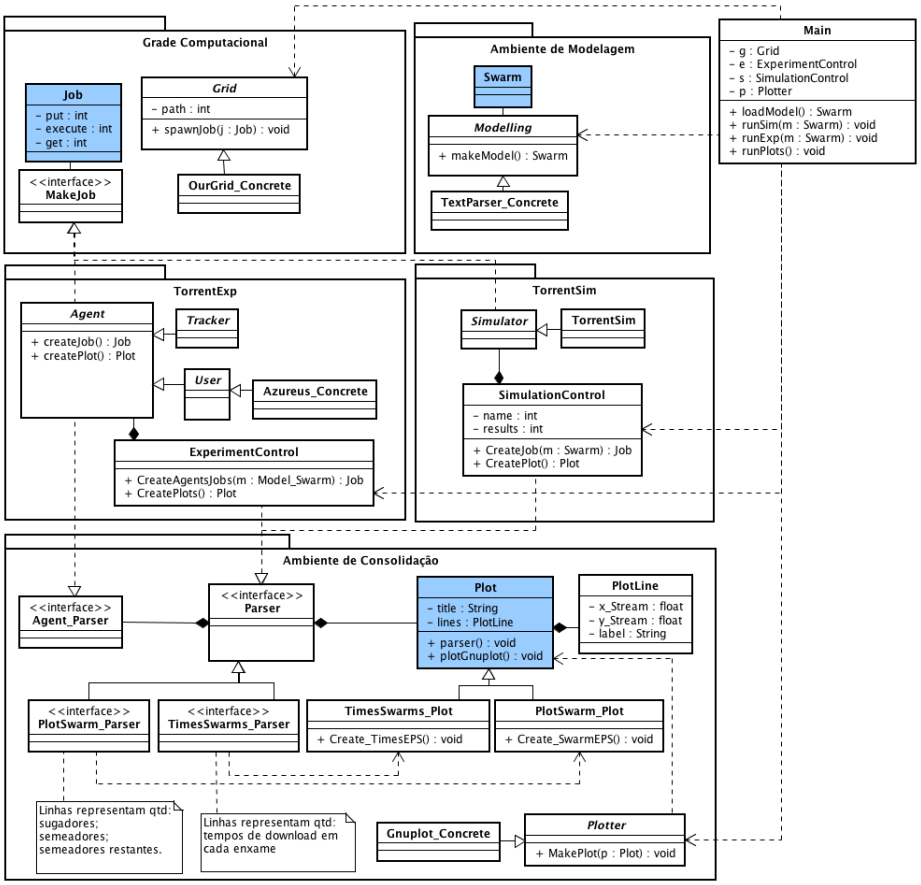


Figura 3. Diagrama de classes da implementação do TorrentLab

A estrutura da implementação é ilustrada através de um diagrama de classes UML, conforme Figura 3. Cada *package* representa um dos componentes do TorrentLab e são interligados através da classe Main. Foram utilizadas estruturas de dados (classes em destaque) para a comunicação entre as partes, permitindo que TorrentExp e TorrentSim compartilhem ambientes comuns ao fluxo de avaliação. Nas subseções seguintes, cada um dos ambientes é comentado, além da infraestrutura subjacente de grade/agregado.

4.1. Ambiente de Modelagem

O Ambiente de Modelagem implementado baseia-se na definição de uma linguagem e um analizador léxico-sintático correspondente. Para descrever o modelo, pode ser utilizada qualquer ferramenta editora de textos. O interpretador recebe como entrada um arquivo com a descrição do enxame e tem como saída um objeto enxame (Swarm), a ser enviado para o TorrentSim e TorrentExp, encapsulando todas as informações. Maiores detalhes sobre a linguagem de descrição são omitidos por conta de sua simplicidade.

Novos parâmetros podem ser livremente definidos : todos os parâmetros são transmitidos para os ambientes de execução (simulação/experimento), mas apenas as classes que implementam a interpretação de tais parâmetros as utilizam, sendo ignorados pelas outras classes. Esta liberdade simplifica a utilização e aumenta a coesão do projeto, transferindo para as classes que precisam dos parâmetros a decisão de utilizá-los ou não.

4.2. Grade Computacional

Dentre as diferentes opções de middleware para grades computacionais, escolheu-se o OurGrid. Esse middleware de código aberto dá suporte à construção de grades computacionais P2P, possibilitando a agregação de um número considerável de recursos. Além disso, o middleware provê um escalonador de aplicações que se adequa perfeitamente à execução das campanhas de simulação do TorrentLab e que pode ser ligeiramente adaptado para atender aos requisitos do escalonamento de tarefas no caso de experimentos do TorrentLab. Um dos princípios de projeto do OurGrid é a facilidade de instalação e manutenção da grade. Finalmente, o middleware já está bastante maduro, uma vez que dá suporte a uma grade computacional pública que está em produção desde dezembro de 2004[1].

Estendeu-se a classe abstrata Grid para o OurGrid. Esta classe recebe objetos encapsulando tarefas do TorrentSim ou TorrentExp. A partir destes objetos, são criados arquivos descritores de tarefas que servem de entrada para OurGrid, mapeando-se os atributos. Em seguida, as tarefas são adicionadas à grade. Ao passo em que os resultados são retornados, eles são enviados para o respectivo ambiente (TorrentSim ou TorrentExp).

4.3. TorrentSim

Em trabalhos recentes [Konrath et al. 2007b, Konrath et al. 2007a, Mansilha et al. 2007], os autores fazem uso de um simulador de BitTorrent para avaliar aspectos de segurança nesse protocolo. A evolução deste simulador, que resulta de uma extensão do arcabouço de simulação Simmcast [Barcellos et al. 2001], deu origem ao TorrentSim. As propriedades de extensibilidade e modularidade do Simmcast são herdadas pelo TorrentSim, de forma que novas simulações de BitTorrent no TorrentLab são feitas estendendo-se o arcabouço disponível.

A conexão com o TorrentLab é feita através da implementação da classe concreta TorrentSim, que estende a classe abstrata Simulator, implementando a interface MakeJob. Esta interface cria a tarefa para o Grid traduzindo as entradas (recebidas no objeto da classe Swarm) para a linguagem do simulador.

O OurGrid provê facilidades para escalonar aplicações do tipo "saco-de-tarefas" (*bag-of-tasks*), que são aplicações paralelas em que o processamento é dividido entre tarefas independentes que não precisam trocar informação entre si. A execução de uma campanha de simulação envolve a execução de diversas simulações independentes e se consitui no exemplo típico de uma aplicação saco-de-tarefa. Dessa forma, o TorrentSim usa diretamente o escalonador de tarefas do OurGrid para executar de forma paralela cada campanha recebida.

4.4. TorrentExp

O ambiente de experimentos implementado presentemente assume que a infraestrutura distribuída é instanciada em uma rede local com relógios físicos frouxamente sincronizados (com erro máximo da ordem de dezenas de ms). Cabe ao módulo de experimentos, a partir do objeto com a definição do enxame (Swarm), criar os objetos Jobs necessários para instanciação da rede modelada. A seguir é discutida a implementação dos elementos que constituem o TorrentExp.

[1]http://status.ourgrid.org/ apresenta o estado da grade em tempo real.

Orquestrador. Este elemento é responsável por escalonar os agentes do usuário para execução na grade computacional. Esses agentes devem ser escalonados de forma a entrarem no enxame nos momentos definidos na configuração do experimento. Como discutido anteriormente, o escalonador do OurGrid é apropriado apenas para o escalonamento de aplicações (ou *jobs* para usar a terminologia do OurGrid) do tipo saco-de-tarefas. No caso dos experimentos do TorrentLab as "tarefas" não podem ser executadas a qualquer momento. Para aproveitar a infraestrutura oferecida pelo OurGrid nós transformamos a execução de um experimento na execução paralela de vários jobs na grade, cada uma consistindo de uma única tarefa. Assim, o papel do Orquestrador se resume a adicionar cada job na fila de entrada do escalonador do OurGrid, obedecendo a ordem de tempo de chegada no enxame especificada na configuração do experimento.

Agentes de usuário. Adotou-se como agente padrão o Azureus (3.01), por este ser o cliente mais utilizado atualmente. Parâmetros fixos são previamente configurados em uma instância em local pré-determinado. Esta instância é compactada e enviada para o nodo remoto da grade. O processo de descompactação, instanciação e controle é feita através de um *script*. Cabe a classe ExperimentControl criar o *script* conforme o objeto Swarm.

Agente rastreador. A instanciação do rastreador foi implementada de outra maneira: executa-se um agente rastreador paralelamente ao ambiente. Isto permite o acompanhamento em tempo real dos enxames instanciados. Atráves da grade computacional este acompanhamento não seria possível, dado que a grade não conhece a semântica dos agentes que está executando. Para a implementação do protótipo, optou-se pelo rastreador interno do Azureus (3.01).

4.5. Consolidação de Resultados

Na implementação do Ambiente de Consolidação dos Resultados, foi desenvolvida a interface (Parser) para os plots da Figura 4 e Figura 5, PlotSwarmParser e TimesSwarmsParser respectivamente. As classes concretas AzureusConcrete e TorrentSim implementam estas interfaces.

Foram implementadas as duas classes de consolidação (PlotSwarmsPlot eTimeSwarmPlot) referentes às interfaces analizadoras. Nelas foram definidos atributos como legendas para os eixos, tal como mostrado nas Figuras 4 e 5. Por fim, foi implementado a classe interface (Plotter) para o software de plotagem Gnuplot[2].

5. Avaliação do Ambiente

Para entendermos quão próximos os resultados produzidos pelo TorrentLab estão da realidade, conduzimos um estudo preliminar cujo objetivo é comparar quantitativamente valores para métricas importantes gerados pelo TorrentSim e TorrentExp, e posicionar os mesmos em relação à avaliação analítica. Esta seção apresenta os resultados e conclusões obtidas com esse estudo.

A rede onde foi instanciado o TorrentLab é composta por aproximadamete 30 máquinas (Pentium 4 / 1 GB RAM) em um enlace de 100 Mbps e agrupadas em um par de comutadores. As seguintes configurações foram adotadas: conteúdo com 60 MB de tamanho. 1 semeador inicial, com largura de banda de download e upload iguais a 2048 Kbps e 512 Kbps, respectivamente, e razão de contribuição 2. O número de sugadores foi variado entre 4 e 64, sendo que os mesmos dispõem de

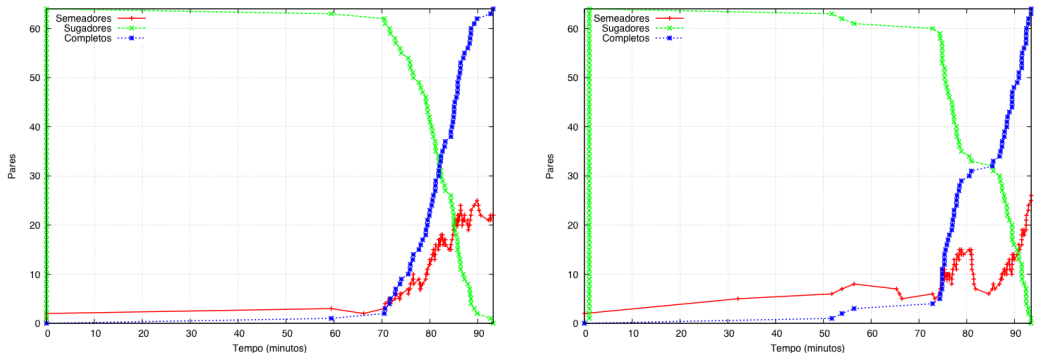
[2]http://www.gnuplot.info

uma largura de banda de 512 e 128 Kbps para download e upload, entram ao mesmo tempo e adotam razão de contribuição alvo 1. Como agente de usuário e rastreador, adotou-se o Azureus (3.01).

5.1. Comparação com experimentos

Esta avaliação comparou o desenvolvimento de um enxame executado via TorrentSim e TorrentExp. Com isto, "co-validamos" os mesmos ao demonstrar que produzem resultados semelhantes. Uma similaridade entre as curvas indicaria, mas não provaria, que os ambientes retornam valores confiáveis.

De forma indireta (não explorada neste artigo), esta avaliação ajudou a demonstrar a apropriabilidade do Ambiente de Modelagem e de Consolidação, pois ambos foram empregados para criar o experimento e analisar os resultados, neste caso de forma gráfica. O Ambiente de Modelagem é usado para descrever o enxame e se vale do mesmo conjunto de parâmetros para os casos experimental e simulacional. A aplicabilidade do Ambiente de Consolidação foi indicada ao permitir a definição de métodos de criação de gráficos em apenas um local, de forma integrada.



(a) Simulação (b) Experimento

Figura 4. Comparação entre ambientes TorrentSim e TorrentExp

As Figuras 4(a) e 4(b) apresentam a dinâmica de dois enxames, um real e outro simulado, que permitem a comparação entre os dois ambientes de avaliação. As curvas representam a população de pares em termos de semeadores, sugadores, e downloads completos no enxame ao longo do tempo (o eixo X representa tempo em minutos). O enxame definido foi configurado com 64 sugadores e executado sob o TorrrentSim e o TorrentExp múltiplas vezes.

O par de gráficos na Figuras 4(a) e 4(b) mostra claramente que, pelo menos para o cenário considerado, os resultados são bastante semelhantes, não apenas no resultado final (tempo médio de dowload), mas também na evolução do enxame. Apesar de omitidos deste trabalho por uma questão de espaço, os resultados foram igualmente animadores em uma gama de cenários que avaliamos.

5.2. Avaliação da propocionalidade de enxames

Enquanto a subseção anterior focou na dinâmica de um enxame em particular, mostrando a similaridade "micro" entre os ambientes simulado e experimental, nesta o objetivo é analisar, em caráter preliminar, o impacto de se usar um conjunto limitado de recursos para executar um experimento com um enxame de proporções maiores, bem como comprovar que os resultados produzidos se encontram em linha com o esperado segundo a literatura.

Foram executados experimentos em que se mediu, para diferentes tamanhos de enxame, o tempo médio de download observado por pares. Uma das propriedades mais conhecidas e valorizadas em BitTorrent é sua escalabilidade ([Massouli and Vojnovi 2005]): devido à divisão em peças e a distribuição das mesmas entre pares, o tempo médio de download cresce suavemente no início com o tamanho do enxame e então estabiliza, dependendo das características do enxame tais como fase (*flash crowd*, estável ou em término), razão de contribuição que os pares obedecem, largura de banda de download e upload, e topologias das redes de sobreposição e física.

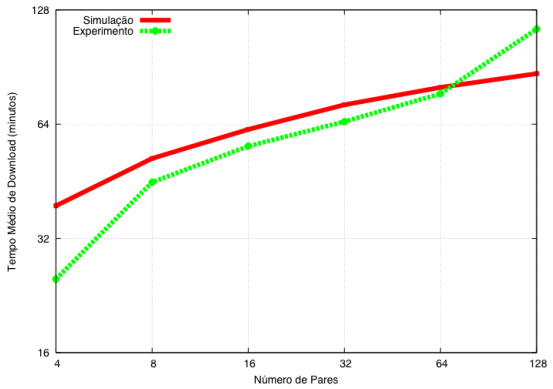


Figura 5. Comparação entre resultados obtidos com TorrentSim e TorrentExp

A Figura 5 apresenta os resultados obtidos com o TorrentSim e TorrentExp, para enxames entre 4 e 128 pares. Os eixos X e Y representam, respectivamente, o tamanho do enxame e tempo médio de download, ambos em escala logarítmica. Como demonstrado na figura, a curva de simulação representa um comportamento suave (correspondente ao que esperávamos encontrar) e próximo à curva experimental para enxames entre 8 e 64 pares. Para enxames de 4 e 128 pares, nota-se uma diferença mais acentuada. Para o primeiro caso, não consideramos significativo o erro dada a peculiaridade de um enxame com 4 pares. Já no segundo caso, houve um crescimento anômalo no tempo médio, gerando uma discrepância notável entre os resultados de simulação e experimental. Após uma campanha de experimentos de avaliação em que estudamos os rastros e resultados parciais, concluímos que tal se deve a um esgotamento dos recursos físicos (processador, memória e rede) dos equipamentos empregados, bastante limitados.

6. Conclusões e Trabalhos Futuros

BitTorrent está se tornando um padrão de fato no compartilhamento de arquivos e distribuição de conteúdo em larga escala, motivando assim muitos esforços de pesquisa relacionados ao protocolo. A investigação sobre o funcionamento das implementações atuais e a avaliação de propostas de modificações requer o emprego de uma ou mais metodologias: analítica, simulacional ou experimental, preferencialmente em combinação. Entretanto, as dificuldades de montar cenários e gerar múltiplos experimentos em sistemas paralelos e comparáveis, bem como a escassez de recursos, acabam por limitar as investigações a apenas uma técnica.

Neste contexto, TorrentLab é apresentado como ambiente para avaliação do BitTorrent. O mesmo foi estruturado de forma a permitir a extensão para novas classes de agentes, simuladores, plotadores e infraestruturas computacionais distribuídas. A implementação apresentada ilustra as relações entre as partes (através de

suas interfaces) e serve como prova-de-conceito, contribuindo para demonstrar sua viabilidade.

O ambiente implementado foi utilizado em duas avaliações. A primeira comparou a evolução em termos de população de pares e downloads completados de um mesmo enxame, investigado tanto no TorrentSim como no TorrentExp. Os gráficos demonstram a similaridade entre os dois métodos. A segunda avaliação examinou uma propriedade fundamental do BitTorret, sua escalabilidade, permitindo uma comparação entre resultados obtidos usando as duas metodologias de avaliação. Os resultados indicam, ainda que em escala limitada, que há uma consistência entre os valores gerados pelo TorrentSim e TorrentLab, e que os mesmos estão em linha com a literatura.

Como trabalhos futuros, vislumbramos três iniciativas principais. Primeiro, a execução de experimentos adicionais, no sentido de melhor validar o TorrentLab e fazer um ajuste fino do protótipo já em operação. Segundo, reavaliar as questões pertinentes ao Orquestrador: dimensionamento de recursos de hardware, tratando de aspectos como gargalos de rede e capacidade de processamento das máquinas. Terceiro e último, enriquecer o ambiente com a adição de novos agentes de usuário.

Referências

Barbera, M., Lombardo, A., Schembra, G., and Tribastone, M. (2005). A markov model of a freerider in a bittorrent P2P network. In *IEEE Global Telecommunications Conference (GLOBECOM '05)*, volume 2, pages 985–989, St. Louis, MO, USA.

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., Facchini, G., Muhammad, H. H., Bedin, G. B., and Luft, P. (2006). Bridging the gap between simulation and experimental evaluation in computer networks. In *Simulation Symposium, 2006. 39th Annual*, pages 8 pp.+.

Bharambe, A. R., Herley, C., and Padmanabhan, V. N. (2006). Analyzing and improving a bittorrent networks performance mechanisms. In *25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–12.

Bindal, R., Cao, P., Chan, W., Medved, J., Suwala, G., Bates, T., and Zhang, A. (2006). Improving traffic locality in bittorrent via biased neighbor selection. pages 66–66.

Das, S., Tewari, S., and Kleinrock, L. (2006). The case for servers in a peer-to-peer world. In *2006 IEEE International Conference on Communications*, volume 1, pages 331–336, Washington, DC, USA. IEEE Computer Society.

Erman, D., Ilie, D., and Popescu, A. (2005). Bittorrent session characteristics and models. In *Proceedings of the 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, pages P30/1–P30/10.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X. (2007). A performance study of bittorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley.

Konrath, M. A., Barcellos, M. P., and Mansilha, R. B. (2007a). 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.

Konrath, M. A., Barcellos, M. P., Silva, J. F., Gaspary, L. P., and Dreher, R. (2007b). Atacando um enxame com um bando de mentirosos: vulnerabilidades em bittorrent. In *XXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007)*, volume 2, pages 883–896.

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.

Mansilha, R. B., Konrath, M. A., and Barcellos, M. P. (2007). Corrupção, mentiras e isolamento: avaliação de impacto de ataques a bittorrent. In *VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG 2007)*.

Massouli, L. and Vojnovi, M. (2005). Coupon replication systems. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, volume 33, pages 2–13, New York, NY, USA. ACM Press.

Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., and Chalmers, D. (2007). The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37(2):95–98.

Nussbaum, L. and Richard, O. (2006). Lightweight emulation to study peer-to-peer systems. In *20th International Parallel and Distributed Processing Symposium, 2006 (IPDPS 2006)*, pages 8 pp.+, Washington, DC, USA. IEEE Computer Society.

Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2007). Do incentives build robustness in bittorrent? In *NSDI'07*, Cambridge, MA.

Pontes, F., Brasileiro, F., and Andrade, N. (2007). Sobre calotes e múltiplas personalidades no bittorrent. Campina Grande, PB, Brasil. Departamento de Sistemas e Computação.

Pouwelse, J. A., Garbacki, P., Epema, D. H. J., and Sips, H. J. (2005). The bittorrent p2p file-sharing system: Measurements and analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*.

Purandare, D. and Guha, R. (2006). Preferential and strata based p2p model: Selfishness to altruism and fairness. In *12th International Conference on Parallel and Distributed Systems, 2006. ICPADS 2006*, volume 1, pages 561–570.

Shneidman, J., Parkes, D., and Massoulie, L. (2004). Faithfulness in internet algorithms. In *Proc. SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04)*, Portland, OR, USA. ACM SIGCOMM.

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.

Urbán, P., Défago, X., and Schiper, A. (2002). Neko: A single environment to simulate and prototype distributed algorithms. *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, pages 981–997.