Uma Arquitetura Para Experimentações em Larga Escala de Redes P2P

Abel P. C. de Souza[1], **Antonio A. de A. Rocha**[1]

[1]Instituto de Computação (IC)
Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

{asouza,arocha}@ic.uff.br

Abstract. *Experiments in networks, especially in peer-to-peer networks (P2P), may require a massive amount of operations to be properly constructed and executed, especially if a researcher is not familiarized with the various existing networking tools. In this study, we developed Networt eXperiment Tool (NeXT), an architecture and tool that helps monitoring the traffic generated by applications like P2P in a network infrastructure. Using NeXT, various analyzes of an application can be implemented and evaluated, disregarding the large amount of manual operations required to perform such experiments.*

Resumo. *Os experimentos em redes, especialmente nas redes par a par (P2P), podem requerer uma quantidade massiva de operações para serem corretamente construídos e executados, especialmente se um pesquisador nao for familiarizado com as diversas ferramentas de redes existentes. No presente trabalho, desenvolveu-se o Network eXperiment Tool (NeXT), uma arquitetura e ferramenta que ajuda a monitorar o tráfego gerado por aplicações distribuídas como as P2P em uma infraestrutura de rede. Com o uso do NeXT, diversas analises de um aplicativo podem ser realizadas e avaliadas, abstraindo a grande quantidade de operac?oes manuais necessárias para realizar tais experimentos.*

1. Introdução

Com as redes P2P possuindo milhares de nos, a tarefa de realizar experimentos para dimensionar algumas das suas características tem se tornado cada vez mais complexa e desafiadora [Mane et al. 2005]. Devido a sua dinamicidade e descentralização, o monitoramento externo de um sistema P2P é uma tarefa bastante complexa [Le Merrer et al. 2006]. Isso dificulta a obtenção continuada de diversos dados e informações acerca da rede. Tais dados podem ser cruciais no cálculo de alguns parâmetros do sistema, como os pacotes de saída para estimar o tamanho de uma rede. Eles devem ser coletados em intervalos de tempo diferentes, devido às mudanças das características de rede ocorridas ao longo do tempo.

No presente trabalho desenvolveu-se uma arquitetura de *software* e uma ferramenta para obtenção de forma não intrusiva dos pacotes trafegados por um aplicativo em uma rede: o *Network eXperiment Tool* (NeXT). O aplicativo escolhido para a realização dos testes foi o SopCast. O seu protocolo cria uma rede P2P sobreposta à infraestrutura da rede operada, que neste trabalho será disponibilizada pelo PlanetLab[1]. O PlanetLab é um

[1]http://www.planet-lab.org

serviço que disponibiliza um conjunto de recursos que podem ser utilizados para testes e pesquisa em redes de computadores e sistemas distribuídos. Ele disponibiliza uma rede privada interligando todos os recursos (nós) e conta com um gerenciamento simplificado que pode ser realizado através de uma Interface de Programação de Aplicações (*Application Programming Interface*, API). O SopCast, por sua vez, é um aplicativo que possibilita a reprodução de *streamings* multimídia em um computador ou dispositivo móvel. Faz uso de um protocolo de comunicação privado e funciona utilizando tecnologias P2PTV para a transferência do conteúdo requisitado, como será detalhado nas demais seções.

2. Trabalhos Relacionados

Na literatura não é possível encontrar muitos trabalhos relacionados a modelos arquiteturais para experimentos. Entretanto, existem algumas ferramentas para propósitos gerais. Por exemplo, com o *Network eXperiment Engine* (NXE)[2] é possível criar uma topologia de rede qualquer, configura-lá e executar um dado experimento. Um experimento é descrito através de um esqueleto, que define um cenário como uma sucessão de datas nas quais os eventos ocorrem. Eles correspondem ao ponto de partida de uma ação (por exemplo, o início de uma nova transferência de dados em uma sessão *web*) combinados com os parâmetros relevantes para esta ação (por exemplo, a lei de distribuição de tamanhos dos arquivos) em um conjunto de *hosts*, cujo tamanho depende do tipo de aplicação que se quer modelar. A configuração de um experimento nessa ferramenta é bastante complexa, já que envolve a modelagem de todas as entidades envolvidas no experimento.

Outra ferramenta, o PlanetMon [Ruoso et al. 2011], se propõe a criar um ambiente para a realização de tarefas essenciais para a execução de um experimento no PlanetLab, incluindo o envio (*deploy*) da aplicação aos nós, a instalação do ambiente de execução e a coleta dos dados. Além disso, é possível monitorar o comportamento dos nós durante a execução do experimento, visualizando a topologia virtual criada através de um mapa. Ele, entretanto, depende de um serviço *web*, é específico para o PlanetLab e demanda muitas configurações para o seu funcionamento *offline*.

3. O Modelo

O modelo proposto é baseado no mecanismo utilizado nos experimentos encontrados em muitos trabalhos científicos. Por exemplo, [Rocha et al. 2014] e [Mane et al. 2005] utilizam um mesmo método para capturar pacotes difundidos numa rede privada por um aplicativo. De modo a prosseguir à validação e generalização dos métodos desenvolvidos, é necessário repetir os mesmos experimentos e eventualmente realizar outros, fazendo uso do mesmo método, porém utilizando outras aplicações. A arquitetura proposta se propõe a ajudar na criação de um ambiente para emular este mecanismo. A diferença mais sutil, entretanto, estará na possibilidade de se utilizar qualquer aplicativo no experimento.

3.1. O Mecanismo Arquitetural

A ideia geral consiste em capturar - por uma entidade chamada monitor - um pequeno subconjunto da população (nós), marcá-los e desconectá-los do meio. Após isso, um novo subconjunto de nós é capturado por um novo monitor. Ao repetir-se esse procedimento, diversos estimadores para o tamanho populacional podem ser inferidos dependendo das

[2]http://ens-lyon.fr/LIP/RESO/Software/NXE/

várias suposições tomadas sobre ambos o sistema e os procedimentos. Utilizando as operações de união às redes como uma primitiva para se obter amostras da população (os endereços IPs), diversos estimadores tem sido desenvolvidos [Mane et al. 2005]. A Figura 1 exemplifica de maneira simplificada como é o funcionamento do método acima a partir da sua emulação em uma infraestrutura e de uma aplicação. É com o entendimento deste mecanismo que uma ferramenta poderá ser utilizada para coletar os pacotes trafegados nessa infraestrutura.

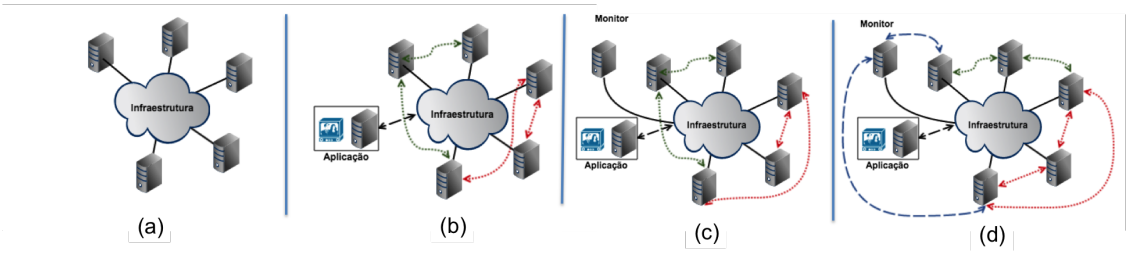

Figura 1. Ilustração do Mecanismo Arquitetural

Na Figura 1a pode-se observar que uma rede com cinco computadores é criada através da infraestrutura, que disponibiliza os recursos computacionais necessários para tal. A cada um destes recursos chama-se nó. Uma aplicação é executada externamente à rede criada e todos os nós consomem o que ela produz. Para simplificar a figura e a representação das conexões dos nós ao servidor, apenas uma seta bidirecional contínua partindo do servidor à rede foi representada (Figura 1b).

Uma listagem com alguns nós conectados à infraestrutura é obtida a cada nova conexão. De posse dessa listagem, uma rede P2P sobreposta à infraestrutura é formada a partir das conexões dos nós uns com os outros. Isso pode ocorrer para evitar uma sobrecarga no tráfego passante pela aplicação, além de garantir a escalabilidade geral do sistema. Essa sobreposição pode ser visualizada na figura a partir das setas tracejadas entre os nós. Um sexto nó junta-se a rede de maneira separada e independente (Figura 1c). Este nó em especial será chamado *Monitor*. O monitor é um nó cuja única função é se juntar e permanecer conectado à rede por um intervalo de tempo limitado (Figura 1d). Isso é feito para que uma coleta, igualmente limitada, dos pacotes que trafegam na rede possa ser realizada e posteriormente analisada. Na figura, isso ocorre apenas entre os vizinhos do monitor. Após o intervalo de tempo, o monitor se desconecta da rede e segue para o passo de pós processamento e análise dos pacotes que trafegaram a partir dele.

Todas as vezes que um monitor se conectar a rede, teoricamente ele receberá da aplicação uma listagem diferente com nós para se conectar e começar a interagir na rede. Para aumentar essa aleatoriedade, propõe-se que mais de um monitor possa ser utilizado ao fim de cada ciclo de captura. Dessa forma, diminui-se a incidência dos mecanismos de seleção não aleatórios (como pelo acesso ao histórico de conexões) sejam utilizados pela aplicação, já que cada novo monitor irá possuir um endereço diferente. Mecanismos desse tipo são usados por aplicações P2P como o SopCast, que os utilizam a fim de aumentarem a eficiência de propagação dos pacotes propagados na rede P2P formada [Mnie-Filali et al. 2013].

4. Arquitetura

A Figura 2a expõe de maneira detalhada o que ocorre em um nó da rede no momento em que um cliente externo, através da ferramenta, inicia remotamente a sua execução. Nela pode-se observar várias etapas sequenciais, delimitadas e organizadas por algarismos romanos. Cada um desses agentes tem conhecimentos limitados acerca da existência dos outros agentes e o que eles fazem, como mostrado pelos três retângulos delimitantes. Há também fluxos de um agente para outro através das setas direcionais que geram alguns arquivos de saída (como o *node.pcap* e o *node.txt*). Os agentes participantes nessas etapas são compostos pelo Usuário, um Nó, um *script* (*Shell*), a ferramenta de captura de pacotes (*Tshark*), um executável programado pelo Usuário e uma aplicação (*Application*), invocada opcionalmente pelo executável do usuário.

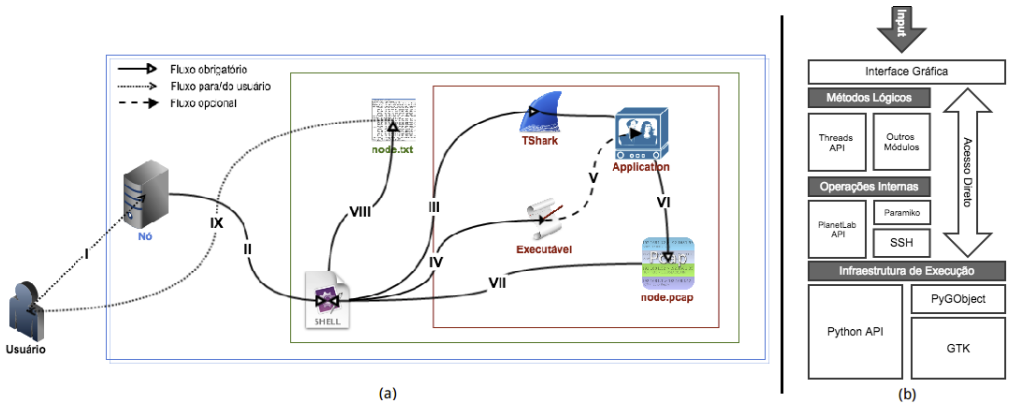

Figura 2. (a) Etapas executadas por um nó; (b) Arquitetura lógica da ferramenta

Na etapa (I), a ferramenta conecta-se remotamente a um nó através de um canal de comunicação seguro (por exemplo, o SSH), disponível em todos os nós da rede. Após o estabelecimento da conexão, a primeira coisa que a ferramenta faz - etapa (II) - é executar um *script* que possui três funções: capturar os pacotes gerados pelo executável fornecido pelo usuário (etapa III), executá-lo (etapa IV) e filtrar os pacotes gerados para um arquivo texto. O executável do usuário pode opcionalmente invocar uma aplicação externa, como um cliente SopCast (indicado pela seta tracejada na etapa V). Todo o tráfego gerado pela etapa (IV) é armazenada no arquivo *node.pcap* (etapa VI), utilizado pelo *script* para filtrar os pacotes (etapa VII) e gerar um arquivo texto contendo os endereços IP contatados pelo Nó, através do executável do usuário (etava VIII). A nomeação deste arquivo texto é dada utilizando o *hostname* do nó (no exemplo da Figura 2a, node.txt). Por fim, na última etapa (IX) esse arquivo é obtido pelo Usuário através da ferramenta.

Todas as etapas explicadas no parágrafo anterior são executadas pelos nós e monitores, pois todos possuem o mesmo conjunto de ferramentas e executam as mesmas operações. A única diferença reside no fato do monitor se conectar separadamente e permanecer conectado por tempo limitado na rede, além de opcionalmente invocar um executável diferente do que é invocado em um nó. Essa homogeneização dos nós é realizada pela ferramenta, que envia o conjunto de aplicações para todos eles antes de iniciar o processo descrito na Figura 2a. A partir desta simplificação, o usuário da ferramenta precisa apenas selecionar os *hosts* que serão nós e os que serão monitores, além de escolher o que será executado em cada um deles.

4.1. Uma Implementação: o NeXT

Apesar de todas as etapas serem complexas se executadas manualmente, com a utilização de uma ferramenta que abstraia todo o processo, elas se tornam simples. Isso se dá pois desenvolveu-se uma interface gráfica para utilização da mesma, facilitando assim a configuração e gerenciamento dos nós, que são ofertados pelo PlanetLab. Nesta subseção, apresenta-se a ferramenta desenvolvida, que implementa a arquitetura apresentada na subseção 3, chamada *Network eXperiment Tool* (NeXT).

A ferramenta foi totalmente desenvolvida em *Python*, uma linguagem de programação de alto nível, interpretada, orientada a objetos e de tipagem dinâmica e forte que possui recursos poderosos em sua biblioteca padrão e diversos módulos e *frameworks* desenvolvidos por terceiros. Um desses, é o *GNOME ToolKit* (GTK)[3]. O GTK é um *framework* multi-plataforma para a criação de interfaces gráficas que possui integração (*bindings*) com Python. O *framework* em si, entretanto, é escrito em C e seu *design* é orientado a objetos com base no sistema de objetos da biblioteca Glib[4]. A facilidade da codificação de um *software* em *Python*, unida com a grande variedade de funcionalidades de interface gráfica providas pelo GTK, subsidiam o uso dessas ferramentas para o desenvolvimento do NeXT.

Entretanto, além do módulo da Interface Gráfica, é necessário o acesso aos servidores do PlanetLab que disponibilizarão a infraestrutura utilizada pela ferramenta, junto com a autenticação. Ademais, outros módulos de configuração e acesso a arquivos se fazem necessários. A Figura 2b ilustra a arquitetura lógica que o *NeXT* segue internamente.

Na camada inferior, reside a *Infraestrutura de Execução*. Nesta camada há todo o aparato de bibliotecas disponibilizado pela API do *Python* e a biblioteca PyGObject[5] (*binding* do GTK para Python) que exibe a interface gráfica. As bibliotecas internas do *Python* incluem métodos de acesso, navegação, leitura e armazenamento no sistema de arquivos da plataforma corrente. Já o GTK apresenta toda a abstração para desenhar janelas no servidor de vídeo do sistema operacional, além de capturar e responder aos eventos de interação do usuário com a interface, como cliques do mouse e tratamento da entrada de dados (*input*).

Por sua vez, a camada de *Operações Internas* fornece os mecanismos de identificação, autenticação, obtenção de informações e atualização dos nós pertencentes ao *Slice* do PlanetLab. Além disso, toda a manipulação remota que existe como logon, envio e recebimento de arquivos e execução de comandos ocorre através da biblioteca *Paramiko*[6], que fornece uma interface em Python para o SSH, o protocolo utilizado para comunicação com os nós.

Por fim, os *Métodos Lógicos* exibem para a Interface Gráfica o que pode ser realizado, abstraindo dela o interior do funcionamento das camadas inferiores. Por exemplo, a interface pode requisitar que todas as conexões remotas com os nós sejam efetuadas em paralelo, utilizando *threads*. Sem essa funcionalidade, a inicialização dos experimentos seria sequencial e demandaria muito tempo caso eles envolvessem muitos nós. Nessa ca-

[3]http://www.gtk.org
[4]http://developer.gnome.org/glib/
[5]https://wiki.gnome.org/PyGObject
[6]http://www.paramiko.org

mada também há a implementação das interfaces que abstraem as demais camadas, como o capturador de eventos da interface gráfica, métodos da API do PlanetLab e os comandos de configuração da ferramenta. Tudo isso é ilustrado como *Outros Módulos* na Figura 2b.

É importante ressaltar que o módulo de Interface Gráfica é o único que possui o acesso direto à camada de execução. Isso ocorre pois ela requer que alguns comandos sejam executados de forma independente, sem a necessidade de intermediação entre terceiros, o que ocasionaria complexidades desnecessárias na sua implementação. Logicamente, ela também é quem interage diretamente com o usuário, exibindo símbolos e signos em formas de ícones e janelas para o usuário.

4.2. Download, Instalação e Requisitos do NeXT

É necessário primeiramente adquirir a ferramenta, disponibilizada através da rede social para desenvolvedores GitHub[7] em *http://github.com/abelpc/next*. Após a aquisição do arquivo, basta expandi-lo em qualquer diretório e executar o arquivo executável *NeXT*. Como todo pacote de *software*, o *NeXT* possui diversas dependências que o compõem e que precisam estar instaladas afim de proverem o correto funcionamento da ferramenta. Ele foi desenvolvido e testado no Sistema Operacional Ubuntu 14.04, utilizando a versão 2.7.5 da linguagem *Python* em conjunto com o *Framework* de Interface *Gráfica GNOME ToolKit* (GTK) versão 3.10 e a biblioteca PyGObject 3.0. Para a conexão remota foi utilizado o OpenSSH versão 2 e a biblioteca *Paramiko*, versão 1.15. Para coletar os pacotes nos nós, utilizou-se o *Wireshark* 1.10. A leitura e análise destes pacotes foram, respectivamente, feitas utilizando o *Impacket* 0.10 e o *Pcapy* 0.9. Para o módulo URL, utilizou-se o *urllib* versão 2. Ao executar-se o binário *NeXT*, logar-se no PlanetLab e carregar os nós, a tela da Figura 3 é exibida para seleção dos monitores e nós em um experimento.

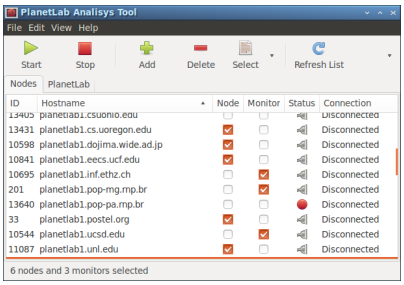

Figura 3. Janela de seleção dos nós

5. Um exemplo de experimento

Essa seção irá descrever um experimento realizado que pode ser reproduzido no salão de ferramentas do SBRC 2015.

O NeXT é uma implementação que executa sobre a infraestrutura do PlanetLab. Dessa forma, para que os experimentos possam ser devidamente executados, um ambiente mínimo de configuração foi constituído no PlanetLab, atraves do seu website *website*[8]. O *Slice* disponibilizado para os experimentos permite o uso simultaneo de até 100 nós,

[7]http://github.com
[8]http://www.planet-lab.org

número suficiente para o experimento aqui proposto. Por isso, 60 nós foram adicionados ao slice, de modo que se disponibilizasse nós adicionais em caso de problemas técnicos com alguns nós no momento dos experimentos.

Além disso, o Servidor SopCast foi executado no Sistema Operacional Ubuntu 14.04. Essa plataforma possui um processador Intel Core i5 de 2.5 GHz, 2 GB de memória RAM e 20 GB de disco rígido e estava conectada diretamente a Internet com velocidade de *download* de 5 Mbps (e 1 Mbps de upload), sem *firewalls* ou bloqueios (como servidores *proxy*). Um canal privado foi configurado no *website* do SopCast, impossibilitando a influência de terceiros na rede sobreposta criada pelo Servidor: apenas quem possui a identificação do canal pode conectar-se a ele. O canal possui uma taxa de transmissão sugerida de 512 Kbps. O *stream* difundido através do Servidor é um vídeo no formato WMV com 200 MB de tamanho, possuindo uma taxa de 30 quadros por segundo e uma resolução de 640 por 480.

A partir da configuração acima, diversos experimentos foram realizados, mas apenas um será mostrado aqui. Neste especificamente, 50 nós foram independentemente executados. Desses, um nó era o Servidor SopCast. Além disso, cinco diferentes máquinas foram usadas como monitor no experimento, porém apenas uma delas se conectava por vez. Assim, no experimento de tamanho 50, haviam 48 nós, um servidor e um dos monitores conectados por vez através da ferramenta. Excetuando-se o servidor SopCast, todos os outros nos foram executados a partir do PlanetLab.

Em todos os experimentos, uma rede sobreposta é formada pelo servidor e pelos nós que se conectam a ele logo depois da sua inicialização. O servidor opera através do *Broker* (agente negociador) do SopCast, que o associa ao endereço IP (e a porta 3912) do *host* local. Todos os nós unem-se à rede SopCast logo depois que os nós a formam. Todos eles recebem o *streaming* inicial do servidor e dos vizinhos que logo se formam através da porta 3908. Os monitores permanecem conectados a rede, cada um, por apenas 10 segundos, quando então se desconectam. Entre a saída de um monitor e a entrada de outro, transcorre-se um período de 60 segundos. Isso foi configurado para que o Servidor possa identificar que aquele nó não mais pertence a rede.

Em todos os nós, monitores ou não, o tráfego de saída é capturado e pós processado para recebimento. Ou seja, apenas os pacotes com endereços IP igual ao do host (nó) e as portas de origem e destino sendo a 3908 é que são colhidos. Com um *script* Python, esses dados são organizados e novamente processados, para que uma Matriz de Adjacência seja montada. Essa matriz irá representar todas as vizinhas constituídas na rede SopCast. Arquivos semelhantes são gerados em todas as máquinas, e após finalizado os experimentos, eles são transmitidos ao ponto central do experimento para ser processado.

Para esse experimento, plotou-se um gráfico com os endereços IP de cada um dos nós no eixo horizontal e, no eixo vertical, a relação do número de vizinhos do nó com o número total de nós na rede SopCast (agora identificado como razão de vizinhança). Além disso, a média aritmética dessas relações (em vermelho) corta horizontalmente os gráficos. No eixo horizontal de cada gráfico, os 5 primeiros endereços IP (204.56.0.138, 129.93.229.138, 200.19.159.35, 204.123.28.51 e 140.247.60.126) identificam os monitores. Os demais endereços identificam os nós (do endereço 128.111.52.64 ao 72.36.112.71,

sem a exibição de todos), que permanecem por todo o experimento conectados a rede SopCast.

A Figura 4 abaixo ilustra o resultado para a execução do experimento. Presumidamente, os nós possuem as maiores relações de vizinhança, com muitos acima da média. Nós com mais de 50 vizinhos (e portanto, com razão de vizinhança acima de 1) representam os nos que trocaram informações com um ou mais monitores. Nota-se que nesse experimento apenas 5 nós não trocaram informações com os monitores. Percebe-se ainda que os monitores tiveram uma relação de vizinhança bem abaixo da média. Isso explica-se pelo pouco tempo que eles permanecem conectados ao *streaming* (10 segundos). Apesar disso, muitas conexões foram realizadas nesse curto espaço de tempo.

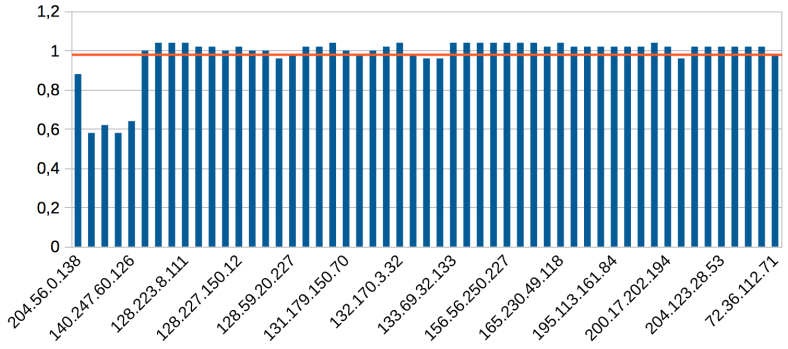

Figura 4. Gráfico de Razão de Vizinhança em um experimento com 50 nós

6. Conclusões

O *NeXT* possibilita o estudo, análise e teste dos dados gerados por um aplicativo em uma rede. Ele fornece uma configuração independente do serviço que se quer avaliar. Ao permitir que o usuário configure o quê e como um aplicativo será operado, ela possibilita que um experimento seja facilmente testado e executado para qualquer tipo de rede. A sua Interface Gráfica abstrai diversas configurações que são necessárias para que um experimento dessa proporção seja efetuado.

Referências

Le Merrer, E., Kermarrec, A.-M., and Massoulie, L. (2006). Peer to peer size estimation in large and dynamic networks: A comparative study. In *15th IEEE International Symposium on High Performance Distributed Computing, 2006*, pages 7–17.

Mane, S., Mopuru, S., Mehra, K., and Srivastava, J. (2005). Network size estimation in a peer-to-peer network. *University of Minnesota, MN, Tech. Rep*, pages 05–30.

Mnie-Filali, I., Rouibia, S., Lopez-Pacheco, D., and Urvoy-Keller, G. (2013). On the impact of the initial peer list in p2p live streaming applications: the case of sopcast. volume 1, pages 142–150. Horizon Research Publishing, USA.

Rocha, A. A. A., Figueiredo, D. R., and Towsley, D. (2014). Estimating the size of peer-to-peer systems from the outside. *Em processo de submissao*.

Ruoso, V. K., Bona, L. C. E., and Jr, E. P. D. (2011). Planetmon: Um arcabouço para execução e monitoração de experimentos no planet-lab. volume 11, pages 31–43. 7o Workshop de Redes Dinamicas e Sistemas Peer-to-Peer (WP2P).