LAURO LUIS COSTA

PRECISÃO E REPETIBILIDADE DE EXPERIMENTOS NO PLANETLAB

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Elias P. Duarte Jr.
Co-Orientador: Luis C. E. Bona

CURITIBA

2014

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CPU	*Central Processing Unit*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
IPSec	*IP Security Protocol*
IPv6	*Internet Protocol version* 6
KB	*Kilobytes* (1024 *bytes*)
MB	*Megabytes* (1024 *Kilobytes*)
ms	milissegundo
P2P	*Peer-to-Peer*
PSSH	*Parallel Secure Shell*
RTT	*Round-Trip Time*
SHA	*Secure Hash Algorithm*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

RESUMO

Desenvolvida como uma rede acadêmica, a Internet cresceu e evoluiu rapidamente. Hoje, 40 anos mais tarde, se tornou uma ampla plataforma de informação, interação social e comércio. Os usuários e as aplicações da Internet atualmente demandam características de desempenho, segurança, escalabilidade e mobilidade que não foram previstas no momento de sua criação. *Testbeds* de larga escala, como o PlanetLab, são peças chave para o desenvolvimento e avaliação de novas aplicações e arquiteturas que atendam a estas demandas. O PlanetLab é um *testbed* de escala planetária, composto por mais de mil nodos espalhados ao redor do mundo, que oferece aos seus usuários um ambiente real para a execução de experimentos. No entanto, a execução de experimentos no PlanetLab é uma atividade que pode se tornar muito complexa, especialmente por envolver uma grande quantidade de nodos e a existência de instabilidades no desempenho nos nodos e na rede que os conecta, prejudicando a precisão e repetibilidade dos resultados obtidos. Por estes motivos, existem diversas ferramentas para gerenciamento de experimentos e descoberta de recursos no PlanetLab. Neste trabalho, apresentamos uma avaliação experimental do impacto da utilização de subconjuntos de nodos selecionados por uma ferramenta de monitoramento da conectividade entre os nodos na precisão e repetibilidade dos resultados obtidos. São realizados experimentos utilizando aplicações com diferentes perfis de utilização de recursos e os resultados obtidos por diferentes subconjuntos de nodos são comparados. A estratégia de seleção de nodos estudada reduziu a variação dos resultados obtidos em até 27% e obteve média de execução até 26% mais baixa que uma das estratégias alternativas. Pode-se concluir que a utilização de subconjuntos de nodos selecionados por esta ferramenta contribui para a precisão, repetibilidade e reprodutibilidade de experimentos realizados no PlanetLab. Este trabalho também apresenta uma proposta de integração da ferramenta de seleção de nodos ao portal de gerenciamento de experimentos PlanetMon, com o objetivo de permitir que usuários do PlanetLab obtenham acesso a ferramenta de seleção de modo conveniente e transparente enquanto gerenciam seus experimentos.

ABSTRACT

The Internet was originally developed as an academic network more than four decades ago. Today, it has established itself as a global platform for human communications, allowing information exchange, social interaction, and e-commerce. Current Internet users and applications require high levels of performance, security, scalability and mobility. These characteristics were not predicted at the time of its creation. Large-scale testbeds like PlanetLab have been developed to allow the design and realistic evaluation of applications and architectures to supply the new Internet demands. PlanetLab is a planetary scale testbed, consisting of more than one thousand nodes spread around the globe, offering its users a realistic environment for experiment execution. However, experiment execution on PlanetLab can become a complex activity, especially because it involves configuring a large number of nodes, and because the environment is highly unstable, due to performance variations of both nodes and their network connections. These instabilities affect the precision and repeatability of the results obtained. There are several tools for experiment management and resource discovery on PlanetLab. In this work, we present an experimental evaluation of the impact of using subsets of nodes selected with different strategies on the precision and repeatability of the results obtained. Experiments using applications with different resource requirements were carried out and are reported. Results show that the selection strategy based on k-cores reduces the variation on the results obtained by up to 27% and resulted on an average execution time up to 26% faster compared to other alternatives. The utilization of subsets of nodes selected with this strategy can thus contribute to the precision, repeatability and reproducibility of experiments executed on PlanetLab. This work also presents the integration of the node selection strategy to the experiment management framework PlanetMon. This integration is intended to allow PlanetLab users to have access to the node selection tool in a convenient and transparent way for managing their experiments.

CAPÍTULO 1

INTRODUÇÃO

Desde seu surgimento, há mais de 40 anos, a Internet vem crescendo e evoluindo rapidamente, de uma rede acadêmica para uma ampla plataforma de informação, interação social e comércio, afetando organizações e indivíduos nas mais diversas formas. As aplicações desenvolvidas, e seu uso, mudaram drasticamente a natureza do tráfego na Internet. Aplicações como a Web, compartilhamento de arquivos, comunicação por voz, *streaming* de vídeos, entre outros, demandam características de desempenho, segurança, escalabilidade e mobilidade que não foram previstas na criação da Internet [16].

Nos últimos anos, pesquisadores propuseram diversas soluções incrementais para atender a estas demandas. IPSec e MobileIP [17], além de principalmente o IPv6 [9], são alguns exemplos. Ao invés da criação de soluções incrementais, muitos pesquisadores acreditam que é necessário recriar a Internet do princípio, sem nenhuma limitação imposta pelos padrões e princípios atuais. Esta abordagem é conhecida como *Clean-Slate Internet* [12]. Mesmo que essa estratégia radical de criação de uma arquitetura completamente nova não possa substituir completamente a Internet, este esforço de pesquisa tem o potencial de trazer resultados para a evolução da rede.

Soluções incrementais, ou mesmo a reconstrução completa da Internet, precisam ser testadas em ambientes de larga escala. *Testbeds* de larga escala e *redes experimentais* são chave para o desenvolvimento de novas soluções. Entre os principais exemplos de *testbeds* estão as redes GENI [28], FIRE [26], FIBRE [25] e PlanetLab [7, 41], este último o foco deste trabalho.

O PlanetLab é uma plataforma aberta para desenvolvimento, implantação e acesso a serviços de escala planetária. Um conjunto de instituições acadêmicas, industriais e governamentais formam o consórcio responsável por garantir o crescimento e manutenção da infraestrutura de hardware do PlanetLab. Instituições se juntam ao consórcio adi-

cionando e gerenciando nodos na rede do PlanetLab. Em contrapartida, pesquisadores destas instituições têm acesso a recursos do conjunto de nodos desta rede, chamados *slices*, para a execução de experimentos. O PlanetLab é utilizado por pesquisadores desde 2003 e, atualmente, é composto por cerca de 1206 nodos em 593 locais ao redor do mundo. Os nodos são *hosts* TCP/IP que se comunicam através da Internet.

O PlanetLab oferece ao usuário um portal Web com suporte básico ao pesquisador para possibilitar o acesso aos recursos do *testbed*. Neste portal Web, o pesquisador pode incluir ou remover nodos de seus *slices* e obter acesso remoto a estes nodos através de um terminal virtual. Porém, diversas atividades comuns ao planejamento e execução de experimentos, como instalar pacotes de software necessários, atualizar a aplicação sendo testada, monitorar o progresso do experimento sendo realizado, entre outras atividades, devem ser executados pelo próprio pesquisador, individualmente em cada nodo. Estas atividades não são triviais, em especial para pesquisadores com pouca experiência na realização de experimentos neste ambiente. Mesmo para pesquisadores mais experientes, a quantidade de trabalho necessário para a configuração é muito significante e estas atividades se tornam bastante complexas quando envolvem uma grande quantidade de nodos.

Diversas ferramentas desenvolvidas por usuários são compartilhadas com toda a comunidade usuária do PlanetLab. Alguns exemplos de ferramentas para o gerenciamento de *slices* são o PlMan [40], Gush [27] e Stork [44], entre outros. O foco deste trabalho está na ferramenta de gerenciamento de *slices* PlanetMon. O PlanetMon é um *framework* proposto para simplificar a instalação, execução e monitoramento de experimentos no PlanetLab. A ferramenta permite que o usuário configure seu *slice* e monitore seus experimentos através de uma interface Web. Esta ferramenta foi proposta em sua versão original em [18]. Neste trabalho realizamos modificações na interface e no fluxo de execução do PlanetMon, integrando a ele uma ferramenta de seleção de nodos para a realização de experimentos.

Pesquisadores necessitam de um ambiente real para a execução de seus experimentos e avaliação de suas propostas. É necessário que o ambiente reproduza condições reais, tais

como congestionamento e perdas ocasionais de conectividade. Entretanto, frequentemente é necessária a existência de um grupo de nodos que apresentem um nível razoável de estabilidade entre si. Não é trivial encontrar tal grupo de nodos no PlanetLab [13]. Existem serviços sendo executados continuamente no PlanetLab que fornecem informações e estatísticas da infraestrutura da própria rede do PlanetLab. São exemplos deste tipo de serviço o CoMon [33], MON [32] e o SWORD [2]. O foco deste trabalho está no serviço de monitoramento da comunicação par-a-par dos nodos proposto em [10, 13]. Este serviço oferece uma ferramenta de seleção de um conjunto de nodos que atendam a parâmetros especificados pelo usuário.

A rede do PlanetLab é um ambiente altamente instável [5], apresentando grandes variações no tempo de resposta e largura de banda disponível para a comunicação entre os nodos, e também variações na disponibilidade de recursos em cada nodo, uma vez que alguns nodos podem ser utilizados por vários pesquisadores simultaneamente, concorrendo por recursos. Apesar destas variações refletirem o comportamento da própria Internet, muitas vezes pesquisadores precisam de maior controle sobre estas variações. Por exemplo, um pesquisador pode estar interessado em testar sua aplicação em um ambiente com características específicas, ou ainda, desejar manter certas características entre diferentes execuções de seus experimentos, de forma que seja possível comparar os resultados obtidos com maior precisão.

Existem diversos trabalhos relacionados à obtenção de experimentos repetíveis em um ambiente real. Em alguns casos, a abordagem utilizada é tornar mais controlável um ambiente real, mas outra abordagem possível é tornar um ambiente controlado mais real. Em [19], os mesmos autores dos artigos originais do PlanetLab afirmam que é muito difícil reproduzir os resultados obtidos no PlanetLab devido às constantes mudanças nas características da rede e dos nodos. No entanto, os autores sugerem que a seleção de nodos estáveis é uma prática aconselhável para se obter maior controle sobre as variações presentes no *testbed*.

A principal contribuição desta dissertação é apresentar uma avaliação experimental do quanto as instabilidades presentes no PlanetLab afetam os resultados de experimentos

realizados e, como a utilização de nodos selecionados através da ferramenta de monitoração das interações fim-a-fim pode reduzir o impacto das variações comuns ao PlanetLab, aumentando a precisão dos experimentos. Também pretende-se demonstrar que o compartilhamento dos parâmetros de seleção de nodos utilizados na ferramenta de seleção é suficiente para a obtenção de um conjunto de nodos com características de estabilidade similar, contribuindo para a repetibilidade e reprodutibilidade dos experimentos.

Nos experimentos realizados neste trabalho, os nodos selecionados pela ferramenta de monitoramento estudada apresentaram melhor desempenho e maior consistência nos resultados obtidos. A média do tempo de execução das aplicações foi até 26% mais baixa que o resultado utilizado por um conjunto de nodos selecionados com uma estratégia alternativa. A variação entre os resultados obtidos durante um período de 40 dias foi avaliada, e os nodos selecionados pela ferramenta de monitoramento obtiveram um coeficiente de variação mais baixo quando comparados aos resultados utilizando nodos da estratégia alternativa.

Os resultados obtidos nos experimentos realizados também demonstraram que a ferramenta de monitoração da estabilidade das conexões fim-a-fim é adequada tanto para aplicações com intensa troca de mensagens quanto aplicações distribuídas com intenso uso de CPU. Os resultados demonstram ainda que a ferramenta é capaz de selecionar conjuntos de nodos com características de estabilidade equivalente, apenas utilizando os mesmos parâmetros de seleção.

O restante deste trabalho está organizado da seguinte forma. O capítulo 2 descreve estratégias para a execução de experimentos no PlanetLab, apresentando algumas das principais ferramentas disponíveis aos seus usuários. O capítulo 3 aborda a dificuldade de se obter precisão nos experimentos realizados no PlanetLab, descreve uma ferramenta para a seleção de nodos estáveis e apresenta uma proposta de integração da ferramenta de seleção de nodos e o portal de gerenciamento de experimentos PlanetMon. O capítulo 4 descreve e apresenta o resultado dos experimentos realizados para avaliar a contribuição da ferramenta de seleção de nodos na precisão dos experimentos realizados. Em seguida são apresentadas as conclusões.

CAPÍTULO 2

EXECUÇÃO DE EXPERIMENTOS NO PLANETLAB

Este capítulo descreve a execução de experimentos no PlanetLab. O capítulo inicia com uma introdução a alguns dos principais *testbeds* de larga escala, seguida de uma descrição mais aprofundada do PlanetLab. Na seção seguinte, são descritas ferramentas e serviços oferecidos aos usuários do PlanetLab para facilitar o planejamento e a execução de experimentos.

2.1 *Testbeds* de Larga Escala

Novas arquiteturas, protocolos e serviços propostos para a Internet do futuro precisam ser testados em um ambiente real, de larga escala e que ofereça condições reais. Portanto, *testbeds* de larga escala são essenciais para a validação destes novos protocolos e serviços. Diversos esforços mundiais têm sido propostos com o objetivo de construir *testbeds* de larga escala. A seguir, alguns exemplos de *testbeds* de larga escala desenvolvidos em programas colaborativos internacionais são apresentados, como GENI e FIRE, seguida de uma descrição mais aprofundada do *testbed* PlanetLab.

Global Environment for Network Innovations (GENI) [28] é um programa colaborativo iniciado em 2006, com o suporte da *National Science Foundation* (NSF) [36], a principal fundação norte-americana de suporte a projetos científicos. O GENI iniciou com o suporte de projetos existentes em um estrutura de rede dedicada, mas utiliza um sistema de federação para que a rede seja expandida com a integração de *testbeds* existentes. Entre os *testbeds* agregados ao GENI, está o PlanetLab. Entre outras tecnologias inovadoras, o GENI disponibiliza um *backbone* OpenFlow [38].

O desenvolvimento do GENI acontece em várias etapas. Os objetivos principais na fase atual de desenvolvimento são evoluir o protótipo atual e oferecer melhores ferramentas e serviços para aumentar o número de experimentos realizados na rede.

Future Internet Research and Experimentation (FIRE) [26] é um projeto de *testbed* da União Europeia iniciado em 2006 e pode ser considerado uma versão europeia equivalente ao GENI. O FIRE utiliza um sistema de federação para unificar diferentes *testbeds* existentes na Europa e sua expansão baseia-se em gradualmente adicionar novos e existentes *testbeds*. Entre os *testbeds* agregados ao FIRE está a porção europeia do PlanetLab (PLE).

Estes programas colaborativos internacionais não estão restritos à América do Norte e Europa, existem diversas alternativas na Ásia e mais recentemente no Brasil. *Japanese Gigabit Network* (JGN2plus) [48] e AKARI [47] podem ser consideradas versões japonesas equivalentes ao FIRE e GENI. Na Coreia do Sul existe a rede nacional de tecnologia e pesquisa KREONET [30], que está conectada à rede global de pesquisa e desenvolvimento GLORIAD [29]. Esta rede 10Gbps contorna o globo envolvendo 15 países do hemisfério norte. No Brasil, existe o *Future Internet Experimentation Between Brazil and Europe* (FIBRE) [25], um projeto colaborativo entre Brasil e Europa iniciado em 2011 e relacionado ao FIRE.

O foco deste trabalho está na realização de experimentos no PlanetLab, portanto o restante desta seção é dedicada a descrever com mais detalhes este *testbed*. O PlanetLab [7] é formado por um consórcio composto por instituições acadêmicas, industriais e governamentais que cooperam no suporte da rede do *testbed*. Este consórcio é responsável pela definição das políticas de uso adequado do *testbed* e pelo crescimento a longo prazo da infraestrutura de hardware do PlanetLab.

A rede do *testbed* PlanetLab é formada por uma coleção de máquinas mantidas por instituições de pesquisa. Estas máquinas estão amplamente distribuídas ao redor do mundo, sendo atualmente cerca de 1206 nodos em 593 locais diferentes. Os nodos são *hosts* TCP/IP que se comunicam através da Internet. Cada um destes nodos executa um pacote de software que inclui: um sistema operacional baseado em Linux, mecanismos para a inicialização de nodos e distribuição de atualizações de software, entre outras ferramentas para o monitoramento do funcionamento e utilização dos recursos dos nodos.

Os pesquisadores das entidades membros do consórcio PlanetLab têm acesso à rede

para execução de experimentos. O pesquisador usuário do PlanetLab tem acesso a um *slice*. Um *slice* é um conjunto de recursos distribuídos alocados na rede do PlanetLab. Na prática um *slice* significa, para o usuário, acesso remoto a máquinas virtuais Unix nos nodos da rede.

O PlanetLab oferece um portal Web onde os usuários podem gerenciar seus *slices*. O pesquisador deve criar um par de chaves pública e privada, e adicionar a chave pública de seu par no portal do PlanetLab. Este par de chaves é utilizado para autenticar o acesso do usuário aos nodos de seu *slice*. Neste portal, o usuário pode visualizar uma lista de nodos disponíveis no PlanetLab e definir quais nodos devem fazer parte de seu *slice*. Quando um novo nodo é adicionado ao *slice*, uma máquina virtual Unix é criada neste nodo. O portal envia a chave pública do usuário à máquina virtual deste nodo, tornando-a disponível ao pesquisador para acesso remoto.

O acesso aos nodos é realizado através do *Secure Shell* (SSH) [21] e autenticado através do par de chaves pública e privada do usuário. O SSH é a ferramenta padrão para aplicações do tipo terminal virtual seguro, sendo amplamente utilizado para a administração remota de servidores. Ao conectar-se a um nodo do PlanetLab utilizando SSH, o usuário obtém acesso remoto a um terminal virtual no ambiente Unix de seu *slice* neste nodo.

O portal Web do PlanetLab oferece o suporte básico para que o pesquisador tenha acesso aos recursos do *testbed*: incluir nodos em um *slice* e enviar a chave pública do usuário para estes nodos, garantindo o acesso remoto através de um terminal virtual. Entretanto, diversas atividades comuns ao planejamento e execução de experimentos são responsabilidade do usuário: instalar pacotes de software necessários, atualizar a aplicação sendo testada, monitorar o progresso do experimento, entre outras atividades, devem ser executadas pelo próprio usuário, *individualmente em cada nodo*. Estas atividades não são triviais e aumentam em complexidade quando envolvem uma grande quantidade de nodos. Para simplificar a realização de experimentos no PlanetLab, diversas ferramentas e serviços foram propostos pela própria comunidade usuária do *testbed*. Algumas destas ferramentas e serviços serão descritas na seção seguinte.

2.2 Ferramentas e Serviços para Usuários do PlanetLab

O portal Web para usuários do PlanetLab oferece apenas as funções básicas para a execução de experimentos. Como descrito na seção anterior, as atividades comuns de planejamento e execução de experimentos pode atingir um alto nível de complexidade, devido à quantidade de nodos que podem fazer parte de um *slice* no PlanetLab. A comunidade de usuários do PlanetLab construiu e disponibilizou diversas ferramentas e serviços para reduzir a complexidade dessas atividades. As subseções seguintes descrevem algumas das principais ferramentas disponíveis para o planejamento e execução de experimentos no PlanetLab e serviços para a descoberta de recursos na rede.

2.2.1 PSSH

O OpenSSH [39] é uma implementação livre, e atualmente a mais utilizada, do protocolo SSH. Utilizando o OpenSSH, um usuário do PlanetLab pode se conectar remotamente a um nodo de seu *slice*. O PSSH [43] implementa uma versão paralela do OpenSSH. O PSSH não foi especificamente desenvolvido para o gerenciamento de nodos do PlanetLab, mas oferece conexão segura e paralela a múltiplos servidores. Caso o usuário necessite executar um comando específico em todos os nodos de seu *slice*, o PSSH pode ser utilizado para se conectar a este conjunto de nodos, definido em uma lista de endereços IP, e executar o comando de forma paralela.

O PSSH pode reduzir a complexidade de várias tarefas, mas grande parte do controle destas atividades continua sendo responsabilidade do usuário. Para utilizar o PSSH na manutenção de um *slice*, o usuário deve manter manualmente uma lista de endereços dos nodos que fazem parte do *slice*. O PSSH permite a execução de comandos em paralelo e mantém a saída dos comandos em arquivos de *log* individuais para cada nodo. Em uma rede como a do PlanetLab, é improvável que todos os nodos estejam disponíveis ao mesmo tempo e respondam ao comando enviado. É responsabilidade do usuário observar os *logs* e garantir que todos os nodos tenham executado o comando desejado. Por este motivo, a utilização do PSSH na manutenção de *slices* no PlanetLab pode se tornar muito complexa

e não é uma solução adequada para usuários com pouca experiência.

2.2.2 Nixes

A ferramenta Nixes [37] oferece um conjunto de *scripts bash* para a instalação e controle de aplicações no PlanetLab. O Nixes simplifica a manutenção de uma grande quantidade de nodos, permitindo ao usuário instalar pacotes ou executar comandos em múltiplos nodos, paralelamente. O usuário interage com a ferramenta em modo linha de comando e deve executar o *script* referente à tarefa desejada, indicando as opções na lista de argumentos. Por exemplo, para a execução em paralelo de um comando em todos os nodos de um *slice*, o usuário deve executar o *script* específico para comandos paralelos do Nixes, o *plcmd*. O comando desejado e seus parâmetros devem ser passados como argumentos na linha de comando. A saída do comando executado em cada nodo é armazenada pelo Nixes em um arquivo de *log*, que poderá então ser analisado pelo usuário.

2.2.3 Stork

Stork [44] é uma ferramenta para instalação de pacotes de software que pode ser utilizada no PlanetLab e também em outros ambientes. A ferramenta tem como objetivo simplificar a distribuição e instalação de pacotes de software em múltiplos *hosts*. O usuário do Stork pode definir grupos de nodos em seu *slice*, e enviar comandos de instalação de pacotes de software para o grupo de nodos em paralelo, sem a necessidade de acessar cada nodo individualmente.

O Stork busca fazer a transferência de arquivos de forma eficiente. Devido à sua arquitetura, o Stork permite que máquinas virtuais em um mesmo nodo compartilhem pacotes de software. Sua arquitetura é composta por um repositório central e *abrigos*. Abrigo é o local onde um nodo mantém cópias dos pacotes de software instalados em algum *slice* neste nodo. Se outro *slice*, neste mesmo nodo, necessita de um pacote de software presente no abrigo, este pacote pode ser obtido diretamente, sem a necessidade de se comunicar com o repositório central, reduzindo a necessidade de transferência de dados na rede. Caso o pacote requisitado não exista no abrigo, este é obtido no repositório

central e uma cópia é armazenada no abrigo, ficando disponível caso seja requisitado por outro *slice* no mesmo nodo.

2.2.4 PlMan

PlanetLab Experiment Manager (PlMan) [40] é uma ferramenta para simplificar tarefas básicas de gerenciamento de *slices* no PlanetLab. Entre estas tarefas básicas, estão a inclusão de nodos em um *slice*, a definição dos nodos utilizados em um experimento, a distribuição confiável dos arquivos necessários no experimento para todos os nodos, a execução de comandos paralelos nos nodos do *slice* e o monitoramento do progresso do experimento.

O objetivo do PlMan é facilitar a utilização do PlanetLab por pesquisadores sem muita experiência na construção de sistemas distribuídos. Estes usuários interagem com o PlMan através de uma interface gráfica, mas também existe a opção de utilização em modo linha de comando, criando *scripts* para executar blocos de comandos.

Seguem duas observações importantes sobre a ferramenta: a primeira é que o PlMan é uma aplicação Java que deve ser instalada e executada localmente no computador do usuário. A segunda observação é que esta aplicação não estava disponível para transferência na página do projeto.

2.2.5 MON

O *Management Overlay Network* MON [32] oferece duas funcionalidades principais: suporte à execução de tarefas de gerenciamento de *slices* e consulta de recursos disponíveis no *slice*. Similar ao PSSH, o MON permite ao usuário executar comandos ou enviar arquivos para uma lista de nodos em paralelo. Enquanto o PSSH se conecta diretamente a cada nodo, o MON cria uma estrutura sob demanda em forma de árvore, com o objetivo de reduzir a quantidade de conexões necessárias entre a máquina do usuário e os nodos do *slice*, otimizando a comunicação necessária para a execução de comandos ou envio de arquivos.

O MON também pode ser utilizado como um serviço para a descoberta de recursos na

rede. O usuário pode fazer consultas sobre o estado atual de seu *slice*, como por exemplo a carga do processador, quantidade de memória livre, uso de disco, quantidade de *slices* ativos no nodo, entre outros. O usuário pode, por exemplo, obter uma lista de nodos que apresentam uma carga no processador abaixo de um certo limiar e com pelo menos uma certa quantidade de memória livre. Esta lista pode então ser a base para definir quais nodos serão utilizados na execução do próximo experimento.

2.2.6 Gush

GENI User Shell (Gush) [1, 27] é uma ferramenta para configuração, gerenciamento e visualização de aplicações distribuídas em ambientes de larga escala. Gush é o sucessor do projeto Plush [42]. Ambos foram propostos inicialmente para suportar a execução de experimentos no PlanetLab, mas possuem extensões para suportar diversos ambientes de larga escala.

A arquitetura do Gush é distribuída. Uma aplicação cliente é executada em cada nodo do ambiente de experimentação. Estes nodos recebem comandos de um servidor central, chamado controlador. O controlador oferece uma interface ao usuário desenvolvedor da aplicação. O usuário interage com esta interface utilizando uma linguagem de descrição propostas pelos desenvolvedores do Gush. A proposta é baseada na abstração de "construção de blocos", especificando blocos simples, que combinados, podem descrever a execução de tarefas complexas.

O usuário da ferramenta pode, por exemplo, descrever um bloco de dependências de software, listando os pacotes de software necessários para a execução de um experimento. Utilizando a linguagem de descrição, o usuário deve indicar cada pacote de software necessário, onde cada pacote está disponível e seu método de instalação. Por exemplo, o usuário pode indicar a necessidade de um pacote disponível na Web e no formato *tar*, incluindo sua URL e os parâmetros de extração. O usuário também pode descrever os recursos necessários para a execução do experimento, incluindo a quantidade de nodos e outros parâmetros.

O usuário deve preparar um arquivo XML chamado especificação da aplicação. Neste

arquivo, blocos são combinados para descrever o fluxo do experimento. O programador da aplicação utiliza este arquivo de especificação da aplicação para interagir com o controlador. O controlador deve encontrar um conjunto de nodos que atenda às especificações do usuário, e então enviar mensagens para os clientes Gush de cada nodo, para que o fluxo do experimento descrito pelo usuário seja executado. O controlador tem duas formas para definir qual o conjunto de nodos disponíveis no ambiente atende às especificações do usuário. O controlador pode utilizar uma lista estática, fornecida pelo próprio usuário, ou se comunicar com serviços externos, como o SWORD ou CoMon, que serão descritos na subseção seguinte.

2.2.7 CoMon

O CoMon [33] é um serviço de monitoramento desenvolvido para o PlanetLab que oferece aos seus usuários informações referentes a nodos e *slices*. A arquitetura do CoMon é baseada em *daemons* espalhados pelos nodos do PlanetLab e um servidor central que recolhe as informações destes nodos e permite ao usuário executar consultas.

Os *daemons* CoMon recolhem diversas informações sobre os nodos, incluindo se o nodo está disponível ou não, *uptime*, se o nodo está respondendo a tentativas de conexão SSH, uso de processador, uso de memória, uso de disco, entre outros. Os dados recentes de cada nodo são mantidos em um servidor central, onde usuários do serviço podem executar consultas e obter uma lista de nodos que atendam uma lista de parâmetros desejados, como por exemplo, apenas nodos apresentando uma baixa utilização de processador e memória, e que estejam respondendo a conexões SSH. Estas consultas também podem ser utilizadas para detectar nodos problemáticos, como por exemplo, nodos que não estejam disponíveis, ou disponíveis mas apresentando problemas como falta de espaço em memória ou disco. Caso o usuário deseje evitar nodos com alto nível de utilização, é possível obter uma lista de nodos que possuem mais de um certo número de *slices* ativos no momento, ou uma lista de nodos que apresentam uma carga de processador acima de um certo limiar.

O projeto CoMon esteve operacional e disponível para consultas desde 2004, mas foi encerrado em 2012.

2.2.8 SWORD

O SWORD [2] é uma ferramenta de seleção de nodos no PlanetLab. O usuário do PlanetLab utiliza o SWORD para localizar um conjunto de nodos que atendam a uma lista de parâmetros desejados. O SWORD se comunica com outras ferramentas de monitoramento do PlanetLab, incluindo o CoMon, para obter informações sobre os nodos. Além das características de cada nodo, como carga do processador, uso de memória, disco e outros, o usuário pode requisitar nodos que atendam parâmetros de comunicação entre si, como largura de banda e tempo de resposta.

O SWORD aceita como entrada um arquivo XML que descreve as características dos nodos desejados pelo usuário. Uma interface Web oferece ao usuário a possibilidade de descrever os parâmetros desejados e gerar este arquivo XML automaticamente. A interface oferece ao usuário uma lista com diversos parâmetros relacionados aos recursos disponíveis nos nodos e a conexão entre os nodos. O usuário escolhe quais parâmetros são relevantes em sua consulta e define valores mínimos e máximos para estes parâmetros. Além da definição de diversos parâmetros, a interface permite a criação de grupos de nodos com parâmetros diferentes em uma mesma consulta. A ferramenta retornará como resultado desta consulta, grupos disjuntos que atendam aos parâmetros estabelecidos pelo usuário.

Uma observação importante é que o SWORD usa dados do CoMon para selecionar nodos. Desde o encerramento do CoMon, a seleção de nodos do SWORD não está disponível.

2.2.9 MyOps Monitor

MyOps [35, 34] é uma ferramenta de monitoramento do próprio PlanetLab. Esta ferramenta é responsável por verificar se os nodos da rede estão funcionando corretamente e enviar notificações automáticas aos responsáveis caso algum problema seja detectado. Se um nodo fica *offline* por um determinado período de tempo ou é detectada a presença de um *firewall* bloqueando as principais portas, uma notificação é enviada à instituição responsável pelo nodo para que as devidas medidas sejam aplicadas e o nodo volte a funcionar de maneira adequada.

O MyOps disponibiliza um portal com uma lista pública do estado dos nodos monitorados e permite ao usuário executar buscas por nodos e obter informações relacionadas, como por exemplo estado (*online* ou *offline*), estado do SSH e *uptime*. Portanto, este portal pode ser utilizado para se obter uma lista de nodos *online* para a execução de experimentos.

Como verificado em julho de 2014, dos 1194 nodos monitorados pela ferramenta, 424 nodos estão classificados como *up* ou disponíveis para a realização de experimentos, e 770 estão classificados como *down*.

CAPÍTULO 3

PRECISÃO E REPETIBILIDADE DE EXPERIMENTOS DISTRIBUÍDOS NO PLANETLAB

Existem três técnicas para avaliação de desempenho de um sistema [14]: o *modelo analítico*, a *simulação* e a *experimentação*. É recomendável que ao menos duas técnicas diferentes sejam utilizadas para uma boa avaliação de desempenho de um sistema. A segunda deve servir como confirmação da primeira. Não se deve, por exemplo, tirar conclusões a partir do resultado de uma simulação sem validá-los através do modelo analítico ou experimentação em um ambiente real.

A escolha da técnica mais adequada depende do estágio atual do projeto e dos recursos necessários, incluindo tempo [14]. Um modelo analítico pode ser construído em qualquer estágio do projeto, enquanto a experimentação demanda ao menos a existência de um protótipo. Um modelo analítico pode não ser tão preciso, uma vez que muitas simplificações e suposições são feitas. A precisão de uma simulação varia de acordo com a quantidade de características do mundo real sendo representadas no modelo. Já os resultados obtidos na experimentação podem oferecer do mais baixo ao mais alto nível de precisão na avaliação de desempenho.

A próxima seção trata justamente da avaliação de experimentos no PlanetLab. A próxima seção descreve uma ferramenta já existente de seleção de nodos para a execução de experimentos no PlanetLab. A seção seguinte descreve a ferramenta de gerenciamento de experimentos PlaneMon. Em seguida, são apresentados os detalhes do trabalho realizado nesta dissertação de mestrado para a integração da ferramenta de seleção ao PlaneMon. A seção seguinte apresenta trabalhos relacionados à repetibilidade de experimentos para o desenvolvimento de aplicações e serviços para a Internet.

3.1 Avaliação de Experimentos no PlanetLab

O planejamento de experimentos é uma etapa fundamental no processo de avaliação de desempenho de um sistema através da experimentação. Um experimento deve ser planejado de forma que ele seja capaz de fornecer o tipo de informação buscada [3]. O primeiro passo é escolher uma métrica adequada para a avaliação de desempenho. *Tempo de resposta*, *vazão*, *utilização de recursos*, *disponibilidade* e *confiabilidade* são algumas das métricas utilizadas.

É importante observar quais são os parâmetros que afetam os valores medidos. Alguns parâmetros são de controle do pesquisador, mas existem fatores externos que não podem ser controlados. Por exemplo, uma variação na carga do processador das máquinas nas quais o experimento é realizado pode afetar os resultados. Mesmo quando um experimento é repetido sob condições tão semelhantes quanto possível, é comum a ocorrência de flutuações nos resultados de cada repetição. Essa variação é chamada *erro experimental* ou *ruído* [6].

Em muitos casos, a avaliação de desempenho tem como objetivo comparar dois sistemas. Uma sequência de experimentos deve ser realizada utilizando cada sistema, esta sequência de repetições é chamada *amostra*. Cada amostra apresenta uma média e um desvio padrão para a métrica selecionada. Os sistemas podem então ser comparados utilizando as médias das respectivas amostras, por exemplo, comparar a média da vazão obtida na sequência de testes de cada sistema.

Uma rede como a do PlanetLab exibe grandes variações em parâmetros importantes, como o tempo de resposta entre nodos, banda disponível, carga em cada um dos nodos, entre outros. Estas características são comuns na Internet, tornando os experimentos do PlanetLab representativos na avaliação de desempenho da proposta de um sistema desenvolvido para a própria Internet.

No entanto, muitas vezes o pesquisador deseja obter maior controle sobre estes parâmetros para avaliar o desempenho de seu sistema quando exposto a características mais específicas. Em outros casos, o pesquisador pretende comparar dois sistemas, e para isso, precisa que as características da rede sejam as mesmas durante a execução dos experimen-

tos com cada sistema. Se as variações em parâmetros importantes forem muito grandes e fora de seu controle, não será possível identificar se a diferença entre os valores médios obtidos com cada sistema está associada a uma diferença entre os sistemas ou a variações indesejadas no ambiente.

Portanto, é desejável encontrar um conjunto de nodos dentro do PlanetLab que apresente uma maior estabilidade durante um longo período de tempo, permitindo que uma sequência de experimentos seja realizada com uma variação menor. Na seção seguinte, uma ferramenta já existente para a seleção de nodos estáveis será descrita. O objetivo desta dissertação é avaliar a capacidade desta ferramenta em reduzir a variação nos resultados de séries de experimentos e determinar qual seria a melhor forma para utilizar esta ferramenta para obter alta precisão e repetibilidade de experimentos no PlanetLab.

3.2 Seleção de Nodos no PlanetLab Baseada na Monitoração de Estabilidade das Interações Fim-a-Fim

O usuário do PlanetLab tem à sua disposição uma grande quantidade de nodos para a execução de experimentos. Alguns serviços de descoberta de recursos no PlanetLab, como o MON [32], CoMon [33] e SWORD [2], foram apresentados no capítulo 2. Estes serviços permitem a seleção de nodos baseada em vários parâmetros relacionados aos recursos individuais de cada nodo, como CPU, memória, disco, entre outros, além da largura de banda na conexão entre os nodos. No entanto, estas informações nem sempre são suficientes para uma escolha de um conjunto de nodos adequada.

A rede do PlanetLab é formada por mais de mil nodos espalhados por todo o mundo, conectados através da Internet. As características de conexão como tempo de resposta, perda de pacotes e banda disponível apresentam grandes variações entre os nodos e entre um mesmo par de nodos em diferentes instantes. Estas variações são naturais da Internet e ocorrem devido a fatores como localização geográfica, características de tráfego, entre outros. Outros fatores relacionados à própria rede do PlanetLab também causam variações nas características de conexão e na disponibilidade de recursos dos nodos, como

a utilização de recursos por vários *slices* em um mesmo nodo executando experimentos simultaneamente.

Em alguns casos, o pesquisador pode estar interessado em expor sua aplicação a estas particularidades da rede do PlanetLab, que em geral representam uma condição pior do que a encontrada na Internet. Entretanto, frequentemente o pesquisador necessita de um grupo de nodos que apresentem um nível razoável de estabilidade entre si. Uma ferramenta para seleção de nodos com base na monitoração da estabilidade das interações fim-a-fim foi proposta em [10, 13]. As subseções seguintes descrevem a arquitetura e as estratégias de seleção empregadas por esta ferramenta.

3.2.1 Ferramenta de Seleção de Nodos: Arquitetura

A arquitetura da ferramenta de seleção de nodos baseada na monitoração de estabilidade das interações fim-a-fim é composta por três módulos: módulo de monitoramento, módulo servidor e módulo cliente. Estes módulos são responsáveis, respectivamente, pelo monitoramento das interações entre os nodos, armazenamento centralizado das informações de monitoramento e interface de interação com o usuário. A organização em módulos da arquitetura deste sistema pode ser observado na Figura 3.1.

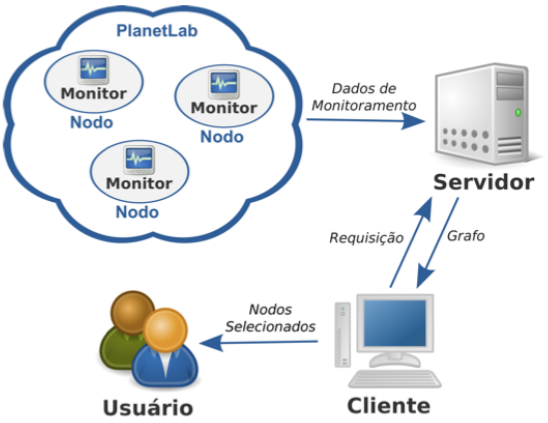


Figura 3.1: Arquitetura da ferramenta de seleção de nodos.

Cada nodo do PlanetLab deve executar um *daemon* de monitoramento. Estes *daemons* se comunicam entre si, enviando e respondendo mensagens utilizando um *socket* UDP. Cada *daemon* deve enviar, periodicamente, mensagens para outros nodos. O tempo até

a resposta de cada mensagem (RTT) é observado e armazenado. Um valor de *timeout* é utilizado, e caso uma resposta não seja obtida antes do *timeout*, o nodo é considerado indisponível.

É importante notar que o RTT é medido na camada de aplicação, portanto o tempo de resposta é influenciado não apenas pelas características de rede, mas também pelo escalonamento de processos no nodo. Se o uso do processador está elevado no momento do recebimento da mensagem, o escalonamento de processos pode causar uma demora maior até que o *daemon* possa processar e responder a mensagem. Esta demora afetará o tempo total entre o envio de uma mensagem e o tempo até o recebimento de sua resposta, afetando o valor de RTT monitorado pelo *daemon*.

O módulo servidor é responsável por armazenar os dados obtidos pelos *daemons* do módulo de monitoramento e oferecer dados ao módulo cliente. O módulo servidor pode ser executado em uma máquina qualquer, não necessariamente em um nodo do PlanetLab, basta que esta máquina execute um sistema de banco de dados e aceite conexões vindas da Internet.

O servidor aguarda conexões dos *daemons* de monitoramento em um *socket* TCP. Um *daemon* se conecta ao servidor e envia seus dados locais, e os dados recebidos de cada *daemon* são então armazenados no banco de dados do servidor. Adicionalmente, o servidor armazena no banco de dados versões sumarizadas destes dados, como média, desvio padrão, valores máximos e mínimos. São mantidos dados sumarizados referentes a cada hora nas últimas 24 horas, a cada dia nos últimos 30 dias, a cada mês nos últimos 12 meses e ao último ano.

O servidor também aguarda requisições do módulo cliente em um *socket* TCP. O módulo cliente pode requisitar ao servidor uma lista de pares de nodos que apresentem uma conexão entre si com RTT abaixo de um certo limiar em um determinado intervalo de tempo. Um par de nodos (a,b) é incluído na lista de resposta ao módulo cliente se, dentro do intervalo de tempo de interesse, ao menos 90% dos dados de RTT observados do nodo a para o nodo b estão abaixo do limiar passado como parâmetro e, ao mesmo tempo, 90% dos dados de RTT observados do nodo b para o nodo a estão abaixo do

mesmo limiar.

O módulo cliente oferece ao usuário uma interface Web para a utilização do sistema. O usuário pode selecionar um intervalo de tempo e um limiar RTT, para obter como resposta, uma lista de pares de nodos que apresentaram um um tempo de resposta entre si abaixo do limiar selecionado durante o período de tempo desejado. Estes dados são obtidos no módulo servidor. A interface exibe a lista de pares de nodos obtidos e também permite ao usuário selecionar um subconjunto destes nodos utilizando diferentes estratégias. A lista de nodos obtida através de uma destas estratégias de seleção pode então ser utilizada para a realização de experimentos. Estas estratégias de seleção serão descritas na seção seguinte.

3.2.2 Estratégias de Seleção

Uma lista de pares de nodos que apresentam tempo de resposta entre si abaixo de um certo limiar em um período de tempo é na verdade uma lista de arestas que forma um grafo, onde cada nodo é representado por um vértice, e uma aresta ligando dois vértices indica que o RTT entre os nodos representados está abaixo de um certo limiar no intervalo de tempo selecionado. A presença de um vértice representando um nodo neste grafo não significa que o nodo faz parte de um subconjunto de nodos estáveis. É desejável encontrar um subconjunto de nodos que apresente um alto grau de conectividade entre si, ou seja, um subgrafo onde os vértices possuem um alto grau de conectividade. Para isto, diferentes estratégias de seleção podem ser utilizadas, como a estratégia do Grau Mínimo, Maior Grau Mínimo, *Core*, *k-Core* e Clique Estável. Estas estratégias serão brevemente descritas a seguir.

A estratégia do Grau Mínimo é a menos restritiva e resulta no maior número de nodos selecionados. O algoritmo recebe como parâmetro um grau mínimo m e constrói um subgrafo removendo todos os vértices do grafo original que possuem grau menor que m. Já a estratégia do Maior Grau Mínimo recebe como parâmetro um número mínimo de vértices n. O algoritmo constrói um subgrafo com o maior grau mínimo possível que inclua ao menos n vértices.

A estratégia *k-Core* constrói um subgrafo com vértices de grau maior ou igual ao parâmetro k. Diferentemente da estratégia do Grau Mínimo, o grau observado é o grau do vértice no subgrafo, portanto as arestas no grafo original, conectando o vértice a vértices não selecionados não são contabilizadas. Já a estratégia *Core* não recebe o valor k como parâmetro. O algoritmo desta estratégia faz uma busca pelo *k-Core* com o valor mais alto possível de k que não resulte em um subgrafo vazio.

A estratégia da Clique Estável consiste em encontrar um subgrafo completo, ou seja, todos os vértices selecionados possuem arestas entre si. A busca pela maior clique em um grafo é um problema de complexidade exponencial e sua utilização na ferramenta de seleção seria inviável devido ao grande número de nodos. Para contornar este problema da complexidade, a estratégia recebe como parâmetros um tamanho mínimo para a clique e um tempo máximo de execução. A estratégia de seleção retorna uma clique de tamanho mínimo, caso encontrada a tempo, ou a maior clique encontrada até o tempo máximo de execução ser atingido.

Os experimentos realizados pelos autores em [10] demonstraram que a melhor estratégia a ser utilizada é o *Core*. As estratégias de Grau Mínimo e Maior Grau Mínimo retornam um grande conjuntos de nodos, mas não garantem forte estabilidade entre os nodos selecionados, uma vez que o grau mínimo considerado é o do grafo como um todo e não apenas do subgrafo selecionado. Já a estratégia da Clique Estável demonstrou ser muito restritiva, retornando pequenos subconjuntos de nodos e que se desfazem em um curto intervalo de tempo. Assim que ocorre uma pequena variação na conexão entre qualquer par de nodos na clique, a aresta entre esse par de nodos deixa de existir, desfazendo a clique. Já a estratégia *Core* apresenta um bom equilíbrio entre o número de nodos no subconjunto selecionado e a estabilidade deste subconjunto ao longo do tempo.

Os autores também conduziram um experimento para comparar o desempenho de uma aplicação no PlanetLab utilizando os nodos selecionados pela estratégia *Core* e os nodos selecionados pela ferramenta SWORD [2] (descrita no Capítulo 2 desta dissertação). Quando a aplicação foi executada utilizando os nodos selecionados pela estratégia *Core*, o tempo de execução apresentou uma média mais baixa e um desvio padrão menor quando

comparados aos tempos de execução da aplicação utilizando os nodos selecionados pelo SWORD.

As seções seguintes descrevem a ferramenta de gerenciamento de experimentos PlanetMon e nossa proposta de integração da ferramenta de seleção de nodos ao PlanetMon, oferecendo o acesso a esta ferramenta à comunidade de usuários do PlanetLab. O capítulo 4 desta dissertação apresentará os resultados de nossos experimentos realizados para avaliar o uso da ferramenta de seleção de nodos para obter maior precisão e repetibilidade de experimentos no PlanetLab.

3.3 PlanetMon

A execução de experimentos distribuídos no PlanetLab é uma atividade que pode atingir um alto nível de complexidade. Conforme mencionado anteriormente, é responsabilidade do pesquisador desenvolver meios para instalar pacotes de software, instalar e atualizar a aplicação desenvolvida, iniciar e monitorar a execução do experimento, tudo isso, em cada nodo do seu *slice*. Esta tarefa se torna especialmente trabalhosa em experimentos envolvendo centenas de nodos.

O PlanetMon [18] é uma ferramenta desenvolvida para facilitar a execução de experimentos no PlanetLab, permitindo a execução de tarefas essenciais comuns a experimentos em ambientes distribuídos. Através de uma interface Web, o pesquisador prepara o ambiente do experimento, fazendo a instalação dos pacotes de software necessários em todos os nodos, além de iniciar, monitorar e recolher resultados dos experimentos realizados.

O usuário do PlanetMon deve criar uma conta no sistema para obter um ambiente exclusivo. Dentro do seu próprio ambiente no portal PlanetMon, o pesquisador deve inserir informações referentes ao seu *slice* e seguir as instruções para que o PlanetMon possa ter acesso aos nodos e executar as tarefas desejadas pelo pesquisador. Para isto, basta que o usuário adicione uma chave pública gerada pelo PlanetMon nas configurações do portal PlanetLab. Uma vez configurado, o usuário pode controlar seu experimento através do portal PlanetMon. Além de selecionar os nodos utilizados no experimento, instalar pacotes de software nestes nodos, iniciar a aplicação e recolher resultados, o

PlanetMon permite ao pesquisador visualizar seu experimento em um mapa utilizando um Gerador de Mapas de Topologia.

O Gerador de Mapas de Topologia obtém as coordenadas geográficas dos nodos, disponíveis no PlanetLab, e monta um mapa com os nodos do experimento. O sistema oferece uma API à aplicação do usuário. Em sua aplicação, o usuário pode incluir código para acionar esta API, requisitando a inclusão de nodos e arestas entre nodos no mapa, além de mudanças de cores nos nodos e arestas. O pesquisador pode utilizar estas funcionalidades para indicar o progresso de sua aplicação. O Gerador de Mapas de Topologias monta e atualiza o mapa em tempo real seguindo os comandos requisitados na API, permitindo ao pesquisador monitorar a execução de seu experimento.

3.4 Integração da Ferramenta de Seleção de Nodos ao PlanetMon

Uma das contribuições deste trabalho é oferecer à comunidade de usuários do PlanetLab acesso à ferramenta de seleção de nodos, de forma conveniente e integrada ao gerenciamento do experimento. A ferramenta de seleção de nodos já possuía uma interface onde o usuário tem a possibilidade de selecionar parâmetros e obter uma lista de nodos do PlanetLab para executar experimentos. Nossa proposta de integração apenas permite que esta seleção de nodos ocorra de forma transparente ao usuário, totalmente integrada ao PlanetMon.

A ferramenta de seleção de nodos está acessível em *monplanetlab.c3sl.ufpr.br/select*. Atualmente, o módulo de monitoramento recolhe informações de aproximadamente 400 nodos do PlanetLab. A razão pela qual não são monitorados todos os nodos são múltiplas. Muitos nodos do PlanetLab não respondem às tentativas de conexão SSH, outros não conseguem completar a transferência do módulo de monitoramento por problemas de conectividade ou falta de espaço em disco. Uma rotina automática, tenta diariamente, instalar ou reparar o módulo de monitoramento em cada nodo do PlanetLab, em um esforço para que o maior número de nodos possível seja monitorado.

A ferramenta PlanetMon original é mantida online em *planetmon.inf.ufpr.br*. Nossa proposta de integração foi realizada em uma cópia *offline* deste sistema. Todas as mo-

dificações realizadas foram planejadas e documentadas de forma que seja muito simples aplicá-las ao sistema original. Algumas telas do sistema original foram alteradas. Toda a interface de comunicação entre o PlanetMon e o servidor da ferramenta de seleção de nodos foi organizada em um módulo e basta que este módulo seja incluído na biblioteca do sistema do PlanetMon.

A principal modificação ocorre na aba *slice* do painel de controle do PlanetMon. Nesta aba do sistema, o usuário grava informações relacionadas a seu *slice* e gerencia a lista de nodos utilizados na realização de experimentos. Nossa modificação permite ao usuário obter a sua lista de nodos através da ferramenta de monitoramento e seleção de nodos.

A Figura 3.2 mostra uma tela do PlanetMon na aba *slice* modificada. O usuário indica qual o algoritmo de seleção deve ser utilizado, número de nodos e o limiar de tempo de resposta entre os nodos selecionados. Estas informações ficam gravadas junto às informações do *slice* do usuário. É possível obter uma lista de nodos imediatamente utilizando estes parâmetros, ou ainda, configurar o sistema para que a seleção de nodos ocorra automaticamente antes da execução de um experimento.

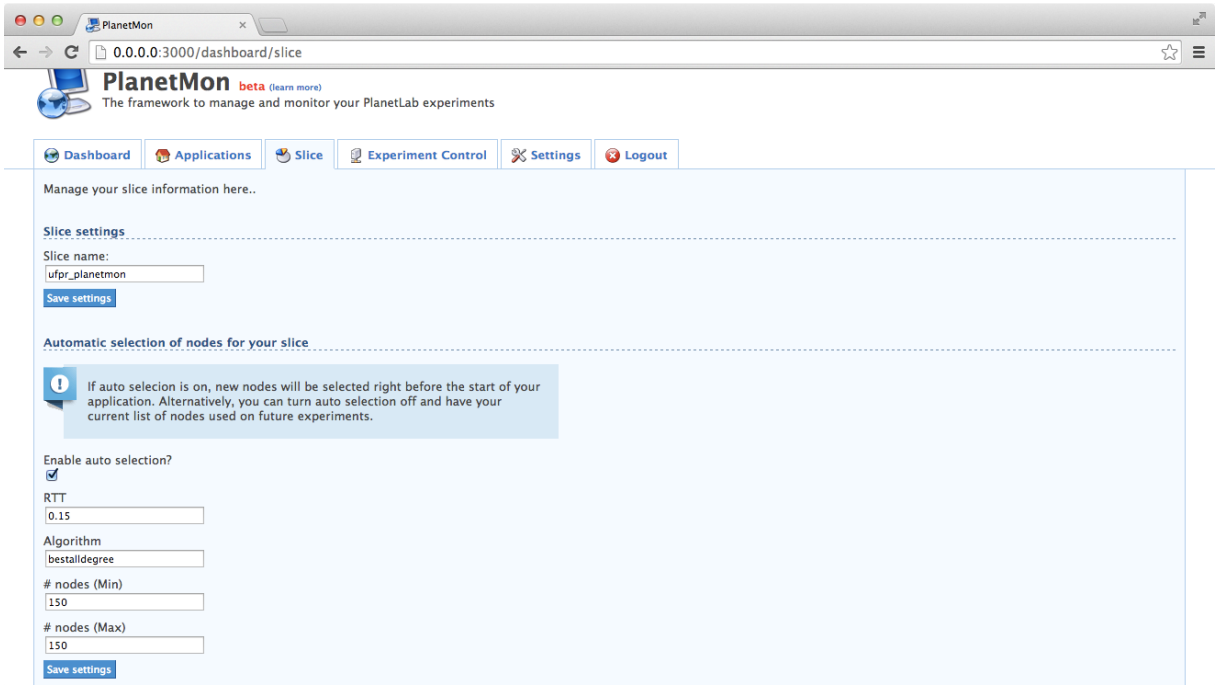


Figura 3.2: Parâmetros para a seleção de uma lista de nodos.

A Figura 3.3 mostra uma tela do PlanetMon após a atualização da lista de nodos utili-

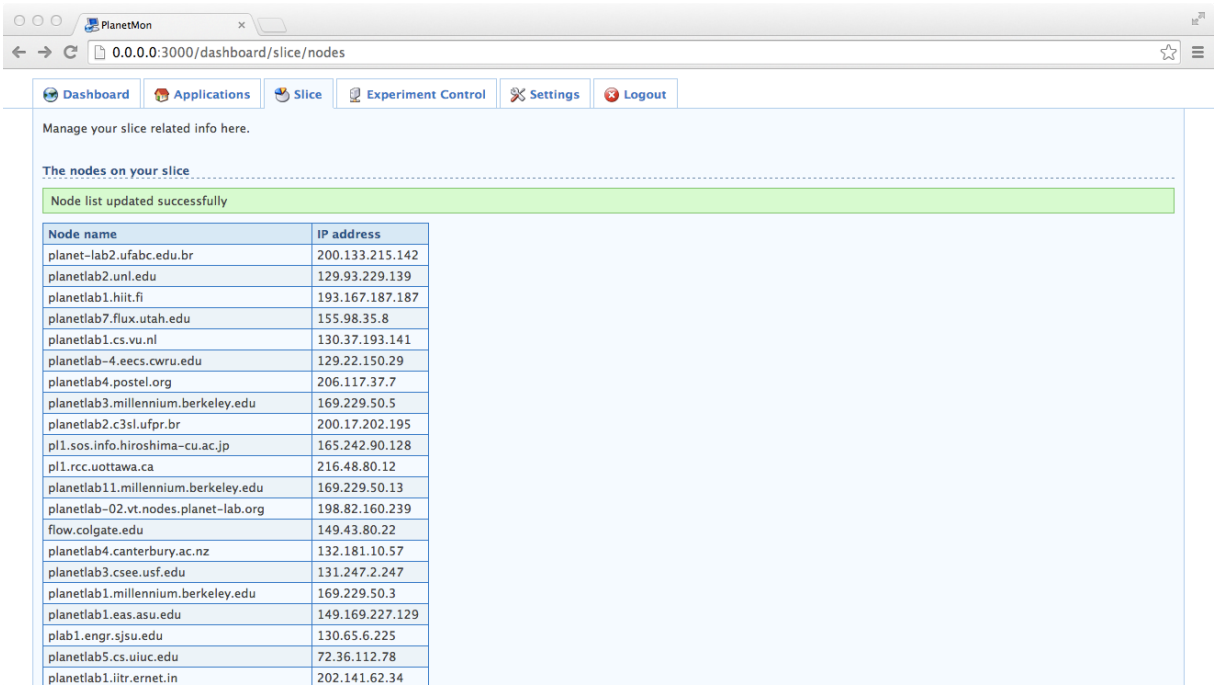


Figura 3.3: Lista de nodos selecionados pela ferramenta integrada ao PlanetMon.

zando a ferramenta de seleção. Esta lista fica gravada no sistema para ser utilizada em um futuro experimento, ela pode ser obtida pelo usuário em formato texto, editada, removida ou substituída. Caso o usuário selecione a opção para realizar automaticamente a seleção de nodos antes da execução de um experimento, a lista de nodos atual é sobrescrita pelo sistema no momento em que o experimento é iniciado.

É importante observar que o PlanetMon não acessa o portal do PlanetLab e altera configurações do usuário automaticamente. Atualmente, é de responsabilidade do usuário garantir que os nodos presentes na lista do PlanetMon pertençam ao seu *slice* no PlanetLab. Qualquer nodo selecionado automaticamente pela ferramenta de seleção que não faça parte do *slice*, não estará acessível para a instalação da aplicação, e não participará do experimento. A automatização da inclusão de nodos no *slice* do usuário foi considerada, mas existe um atraso de minutos ou até horas entre a inclusão de um nodo no *slice* e sua disponibilidade ao usuário, portanto esta automatização não foi considerada uma solução adequada e foi descartada.

3.5 Repetibilidade de Experimentos: Trabalhos Relacionados

É muito importante para pesquisadores que suas novas propostas de protocolos e aplicações sejam testadas em um ambiente real e ao mesmo tempo controlável, de forma que os resultados dos experimentos sejam representativos e também repetíveis. Estas duas condições são conflitantes, pois não existe muito controle sobre os parâmetros de um ambiente real. Algumas abordagens buscam maior controle sobre as variáveis de um ambiente real, enquanto outros trabalhos buscam oferecer mais realismo em um ambiente controlado.

A execução de experimentos repetíveis em condições realísticas para redes sem fio é abordada em [15]. Experimentos com comunicação sem fio são difíceis de repetir com precisão e modelos de simulação abstratos utilizam parâmetros teóricos não realísticos. Os autores apresentam modelos para a camada física e MAC de redes sem fio baseados em medições realizadas em redes reais, para serem utilizados em simulações e emulações. Os autores concluíram que seus modelos oferecem maior realismo para simulações, aproximando o ambiente simulado ao encontrado em um ambiente de experimentação de um *testbed*.

Uma infraestrutura de rede virtual, chamada VINI, é proposta em [4]. A ferramenta foi projetada com foco em oferecer realismo nas condições de rede e tráfego, controle sobre eventos na rede e utiliza a virtualização para oferecer a flexibilidade de se construir diferentes topologias de rede sobre uma mesma infraestrutura física. Os autores realizaram experimentos instanciando VINI em um conjunto de nodos no PlanetLab e concluíram que a ferramenta é eficiente e apresenta condições de rede representativas.

Um *framework* para a automatização do projeto e execução de experimentos chamado Weevil é apresentado em [20]. O trabalho foca-se na automatização da geração de cargas de trabalho e controle de parâmetros para a obtenção de resultados representativos e repetíveis. Os autores ressaltam a necessidade de se possuir controle sobre o ambiente de experimentação e as dificuldades de se obter tal controle no PlanetLab. Os autores validaram suas propostas através de séries de experimentos realizados no *testbed* EmuLab [24].

Os mesmos autores dos artigos originais do PlanetLab apresentam “verdades, mitos

e boas práticas" para a execução de experimentos no PlanetLab em [19]. Os autores afirmam que os resultados dos experimentos no PlanetLab não são reproduzíveis, e que o *testbed* foi projetado para oferecer condições reais de rede e não um ambiente controlável. Entretanto, no caso de aplicações que executam por longos períodos de tempo, os autores sugerem que pesquisadores devem ser capazes de detectar padrões no comportamento da rede e entender o desempenho e confiabilidade obtido por suas aplicações e serviços. No caso de aplicações de curta duração, os autores sugerem a seleção de nodos estáveis utilizando a ferramenta CoMon (hoje não mais disponível, como mencionado no capítulo 2 desta dissertação) para a identificação de nodos problemáticos ou que apresentem uma carga de sistema alta.

Selecionar conjuntos de nodos estáveis é exatamente a estratégia para reduzir a variação nos resultados dos experimentos discutida nesta dissertação. O capítulo 4 apresenta uma avaliação experimental do quanto as instabilidades presentes no PlanetLab afetam os resultados dos experimentos realizados e como a utilização da ferramenta de seleção de nodos (apresentada anteriormente) pode aumentar a precisão dos experimentos realizados, contribuindo para a repetibilidade dos resultados obtidos.

CAPÍTULO 4

RESULTADOS EXPERIMENTAIS

Este capítulo descreve os resultados experimentais obtidos na avaliação da repetibilidade de experimentos no *testbed* PlanetLab. A realização de experimentos em um ambiente instável dificulta a repetibilidade de experimentos, uma vez que os resultados obtidos em ocasiões distintas podem ser muito diferentes e estas diferenças são causadas por fatores desconhecidos ou não controláveis.

O PlanetLab é composto por mais de mil nodos espalhados ao redor do mundo conectados através da Internet. Diversos usuários utilizam o sistema simultaneamente e suas aplicações concorrem pelos recursos dos nodos. Estas características tornam o PlanetLab um ambiente muito instável e, consequentemente, é difícil obter resultados precisos e repetíveis.

O principal objetivo dos experimentos apresentados neste capítulo é demonstrar que a escolha de um subconjunto de nodos adequado pode reduzir o impacto da instabilidade do ambiente de testes nos resultados obtidos, tornando-os mais precisos. A métrica utilizada para avaliar precisão dos experimentos realizados será o *coeficiente de variação* (desvio padrão dividido pela média). O coeficiente de variação é proporcional à média, portanto é adequado para a comparação da variação entre o resultados de amostras com médias diferentes.

A seleção de subconjuntos de nodos baseada no monitoramento da estabilidade das interações fim-a-fim [13], descrita no Capítulo 3, faz uso apenas dos dados de tempo de resposta entre nodos, obtidos na camada de aplicação. Também faz parte do objetivo dos experimentos apresentados neste capítulo demonstrar que esta estratégia de seleção é adequada para aplicações distribuídas com diferentes perfis de utilização de recursos, ou seja, a informação obtida no monitoramento é suficiente para reduzir a variância entre uma sequência de resultados obtidos, para uma aplicação distribuída que apresente alto

uso de um recurso não analisado, como por exemplo, carga da CPU.

Foram implementadas três aplicações distribuídas para serem executadas no ambiente do PlanetLab. As aplicações escolhidas apresentam diferentes características de uso de recursos, variando de intensa troca de mensagens e reduzido uso de CPU a intenso uso de CPU e reduzida troca de mensagens. Foram obtidos subconjuntos de nodos utilizando a estratégia de seleção descrita no Capítulo 3 e também através de uma estratégia alternativa. Para cada aplicação, a precisão dos resultados obtidos utilizando cada subconjunto de nodos é comparada utilizando o coeficiente de variação.

Os experimentos foram organizados em *rodadas de experimentos* e *séries*. Uma série é composta por 30 rodadas de experimentos e cada rodada de experimentos consiste na execução das três aplicações utilizando cada um dos subconjuntos de nodos. Uma rodada de experimentos se completa em aproximadamente 150 minutos, a próxima rodada inicia-se automaticamente logo ao fim da anterior, sem nenhuma restrição de horário ou dia da semana. As séries são iniciadas manualmente e têm duração de aproximadamente 3 dias. No total, foram executadas 300 rodadas de experimentos organizadas em 10 séries, sendo a primeira série iniciada em 20 de abril de 2014 e a última em 25 de maio de 2014.

Este capítulo está organizado da seguinte maneira. A próxima seção descreve as estratégias e os parâmetros utilizados na seleção dos subconjuntos de nodos utilizados para a realização dos experimentos. A próxima seção apresenta detalhes das aplicações utilizadas nos experimentos e os resultados obtidos. A seção seguinte apresenta uma análise geral dos resultados dos experimentos realizados.

4.1 Seleção dos Subconjuntos de Nodos

Apesar de o PlanetLab ser formado por mais de mil nodos, o número de nodos acessíveis via SSH durante a execução destes experimentos foi muito mais baixo, usualmente em torno de 400 nodos. Para a execução dos experimentos, foram utilizados quatro subconjuntos de 150 nodos. Dois subconjuntos foram selecionados utilizando a estratégia de seleção de nodos apresentada no Capítulo 3, chamados `core-150ms` e `core-150ms-everyround`. Os outros dois subconjuntos, obtidos utilizando uma estratégia alternativa, são chamados

`ping.c3sl` e `ping.uk`.

Os subconjuntos *core* (`core-150ms` e `core-150ms-everyround`) foram selecionados utilizando os seguintes parâmetros: 150 nodos, dados de monitoramento obtidos na última hora, limiar tempo de resposta de 150ms e estratégia de seleção *core*. A estratégia de seleção *core* foi apontada em [13] como a melhor estratégia para a seleção de nodos estáveis. Uma seleção de nodos utilizando estes parâmetros usualmente resulta em um subconjunto de nodos com grau mínimo próximo a 120, significando que todos os nodos do subconjunto têm conexão com latência menor que 150ms para pelo menos outros 120 nodos dentro do subconjunto selecionado.

O subconjunto `core-150ms` foi obtido antes do início dos experimentos, no dia 20 de abril de 2014. Este subconjunto permanece inalterado durante a execução de todos os experimentos. Já o subconjunto `core-150ms-everyround` é obtido novamente no início de cada *rodada de experimentos*. Os nodos são obtidos sempre utilizando os mesmos parâmetros de seleção.

Durante uma das séries de experimentos realizados, a composição do subconjunto `core-150ms-everyround` foi observada. Em um período de aproximadamente três dias, tempo necessário para completar a série, um total de 262 nodos distintos fizeram parte do subconjunto `core-150ms-everyround` em ao menos uma das 30 rodadas de experimento. Destes 262 nodos, 59 foram constantemente selecionados em todas as rodadas e 123 foram selecionados mais de 20 vezes. Por outro lado, 42 nodos foram selecionados 5 vezes ou menos e apenas 11 foram selecionados uma única vez. Também foi observado que 126 nodos selecionados na primeira rodada de experimentos estavam presentes no subconjunto selecionado na trigésima rodada de experimentos.

Para comparação, selecionamos outros dois subconjuntos de nodos com uma estratégia alternativa, utilizando os valores de tempos de resposta obtidos pela troca de mensagens *Echo Request* e *Echo Reply* do protocolo ICMP, implementada pelo comando *ping*. Nesta estratégia, um nodo central é escolhido e é construída uma lista com os tempos de resposta obtido pela média de cinco execuções do comando *ping* de cada nodo para o nodo central escolhido. Os nodos que obtiverem os valores de tempo de resposta mais baixos são ser

selecionados. Ambos os subconjuntos foram selecionados antes do inicio dos experimentos, em 20 de abril de 2014, e permanecem inalterados durante as execuções de todos os experimentos.

O primeiro subconjunto de nodos obtidos através desta estratégia, chamado `ping.c3sl`, utilizou como nodo central o nodo *planetlab2.c3sl.ufpr.br*, um dos dois nodos mantidos no PlanetLab pelo Centro de Computação Científica e Software Livre na Universidade Federal do Paraná.

O segundo subconjunto de nodos obtido através desta estratégia, chamado `ping.uk` utilizou como nodo central o *planetlab2.xeno.cl.cam.ac.uk*. Este nodo foi escolhido por estar constantemente entre os nodos estáveis selecionados utilizando a estratégia de seleção *core*, portanto este subconjunto deve conter muitos dos nodos também presentes no subconjunto `core-150ms`.

O subconjunto `ping.uk` é bastante semelhante ao subconjunto `core-150ms`, dos 150 nodos selecionados, 100 estão presentes nos dois subconjuntos. Já o subconjunto `ping.c3sl` é bastante diferente dos outros subconjuntos, apenas 11 de seus nodos também estão presentes no subconjunto `ping.uk` e 27 estão presentes no subconjunto `core-150ms`. Portanto, o subconjunto `ping.uk` é bastante similar aos subconjuntos obtidos a partir da estratégia *core*, enquanto o subconjunto `ping.c3sl` é bastante diferentes dos outros três subconjuntos.

4.2 Aplicações Executadas e Resultados

Os experimentos foram realizados no período entre 20 de abril e 28 de maio de 2014. Os experimentos foram executados de forma serializada, alternado os subconjuntos de nodos e as aplicações utilizadas. É importante destacar que, apesar de nossos experimentos serem realizados individualmente, a todo momento existem diversas aplicações de outros usuários concorrendo pelos recursos dos nodos.

Para a realização dos experimentos foram utilizadas três aplicações. Duas destas aplicações foram implementadas utilizando o *framework* Map-Reduce [8] e a terceira aplicação utiliza o protocolo *BitTorrent* [23]. As duas aplicações Map-Reduce foram desenvolvidas

utilizando o Mincemeatpy [31], uma implementação *leve* do Map-Reduce em Python. O Mincemeatpy permite o desenvolvimento de aplicações simples e não requer a instalação e configuração complexa antes da execução de um experimento.

Uma aplicação Mincemeatpy possui um único nodo *master* e vários *workers*. Os *workers* se conectam ao nodo *master* e executam tarefas assinaladas a eles. O nodo *master* organiza as funções *Map* e *Reduce* como tarefas. Na fase *Map*, todos os *workers* realizam tarefas da função *Map* e na fase *Reduce*, todos os *workers* realizam tarefas da função *Reduce*. Nesta implementação, os *workers* não se comunicam entre si, toda comunicação é gerenciada pelo nodo *master*, portanto as aplicações apresentam uma topologia em estrela.

As subseções seguintes apresentam mais detalhes sobre as características de cada aplicação e os resultados obtidos nos experimentos realizados com cada uma delas. A primeira subseção apresenta a aplicação *Triangular.py*, a subseção seguinte apresenta a aplicação *Bit.py* e a terceira subseção apresenta a aplicação *Torrent*.

4.2.1 Aplicação Map-Reduce 1: *Triangular.py*

A primeira aplicação desenvolvida utilizando o Mincemeatpy, *Triangular.py*, retorna uma lista com os N primeiros números triangulares. O n-ésimo número triangular T_n é dado por $(n^2 + n)/2$, para n pertencente ao conjunto dos números naturais [46]. Visualmente, um número triangular é a contagem de objetos que, organizados como na Figura 4.1, formam um triângulo equilátero. A Figura 4.1 exibe os seis primeiros números triangulares.

Com os parâmetros selecionados para esta sequência de experimentos, a aplicação calcula os primeiros 8192 números triangulares. O código das funções *map* e *reduce* da aplicação *Triangular.py* pode ser observado no Algoritmo 1. A implementação destas funções não é a forma mais eficiente de se encontrar números triangulares, mas tem como característica o baixo uso de CPU e intensa troca de mensagens. Para calcular os primeiros 8192 números triangulares, a função *map* emite mais de 33 milhões de pares (*key, i*) (linha 3), e a função *reduce* calcula a soma dos valores emitidos para cada chave *key* (linha 7), sendo essa soma o *key*-ésimo número triangular.

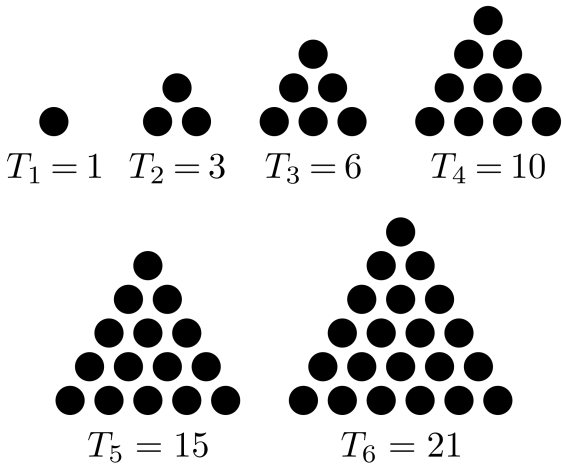


Figura 4.1: Seis primeiros números triangulares.

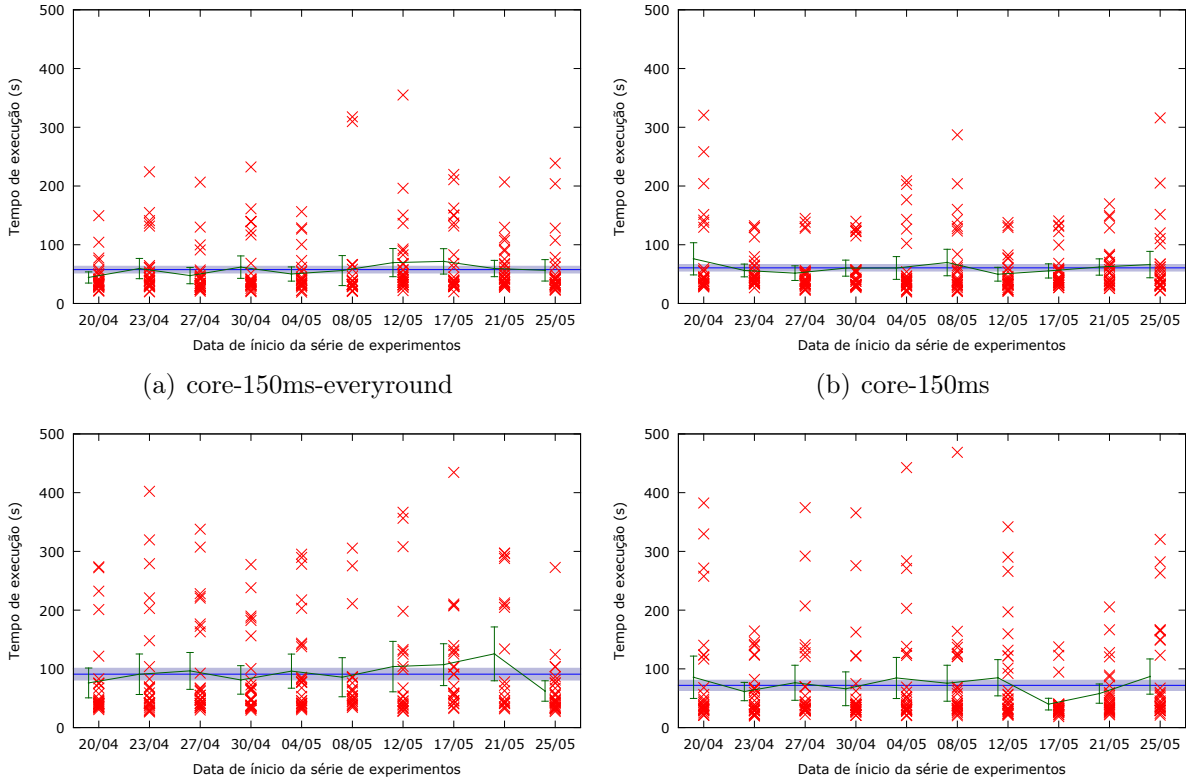
Algorithm 1 Map Reduce Triangular.py

```
1: function MAP(key, value)
2:     for i ← 1 to value do
3:         yield key, i
4:     end for
5: end function

6: function REDUCE(key, value)
7:     return sum(value)
8: end function
```

Os gráficos na Figura 4.2 apresentam os resultados obtidos com a execução da aplicação *Triangular.py*. Foram realizadas 300 rodadas de experimentos, organizadas em 10 séries de 30 repetições da aplicação para cada subconjunto. O eixo y indica o tempo de execução em segundos e o eixo x indica a data de execução de cada série de experimentos. A linha azul cruzando o gráfico representa a média do tempo de execução destas 300 repetições, e a sombra cinza o intervalo de confiança no nível de 95%. A linha verde apresenta a média e intervalo de confiança no nível de 95% para cada série de 30 repetições. Cada x vermelho representa o tempo de execução de um experimento.

Observando a distribuição dos x vermelhos, nota-se a presença de diversos valores muito distantes da média nos quatro gráficos, como por exemplo os dois resultados acima de 300 segundos obtidos pelo subconjunto `core-150ms-everyround` na série iniciada no dia 08/05. Porém, os valores distantes da média aparecem com maior frequência para os



(a) core-150ms-everyround

(b) core-150ms

(c) ping.c3sl

(d) ping.uk

Figura 4.2: Tempos de execução da aplicação *Triangular.py* para cada subconjunto de nodos.

subconjuntos obtidos com a estratégia *ping*. Observando-se a linha verde, comparando as médias obtidas em cada série de 30 repetições e a média geral obtida por todas as 300 repetições do experimento, a distância entre estas médias é menor nos resultados obtidos com o subconjunto *core* comparados aos resultados obtidos com os subconjuntos *ping*.

A Tabela 4.1 apresenta informações referentes aos tempos de execução de todas as 300 repetições da aplicação *Triangular.py* com cada subconjunto de nodos. A tabela exibe as informações considerando 100% dos dados obtidos e, adicionalmente, as mesmas informações considerando apenas 95% dos dados, excluindo 2.5% dos resultados mais altos e mais baixos obtidos.

Todos os subconjuntos apresentaram um coeficiente de variação bastante alto, mas a estratégia de seleção *core* contribuiu para o aumento da precisão, obtendo coeficientes de variação relativamente mais baixos, comparados aos subconjuntos *ping*. Observando a última linha da tabela, com 5% dos dados discrepantes descartados, o coeficiente de

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
100% dos dados				
Média (s)	57,65	60,72	91,02	71,93
Desvio Padrão (s)	50,79	48,59	83,40	77,02
Mínimo (s)	19,31	19,60	26,59	18,58
Máximo (s)	354,99	320,66	434,45	468,62
Coeficiente de variação	0,881	0,800	0,916	1,070
95% dos dados				
Média (s)	52,79	56,36	85,08	65,59
Desvio Padrão (s)	37,26	36,65	71,57	60,09
Mínimo (s)	21,02	21,69	29,13	20,95
Máximo (s)	206,89	202,69	307,26	320,42
Coeficiente de variação	0,705	0,650	0,841	0,916

Tabela 4.1: Sumarização dos experimentos realizados com a aplicação *Triangular.py*.

variação dos subconjuntos *core* é de 0,705 e 0,650, enquanto os subconjuntos *ping* obtiveram 0,841 e 0,916. Ainda observando os resultados eliminando 5% dos dados mais discrepantes, os subconjuntos *core* obtiveram valores máximos próximos a 200 segundos, enquanto o tempo máximo obtido pelos subconjuntos *ping* ultrapassou os 300 segundos.

Já observando-se as médias de tempo de execução, é possível perceber que os subconjuntos *core* obtiveram um tempo de execução médio mais baixo que os subconjuntos *ping*. Considerando os resultados eliminando 5% dos dados discrepantes, as médias de tempo de execução dos subconjuntos *core* foram de 52,79s e 56,36s, contra as médias de 85,08s e 65,59s obtidas pelos subconjuntos *ping*.

Em geral, os subconjuntos obtidos com a estratégia *core* resultaram em tempo de execuções mais baixos e resultados mais consistentes do que os subconjuntos obtidos com a estratégia *ping*. O subconjunto `core-150ms-everyround` obteve uma média de tempo de execução mais baixa que o subconjunto `core-150ms`. A seleção de novos nodos no início de cada rodada de experimento potencialmente causa maior variação nos resultados obtidos, no entanto o subconjunto `core-150ms-everyround` obteve um coeficiente de variação próximo ao do subconjunto `core-150ms`, demonstrando a capacidade da estratégia *core* de selecionar subconjuntos de nodos com características de conectividade semelhante ao longo do tempo.

4.2.2 Aplicação Map-Reduce 2: *Bit.py*

A segunda aplicação desenvolvida foi inspirada no problema resolvido por *mineradores* da cripto moeda Bitcoin [22]. O Bitcoin, lançado em 2009, é uma moeda com criação e transferência baseada em protocolos de código aberto de criptografia. A segurança e confiabilidade desta moeda vem da dificuldade de se validar uma transação incorreta. As transações de Bitcoins são organizadas em blocos e validadas por uma rede P2P. A validação destas transações requer grande poder de processamento, mas gera recompensa financeira aos usuários envolvidos, chamados *mineradores*.

A validação de um bloco de transações ocorre utilizando a função *hash* SHA-256 [11]. De forma simplificada, o objetivo de um minerador de Bitcoin é, dado um valor de entrada que representa o bloco de transações e informações relacionadas ao bloco anterior, encontrar um valor v tal que SHA-256$(entrada+v) < h$, sendo h um valor em hexadecimal de 256 bits, usualmente com muitos zeros à esquerda. Quanto maior o número de zeros à esquerda de h, mais difícil se torna encontrar um valor v que resulte em uma *hash* menor que h.

A aplicação implementada resolve uma versão simplificada deste problema. O Algoritmo 2 mostra o pseudo código das funções *map* e *reduce* desta aplicação. A função *map* calcula a função SHA256 (linha 4), variando os valores de v dentro do intervalo $[a,b]$. Quando a *hash* resultante possui 5 zeros ou mais, é emitido um par de valores representando o número de zeros à esquerda encontrado e o valor de v utilizado (linha 7). A função *reduce* apenas conta quantos valores de v foram encontrados para cada quantidade de zeros à esquerda (linha 12). Esta aplicação tem como característica intenso uso de CPU e baixa quantidade de troca de mensagens.

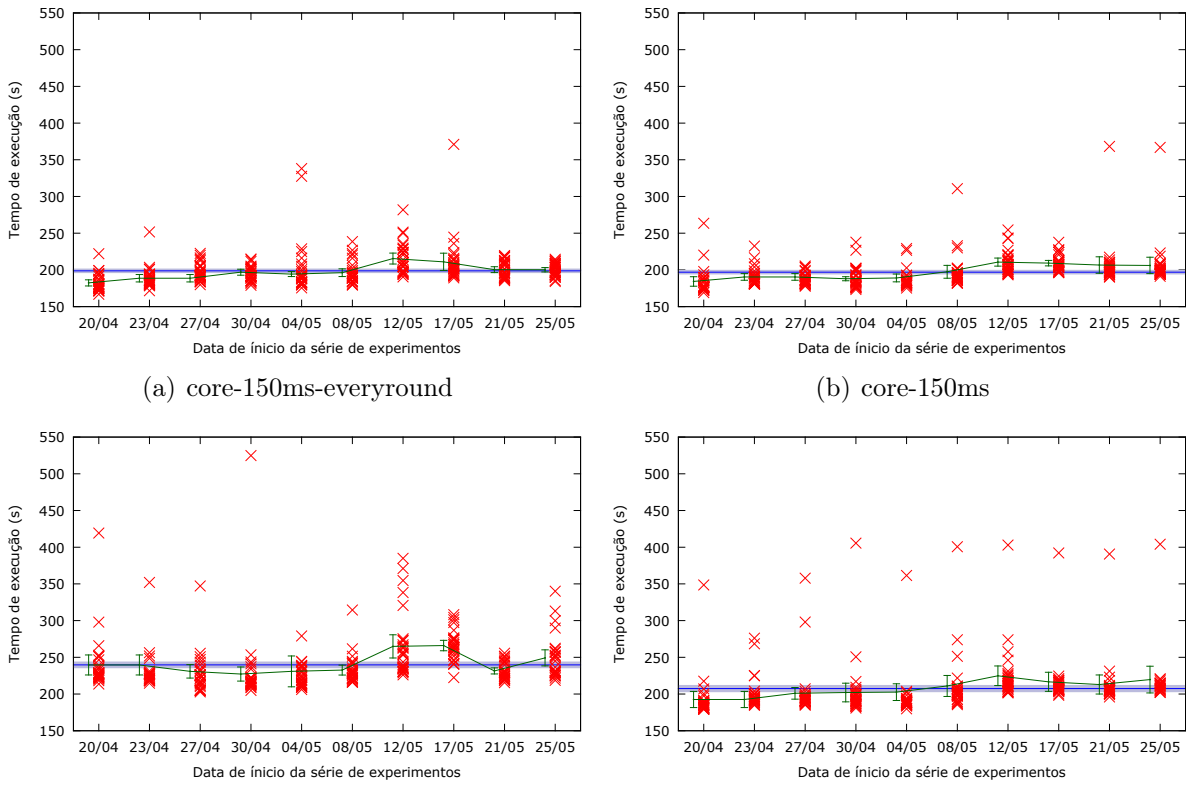
Com os parâmetros selecionados, durante uma execução da aplicação, a função SHA-256 é calculada para quase 7 bilhões de valores de v. Destes bilhões de valores para v, mais de 6 mil resultaram em 5 zeros à esquerda da *hash* resultante, mais de 300 com 6 zeros, 36 com 7 zeros e apenas 2 com 8 zeros. Seria necessário testar aproximadamente 68 bilhões de valores para v para se encontrar uma *hash* com 9 zeros, e mais de um trilhão para 10 zeros. Por exemplo, a palavra "exemplo" concatenada com o valor de v 1066323

resulta na *hash 00000a45b18bf5c2961363ad1137f0b6*, que contém 5 zeros à esquerda.

Algorithm 2 Map Reduce Bit.py

```
1:  function MAP(key, value)
2:      a, b ← value.split()          ▷ Intervalo (a, b) de possíveis valores para v
3:      for v ← a to b do
4:          h ← SHA256(key + v)
5:          z ← conta_zeros_esquerda(h)
6:          if z ≥ 5 then
7:              yield z, v
8:          end if
9:      end for
10: end function

11: function REDUCE(key, value)
12:     return "key : len(value)"
13: end function
```



(a) core-150ms-everyround

(b) core-150ms

(c) ping.c3sl

(d) ping.uk

Figura 4.3: Tempos de execução da aplicação Bit.py para cada subconjunto de nodos.

Os gráficos na Figura 4.3 apresentam os resultados obtidos com a execução da aplicação *Bit.py*. Foram realizadas 300 rodadas de experimentos, organizadas em 10 séries de 30

repetições da aplicação para cada subconjunto. O eixo y indica o tempo de execução em segundos e o eixo x indica a data de execução de cada série de experimentos. A linha azul cruzando o gráfico representa a média do tempo de execução destas 300 repetições, e a sombra cinza o intervalo de confiança no nível de 95%. A linha verde apresenta a média e intervalo de confiança no nível de 95% para cada série de 30 repetições. Cada x vermelho representa o tempo de execução de um experimento.

Observando a distribuição dos x vermelhos, nota-se a presença de diversos valores muito distantes da média nos resultados dos subconjuntos utilizando a estratégia *ping*. Os resultados dos subconjuntos obtidos utilizando a estratégia *core* apresentam menor dispersão. Por exemplo o gráfico do subconjunto `core-150ms- everyround` apresenta alguns resultados mais distantes da média nos dias 4 e 17 de maio, já os dois subconjuntos *ping* apresentam algum resultados distante da média em todas as séries de experimentos.

Observando-se a linha verde nos quatro gráficos, nota-se um comportamento comum aos quatro subconjuntos, com resultados melhores nas primeiras séries de experimentos, e uma piora nos experimentos realizados após o dia 12/05. Esta alteração indica alguma variação no desempenho da rede do PlanetLab, afetando todos os nodos. No entanto, o impacto deste comportamento foi mais suave nos resultados dos subconjuntos *core*. Ainda sobre os subconjuntos *core*, destaca-se um intervalo de confiança menor, tanto para cada série de experimentos (em verde), quanto para as 300 repetições combinando os resultados das 10 séries (sombra cinza).

A Tabela 4.2 apresenta informações referentes aos tempos de execução de todas as 300 repetições da aplicação *Bit.py* com cada subconjunto de nodos. A tabela exibe as informações considerando 100% dos dados obtidos e, adicionalmente, as mesmas informações considerando apenas 95% dos dados, excluindo 2.5% dos resultados mais altos e mais baixos obtidos.

Esta aplicação resultou em coeficientes de variação mais baixos nos resultados dos quatro subconjuntos, mas a estratégia de seleção *core* obteve maior precisão com coeficientes de variação relativamente mais baixos. Observando a última linha da tabela, com 5% dos dados discrepantes descartados, o coeficiente de variação dos subconjuntos *core*

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
100% dos dados				
Média (s)	198,74	196,94	239,71	207,47
Desvio Padrão (s)	21,54	21,58	34,36	36,26
Mínimo (s)	166,86	169,15	202,82	179,22
Máximo (s)	371,02	368,24	524,87	405,59
Coeficiente de variação	0,108	0,109	0,143	0,174
95% dos dados				
Média (s)	196,86	195,02	236,39	202,82
Desvio Padrão (s)	13,21	13,09	22,14	18,86
Mínimo (s)	174,89	175,31	206,37	180,95
Máximo (s)	238,54	237,70	338,23	348,51
Coeficiente de variação	0,067	0,067	0,093	0,092

Tabela 4.2: Sumarização dos experimentos realizados com a aplicação *Bit.py*.

é de 0,067 para ambos, enquanto os subconjuntos *ping* obtiveram 0,093 e 0,092. Ainda observando o último grupo de linhas na tabela, considerando os resultados eliminando 5% dos dados mais discrepantes, os subconjuntos *core* obtiveram valores máximos próximos a 240 segundos, enquanto o tempo máximo obtidos pelos subconjuntos *ping* ultrapassou os 330 segundos.

Já observando-se as médias de tempo de execução, é possível perceber que os subconjuntos *core* obtiveram um tempo de execução médio mais baixo que os subconjuntos *ping*. Considerando os resultados eliminando 5% dos dados discrepantes, as médias de tempo de execução dos subconjuntos *core* foram de 195,86s e 195,02s, contra as médias de 236,39s e 202,82s obtidas pelos subconjuntos *ping*.

Os subconjuntos `core-150ms` e `core-150ms-everyround` exibiram comportamento bastante similar entre si e superiores aos subconjuntos `ping.c3sl` e `ping.uk`, tanto na variação entre as amostras (coeficiente de variação), quanto na média de tempo de execução.

Por ser uma aplicação que faz uso intenso de CPU, este experimento gera um resultado muito significante: a estratégia de seleção de nodos proposta em [13] e descrita no Capítulo 3, com a monitoração da conectividade entre pares de nodos passando pela camada de aplicação, é suficiente para selecionar um subconjunto de nodos mesmo para um experimento com baixo nível de troca de mensagens e intenso uso de CPU, sem a

necessidade de um *benchmark* para avaliação da capacidade de processamento do nodo, monitoramento direto da carga do sistema ou mesmo especificações de hardware.

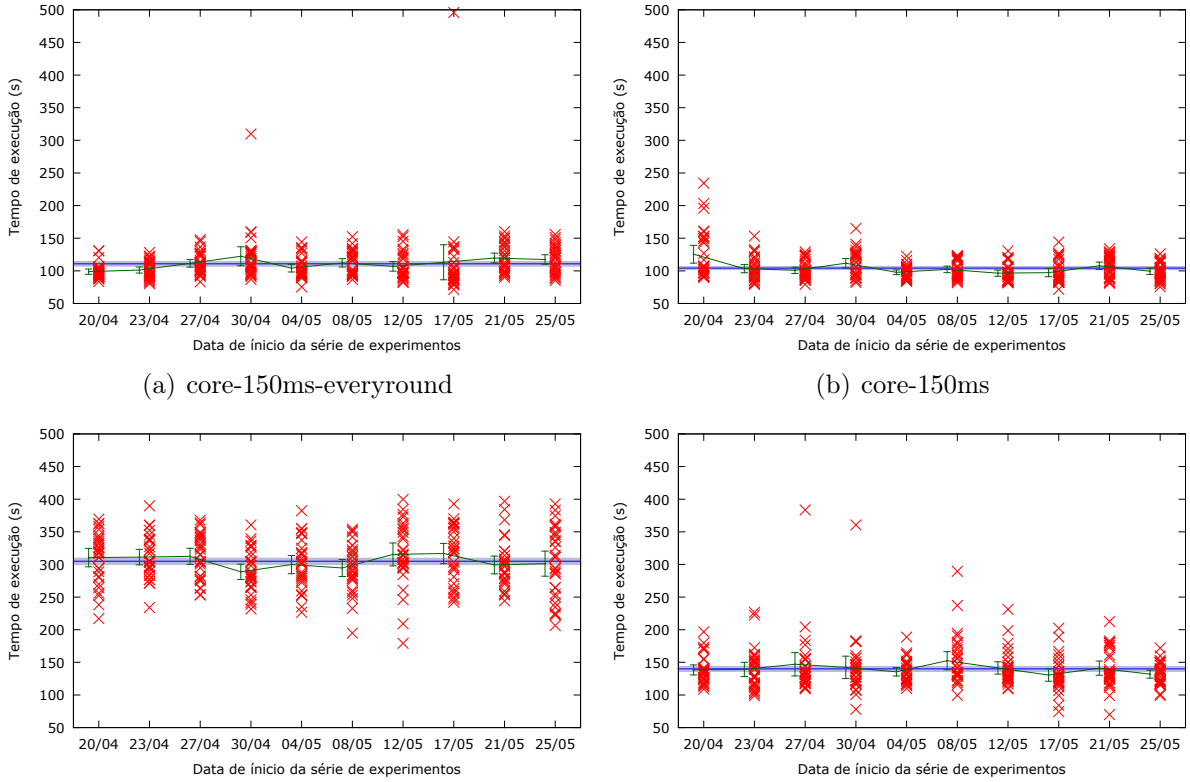
4.2.3 Aplicação 3: *Torrent*

A terceira aplicação utilizada para os experimentos apresentados neste capítulo utiliza uma implementação do protocolo *BitTorrent* [23], o Transmission [45], software padrão nas distribuições Ubuntu atuais. O protocolo *BitTorrent* suporta o compartilhamento de arquivos em redes P2P. A primeira versão do protocolo foi apresentada em 2001 e, atualmente, é uma das aplicações mais comuns para transferência de arquivos.

Para a execução dos experimentos com *torrents* apresentados neste capítulo, foi utilizado um arquivo gerado com dados aleatórios de tamanho 27MB. Este arquivo está, inicialmente, disponível em apenas um *host*. Na fase inicial da execução do experimento, os outros nodos obtém diferentes partes do arquivo a partir do original. A execução procede e a disponibilidade do arquivo aumenta, permitindo que os nodos enviem e recebam partes do arquivo entre si. O protocolo *BitTorrent* apresenta uma topologia em malha.

O Transmission foi executado utilizando as configurações padrões com a exceção de duas opções. A quantidade de transferências simultâneas foi alterada para 150, permitindo que todos os nodos se conectem simultaneamente com todos os nodos durante a execução do experimento. A segunda opção alterada foi o *ratio*. Quando um nodo completa a transferência do arquivo, ele continua transferindo o arquivo para outros nodos, o valor do *ratio* controla por quanto tempo isso ocorre. Um valor de *ratio* de, por exemplo 2, indica que o nodo deve permanecer na rede até que ele transfira o dobro da quantidade de dados recebidos. O valor de *ratio* utilizado nos experimentos apresentados neste capítulo foi de 1.25. Esse valor baixo aumenta a necessidade de cada nodo se comunicar com muitos outros nodos para obter o arquivo completo.

Os gráficos na Figura 4.4 apresentam os resultados obtidos com a execução da aplicação *Torrent*. Foram realizadas 300 rodadas de experimentos, organizadas em 10 séries de 30 repetições da aplicação para cada subconjunto. O eixo y indica o tempo de execução em segundos e o eixo x indica a data de execução de cada série de experimentos. A linha azul



(a) core-150ms-everyround

(b) core-150ms

(c) ping.c3sl

(d) ping.uk

Figura 4.4: Tempos de execução da aplicação Torrent para cada subconjunto de nodos.

cruzando o gráfico representa a média do tempo de execução destas 300 repetições, e a sombra cinza o intervalo de confiança no nível de 95%. A linha verde apresenta a média e intervalo de confiança no nível de 95% para cada série de 30 repetições. Cada x vermelho representa o tempo de execução de um experimento.

Observando-se a distribuição dos x vermelhos, é possível notar uma dispersão menor das amostras para os subconjuntos da estratégia *core*, comparados aos subconjuntos da estratégia *ping*. O subconjunto `core-150ms` apresentou algumas perturbações nos resultados da primeira série de testes (20/04). Já o subconjunto `core-150ms-everyround` obteve duas amostras muito distantes da média, uma em 30/04 e a outra em 17/05. O subconjunto `ping.c3sl` aparente uma maior dispersão das amostras, e essa dispersão se mantém em todas as séries de experimentos. Já o subconjunto `ping.uk` obteve muitas amostras concentradas em torno da média, mas com algumas exceções em todas as séries de experimentos.

A Tabela 4.3 apresenta informações referentes aos tempos de execução de todas as 300

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
100% dos dados				
Média (s)	110,72	104,23	304,97	139,98
Desvio Padrão (s)	31,53	20,31	40,83	32,71
Mínimo (s)	71,40	71,58	179,13	70,00
Máximo (s)	495,91	234,40	399,75	383,48
Coeficiente de variação	0,284	0,194	0,133	0,233
95% dos dados				
Média (s)	108,49	102,79	304,97	137,64
Desvio Padrão (s)	17,53	15,48	36,18	21,36
Mínimo (s)	82,35	81,23	225,15	99,34
Máximo (s)	152,49	152,63	382,36	204,30
Coeficiente de variação	0,161	0,150	0,118	0,155

Tabela 4.3: Sumarização dos experimentos realizados com a aplicação *Torrent*.

repetições da aplicação *Torrent* com cada subconjunto de nodos. Novamente, a tabela exibe as informações considerando 100% dos dados obtidos e, adicionalmente, as mesmas informações considerando apenas 95% dos dados, excluindo 2.5% dos resultados mais altos e mais baixos obtidos.

Observando-se o coeficiente de variação, é possível notar que os experimentos realizados utilizando os subconjuntos *core* não resultaram em uma menor valor. Na última linha da tabela, com 5% dos dados discrepantes descartados, os coeficientes de variação dos subconjuntos *core* são muito similares ao subconjunto `ping.uk`, 0,161, 0,15 e 0,155 respectivamente. Já o subconjunto `ping.c3sl` obteve o menor coeficiente de variação, 0,118, devido a sua elevada média.

Por outro lado, observando-se as médias dos tempos de execução, os subconjuntos *core* obtiveram resultados superiores aos subconjuntos *ping*. Considerando os resultados com 5% dos dados discrepantes descartados, os subconjuntos *core* obtiveram média de 108,49s e 102,79, 3 vezes mais rápidos que o subconjunto `ping.c3sl`, com 304,97s, e 33% mais rápidos que o subconjunto `ping.uk`, com média de 137,64s.

As características de funcionamento do *BitTorrent* são responsáveis por alguns detalhes nos resultados obtidos com este experimento. O algoritmo do *BitTorrent* se adapta às variações da rede entre um experimento e outro, gerando resultados mais consistentes. Desta forma, o alto grau de conectividade nos subconjuntos obtidos com a estratégia *core*

não faz uma diferença tão significativa no coeficiente de variação. No entanto, o *BitTorrent* consegue tirar proveito da conectividade entre os nodos obtidos pela estratégia *core* para completar a transferência de arquivo de forma significativamente mais rápida do que utilizando os nodos obtidos pela estratégia *ping*.

Os experimentos realizados com a aplicação *Torrent* também podem ser analisados utilizando uma outra métrica: a quantidade de nodos que completaram a transferência do arquivo. As três principais razões para que a transferência do arquivo não se complete são: nodo está *offline* no início do experimento (ou se torna *offline* durante o experimento), a falta de espaço em disco e a baixa conectividade entre um nodo e os demais para que a transferência se complete antes do tempo limite do experimento.

Alguns nodos do PlanetLab apresentam espaço em disco 0 bytes. Como nenhuma das estratégias de seleção de nodos utilizadas nestes experimentos considera o espaço em disco como parâmetro de seleção, este problema deve afetar os quatro subconjuntos de nodos de forma semelhante. Portanto a diferença entre o número de nodos que completaram a transferência de arquivos nestes experimentos com a aplicação *Torrent* se deve principalmente à conectividade entre os nodos e a falha em se transferir o arquivo antes do tempo limite do experimento, configurado em 10 minutos para o arquivo de teste de 27MB. Para que o nodo complete a transferência em tempo, ele precisa transferir em média 46KB/s.

Nas 300 repetições do experimento para cada subconjunto de 150 nodos, o subconjunto `core-150ms-everyround` obteve uma média de 142,95 transferências completas, o subconjunto `core-150ms` obteve média de 139,76. Para os subconjuntos da estratégia *ping*, média de 104,66 para o subconjunto `ping.c3sl` e 134,14 para o subconjunto `ping.uk`. Como o subconjunto `core-150ms-everyround` é selecionado antes do início de cada rodada de experimentos, estes nodos foram monitorados pela ferramenta de seleção recentemente e tem maior probabilidade de estarem disponíveis para o início do experimento.

Como apresentado anteriormente na Tabela 4.3, o subconjunto `ping.c3sl` obteve um coeficiente de variação menor do que os outros 3 subconjuntos, os quais obtiveram coeficientes similares entre si. Uma possível causa para este resultado é a participação ativa de um menor número de nodos no experimento, uma vez que uma quantidade menor

de nodos completou a transferência do arquivo e gerou resultados. Das 300 repetições do experimento utilizando o subconjunto `ping.c3sl`, em apenas 33 ocasiões mais de 120 dos 150 nodos completaram a transferência do arquivo. Os subconjuntos `core-150ms-every`, `core-150ms` e `ping.uk` obtiveram mais de 120 dos 150 nodos completando a transferência do arquivo, respectivamente, em 298, 300 e 299 das 300 repetições do experimento.

É importante destacar que o subconjunto `ping.c3sl` contém nodos espalhados pela América do Sul, América do Norte e alguns nodos na Europa, portanto um desempenho mais baixo e uma maior dificuldade em completar os experimentos já eram esperados.

4.3 Discussão

Nas seções anteriores, foram apresentados experimentos realizados durante um período de 40 dias, utilizando aplicações com diferentes perfis de utilização de recursos. *Triangular.py* apresenta baixa uso de CPU e intensa troca de mensagens, enquanto *Bit.py* apresenta intenso uso de CPU e uma quantidade baixa de troca de mensagens. Já o *Torrent* é uma aplicação real, utilizado diariamente por milhões de usuários na Internet.

Em todos os experimentos, os subconjuntos selecionados pela estratégia *core* obtiveram médias de tempo de execução mais baixos que os obtidos pelos subconjuntos selecionados pela estratégia *ping*. O subconjunto `core-150ms-everyround` foi em média 24% mais rápido que o subconjunto `ping.uk` na aplicação *Triangular.py*, 3% na aplicação *Bit.py*, 26% na aplicação *Torrent*. Comparada ao subconjunto `ping.c3sl`, o subconjunto `core-150ms-everyround` foi em média 61% mais rápido na aplicação *Triangular.py*, 20% mais rápido na aplicação *Bit.py* e quase três vezes mais rápido na aplicação *Torrent*.

A estratégia *core* também foi mais eficiente em minimizar o impacto de fatores externos nos resultados dos experimentos, reduzindo o coeficiente de variação dos resultados obtidos. O subconjunto `core-150ms` obteve um coeficiente de variação de 0,65 na aplicação *Triangular.py*, contra 0,916 do subconjunto `ping.uk` e 0,841 do subconjunto `ping.c3sl`. Os coeficientes obtidos na aplicação *Bit.py* também foram menores, 0,067 do subconjunto `core-150ms` contra 0,092 e 0,093 dos conjuntos `ping.uk` e `ping.c3sl`.

O menor coeficiente de variação indica que a utilização de nodos selecionados pela

ferramenta de monitoramento reduz o impacto das instabilidades da rede sobre os experimentos, aumentando a precisão dos resultados obtidos e contribuindo para a repetibilidade dos experimentos.

Comparando os dois subconjuntos obtidos pela estratégia *core*, os resultados obtidos são bastante similares, com médias de tempo de execução equivalentes nas aplicações *Triangular.py* e *Bit.py*, e coeficientes de variação similares nas aplicações *Bit.py* e *Torrent*. Os resultados obtidos pelo subconjunto `core-150ms` demonstram a estabilidade dos nodos selecionados pela estratégia *core*. Durante um período de 40 dias, o subconjunto permaneceu fixo e manteve as propriedades de conectividade entre os nodos, obtendo resultados com baixa variação durante o período de experimentação.

A estratégia de seleção de nodos `core-150ms-everyround` inclui um número maior de nodos no experimento. Como foi observado em um período de três dias, a seleção de 150 nodos executada antes do início de cada uma das 30 rodadas de experimento de uma série incluiu um total de 262 nodos diferentes. É esperado que a utilização de diferentes nodos resulte em uma maior variação dos resultados. No entanto, os resultados obtidos pelo subconjunto `core-150ms-everyround` foram muito similares ao subconjunto fixo `core-150ms`. Portanto, a estratégia de seleção de nodos estáveis é capaz de selecionar subconjuntos de nodos com características de estabilidade muito semelhantes apenas utilizando os mesmos parâmetros de seleção. A capacidade de encontrar subconjuntos com características de conectividade semelhantes utilizando apenas os mesmos parâmetros de seleção também contribui para a reprodutibilidade dos experimentos.

CAPÍTULO 5

CONCLUSÃO

Novas soluções, arquiteturas e serviços para a Internet precisam ser testados em um ambiente de larga escala. O PlanetLab é um *testbed* de escala planetária, com mais de mil nodos espalhados ao redor do mundo, o qual oferece aos pesquisadores um ambiente real para a execução de experimentos. No entanto, a execução de experimentos distribuídos no PlanetLab pode atingir um alto grau de complexidade, especialmente por envolver uma grande quantidade de nodos e a existência de instabilidades e variações de desempenho nos nodos e na própria rede. Este trabalho de dissertação apresentou uma avaliação experimental do impacto do uso de uma ferramenta de seleção na precisão dos resultados obtidos em experimentos realizados no PlanetLab.

A ferramenta de seleção de nodos estudada neste trabalho monitora a estabilidade das conexões entre os pares de nodos PlanetLab. A estratégia de seleção *core* constrói um grafo utilizando as informações monitoradas e obtém um subconjunto de nodos estáveis para a realização de experimentos. Os experimentos realizados neste trabalho utilizaram três aplicações com diferentes perfis de utilização de recursos e quatro subconjuntos de nodos foram obtidos a fim de comparar a estratégia de seleção *core* a uma estratégia alternativa. Durante um período de 40 dias, cada aplicação foi executada 300 vezes utilizando cada subconjunto de nodos.

Os resultados obtidos confirmaram que o PlanetLab é um ambiente instável e é muito difícil obter repetibilidade nos experimentos realizados. No entanto, estas variações não atingem todos os nodos da mesma maneira e com a mesma intensidade. A utilização de subconjuntos de nodos obtidos pela estratégia de seleção *core* reduziu as variações observadas nos resultados obtidos, aumentando a precisão e contribuindo para a repetibilidade dos experimentos. Além disso, os subconjuntos de nodos selecionados pela ferramenta de monitoração obtiveram melhores médias de tempo de execução para as aplicações testa-

das, demonstrando a capacidade da estratégia de seleção de encontrar um subconjunto de nodos com melhor conectividade entre todos os pares, confirmando os resultados originais obtidos pelo autor da ferramenta. Os resultados demonstraram ainda que a ferramenta de monitoramento é capaz de selecionar um subconjunto de nodos que resulte em experimentos mais rápidos e com resultados mais consistentes mesmo para uma aplicação com intenso uso de CPU e quantidade reduzida de troca de mensagens, sem a necessidade de execução de *benchmarks* para avaliar a capacidade de processamento do nodo, monitoração direta da carga de CPU ou mesmo especificações de hardware. Os experimentos demonstraram ainda a capacidade da ferramenta de monitoramento de selecionar conjuntos de nodos com desempenho muito similar durante um longo período de tempo. A ferramenta obtém conjuntos similares apenas utilizando os mesmos parâmetros de seleção, contribuindo para a reprodutibilidade dos experimentos.

O planejamento de experimentos, gerenciamento de nodos e a execução e monitoramento de experimentos no PlanetLab são atividades complexas. Este trabalho ofereceu como contribuição à comunidade de pesquisadores usuários do PlanetLab uma proposta de integração da ferramenta de seleção de nodos estudada ao portal de gerenciamento de experimentos PlanetMon. A versão integrada do PlanetMon é uma forma conveniente de se gerenciar experimentos no PlanetLab utilizando subconjuntos de nodos estáveis, selecionados pela ferramenta de monitoração de forma transparente.

Os experimentos realizados neste trabalho demonstraram que a ferramenta de seleção de nodos baseada no monitoramento das interações fim-a-fim é eficiente em obter subconjuntos de nodos estáveis para a realização de experimentos. Porém, existem outras informações sobre os nodos que a ferramenta de seleção não foi projetada para monitorar. Durante a execução dos experimentos, foi detectada especialmente a falta de informações relacionadas à disponibilidade de memória e espaço em disco nos nodos. Como um trabalho futuro, para que o PlanetMon integrado à ferramenta de monitoração se torne uma solução mais completa para a execução de experimentos no PlanetLab, seria importante que a monitoração destes parâmetros fosse incorporada ao PlanetMon e utilizada para filtrar os nodos selecionados pela ferramenta de seleção.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] J. Albrecht e D. Y. Huang. Managing distributed applications using gush. *Testbeds and Research Infrastructures. Development of Networks and Communities*, páginas 401–411. Springer, 2011.

[2] J. Albrecht, D. Oppenheimer, A. Vahdat, e D.A. Patterson. Design and implementation trade-offs for wide-area resource discovery. *ACM Transactions on Internet Technology (TOIT)*, 8(4):18, 2008.

[3] B. Barros Neto e I.S. Scarminio. *Como Fazer Experimentos: pesquisa e desenvolvimento na ciência e indústria*. Unicamp, 2002.

[4] A. Bavier, N. Feamster, M. Huang, L. Peterson, e J. Rexford. In vini veritas: Realistic and controlled network experimentation. *SIGCOMM Comput. Commun. Rev.*, 36(4):3–14, 2006.

[5] L.C.E. Bona, K.V.O. Fonseca, e E.P. Duarte Jr. Hyperbone: a scalable overlay network based on a virtual hypercube. *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, páginas 1025–1030. IEEE, 2008.

[6] G.E.P. Box, W.G. Hunter, e J.S. Hunter. *Statistics for experimenters: an introduction to design, data analysis, and model building*. Wiley series in probability and mathematical statistics. Wiley, 2005.

[7] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, e M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[8] J. Dean e S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI 2004*, páginas 137–150, 2004.

[9] S.E. Deering. RFC 2460: Internet protocol, version 6 (ipv6) specification, 1998.

[10] E.P. Duarte Jr, T. Garrett, L.C.E. Bona, R. Carmo, e A.P. Zuge. Finding stable cliques of planetlab nodes. *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, páginas 317–322. IEEE, 2010.

[11] D. Eastlake 3rd. RFC 6234: Secure hash algorithm, 2001.

[12] A. Feldmann. Internet clean-slate design: what and why? *ACM SIGCOMM Computer Communication Review*, 37(3):59–64, 2007.

[13] T. Garrett. Seleção de nodos para a execução de experimentos no planetlab baseada no monitoramento de estabilidade das interações fim-a-fim. Dissertação de Mestrado, Universidade Federal do Paraná, 2011.

[14] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley Professional Computing. Wiley, 1991.

[15] V. Lenders e M. Martonosi. Repeatable and realistic experimentation in mobile wireless networks. *Mobile Computing, IEEE Transactions on*, 8(12):1718–1728, Dezembro de 2009.

[16] J. Pan, S. Paul, e R. Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26–36, 2011.

[17] C.E. Perkins. Mobile ip. *Communications Magazine, IEEE*, 35(5):84–99, 1997.

[18] V.K. Ruoso, L.C.E Bona, e E.P. Duarte Jr. Planetmon: Um arcabouço para a execução e monitoração de experimentos no planetlab. *7o Workshop de Redes Dinâmicas e Sistemas Peer-to-Peer (WP2P)*, páginas 31–43. 29o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2011.

[19] N. Spring, L. Peterson, A. Bavier, e V. Pai. Using planetlab for network research: Myths, realities, and best practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24, 2006.

[20] Wang. Y., A Carzaniga, e A.L. Wolf. Four enhancements to automateddistributed system experimentation methods. *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on*, páginas 491–500, Maio de 2008.

[21] T. Ylonen. Rfc 4251: The secure shell (ssh) protocol architecture, 2006.

[22] Bitcoin. `https://bitcoin.org/en/`. Acessado em julho de 2014.

[23] Bittorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`. Acessado em julho de 2014.

[24] Emulab. `http://www.emulab.net`. Acessado em julho de 2014.

[25] Fibre project. `http://fibre-ict.eu`. Acessado em junho de 2014.

[26] Future internet research and experimentation. `http://www.ict-fire.eu`. Acessado em junho de 2014.

[27] Geni user shell. `http://gush.cs.williams.edu/trac/gush`. Acessado em julho de 2013.

[28] Global environment for network innovations (geni). `http://www.geni.net`. Acessado em junho de 2014.

[29] Gloriad. `http://www.gloriad.org/gloriaddrupal/`. Acessado em julho de 2014.

[30] Kreonet. `http://www.kreonet.net/e_service0101/`. Acessado em julho de 2014.

[31] Lightweight mapreduce in python. `https://github.com/michaelfairley/mincemeatpy`. Acessado em julho de 2014.

[32] Management overlay networks (mon). `http://cairo.cs.uiuc.edu/projects/mon/`. Acessado em junho de 2014.

[33] A monitoring infrastructure for planetlab (comon). `http://comon.cs.princeton.edu`. Acessado em junho de 2014.

[34] Myops node list. http://monitor.planet-lab.org/monitor/node. Acessado em julho de 2014.

[35] Myops node list PLE. http://monitor.planet-lab.eu/monitor/node. Acessado em julho de 2014.

[36] National science foundation. http://www.nsf.gov. Acessado em junho de 2014.

[37] Nixes tool set. http://aqualab.cs.northwestern.edu/projects/149-nixes-tool-set. Acessado em junho de 2014.

[38] Open networking foundation. https://www.opennetworking.org. Acessado em junho de 2014.

[39] Openssh. http://www.openssh.com. Acessado em junho de 2014.

[40] Planetary scale control plane. http://research.cs.washington.edu/networking/cplane/. Acessado em junho de 2014.

[41] Planetlab. https://www.planet-lab.org. Acessado em julho de 2014.

[42] Planetlab user shell. http://plush.cs.williams.edu. Acessado em julho de 2013.

[43] Pssh. https://code.google.com/p/parallel-ssh/. Acessado em junho de 2014.

[44] Stork project. http://www.cs.arizona.edu/stork/tutmain.html. Acessado em junho de 2014.

[45] Transmission. http://www.transmissionbt.com. Acessado em julho de 2014.

[46] Triangular numbers. http://mathworld.wolfram.com/TriangularNumber.html. Acessado em julho de 2014.

[47] AKARI. http://www.nict.go.jp/en/photonic_nw/archi/akari/akari-top_e.html. Acessado em junho de 2014.

[48] JGN-X. http://www.jgn.nict.go.jp/jgn2plus_archive/english/about_us/index.html. Acessado em junho de 2014.

APÊNDICE A: SUBCONJUNTOS DE NODOS DO PLANETLAB UTILIZADOS NOS EXPERIMENTOS

Subconjunto core-150ms

1. pl2.cs.yale.edu
2. onelab3.warsaw.rd.tp.pl
3. planetlab1.cs.uit.no
4. planetlab-04.cs.princeton.edu
5. planetlab1.tau.ac.il
6. planet-plc-5.mpi-sws.org
7. planetlab1.temple.edu
8. planetlab2.wiwi.hu-berlin.de
9. planetlab2.xeno.cl.cam.ac.uk
10. pl1.cs.yale.edu
11. planetlab1.cesnet.cz
12. planetlab1.rd.tut.fi
13. planetlab-01.bu.edu
14. planetvs1.informatik.uni-stuttgart.de
15. plab2.ple.silweb.pl
16. planetlab2.informatik.uni-wuerzburg.de
17. inriarennes1.irisa.fr
18. host4-plb.loria.fr
19. planetlab3.xeno.cl.cam.ac.uk
20. onelab4.warsaw.rd.tp.pl
21. ops.ii.uam.es
22. planetlab1.nrl.eecs.qmul.ac.uk
23. planetlab2.nrl.eecs.qmul.ac.uk
24. planetlab4.hiit.fi
25. planetlabpc0.upf.edu
26. planetlab4.cs.st-andrews.ac.uk
27. planetlab2.cesnet.cz
28. planetlab2.ionio.gr
29. onelab3.info.ucl.ac.be
30. planetlab1.xeno.cl.cam.ac.uk
31. peeramide.irisa.fr
32. planetlab01.tkn.tu-berlin.de
33. planetlab-13.e5.ijs.si
34. planetlab-coffee.ait.ie
35. ple2.tu.koszalin.pl
36. planet2.elte.hu
37. planetlab1.sics.se
38. planetlab1.lkn.ei.tum.de
39. aguila2.lsi.upc.edu
40. planet-plc-4.mpi-sws.org
41. planetlab1.wiwi.hu-berlin.de
42. aguila1.lsi.upc.edu
43. planetlab1.montefiore.ulg.ac.be
44. planetlab1.cs.pitt.edu
45. planetlab1.jhu.edu
46. planetlab2.dit.upm.es
47. ple3.ipv6.lip6.fr
48. planck227ple.test.ibbt.be
49. zoi.di.uoa.gr
50. kostis.di.uoa.gr
51. merkur.planetlab.haw-hamburg.de
52. planetlab1.exp-math.uni-essen.de
53. planetlabtwo.ccs.neu.edu
54. planet1.l3s.uni-hannover.de
55. whitefall.planetlab.cs.umd.edu
56. plewifi.ipv6.lip6.fr
57. ple6.ipv6.lip6.fr
58. ple2.ipv6.lip6.fr
59. plab-2.diegm.uniud.it
60. planetvs2.informatik.uni-stuttgart.de
61. osiris.planetlab.cs.umd.edu
62. planet1.servers.ua.pt
63. planetlab1.tmit.bme.hu
64. plab-1.diegm.uniud.it
65. planetlab-2.ida.liu.se
66. planetlab2.mini.pw.edu.pl
67. planetlab-02.bu.edu
68. peeramidion.irisa.fr
69. planetlab3.inf.ethz.ch
70. plab4.ple.silweb.pl
71. planetlab1.just.edu.jo
72. planetlab2.cs.vu.nl
73. node1.planetlab.albany.edu
74. planetlab2.fct.ualg.pt
75. planetlab1.ionio.gr
76. planetlab2.unineuchatel.ch
77. planetlab2.jhu.edu
78. planetlab1.ifi.uio.no
79. planetlab2.ifi.uio.no
80. planetlab1.cs.vu.nl
81. dfn-ple1.x-win.dfn.de
82. onelab2.warsaw.rd.tp.pl
83. planetlab2.sics.se
84. vicky.planetlab.ntua.gr
85. planet1.unipr.it
86. ple2.dmcs.p.lodz.pl
87. planetlab1.unineuchatel.ch
88. planetlab2.montefiore.ulg.ac.be
89. planet2.servers.ua.pt
90. planetlab-node-01.ucd.ie
91. inriarennes2.irisa.fr
92. planetlab2.exp-math.uni-essen.de
93. planetlab2.pjwstk.edu.pl
94. host1.planetlab.informatik.tu-darmstadt.de
95. planetlab1.informatik.uni-wuerzburg.de

96. rochefort.infonet.fundp.ac.be
97. iraplab1.iralab.uni-karlsruhe.de
98. planetlab3.hiit.fi
99. gschembra4.diit.unict.it
100. planetlab-1.research.netlab.hut.fi
101. onelab2.info.ucl.ac.be
102. planetlab2.ci.pwr.wroc.pl
103. planetlab01.alucloud.com
104. planetlab-4.imperial.ac.uk
105. planetlab2.cs.uit.no
106. dplanet2.uoc.edu
107. planetlab-12.e5.ijs.si
108. iraplab2.iralab.uni-karlsruhe.de
109. planetlab1.extern.kuleuven.be
110. planetlab2.extern.kuleuven.be
111. orval.infonet.fundp.ac.be
112. planetlab02.tkn.tu-berlin.de
113. planetlab2.urv.cat
114. prata.mimuw.edu.pl
115. ple1.dmcs.p.lodz.pl
116. pl001.ece.upatras.gr
117. planetlabeu-1.tssg.org
118. planetlab1.ci.pwr.wroc.pl
119. miranda.planetlab.cs.umd.edu
120. planetlab2.rd.tut.fi
121. planetlab1.utt.fr
122. dplanet1.uoc.edu
123. planetlab2.utt.fr
124. planetlab02.alucloud.com
125. planetlabeu-2.tssg.org
126. planetlab3.upc.es
127. planetlab1.eecs.jacobs-university.de
128. planet2.unipr.it
129. pl002.ece.upatras.gr
130. planetlab-node1.it-sudparis.eu
131. planetlab-node3.it-sudparis.eu
132. planetlab1.warsaw.rd.tp.pl
133. ple2.cesnet.cz
134. ple1.cesnet.cz
135. lefthand.eecs.harvard.edu
136. planetlab1.urv.cat
137. planet1.elte.hu
138. planetlab1.di.fct.unl.pt
139. salt.planetlab.cs.umd.edu
140. planetlab3.mini.pw.edu.pl
141. planetlab4.inf.ethz.ch
142. host2.planetlab.informatik.tu-darmstadt.de
143. planetlab1.u-strasbg.fr
144. planetlab2.u-strasbg.fr
145. roti.mimuw.edu.pl
146. planetlab4.mini.pw.edu.pl
147. pl2.rcc.uottawa.ca
148. pl2.uni-rostock.de
149. planetlab1.mini.pw.edu.pl
150. planetlab-3.imperial.ac.uk

Subconjunto ping.c3sl

1. planetlab1-saopaulo.lan.redclara.net
2. planetlab2-saopaulo.lan.redclara.net
3. planetlab1.pop-mg.rnp.br
4. planetlab2.pop-mg.rnp.br
5. planetlab2-buenosaires.lan.redclara.net
6. planet-lab4.uba.ar
7. planet-lab1.uba.ar
8. planetlab2-santiago.lan.redclara.net
9. planetlab1-santiago.lan.redclara.net
10. planet-lab3.uba.ar
11. planet-lab2.uba.ar
12. planetlab2.pop-pa.rnp.br
13. h-200-129-132-18.pop-pa.rnp.br
14. planetlab2.cti.espol.edu.ec
15. planetlab1.cti.espol.edu.ec
16. node2.planetlab.mathcs.emory.edu
17. pl2.eecs.utk.edu
18. pl1.eecs.utk.edu
19. planetlab02.alucloud.com
20. planetlab01.alucloud.com
21. endor.gpolab.bbn.com
22. plnode-04.gpolab.bbn.com
23. ebb.colgate.edu
24. planet4.cc.gt.atl.ga.us
25. planet2.cc.gt.atl.ga.us
26. planet1.cc.gt.atl.ga.us
27. planetlab2.clemson.edu
28. planetlab1.clemson.edu
29. pl1.cis.uab.edu
30. pl2.cis.uab.edu
31. planetlab-02.bu.edu
32. planetlab-01.bu.edu
33. planetlab-02.vt.nodes.planet-lab.org
34. planetlab-01.vt.nodes.planet-lab.org
35. planetlab-03.vt.nodes.planet-lab.org
36. planetlab-04.vt.nodes.planet-lab.org
37. planetlab1.temple.edu
38. planetlab2.cnis.nyit.edu
39. planetlab1.cnis.nyit.edu
40. pllx2.parc.xerox.com
41. pllx1.parc.xerox.com
42. pl-node-0.csl.sri.com
43. pli1-pa-6.hpl.hp.com
44. pli1-pa-4.hpl.hp.com
45. pl-node-1.csl.sri.com
46. planetlab2.poly.edu
47. planetlab1.eecs.ucf.edu

48. planetlab2.eecs.ucf.edu
49. node2.planetlab.albany.edu
50. salt.planetlab.cs.umd.edu
51. planetlab3.cs.columbia.edu
52. whitefall.planetlab.cs.umd.edu
53. miranda.planetlab.cs.umd.edu
54. node1.planetlab.albany.edu
55. osiris.planetlab.cs.umd.edu
56. planetlab2.acis.ufl.edu
57. planetlab1.cis.upenn.edu
58. planetlab2.cis.upenn.edu
59. planetlab1.acis.ufl.edu
60. planetlab1.rutgers.edu
61. planetlab2.rutgers.edu
62. planetlab-04.cs.princeton.edu
63. righthand.eecs.harvard.edu
64. planetlab1.cs.stevens-tech.edu
65. lefthand.eecs.harvard.edu
66. planetlabone.ccs.neu.edu
67. planetlabtwo.ccs.neu.edu
68. planetlab1.cs.uml.edu
69. planetlab2.cs.uml.edu
70. pluto.cs.brown.edu
71. planetlab2.cs.umass.edu
72. planetlab1.cs.umass.edu
73. earth.cs.brown.edu
74. jupiter.cs.brown.edu
75. pl1.cs.yale.edu
76. pl1.csl.utoronto.ca
77. pl2.cs.yale.edu
78. planetlab1.cs.pitt.edu
79. planetlab-3.cmcl.cs.cmu.edu
80. planetlab-1.cmcl.cs.cmu.edu
81. planetlab-2.cmcl.cs.cmu.edu
82. planetlab2.cs.pitt.edu
83. planetlab1.tsuniv.edu
84. planetlab1.jhu.edu
85. planetlab2.jhu.edu
86. planetlab1.cnds.jhu.edu
87. planetlab3.cnds.jhu.edu
88. planetlab2.cnds.jhu.edu
89. planetlab2.cs.unc.edu
90. planetlab2.csuohio.edu
91. planetlab1.cs.unc.edu
92. planetlab4.cs.uiuc.edu
93. planetlab2.cs.uiuc.edu
94. planetlab2.tsuniv.edu
95. planetlab1.csuohio.edu
96. planetlab1.cs.uiuc.edu
97. plink.cs.uwaterloo.ca
98. pl1.rcc.uottawa.ca
99. planetlab2.cs.purdue.edu
100. pl2.rcc.uottawa.ca
101. planetlab1.cs.purdue.edu
102. jupiter.planetlab.carleton.ca
103. saturn.planetlab.carleton.ca
104. planetlab-4.eecs.cwru.edu
105. planetlab2.utdallas.edu
106. planetlab-2.cse.ohio-state.edu
107. kupl1.ittc.ku.edu
108. kupl2.ittc.ku.edu
109. planetlab-5.eecs.cwru.edu
110. planetlab-1.cse.ohio-state.edu
111. planetlab1.thlab.net
112. planetlab-1.cs.uic.edu
113. planetlab-2.cs.uic.edu
114. planetlab4.wail.wisc.edu
115. planetlab3.wail.wisc.edu
116. planetlab1.utdallas.edu
117. plab4.eece.ksu.edu
118. planetlab1.uta.edu
119. kc-sce-plab1.umkc.edu
120. pl1.bell-labs.fr
121. kc-sce-plab2.umkc.edu
122. plab3.eece.ksu.edu
123. planetlab2.thlab.net
124. planetlab2.cs.okstate.edu
125. planetlab1.cs.okstate.edu
126. planetlab2.dit.upm.es
127. planetlab2.netlab.uky.edu
128. planetlab1.netlab.uky.edu
129. ops.ii.uam.es
130. utet.ii.uam.es
131. planetlab1.csee.usf.edu
132. planetlab2.csee.usf.edu
133. planetlab4.csee.usf.edu
134. planetlab6.csee.usf.edu
135. planetlab3.csee.usf.edu
136. planetlab5.csee.usf.edu
137. roam1.cs.ou.edu
138. planetlab1.eecs.umich.edu
139. aguila1.lsi.upc.edu
140. aguila2.lsi.upc.edu
141. planetlab1.upc.es
142. planetlab2.upc.es
143. planetlab3.upc.es
144. planetlabpc0.upf.edu
145. ricepl-2.cs.rice.edu
146. ricepl-1.cs.rice.edu
147. ricepl-5.cs.rice.edu
148. vn4.cse.wustl.edu
149. planetlab2.urv.cat
150. planetlab1.urv.cat

Subconjunto ping.uk

1. planetlab2.xeno.cl.cam.ac.uk
2. planetlab3.xeno.cl.cam.ac.uk
3. planetlab1.xeno.cl.cam.ac.uk
4. planetlab1.nrl.eecs.qmul.ac.uk
5. planetlab2.nrl.eecs.qmul.ac.uk
6. planetlab-1.imperial.ac.uk
7. planetlab-3.imperial.ac.uk
8. planetlab-4.imperial.ac.uk
9. planetlab-2.imperial.ac.uk
10. planetlab4.cs.st-andrews.ac.uk
11. planetlab3.cs.st-andrews.ac.uk
12. pl2.ccsrfi.net
13. planetlab-node-01.ucd.ie
14. planetlab2.cs.vu.nl
15. planetlab1.cs.vu.nl
16. planetlab1.extern.kuleuven.be
17. planetlab2.extern.kuleuven.be
18. planetlab-coffee.ait.ie
19. planetlab-tea.ait.ie
20. planck227ple.test.ibbt.be
21. planetlab2.montefiore.ulg.ac.be
22. planetlab1.montefiore.ulg.ac.be
23. planetlab-node3.it-sudparis.eu
24. planetlab-node1.it-sudparis.eu
25. planetlabeu-1.tssg.org
26. planetlabeu-2.tssg.org
27. onelab3.info.ucl.ac.be
28. onelab2.info.ucl.ac.be
29. plewifi.ipv6.lip6.fr
30. onelab1.info.ucl.ac.be
31. ple4.ipv6.lip6.fr
32. ple2.ipv6.lip6.fr
33. ple3.ipv6.lip6.fr
34. ple6.ipv6.lip6.fr
35. rochefort.infonet.fundp.ac.be
36. orval.infonet.fundp.ac.be
37. host3-plb.loria.fr
38. host4-plb.loria.fr
39. inriarennes2.irisa.fr
40. peeramidion.irisa.fr
41. peeramide.irisa.fr
42. inriarennes1.irisa.fr
43. pl1.bell-labs.fr
44. planetlab1.thlab.net
45. planetlab2.thlab.net
46. planet1.l3s.uni-hannover.de
47. planet2.l3s.uni-hannover.de
48. planetlab2.unineuchatel.ch
49. planetlab1.unineuchatel.ch
50. planetlab-1.imag.fr
51. planetlab1.inf.ethz.ch
52. planetlab4.inf.ethz.ch
53. planetlab1.utt.fr
54. planetlab3.inf.ethz.ch
55. planetlab2.utt.fr
56. planetlab01.tkn.tu-berlin.de
57. planetlab02.tkn.tu-berlin.de
58. node2pl.planet-lab.telecom-lille1.eu
59. node1pl.planet-lab.telecom-lille1.eu
60. planetlab1.wiwi.hu-berlin.de
61. planetlab2.wiwi.hu-berlin.de
62. planetlab2.diku.dk
63. planetlab2.u-strasbg.fr
64. planetlab1.u-strasbg.fr
65. planet1.inf.tu-dresden.de
66. planet2.inf.tu-dresden.de
67. planetlab2.eurecom.fr
68. planetlab1.eurecom.fr
69. planetlab2.cesnet.cz
70. ple2.cesnet.cz
71. ple1.cesnet.cz
72. planetlab1.cesnet.cz
73. merkur.planetlab.haw-hamburg.de
74. mars.planetlab.haw-hamburg.de
75. planetlab1.di.unito.it
76. planetlab3.di.unito.it
77. onelab10.pl.sophia.inria.fr
78. planetlab-2.ing.unimo.it
79. pl2.uni-rostock.de
80. pl1.uni-rostock.de
81. planet2.unipr.it
82. planetlab-1.ing.unimo.it
83. planet1.unipr.it
84. planetlab1.informatik.uni-goettingen.de
85. plab2.create-net.org
86. dfn-ple1.x-win.dfn.de
87. planetlab2.science.unitn.it
88. planet2.elte.hu
89. planetlab1.science.unitn.it
90. planet1.elte.hu
91. plab1.create-net.org
92. planetlab1.eecs.jacobs-university.de
93. planetlab1.tmit.bme.hu
94. planetlab3.informatik.uni-erlangen.de
95. planetlab1.s3.kth.se
96. planetlab1.sics.se
97. planetlab2.s3.kth.se
98. planetlab-2.ssvl.kth.se
99. planetlab2.sics.se
100. planetlab1.informatik.uni-erlangen.de
101. planetlab-1.ssvl.kth.se
102. planetlab1.informatik.uni-wuerzburg.de
103. planetlab2.informatik.uni-wuerzburg.de

104. planetlab2.dit.upm.es
105. planetlab1.exp-math.uni-essen.de
106. planetlab2.exp-math.uni-essen.de
107. plab-1.diegm.uniud.it
108. ops.ii.uam.es
109. utet.ii.uam.es
110. planetlab-2.ida.liu.se
111. planet-lab-node2.netgroup.uniroma2.it
112. planet-lab-node1.netgroup.uniroma2.it
113. planetlab2.ifi.uio.no
114. planetlab1.ifi.uio.no
115. plab-2.diegm.uniud.it
116. planetvs2.informatik.uni-stuttgart.de
117. planetvs1.informatik.uni-stuttgart.de
118. planetlab-2.man.poznan.pl
119. planetlab-1.research.netlab.hut.fi
120. planetlab-2.research.netlab.hut.fi
121. planetlab4.hiit.fi
122. planetlab3.hiit.fi
123. planetlab1.warsaw.rd.tp.pl
124. onelab4.warsaw.rd.tp.pl
125. onelab2.warsaw.rd.tp.pl
126. onelab3.warsaw.rd.tp.pl
127. iraplab2.iralab.uni-karlsruhe.de
128. iraplab1.iralab.uni-karlsruhe.de
129. planetlab-12.e5.ijs.si
130. planetlab-13.e5.ijs.si
131. planetlab2.rd.tut.fi
132. planetlab1.rd.tut.fi
133. prata.mimuw.edu.pl
134. roti.mimuw.edu.pl
135. planetlab1.pjwstk.edu.pl
136. planetlab2.lkn.ei.tum.de
137. planetlab2.mini.pw.edu.pl
138. planetlab1.lkn.ei.tum.de
139. planetlab2.pjwstk.edu.pl
140. planetlab3.mini.pw.edu.pl
141. planetlab4.mini.pw.edu.pl
142. planetlab2.ci.pwr.wroc.pl
143. planetlab1.ci.pwr.wroc.pl
144. aguila2.lsi.upc.edu
145. aguila1.lsi.upc.edu
146. planetlab3.upc.es
147. planetlab1.upc.es
148. planetlab2.upc.es
149. planet-plc-1.mpi-sws.org
150. planetlab2.virtues.fi