Ricardo Martins

Criação de uma infraestrutura para execução remota do Netkit

São José – SC

Agosto / 2014

Ricardo Martins

Criação de uma infraestrutura para execução remota do Netkit

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Marcelo Maia Sobral, Dr.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Agosto / 2014

Monografia sob o título *"Criação de uma infraestrutura para execução remota do Netkit"*, defendida por Ricardo Martins e aprovada em 22 de agosto de 2014, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Marcelo Maia Sobral, Dr.
Orientador

Prof. Emerson Ribeiro de Mello, Dr.
IFSC

Prof. Eraldo Silveira e Silva, Dr.
IFSC

Todo homem que encontro é superior a mim em alguma coisa. Por isso, dele sempre aprendo alguma coisa. R. W. Emerson

Agradecimentos

Dedico meu agradecimento ao IFSC, instituição de ensino em que tenho orgulho de ter estudado.

A todos os professores, por colaborarem pelo meu desenvolvimento como pessoa e como aluno.

Ao meu orientador Marcelo Maia Sobral, por ter a paciencia de me aguentar por todo esse tempo pelo suporte prestado e por suas várias correções e incentivos.

E a todos, amigos e familiares, que perderam tempo me ajudando a escrever esta monografia.

Resumo

O Netkit2 é um ambiente de criação de redes virtuais criado para facilitar a realização de experimentos com redes virtuais. O Netkit2 se apresenta como um programa que, para ser utilizado, deve ser instalado em um computador com sistema operacional Linux. Este trabalho propõe a elaboração de uma infraestrutura para a execução remota do Netkit2 em um ou mais servidores, o que deve tornar possível a criação de um laboratório virtual de redes de computadores. Para isso, propõe-se uma extensão para o Netkit2 capaz de executar, controlar e oferecer uma interface de acesso remoto para experimentos com redes de computadores virtuais, denominadas instâncias do Netkit.

Do ponto de vista do usuário, o sistema proposto se apresenta como um serviço web, o qual segue o estilo de arquitetura conhecido como *Representational State Transfer* (REST). Assim, o sistema é modelado como um conjunto de recursos, tais como redes virtuais, instâncias e máquinas virtuais, dentre outros, os quais podem se acessados, criados, removidos e modificados por meio de operações usando o protocolo *Hypertext Transfer Protocol* (HTTP). Isso deve possibilitar que se criem interfaces web para acesso ao sistema, o que evita a instalação de software nos dispositivos dos usuários, as quais, no momento, não são objeto de estudo neste trabalho.

Para fins de demonstração, desenvolveu-se um protótipo do sistema capaz de interpretar e executar um subconjunto das operações da interface de acesso especificada. Esse protótipo foi capaz de interpretar comandos para publicar configurações de experimentos no servidor, iniciar e terminar instâncias de redes virtuais do Netkit2, além de interagir com essas instâncias.

Palavras chave: Serviços web, Netkit2, REST, redes virtuais.

Abstract

The Netkit2 is a virtual environment for creating networks made to facilitate the realization of experiments with virtual networks. The Netkit2 is a program that should be installed on a computer with Linux operating system to be used. This paper proposes the development of an remote infrastructure for execution of Netkit2 on one or more servers, which should make it possible to create a virtual laboratory of computer networks. For this propose, will be create an extension to the Netkit2 to able execute, control and provide a remote access interface for experiments with networks of virtual machines, called Netkit instances.

From the user point of view, the proposed system is presented as a web service, which follows the architectural style known as REST. Thus, the system is modeled as a set of resources such as virtual networking, forums and virtual machines, among others, which can be accessed, created, removed and modified by a process using HTTP protocol. This should enable you to create web interfaces for accessing the system, which prevents the installation of software on the devices of users is not necessary, which are not object of study in this work.

For demonstration purposes, in this paper was developed a prototype system capable of interpreting and executing a subset of the specified access interface operations. This prototype was able to interpret commands to publish experiments settings on the server, start and stop instances of virtual networks Netkit2, and interact with those instances.

Keywords: Web Services, Netkit2, REST, virtual networks.

Sumário

Lista de Figuras

Lista de Códigos

Lista de Abreviaturas

AOS Arquitetura Orientada a Serviços

DNS *Domain Name Server*

HTTP *Hypertext Transfer Protocol*

ID Identificação Única

IFSC Instituto Federal de Santa Catarina

JSON *JavaScript Object Notation*

MPLS *Multi Protocol Label Switching*

OSPF *Open Shortest Path First*

REST *Representational State Transfer*

RIP *Routing Information Protocol*

ROA *Resource Oriented Architecture*

SIP *Session Initiation Protocol*

SO Sistema Operacional

SOAP *Simple Object Access Protocol*

UML *User Mode Linux*

URI *Unique Resource Identifier*

URL *Uniform Resource Locator*

VM *Virtual Machine*

VoIP *Voz Over IP*

XML *eXtensible Markup Language*

W3C *World Wide Web Consortium*

WWW *World Web Wide*

1 Introdução

O Instituto Federal de Santa Catarina (IFSC), Câmpus São José, disponibiliza um software de virtualização chamado Netkit2, que tem por objetivo facilitar as simulações e experimentos relacionados à infraestrutura e serviços de redes de computadores. Esse software está na sua segunda versão e possui muitas melhorias em relação ao projeto original. Porém ainda impõe algumas dificuldades ao usuário, como o fato de executar apenas no Sistema Operacional (SO) Linux. Além disso, sua instalação é demorada e utiliza arquivos muito grandes.

Esse trabalho visa a adaptação do software Netkit2, já existente, para que ele possa ser acessado remotamente utilizando para isso o serviço web, ou seja, o usuário não precisa depender de seu SO para executar o Netkit2. Para essa adaptação foi desenvolvido um controlador de instâncias que faz uma ligação entre os usuários conectados e o Netkit2 e uma interface RESTful para administrar as ações fornecidas pelos usuários. Para fins acadêmicos, essa adaptação será nomeada neste trabalho de Web-NetKit.

1.1 Motivação

A motivação desse trabalho foi a necessidade de aperfeiçoar o Netkit2, disponível no IFSC, pra que o mesmo possa ser utilizado independentemente do sistema operacional. Além disso, a criação dessa ferramenta estimularia os alunos na realização dos experimentos em qualquer computador dentro ou fora da instituição, incentivando e facilitando sua utilização.

1.2 Objetivos

O objetivo principal desse trabalho é desenvolver um sistema que possibilite a execução remota de múltiplas redes virtuais do Netkit2 em um servidor, e o acesso a essas redes virtuais através de uma interface do tipo serviço web.

- Objetivos Específicos:

1. Possibilitar que múltiplos usuários executem redes virtuais remotamente e de forma independente;
2. Definir uma infraestrutura para a execução de redes remotas, na qual deve ser executado o serviço a ser desenvolvido;
3. Especificar uma interface do tipo serviço web para acessá-lo.

1.3 Organização do texto

Esta monografia está assim organizada: o Capítulo 1 destina-se à contextualização das ferramentas utilizadas para este estudo, dando uma visão geral sobre a monografia. O capítulo 2 é reservado à exposição das ferramentas utilizadas, mostrando o funcionamento de cada uma delas e como tais foram utilizadas no trabalho, bem como uma explicação detalhada de tais ferramentas. O Capítulo 3 trata da utilização de todas as ferramentas e como elas foram usadas e modificadas para a correta comunicação com o Netkit2, bem como um exemplo de funcionamento após a conclusão. Fechando o trabalho, teço minhas considerações final, listo as referências bibliográficas e incluo os trabalhos futuros.

2 *Fundamentação*

2.1 Netkit2

O Netkit2 (GNOME-NETKIT, 2014) é uma ferramenta de ensino de redes de computadores, com a qual se realizam experimentos com redes virtuais. Ele foi desenvolvido no IFSC, Câmpus São José, tendo se inspirado em seu homólogo criado pela Universidade de Roma (NETKIT, 2011). Para realizar isso, ele utiliza máquinas virtuais implementadas com o *User Mode Linux* (UML) contida no kernel[1] do SO Linux, que serve para virtualizar versões do Linux sem interferir nos processos do sistema hospedeiro. Através dessa facilidade, é possível criar vários computadores virtuais e controlar individualmente seu comportamento.

Utilizando esse *software* é possível criar muitos cenários de rede contendo vários tipos de equipamentos (tais como *routers*, *switchs ethernet* gerenciáveis, dentre outros) e implementar serviços de rede (como por exemplo *Domain Name Server* (DNS), *World Web Wide* (WWW), entre outros). A Figura 2.1 ilustra um cenário criado para representar uma rede empresarial, com duas unidades que se comunicam através de um link *Multi Protocol Label Switching* (MPLS), em uma das unidades existe um PBX IP, usado para intermediar chamadas *Voz Over IP* (VoIP) segundo o modelo *Session Initiation Protocol* (SIP). Esta rede tem acesso à nuvem, que é o acesso a internet. Com o experimento em execução, pode-se analisar os serviços e o comportamento da rede bem como o tráfego de dados e os protocolos envolvidos. .

[1] "*Kernel* é o núcleo do sistema operacional, a parte mais próxima do nível físico" FERREIRA (2003).

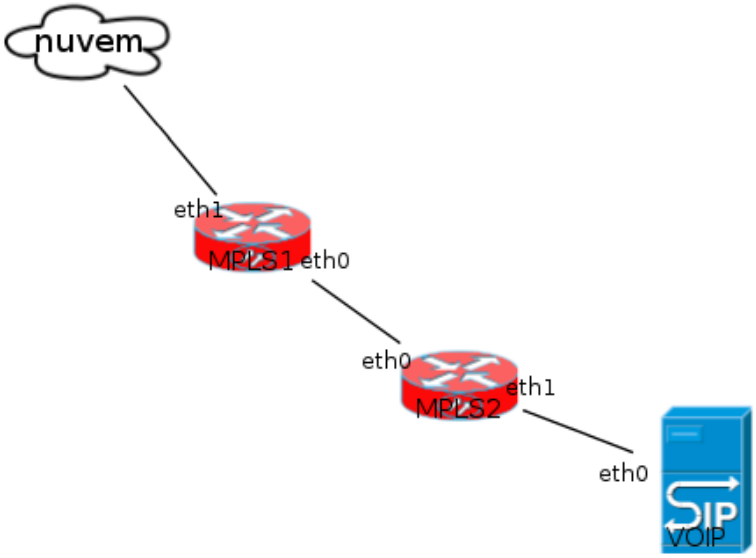

Figura 2.1: Cenário criado utilizando o Netkit2

Enquanto a rede está em execução, cada equipamento virtual possui um console[2] por onde o usuário pode interagir com a máquina podendo alterar configurações ou adicionar outros serviços. É possível utilizar os aplicativos "tcpdump" e "wireshark" para analisar o tráfego da rede ou o "ifconfig" e "route" para se alterar ou visualizar configurações da rede.

2.1.1 Criação de redes virtuais

A criação de uma rede virtual no Netkit2 se faz por meio de um arquivo de configuração. Todos os equipamentos devem ser corretamente listados nesse arquivo que deve ter, obrigatoriamente, a extensão ".conf".

Na versão 2.6, que é a versão corrente do Netkit2 usada neste trabalho, os equipamentos podem ser do tipo:

- ***Gateway***: Um ou mais computadores comuns pré-configurado para se comportar como *gateway* entre duas ou mais redes;
- ***Generic***: Um computador comum, sem nada de especial;
- **MPLS**: Um roteador com suporte ao protocolo MPLS;
- **Pbx**: Um computador com o Asterisk instalado, utilizado para criação de um "pbx ip";

[2]Console é uma interface em modo texto onde o usuário tem acesso a um determinado equipamento.

- **PPPoE**: Um computador do tipo *gateway* capaz de estabelecer conexões PPPoE[3];
- ***Router***: Funciona como um roteador, implementando vários protocolos de roteamento;
- ***Switch***: Um *switch ethernet* para interligar todos os equipamentos nele conectados. Tal equipamento possui suporte a "VLANs" (IEEE 802.1q), "Spanning Tree" (IEEE 802.1d), controle de acesso (IEEE 802.1x) e agregação de enlace (IEEE 802.3ad).

O Código 2.1 mostra a implementação do cenário da Figura 2.1 mostrada anteriormente. Primeiramente deve-se declarar quais os equipamentos que fazem parte da rede a ser criada podendo ser de diferentes tipos. Em seguida, esses equipamentos tem suas interfaces de rede interligadas e configuradas na topologia desejada. Por fim, são definidas as rotas estáticas e os *gateway's* padrão.

```
1  # DECLARACAO DOS EQUIPAMENTOS
2  MPLS1[type]=mpls
3  MPLS2[type]=mpls
4  VOIP[type]=pbx
5
6  # LINK'S E IP'S
7  MPLS1[eth0]=router-router:ip=10.0.0.1/30
8  MPLS1[eth1]=uplink:ip=dhcp
9  MPLS2[eth0]=router-router:ip=10.0.0.2/30
10 MPLS2[eth1]=voip-router2:ip=192.168.0.254/24
11 VOIP[eth0]=voip-router2:ip=192.168.0.1/24
12
13 # ROTAS E GATEWAY'S
14 MPLS2[route]=192.168.0.0/24:gateway=10.0.0.2
15 VOIP[default_gateway]=192.168.0.254
```

Código 2.1: Arquivo de configuração da rede no Netkit2.

Todas as máquinas em execução utilizam um SO básico, fornecido pelo próprio Netkit2. O que as diferencia são os *softwares* executados na inicialização de cada sistema, por exemplo: uma máquina virtual iniciada sem nenhuma configuração específica funcionará como um computador normal, porém essa mesma máquina pode se tornar um roteador caso execute algum *software* de roteamento, como o "Quagga"[4], carregando assim uma tabela de roteamento IP.

[3]PPPoE (sigla em inglês para *Point-to-Point Protocol Over Ethernet*) é um protocolo de rede para conexões de usuários em uma rede *Ethernet* para alcançar a internet. Esse protocolo estabelece sessão e realiza autenticação entre o provedor de acesso a internet e o usuário.

[4]Quagga é um software utilizado para roteamento, ele implementa protocolos de roteamento como o *Open Shortest Path First* (OSPF), o *Routing Information Protocol* (RIP), dentre outros.

Ao ser executado, o Netkit2 cria um subdiretório de trabalho, onde ficam salvas todas as informações referentes a um experimento. Dentro desse subdiretório, há outro subdiretório para cada máquina virtual, usado para preservar arquivos da respectiva máquina virtual, caso desejado. Existem também alguns arquivos necessários para a execução das máquinas virtuais.

- **Nome_da_maquina.disk**: Esse arquivo contém a imagem do disco virtual em formato "COW"[5]. Apesar de seu tamanho parecer ser de 4GB, na realidade ele é bem menor. Ele contém de fato apenas os blocos do arquivo de imagem original que foram modificados dentro da máquina virtual correspondente;
- **Nome_da_maquina-auto.sh**: Arquivo que o Netkit2 cria para inserir comandos durante a inicialização do sistema;
- **Shared**: Apesar de não ser um arquivo necessário para a execução das máquinas virtuais, essa pasta é utilizada para se compartilhar arquivos da máquina hospedeira entre as diversas máquinas virtuais.

A criação e manutenção do subdiretório de trabalho e seu conteúdo é feita pelo Netkit2, não sendo recomendável manipulá-los diretamente. A Figura 2.2 mostra a estrutura da pasta criada pelo Netkit2.

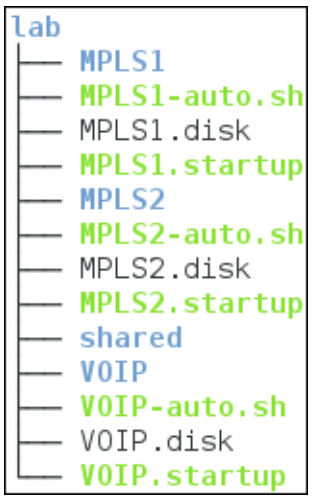

Figura 2.2: Estrutura dos diretórios

O principal motivo do Netkit2 permitir a emulação de diversos equipamentos sobre o mesmo hospedeiro é a reduzida quantidade de recursos utilizada em cada virtualização. Cada máquina virtual consome tipicamente 32MB de memória RAM e ocupa até 80% do processador

[5]COW (*Copy On Write*) é uma técnica utilizada para se aumentar a performance nas operações de escrita de dados entre áreas físicas.

(pico obtido durante a inicialização das máquinas simultaneamente). Em um computador com um núcleo e 1GB de memória RAM, podem-se rodar em torno de 32 dessas máquinas virtuais, com tempo de resposta aceitável.

2.1.2 O Gnome-Netkit

O Gnome-Netkit é uma aplicação gráfica para facilitar o uso do Netkit2 pelos usuários. Por meio da sua interface é possível carregar os arquivos de configuração e visualizar o cenário implementado pelo experimento antes mesmo de inicializá-lo. Outra característica dessa aplicação é que ele disponibiliza os terminais de controle das máquinas virtuais sobre uma mesma interface, facilitando o acesso a consoles das máquinas virtuais durante o experimento, como pode-se ver na Figura 2.3.

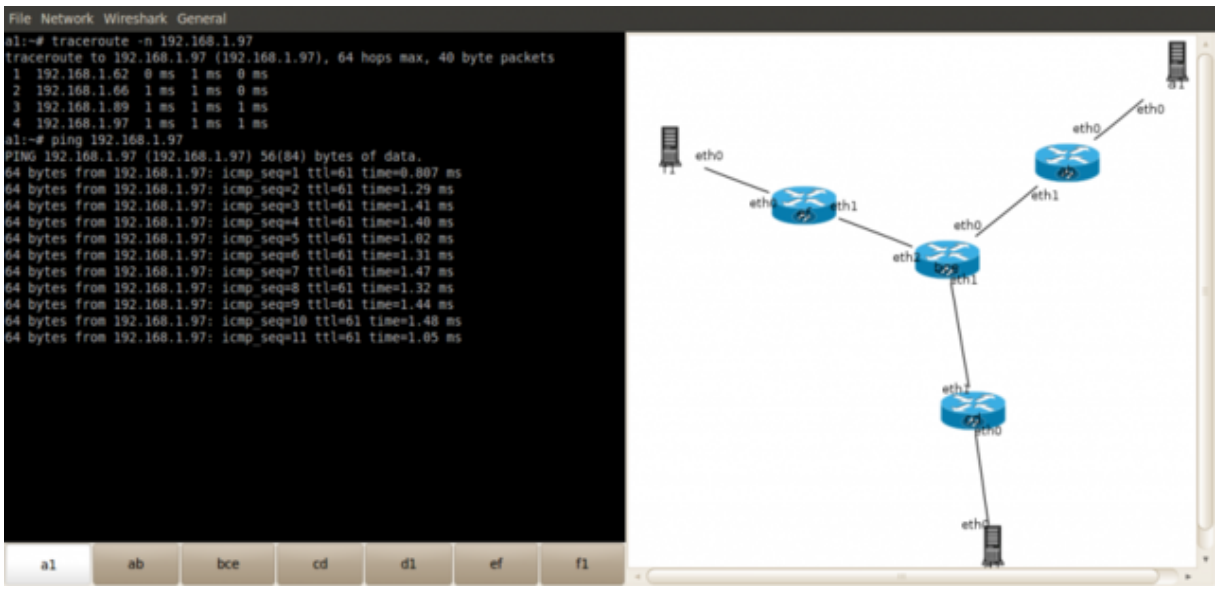

Figura 2.3: Gnome-Netkit em execução

Com essa aplicação pode-se também iniciar, parar e reiniciar, de forma automática, todas as máquinas do experimento. Porém, assim como o próprio Netkit2, o Gnome-Netkit só é executado na máquina local, onde o serviço está disponibilizado, e exige Linux como sistema hospedeiro.

2.1.3 API do Netkit2

De acordo com (GNOME-NETKIT, 2014) "O Netkit2 possui uma API[6] para iniciar e parar experimentos, entre outras funções. Para usá-la é recomendável usar "NetkitParser", uma classe que interpreta arquivos de configuração e a partir deles cria objetos "Network". A classe "Network" representa uma rede virtual do Netkit2, e com ela se pode inciar, parar, ou mudar preferências."

Essa API é composta por cinco classes principais, como ilustrado na Figura 2.4:

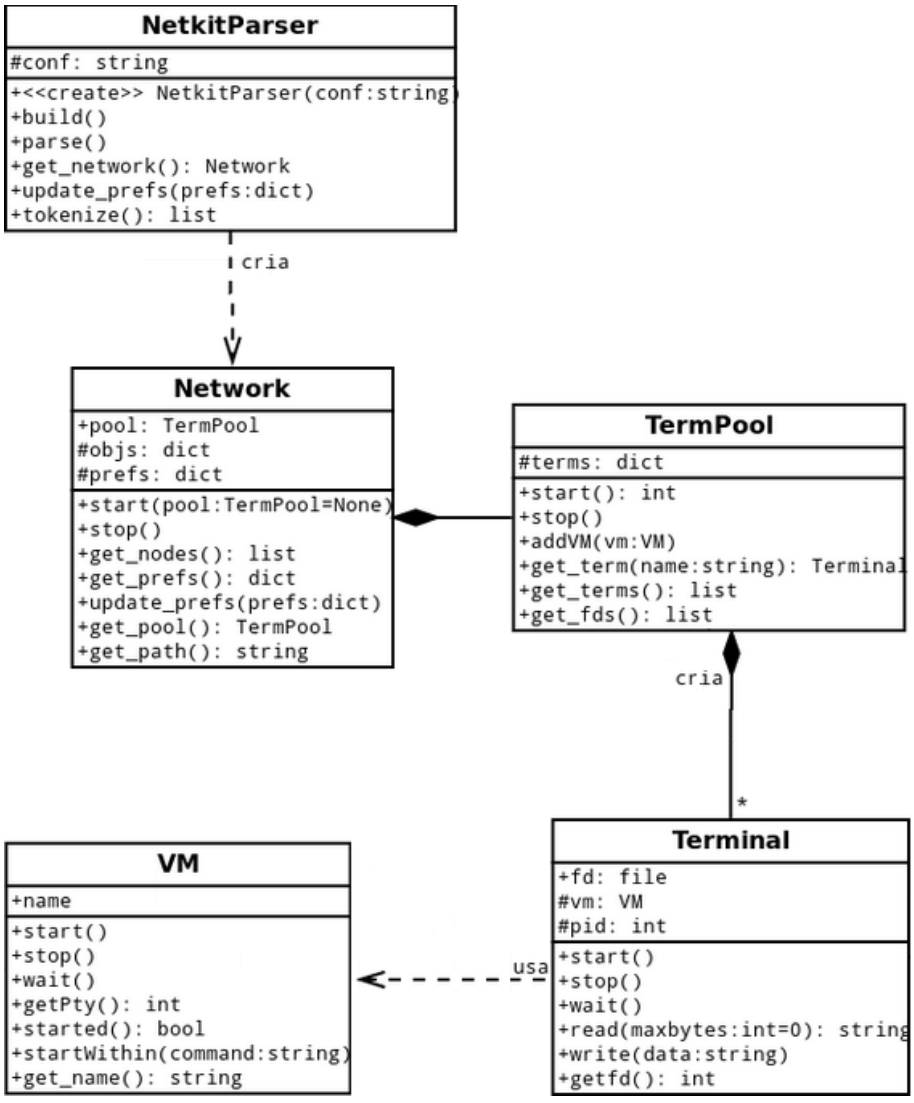

Figura 2.4: Diagrama de classes do Netkit2

Abaixo, uma breve descrição das classes envolvidas sendo posteriormente explicada cada uma delas.

[6] API é um conjunto de funcionalidades estabelecidos por uma aplicação para que seja possível sua utilização por outras aplicações.

- **Classe NetkitParser**: É utilizada para interpretar o arquivo ".conf", ou seja, baseado no arquivo de configuração, essa classe cria todos os componentes de rede e suas ligações, mas não faz nenhuma configuração e nem inicia o laboratório. Ela irá retornar um objeto da classe "network";
- **Classe Network**: Essa classe representa uma rede a ser carregada a partir de um arquivo de configuração, que comanda um "TermPool";
- **Classe TermPool**: Ela é responsável por criar um console para cada máquina virtual da rede, enviando o comando de iniciar ou parar as máquinas virtuais;
- **Classe Terminal**: Define como devem se apresentar objetos a serem usados como terminais para as VM do Netkit. Objetos que implementam essa classe não devem ser instanciados diretamente, pois são criados por "Termpool".
- **Classe VM**: Essa classe é responsável por gerenciar as máquinas virtuais UML, enviando comandos para iniciar ou parar uma rede e selecionar uma máquina virtual ativa.

Classe Netkit Parser

Um objeto da classe NetkitParser implementa um *parser* (analisador sintático). Com ele se pode interpretar um arquivo de configuração de uma rede, e criar a rede ali descrita. Se houver erros de sintaxe, ou mesmo semânticos, o *parser* dispara uma exceção avisando sobre o erro e em que parte do arquivo ele foi encontrado. Se o arquivo de configuração não tiver erros, o *parser* cria um objeto "Network" para representar a rede. O *parser* pode também ser usado para analisar um arquivo de configuração, sem criar um objeto "Network", desta forma apenas verificando a correção da configuração.

Classe Network

A classe Network representa redes do Netkit2. Um objeto dessa classe contém todas as especificações para a criação e configuração de uma rede. Através dele se pode iniciar e para a execução de uma rede. No entanto, objetos da classe Network devem ser criados por meio de um objeto "NetkitParser".

Essa classe pode realizar:

- O início de uma rede;
- O término de uma rede;

- Mostrar uma lista contendo todas as redes disponíveis;
- Mostrar o diretório de trabalho;
- Mostrar e modificar preferências da rede virtual.

Classe VM

A classe VM representa máquinas virtuais UML. Um objeto dessa classe pode iniciar, parar e acessar a console de uma *Virtual Machine* (VM). Objetos dessa classe são criados automaticamente ao se inciar uma rede. Essa classe pode realizar as seguintes ações:

- Executar uma VM UML correspondente a um nodo de uma rede virtual. A máquina virtual é executada em um processo filho, e sua console é conectada ao lado escravo de um pseudo-terminal;
- Terminar a execução de uma VM.

Interface TermPool

A interface TermPool define um concentrador de terminais para as máquinas virtuais. Sua função é criar um terminal para cada VM a ser executada. Esses terminais devem ser objetos que implementam a interface "Terminal", os quais desempenham esse papel para as VM. Quer dizer, cada objeto "Terminal" deve se associar à console de uma VM e agir como um terminal de fato.

Interface Terminal

A interface Terminal define como devem se apresentar objetos a serem usados como terminais para as VMs do Netkit2. Um Terminal tem o papel de interfacear o console de uma VM, dependendo da forma com que se deseje acessar a rede em execução. Objetos que implementam essa interface não devem ser instanciados diretamente, pois são criados por "TermPool".

2.2 Serviços WEB

Os serviços WEB possuem várias definições, mas todos seguem uma Arquitetura Orientada a Serviços (AOS). De acordo com o (MELLO et al., 2006) apud (W3C, 2004) esse tipo de serviço "Trata-se de uma aplicação identificada através de uma *Unique Resource Identifier*

(URI), que possui interfaces bem definidas e descritas em *eXtensible Markup Language* (XML). As interações com outras aplicações se faz através de trocas de mensagens XML utilizando protocolos padrões da Internet.". Já TIDWELL, SNELL e KULCHENKO (2001), define serviço WEB como "uma funcionalidade de uma aplicação acessível por meio de uma interface de rede construída utilizando tecnologias padrões de Internet". Entre essas definições há o protocolo HTTP utilizado por ambas para prover acesso aos serviços e ao transporte de conteúdos.

Existem diversas formas de implementar serviços WEBs. Muitos já estão amplamente difundidos, como o *Simple Object Access Protocol* (SOAP) que é um protocolo padronizado e reconhecidos pelo *World Wide Web Consortium* (W3C) é importante destacar que ele está diretamente ligado ao XML e faz uso dessa linguagem para a troca de mensagens, porém tal protocolo tem desvantagens em termos de desempenho e escabilidade conforme cita (FILHO, 2009). Além desses, existe o REST, que é um estilo de arquitetura de aplicação e foi desenvolvido por (FIELDING, 2000), em sua tese de PhD.

2.2.1 *Representational State Transfer* (REST)

De acordo com (WEBBER; PARASTATIDIS; ROBINSON, 2010) "Como parte de seu trabalho de doutorado, (FIELDING, 2000) generalizou princípios arquitetônicos da WEB e os apresentou como um quadro de restrições, ou um estilo arquitetônico. Através desse estilo, Fielding descreveu como sistemas de informação distribuídos devem ser construídos e operados. Ele descreveu a interação entre os recursos, bem como o papel de identificadores únicos em tais sistemas. Também escreveu sobre o uso de um conjunto limitado de operações com a semântica uniforme para construir uma infra-estrutura onipresente que pode suportar qualquer tipo de aplicação." Assim ele pôde desenvolver uma arquitetura, que foi nomeada de REST, que descreve como melhor usar os recursos de uma aplicação WEB.

NUNES e DAVID (2005) menciona que o estilo REST utiliza um conjunto de interfaces genéricas para promover interações sem estado (*stateless*) através da transferência de representações de recursos, em vez de operar diretamente sobre esses recursos. O conceito de recurso é a principal abstração deste estilo. Conforme (MORO; DORNELES; REBONATTO, 2011) apud (FIELDING, 2000) "para que os princípios deste estilo sejam respeitados, um conjunto de restrições deve ser seguido:

- **Cliente-Servidor**: esta característica é mais comumente encontrada em aplicações Web. Um servidor, com um conjunto de serviços disponíveis, aguarda requisições a estes serviços. Um cliente, que deseja que um serviço disponível no servidor seja executado, envia uma

requisição para o servidor. O servidor então pode tanto rejeitar como executar o serviço solicitado, e retornar uma resposta ao cliente;

- ***Stateless* (Sem estado)**: outra restrição imposta pelo estilo REST diz respeito à interação entre cliente e servidor. A comunicação deve ser feita sem o armazenamento de qualquer tipo de estado no servidor, ou seja, cada requisição do cliente para o servidor deve conter todas as informações necessárias para que ela seja entendida. Portanto, estados de sessão, quando necessários, devem ser totalmente mantidos no cliente;

- **Cache**: utiliza-se a cache para melhorar o desempenho das interações. Ela exige que os dados de uma resposta, vindos de uma requisição ao servidor, sejam marcados como *cacheable* ou *noncacheable* (passíveis ou não de utilização da cache). Se uma resposta é setada como *cacheable*, então ela será reutilizada como resposta para as futuras requisições equivalentes;

- **Interface Uniforme**: a característica central que diferencia o estilo arquitetural REST de outros estilos baseados em rede é sua ênfase em uma interface uniforme entre os componentes (cliente, servidor). Com o objetivo de obter uma interface uniforme, REST define quatro requisitos de interface: identificação de recursos, manipulação de recursos através de representações, mensagens auto-descritivas e hipermídia como mecanismo de estado da aplicação."

Todas as aplicações que utilizam o princípio REST são chamadas de RESTful e de acordo com (RICHARDSON; RUBY, 2007) essa é uma arquitetura orientada a recursos *Resource Oriented Architecture* (ROA).

(NGOLO, 2009) mostra um bom exemplo sobre ROA, ele diz "Os servidores possuem recursos e os clientes consomem recursos, um recurso pode ser qualquer informação a qual se possa atribuir um nome. Pode ser um documento, uma imagem, um vídeo. O *Uniform Resource Locator* (URL) é usado como identificador do recurso, ou seja, o URL é o nome do recurso. O protocolo HTTP é utilizado pelo REST para acessar recursos de todos os tipos.", ele ainda afirma que "o REST na sua essência utiliza o protocolo HTTP para alterar o estado do recurso". Para cada operação há uma ação, a Tabela 2.1 ilustra a utilização da interface REST juntamente com as quatro operações mais comuns e suas respectivas ações.

Método HTTP	Ação
GET	Busca recursos
POST	Cria recursos
PUT	Modifica recursos
DELETE	Remove recursos

Tabela 2.1: Operações do REST

É possível perceber que a operação PUT é utilizada para se atualizar ou editar um recurso já existente, já a operação DELETE é utilizada para finalizar um recurso. Através da operação GET é possível mostrar ou listar recursos disponíveis. Por fim, a operação POST é utilizada para criar um novo recurso.

2.2.2 *JavaScript Object Notation* (JSON)

De acordo com (RFC7159, 2014) JSON "é um conjunto de regras de formatação, derivado da linguagem de programação ECMAScript, para estruturar o armazenamento e transmissão de informações no formato texto, independente da linguagem utilizada.

(CHANCHÍ et al., 2011) diz que o JSON é constituído de 2 estruturas básicas. O primeiro é uma coleção de pares "nome-valor" ou estruturas, que, em várias linguagens são conhecidos como dicionários. O segundo é uma lista ordenada de valores e que, na maioria das linguagens, é representado como matrizes/vetores.

De acordo com (FONSECA; SIMOES, 2007) "O JSON pode representar quatro tipos primários (strings, números, booleanos e nulos) e dois tipos estruturados (objetos e vetores). Um objeto é uma coleção não ordenada de pares nome/valor, onde o nome é uma string e o valor pode ser uma string, número, booleano, nulo, objeto ou vetor. Um vetor é uma sequência ordenada de zero ou mais valores. Assim o JSON foi desenhado com o objetivo de ser simples, portável, textual, e um subconjunto do JavaScript.".

A sintaxe do JSON é organizada da seguinte maneira:

- O formato dos dados é definido sempre em pares e separado por virgulas;
- As chaves "{}" contém objetos;
- Os colchetes "[]" expressam matrizes/vetores.

O seguinte exemplo mostra um dicionário JSON, nomeado de “pessoa”, com dois objetos cada. O primeiro objeto é representado pelo nome “alcunha” e seu conteúdo é “Luan”. O segundo objeto desse dicionário é nomeado de “idade” e seu valor é “10”.

Variável pessoa = {“alcunha” : “Luan” , “idade” : “10”}

3 *Sistema para execução remota do Netkit2*

Com o intuito de permitir que o Netkit2 seja acessado remotamente foi criada uma infraestrutura capaz de centralizar sua execução. Com isso, usuários podem executar seus experimentos sem a necessidade de instalar o *software*. Além disso, abre-se a possibilidade dos usuários realizem seus experimentos a partir de dispositivos que usem qualquer sistema operacional.

Um elemento chave dessa infraestrutura é o controlador de instâncias do Netkit2. Esse controlador é uma aplicação executada em um servidor, o qual é responsável por gerenciar as conexões dos usuários estando diretamente vinculado ao *software*. Os clientes interagem com o controlador por meio de uma interface do tipo serviço web. Através dessa interface pode-se iniciar, terminar, selecionar e obter informações de redes virtuais em execução. Além disso, um cliente pode interagir com o terminal (console) de uma máquina virtual de uma rede virtual. Essa interação com os terminais é intermediada pelo controlador, que multiplexa os acessos entre clientes e terminais de máquinas virtuais.

O controlador desenvolvido é capaz de interpretar uma requisição HTTP e convertê-la em comandos para o Netkit2. Assim, pode-se executá-lo remotamente, bastando apenas que essa requisição seja compatível com o controlador, permitindo a mais de um usuário se conectar ao mesmo tempo. No exemplo da Figura 3.1, os três clientes enviam o sinal para iniciar o experimento, o Netkit2 inicia todas as máquinas virtuais dentro do servidor e abre uma console para cada máquina, apenas uma console é enviada ao cliente, pois o servidor fica limitado em trabalhar com apenas uma máquina virtual por vez podendo selecionar qual console deve ficar ativo.

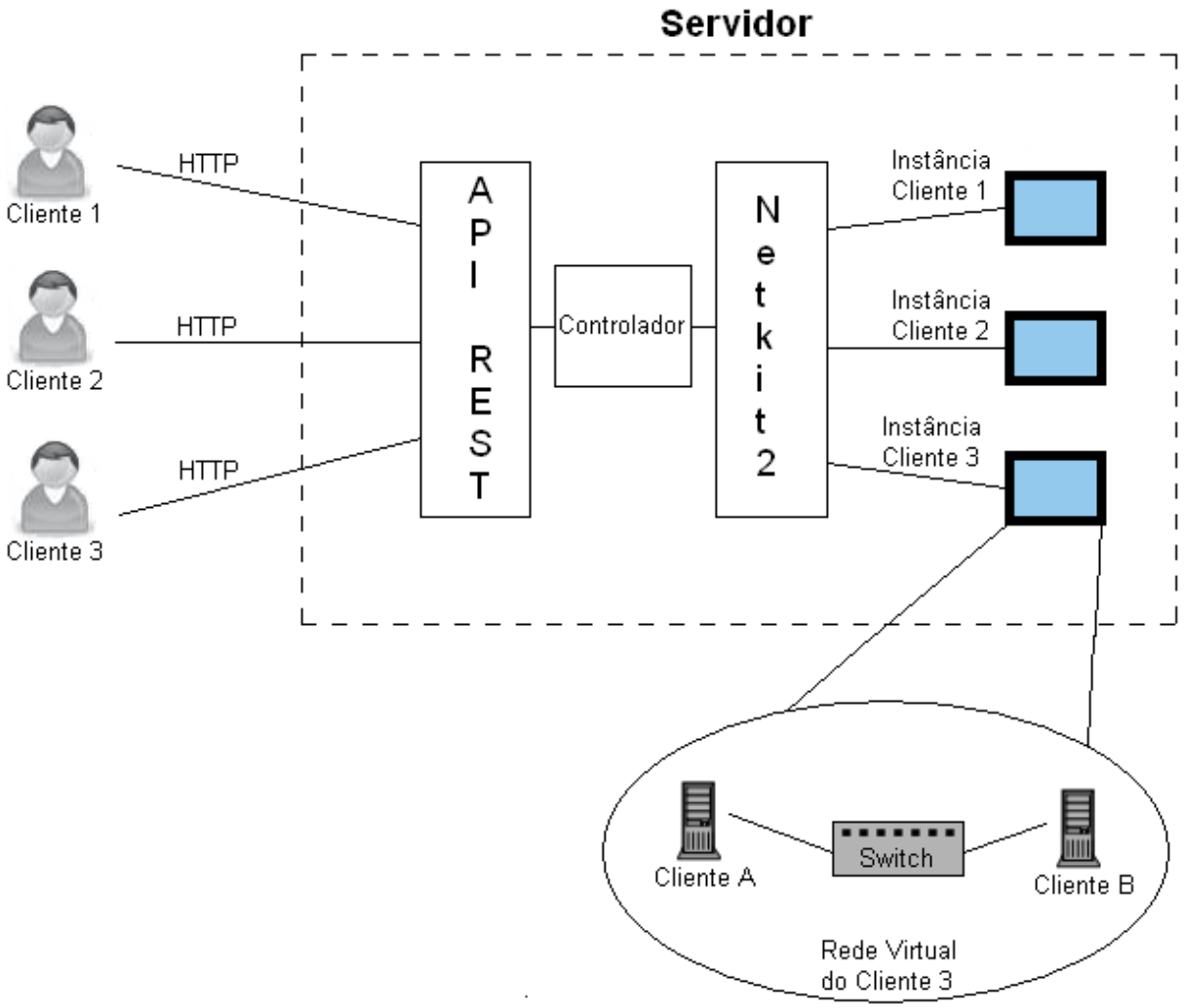

Figura 3.1: Panorama do sistema.

3.1 Arquitetura do Sistema

A arquitetura do sistema é composta por três blocos funcionais, conforme mostrado na Figura 3.1:

1. **Interface RESTful**: Responsável por receber e interpretar solicitações através de serviços web e encaminhar essas solicitações para o Controlador;

2. **Controlador de Instâncias**: Ele "converte" as solicitações enviadas pela interface RESTful em comandos reconhecidos pelo Netkit2;

3. **Netkit2**: É o software utilizado para criar as redes virtuais, conforme citado no Capítulo 2.

A estrutura do software que implementa essa arquitetura está resumidamente descrita pelo diagrama de classes apresentado na figura 3.2. As classes relacionadas a cada bloco funcional foram destacadas, e estão descritas com mais detalhes nas seções 3.1.1 a 3.1.3.

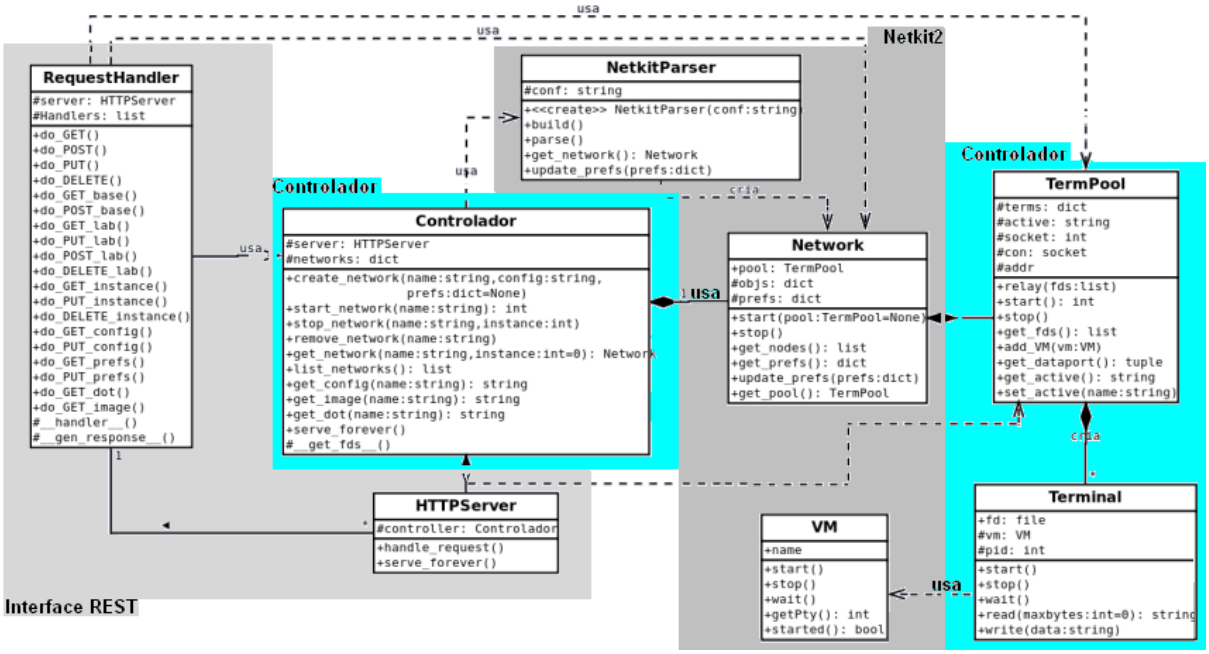

Figura 3.2: Diagrama de classe completo do sistema.

3.1.1 Controlador de Instâncias

O controlador de instâncias interpreta requisições dos clientes para iniciar, terminar e acessar instâncias do Netkit2, entre outras operações. A principal classe que implementa esse bloco funcional está representado na Figura 3.3. As operações dessa classe são invocadas por meio da interface RESTful, descrita na seção 3.1.2. Além disso, o controlador é responsável por fazer a multiplexação entre os diversos cliente e suas respectivas instâncias.

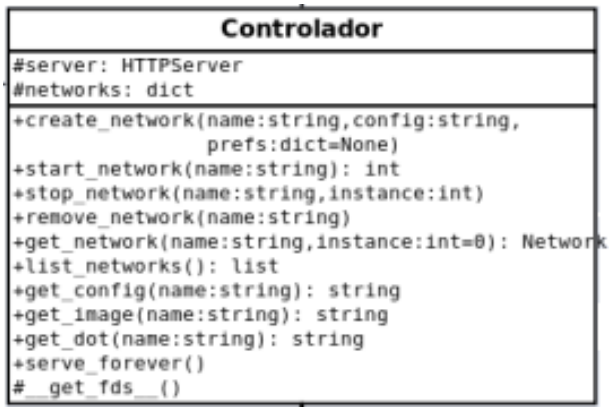

Figura 3.3: Componentes do controlador.

O controlador identifica cada instância com uma Identificação Única (ID), que é gerado aleatoriamente, e o relaciona em um dicionário próprio juntamente com o endereço IP do cliente. Sabendo isso, o controlador consegue identificar a quem pertence cada uma das instâncias em execução.

Os serviços acessados pelo cliente e fornecido pelo controlador são:

- Iniciar e parar uma rede;
- Modificar uma rede;
- Selecionar uma máquina ativa;
- Obter informações da rede.

Esses serviços estão detalhados na seção 3.1.2.

3.1.2 Interface RESTful

A interface de acesso ao controlador de instâncias segue um modelo RESTful. Ela disponibiliza várias operações, conforme apresentado na Tabela 3.1. Neste trabalho foram implementadas somente quatro operações, as quais são suficientes para demonstrar a execução remota de experimentos.

Recurso	**Método**	**Resultado**		**Descrição da Operação**
		Status	**Descrição**	
/Netkit	GET	200	Sucesso, e retorna redes existentes no corpo da resposta	Obtém a relação de redes existentes no catálogo do controlador.
		400	Erro de sintaxe	
/Netkit/name	POST	201	Sucesso: URI da rede criada no cabeçalho Location	Cria uma nova rede identificada por *"name"*.
		400	Erro de sintaxe na configuração, e retorna o local do erro no corpo da resposta	
		409	Erro: rede já existe	
/Netkit/ID	POST	201	Sucesso, com URI da instância no cabeçalho Location	Inicia a execução de uma rede
		400	Erro genérico	
		404	Rede não existe	
/Netkit/ID	DELETE	200	Sucesso	Termina a execução de uma rede
		400	Erro genérico	
		404	Rede não existe	
		423	Há Instâncias em execução	

Tabela 3.1: Tabela de operações oferecida pela interface REST

3.1.3 Implementação das Operações

Na versão atual do controlador, quatro operações foram implementadas. Essas operações são suficientes para realizar e acessar experimentos remotos, sendo elas:

1. **Início da rede**: Iniciar uma instância que representa uma rede virtual;
2. **Parar a rede**: Finaliza a execução de uma instância;
3. **Trocando Máquina ativa**: Altera a máquina virtual ativa cujo console pode ser acessado pelo cliente;
4. **Listar redes**: Obtêm informações sobre uma rede.

Início da rede

Sabendo qual rede se deseja iniciar, o cliente deve enviar uma requisição (como mostra na figura 3.4, passo 1) para a interface RESTful que encaminhara a solicitação de início ao controlador (figura 3.4, passo 2). Ele adicionará uma identificação para esta Instância e solicitará

ao Netkit2 a criação da rede (figura 3.4, passo 3), criando um objeto Network que representa a rede a ser executada. Utilizando ela o Netkit2 relaciona todas as máquinas virtuais ao TermPool (figura 3.4, passo 4) possibilitando a interação entre o console da vm ativa e o cliente. A partir desse ponto, o cliente irá receber dados das máquinas virtuais iniciadas e poderá enviar comando para as mesmas (figura 3.4, passo 5). Por fim, o cliente receberá uma confirmação indicando que o laboratório foi iniciado corretamente (figura 3.4, passo 6)

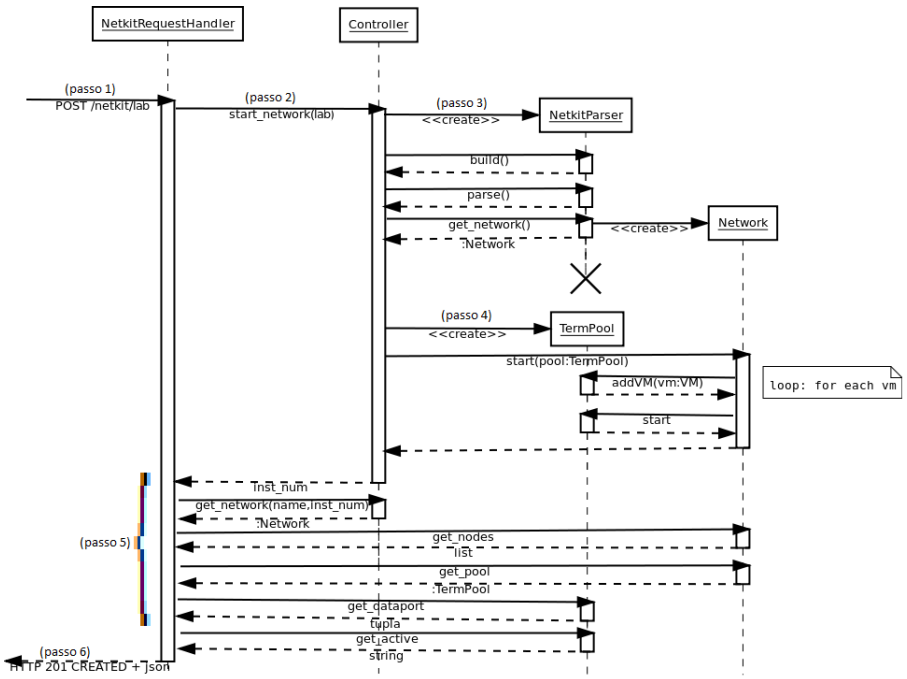

Figura 3.4: Inicio da máquina

Parar a rede

Para parar a rede, usa-se a requisição mostrada, como exemplo, na figura (3.5, passo 1) para o controlador, este enviará uma requisição para o RESTful para parar todos a Instância em execução(3.5, passo 2). O controlador encaminhará o comando para o TermPool através do objeto Network (3.5, passo 3) parando todas as máquinas em execução da rede. Como último passo, o controlador envia ao cliente a confirmação do término da rede (3.5, passo 4).

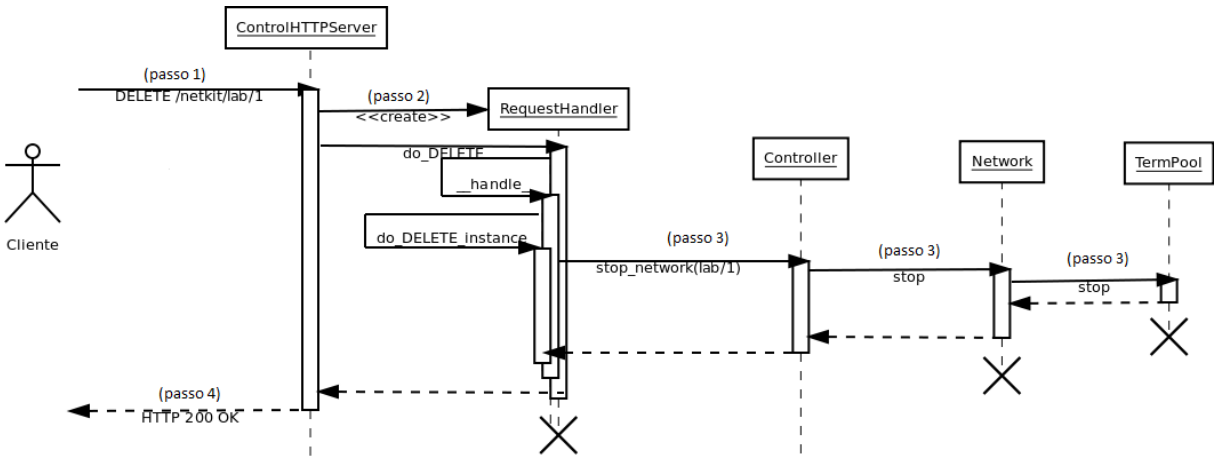

Figura 3.5: Término da máquina

Trocando VM ativa

Por último, é possível chavear entre todas as máquinas que estão em execução. Para isso usa-se como exemplo a requisição mostrada na figura (3.6, passo 1). O RESTful enviará uma solicitação ao controlador solicitando o objeto Network da rede (figura 3.6, passo 2). O próprio RESTful enviará ao TermPool qual a vm que deverá ser a ativa (figura 3.6, passo 3). Para finalizar a requisição é enviado uma resposta de confirmação ao cliente (figura 3.6, passo 4).

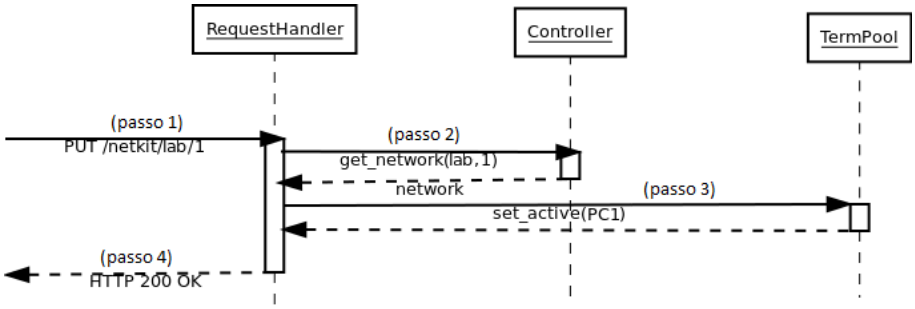

Figura 3.6: Mudar máquina ativa

3.2 Experimentos

O cliente é um software qualquer capaz de enviar comandos ao controlador através da interface RESTful, acessando o console remoto da máquina ativa da rede virtual. Esse software pode ser implementado em modo texto, gráfico ou mesmo web, bastando ser capaz de interagir com a interface RESTful do controlador. Um exemplo de comunicação entre o cliente e o controlador está mostrado na figura 3.7.

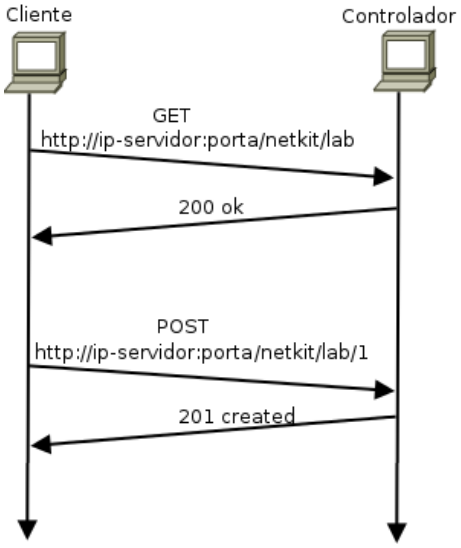

Figura 3.7: Iniciando uma Instância

3.2.1 Usando o Telnet

É possível utilizar o utilitário telnet, disponível em qualquer SO, para executar o experimento remotamente. Para solicitar uma operação ao controlador, usa-se o programa telnet para enviar uma requisição para a interface RESTful. O resultado dessa requisição deve ser interpretado pelo usuário, de forma a poder enviar novas requisições ao controlador. Desta forma, usando o telnet, pode-se testar manualmente o acesso as sistema de execução remota de instâncias do Netkit2.

No exemplo de execução contido na Figura 3.8 o cliente solicita ao controlador o inicio da rede “n1” (através da interface RESTful e usando a requisição “POST /netkit/n1 HTTP/1.0”). O resultado da requisição é o endereço IP e a porta em que deve-se conectarpara acesso aos terminais das máquinas virtuais. Dentre as informações retornadas, destacam-se:

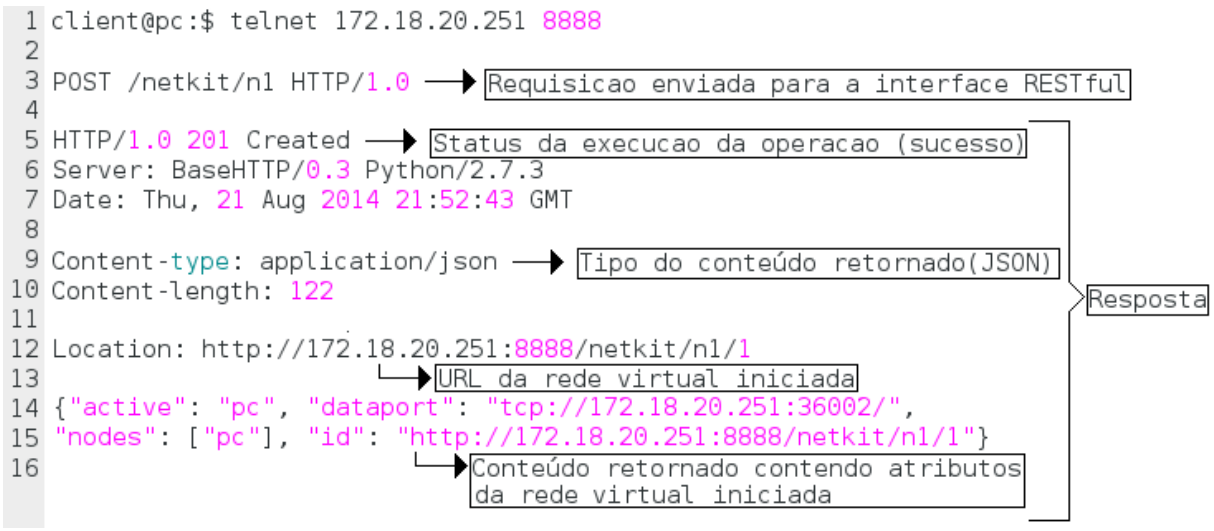

Figura 3.8: Iniciando a execução do experimento.

Utilizando novamente o telnet para se conectar no IP e porta indicados, o cliente tem acesso ao terminal da máquina virtual ativa, conforme mostra a Figura 3.9.

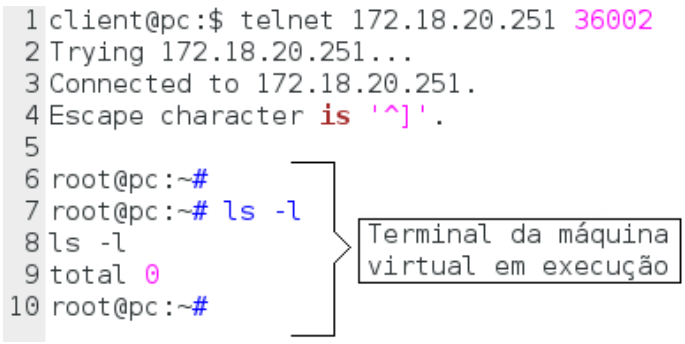

Figura 3.9: Visualizando a máquina ativa.

Por fim, o cliente pode terminar o experimento, como mostra a Figura 3.10, enviando para a interface RESTful uma requisição “DELETE /netkit/n1/1 HTTP/1.0”.

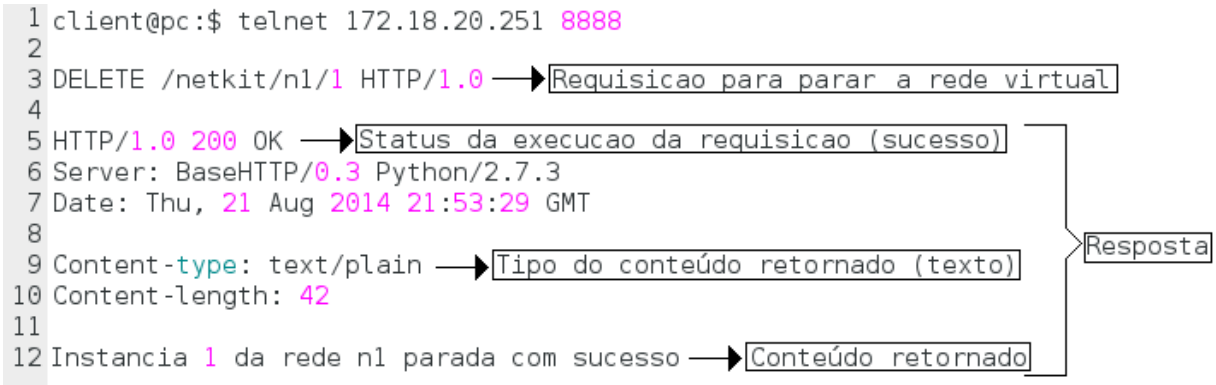

Figura 3.10: Finalizando a execução do experimento.

4 Conclusões

Este trabalho propôs um sistema de execução remota para o Netkit2, com o objetivo de criar um serviço para realização de experimentos com redes virtuais. Para isso, foi nescessário desenvolver um controlador de instâncias que é responsável pelo gerenciamento de conexões e pelos protocolos envolvidos e uma interface RESTful, desenvolvida para interpretar as requisições dos usuário.

Com o controlador desenvolvido é possível que multiplos usuários acessem o Netkit2 simultaneamente usando um serviço web. Através desse serviço cada usuário pode iniciar uma instância que representa uma rede virtual. O serviço web utilizado basea-se no estilo arquitetônico chamado REST, o qual se aapresenta como uma interface RESTful, que recebe as requisições dos clientes. A infraestrutura desenvolvida utiliza o controlador e a interface RESTful para interligar os clientes remotos ao Netkit2.

Desse modo é possível que vários usuários criem redes virtuais remotamente sem necessidade de instalar o Gnome-Netkit em computadores locais. Isso torna possível manter os experimento salvos em um servidor remoto para posterior execução.

4.1 Trabalhos Futuros

Muitas melhorias ainda podem ser realizadas no controlador de instâncias, dentre as principais pode-se citar:

- O desenvolvimento de uma interface web para que seja possível aos usuários utilizarem navegadores web para acesso ao controlador e assim utilizarem outros sistemas operacionais;
- Aplicação do controlador em uma infraestrutura em nuvem do tipo SaaS (software como serviço) para distribuir a carga de processamento do servidor permitindo um melhor gerenciamentos dos recursos locais;

- Implementação de controle de acesso para os clientes do sistema, melhorando a segurança e permitindo que usuário possam armazenar seus experimentos no servidor remoto;
- Impor restrições quanto ao uso de recursos pelos experimentos impedindo que um único usuário utilize todo os recursos do servidor.

Referências Bibliográficas

CHANCHÍ, G. E. et al. Esquema de servicios para televisión digital interactiva, basados en el protocolo rest-json. p. 3, 2011. Acessado em 10-10-2014. Disponível em: <http://seer.ufrgs.br/cadernosdeinformatica/article/view/v6n1p233-240>.

FERREIRA, R. E. *Linux Guia do Administrador do Sistema*. Brasil: Novatec, 2003. 27 p.

FIELDING, R. *Representational State Transfer (REST)*. Irvine, 2000. Acessado em 07-04-2014. Disponível em: <http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm>.

FILHO, O. F. F. *Serviços Semânticos: Uma Abordagem RESTful*. Brasil: [s.n.], 2009. 50 p.

FONSECA, R.; SIMOES, A. *Alternativas ao XML: YAML e JSON*. Portugal, 2007. 8 p. Acessado em 10-10-2014. Disponível em: <http://hdl.handle.net/1822/6230>.

GNOME-NETKIT. *Gnome-Netkit*. Brasil, 2014. Acessado em 04-04-2014. Disponível em: <http://wiki.sj.ifsc.edu.br/wiki/index.php/Netkit2>.

MELLO, E. R. d. et al. Segurança em serviço web. p. 4, 2006. Acessado em 10-10-2014.

MORO, T. D.; DORNELES, C. F.; REBONATTO, M. T. Web services ws versus web services rest. p. 40–4–1, 2011. Acessado em 02-02-2015. Disponível em: <http://www.seer.ufrgs.br/reic/article/viewFile/22140/12928>.

NETKIT. *Netkit*. Itália, 2011. Acessado em 04-10-2013. Disponível em: <http://wiki.netkit.org/index.php>.

NGOLO, M. A. F. *Arquitectura Orientada a Serviços REST para Laboratórios Remotos*. Lisboa, 2009. 37–42 p. Acessado em 10-10-2014. Disponível em: <http://run.unl.pt/bitstream/10362/2079/1/Ngolo_2009.pdf>.

NUNES, S.; DAVID, G. Uma arquitectura web para serviços web. p. 5, 2005. Acessado em 10-10-2014. Disponível em: <http://hdl.handle.net/10216/281>.

RFC7159. *The JavaScript Object Notation (JSON) Data Interchange Format*. Itália, 2014. Disponível em: <http://tools.ietf.org/html/rfc7159>.

RICHARDSON, L.; RUBY, S. *RESTful Web Services*. EUA, 2007. 13–42 p. Acessado em 10-10-2014.

TIDWELL, D.; SNELL, J.; KULCHENKO, P. *Programming Web Service With SOAP*. EUA: O'Reilly Media, Inc., 2001. 1–4 p.

W3C. *Web Services Architecture*. EUA, 2004. Acessado em 07-04-2014. Disponível em: <http://www.w3.org/TR/ws-arch/#introduction>.

WEBBER, J.; PARASTATIDIS, S.; ROBINSON, I. *REST In Practice*. EUA: O'Reilly Media, Inc., 2010. 16–20 p.