Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-graduação em Ciência da Computação

Arlison Wady Sousa Martins

Um Ambiente Virtual Baseado em Nuvem para Simulação de Redes de Computadores com o NS-3

São Luís
2016

Arlison Wady Sousa Martins

Um Ambiente Virtual Baseado em Nuvem para Simulação de Redes de Computadores com o NS-3

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de MESTRE em Ciência da Computação.

Orientador: Prof. Rafael Fernandes Lopes

Doutor em Engenharia Elétrica - UFMA

Coorientador: Prof. Mário Antônio Meireles Teixeira

Doutor em Ciência da Computação e Matemática Computacional - UFMA

São Luís

2016

Martins, Arlison Wady Sousa

Um Ambiente Virtual Baseado em Nuvem para Simulação de Redes de Computadores com o NS-3 / Arlison Wady Sousa Martins - 2016

xx.p

1.Computação em Nuvem 2.Simuladores . I.Título.

CDU xxxx

Arlison Wady Sousa Martins

Um Ambiente Virtual Baseado em Nuvem para Simulação de Redes de Computadores com o NS-3

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de MESTRE em Ciência da Computação.

Aprovado em ____ de _____ de _____

BANCA EXAMINADORA

Prof. Rafael Fernandes Lopes

Doutor em Engenharia Elétrica - UFMA

Prof. Mário Antônio Meireles Teixeira

Doutor em Ciência da Computação - UFMA

Prof. André Castelo Branco Soares

Doutor em Ciência da Computação - UFPI

Prof.Samyr Béliche Vale

Doutor em Ciência da Computação - UFMA

Agradecimentos

Agradeço a todos aqueles que direta ou indiretamemente contribuíram para a concretização desta dissertação.

A Deus, por ter me dado condições de lutar e alcançar os objetivos pretendidos.

Aos meus pais, Sebastião Ferreira Martins e Maria José Sousa Martins, pelo amor, educação e apoio a mim dedicado.

À minha esposa, Dalcione Martins Lima Martins, aos meus filhos Rodrigo e Breno, por compreenderem os meus momentos de ausência.

Aos meus irmãos Martsjambes Sousa Martins e Adson José Sousa Martins e em especial ao primogenito Ruy Marth Sousa Martins (in memoriam), que sempre me deram força e incentivo.

Ao meus orientadores, os professores Rafael Fernandes Lopes e Mário Meireles Teixeira, pela grande e inestimável contribuição que foi fundamental na elaboração deste trabalho.

Aos professores André Castelo Branco Soares e Samyr Béliche Vale por terem aceito o convite para participação na minha banca de mestrado.

Ao professores do Programa de Mestrado, cujos ensinamentos em muito contribuíram para que mais esta etapa de minha história acadêmica pudesse ser cumprida.

Aos meus colegas do Laboratório de Sistemas Distribuídos e Inteligentes – LSD*i*, pela amizade e troca de conhecimentos.

Aos demais colegas que direta ou indiretamente contriuíram para a realização deste projeto.

Resumo

O avanço no desenvolvimento de novas tecnologias e protocolos de redes contribuiu para o crescimento das redes de computadores, e consequentemente aumentou a necessidade de se utilizar simulações para projetar e analisar estas redes. Tais simulações precisam ser avaliadas e comparadas em diferentes cenários, em que nem sempre existem recursos computacionais disponíveis para executá-las. Em virtude disso, a utilização dos recursos computacionais da nuvem se mostra uma abordagem interessante devido às suas características intrínsecas, como a elasticidade e o provisionamento de recursos sob demanda. Nesse contexto, este trabalho propõe um modelo arquitetural, serviço e repositório de simulações de redes na nuvem da Amazon que utiliza a tecnologia de serviços web.

Palavras-chave: Simulação, Serviços Web, Computação em Nuvem.

Abstract

The breakthrough in the development of new technologies and network protocols contributed to the growth of computer networks, and consequently increased the need to use simulations to design and analyze these networks. Such simulations need to be evaluated and compared in different scenarios where there are not always available computing resources to execute them. The use of computing resources of cloud shows an interesting approach because of their intrinsic characteristics, elasticity and provisioning capabilities on demand. In this context, this work proposes an architectural model, service and network simulations repository in the Amazon cloud that uses web services technology.

Keywords: Simulation, Web Services, Cloud Computing.

Determinação coragem e autoconfiança são fatores decisivos para o sucesso. Se estamos possuídos por uma inabalável determinação conseguiremos superá-los. Independentemente das circunstâncias, devemos ser sempre humildes, recatados e despidos de orgulho. (Dalai Lama)

Sumário

Lista de Figuras

1 Introdução

Nos últimos anos, as redes de computadores têm experimentado uma grande expansão em termos do número de dispositivos conectados. O surgimento de novas aplicações computacionais tem motivado esse crescimento, requerendo o desenvolvimento de novas tecnologias e protocolos de rede. No entanto, realizar a montagem e a configuração de uma infraestrutura para a avaliação experimental desses novos recursos pode impor um custo elevado em termos financeiros e de tempo.

Por conta dessas dificuldades, projetistas de arquiteturas, protocolos e serviços de rede têm utilizado simuladores com o intuito de realizar experimentos considerando diferentes cenários de aplicação destas tecnologias. O uso de simuladores de rede permite o desenvolvimento e a análise de topologias complexas e novos protocolos, sem a necessidade de implementá-las fisicamente. Esse mecanismo proporciona, portanto, uma grande economia de tempo e dinheiro, permitindo que mudanças nas tecnologias de rede sejam facilmente testadas e avaliadas em grande escala, com potencialmente milhares de nós.

No entanto, as simulações computacionais têm se tornado cada vez mais complexas, requerendo uma grande quantidade de recursos computacionais para sua execução. Nesse contexto, plataformas de alto desempenho têm sido largamente adotadas na literatura para a execução de simulações dos mais diversos tipos [Minkovich et al., 2014, Du et al., 2013, Hollman and Marti, 2003, Marti et al., 2000, Abramson et al., 1999]. Por outro lado, a implantação e a manutenção de *clusters* de alto desempenho pode gerar um alto custo, inviabilizando a adoção desta solução.

Dessa forma, com o intuito de contribuir com uma solução para essa problemática, esta dissertação propõe utilizar recursos computacionais disponíveis em nuvem de computadores para execução de simulações de redes de computadores. Uma vez que as plataformas de Computação em Nuvem (*Cloud Computing*) fornecem acesso de modo conveniente e sob demanda a um conjunto de recursos computacionais configuráveis que podem ser rapidamente adquiridos e liberados com o mínimo esforço gerencial ou interação com o provedor de serviços [Mell and Grance, 2011]. Essas plataformas são capazes de prover recursos de processamento e armazenamento de alto desempenho para

diversos tipos de aplicações.

Entre os principais fatores que motivaram a adoção de tecnologias de computação em nuvem, merecem destaque os seguintes [Vaquero et al., 2008]:

1. Virtualização de recursos: os recursos da nuvem, como computadores, memória, capacidade de armazenamento e até mesmo a rede de interconexão são virtualizados de forma a gerar uma abstração dos recursos de *hardware*, tornando-os independentes de localização. Dessa maneira, esses recursos podem ser disponibilizados e consumidos como utilitários;

2. Independência de localização: na computação em nuvem, os serviços se tornam acessíveis a partir de qualquer localização geográfica com acesso à Internet;

3. Elasticidade: é a capacidade de provisionar e liberar rapidamente grandes quantidades de recursos computacionais em tempo de execução, sob demanda;

4. Modelo de pagamento baseado no consumo: no paradigma da computação em nuvem, os consumidores de serviços e recursos computacionais pagarão aos provedores apenas quando utilizarem seus serviços (*Pay-As-You-Go*).

Nesse contexto, esta dissertação especifica e implementa uma plataforma, disponível em uma nuvem de computadores, para a execução de simulações de rede de alto desempenho, chamada NSCloud (*Network Simulation in the Cloud*).

Diversos simuladores de rede estão disponíveis na literatura, sob diferentes licenças e com diferentes funcionalidades. Dentre eles, merece destaque o simulador NS-3 (*Network Simulator* 3), largamente adotado pela comunidade de pesquisa em redes de computadores. O NS-3 é um simulador de redes baseado em eventos discretos desenvolvido especialmente para pesquisa e uso educacional sob a licença GNU GPL (*General Public License*).

O NS-3 consiste em um sistema que permite simular e avaliar o desempenho de diversos cenários de redes, desde os mais simples aos mais complexos, utilizando as linguagens de programação C/C++ e Python. Além disso, outra vantagem do NS-3 é o fato dele ser totalmente gratuito e com código-fonte aberto, o que permite ao usuário proceder os ajustes que julgar necessários em sua estrutura. O simulador oferece suporte à simulação de um grande número de tecnologias de rede (com e sem fio) e diferentes

cenários. Por conta dessas características, o NS-3 foi escolhido como o simulador a ser utilizado inicialmente na plataforma NSCloud.

1.1 Objetivos

Este trabalho tem como objetivo propor e construir uma plataforma baseada em nuvem de computadores para execução de simulações de rede de alto desempenho, além de implementar serviço web de simulação baseado na arquitetura REST (*Representational State Transfer*) a ser incorporado na plataforma. Os objetivos específicos desta dissertação são:

- Investigar o estado da arte sobre Computação em Nuvem, e em específico, a nuvem da Amazon *Web Services* ;
- Realizar um estudo sobre o simulador NS-3 (*Network Simulator* 3);
- Definir um modelo arquitetural em nuvem do ambiente virtual de simulação NSCloud;
- Prototipar e validar o ambiente NSCloud na nuvem da Amazon;
- Desenvolver uma ferramenta baseada na web para utilização dos serviços em nuvem;
- Criar um repositório na nuvem contendo o resultado das simulações;
- Verificar e validar a plataforma e soluções desenvolvidas com auxílio de um conjunto de cenários representativos executados no NS-3.

1.2 Estrutura da dissertação

Esta dissertação está estruturada da seguinte maneira: o Capítulo 2 detalha a revisão bibliográfica sobre Computação em Nuvem e provedores de serviços, dando ênfase à nuvem AWS (*Amazon Web Services*). No Capítulo 3 é realizada uma revisão bibliográfica específica sobre alguns conceitos de simulação, assim como um estudo sobre alguns simuladores, com ênfase no simulador objeto de estudo deste trabalho, o NS-3. No Capítulo 4 são discutidos os trabalhos relacionados. O Capítulo 5 trata da plataforma de

simulação desenvolvida, composta por seu modelo arquitetural em nuvem e serviço Web de simulação. No capítulo 6 são apresentados os experimentos de validação da plataforma NSCloud realizados e, no Capítulo 7, são discutidas as conclusões e trabalhos futuros.

2 Computação em Nuvem

A Computação em Nuvem é definida por diversos autores na literatura, dentre os quais podemos citar Peter Mell e Timothy Grace do NIST (*National Institute of Standards and Technology*) entidade federal não regulatória vinculada ao Departamento de Comércio do Governo dos Estados Unidos, que define:

> Computação em nuvem como um modelo que possibilita acesso de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com o mínimo esforço gerencial ou interação com o provedor de serviços [Mell and Grance, 2011].

Vaquero [Vaquero et al., 2008] em seu estudo "*A Break in the Clouds: Towards a Cloud Definition*" propõe uma definição completa de Computação em Nuvem como:

> Uma larga gama de recursos virtualizados de fácil acesso e utilização (como *hardware*, plataformas de desenvolvimento e/ou serviços). Tais recursos podem ser reconfigurados dinamicamente para se ajustarem à carga de utilização (escala), permitindo também uma ótima utilização destes. Essa gama de recursos é tipicamente utilizada em num modelo de pagamento por utilização, com garantias dadas pelo fornecedor da infraestrutura por meio de SLA's "customizados".

Segundo Brantner [Brantner et al., 2008] Computação em Nuvem é:

> uma evolução dos serviços e produtos de tecnologia da informação sob demanda, também chamada de *Utility Computing*, computação utilitária, que tem como objetivo fornecer componentes básicos como armazenamento, processamento e largura de banda de uma rede como uma "mercadoria" por meio de provedores especializados com um baixo custo por unidade utilizada.

Por fim, o F5 Network em seu *Cloud Computing Survey* define Computação em Nuvem com sendo *"um estilo de computação no qual recursos dinamicamente escaláveis e geralmente virtualizados são disponibilizados sob a forma de serviços"* [Networks, 2009].

Nesse novo paradigma de utilização de recursos computacionais, os clientes abrem mão da administração de infraestrutura própria e dispõem de serviços oferecidos por terceiros, delegando responsabilidades e assumindo custos estritamente proporcionais à quantidade de recursos que utilizam [Coutinho, 2013].

Esta dissertação adota a definição do NIST que é amplamente utilizada e aceita. Além da definição, o NIST também propõe um modelo para a computação

em nuvem composto por: características essenciais, modelos de serviços e modelos de implantação conforme ilustra a Figura 2.1. As características essenciais, os modelos de serviços e modelos de implantação serão definidos nas seções seguintes.

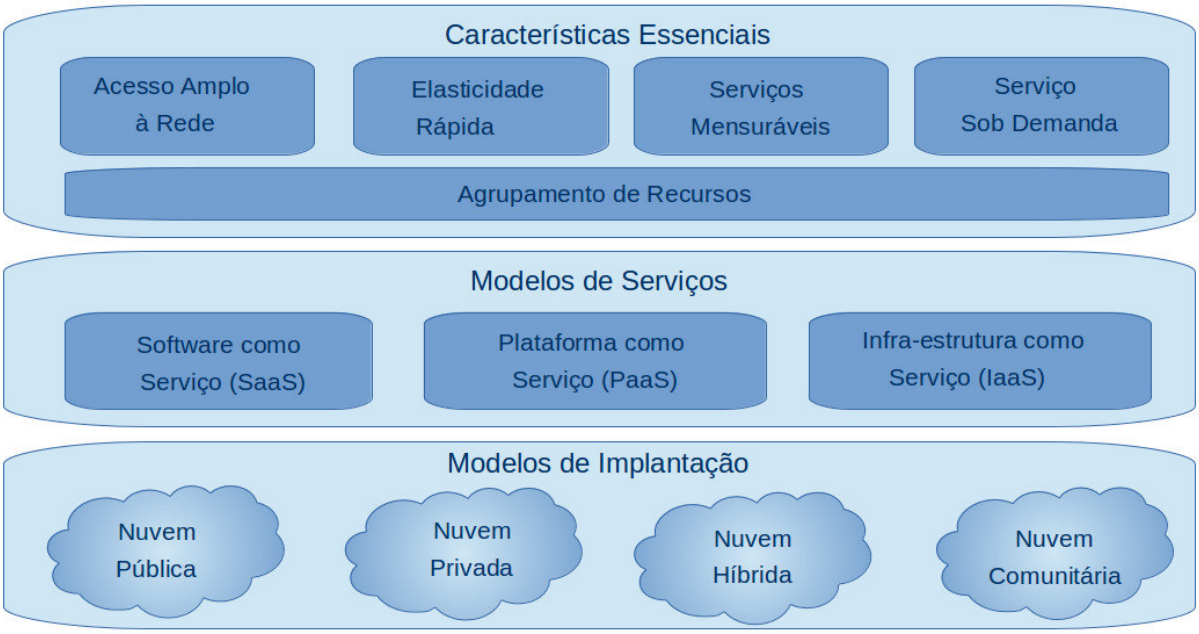

Figura 2.1: Representação Gráfica do Modelo do NIST para Computação em Nuvem.

2.1 Características Essenciais

De acordo com o NIST, as cinco principais características do modelo de computação em nuvem são: [Mell and Grance, 2011]

- Acesso amplo à rede – recursos e serviços disponíveis na rede que são acessados por meio de mecanismos padronizados que possibilitam o uso por intermédio de diferentes plataformas de *hardware*, bastando apenas o acesso à Internet por um dispositivo, para que se tenha acesso aos serviços fornecidos pelo provedor;

- Elasticidade rápida – recursos podem ser eficazmente provisionados e liberados, tanto para aumentar a capacidade, quanto para liberar recursos computacionais. Ou seja, esse fornecimento e liberação de recursos por parte dos provedores de serviços se dá de acordo com as necessidades dos consumidores. Assim, para um consumidor de recursos da nuvem, as capacidades disponíveis para aprovisionamento apresentam-se de forma ilimitada, podendo ser requisitada em qualquer quantidade e a qualquer momento;

- Serviços mensuráveis – controle e otimização do uso dos recursos por meio de mecanismos que fazer o monitoramento, controle e emissão de relatórios de cobrança, em níveis apropriados para cada tipo de serviço (largura de banda utilizada, armazenamento, processamento, etc.), atingindo-se assim mais transparência para os provedores e consumidores;

- Serviços sob demanda – é a capacidade que o consumidor possui de provisionar recursos computacionais sem a necessidade da intervenção humana. Por exemplo, o consumidor poderá solicitar uma nova instância de máquina virtual de forma automática e de acordo com a sua necessidade, e sem intervenção humana por parte do provedor de serviços;

- Agrupamento de Recursos – serviços acessíveis simultaneamente por múltiplos consumidores, com diferentes recursos físicos e virtuais alocados e realocados dinamicamente conforme a demanda. Igualmente, os consumidores não dispõem de conhecimento ou controle da localização exata dos recursos fornecidos pelo provedor.

2.2 Modelo de Serviços

O NIST define três modelos de serviços em nuvem frequentemente encontrados na literatura e universalmente aceitos, que define um padrão arquitetural para solucões em Computação em Nuvem. São eles: SaaS (*Software as a Service* – *Software* como Serviço), PaaS (*Plataform as a Service* – Plataforma como Serviço) e IaaS (*Infrastructure as a Service* – Infraestrutura como Serviço). Os serviços de SaaS e PaaS incluem os serviços de IaaS [Mell and Grance, 2011].

- SaaS (*Software* como Serviço) – modelo composto de aplicativos que são oferecidos pela Internet como serviço ou prestação de serviços. Tais aplicativos não são adquiridos pelo usuário, e sim alugados por meido de uma subscrição ou pago consoante a sua utilização. Como exemplo disponível no mercado podemos citar o Google Docs e o Gmail da Google e o Amazon SES (Serviço de Email Simples) da Amazon;

- PaaS (Plataforma como Serviço) – modelo que oferece uma infraestrutura para implementar e testar aplicações em nuvem. Tal infraestrutura pode ser constituída

por sistemas operacionais, linguagens de programação, sistemas de gerenciamento de banco de dados, servidores de aplicação, entre outros. Nessa camada o consumidor não gere ou controla a infraestrutura subjacente, composto por redes, servidores, sistemas operacionais ou áreas de armazenamento, mas controla as aplicações desenvolvidas e eventualmente as configurações do ambiente de trabalho do sistema de desenvolvimento. Alguns exemplos de serviços que podem ser classificados nessa modalidade são: o Force.com, da SalesForce, o Google App Engine, da Google e o Amazon Elastic BeanStalk, da Amazon;

- IaaS (Infraestrutura como Serviço) – corresponde ao modelo que disponibiliza a infraestrutura computacional como serviço para os demais modelos (PaaS e SaaS). Seu objetivo é prover recursos, tais como servidores, armazenamento, rede, dentre outros, para construir um ambiente de aplicação sob demanda, que pode incluir sistemas operacionais e aplicativos. Essa infraestrutura se baseia na virtualização de recursos computacionais, sendo dinamicamente escalonável para aumentar ou diminuir os recursos de acordo com a necessidade da aplicação. Por possuir essa característica, a cobrança é feita pela utilização ou pela quantidade de recursos contratados, e os custos para os consumidores em relação aos sistemas de tecnologia da informação tradicionais são bastante inferiores, uma vez que não é necessário adquirir e manter servidores, equipamentos de rede, custo com espaço de instalação, pessoal técnico, etc. O Amazon EC2 (*Elastic Cloud Computing*) da Amazon, o BlueCloud da IBM, e o Eucalyptus (*Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*) da Hewlett-Packard, são exemplos de serviços do tipo IaaS.

O modelo de referência do NIST é complementado pela CSA (*Cloud Security Alliance*), que defende a inclusão de uma característica essencial no modelo, a *multi-tenancy*, que é considerada como sendo indispensável para os aplicativos operados em nuvem [CSA, 2015]. *Multi-tenancy* permite que multiplos clientes (*tenants*) compartilhem os mesmos recursos físicos, como por exemplo, os aplicativos *Enterprise Resource Planning* (ERP), mas permaneçam logicamente isolados [Taurion, 2009].

2.3 Modelos de Implantação

Quanto à disponibilidade, existem vários tipos de modelos de implantação para os ambientes de Computação em Nuvem. O processo de negócio, o tipo de informação e o nível de visão desejado são responsáveis por definir a restrição de abertura ou acesso à nuvem, conforme descrito a seguir [Coutinho, 2013]:

- Nuvem privada – tipo de infraestrutura de nuvem que é utilizada exclusivamente por uma organização, sendo local ou remota, administrada pela própria empresa ou por terceiros;
- Nuvem pública – sua infraestrutura de nuvem é disponibilizada para o público, sendo acessada por qualquer usuário que conheça a localização do serviço;
- Nuvem comunitária – tipo de nuvem que fornece uma infraestrutura compartilhada por uma comunidade de organizações com interesses comuns;
- Nuvem híbrida – infraestrutura de nuvem composta de duas ou mais nuvens, que podem ser do tipo privada, pública ou comunitária, e que continuam a serem entidades únicas, mas conectadas por meio de tecnologias proprietárias ou padronizadas, que permitem a portabilidade de aplicações e dados.

2.4 Provedores de Nuvem

Um provedor de nuvem é responsável por fornecer recursos virtualizados, conforme os modelos de serviços descritos anteriormente. Cada provedor define como um determinado ambiente será disponibilizado ao consumidor, ou seja, quais são os padrões, aplicações e tecnologias que estarão disponíveis. Atualmente, existem vários provedores de nuvem, tais como OpenNebula, Google App Engine, Microsoft Azure e Amazon Web Services (AWS), sendo este último utilizado nesta dissertação. Na seção a seguir será definida a composição, conceitos e alguns serviços disponíveis na nuvem da Amazon.

2.4.1 Amazon *Web Services* (AWS)

A Amazon Web Services fornece recursos de computação e serviços sob demanda pela Internet. Dentre os serviços mais utilizados estão os de hospedagem de

sites, banco de dados e armazenamento. Todos os recursos são cobrados somente pelo uso.

Um dos benefícios da computação em nuvem oferecido pela Amazon é o termo elasticidade e escabilidade automática. O termo elasticidade diz respeito a algo cuja disponibilidade cresce conforme a demanda, e conforme o crescimento podem ser disponibilizados novos recursos que podem ser ajustados dinamicamente.

A plataforma de computação em nuvem da AWS é subdividida por regiões, que correspondem às regiões geográficas de sua atuação no mundo, zonas de disponibilidade e pontos de presença. As zonas de disponibilidade representam diferentes centros de dados isolados que estão alocados dentro de diferentes regiões. Por fim, os pontos de presença, são servidores espalhados dentro de uma região, especializados em entregar conteúdo com altas taxas e baixa latência. Atualmente as regiões que contemplam a sua infraestrutura são: Oeste dos EUA (*Oregon Califórnia*), Leste dos EUA (Virgínia), América do Sul (São Paulo), União Europeia (Dublin, Frankfurt), Ásia-Pacífico (Cingapura, Sydney, Tóquio) e China (Pequim) [AWS, 2015a].

No relatório mundial sobre recursos críticos de infraestrutura de nuvem pública como um serviço, o Gartner [Gar, 2015] classificou a Amazon Web Services com a maior pontuação em todos os quatro casos de uso a seguir: desenvolvimento de aplicativos (4,81 de 5), computação em lote (4,81 de 5), aplicativos nativos da nuvem (4,84 de 5) e aplicativos de negócios em geral (4,53 de 5), conforme ilustra a Figura 2.2.

A Amazon AWS fornece *softwares* como serviço (SaaS), e como exemplo podemos citar o serviço de envio de mensagem do tipo *push* SNS (*Simple Notification Service*). Além disso, fornece infraestrutura como serviço (IaaS), e como exemplo temos o EC2 (*Amazon Elastic Compute Cloud*). Também fornece PaaS, disponibilizando um ambiente que permite a execução de aplicações, facilitando o gerenciamento e a manutenção. Como exemplo podemos citar o Elastic Beanstalk, que é um serviço que facilita a instalação e manutenção de aplicações na nuvem da AWS [Lecheta, 2014]. A Figura 2.3 ilustra os serviços disponibilizados pela nuvem da Amazon.

É importante ressaltar que o conceito de serviço vai além do que é comumente atribuído, isto é, serviço na nuvem é qualquer tipo de recurso disponível, como: *software*, *hardware*, redes, banco de dados, etc.

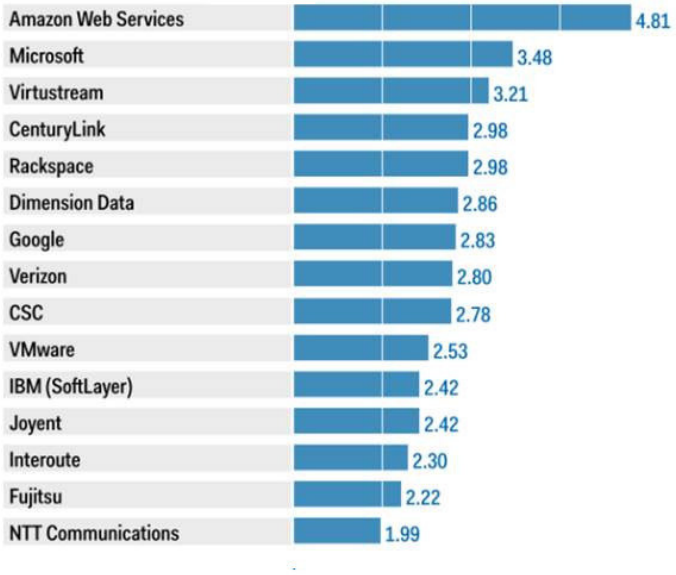

Figura 2.2: Classificação por Produto ou Serviços de Desenvolvimento de aplicações [Gar, 2015].

Descrição de serviços AWS

A Amazon AWS classifica seus serviços, conforme a Figura 2.3, como sendo de computação, armazenamento e distribuição de conteúdo, banco de dados, dados analíticos, serviços móveis, Internet das coisas, ferramentas de desenvolvimento, ferramentas de gerenciamento, segurança e identidade e serviços de aplicações. Dentre os diversos serviços, serão descritos a seguir os que irão compor a plataforma NSCloud.

- Amazon EC2 (*Elastic Compute Cloud*) – é um serviço que permite lançar instâncias (criar máquinas virtuais) na nuvem com variadas configurações de Sistemas Operacionais, tais como Linux e Windows, assim como diferentes configurações de processador, memória e armazenamento de dados. Devido à Amazon possuir o conceito de regiões geográficas para fornecer alta disponibilidade para as aplicações e tornar mais rápido o acesso a determinados locais, as instâncias de servidores podem ser criadas em qualquer região das descritas anteriormente. Elas são classificadas de acordo com o tipo (*t2, m3, c3 e r3*) que definem tipo de armazenamento, quantidade de memória e processador. No Apêndice A.1 é ilustrado o ambiente em que as máquinas são lançadas, assim como seus atributos, tais como: nome, identificador

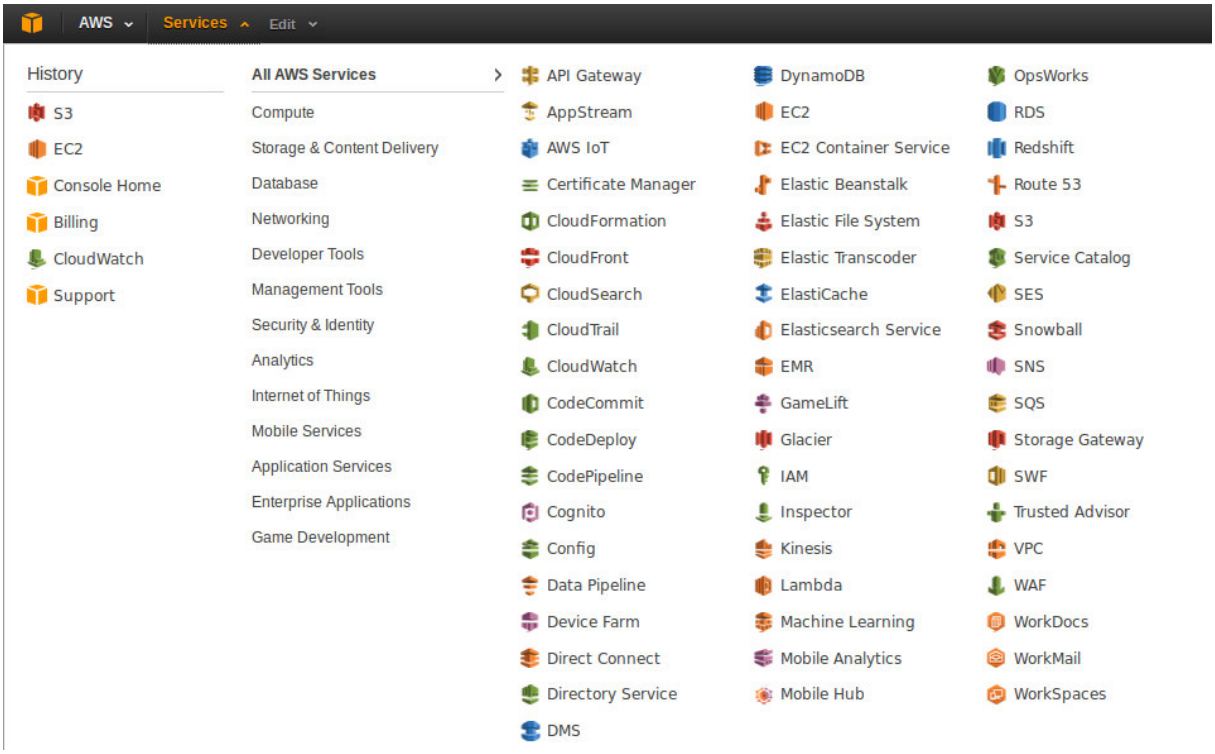

Figura 2.3: Serviços disponíveis na nuvem da Amazon [AWS, 2015a].

da instância, tipo, zona de disponibilidade, *status* e DNS público;

- Amazon S3 (*Simple Storage Service*) – é um serviço de armazenamento de dados amplamente utilizado por soluções bem conhecidas como o Dropbox, que armazena os dados de forma criptografada no S3. Esse serviço armazena dados como objetos, que podem ser organizados em pastas, dentro de *buckets*, e cada objeto dentro do S3 é um arquivo composto pelos seus dados e metadados que descrevem este arquivo. Existe uma limitação no tamanho dos objetos armazenados no S3, que pode variar de 1 *byte* a 5 *terabytes*, e na quantidade de *buckets* criados, que é de 100 *buckets* por conta de usuário. Além do armazenamento de arquivos, é possível também armazenar *sites* de conteúdo estático. No Apêndice A.2 é ilustrado o ambiente em que são criados e listados os *buckets*, pastas e adicionados objetos nos mesmos;

- Amazon SES (*Email Simple Service*) – serviço disponível pela AWS para envio de e-mails. O SES permite o envio de duas mil mensagens dentro de instâncias EC2 ou Beanstalk gratuitamente por dia, e fora desse ambiente é cobrado um determinado valor para cada mil e-mails enviados. Além disso, é disponibilizada uma API para o envio de e-mails com Android, iOS, Java, .NET, Node.js, Python, PHP e Ruby;

- Amazon EBS (*Elastic Block Store*) – serviço que permite a criação de volumes de

armazenamento anexados a instâncias Amazon EC2. Os volumes são dispostos em zona de disponibilidade específica, em que são replicados automaticamente a fim de protegê-los de falhas em componentes. A Amazon oferece várias opções que permitem a otimização e desempenho de armazenamento, estando divididas nas categorias de armazenamento sustentado por SSD (*Solid State Drive*), para cargas de trabalho transacionais, como banco de dados e volumes de inicialização, e armazenamento sustentado por HDD (*Hard Disk Drive*), para cargas de trabalho com alto consumo da taxa de transferência, como processamento de *logs* [Ama, 2015]. Os volumes podem ser criados e associados a instâncias EC2, mas podem persistir independente delas. Além disso, ao serem criados, automaticamente são replicados dentro da mesma zona de disponibilidade;

- Amazon IAM *Identity and Access management* – permite gerenciar o controle de acesso aos serviços e recursos da AWS pelos usuários. Ou seja, ele permite criar e gerenciar permissões de acesso aos serviços concedidos aos usuários;

- Amazon AMI *Amazon Machine Image* – proporciona a escolha de imagem AMI de acordo com o sistema operacional e distribuição Linux para a instância EC2. Além do mais, permite salvar uma imagem AMI de uma instância EC2 completa, com os vários *softwares* instalados.

2.5 Considerações Finais

Neste capítulo foram apresentadas algumas definições encontradas na literatura sobre computação em nuvem. Além disso, foram descritas suas características, modelos de serviços e implantação. Também foram objetos de estudo alguns serviços disponíveis na nuvem da Amazon que serão utilizados nesta dissertação.

3 Simulação Computacional

Neste capítulo é apresentada uma revisão bibliográfica a respeito de simulação computacional e da sua importância como ferramenta na tomada de decisões. Além de uma descrição e breve comparação entre alguns simuladores comumente utilizados.

Segundo HARRELL [Harrell et al., 2000], simulação é a imitação de um sistema real modelado em computador para avaliação e melhoria do seu desempenho, isto é, é a importação da realidade para um ambiente controlado onde se pode estudar o comportamento do mesmo sob diversas condições sem riscos físicos e/ou grandes custos envolvidos. Por outro lado, EHRLICH [Ehrlich, 1991], define como sendo um método empregado para estudar o desempenho de um sistema por meio da formulação de um modelo matemático, o qual deve reproduzir, da maneira mais fiel possível, as características do sistema original. Esse autor afirma ainda que, por meio da simulação não é possível obter, de imediato, resultados que levem à otimização de um objetivo desejado. Entretanto, é possível simular, por meio do modelo, uma série de experimentos em diferentes condições e, posteriormente, escolher a condição cujos resultados sejam mais aceitáveis.

A simulação possibilita o estudo sobre os sistemas ainda na fase de concepção, antes de serem efetivamente implementados. Assim, ela pode ser usada como uma ferramenta para prever os efeitos de mudanças em sistemas existentes e também como uma ferramenta para avaliar e validar o desempenho de novos sistemas. Além disso, as técnicas de simulação permitem a obtenção de respostas a eventos que não ocorrem naturalmente e com frequência nos sistemas reais, viabilizando a busca por detalhes às vezes não permitidos nesses sistemas.

A instalação de infraestrutura semelhante a um sistema real em diferentes ambientes teria um custo elevado (embora seus resultados sejam mais precisos) com a inclusão de mais máquinas, tanto do ponto de vista financeiro como também do tempo despendido nesse processo. Assim, a simulação computacional é apresentada como uma solução viável para estudo de redes de computadores, já que os custos são menores e pode-se atingir resultados a partir de diferentes topologias e quantidade de estações, com

um tempo relativamente pequeno, sendo apenas necessário a configuração dos parâmetros de simulação.

3.1 Tipos de Simulação

Simulações, como a maioria dos métodos de análise, envolvem sistemas e modelos que os representam [Kelton, 2003]. Os tipos de simulação estão diretamente relacionados com a forma em que as simulações são realizadas (levando em consideração ou não o tempo, variáveis discretas ou contínuas, distribuições estatísticas etc.). HARRELL[Harrell et al., 2000], as classifica em:

- Estática e dinâmica – tipo de simulação que visa representar o estado do sistema levando ou não em consideração a variável tempo. A estática não adota o tempo como parâmetro, e a dinâmica utiliza. As simulações dinâmicas são apropriada para análise de sistemas que sofrem alterações ao longo do tempo de simulação;
- Determinística e estocástica – são tipos de simulações em que as entradas são, respectivamente, constantes ou regidas por distribuições de probabilidade (com entradas variando seu estado teremos saídas também variáveis, gerando um novo conjunto de resultados).
- Discreta e Contínua – simulações em que as variáveis mudam instantaneamente em pontos específicos de tempo, ou podem mudar de estado continuamente, respectivamente.

3.2 Vantagens e Desvantagens da Simulação

O uso de simulações nos leva a obtenção de algumas vantagens, dentre as quais podemos citar [1,5,15]:

- Um modelo pode ser utilizado inúmeras vezes para avaliar projetos propostos;
- Geralmente métodos analíticos são mais difíceis de aplicar do que a simulação;
- Possuem grande flexibilidade, pois se aplicam aos mais variados modelos;

- Os modelos analíticos requerem maior número de simplificações que os modelos de simulação, que por sua vez possuem maiores níveis de detalhes, podendo assim analisar melhor o sistema;
- O tempo de simulação é independente do tempo real, ou seja, durante a simulação pode-se acelerar ou retardar a reprodução dos fenômenos, para estudá-los melhor;
- Possui um processo de modelagem evolutivo. Inicia-se com um modelo simples e aumenta-se sua complexidade aos poucos, observando as peculiaridades do problema;
- Os resultados de uma simulação, submetidos a uma série de etapas de modelagem, teste e validação, têm melhor aceitação que a opinião de uma única pessoa.

Os autores acima citados também apontam as desvantagens relacionadas ao uso da simulação, dentre elas:

- Alguns sistemas complexos levam muito tempo sendo modelados ou executando. A tentativa de simplificar o modelo pode levar a resultados inconsistentes com o mundo real;
- Em algumas simulações os resultados são de difícil interpretação. É difícil determinar quando uma observação durante uma "rodada" da simulação se deve a alguma relação relevante do sistema, ou a processos aleatórios intrínsecos ao modelo;
- A construção de modelos requer treinamento, a técnica é aprendida e aperfeiçoada com o tempo por conta da experiência;
- A programação de um modelo pode tornar-se uma tarefa árdua e dispendiosa se os recursos computacionais, principalmente a ferramenta de simulação escolhida, não forem apropriados.

3.3 Ferramentas de Simulação

Nesta seção é realizada uma breve descrição de alguns simuladores de rede. Entretanto, serão evidenciadas as características e composição do simulador NS-3 com um maior nível de detalhamento, visto que esse simulador foi adotado na construção da Plataforma NSCloud.

3.3.1 OPNET Modeler

O OPNET Modeler é um simulador para redes de comunicação de dados proposto por Alain Cohenś em seu trabalho de graduação no MIT (*Massachusetts Institute of Technology*) em 1987, e desde de outubro de 2012 passou a ser desenvolvido e comercializado pela Riverbed Technology. Ele fornece uma interface gráfica com o usuário para a modelagem da rede, execução das simulações e análise dos dados, conforme mostrado na Figura 3.1. Também, possui uma biblioteca que contém diversos modelos de dispositivos comerciais e tecnologias de redes.

Além dos modelos previamente definidos, o simulador permite ao usuário criar seus próprios modelos, como por exemplo, transferência de arquivos, ou dispositivos como comutadores. Essa ferramenta posibilita a análise do projeto de uma rede de comunicação antes de sua operação comercial, testes de equipamentos, desenvolvimento de novas tecnologias, protocolos e dispositivos [Chang, 1999]. O OPNET utiliza a linguagem de progração C e em lançamentos recentes foi incluído o suporte à linguagem C++. As configurações iniciais (topologia e parâmetros) geralmente são obtidas por meio de interface grafica do usuário (*Graphical User Interface*), um conjunto de arquivos XML ou por meio de chamadas a bibliotecas em C [Chang, 2012].

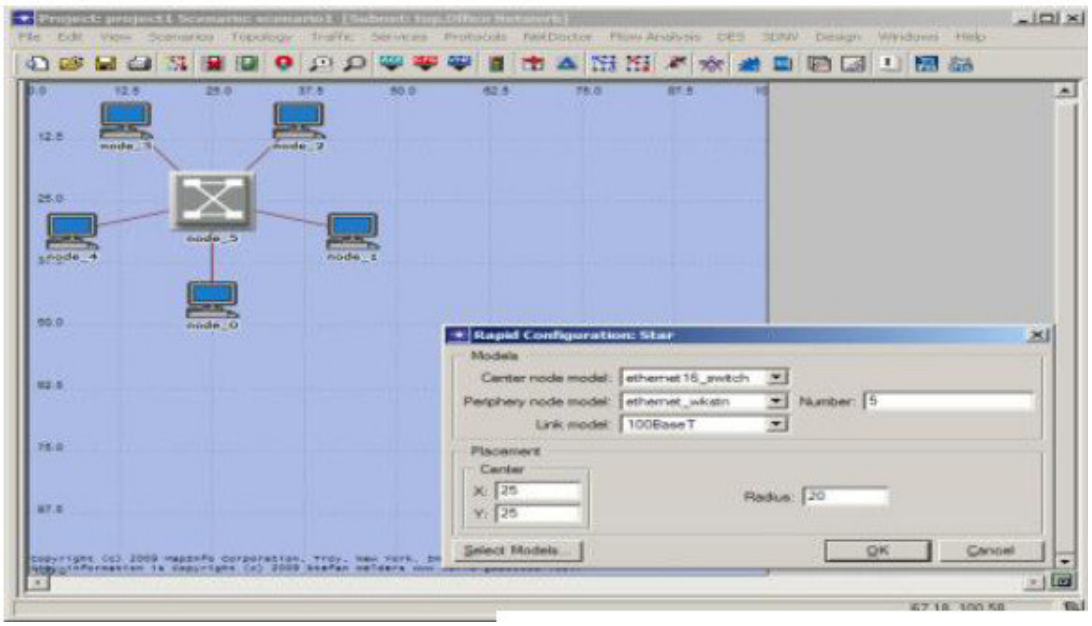

Figura 3.1: Interface gráfica da OPNET.

3.3.2 OMNET++

O OMNeT (*Objective Modular Network Testbed in C++*) é um simulador de eventos discretos, orientado a objetos, desenvolvido na linguagem C++ e baseado em componentes (módulos) hierarquicamente aninhados que se comunicam por troca de mensagens. Possuidor de uma arquitetura genérica e flexível que permite que ele seja utilizado na modelagem [OMN, 2015]. Além disso, possui um ambiente integrado de desenvolvimento (IDE) próprio, conforme a Figura 3.2, com diversas ferramentas que facilitam a implementação e a análise dos resultados das simulações.

A estrutura do modelo de simulação no OMNeT é definida com a linguagem NED (*Network Description*), utilizada na descrição de topologia usada para definir esses modelos (módulos e suas interconexões).

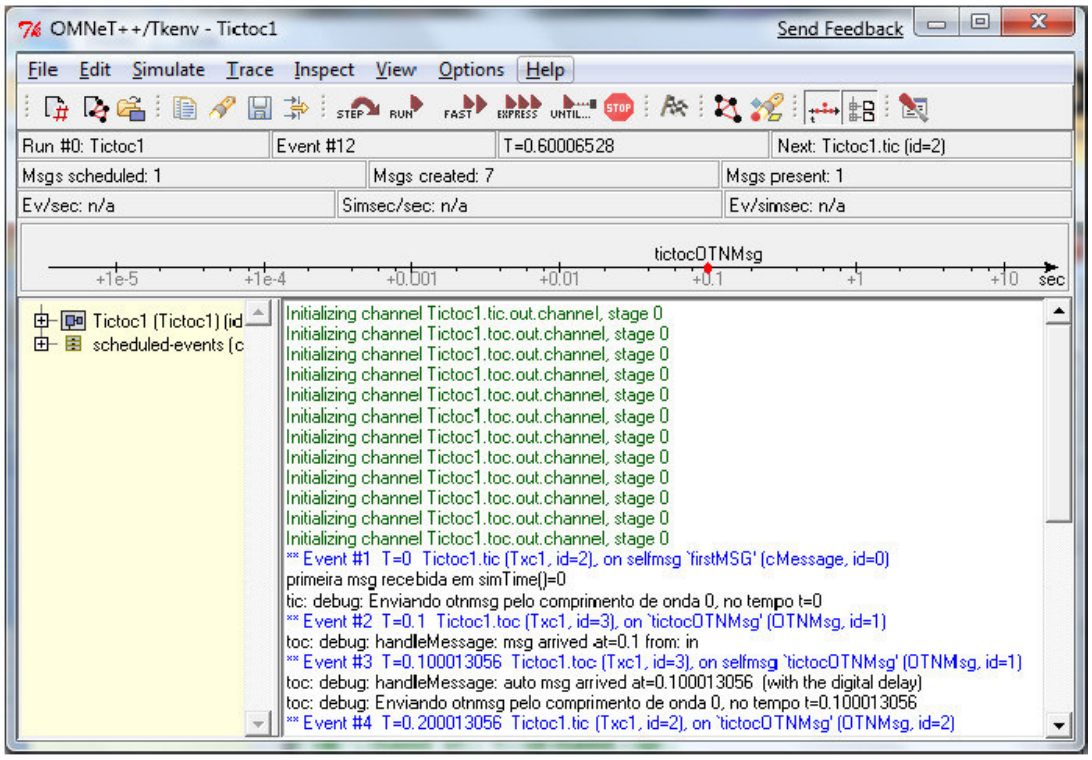

Figura 3.2: Interface gráfica OMNET++.

3.3.3 Network Simulator 2

O Network Simulator (NS-2) é um simulador de eventos discretos de código aberto, orientado à pesquisa em redes de computadores e desenvolvido pelo projeto VINT (*Virtual InterNetWork Testbed*). Esse projeto é composto pelo laboratório PARC (*Palo*

Alto Research Center) da Xerox, UCB (*University of California at Berkeley*), LBNL, e USC/ISI (*University of Southern California*). O NS-2 utiliza em sua programação as linguagens C++ para a estrutura básica (protocolos, agentes) e OTCL (*Object-oriented Tool Command Language*) para uso como *front-end*. Esse simulador conta com versões para os sistemas operacionais FreeBSD, Linux, SumOs, Solaris e Windows [Berkeley and ISI, 1998].

O NS-2 visa permitir a simulação de diferentes cenários de simulação, desde redes locais a redes de longa distância. Sua implementação suporta diferentes protocolos e arquiteturas, e ele pode ainda ser associado a diversas ferramentas, dentre as quais podemos citar o "Nam" (*Network Animator*), que é um ambiente que possibilita a reprodução de animações gráficas da rede simulada, e o *Nam Graphical Editor*, que é uma interface gráfica para a entrada de dados no simulador. Uma simulação no NS-2 consiste basicamente em planejar a simulação, definir os nós, definir a ligação entre os nós (topologia), definir o tráfego que será injetado na rede e analisar os resultados.

3.3.4 Network Simulator 3

O *Network Simulator* 3 é um simulador de eventos discretos, para pesquisa e uso educacional, desenvolvido em um projeto de código aberto com licença GNU GPLv2, que permite aos pesquisadores estudar diversos protocolos em ambientes controlados e em sistemas de larga escala [nsn, 2015a]. Esse projeto é financiado por várias instituições como a Universidade de Washington, o Georgia Institute of Technology e o ICSI Center for Internet Research, e o suporte por parte do Planete research group do INRIA Sophia-Antipolis.

O NS-3 é escrito nas linguagens C++ e Python e está disponível para os sistemas operacionais Linux, OSX e Windows através da utilização do Cygwin, possui arquitetura modularizada e orientada a objetos, facilitando o entendimento e reuso. Ademais, encontra-se em constante desenvolvimento, tanto em nível de suas capacidades de simulação, quanto em nível das ferramentas disponibilizadas por terceiros, como as ferramentas de animação gráfica Nam e ns-3-pyviz, dentre outras.

Este simulador foi completamente reescrito, e sua composição arquitetural se baseou nos princípios de escalabilidade, extensibilidade, modularidade, emulação, projeto claro e extensa documentação [Lacage and Henderson, 2006]. Esse novo modelo

arquitetural impossibilita a compatibilidade entre as descrições de cenários com o NS-2. Sua estrutura é baseada em módulos, distribuídos conforme a Figura 3.3, havendo dependências dos módulos superiores em relação aos inferiores [nsn, 2015b].

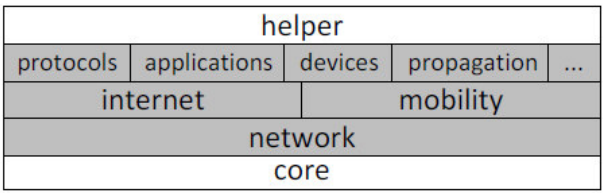

Figura 3.3: Organização Modular do NS-3.

Os módulos *'Core'* e *'Network'* são módulos base para qualquer tipo de simulação. O *'Core'* é constituído por variáveis globais, listas de atributos e outros. Possui suporte à depuração de cenários de redes, utilização de *"smart points"* na otmização e gestão de memória, e detém um conjunto de classes que podem ser utilizadas de forma a recolher dados estatísticos [Sousa, 2013].

O módulo *'Mobility'* possui as declarações necessárias à abordagem de movimento dos nós na rede, ou seja, é o módulo que contém os modelos de mobilidade estático e aleatório, além de ser responsável por monitorar todos os objetos existentes na rede, de acordo com sua posição e velocidade.

O módulo *'Applications'* permite a instalação de aplicações em determinados nós para transmissão pelos canais previamente definidos. Além disso, dentre várias funcionalidades ofertadas por este módulo, temos o estabelecimento de tempo em que a aplicação deve estar gerando tráfego ativamente, e os tempos em que a mesma não gerará tráfego. Já o módulo *'Helper'* possibilita a criação de cenários de forma mais descomplicada invocando todos os objetos dos módulos anteriormente citados.

Além dos módulos citados acima, o *Network Simulator 3* se utiliza de alguns conceitos e abstrações que são comumente usados por profissionais da área de redes de computadores, tais como:

- *Node* (nó) – abstração de dispositivo que se conecta à rede, sendo representado pela classe *Node* implementada em C++, que é responsável por fornecer os métodos para gerenciar as representações dos dispositivos computacionais nas simulações;
- *Application* (aplicação) – são responsáveis por gerar e direcionar as atividades na rede simulada. Esse papel é executado pela classe *Application*;

- *Channel* (canal) – corresponde ao meio pelo qual os dados são transmitidos: o ar como canal de transmissão em redes sem fio ou o cabo pelo qual um computador se conecta a uma rede *Ethernet* convencional. A abstração do canal é realizada pela classe *Channel* que é responsável por receber as conexões de *Nodes* e gerenciar as comunicações entre os que estão integrados ao canal;

- *Net Device* (dispositivo de rede) – corresponde aos dispositivos de rede controlados por meio de *drivers*, que no NS-3 são representados pela classe *NetDevice*, que fornece métodos para gerenciar conexões com os objetos *Node* e *Channel*.

3.3.5 Principais Característas

A comparação entre os simuladores pode ser sistematizada com a análise de parâmetros importantes. Nesse sentido, os principais pontos de análise são: licença de utilização, que mostra a disponibilidade da aplicação para uso e acessibilidade de seu código-fonte, permitindo uma melhor capacidade de compreensão de funcionamento; arquitetura de desenvolvimento adotada; linguagem de programação suportada, o que reflete principalmente em questões de desempenho e portabilidade da ferramenta; técnica de simulação usada; e presença ou ausência de interface gráfica para o usuário, o que simplifica e melhora a experiência do usuário com a ferramenta.

Tabela 3.1: Característas dos simuladores.

Simulador	Licença	Arquitetura	Linguagem	Técnica	Interface
OPNET	Comercial, Acadêmico	Orientada a Objetos	C / C++	Eventos Discretos, Híbridos, Analíticos	GUI integrado
OMNeT++	Software Livre, Comercial	Componentes	C++ / NED	Eventos Discretos	Eclipse
NS-2	Software Livre	Orientada a Objetos	OTcl / C++	Eventos Discretos	NAM (limitado), XGraph, etc.
NS-3	Software Livre	Orientada a Objetos	C++ / Python	Eventos Discretos	NetAnim, PyViz e etc

Quanto a licença conforme ilustrado na Tabela 3.1, temos o domínio de dois tipos de licença: a comercial, que é baseada no pagamento para o uso do *software*, e a licença GNU GPL que dá liberdade para executar, copiar, distribuir, estudar e alterar o software. Além disso, existem licença para uso em ambientes acadêmicos, para fins de

ensino e pesquisa, não comercial.

Nas colunas “Arquitetura” e “Linguagem” da Tabela 3.1, são definidos os paradigmas de desenvolvimento e linguagens dos simuladores apresentados, havendo a predominância do paradigma de orientação a objetos e da linguagem C++, a pesar do OMNeT++ ser orientado a componentes. Além do mais, temos a presença de mais de uma linguagem usada por simulador.

Quanto à técnica de simulação empregada, a maioria dos simuladores usam a simulação baseada em eventos discretos (variáveis de estado mudam de acordo com valores discretos de tempo), comum a vários simuladores de redes. E entre os simuladores examinados, optamos nesta dissertação pelo uso do *NS-3*, que possui ampla aceitação pela comunidade acadêmica, vasta documentação, e comunidade de colaboradores que asseguram um desenvolvimento contínuo da ferramenta.

3.4 Considerações Finais

Neste capítulo foram definidos os conceitos, tipos, vantagens e desvantagens do uso da simulação computacional, além da apresentação de alguns simuladores e de análise comparativa entre os mesmos. Dando ênfase na descrição do simulador NS-3 que objeto de estudo desta dissertação.

4 Trabalhos Relacionados

Para o desenvolvimento desta proposta de Plataforma e serviço de simulação em nuvem foram analisados cinco trabalhos. Para determinar a escolha dos trabalhos analisados foram levadas em consideração: descrição de simulação de redes em documento XML (*Extensible Markup Language*), simulação em nuvem e plataforma de simulação em nuvem. A seguir são apresentadas essas abordagens.

4.1 Interoperabilidade e Otimização da Gestão de Redes com a *Framework* NSDL [Marques, 2014]

O trabalho de Marques [Marques, 2014] busca promover a interoperabilidade entre ferramentas de diversos domínios de rede, em especial as ferramentas do domínio da simulação de redes, e que suportem as redes com fios, as redes sem fios e as com Qualidade de Serviço. Além disso, este trabalho propõe a criação de uma estrutura de dados que inclua todos os objetos de redes atuais e futuros, e que permita, de forma simples e completa, englobar as informações contidas nas ferramentas de rede atuais e futuras. Tais objetivos serão possíveis a partir da proposta de implementação de um arcabouço para a integração de ferramentas de rede, de uma linguagem chamada NSDL (*Network Scenario Description Language*) para a descrição de redes e de todos os seus componentes, além de metodologia para a integração dos componentes, ferramentas e linguagem, no arcabouço [Marques, 2014].

4.1.1 O arcabouço NSDL

O objetivo do arcabouço é propor um estrutura onde diferentes aplicações de rede possam ser integradas e onde existam mecanismos que suportem a interoperabilidade entre essas aplicações. Ele não deverá acomodar apenas as aplicações atuais, mas também, outras aplicações a serem desenvolvidas ou integradas futuramente.

O arcabouço é constituído por três camadas, conforme é ilustrado na

Figura 4.1. Cada uma delas possui um conjunto específico de funcionalidades e aplicações. A camada do topo inclui as aplicações que permitem a interação entre o utilizador e os cenários de rede. A camada central, é constituída pelos componentes responsáveis pela integração das ferramentas, e é nessa camada que o componente central, a linguagem NSDL se situa.

A linguagem foi escrita para suportar a descrição de redes de dados e a informação de cenários em que elas possam operar. A camada de base inclui as aplicações que executam alguma tarefa sobre as redes e permitem obter dados e informações diversas [Marques, 2014].

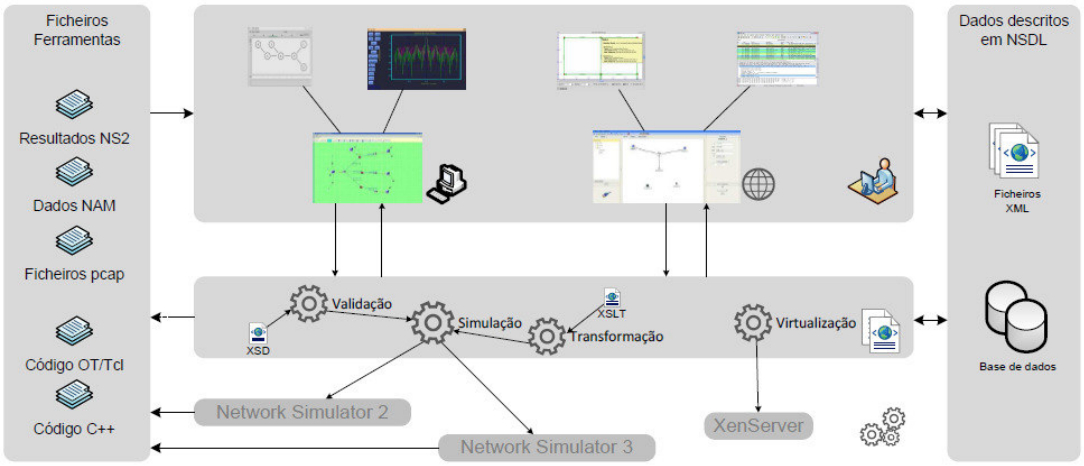

Figura 4.1: Estrutura em três camadas - arcabouço NSDL.

4.1.2 Linguagem NSDL

A linguagem NSDL surgiu após um levantamento sobre as características mais importantes em uma linguagem para a descrição de redes de comunicação, tendo se sobresaído as características de simplicidade, extensibilidade e independência. Essa linguagem seguiu estes princípios ao longo da sua definição, de forma a poder ser considerada uma solução para a interoperabilidade entre ferramentas de rede. Além disso, ela utilizou a linguagem XML (*Extensible Markup Language*) como forma de descrição de topologias e objetos de rede, descrevendo esses elementos por meio de marcações XML.

A NSDL dividiu os dados da rede em dois grandes grupos: `<network>` e `<scenarios>`. O elemento `<network>` inclui os objetos de rede comuns, como `<node>`, `<link>`, `<application>`, `<protocol>`, `<interface>` e `<domain>`. No elemento `<scenarios>` estarão os componentes dos cenários de rede, como por exemplo, os

cenários de simulação e os cenários de visualização. Os primeiros focam na descrição de experiências de simulação sobre a rede definida, enquanto que os segundos focam na configuração da visualização da rede em ambientes gráficos.

Além do arcabouço e da linguagem, o autor desenvolveu bibliotecas para suporte a redes com Qualidade de Serviço no âmbito do NS-2, suporte às redes sem fio no domínio do NS-3, e bibliotecas para a tradução de descrições de redes NSDL com a finalidade de executar simulações em ambientes virtualizados.

4.1.3 Autoria e Simulação de Cenários de Redes em NS-3 [Sousa, 2013]

O objetido do trabalho de Sousa [Sousa, 2013] é fornecer aos utilizadores uma ferramenta colaborativa que permita descrever os seus próprios cenários de redes, baseando-se numa linguagem XML (*Extensible Markup Language*) denominada NSDL (*Network Scenario Description Language*), traduzindo esta estrutura XML para um *script* C++ que possa ser executado no ambiente do NS-3. Tal solução permite a rápida criação de cenários de redes através de estruturas de dados XML para posterior tradução para C++ e então, execução no NS-3. Este trabalho visa a otimização da autoria de cenários de redes com a introdução de um perfil NS-3 para o NSDL [Sousa, 2013]. Linguagem de descrição de cenários de redes proposta Marques [Marques, 2014] ao criar o arcabouço de mesmo nome, objetivando a interoperabilidade entre os diferentes simuladores existentes.

A NSDL contém um conjunto de *tags* que formam o mundo lexical de *tags* que são permitidas. Este conjunto de *tags* está incluída no perfil base pois são utilizadas para especificar elementos que são comuns à maioria das ferramentas de simulação de cenários de redes. O conceito de perfil foi introduzido para as eventuais extensões da NSDL base, como é o caso do perfil do NS-3 proposto neste trabalho. Uma extensão da linguagem é formada pelo perfil base e um conjunto de tags novas que permitem especificar objetos que existam exclusivamente no contexto da plataforma destino.

No perfil proposto novos elementos foram adicionados com o propósito de suprir as seguintes situações, sejam variáveis globais específicas do simulador NS-3, sejam tecnologias desenvolvidas no domínio do NS-3, e consequentemente não existam no NS-2, ou sejam limitações/exigências da plataforma para que determinado elemento seja

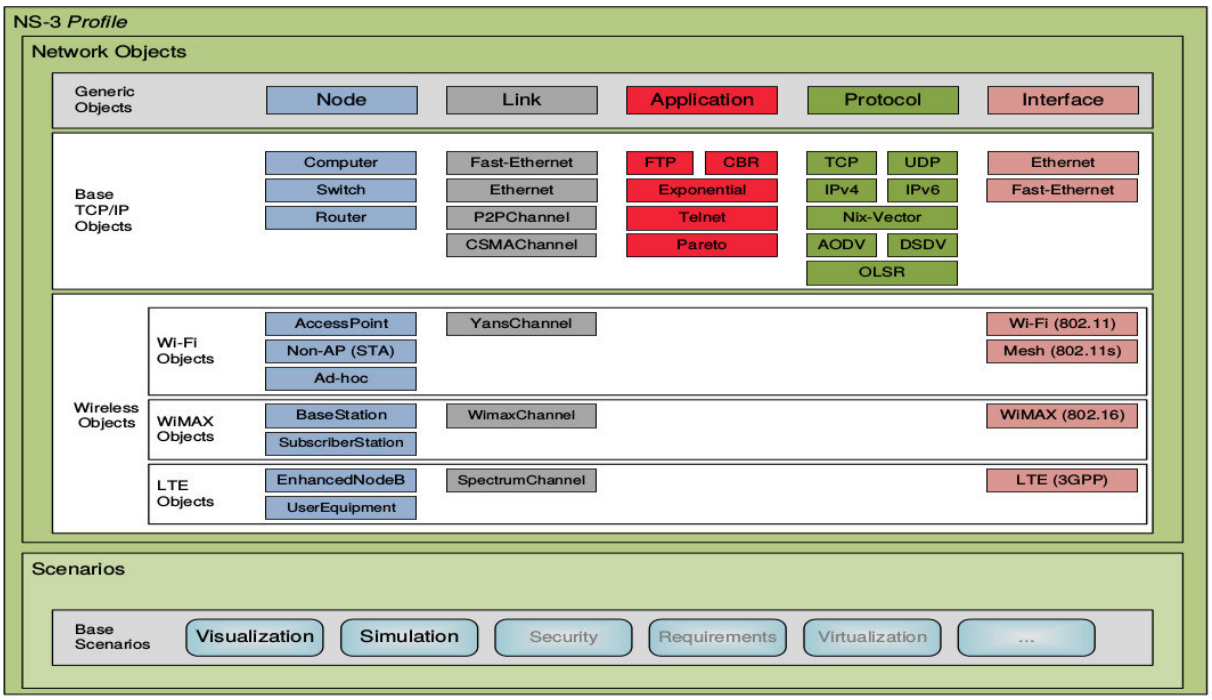

Figura 4.2: Extensão NSDL NS-3.

especificado correctamente (por exemplo o caso em que objectos previamente existentes necessitem de um atributo extra como é o caso dos ficheiros de output que necessitam de uma fonte de dados (<`source/`>)[Sousa, 2013].

De acordo com a Figura 4.2, o perfil do NS-3 é constituído em *Network Objects* pelo *Generic Objects* que são coincidentes com qualquer perfil que seja eventualmente desenvolvido para a NSDL, a Base TCP/IP *Objects* que teve as colunas de referentes às especificações de *Link e Protocol* extendidas, e a *Wireless Objects*, dividida em três ambientes tecnológicos distintos (*Wi-Fi, WiMax e LTE*).

Os elementos adicionados no perfil foram: no nível de enlace, *CSMAChannel*(<`link.csma/`>), de protocolo, *IPV6*(<`ipv6/`>), AODV - *Ad hoc On-Demand Distance Vector*(<`aodv/`>), *DSDV - Destination-Sequenced Distance Vector*(<`dsdv/`>), *Nix-Vector Routing*(<`nix.vector/`>), *OLSR - Optimized Link State Routing Protocol*(<`olsr/`>), dentre outros para a especificação *Wireless Objects*.

O autor propôs a aplicação do mapeamento NSDL / NS-3 seguindo o processo de levantamento de requisitos e análise, modelando alguns casos de utilização, e em seguida a codificação usando a arquitetura MVC (*Model-View-Controller*) e linguagem de *script* PHP (*Hypertext Preprocessor*). Além de estudo de caso com diversos cenários de redes para validar a abordagem do mapeamento NSDL / NS-3.

4.1.4 Wireless Ad Hoc Network Simulation in Cloud Environment [Djinevski et al.,]

Neste trabalho o autor serve-se do ambiente de computação em nuvem OpenStack [Sta, 2016] para executar simulações de redes sem fio *ad hoc*, levando em consideração os detalhes do terreno. Além disso, ele se propõe a desenvolver uma extensão 3D para o simulador de redes no NS-2, e uma extensão para execução de GPUs (*Graphics Processing Unit*) [Djinevski et al.,].

Os autores também apresentam os resultados das simulação de redes na nuvem privada. Adicionalmente comparam os resultados obtidos na simulação em nuvem com os resultados da simulação em um ambiente local, usando o mesmos cenários.

Após a definição da metodologia, implantação da infraestrutura de hardware e software, da execução dos cenários de simulação, e da obtenção dos resultados, o autor conclui que a simulação no ambiente de nuvem gasta um tempo maior para ser concluída.

4.1.5 Simulation Platform: A cloud-based online simulation environment [Yamazaki et al., 2011]

Para modelagem neural multi-escala e multimodal, é necessário manipular vários modelos neurais descritos perfeitamente em diferentes níveis.

Segundo o autor, é necessário desenvolver uma melhor maneira de validar um modelo de componentes, pois atualmente a modelagem neural manipula vários modelos neurais descritos perfeitamente em diferentes níveis. Nesse processo é necessário baixar o modelo de um banco de dados, extrair, ler as instruções, compilar se escrito em uma linguagem de programação geral, tal como C, instalar um simulador neural apropriado caso seja um modelo escrito para um simulador como Gênesis, e só depois de todos esses passos é que será realizada a a simulação propriamente dita.

Neste trabalho, é definido projeto de desenvolvimento de serviço web baseado em nuvem para simulação *on-line* chamado “Plataforma de Simulação”. Essa plataforma é constituída por uma nuvem de máquinas virtuais executando o sistema operacional GNU/Linux, nas quais vários programas, incluindo ferramentas de desenvolvimento como compiladores e bibliotecas, simuladores neurais como GENESIS, NEURONN and NEST

e *software* científico como Gnuplot, são pré-instalados [Yamazaki et al., 2011].

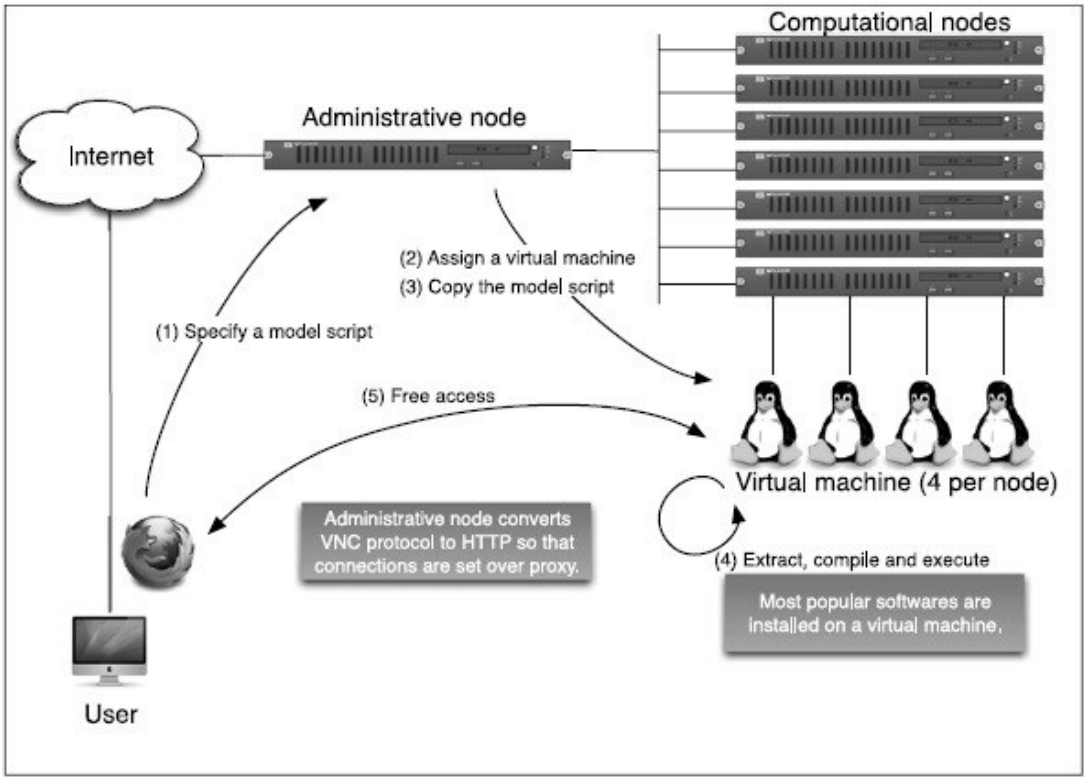

Figura 4.3: Plataforma de simulação [Yamazaki et al., 2011].

A plataforma definida conforme a Figura 4.3 possibilita ao usuário acessar remotamente a máquina por meio de um navegador web para que ele possa enviar solicitação para o nó administrativo, a fim de que ele possa iniciar uma simulação de um modelo, sem que seja necessária a instalação de qualquer software, apenas um navegador web em seu computador. Após o recebimento da solicitação, o nó administrativo verifica a disponibilidade das máquinas no *pool* de máquinas virtuais, caso todas estejam ocupadas, é retornada mensagem de erro, senão, é feita a cópia do arquivo de script de simulação para a máquina escolhida e em seguida o processo de simulação é iniciado. Posteriormente é informada URL do arquivo resultado para o usuário, e a sessão é encerrada.

Assim, a plataforma de simulação se propõe a se preparar todo um ambiente essencial para a execução de simulações.

4.1.6 Uma Proposta de Portal de Aplicação do NS-3 para Aprendizagem de Redes de Computadores [Maciel et al., 2014]

Neste artigo é apresentada uma proposta de portal de aplicação do NS-3 para o ambiente de aprendizagem de redes de computadores, permitindo interação do estudante com uma interface web amigável para submissão de simulações em um ciberambiente computacional composto por infraestruturas heterogêneas como *clusters* e nuvens. Além do Portal, também é apresentada arquitetura que permite a interação do portal com o ciberambiente.

O autor apresenta a vantagem de utilização de simuladores de redes em oposição ao uso de redes reais, devido ao alto custo dessas redes, tais como custo com montagem e instalação, configuração e manutenção da infraestrutura. Em compensação, aponta um esforço maior por parte dos estudantes devido à necessidade de aprendizado sobre o funcionamento do simulador.

Além do esforço citado, os estudantes necessitam recursos computacionais apropriados para a realização das simulações, tais como: memória, processamento e armazenamento e etc. E dependendo do cenário de simulação definido, a execução da simulação em um computador comum pode requerer um tempo maior para sua conclusão [Maciel et al., 2014].

Algumas instituições de ensino disponibilizam acesso a ambientes de alto poder computacional, como grades computacionais, supercomputadores (*clusters*)e nuvens computacionais, como é o caso do Centro Nacional de Processamento de Alto Desempenho da Universidade Federal do Ceará (CENAPAD-UFC). Com o ambiente disponibilizado, o usuário terá que conhecer sobre esse ambiente para que ele possa submeter uma execução a um escalonador de recursos de um *cluster*, por exemplo. Ou seja, ele deverá conhecer os comandos e detalhes de funcionamento de cada um dos ambientes.

Objetivando superar tais problemas, os estudantes podem usar o portal de aplicação, desenvolvido com foco em um dos módulos do NS-3 denominado *mesh*, que é capaz de estimular e facilitar a interação do estudante com o NS-3. Assim, é possível diminuir o esforço em detalhes do ferramental tecnológico e aumentar a concentração do estudante no objeto de estudo [Bastos and Gomes, 2012].

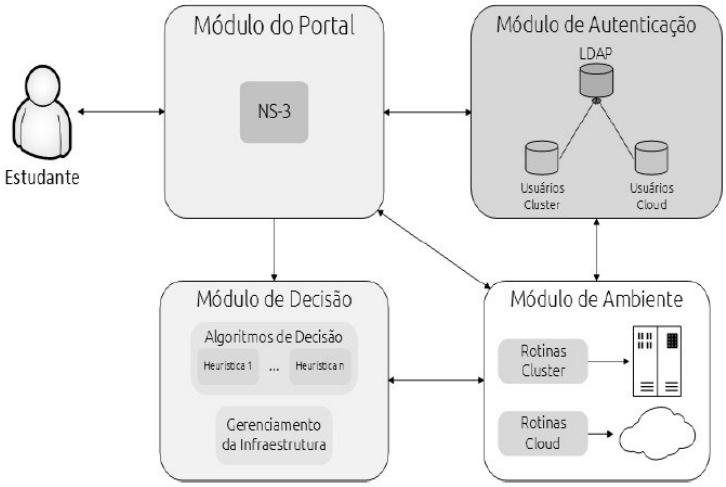

Figura 4.4: Arquitetura do ciberambiente de suporte ao portal de aplicação [Maciel et al., 2014].

A arquitetura proposta é constituída pelos módulos do portal, autenticação, decisão e ambiente, conforme ilustra a Figura 4.4. O módulo do portal tem como objetivo fornecer ao estudante uma interface web amigável que facilita a acesso ao ambiente do NS-3; o de autenticação é responsável por implementar a autenticação do estudante ao portal de aplicação; o de decisão é encarregado de decidir em que ambiente computacional a simulação será executada, com base nas necessidades dos usuários e nas características da infraestrutura computacional; por fim, o módulo de ambiente, que é composto de diferentes infraestrutura de computação de alto desempenho, é incumbido de executar as submissões de simulações.

Em seguida, os autores se utilizam de algumas definições complementares, tais como: Redes em Malha Sem Fio *(Wireless Mesh Networks - WMN)*, topologia em grade e topologia em disco aleatório que são objetos de estudo na implementação das simulações. Além disso, são apresentadas as interfaces web com seus respectivos parâmetros de simulação para as respectivas topologias, conforme mostrado na Figura 4.5.

De acordo com os autores, o uso de portais científicos particularizados por aplicação oferece vantagens aos processos de ensino e aprendizagem, pois os portais propiciam uma maior produtividade, em virtude de não exigir um gasto extra de tempo com o processo de instalação e configuração da aplicação. Dessa forma, não é necessário alocar recursos próprios para execução da aplicação, já que o processamento é feito em ambientes de *cloud* ou *clusters*.

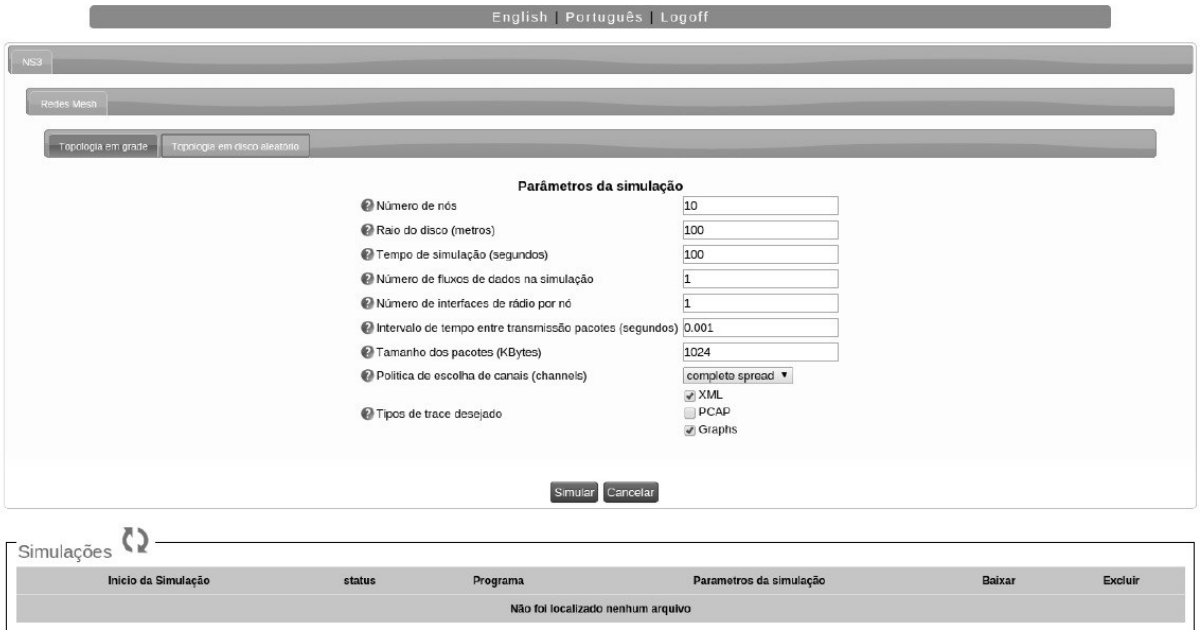

Figura 4.5: Tela de submissão de simulação.

4.2 Considerações Finais

Neste capítulo foram apresentadas algumas abordagens encontradas na literatura, voltadas para a descrição de linguagens de simulação e na disponibilização de um ambiente de simulação em nuvem. A tese sobre "Interoperabilidade e Otimização da Gestão de Redes com a *framework NSDL*" [Marques, 2014], sugeriu a implementação da linguagem NSDL e de um arcabouço visando a interoperabilidade entre ferramentas de simulação. Além disso, desenvolveu o conceito de perfil para a linguagem NSDL, validando-a por meio de um experimento de simulação específico para os simuladores NS-2 e NS-3.

A dissertação sobre "Autoria e Simulação de Cenários de Redes em NS-3" [Sousa, 2013], baseada no trabalho anterior [Marques, 2014], recomendou novas extensões para o perfil do simulador NS-3, como por exemplo extensões para as tecnologias Wi-Fi, WiMax (*Worldwide Interoperability for Microwave Access*) e LTE (*Long Term Evolution*).

Os dois trabalhos serviram de base no processo de definição de como compor arquivo XML com os objetos de simulação. No trabalho *"Wireless Ad Hoc Network Simulation in Cloud Environment"* os autores utilizam o ambiente de computação em nuvem para executar simulações de redes sem fio em uma nuvem privada.

O artigo *"Simulation Platform: A cloud-based online simulation environment"* implementa um modelo arquitetural composto por um conjunto de nós que distribuem a requisição de simulações para o seu *pool* de máquinas. Entretanto, o escopo das simulações realizadas é diferente da proposta nesta disssertação.

No artigo *"Uma Proposta de Portal de Aplicação do NS-3 para Aprendizagem de Redes de Computadores"*, o foco é direcionado a redes mesh, redes em malha sem fio (*Wireless Mesh Networks - WMN*). Não é apresentado em sem modelo arquitetural a forma de como a simulação é distribuída na nuvem para execução.

5 Plataforma de Simulação em Nuvem - NSCloud

Este capítulo tem como objetivo apresentar a plataforma de simulação em nuvem NSCloud, que foi desenvolvida para suprir a necessidade de utilização de simulações para projetar e analisar as redes de computadores, utilizando os recursos computacionais providos pela nuvem, que possui características peculiares para o desenvolvimento deste trabalho. Esta plataforma é baseada no simulador NS-3 e constituída por três módulos: o módulo da aplicação, que foi desenvolvido baseada na arquitetura REST (*Representational State Transfer*), o módulo que define o registro XML para compor os parâmetros da simulação, e o módulo do modelo arquitetural que é composto por máquinas virtuais e serviços disponíveis na nuvem da Amazon.

5.1 Serviços Web de Simulação

Segundo o W3C (*World Wide Web Consortium*), os serviços web, são aplicações autocontidas, que possuem interface baseada em XML e que descrevem uma coleção de operações acessíveis a partir da rede, independentemente da tecnologia usada na implementação do serviço [W3C, 2015]. Por meio desta tecnologia é possível promover a interoperabilidade entre aplicações que tenham sido desenvolvidas em diferentes plataformas, tornando-as compatíveis e permitindo trocas de dados entre as aplicações por meio da linguagem XML. E por propiciar a interoperabilidade, a tecnologia de serviços web foi utilizada no desenvolvimento da aplicação que visa a integração com diferentes ferramentas no processo de simulação.

5.1.1 Tecnologias de Desenvolvimento Web Utilizadas

No âmbito de desenvolvimento web, algumas tecnologias possuem as especificações necessárias à concretização deste projeto. Dentre essas ferramentas, merece destaque as seguintes:

- XML (*Extended Markup Language – Linguagem de Marcação Estendida)* – é um subconjunto da SGML (*Standard Generalized Markup Language – Linguagem de Marcação Padrão Generalizada)* originalmente concebida pelo W3C(*World Wide Web Consortium*) para enfrentar os desafios das publicações eletrônicas em larga escala, desempenhando um papel cada vez mais importante na troca de uma ampla variedade de dados na web e em outras plataformas [Consortium, 2015a]. Essa especificação determina regras para a criação de documentos genéricos e estruturados que podem acrescentar informação semântica ao texto por meio de marcações *(tags)*, facilitando a manipulação, pesquisa e extração de dados. Neste trabalho a linguagem XML foi utilizada na criação de arquivo XML com os parâmetros de simulação baseado no simulador NS-3;

- XSLT (*Extensible Stylesheet Language Transformation – Transformação em Linguagem de Folhas de Estilos Extensível*) – é uma linguagem para transformar diferentes formatos de documentos XML [Consortium, 2015b]. Uma transformação descrita em XSLT decreve regras para transformar uma árvore fonte em uma árvore resultado XML. Essa transformação é possível devido a associação de padrões e modelos. Um padrão é comparado com elementos da árvore fonte, e um modelo é instanciado para a geração da árvore resultado. Neste trabalho utilizamos XSLT para transformar o arquivo XML, que contém os parametros de simulação, em um arquivo C++ contendo o *script* NS-3 a ser simulado;

- REST (*Representational State Transfer – Transferência de Estado Representacional*) – termo definido por Roy Felding em sua tese de doutorado no qual ele descreve sobre um estilo arquitetural de *software* sobre um sistema operado em redes [Fielding, 2000].

- Linguagem JSP (*Java Server Pages*) – linguagem de *script* que tem como objetivo a criação de conteúdo dinâmico para páginas web. AJAX (*Asynchronous Javascript and XML*) – corresponde a um conjunto de técnicas para programação e desenvolvimento web que utiliza tecnologias como Javascript e XML para carregar informações de forma assíncrona de páginas web [Garrett, 2005].

5.1.2 Modelagem do Serviço

Após o levantamendo de requisitos, foi possível modelar algumas visões do sistema a ser desenvolvido utilizando a linguagem UML (*Unified Modeling Language*), linguagem padrão para visualizar, especificar, construir e documentar os artefatos de software [UML, 2015]. O primeiro diagrama a ser modelado foi um diagrama de caso de uso, objetivando a representação das funcionalidades externamente observáveis do sistema e dos elementos externos que interagem com o mesmo, conforme é ilustrado na Figura 5.1.

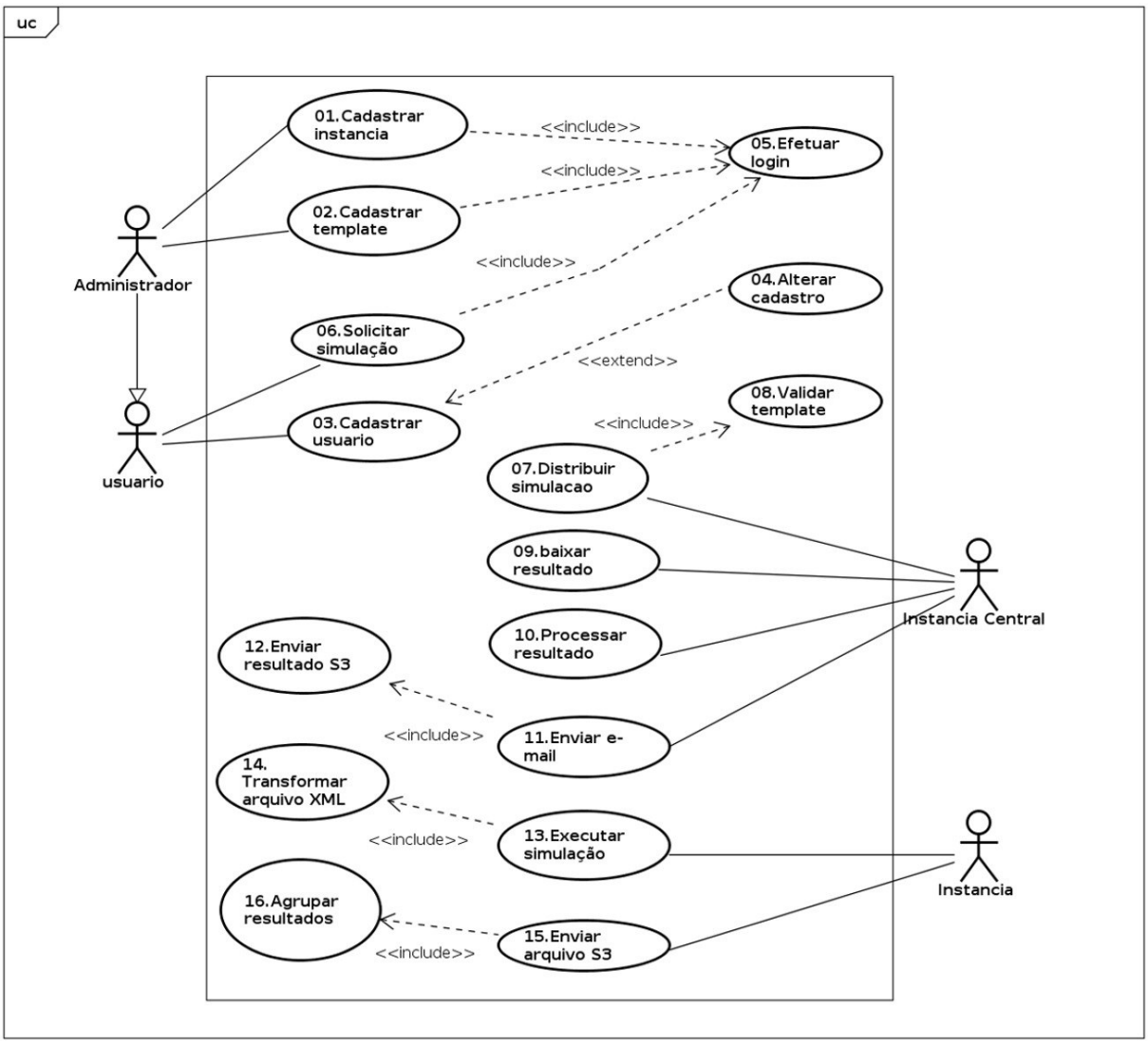

Figura 5.1: Diagrama de caso de uso do serviço NSCloud.

A seguir serão listados os casos de uso do sistema, além de uma breve descrição sobre os mesmos:

- (CSU01) Cadastrar instância: possibilita o cadastro do endereço IP (*Internet Protocol*) e porta da máquina em que o serviço estará disponível.

- (CSU02) Cadastrar *template*: viabiliza o cadastro de novos modelos.
- (CSU03) Cadastrar usuário: efetua o cadastro do usuário e atribui a ele um perfil.
- (CSU04) Alterar cadastro do usuário: alterar os dados do cadastro do usuário.
- (CSU05) Efetuar login: autenticação do usuário para acessar o ambiente.
- (CSU06) Solicitar simulação: permite ao usuário requerer a simulação, indicando o arquivo XML com os parâmetros da simulação, quantidade de simulações por instância e descrição da simulação.
- (CSU07) Distribuir simulação: possibilita o envio do arquivo XML com os parâmetros de simulação para as instâncias.
- (CSU08) Validar *template*: responsável por verificar a existência de modelos de transformação para a simulação requerida.
- (CSU09) Baixar resultado: permite a realização do *download* dos arquivos contendo os resultados parciais de cada instância.
- (CSU10) Processar resultado: efetua o cálculo da média, desvio padrão e índice de confiança.
- (CSU11) Enviar e-mail: possibilita o envio de e-mail contendo *link* com o resultado final da simulação.
- (CSU12) Enviar resultado S3: envia o arquivo contendo o resultado final para o repositório no S3.
- (CSU13) Executar simulação: permite a execução do *script* de simulação pelo simulador NS-3.
- (CSU14) Transformar arquivo XML: efetuar a transformação do arquivo XML com os parâmetros de simulação em *script* de simulação.
- (CSU15) Enviar arquivo S3: proporciona o envio de arquivo contendo o resultado parcial para o repositório no S3.
- (CSU16) Agrupar resultados: responsável por realizar a fusão dos resultados das simulações, adicionando-os em um único arquivo.

O ator com perfil de administrador poderá efetuar o cadastro de modelos de transformação e de instâncias. Além disso, ele pode interagir com os casos de uso que o usuário comum interage, desde que ele esteja devidamente autenticado no sistema. O usuário poderá efetuar o seu cadastro, alterar o cadastro e solicitar simulação, desde que esteja autenticado no sistema.

O ator instância central poderá distribuir a simulação para as instâncias, depois de validar o modelo, obter os resultados parciais do repositório no S3, processar os resultados e enviar um e-mail ao usuário indicando o término da simulação por intermédio do SES. Por outro lado, o ator Instância executa a simulação requerida, transforma arquivo XML em arquivo de *script* de simulação para o simulador NS-3, processa os resultados e envia-os para o repositório no S3.

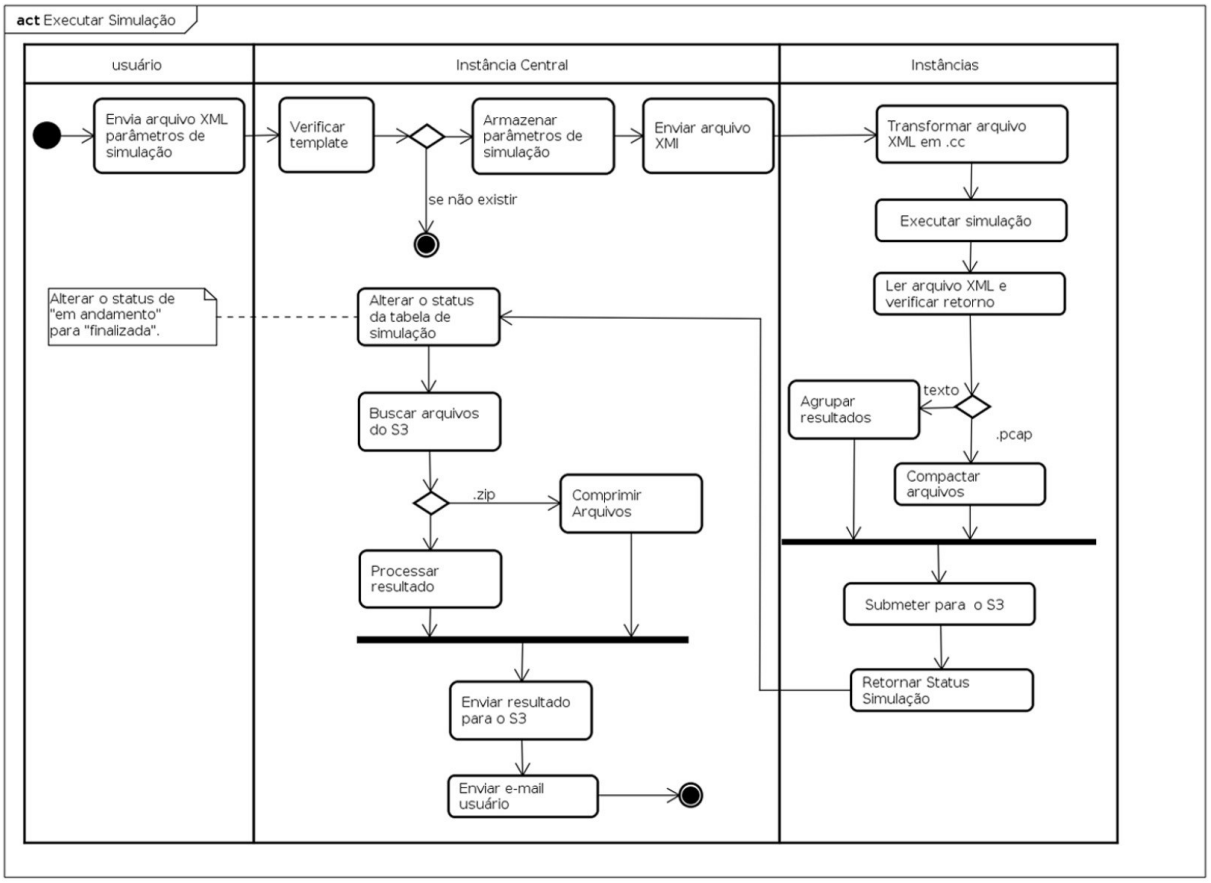

Figura 5.2: Diagrama de atividade - executar simulação.

Posteriormente, foi feita modelagem do diagrama de atividade, que é um tipo especial de diagrama de estado em que são representados os estados de uma atividade em vez dos estados de um objeto. Ao contrário dos diagramas de estados, que são orientados a eventos, diagramas de atividade são orientados a a fluxo de controle [Bezerra, 2015].

Na elaboração do diagrama de atividade do fluxo de execução da simulação, foram identificadas as seguintes *swim lanes* (raias de natação): *Usuário, Instância Central e Instâncias*, conforme apresentado na Figura 5.2. Cada raia é constituída pelas atividades que serão realizadas por um agente específico.

Inicialmente o usuário solicita a simulação por meio da atividade *"Enviar arquivos XML parâmetros de simulação"*. Em seguida, a *Instância Central* irá verifica se existe um arquivo de transformação correspondente à simulação desejada por meio da atividade *"Verificar Template"*. Caso não exista, o fluxo segue para o estado final, caso contrário, segue para a próxima atividade *"Enviar arquivo XML"* que também é executada pela *Instância Central*.

As *Instâncias* irão receber o arquivo XML e transformá-los em um *script* de simulação durante a tarefa *"Transformar arquivo XML em .cc"*. Posteriormente a simulação será executada por meio da tarefa *"Executar simulação"*.

A tarefa seguinte *"Ler Arquivo XML e verificar retorno"* verifica o tipo de resultado desejado pelo usuário. Se for do tipo *texto*, a tarefa *"Agrupar resultados"* será executada, ficando incumbida de anexar o resultado de todas as simulações em um único arquivo. Por outro lado, se o resultado for do tipo *.pcap*, os resultados deverão ser compactados na tarefa *"Compactar Arquivos"*.

Após o agrupamento dos resultados ou a compactação dos arquivos, os resultados dessas tarefas deverão submetidos ao repositório no S3 na tarefa *"Submeter para o S3"*. Posteriormente é retornado o *status* informando o término da simulação pela *Instância* para a *Instância Central*, por meio da tarefa *"Retornar Status Simulação"*. A *Instância Central* faz a alteração do status da simulação em *"Alterar o status da tabela de simulação"* e em seguida efetua o *download* dos arquivos resultado em *"Buscar arquivos do S3"* para agrupa-los em um único arquivo, e efetuar o cálculo da média, desvio padrão e intervalo de confiança na tarefa *"Processar resultado"* ou compactar os arquivos já compactados pelas suas respectivas instâncias na tarefa "*Comprimir Arquivos*. Logo após, o resultado final é enviado ao repositório no S3 e um e-mail é enviado ao usuário requisitante, indicando o término da simulação e um *link* para acessá-lo no repositório.

5.1.3 Implementação do Serviço

A aplicação foi dividida em dois módulos: o do cliente e o do serviço. O primeiro foi desenvolvido com o arcabouço *BootStrap* e a linguagem JSP (*Java Server Page*), gerando as interfaces de interação com o usuário para os requisitos levantados, e o modelo de casos de uso proposto. Esse módulo permite a interação com os serviços por meio das seguintes interfaces:

- *Efetuar autenticação – interface* que proporciona a validação do usuário no sistema através da inclusão do nome e senha. Caso o mesmo não esteja cadastrado, é possível efetuar o cadastro por meio do *link "Create new user"* que irá redirecionar o usuário para a tela de cadastro.

- *Registrar usuário* – tela em que serão fornecidos os dados para os campos *user, password, e e-mail*. Além da escolha do perfil, que pode ser de *administrator* ou *user*.

- *Registrar instância – interface* que possibilita a inclusão do caminho do recurso URI(*Uniform Resource Identifier*) em nova máquina a ser inserida na plataforma. Nesse processo somente o usuário com perfil de administrador poderá fazer o cadastro, desde que sejam informados os campos, *name*, *address* IP(*Internet Protocol*) e *port*.

- *Registrar* template – tela que propicia a inclusão de novos modelos de transformação XSLT(*Extensible Stylesheet Language Transformation*). É possível a inclusão apenas por quem possui o perfil de administrador.

- *Requerer simulação – interface* que permite submeter as simulações desejadas, listar as já realizadas, adicionar instâncias e templates. Na submissão é necessário informar o número de simulações que serão realizadas por instância (*Number of Simulation to Instance*), a descrição da simulação (*Simulation Description*) além do arquivo XML com os parâmetros de simulação desejados. A lista de simulações realizadas é composta pelo identificador da simulação (*ID Simulation*), a descrição da simulação (*Description*), *link* em que se encontra o resultado da simulação caso a mesma seja concluída, e a situação da simulação *(status)* que indica se a mesma já foi concluída, ou se ainda está sendo processada, conforme ilusta a Figura 5.3.

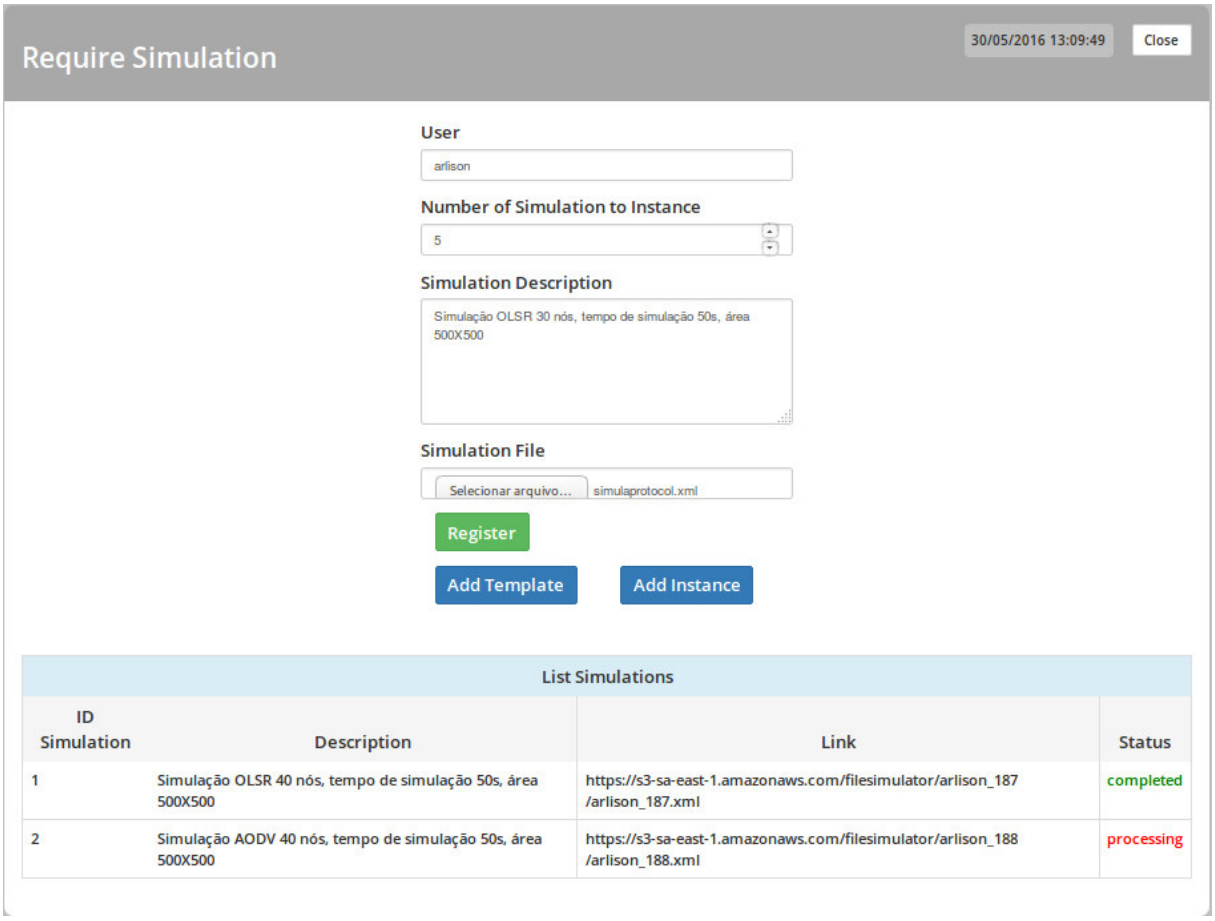

Figura 5.3: Executar simulação.

Na Figura 5.3 é ilustrada a interface para requerer as simulações, além de permitir a inclusão de modelos e instâncias, nessas duas últimas funcionalidades, é necessário que o usuário esteja cadastrado com o pefil de *administrator* para que ele possa executá-las.

Módulo do Serviço

Para o desenvolvimento do módulo foi utilizado o ambiente de desenvolvimento integrado Netbeans 8.0, o container Apache Tomcat 8.0, o SGBD MySql 5.5 e máquina virtual Java 1.8, todas ferramentas *open source*. Ele é composto pelos recursos `usuario`, `instancia`, `template`, `simulacao` e `arquivos`, que serviram-se das seguintes tecnologias no processo de desenvolvimento: XML, XSD, XSLT, Jersey RESTful e a linguagem Java. Além do padrão de projeto DAO (*Data Access Object*) que foi empregado, visando separar as regras de negócio das regras de acesso ao banco de dados.

O recurso `usuario` se propõe a efetuar o cadastro do usuário, o recurso

`instancia` é encarregado de realizar o cadastro de novas instâncias que por ventura forem lançadas na arquitetura, o recurso `template` se dispõe a fazer o cadastro de novos modelos, o recurso `simulação` é responsável pela solicitação de simulações para as instâncias e o recurso `arquivos` é encarregado de executar as simulações requeridas. Os principais `simulacao` e `arquivos` serão descritos a seguir.

O recurso `simulação` da listagem 5.1 implementado na classe `CentralRS` é composto por dois métodos principais, o método `requererSimulacao` e o método `simulacaoAsincronaCompletionCallback`. O método `requererSimulacao` recebe via POST os parâmetros `strusuario, idlogin, strqtdsimula, strdescricao` e `estrnamefile` do formulário (linhas 8 a 13), e posteriormente o arquivo de simulação é enviado para os endereços das instâncias armazenadas no banco de dados (linhas 19 a 25). Em seguida os parâmetros recebidos são armazenados no banco de dados (linha 31), e é feita de forma assíncrona uma requisição de simulação para cada instância armazenada por meio do método `simulacaoAsincronaCompletionCallback` (linhas 36 a 41), logo após é redirecionado para o fórmulário de simulação no cliente.

No método `simulacaoAsincronaCompletionCallback` após completada a simulação (linha 65), é feito o *download* dos resultados de cada instância no `bucket filesimulator` do repositório no S3 (linha 66) e efetuado os cálculos da média, desvio padrão e intervalo de confiança, e em seguida o resultado final é convertido em XML e submetido para o repositório no S3 (linha 68), conforme ilustrado na Figura A.3, posteriormente o *link* contendo a *URL* do resultado é armazenada no banco e um *e-mail* de confirmação de conclusão é enviado ao usuário, com *link* e URL do caminho em que estão localizados os resultados (linhas 72 e 73).

```
1  @Path("/simulacao")
2  public class CentralRs {
3
4    @POST
5    @Produces(MediaType.TEXT_HTML)
6    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
7
8    public void requererSimulacao(@FormParam("strusuario")
9      String strusuario,
10     @FormParam("idlogin") String idLogin,
11     @FormParam("strqtdsimula") String strqtdsimula,
12     @FormParam("strdescricao") String strDescricao,
13     @FormParam("strnamefile") String strNameFile,
14
15     @Context HttpServletResponse servletResponse) throws IOException{
16           .
17           .
18           .
19      for(int count = 0; count < listInstancia.size(); count++) {
20         strIp = listInstancia.get(count).getIP();
```

```
21          executa = Runtime.getRuntime().exec("scp -i /home/repos/
22          chave.pem -r /home/repos/ns-3-allinone/ns-3-dev/
23          scratch/strNameFile ec2-user@" +strIp+":/home/repos/
24          ns-3-allinone/ns-3-dev/scratch");
25        }
26
27        if(executa.waitFor() == 0) {
28
29          Simulacao simulacao = new Simulacao(Integer.parseInt
30             (strqtdsimula), 0, Integer.parseInt(idLogin), strDescricao);
31          simulacaoDao.adiciona(simulacao);
32          Usuario usuario = new Usuario(Integer.parseInt(idLogin),
33                          strusuario);
34          simulacao.setId(simulacaoDao.buscaIdSimula());
35
36          for(int count = 0; count < listInstancia.size(); count++) {
37            SimulaInstancia simulacaoInstancia = new SimulaInstancia(
38            listInstancia.get(count), simulacao, usuario, strNameFile);
39            simulacaoAsincronaCompletionCallback(simulacaoInstancia,
40            strusuario,count+1,simulacao.getId(),simulacao.getQtdSimula());
41           }
42
43           servletResponse.sendRedirect("http://54.94.239.252:8080/
44          wsclients3/formsimulation.jsp?strusuario=" + strusuario +
45          "&idLogin=" + idLogin);
46        }
47    }
48
49    private void simulacaoAsincronaCompletionCallback(SimulaInstancia
50      simulacaoInstancia, String strUsuario, int idInstancia,
51      int idSimulacao,int qtdSimulacao) {
52                      .
53                      .
54                      .
55      AsyncInvoker async = client.target(strTarget).request().async();
56      Future<Response> future;
57      future = async.get(
58
59         new InvocationCallback<Response>() {
60           @Override
61           public void completed(Response response) {
62
63             simulacaoInstancia.setResultado(response.readEntity(
64                                              String.class));
65             simulacaoDao.altera(simulacaoInstancia.getSimulacao());
66             String stringResult = downloadToFileResult(strUsuario,
67             idSimulacao);
68             String urlLink = sendToFileSimulator(stringResult,
69                              strUsuario,idSimulacao);
70
71               if(!"".equals(urlLink)){
72                 simInstanciaDao.adiciona(simulacaoInstancia, urlLink);
73                 SendMail(urlLink);
74               }
75           }
76           @Override
77           public void failed(Throwable throwable) {
78                 throwable.getStackTrace();
79           }
80         }
81     );
82  }
```

Listing 5.1: Trecho de código do recurso simulação, classe `CentralRS.java`.

O recurso `arquivos` é composto pela classe `InstanceRS` que possui o

método `simulaArquivo` que recebe via GET os parâmetros `strusuario`, `strarquivo`, `idinstancia`, `idsimulacao` e `qtdsimulacao` (linhas 9 a 17) da instância central `(CentralRS)` por meio do método `SimulacaoAsincronaCompletionCallback`. Esse método é encarregado de transformar o arquivo XML em um *script* de simulação (linhas 21 a 32), verificar se o arquivo de simulação existe, e em seguida executar as simulações de acordo com a quantidade definida pelo usuário utilizando uma semente aleatória para cada simulação, de acordo com o tempo em milisegundos (linhas 59 a 82). Por fim o método, verifica o tipo de retorno desejado no arquivo XML e agrupa os resultados em um único arquivo. Por fim, o sistema os envia ao `bucket filesimulator` do repositório de simulações no S3 (linhas 84 a 98).

```
1
2  @Path("/arquivos")
3
4  public class InstanceRS {
5
6   private final ExecutorService executorService = java.util.concurrent.
7                       Executors.newCachedThreadPool();
8
9    @GET
10   @Produces(value = MediaType.TEXT_PLAIN)
11   public void simulaArquivo(
12      @Suspended final AsyncResponse asyncResponse,
13      @QueryParam(value = "strusuario") final String strusuario,
14      @QueryParam(value = "strarquivo") final String strarquivo,
15      @QueryParam(value = "idinstancia") final int idinstancia,
16      @QueryParam(value = "idsimulacao") final int idsimulacao,
17      @QueryParam(value = "qtdsimulacao") final int qtdsimulacao)
18                                  .
19                                  .
20                                  .
21      transform(dataXML, inputXSL, outputCC);
22      String fileOut = getFile("/home/repos/ns-3-allinone/ns-3-dev/
23                                scratch/simulaprotocol.cc");
24      fileOut = fileOut.replace("<?xml version=\"1.0\"
25                      encoding=\"UTF-8\"?>", "")
26                                      .replace("&lt;", "<")
27                                      .replace("&gt;", ">")
28                                      .replace("&amp;", "&");
29
30
31       fileReplace("/home/repos/ns-3-allinone/ns-3-dev/scratch/
32                           simulaprotocol.cc",fileOut);
33
34    executorService.submit(() -> {
35        asyncResponse.resume(doSimulaArquivo(strusuario, strarquivo,
36                        idinstancia, idsimulacao, qtdsimulacao));
37   });
38  }
39
40
41   public String doSimulaArquivo(
42     @QueryParam(value = "strusuario") final String strusuario,
43     @QueryParam(value = "strarquivo") final String strarquivo,
44     @QueryParam(value = "idinstancia") final int idinstancia,
45     @QueryParam(value = "idsimulacao") final int idsimulacao,
46     @QueryParam(value = "qtdsimulacao")  final  int qtdsimula)
47                          throws JDOMException, IOException
48   {
```

```
49
50    File file = new File("/home/repos/ns-3-allinone/ns-3-dev/scratch/" +
51                                 strarquivo);
52    String typeResult = getTypeSimulaton(strarquivo);
53    String userIdSimula = strusuario+"_"+idsimulacao;
54    String caminho = file.getName();
55    String[] nome_arquivo = caminho.split("\\.");
56
57    if(file.exists()){
58
59      File novo = new File("/home/repos/ns-3-allinone/ns-3-dev/");
60      Process executa = null;
61
62      Calendar calendar = Calendar.getInstance();
63      long seg = calendar.getTimeInMillis();
64
65      for(int i = 0; i < qtdsimula; i++){
66
67        try {
68          Calendar ca = Calendar.getInstance();
69          long mili = ca.getTimeInMillis();
70
71          executa = Runtime.getRuntime().exec(new String[] {"/bin/bash",
72          "-c", "sudo ./waf --command-template='%s --RngRun=" + (mili + i)
73          + "' --run " + nome_arquivo[0] + "> result"+i+".rst >&1"},
74          null, novo);
75
76        }catch (IOException ex) {}
77
78           try {
79             if(executa.waitFor() == 0) {
80          }
81        }catch (InterruptedException ex) {}
82      }
83
84      if(typeResult.equals("csv")){
85
86         String pathFileResultMix = MixFile(getFileType("rst"),
87                                 userIdSimula);
88
89         sendToFileSimulator(fileResultMix,strusuario,idinstancia,
90                                 idsimulacao);
91
92         return "simulacao em arquivo csv concluida!";
93
94      }else{
95         compactFiles(nome_arquivo[0],"rst");
96
97         sendToFileSimulator("arquivo",strusuario,idinstancia,idsimulacao);
98
99         return nome_arquivo[0] + ".zip";
100     }
101
102   }
103
104   return "";
105 }
```

O `bucket filesimulator` é composto por pastas que são intituladas de acordo com a seguinte nomenclatura: usuário e *id* da simulação, respectivamente. Internamente são listados os arquivos contendo os resultados da simualção de cada instância, nomeados da seguinte forma: usuário, *id* da simulação, *id* da instância, dia, mes, ano e segundos

e a extensão *".rst"*. O arquivo com o resultado final segue o mesmo critério de nome dado para as pastas do repositório, conforme apresentado no Apêndice A.2. O código completo dos recursos `simulação` e `arquivos` foram disponibilizados nos Apêndices A.4 e A.5, respectivamente.

5.2 Documentação das Simulações

O arquivo XML se propõe a descrever vários cenários de redes com tecnologias diferentes, possibilitando a modelagem de simulações. Sua composição foi baseada em estudo realizado por Eduardo Marques [Marques, 2014] em sua tese de doutorado sobre o estudo das linguagens de descrição de redes existentes, e comparativo entre elas. A partir deste estudo foi possível identificar características relevantes como simplicidade das descrições e possibilidade de inclusão de novos elementos. Essas características foram levadas em consideração no desenvolvimento do arquivo de simulação baseado no NS-3 (*Network Simulator 3*).

5.2.1 Estrutura do Documento

A definição do registro de arquivo de simulação também tomou como referência os conceitos e abstrações usados por profissionais da área de redes e empregados pelo simulador NS-3.

O elemento <`simulation`> integra os objetos que são comumente usados em redes, como o <`node`>, o <`channel`>, o <`protocol`>, o <`interface`> e o <`application`>, além dos elementos <`description`>, <`nodeGroup`> e <`result`> definidos no desenvolvimento da linguagem, conforme a a estrutura desses elementos, apresentada na Figura 5.4. Os elementos serão detalhados a seguir:

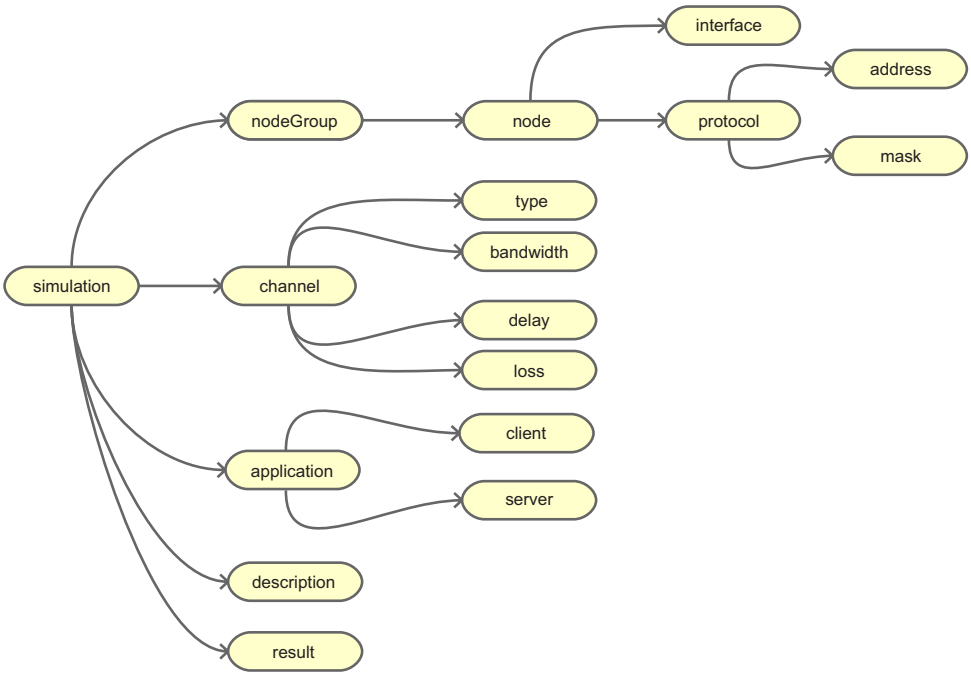

Figura 5.4: Estrutura base de registro XML para a descrição de simulações.

- <node> – representa os equipamentos ativos em redes de comunicação. Na definição da linguagem estão incluídos em um objeto <nodeGroup> que tem como propósito o agrupamento de todos os nós da rede. O <node> também contém dois outros objetos, o <interface> que corresponde ao canal de comunicação (na ausência de canal físico) entre os nós, e o <protocol> que é encarregado por definir o protocolo, a faixa de endereçamento IP (*Internet Protocol*) e a máscara de sub-rede usados na simulação.

- <channel> – corresponde ao canal de comunicação entre dois nós, sendo usado para adicionar as interfaces de rede, responsáveis pela transferência das mensagens entre os entre os nós. Ele possui o objeto <type> que é encarregado de descrever o tipo de tecnologia empregada no canal. Além disso, ele possui ainda três elementos: <bandwidth>, <delay> e <loss> que permitem indicar algumas características básicas de um canal: a capacidade de transmissão, o atraso, e a taxa de perdas.

- <application> – é encarregado por gerar o tráfego que é transmitido na rede, como por exemplo o FTP *(File Transfer Protocol)* e o Telnet. Além de receber os pacotes perdidos e/ou descartados.

- <description> - contém um texto descritivo sobre a simulação pretendida.

- <result> - compreende o tipo de resultado proposto pela simulação, ou seja, se o resultado pretendido será composto por arquivos do tipo .pcap (tipo de arquivo usado pela ferramenta Wireshark para analisar os protocolos), ou arquivo do tipo texto.

Todos os elementos possuem os atributos *@id* e *@name*. No entanto, existem alguns atributos que são específicos para cada elemento. Na Tabela 5.2.1 estão dispostos os elementos com seus respectivos atributos.

Tabela 5.1: Atributos específicos dos elementos do arquivo.

Elementos	Atributos		
simulation	type	totaltime	
channel	type	src	dest
nodeGroup	numberNodes		
node	applicationNode		
result	type		
protocol	type		

Em redes do tipo Wifi, Wimax e LTE, inseridas no elemento (<type>) devem ser incluídos novos elementos internos aos tipos citados, como por exemplo em redes MANETS é necessária a inclusão de um elemento que indique a quantidade de nós ativos (<activeNodes>), elementos relacionados à velocidade do padrão de mobilidade (<nodeSpeed>) e (<nodePause>), relacionados à transmissão de dados <dataRate>, contendo o modo correspondente (<phyMode>) e o elemento (<dimension>) contendo os elementos (<minimum>) e (<maximum>).

A partir dos elementos citados, documentos XML podem ser criados contendo os parâmetros da simulação desejada que serão posteriormente transformados em *scripts* de simulação a serem executados pelo simulador NS-3, conforme apresentado no Apêndice B.1. Para a transformação dos documentos serão usados modelos de transformação XSLT para as respectivas simulações, de acordo com o Apêndice B.2.

5.3 Modelo Arquitetural do NSCloud

Nesta seção é apresentada a arquitetura da Plataforma NSCloud implantada na nuvem da Amazon, que é composta inicialmente por quatro máquinas virtuais, com o sistema operacional GNU/Linux, a distribuição Amazon Linux AMI, um serviço responsável pelo armazenamento dos arquivos contendo os resultados das simulações, o Amazon S3 (*Simple Storage Service*) e um serviço incumbido do envio de e-mail, o Amazon SES (*Email Simple Service*), conforme ilustra a Figura 5.5. Eventualmente, novas instâncias (terminologia empregada pela Amazon para a inclusão de máquinas na nuvem) podem ser lançadas para compor a plataforma.

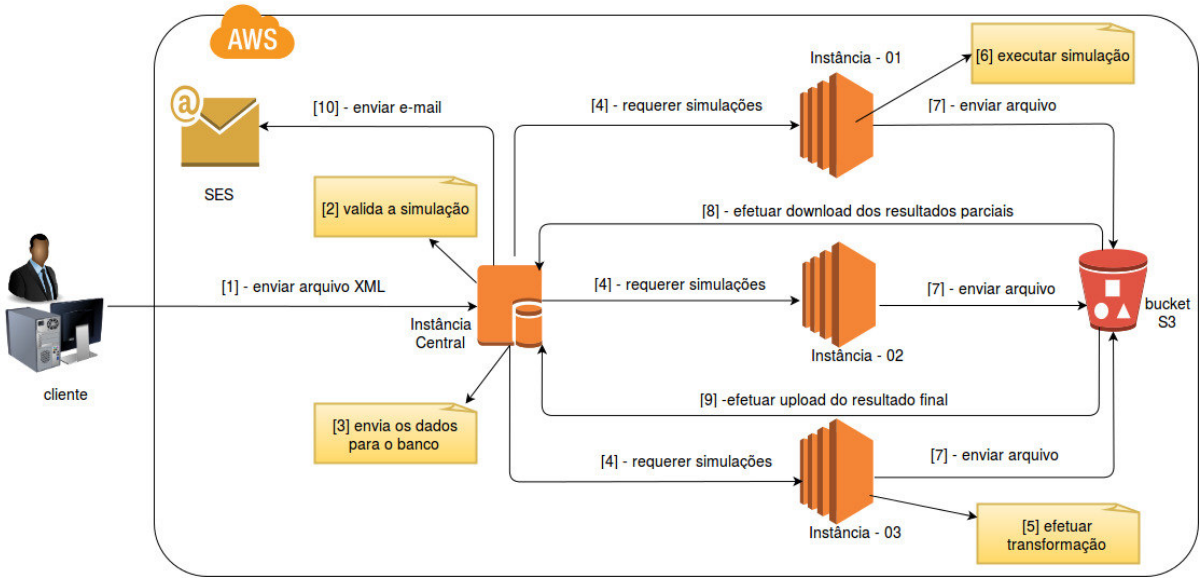

Figura 5.5: Modelo Arquitetural da Plataforma de Simulação NSCloud.

Os procedimentos de funcionamento da plataforma segue uma ordem de numeração, conforme ilustra a Figura 5.5. Os detalhes de cada numeração são listados em seguida:

1. O usuário prepara o arquivo XML com os parâmetros da simulação e o envia por meio do *frontend* para a instância central;
2. A instância central valida o arquivo XML, verificando a existência de modelo de transformação correspondente;
3. Os parâmetro da simulação são enviados para a base de dados pela instância central;
4. O arquivo XML é replicado para as demais instâncias e a simulação é requerida;

5. As instâncias efetuam a transformação do arquivo XML em script de simulação C++, por meio da linguagem XSLT;

6. As simulações são executadas pelas instâncias;

7. Os arquivos contendo os resultados das simulações são agrupados em um arquivo e enviados para um repositório no S3;

8. A instância central efetua o *download* dos resultados parciais de cada instância, agrupa-os em um único arquivo, e efetua o cálculo da média, desvio padrão e intervalo de confiança, e em seguida o arquivo com o resultado é transformado em arquivo XML;

9. O upload do arquivo XML contendo o resultado final para o repositório é efetuado pela instância central;

10. A instância central envia e-mail contendo o link para o resultado no repositório para o usuário que solicitou a execução da simulação.

Caso o resultado da simulação especificada no arquivo XML, for do tipo *".pcap"*, os passos a partir do sétimo são alterados para:

1. Os arquivos contendo os resultados das simulações são compactados em um único arquivo e enviados para um repositório no S3;

2. A instância central efetua o *download* dos arquivos compactados de cada instância, efetua a compactação dos mesmos em um único arquivo;

3. É efetuado o upload do arquivo compactado para o repositório no S3 pela instância central;

4. A instância central envia e-mail contendo o link para o resultado no repositório para o usuário que solicitou a execução da simulação.

A escolha das configurações das máquinas para compor a plataforma se dá de acordo com os tipos ofertados pela Amazon, que as classifica como sendo de uso geral, otimizadas para computação e otimizadas para memória, que se adequam a diversos casos de uso. Esses tipos consistem em várias combinações de CPU (*Central Processing Unit*),

memória, tecnologia de armazenamento e capacidade de rede. Há um custo diferenciado dependendo da região em que a máquina será lançada [AWS, 2015b].

O tipo de instância selecionada para execução das simulações na arquitetura NSCloud foi a `c4.xlarge` que possui um processador personalizado Intel Xeon E5-2666 v3, otimizado especificamente para o EC2, além de outros elementos de *hardware* descritos na Tabela 5.3. E para a instância central da arquitetura o tipo de instância escolhida foi a `t2.micro` que possui uma especificação de *hardware* menor do que a instância `c4.xlarge`, mas atende perfeitamente o seu propósito de computação.

Tabela 5.2: Especificação máquinas virtuais.

Instância	c4.xlarge
CPU	Intel Xeon E5-2666 v3
RAM	7,5 GHz
SSD	8 GB
OS	Amazon Linux AMI
Kernel	3.13.48-33.39.amzn1.x86_64

Nas máquinas virtuais reponsáveis pela simulação, estão previamente instaladas ferramentas de desenvolvimento como compiladores, bibliotecas, container Apache Tomcat, simulador NS-3 e serviço web que tem por finalidade a execução das simulações solicitadas. Além das ferramentas citadas, a máquina central também possui um SGBD (Sistema de Gerenciamento de Banco de Dados) MySql instalado. Na Tabela 5.3 estão listados os softwares instalados na plataforma com suas respectivas versões.

Tabela 5.3: Especificação de *softwares* instalados.

	Nome	Versão
Container web	Apache Tomcat	8.0.23
Máquina Virtual	Amazon Linux AMI	2016.03
SGBD	MYSQL	5.5.49
Simulador	NS - 3	3.21
JDK	VM Java	1.8.0_05-b13

A inclusão de novas instâncias incubidas de executarem simulações pode ocorrer por meio da criação da imagem de instância já existente, caso em que não é

necessária a instalação e configuração de nenhuma ferramenta, ou lançamento de instância nova, caso em que é preciso instalar e configurar as ferramentas citadas na Tabela 5.3.

5.3.1 Levantamento de Custos

Nesta seção é realizada uma estimativa dos custos relativos ao uso dos componentes utilizados e dos que possivelmente possam ser incluídos na plataforma NSCloud, baseando-se na calculadora disponível pela Amazon.

Inicialmente são calculados os valores estimados para o uso de instâncias do tipo *t2.micro*, e em seguida, do tipo *t2.medium* que possuem características e custos diferentes, de acordo com a Tabela 5.3.1. Foi escolhida a região da AWS em que está disposta a plataforma NSCloud, que no caso é a da América do Sul, em São Paulo, e o sistema operacional Linux.

Tabela 5.4: Tabela de Avaliação de Custos de Serviços AWS na região da América do Sul (São Paulo) em Julho de 2016.

Quantidade	Descrição	vCPU	Memória *(GB)*	Uso(hora/mês)	Custo Mensal(US$)
1	Instância t2.micro	1	1	480	12,96
3	Instância t2.medium	2	4	480	155,52
3	Instância m3.large	2	7.5	480	273,60
3	Instância c4.large	2	3,7	480	233,28
3	Instância c4.xlarge	4	7,5	480	468,28
4	Elastic IP	—	—	480	14,64
1	S3	—	—	720	2,68
1	SES	—	—	720	2,20

A estimativa do custo total definido a partir da Tabela 5.3.1 é de US$ 188,00 mensal para o serviço de armazenamento de arquivos S3, com o padrão de armazenamento de *10GB*, quantidade de solicitações POST/PUT/COPY/LIST e GET, de 1000 solicitações cada, e com a transferência externa de dados de entrada e saída de 10GB cada. Além do serviço de envio de *e-mail* Amazon SES que permite o envio de 100 mensagens por dia, transferência de até 10GB de anexos por mês, transferência de dados de entrada e saída da conta para a Internet de até 10GB cada, uma instância `t2micro` e três instâncias `t2.medium`.

Para máquinas com um poder computacional maior como as `m3.large` o valor do custo mensal passa a ser de US$ 306,08, e para máquinas do tipo `c4.xlarge` escolhidas para compor a arquitetura da plataforma NSCloud, o valor mensal é de US$ 500,76. Esses tipos de instâncias além de se diferenciarem pela quantidade de CPU e memória, também utilizam tecnologias diferenteciadas de armazenamento de dados, que no caso são do tipo `SSD` (*solid-state drive*). Na região geográfica do Norte da Virgínia (EUA) o custo mensal com instancias `m3.large` é de US$ 220,67, e com máquinas virtuais `c4.xlarge` o valor mensal passa a ser de US$ 497,37.

5.4 Considerações Finais

A implementação da plataforma de simulação em nuvem é um projeto importante, pois mostrou-se eficiente no processo de distribuição, gerenciamento e inclusão de máquinas virtuais de acordo com a característica inerentes à computação em nuvem, possibilitando a inclusão e ou configuração das máquinas de acordo com as necessidades de recursos de computação para as simulações desejadas.

Além disso, o serviço web de simulação se mostra de grande relevância no contexto prático, uma vez que possibilita a execução de simulações de forma distribuída, automatizando o cálculo da média, do desvio padrão e do intervalo de confiança dos resultados obtidos. Além disso, a infraestrutura proposta disponibiliza os resultados das simulações realizadas em repositório, além de sinalizar para os usuários o fim das simulações por meio de um e-mail.

A plataforma NSCloud cumpriu com os seus objetivos inicias, conseguindo assim um ambiente completo para simulações e um repositório em nuvem com o simulador NS-3.

6 Validação da Plataforma NSCloud

Este capítulo tem por objetivo apresentar as estratégias utilizadas na validação da plataforma NSCloud e os resultados obtidos. O presente capítulo é segmentado em três seções. A seção dos experimentos realizados, a seção de resultados obtidos e a seção com as considerações finais sobre o capítulo.

6.1 Experimentos realizados

Um experimento pode somente mostrar existência de uma falha em uma teoria, mas não sua ausência. Isso significa que experimentos não devem ser vistos como provas, mas como evidências. Seus resultados são válidos para o conjunto o qual foi objeto de pesquisa [Travassos et al., 2002]. A idéia é utilizá-los para indução visando generalizar os resultados e derivar teorias a partir da observação [Tichy, 1998]. O tipo de experimento mais adequado vai depender dos objetivos do estudo, das propriedades do processo ou dos resultados finais esperados [Wohlin et al., 2000].

6.1.1 Cenário Redes *ad hoc*

No processo de avaliação e validação da Plataforma e serviços web implementados foram criados alguns cenários de redes ad hoc móveis, Mobile ad hoc Networks - MANET's, que foram criadas para suprir as necessidades de estabelecer conexões entre dispositivos móveis sem infraestrutura previamente definida. Uma MANET é um sistema autônomo composto por nós móveis que não dependem de nenhuma infra-estrutura para operar. Ou seja, os nós se comunicam sem nenhum ponto de acesso controlando o acesso ao meio, e para que isso seja possível, o nó de destino deverá estar no alcance de transmissão do nó de origem, ou algum outro nó intermediário possa encaminhar o pacote para o nó de destino. Assim, os pacotes são encaminhados por nós intermediários, até que atinjam seu destino. Essa comunicação entre os dispositivos se dá através dos protocolos de roteamento, que podem ser classificados em três categorias: proativo, reativo e híbrido. Os protocolos proativos trocam informações

da rede periodicamente a fim de se manter rotas para os demais nós da rede. Os protocolos reativos estabelecem o caminho para o destino somente quando requisitados. Já o modelo híbrido, corresponde à uma combinação dos dois modelos de protocolos anteriormente descritos.

Os protocolos escolhidos para execução das simulações foram o protocolo reativo *Ad hoc Ondemand Distance Vector (AODV)* e o próativo *Optimized Link State Routing (OLSR).*

O *AODV* é um protocolo que possui o procedimento de descoberta de rotas baseado na origem, sendo comumente utilizado em trabalhos que envolvem MANETs. Quando um nó deseja enviar dados para algum destino para o qual ele não possua rota válida, inicia-se um processo de descoberta do caminho. O nó de origem envia para seus vizinhos uma mensagem route request *(RREQ)* e estes, por sua vez, as enviam aos seus vizinhos e assim sucessivamente até que se chegue ao nó desejado ou a algum nó que possua uma rota recente para o nó de destino. Este protocolo utiliza o número de sequência de destino (destination sequence number) para ter a certeza de que todas as rotas estão livres de laços *(loop free)* e que elas possuam as mais recentes informações sobre o nó de destino [Perkins et al., 2003].

O OLSR *Optimized Link State Routing* é um protocolo baseado no algoritmo de estado de enlace *(link state)* [Kurose and Ross, 2010], orientado a tabelas de roteamento, que usa uma otimização chamada *Relay - MPR* para a troca de informações de topologia com outros nós da rede, sendo destinado a redes de alta escalabilidade.

Os cenários empregados no experimento são compostos pelos protocolos acima citados, e as seguintes méticas: *Packet Loss Ratio*, *Average Endtoend Delay* definidas como:

- *Packet Loss Ratio (PLR)*: É calculada pela divisão do total de pacotes perdidos multiplicado por cem e dividido pelo número de pacotes gerados, descrevendo assim a taxa de perda dos pacotes. Sendo está métrica um ótimo indicador da competência do protocolo de roteamento.

- *Average Endtoend Delay*: Calculado pelo atraso de cada pacote recebido, incluído todo atraso possível, como buffer usado no processo de descoberta de rotas, filas de processamento nas interfaces, tempos de propagação e de transferência. Seu

resultado final é dado em segundos, sendo utilizada a expressão seguinte para o cálculo: Delaysum / rxPacketsum;

No estudo efetuado, usando o simulador NS-3, foram definidos os seguintes parâmetros: modelo de mobilidade, área de simulação, quantidade de nós, modelo de tráfefo, tempo de simulação, velocidade mínima, velocidade máxima, rádio range fixo, taxa de geração de pacotes, fontes ativas, quantidade de execuções e nível de confiança conforme a Tabela 6.1. E na obtenção dos resultados das simulações, foi empregado o módulo *FlowMonitor*, módulo do NS-3 utilizado no monitoramento que permite análisar os resultados obtidos. O modelo de mobilidade escolhido foi o Random Waypoint, que tem sido a base para a maioria das avaliações de MANETs [PalChaudhuri et al., 2005]. Neste modelo os nós móveis selecionam um destino em uma área retanguar de forma aleatória e se desloca até ele. Ao chegar ao destino o nó pode fazer uma pausa por um tempo aleatório, e em seguida ele escolhe uma nova direção e reinicia todo o processo.

Tabela 6.1: Resumo dos Parâmetros da Simulação.

Parâmetros da Simulação	Valor
Modelo de mobilidade	Random Waypoint
Área de simulação	500m x 500m
Quantidade de nós	20, 30 e 40
Modelo de tráfego	CBR
Tempo de simulação	50s
Velocidade mínima	5 m/s
Velocidade máxima	15 m/s
Rádio range fixo	250m
Taxa de geração de pacotes por fonte	4 pacotes por segundo
Fontes ativas	20% do nº total de nós
Camada MAC	802.11 com 11 Mbps
Quantidade de execuções	15
Nível de confiança	95%

6.2 Resultados Obtidos

Para análise dos resultados obtidos, relativamente às métricas em estudo, foram construídos gráficos, cujos resultados levam em consideração o intervalo de confiança de 95%. Tais resultados estão dispostos nas tabelas no Apêndice B.3.

6.2.1 Análise da Perda de Pacotes - Packet Loss Ratio

O *packet loss ratio* é uma métrica que nos permite fazer uma análise sobre como reage a rede a determinado tráfego. Está métrica é importante, tanto para o protocolo UDP *(User Datagram Protoco)* como para o TCP *(Transmission Control Protocol)*, pois no primeiro, um pacote perdido não é retransmitido, e no segundo, quanto maior for o *packet loss ratio*, maior será o número de retransmissões, diminuíndo o *throughput*. A primeira métrica avaliada foi a perda de pacotes que foi avaliada em porcentagem. No Apêndice B.3 são apresentados os resultados das simulações de acordo com os respectivos nós. E a partir dos resultados, é gerado um gráfico comparativo contendo a taxa de perda de pacotes de cada nó, conforme ilusta a figura 6.1.

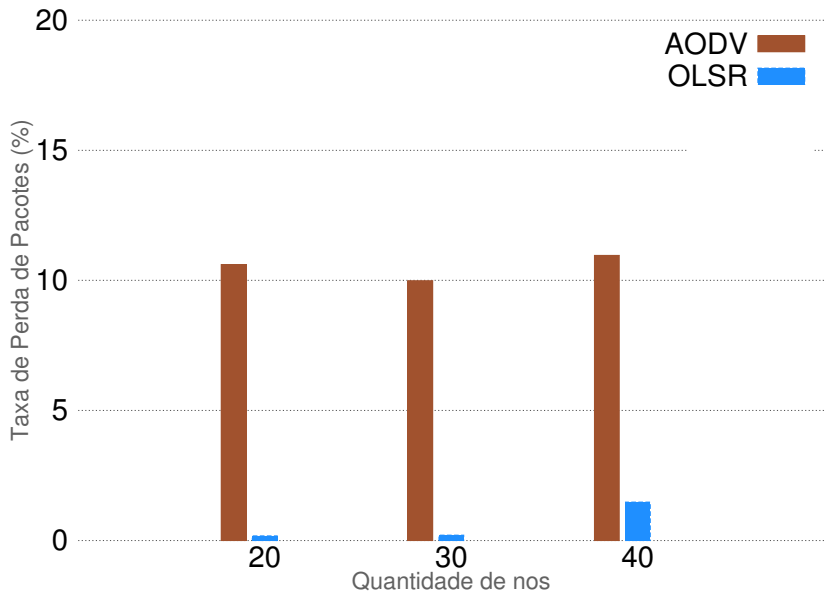

Figura 6.1: Gráfico comparativo da taxa de perda de pacotes dos protocolos AODV e OLSR.

A partir dos dados coletados, observa-se, de maneira geral, que a perda de pacotes é acentuada no protocolo AODV. Com a inclusão de 40 nós a porcentagem de perda média encontrada a partir dos 6714 pacotes enviados e dos 736 pacotes perdidos

para este protocolo é de aproximadamente 10%. Além dessa perda, também existe uma média dos pacotes excluídos que é de 290. Enquanto que o protocolo OLSR apresenta uma perda média a partir de 1486 pacotes enviados, e 22 pacotes perdidos de 1,5%, não havendo pacotes excluídos.

6.2.2 Average End-to-End Delay

Com base nos dados reunidos, é possível constatar que o atraso é menor no protocolo OLSR, e relação ao AODV. Isso se ratifica pela própria natureza pró-ativa do protocolo OLSR, que é oritentado a tabelas de roteamento. Enquanto que o AODV define suas rotas quando necessário, isto é, este protocolo gasta um tempo maior na tomada de decisão.

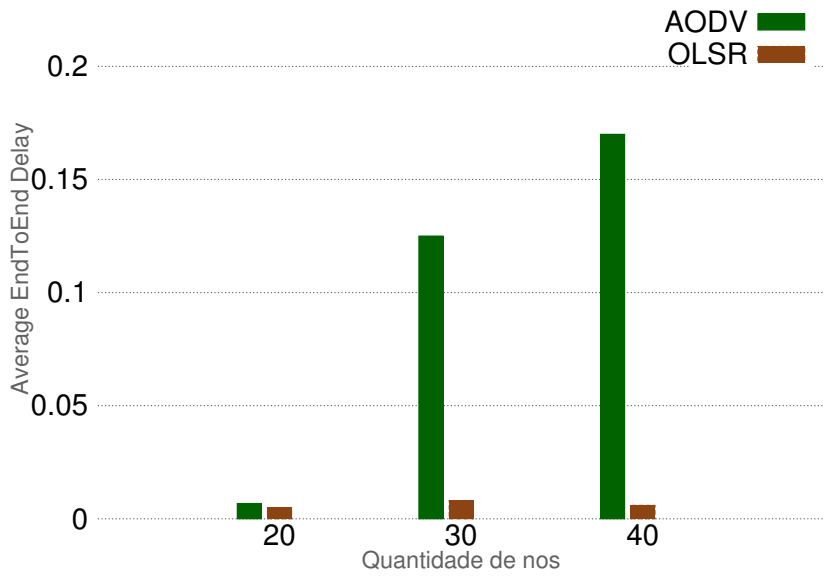

Figura 6.2: Gráfico comparativo da taxa de perda de pacotes dos protocolos AODV e OLSR.

Utilizando 30 nós conectados o tempo médio de atraso (Average End-to-End Delay) do protocolo OLSR apresentou melhores resultados, com um resultado de 0,087 segundos contra os 0,125 segundos do AODV. E com o aumento de nós na rede é possível perceber o aumento do atraso para os dois protocolos, entretanto, para o protocolo AODV esse aumento é mais expressivo, devido as suas especificidades.

6.3 Considerações Finais

A análise dos resultados da seção anterior é claramente favorável à Plataforma *NSCloud* e serviço web de simulação. Pois possibilita a distribuição de simulações entre máquinas, cálculo da média, desvio padrão e intervalo de confiança de acordo com os parâmetros de simulações pretendidos.

7 Conclusão

O presente trabalho teve como objetivo a implementação de plataforma de simulação em nuvem para execução de simulações de redes de computadores. Objetivando solucionar a problemática voltada para a necessidade de grande quantidade de recursos computacionais para execução de simulações computacionais. A plataforma de simulação implementada, denominada NSCloud se propôs a utilizar os recursos computacionais da nuvem para a execução de simulações de redes de computadores baseada no simulador NS-3. E na composição da plataforma, foi incorporado serviço web para efetuar a distribuição, processamento e armazenamento dos resultados das simulações.

Para atingir o objetivo proposto fez-se inicialmente uma revisão da literatura sobre as definições, modelos e características da computação em nuvem. Além de estudo sobre simulação computacional e simuladores. A partir desse estudo foi definido o simulador NS-3 como ferramenta de simulação de redes de computadores da plataforma.

A plataforma, assim como o serviço web foram desenvolvidos de modo a permitir a adição de novas máquinas, propiciando uma maior escalabilidade na execução das simulações.

Dado o seu funcionamento e do serviço web de simulações baseado no simulador NS-3 (*Network Simulator 3*), com os resultados obtidos e a partir de estudo de caso apresentado no capítulo anterior, pode-se validar a plataforma NSCloud, e afirmar que o objetivo principal do processo foi alcançado com sucesso. Em fim, acredita-se que esta dissertação contribuiu na resolução da problemática de necessidade de recursos computacionais para a execução de simulações, provendo a plataforma NSCloud como ferramenta para auxiliar no processo de execução de simulações.

7.1 Contribuições

As principais contribuições desta dissertação estão relacionadas com a implementação de plataforma de simulação, conforme apresentado no capítulo 5, entre as quais podemos citar: a criação de repositório centralizado contendo os resultados das

simulações, e o desenvolvimento de um serviço web de simulação baseado no simulador NS-3 que possibilitou a distribuição das simulações em diversas máquinas, permitindo a execução das simulações em paralelo, além de automatizar o processo de cálculo da média, desvio padrão e intervalo de confiança de um conjunto de simulações executadas.

7.2 Trabalhos Futuros

Pesquisas futuras podem aproveitar a plataforma de simulação para experimentar simulações em outras configurações de *hardware*, regiões e zonas de disponibilidade acessíveis na nuvem da Amazon. Além disso, permite a inclusão de novos serviços disponíveis a fim de que a plataforma possa proporcionar mais benefícios para o usuário.

Contudo, durante o desenvolvimento da aplicação, percebeu-se a necessidade de termos uma linguagem que permita descrever os mais diversos cenários de simulação, fazendo com que o serviço de simulação se torne independente do simulador que será utilizado para executar os experimentos e que irá submeter os parâmetros da simulação desejada. Além disso, uma nova funcionalidade poderia ser acrescentada, objetivando a plotagem de gráficos a partir do arquivo resultado gerado.

Por outro lado, mesmo com a definição sobre os diversos tipos de máquinas existentes na nuvem da Amazon, surgiu um questionamento sobre os recursos computacionais ideais para compor a plataforma. Nesse caso é pertinente estudo a fim de que os recursos sejam providos de modo eficaz sem comprometer a qualidade do serviço contratado.

Referências Bibliográficas

[AWS, 2015a] (2015a). Amazon web services. https://aws.amazon.com/pt/about-aws/. Acessado: 2015-04.

[AWS, 2015b] (2015b). Amazon web services. https://aws.amazon.com/pt/ec2/pricing/. Acessado: 2015-04.

[Gar, 2015] (2015). Critical capabilities for public cloud infrastructure as a service, worldwide. `https://www.gartner.com/doc/reprints?id=1-2QQX6UM&ct=151027&st=sb`. Acessado: 2015-11.

[Ama, 2015] (2015). Detalhes do produto amazon ebs. https://aws.amazon.com/pt/ebs/details/. Acessado: 2015-08.

[UML, 2015] (2015). Introduction to omgś unified modeling language. http://www.uml.org/what-is-uml.htm. Acessado: 2015-10.

[nsn, 2015a] (2015a). Network simulator - 3. `http://www.nsnam.org`. Acessado: 2015-04.

[nsn, 2015b] (2015b). Network simulator - 3. `https://www.nsnam.org/docs/release/3.25/manual/html/organization.html`. Acessado: 2015-04.

[OMN, 2015] (2015). Omnet++ simulation manual. https://omnetpp.org/doc/omnetpp/manual/. Acessado: 2015-12.

[CSA, 2015] (2015). Security guidance for critical areas of focus in cloud computing. `http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf`. Acessado: 2015-06-15.

[W3C, 2015] (2015). Web services architecture, w3c working group note 11 february 2004. https://www.w3.org/TR/ws-arch/. Acessado: 2015-05.

[Sta, 2016] (2016). Open source software for creating private and public clouds. Acessado: 2016-01.

[Abramson et al., 1999] Abramson, D., Sosic, R., and Power, K. (1999). Simulating computer networks using clusters of pcs. In *HPCTelePar'99 at the 1999 Advanced Simulation Technologies Conference (ASTC'99), April 11-15*.

[Bastos and Gomes, 2012] Bastos, B. F. and Gomes, A. T. A. (2012). Uma ferramenta para prototipagem rápida de portais científicos em grades computacionais. *Salão de Ferramentas do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 896–903.

[Berkeley and ISI, 1998] Berkeley, U. and ISI, U. (1998). The network simulator ns-2. *Part of the VIN T project. Available from http://www. isi. edu/nsnam/ns*.

[Bezerra, 2015] Bezerra, E. (2015). *Princípios De Análise E Projeto De Sistemas Com Uml*, volume 3. Elsevier Brasil.

[Brantner et al., 2008] Brantner, M., Florescu, D., Graf, D., Kossmann, D., and Kraska, T. (2008). Building a database on s3. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 251–264. ACM.

[Chang, 1999] Chang, X. (1999). Research on mobile cloud computing: Review,trend and perspectives. *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future*, pages 1–8.

[Chang, 2012] Chang, X. (2012). Network simulation tools survey. *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 1(4):1–8.

[Consortium, 2015a] Consortium, W. W. W. (2015a). Web services architecture. Acessado: 2015-05.

[Consortium, 2015b] Consortium, W. W. W. (2015b). Xsl transformations (xslt). Acessado: 2015-05.

[Coutinho, 2013] Coutinho, E. (2013). Elasticidade em computação na nuvem: Uma abordagem sistematica. *SBRC*, pages 215–258.

[Djinevski et al.,] Djinevski, L., Filiposka, S., Mishkovski, I., and Trajanov, D. Wireless ad hoc network simulation in cloud environment.

[Du et al., 2013] Du, J., Ao, F., Qin, F., Zhou, Y., and Huang, C. (2013). Cluster performance oriented characteristics of network simulations. In *10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2013*, pages 653–657.

[Ehrlich, 1991] Ehrlich, P. J. (1991). *Pesquisa operacional: curso introdutório*. Atlas.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

[Garrett, 2005] Garrett, J. J. (2005). Ajax: A new approach to web applications.

[Harrell et al., 2000] Harrell, C., Bowden, R., and Ghosh, B. K. (2000). *Simulation using promodel*. McGraw-Hill Higher Education.

[Hollman and Marti, 2003] Hollman, J. and Marti, J. (2003). Real time network simulation with pc-cluster. *IEEE Transactions on Power Systems*, 18(2):563–569.

[Kelton, 2003] Kelton, S. (2003). Sturrock: Simulation with arena®.

[Kurose and Ross, 2010] Kurose, J. F. and Ross, K. W. (2010). Redes de computadores e a internet (preferencialmente a 5ª edição).

[Lacage and Henderson, 2006] Lacage, M. and Henderson, T. R. (2006). Yet another network simulator. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 12. ACM.

[Lecheta, 2014] Lecheta, R. R. (2014). *AWS para Desenvolvedores: Aprenda a instalar aplicações na nuvem da Amazon AWS*. Novatec Editora.

[Maciel et al., 2014] Maciel, F. A. O., de Oliveira, C. T., Neto, R. C., Andrade, R. M., and Bezerra, R. L. (2014). Uma proposta de portal de aplicação do ns-3 para aprendizagem de redes de computadores.

[Marques, 2014] Marques, E. M. D. (2014). *Interoperabilidade e Otimizaccão da Gestão de Redes com a Framework NSDL*. PhD thesis, Universidade da Madeira.

[Marti et al., 2000] Marti, J. R., Hollman, J., and Calvino-Fraga (2000). Implementation of a real-time distributed network simulator with pc-cluster. In *International Conference on Parallel Computing in Electrical Engineering, 2000. PARELEC 2000*, pages 223–227.

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The nist definition of cloud computing. *Computer Networks*, 51(7):1–7.

[Minkovich et al., 2014] Minkovich, K., Thibeault, C., O'Brien, M., Nogin, A., Cho, Y., and Srinivasa, N. (2014). Hrlsim: A high performance spiking neural network simulator for gpgpu clusters. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):316–331.

[Networks, 2009] Networks, F. (2009). Cloud computing survey. pages 1–11.

[PalChaudhuri et al., 2005] PalChaudhuri, S., Le Boudec, J.-Y., and Vojnovic, M. (2005). Perfect simulations for random trip mobility models. In *Proceedings of the 38th annual Symposium on Simulation*, pages 72–79. IEEE Computer Society.

[Perkins et al., 2003] Perkins, C., Belding-Royer, E., and Das, S. (2003). Ad hoc on-demand distance vector (aodv) routing. Technical report.

[Sousa, 2013] Sousa, J. J. F. (2013). Autoria e simulação de cenários de redes em ns-3. Master's thesis.

[Taurion, 2009] Taurion, C. (2009). Cloud computing: computação em nuvem: transformando o mundo da tecnologia da informação. *Rio de Janeiro: Brasport*, 2(2):2–2.

[Tichy, 1998] Tichy, W. F. (1998). Should computer scientists experiment more? *Ieee Computer*, 31(5):32–40.

[Travassos et al., 2002] Travassos, G. H., Gurov, D., and Amaral, E. (2002). *Introdução à engenharia de software experimental*. UFRJ.

[Vaquero et al., 2008] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.

[Vuckovik et al., 2010] Vuckovik, M., Trajanov, D., and Filiposka, S. (2010). Durkin's propagation model based on triangular irregular network terrain. In *International Conference on ICT Innovations*, pages 333–341. Springer.

[Wohlin et al., 2000] Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., and Wesslen, A. (2000). Experimentation in software engineering: an introduction. 2000.

[Yamazaki et al., 2011] Yamazaki, T., Ikeno, H., Okumura, Y., Satoh, S., Kamiyama, Y., Hirata, Y., Inagaki, K., Ishihara, A., Kannon, T., and Usui, S. (2011). Simulation platform: A cloud-based online simulation environment. *Neural Networks*, 24(7):693–698.

A Apêndice

A.1 Interface do Amazon EC2 Contendo as Máquinas da Arquitetura

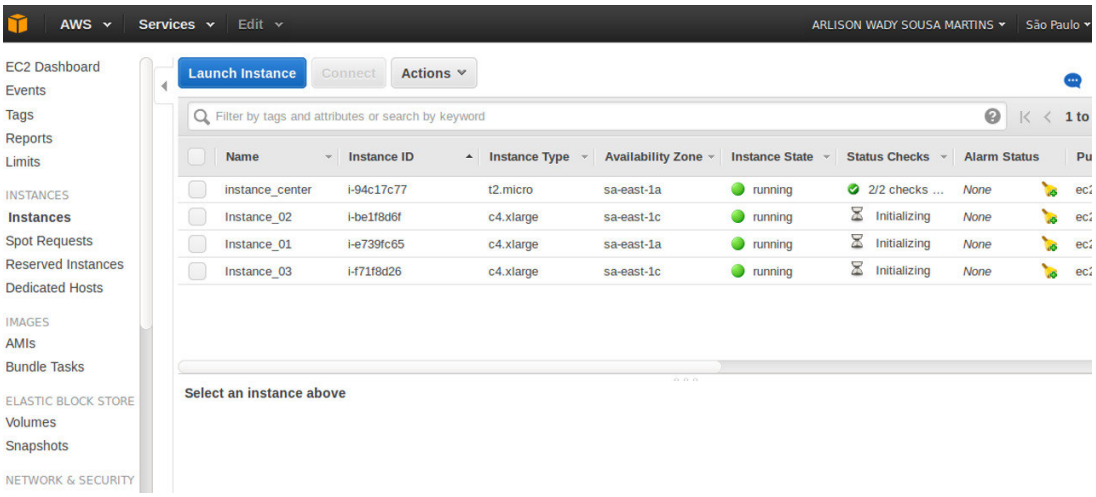

Figura A.1: Instâncias da plataforma NSCloud lançadas na Nuvem

A.2 Interface da Amazon S3 Contendo Repositório de Simulações

Figura A.2: Repositório de Simulações

A.3 Interface contendo resultado de simulação no repositório da Amazon S3

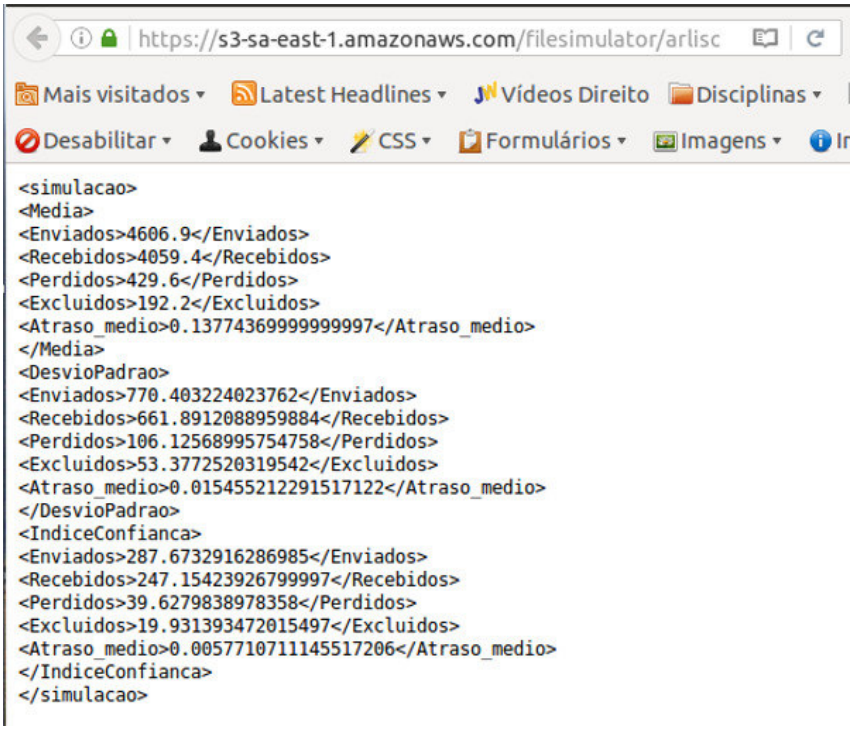

Figura A.3: Arquivo XML de resultado de simulação

A.4 Implementação do Recurso Simulacao

```
1
2  @Path("/simulacao")
3  public class CentralRs {
4
5   @Context
6   UriInfo uriInfo;
7   @Context
8   Request request;
9
10  Connection conn = null;
11  SimulacaoDao simulacaoDao = null;
12  SimulaInstanciaDao  simInstanciaDao = null;
13  InstanciaDao instanciaDao = null;
14
15  public CentralRs(){
16     conn =  new ConnectionFactory().getConnection();
17     simulacaoDao = new  SimulacaoDao();
18     simInstanciaDao = new SimulaInstanciaDao();
19     instanciaDao = new InstanciaDao();
20  }
21
22  @POST
23  @Produces(MediaType.TEXT_HTML)
24  @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
25  public void requererSimulacao(@FormParam("strusuario")String strusuario,
26    @FormParam("idlogin") String idLogin,
```

```
27      @FormParam("strqtdsimula") String strqtdsimula,
28      @FormParam("strdescricao") String strDescricao,
29      @FormParam("strnamefile") String strNameFile,
30
31      @Context HttpServletResponse servletResponse) throws
32      IOException{
33
34      String xslType = getFileSimulator(strNameFile);
35      if (xslType.equals(""))
36          servletResponse.sendRedirect("");
37
38      Process executa = null;
39
40      List<Instancia> listInstancia = instanciaDao.getListaInstancia();
41
42      String strIp = null;
43
44      try {
45
46        for(int count = 0; count < listInstancia.size(); count++) {
47          strIp = vetStr[1].substring(2);
48          strIp = listInstancia.get(count).getIP();
49          executa = Runtime.getRuntime().exec("scp -i /home/repos/
50          cloudmestrado.pem -r /home/repos/ns-3-allinone/ns-3-dev/
51          scratch/simulaprotocol.xml ec2-user@" +strIp+":
52          /home/repos/ns-3-allinone/ns-3-dev/scratch");
53      }
54
55      if(executa.waitFor() == 0) {
56
57        Simulacao simulacao = new Simulacao(Integer.parseInt(strqtdsimula),
58                              0, Integer.parseInt(idLogin), strDescricao);
59
60        simulacaoDao.adiciona(simulacao);
61
62        Usuario usuario = new Usuario(Integer.parseInt(idLogin),strusuario);
63
64        simulacao.setId(simulacaoDao.buscaIdSimula());
65
66        List<Instancia> listInstancia = instanciaDao.getListaInstancia();
67
68        for(int count = 0; count < listInstancia.size(); count++) {
69
70           SimulaInstancia simulacaoInstancia = new SimulaInstancia(
71             listInstancia.get(count), simulacao, usuario, strNameFile);
72
73           SimulacaoAsincronaCompletionCallback(simulacaoInstancia,
74         strusuario,count+1, simulacao.getId(),simulacao.getQtdSimula());
75           }
76
77           servletResponse.sendRedirect("http://54.94.239.252:8080/
78           wsclients3/formsimulation.jsp?strusuario=" + strusuario +
79           "&idLogin=" + idLogin);
80      }
81
82      }catch (InterruptedException ex) {
83        System.out.println("error: Envio do Arquivo XML" + ex.getMessage());
84      }catch (IOException ex) {
85        Logger.getLogger(CentralRs.class.getName()).log(Level.SEVERE, null,
86        ex);
87        System.out.println("error: Excecao IO" + ex.getMessage());
88      }
89  }
90
91  private void SimulacaoAsincronaCompletionCallback(SimulaInstancia
92    simulacaoInstancia, String strUsuario, int idInstancia,
93    int idSimulacao,int qtdSimulacao) {
94
```

```
95    Client client = ClientBuilder.newClient();
96
97    String strTarget = simulacaoInstancia.getUrl() + "&idinstancia="+
98      idInstancia+"&idsimulacao="+idSimulacao+"&qtdsimulacao="+
99      qtdSimulacao;
100
101   AsyncInvoker async = client.target(strTarget).request().async();
102
103   Future<Response> future;
104   future = async.get(
105
106            new InvocationCallback<Response>() {
107
108            @Override
109            public void completed(Response response) {
110
111                 simulacaoInstancia.setResultado(response.readEntity(
112                           String.class));
113                 simulacaoDao.altera(simulacaoInstancia.getSimulacao());
114
115                 try {
116                      String stringResult = downloadToFileResult(
117                           strUsuario,idSimulacao);
118                      stringResult = getXMLResult();
119
120                      String urlLink = sendToFileSimulator(stringResult,
121                         strUsuario,idSimulacao);
122
123                      if(!"".equals(urlLink)){
124                           simInstanciaDao.adiciona(simulacaoInstancia,
125                             urlLink);
126                           SendMail(urlLink);
127                      }
128                 } catch (IOException ex) {
129                              Logger.getLogger(CentralRs.class.getName())
130                       .log(Level.SEVERE, null, ex);
131                 } catch (InterruptedException ex) {
132                              Logger.getLogger(CentralRs.class.getName())
133                       .log(Level.SEVERE, null, ex);
134            }
135           }
136
137           @Override
138              public void failed(Throwable throwable) {
139                     throwable.getStackTrace();
140              }
141           }
142         );
143   }
144
145   private String downloadToFileResult(String strUsuario,
146            int idSimulacao) throws IOException, InterruptedException{
147
148     System.out.println("Entrando no Metodo sendToFileSimulator");
149
150     String accessKey = "AKIAJ4Q5MUGHXSADFGH33OZQ";
151     String secretKey = "ogvXfGv9lF4-ibg6rh1dZQnT86oVBAeAt8uV4FDa";
152
153     AWSCredentials credentials = null;
154
155     String folderUser = strUsuario + "_" + idSimulacao +"/";
156
157     try {
158
159      credentials = new BasicAWSCredentials(accessKey, secretKey);
160
161     }catch (Exception e) {
162        throw new AmazonClientException(
```

```
163       "Cannot load the credentials from the credential profiles
164                                 file. ", e);
165     }
166
167
168     AmazonS3 s3 = new AmazonS3Client(credentials);
169
170
171     Region usWest2 = Region.getRegion(Regions.SA_EAST_1);
172
173     s3.setRegion(usWest2);
174
175     String bucketName = "filesimulator";
176
177     String finalResult = "";
178
179     try {
180
181      File file = new File("/home/repos/");
182
183      TransferManager tm = new TransferManager(s3);
184
185      MultipleFileDownload mfd = tm.downloadDirectory(bucketName,
186                      folderUser, file);
187
188      mfd.waitForCompletion();
189      tm.shutdownNow();
190
191      String pathFile = MixFile(getFileType(folderUser),folderUser);
192
193      NSCloudMediaResultado ns = new NSCloudMediaResultado();
194      ns.loadResultado(pathFile);
195      ns.calcularMedia();
196
197      finalResult = ns.getMediaResultado();
198
199      finalResult =  finalResult.split("/")[0];
200
201
202      } catch (AmazonServiceException ase) {
203         System.out.println("Error Message: " + ase.getMessage());
204         System.out.println("Error Type:    " + ase.getErrorType());
205         System.out.println("Request ID:    " + ase.getRequestId());
206      } catch (AmazonClientException ace) {
207         System.out.println("Error Message: "+ ace.getMessage());
208      }
209        return finalResult;
210 }
211
212 private String sendToFileSimulator(String stringResult,String strUsuario,
213                int idSimulacao) throws IOException{
214
215     String accessKey = "AKIAJ4Q5MUGHXSADFGH33OZQ";
216     String secretKey = "ogvXfGv9lF4-ibg6rh1dZQnT86oVBAeAt8uV4FDa";
217
218     AWSCredentials credentials = null;
219
220     String nameFile = String.format("%s_%s.xml", strUsuario, idSimulacao);
221
222   try {
223     credentials = new BasicAWSCredentials(accessKey, secretKey);
224
225   }catch (Exception e) {
226     throw new AmazonClientException(
227     "Cannot load the credentials from the credential profiles file.", e);
228   }
229   AmazonS3 s3 = new AmazonS3Client(credentials);
230   Region usWest2 = Region.getRegion(Regions.SA_EAST_1);
```

```
231   s3.setRegion(usWest2);
232
233   String bucketName = "filesimulator";
234
235   String nameFolder = String.format("%s_%s/", strUsuario, idSimulacao);
236
237   String url = "";
238
239   try {
240
241     String bucketLocation = s3.getBucketLocation(bucketName);
242
243     url = String.format("https://s3-%s.amazonaws.com/%s/%s%s",
244               bucketLocation, bucketName, nameFolder, nameFile);
245
246     PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
247          nameFolder+nameFile, createSampleFile(stringResult));
248     putObjectRequest.withCannedAcl(CannedAccessControlList.
249                       PublicReadWrite);
250     PutObjectResult putObject = s3.putObject(putObjectRequest);
251
252     }catch(AmazonServiceException ase) {
253       System.out.println("Error Message:     " + ase.getMessage());
254       System.out.println("HTTP Status Code: " + ase.getStatusCode());
255       System.out.println("AWS Error Code:   " + ase.getErrorCode());
256       System.out.println("Error Type:       " + ase.getErrorType());
257       System.out.println("Request ID:       " + ase.getRequestId());
258     } catch (AmazonClientException ace) {
259       System.out.println("Caught an AmazonClientException, which
260                 + means the client encountered "
261                     + "a serious internal problem while
262                     + "trying to communicate with S3,");
263             System.out.println("Error Message: " + ace.getMessage());
264         }
265         return url;
266     }
267 private static File createSampleFile(String result) throws
268                                              IOException {
269
270   String type[] = result.split("\\.");
271
272   if(type[1].equals("zip")){
273    File fileZip = new File("/home/repos/ns-3-allinone/ns-3-dev/"+result);
274    return fileZip;
275
276   }else{
277    File file = File.createTempFile("aws-java-sdk", ".txt");
278
279    Writer writer = new OutputStreamWriter(new FileOutputStream(file));
280    writer.write(result);
281    writer.close();
282
283    return file;
284
285    }
286
287
288 }
289
290 private String getFileSimulator(String strusuario) throws IOException{
291
292   String accessKey = "AKIAJ4Q5MUGHXSADFGH33OZQ";
293   String secretKey = "ogvXfGv9lF4-ibg6rh1dZQnT86oVBAeAt8uV4FDa";
294
295   AWSCredentials credentials = null;
296
297   try {
298     credentials = new BasicAWSCredentials(accessKey, secretKey);
```

```
299   } catch (Exception e) {
300     throw new AmazonClientException(
301     "Cannot load the credentials from the credential profiles file. " +
302     "Please make sure that your credentials file is at the correct " +
303     "location (/home/arlison/.aws/credentials), and is in valid format.",
304     e);
305   }
306
307   AmazonS3 s3 = new AmazonS3Client(credentials);
308   Region usWest2 = Region.getRegion(Regions.SA_EAST_1);
309   s3.setRegion(usWest2);
310
311   String bucketName = "filesimulator";
312   String key = strusuario;
313   String url = "";
314
315
316   try {
317
318    for (Bucket bucket : s3.listBuckets()) {
319         System.out.println(" - " + bucket.getName());
320    }
321
322    } catch (AmazonServiceException ase) {
323     System.out.println("Error Message:    " + ase.getMessage());
324     System.out.println("Request ID:       " + ase.getRequestId());
325    } catch (AmazonClientException ace) {
326       System.out.println("Error Message: " + ace.getMessage());
327    }
328         return "";
329 }
330 private void displayTextInputStream(InputStream input) throws
331                                         IOException {
332  BufferedReader reader= new BufferedReader(new InputStreamReader(input));
333   while (true) {
334      String line = reader.readLine();
335      if (line == null) break;
336        System.out.println("    " + line);
337      }
338        System.out.println();
339   }
340
341 public String MixFile(File[] file,String userIdSimula)throws IOException{
342
343  BufferedWriter buffWrite = new BufferedWriter(new FileWriter("/home/
344                    repos/" + userIdSimula + "resultInstance.csv"));
345  FileReader fr = null;
346
347
348             for (File file1 : file) {
349
350                 try {
351                     fr = new FileReader(file1);
352
353                     BufferedReader lerArq = new BufferedReader(fr);
354                     String linha = lerArq.readLine();
355
356                     linha = linha.split("/")[0];
357                     buffWrite.append(linha + "\n");
358
359                 }catch (FileNotFoundException ex) {
360                     System.out.println(ex.getMessage());
361                 }
362
363             }
364             buffWrite.close();
365
366             return "/home/repos/" + userIdSimula +"resultInstance.csv";
```

```
367         }
368
369     private File[] getFileType(String folderUser){
370       FileFilter filter = (File file) -> file.getName().endsWith(".rst");
371       File dir = new File("/home/repos/" + folderUser );
372       File[] files = dir.listFiles(filter);
373
374      return files;
375     }
376 }
```

Listing A.1: Recurso Simulação classe *CentralRs.java*.

A.5 Implementação do Recurso Arquivos

```
1
2  @Path("/arquivos")
3
4  public class InstanceRS {
5
6
7      @Context
8      UriInfo uriInfo;
9      @Context
10     Request request;
11
12     /**
13      *
14      * @param path
15      * @return
16      */
17
18   public String getFile(String path){
19
20     String linha = "";
21     String saida = "";
22
23     File file = new File(path);
24
25     try {
26       BufferedReader lerArq = new BufferedReader(new FileReader(file));
27
28       while ((linha = lerArq.readLine()) != null) {
29           saida += linha + "\n";
30       }
31
32       lerArq.close();
33
34    }catch(IOException e) {
35       System.err.printf("Erro na abertura: %s.\n", e.getMessage());
36    }
37    System.out.println("Retorno do arquivo XML como string: " + saida);
38    return saida;
39  }
40
41
42  private final ExecutorService executorService = java.util.concurrent.
43                      Executors.newCachedThreadPool();
44
45
46   @GET
47   @Produces(value = MediaType.TEXT_PLAIN)
48   public void simulaArquivo(
49      @Suspended final AsyncResponse asyncResponse,
```

```
50      @QueryParam(value = "strusuario") final String strusuario,
51      @QueryParam(value = "strarquivo") final String strarquivo,
52      @QueryParam(value = "idinstancia") final int idinstancia,
53      @QueryParam(value = "idsimulacao") final int idsimulacao,
54      @QueryParam(value = "qtdsimulacao") final int qtdsimulacao)
55
56     throws JDOMException, IOException
57     {
58
59
60      String[] nameFile = strarquivo.split(".");
61      System.out.println("Arquivo Instance: " + strarquivo);
62      String dataXML = "/home/repos/ns-3-allinone/ns-3-dev/scratch/"
63                               +strarquivo;
64      String inputXSL = "/home/repos/ns-3-allinone/ns-3-dev/scratch/
65                           simulaprotocol.xsl";
66      String outputCC = "/home/repos/ns-3-allinone/ns-3-dev/
67                            scratch/simulaprotocol.cc";
68
69    try {
70      try {
71       transform(dataXML, inputXSL, outputCC);
72       String fileOut = getFile("/home/repos/ns-3-allinone/ns-3-dev/
73                               scratch/simulaprotocol.cc");
74
75       fileOut = fileOut.replace("<?xml version=\"1.0\"
76                               encoding=\"UTF-8\"?>", "")
77                                     .replace("&lt;", "<")
78                                     .replace("&gt;", ">")
79                                     .replace("&amp;", "&");
80
81       try {
82         fileReplace("/home/repos/ns-3-allinone/ns-3-dev/scratch/
83                               simulaprotocol.cc",fileOut);
84
85       }catch(IOException ex) {
86         Logger.getLogger(InstanceRS.class.getName()).log(Level.SEVERE,
87                                       null, ex);
88       }
89     }catch(FileNotFoundException ex) {
90         Logger.getLogger(InstanceRS.class.getName()).log(Level.SEVERE,
91                                       null, ex);
92     }
93    } catch (TransformerException ex) {
94        Logger.getLogger(InstanceRS.class.getName()).log(Level.SEVERE,
95                                       null, ex);
96    }
97
98    executorService.submit(() -> {
99    try {
100    asyncResponse.resume(doSimulaArquivo(strusuario, strarquivo,
101                        idinstancia, idsimulacao, qtdsimulacao));
102   } catch (JDOMException | IOException ex) {
103     Logger.getLogger(InstanceRS.class.getName()).log(Level.SEVERE,
104                                       null, ex);
105   }
106  });
107 }
108
109 public String doSimulaArquivo(
110     @QueryParam(value = "strusuario") final String strusuario,
111     @QueryParam(value = "strarquivo") final String strarquivo,
112     @QueryParam(value = "idinstancia") final int idinstancia,
113     @QueryParam(value = "idsimulacao") final int idsimulacao,
114     @QueryParam(value = "qtdsimulacao")  final  int qtdsimula)
115              throws JDOMException, IOException
116     {
117
```

```
118     File file = new File("/home/repos/ns-3-allinone/ns-3-dev/scratch/"
119                                 + strarquivo);
120
121
122     String typeResult = getTypeSimulaton(strarquivo);
123
124     String userIdSimula = strusuario+"_"+idsimulacao;
125
126     String caminho = file.getName();
127
128     String[] nome_arquivo = caminho.split("\\.");
129
130     if(file.exists()){
131
132       File novo = new File("/home/repos/ns-3-allinone/ns-3-dev/");
133
134       Process executa = null;
135
136      for(int i = 0; i < qtdsimula; i++){
137
138       try {
139          executa = Runtime.getRuntime().exec(new String[] {"/bin/bash",
140          "-c", "sudo ./waf --command-template='%s --RngRun=" + (i + 1) +
141          "' --run " + nome_arquivo[0] + "> result"+i+".rst >&1"}, null,
142                                       novo);
143       }catch (IOException ex) {}
144
145       try {
146            if(executa.waitFor() == 0) {
147
148            }
149       }catch (InterruptedException ex) {}
150     }
151
152     if(typeResult.equals("csv")){
153
154       String fileResultMix = MixFile(getFileType("rst"),userIdSimula);
155       NSCloudMediaResultado result = new NSCloudMediaResultado();
156       result.loadResultado(fileResultMix);
157       result.calcularMedia();
158
159       String strResultMedia = result.getMediaResultado();
160       sendToFileSimulator(strResultMedia,strusuario,idinstancia,
161                                 idsimulacao);
162
163       return "simulacao em arquivo csv concluida!";
164
165       }else{
166         compactFiles(nome_arquivo[0],"rst");
167         sendToFileSimulator("arquivo",strusuario,idinstancia,
168                                 idsimulacao);
169
170         return nome_arquivo[0] + ".zip";
171       }
172
173  }
174
175     return null;
176 }
177
178 private void transform(String dataXML, String inputXSL, String outputCC)
179             throws TransformerException, FileNotFoundException{
180
181
182   TransformerFactory tFactory = TransformerFactory.newInstance();
183             Source xslDoc = new StreamSource(inputXSL);
184             Source xmlDoc = new StreamSource(dataXML);
185
```

```
186             String outputFileName = outputCC;
187             OutputStream newFile = new FileOutputStream(outputFileName);
188 
189             Transformer transformer = tFactory.newTransformer(xslDoc);
190             transformer.transform(xmlDoc, new StreamResult(newFile));
191 }
192 
193 private void fileReplace(String path, String fileResult) throws
194                                 IOException {
195 
196   BufferedWriter buff = new BufferedWriter(new FileWriter(path));
197 
198   buff.write(fileResult);
199   buff.close();
200 }
201 
202 private String getTypeSimulaton(String strFileName) throws
203                         JDOMException, IOException{
204 
205   File file = new File("/home/repos/ns-3-allinone/ns-3-dev/scratch/"
206                                   +strFileName);
207 
208   if(file.exists()){
209     SAXBuilder sb = new SAXBuilder();
210 
211     Document d = sb.build(file);
212 
213     Element simulation = d.getRootElement();
214 
215     List elements = simulation.getChildren();
216     Iterator i = elements.iterator();
217 
218     Element element = null;
219 
220    while (i.hasNext()) {
221       element = (Element) i.next();
222    }
223    return element.getAttributeValue("fileType");
224 
225   }else{
226      return ("inexistente");
227   }
228 }
229 
230 private void compactFiles(String strFile, String type){
231 
232  File[] files = getFileType(type);
233 
234  String[] nameFiles = new String[files != null ? files.length : 0];
235 
236  for (int i = 0; i < files.length; i++) {
237 
238    nameFiles[i] = "/home/repos/ns-3-allinone/ns-3-dev/"+files[i].
239                                   getName();
240 
241   }
242   zipFiles(nameFiles, "/home/repos/ns-3-allinone/ns-3-dev/" + strFile
243                                   + ".zip");
244 }
245 
246 private File[] getFileType(String type){
247    FileFilter filter = (File file) -> file.getName().endsWith("."+type);
248    File dir = new File("/home/repos/ns-3-allinone/ns-3-dev/");
249    File[] files = dir.listFiles(filter);
250 
251    return files;
252 }
253 private void zipFiles(String[] srcFiles, String zipFile) {
```

```
254     try {
255
256          byte[] buffer = new byte[1024];
257          FileOutputStream fos = new FileOutputStream(zipFile);
258          ZipOutputStream zos = new ZipOutputStream(fos);
259
260          for (int i = 0; i < srcFiles.length; i++) {
261             File srcFile = new File(srcFiles[i]);
262
263             FileInputStream fis = new FileInputStream(srcFile);
264
265             zos.putNextEntry(new ZipEntry(srcFile.getName()));
266
267             int length;
268
269             while ((length = fis.read(buffer)) > 0) {
270
271                zos.write(buffer, 0, length);
272
273             }
274
275             zos.closeEntry();
276             fis.close();
277
278          }
279              zos.close();
280
281          } catch (IOException ioe) {
282              System.out.println("Error creating zip file: " + ioe);
283
284          }
285 }
286 public String MixFile(File[] file, String userIdSimula) throws
287                               IOException{
288
289     BufferedWriter buffWrite = new BufferedWriter(new FileWriter(
290           "/home/repos/ns-3-allinone/ns-3-dev/resultInstance.csv"));
291
292     FileReader fr = null;
293
294     for (File file1 : file) {
295
296     try {
297        fr = new FileReader(file1);
298
299        BufferedReader lerArq = new BufferedReader(fr);
300        String linha = lerArq.readLine();
301
302        while (linha != null) {
303           System.out.printf("%s\n", linha);
304           buffWrite.append(linha + "\n");
305           linha = lerArq.readLine(); //
306
307        }
308        }catch (FileNotFoundException ex) {
309              System.out.println(ex.getMessage());
310     }
311
312     }
313        buffWrite.close();
314       return "/home/repos/ns-3-allinone/ns-3-dev/resultInstance.csv";
315 }
316 private void sendToFileSimulator(String resutInstance, String strUsuario,
317              int idInstancia, int idSimulacao) throws IOException{
318
319  System.out.println("Entrando no Metodo sendToFileSimulator");
320
321  String accessKey = "AKIAJ4Q5MUGHFVD33OZQ";
```

```
322  String secretKey = "ogvXfGv9lJ5+ids8rg1dZQnT86oVBAeAt8uV4FDa";
323
324  AWSCredentials credentials = null;
325
326  Calendar calendar = Calendar.getInstance();
327  int dia = calendar.get(Calendar.DAY_OF_MONTH);
328  int mes = calendar.get(Calendar.MONTH);
329  int ano = calendar.get(Calendar.YEAR);
330  long seg = calendar.getTimeInMillis();
331
332  String nameFile = strUsuario +"_"+idSimulacao+"_"+idInstancia+"_"+dia
333                              +mes+ano+seg+".rst";
334
335  try {
336       credentials = new BasicAWSCredentials(accessKey, secretKey);
337
338  }catch (Exception e) {
339     throw new AmazonClientException(
340    "Cannot load the credentials from the credential profiles file. " +
341    "Please make sure that your credentials file is at the correct " +
342    "location (/home/arlison/.aws/credentials), and is in valid format.",
343    e);
344  }
345
346          AmazonS3 s3 = new AmazonS3Client(credentials);
347
348          Region usWest2 = Region.getRegion(Regions.SA_EAST_1);
349
350          s3.setRegion(usWest2);
351
352          String bucketName = "filesimulator";
353
354          String nameFolder = strUsuario +"_"+idSimulacao+"/";
355
356         try {
357
358          PutObjectRequest putObjectRequest = new PutObjectRequest(
359     bucketName, nameFolder+nameFile, createSampleFile(resutInstance));
360          putObjectRequest.withCannedAcl(CannedAccessControlList.
361                              PublicReadWrite);
362          PutObjectResult putObject = s3.putObject(putObjectRequest);
363
364          }catch (AmazonServiceException ase) {
365            System.out.println("Error Message:    " + ase.getMessage());
366          } catch (AmazonClientException ace) {
367              System.out.println("Error Message: " + ace.getMessage());
368          }
369
370     }
371     private File createSampleFile(String result) throws IOException {
372
373
374          System.out.println("Criando Arquivo temporario");
375
376          File file = File.createTempFile("aws-java-sdk", ".txt");
377          //file.deleteOnExit();
378
379          Writer writer = new OutputStreamWriter(new FileOutputStream(file));
380          writer.write(result);
381          writer.close();
382              return file;
383
384     }
385
386  }
```

Listing A.2: Recurso Arquivos classe *InstanceRS.java*.

B Apêndice

B.1 Documento XML com parâmetros de Simulação

```
<?xml version="1.0" encoding="UTF-8"?>
 <simulation name= "simulation_01" type="manet" totalTime = "50.0">
      <nodeGroup numberNodes = "40">
            <node id="0" applicationNode = "AODV" value = "2">
            <protocol name="AODV" type="IPV4">
                <adress>172.19.0.0</adress>
                <mask>255.255.0.0</mask>
            </protocol>
        </node>
      </nodeGroup>
      <interface></interface>
      <channel>
        <type>
            <manet name="manet">
                <activeNodes>8</activeNodes>
                <nodeSpeed>15</nodeSpeed>
                <nodePause>0</nodePause>
                <dataRate>2048bps</dataRate>
                <phyMode>DsssRate11Mbps</phyMode>
                <dimension>
                    <vertical>
                        <minimun>0.0</minimun>
                        <maximun>500.0</maximun>
                    </vertical>
                    <horizontal>
                        <minimun>0.0</minimun>
                        <maximun>500.0</maximun>
                    </horizontal>
                </dimension>
            </manet>
            <wifi name="wifi">
                <standard></standard>
                <propagation></propagation>
                <moble_model></moble_model>
                <qos>true</qos>
                <client>
                    <maxpackets></maxpackets>
                    <interval></interval>
                    <packetsize></packetsize>
                </client>
            </wifi>
        </type>
        <bandwidth></bandwidth>
        <delay></delay>
    </channel>
      <aplications>
        <server name="appserver">
            <start>1.0</start>
            <stop>10.0</stop>
        </server>

        <client name="appclient">
            <start>1.0</start>
            <stop>10.0</stop>
        </client>
    </aplications>
```

```
    <description>Simulacao com protocolos AODV, com tempo de 50s, area
                 500x500m, 40 n s sendo 8 ativos
    </description>
    <result type = "csv" />

 </simulation>
```

Listing B.1: Exemplo de documento com parâmetros de simulação

B.2 Trecho de Template de Transformação de Documento XML

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                           version="1.0">

<xsl:template match="/">

<![CDATA[


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("simulaprotocol");


class RoutingExperiment
{
public:
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfileName);
  std::string CommandSetup (int argc, char **argv);

RoutingExperiment::RoutingExperiment ()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfileName ("simulaprotocol.csv"),
    m_traceMobility (false),
]]>
   m_protocol (<xsl:value-of select="simulation/nodeGroup/node/@value"/>)
{
}

<![CDATA[

void
RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  while ((packet = socket->Recv ()))
    {
      bytesTotal += packet->GetSize ();
      packetsReceived += 1;
      NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
    }
}

int main (int argc, char *argv[])
{
  RoutingExperiment experiment;
  std::string CSVfileName = experiment.CommandSetup (argc,argv);
```

```
51
52    //blank out the last output file and write the column headers
53    std::ofstream out (CSVfileName.c_str ());
54    out << "SimulationSecond," <<
55    "ReceiveRate," <<
56    "PacketsReceived," <<
57    "NumberOfSinks," <<
58    "RoutingProtocol," <<
59    "TransmissionPower" <<
60    std::endl;
61    out.close ();
62
63 ]]>
64    int nSinks = <xsl:value-of select="simulation/channel/type/manet/
65                                                    activeNodes"/>;
66
67    double txp = 7.5;
68    experiment.Run (nSinks, txp, CSVfileName);
69 }
70
71 void
72 RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
73 {
74  Packet::EnablePrinting ();
75  m_nSinks = nSinks;
76  m_txp = txp;
77  m_CSVfileName = CSVfileName;
78
79  int nWifis = <xsl:value-of select="simulation/nodeGroup/@numberNodes"/>;
80
81  double TotalTime = <xsl:value-of select="simulation/@totalTime"/>;
82  std::string rate ("<xsl:value-of select="simulation/channel/type/manet/
83                dataRate"/>");
84  std:string phyMode("<xsl:value-of select="simulation/channel/type/manet/
85                phyMode"/>");
86  std::string tr_name ("simulaprotocol");
87
88  int nodeSpeed = <xsl:value-of select="simulation/channel/type/manet/
89           nodeSpeed"/>;
90  int nodePause = <xsl:value-of select="simulation/channel/type/manet/
91           nodePause"/>;
92  m_protocolName = "protocol";
93
94
95  YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
96  YansWifiChannelHelper wifiChannel;
97  wifiChannel.SetPropagationDelay ("ns3::
98                   ConstantSpeedPropagationDelayModel");
99  wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
100 wifiPhy.SetChannel (wifiChannel.Create ());
101
102 pos.Set("X", StringValue ("ns3::UniformRandomVariable[Min=<xsl:value-of
103 select="simulation/channel/type/manet/dimension/vertical/minimun"/>|
104 Max=<xsl:value-of select="simulation/channel/type/manet/dimension/
105                          vertical/maximun"/>]"));
106 pos.Set("Y", StringValue ("ns3::UniformRandomVariable[Min=<xsl:value-of
107               select="simulation/channel/type/manet/dimension/
108               horizontal/minimun"/>|Max=<xsl:value-of select=
109 "simulation/channel/type/manet/dimension/horizontal/maximun"/>]"));
110
111 <![CDATA[
112  Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<
113  PositionAllocator>(); streamIndex += taPositionAlloc->AssignStreams
114  (streamIndex);
115
116  std::stringstream ssSpeed;
117
118  ssSpeed <<"ns3::UniformRandomVariable[Min=5.0|Max="<< nodeSpeed << "]";
```

```
119  std::stringstream ssPause;
120  ssPause <<"ns3::ConstantRandomVariable[Constant="<< nodePause << "]";
121 ]]>
122
123  switch (m_protocol)
124   {
125     case 1:
126       list.Add (olsr, 100);
127       m_protocolName = "OLSR";
128       break;
129     case 2:
130       list.Add (aodv, 100);
131       m_protocolName = "AODV";
132       break;
133     default:
134 <![CDATA[
135       NS_FATAL_ERROR ("Protocolo Inexistente:" << m_protocol);
136   }
137
138   internet.SetRoutingHelper (list);
139   internet.Install (adhocNodes);
140
141
142   NS_LOG_INFO ("Atribuindo enderecos IP");
143
144   Ipv4AddressHelper addressAdhoc;
145 ]]>
146  addressAdhoc.SetBase ("<xsl:value-of select="simulation/nodeGroup/node/
147                              protocol/adress"/>",
148   "<xsl:value-of select="simulation/nodeGroup/node/protocol/mask"/>");
149
150   Ipv4InterfaceContainer adhocInterfaces;
151   adhocInterfaces = addressAdhoc.Assign (adhocDevices);
152
153   OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
154   onoff1.SetAttribute ("OnTime", StringValue (
155     "ns3::ConstantRandomVariable [Constant=1.0]"));
156   onoff1.SetAttribute ("OffTime", StringValue (
157     "ns3::ConstantRandomVariable [Constant=0.0]"));
158
159 <![CDATA[
160   for (int i = 0; i < nSinks; i++)
161     {
162       Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.
163                     GetAddress(i), adhocNodes.Get (i));
164
165       AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.
166                               GetAddress (i), port));
167       onoff1.SetAttribute ("Remote", remoteAddress);
168
169       Ptr<UniformRandomVariable> var = CreateObject
170                         <UniformRandomVariable> ();
171       ApplicationContainer temp = onoff1.Install (adhocNodes.Get
172                               (i + nSinks));
173       temp.Start (Seconds (var->GetValue (0.0,1.0)));
174       temp.Stop (Seconds (TotalTime));
175     }
176
177   Ptr<FlowMonitor> flowmon;
178   FlowMonitorHelper flowmonHelper;
179   flowmon = flowmonHelper.InstallAll ();
180
181   NS_LOG_INFO ("Run Simulation.");
182
183   CheckThroughput ();
184
185   Simulator::Stop (Seconds (TotalTime));
186   Simulator::Run ();
```

```
187
188     flowmon->CheckForLostPackets ();
189     std::map<FlowId, FlowMonitor::FlowStats> stats =
190             flowmon->GetFlowStats ();
191
192     uint32_t txPacketsum = 0;
193     uint32_t rxPacketsum = 0;
194     uint32_t DropPacketsum = 0;
195     uint32_t LostPacketsum = 0;
196     double Delaysum = 0;
197
198     for (std::map<FlowId, FlowMonitor::FlowStats>::
199      const_iterator i = stats.begin (); i != stats.end (); ++i)
200     {
201         txPacketsum += i->second.txPackets;
202         rxPacketsum += i->second.rxPackets;
203         LostPacketsum += i->second.lostPackets;
204         DropPacketsum += i->second.packetsDropped.size();
205         Delaysum += i->second.delaySum.GetSeconds();
206     }
207
208     std::cout << "Enviados:" << txPacketsum << ";
209     Recebidos:" <<rxPacketsum <<";Perdidos:"<<LostPacketsum << ";
210     Excluidos:" << DropPacketsum << ";
211     Atraso_medio:" << Delaysum / rxPacketsum << "\n";
212
213
214
215 ]]>
216     Simulator::Destroy ();
217
218 }
219
220 </xsl:template>
221 i</xsl:stylesheet>
```

Listing B.2: Trecho de Template de Transformação de Documento XML

B.3 Avaliação em cenários AODV e OLSR

Neste Apêndice são expostas tabelas contendo os resultados das simulações apresentadas no capítulo 6, relativos aos protocolos AODV e OLSR.

Tabela B.1: Resultado para 20 nós.

	AODV			OLSR		
Pacotes	Média	Desvio Padrão	Intervalo Confiança	Média	Desvio Padrão	Intervalo Confiança
Enviados	2855.133	440.636	244.016	1121.199	13.390	7.415
Recebidos	2474.933	419.407	232.260	1087.333	30.791	17.051
Perdidos	303.933	120.835	66.916	21.466	20.642	11.431
Excluídos	71.466	24.850	13.761	0.0	0.0	0.0
Atraso Médio	0.068	0.010	0.006	0.005	0.004	0.002

Tabela B.2: Resultado para 30 nós.

	AODV			OLSR		
Pacotes	Média	Desvio Padrão	Intervalo Confiança	Média	Desvio Padrão	Intervalo Confiança
Enviados	4294.46	1135.65	628.905	1116.2	15.4096	8.5335
Recebidos	3769.4	938.539	519.7461	1085.53	21.3369	11.816
Perdidos	429.66	251.662	139.366	24.6	13.8295	7.658
Excluídos	170.60	70.345	38.9562	0.0	0.0	0.0
Atraso Médio	0.1250	0.02731	0.015126	0.0087	0.00586	0.003250

Tabela B.3: Resultado para 40 nós.

	AODV			OLSR		
Pacotes	Média	Desvio Padrão	Intervalo Confiança	Média	Desvio Padrão	Intervalo Confiança
Enviados	6714.466	1157.810	641.174	1486.333	9.925	5.496
Recebidos	5781.599	1073.286	594.366	1454.133	18.749	10.383
Perdidos	736.266	225.479	124.866	22.799	14.654	8.115
Excluídos	290.933	82.811	45.859	0.0	0.0	0.0
Atraso Médio	0.170	0.0154	0.008	0.006	0.004	0.002