UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

KARL HANSIMULLER ALELAF FERREIRA

Integrando o Network Simulator 2.0 a um Ambiente Virtual de Aprendizagem

MOSSORÓ – RN

2014

KARL HANSIMULLER ALELAF FERREIRA

Integrando o Network Simulator 2.0 a um Ambiente Virtual de Aprendizagem

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. Rommel Wladimir de Lima – UERN.

MOSSORÓ – RN

2014

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Ferreira, Karl Hansimuller Alelaf.

Integrando o Network Simulator 2.0 a um Ambiente Virtual de Aprendizagem. / Karl Hansimuller Alelaf Ferreira. - Mossoró, RN, 2014.

86 f.

Orientador(a): Prof. D.Sc. Rommel Wladimir de Lima

Dissertação (Mestrado em Ciência da Computação). Universidade do Estado do Rio Grande do Norte. Programa de Pós-Graduação em Ciência da Computação.

1. Sistemas multiagentes - Dissertação. 2. NS2. 3. Simulação. I. Lima, Rommel Wladimir de. II. Universidade do Estado do Rio Grande do Norte. III.Título.

UERN/ BC CDD 004

Bibliotecária: Elaine Paiva de Assunção – CRB - 15/492

KARL HANSIMULLER ALELAF FERREIRA

Integrando o Network Simulator 2.0 a um Ambiente Virtual de Aprendizagem

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 18 / 07 / 2014.

BANCA EXAMINADORA

Prof. D.Sc. Rommel Wladimir de Lima – UERN
Presidente

Prof. D.Sc. Karla Darlene Nepomuceno Ramos – UERN
Membro Interno

Prof. D.Sc. Leonardo Cunha de Miranda – UFRN
Membro Externo

Dedico este trabalho a meu bom Senhor Jesus, a minha família e minha noiva.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado forças e sabedoria para conseguir realizar esta pesquisa. Agradeço a toda minha família, em especial minha mãe Olga, meu pai Flábio, minha irmã Kamila, minhas avós Lourdes e Joanice, a minha bisavó Maria José (*in memorian*), aos meus tios Hiran e Cesar (*in memorian*) e tias Nara, Fluvia, Elisangela, Ana e especialmente minha tia Neide. Meu primo Ryan e primas Chiê, Jaqueline, Rosarlyny, Maria Clara e Ana Rita, que estiveram sempre ao meu lado.

Agradeço em especial a minha prima Fabíola, seu esposo Ivanilson (Picolé Equipadora) e meu sobrinho Felipe, que me acolheram tão carinhosamente nesses dois anos de mestrado em sua casa em Mossoró. Agradeço a minha amada noiva Alice Silva que teve paciência e por várias noites ao celular me aconselhou e prontamente estava com uma palavra de amor e de motivação.

Agradeço aos professores do Programa de Pós-Graduação em Ciência da Computação – UERN/UFERSA, pelos conhecimentos transmitidos, especialmente ao meu orientador Rommel Wladimir de Lima. Agradeço também aos amigos que conquistei no mestrado, que por dois anos compartilhamos alegrias e tristezas. Em especial Dickson, Ernando, Antônio Cabó, Osvaldo e Marcos Vinicius. Ao amigo Flávio Silva que me ajudou a resolver alguns problemas no estudo de algumas linguagens utilizadas nesse projeto.

A CAPES, pelo apoio financeiro que viabilizou a realização deste trabalho.

Por fim, agradeço a todos que, direta ou indiretamente contribuíram para a minha formação.

"*Os céus proclamam a glória de Deus, e o firmamento anuncia as obras das suas mãos. As palavras dos meus lábios e o meditar do meu coração sejam agradáveis na tua presença, SENHOR, rocha minha e redentor meu!*" (Salmos 19. 1- 2; 14)

RESUMO

O desenvolvimento de atividades práticas em disciplinas de redes de computadores constitui um ponto chave no aprendizado do aluno. Contudo, muitas vezes não é possível contar com um espaço físico que dê suporte ao desenvolvimento das práticas de rede, seja pela falta de equipamentos ou por falta de horários. Dessa forma a utilização de simuladores para suprir a carência desses equipamentos e horários indisponíveis tem sido uma abordagem bastante aplicada. Dentre as várias ferramentas de simulação utilizadas, destaca-se o Network Simulator 2, uma ferramenta que permite a simulação de vários tipos de cenários como redes IP e suporte a diversos tipos de tráfegos. Apesar das grandes vantagens do NS2, o mesmo possui algumas limitações como à necessidade do aprendizado de uma nova linguagem de especificação para criação e modelagem dos cenários de redes. Dessa forma, o presente trabalho tem como objetivo desenvolver uma interface gráfica para integrar o NS2 a um Ambiente Virtual de Aprendizagem (AVA), proporcionando uma ferramenta para modelagem, simulação e animação das simulações de redes de forma automatizada. Para tanto, foi concebida uma metodologia, sendo executada em 6 etapas, onde a ferramenta apresentada é integrada a um AVA como um novo bloco para adição de funcionalidades.

Palavras-Chave: NS2, Simulação, AVA, Agentes e Sistemas Multiagentes.

ABSTRACT

The development of practical activities in the disciplines of computer networks is a key point in student learning. However, it is often not possible to have a physical space that supports the development of practical network or the lack of equipment or lack of time. Thus, the use of simulators to meet the shortage of such equipment and unavailable times has been a very applied approach. Among the various simulation tools used, highlight the Network Simulator 2, a tool that allows the simulation of various scenarios such as IP networks and support different types of traffic. Despite the great advantages of NS2, it has some limitations as to the need for a new specification language for creation of network objects, and it also does not have a space for creation and modeling of network scenarios learning. Thus, this study aims to develop a graphical interface to integrate the NS2 to a Virtual Learning Environment (VLE), providing a tool for modeling, simulation and animation of simulations of networks of automated way. To that end, we designed a methodology, running in 6 steps, where the presented tool is integrated with a VLE as a new block for added functionality.

Keywords: NS2, Simulation, VLE, Agents and Multiagent Systems.

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE FIGURAS

LISTA DE SIGLAS

AC – *Agent Container;*

ACK – *Acknowledge;*

AID – *Agente Identificaton*;

AgM – Agente Moodle;

AgN – Agente NS;

AMS – *Agent Management System;*

API – *Application Programming Interface;*

AVA – Ambiente Virtual de Aprendizagem;

BST – *Bi-direcional Shared Tree mode*

CBR – *Constant Bit Rate*;

CMS – *Content Management System;*

CSS3 – *Cascading Style Sheets*, versão 3;

CSELT – *Centro Studi E Laboratori Telecomunicazioni*;

ctrMcast – *centralized Multicast*;

DARPA – *Defense Advanced Research Projects Agency*;

DF – *Directory Facilitator;*

DM – *Dense Mode*;

DV – *Distance Vector*;

DVRMP – *Distance Vector Multicast Routing Protocol*;

EaD – Educação a Distância;

FIFO – *First In, First Out*;

FIPA – *Foundation for Intelligent Physical Agent*;

FTP – *File Transfer Protocol*;

HTML5 – *Hypertext Markup Language*, versão 5;

HTTP – *Hypertext Transfer Protocol*;

ISI – *Information Sciences Institute*;

JADE – *Java Agent Development framework;*

J2EE – *Java 2 Platform, Enterprise Edition;*

J2SE - *Java 2 Platform, Standard Edition;*

JRE – *Java Runtime Environment;*

LBNL – *Lawrence Berkeley National Laboratory*;

MOODLE – *Modular Object-Oriented Dynamic Learning Environment*;

MTS – *Message Transport System;*

NAM – *Network Animator;*

NFS – *National Science Foundation*;

NSMoo – *Network Simulator for Moodle;*

NS2 – *Network Simulator* 2;

OTCL – *Object-oriented Toll Command Language;*

PARC – *Xerox Palo Alto Research*;

PIMDM – *Protocol Independent Multicast-dense mode*;

PHP – *Hypertext Preprocessor;*

SPF – *Shortest Path First*;

SCORM – *Sharable Content Object Reference Model*

TCP – *Transmission Control Protocol*;

TIC – Tecnologias de Informação e Comunicação;

UDP – *User Datagram Protocol;*

UERN – Universidade do Estado do Rio Grande do Norte;

UFERSA – Universidade Federal Rural do Semi-Árido;

USC – *University of Southern California*;

UTP - *Unshielded Twisted Pair;*

W3C – *Word Wide Web Consortium;*

WHATWG – *Web Hypertext Application Technology Working Group;*

XHTML – *eXtensible HyperText Markup Language*;

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Os recursos tecnológicos têm evoluído de forma rápida nas últimas décadas. Os avanços por meio da Internet têm trazido oportunidades para o desenvolvimento de atividades a distância, favorecendo a expansão da educação. Segundo Kenski (2008), as velozes transformações tecnológicas da atualidade impõem novos ritmos e dimensões à tarefa de ensinar e aprender, proporcionando ao sistema educacional caminhar a novas possibilidades no desenvolvimento de novas técnicas para aumentar a interação.

Dessa forma os estudos e avanços das Tecnologias de Informação e Comunicação (TIC) denotam atualmente uma nova abordagem no desenvolvimento de novas técnicas, disponibilizando novos recursos para o âmbito educacional. Assim as TIC têm contribuído de maneira significativa para o redimensionamento das estratégias de ensinar e aprender, proporcionando diferentes ferramentas ou metodologias para o apoio ao processo de aprendizagem (AMARAL et al., 2011).

Dois exemplos dessas ferramentas que podem ser utilizadas são os Ambientes Virtuais de Aprendizagem (AVA) e os Simuladores. Os AVAs são bastante propícios para o desenvolvimento de atividades, oferecendo um espaço repleto de interatividade e recursos que podem ser usados no meio acadêmico, enquanto que os simuladores são ferramentas de simulação que promovem o acesso a experimentos a partir de um espaço virtual, compensando à falta de interação e a indisponibilidade de horários ou de recursos necessários às experiências práticas (ARIEIRA et al., 2009).

Dessa forma, a utilização de simuladores inseridos em um AVA vem ganhando impulso possibilitando o desenvolvimento de competências técnicas, melhorando a aprendizagem e a fixação do conhecimento teórico. Para Pinheiro et al. (2009), a utilização de ferramentas de simulação agregadas a um AVA pode representar situações e comportamentos difíceis de serem representados na vida real, servindo como uma forma de preparar e treinar pessoas para que aprendam a lidar com situações reais.

No contexto das redes de computadores é essencial o desenvolvimento de práticas que comportem a utilização e configuração de equipamentos, a fim de tornar o usuário capacitado para as mais diversas realidades práticas. Segundo Voss et al. (2012), os temas técnicos na

área de redes de computadores envolve conceitos difíceis de serem entendidos na forma pedagógica tradicional.

Segundo Oliveira et al. (2012), dentre os diversos simuladores existentes que podem ser utilizados para dar apoio a simulação de redes de computadores, o *Network Simulator* 2 (NS2), destaca-se por ser um dos mais utilizados no meio acadêmico, possuindo várias contribuições de pesquisadores da área de redes. Trata-se de um simulador orientado a objetos escrito em C++ com um interpretador OTCL (*Object-Oriented Tool Command*) como *frontend* (OLIVEIRA et al., 2012).

Visando contribuir para melhorar as condições de ensino/aprendizagem em disciplinas de redes de computadores, e aproveitando a possibilidade de desenvolver ferramentas utilizando AVAs, este trabalho apresenta uma ferramenta para integrar o NS2 a um AVA.

A seção a seguir, aborda a problemática envolvida com o tema deste trabalho em relação à falta de um espaço apropriado para desenvolver práticas em redes de computadores e a justificativa do trabalho apresentado.

1.2 PROBLEMÁTICA E JUSTIFICATIVA

Em áreas do conhecimento que demandam muitas atividades práticas, como redes de computadores, engenharia, química ou física, o uso do laboratório é um componente essencial do currículo. Dessa forma a combinação da teoria com as atividades práticas é um elemento de suma importância para que ocorra a aprendizagem significativa (CALLAGHAN et al., 2008).

Segundo Amaral et al. (2011), um fator relevante que dá ênfase à necessidade de uso de simuladores como forma de complementar as atividades práticas é justamente a questão da dificuldade de acesso que muitos enfrentam com a falta de horários ou pela falta de laboratórios com equipamentos suficientes nas instituições.

Pode-se destacar também que o investimento na utilização de simuladores permite, para as instituições de ensino, a diminuição do custo de aquisição e manutenção dos laboratórios, como também permite a inclusão digital dos usuários, ao mesmo tempo em que pode estimular sua capacidade criativa e investigativa, bem como seu desenvolvimento pessoal (HASSAN, 2003).

Assim, os simuladores apresentam-se não apenas como uma tendência nos dias atuais, mas também, como um forte elemento cooperador para educação, pesquisa e desenvolvimento científico (AMARAL et al., 2011). Nesse sentido, muitas instituições de ensino, adotam a utilização de simuladores inseridos em AVAs para que os usuários possam interagir entre si e com os professores utilizando ferramentas que complementem suas atividades laboratoriais.

Especificamente para redes de computadores é necessário que o usuário realize experiências com objetivos imprescindíveis como: localizar o estudante sob a pilha de protocolos; visualizar as características de *hosts*, enlaces e portas; e analisar seu comportamento em diferentes topologias e cenários (BELZERANA; GONZALEZ-BARBONE, 2008).

Dessa forma o NS2 se caracteriza como uma ferramenta de simulação que pode ser utilizada para alcançar os objetivos no parágrafo anterior. Apesar de suas características positivas, o NS2 apresenta algumas restrições devido à falta de um ambiente de desenvolvimento para tornar o processo de criação e modelagem de cenários mais interativos.

Assim, criar cenários apenas utilizando *scripts* não leva o usuário a mapear imediatamente o cenário que será simulado. É preciso também levar em consideração o pouco tempo existente na disciplina para proporcionar uma maior familiarização com a linguagem OTCL que é utilizada no NS2. Limitações como estas podem prejudicar a utilização do simulador na disciplina de redes de computadores.

Com base na problemática exposta, na seção seguinte será descrito o objetivo a ser alcançado com este trabalho.

1.3 OBJETIVOS

Este trabalho tem como principal objetivo integrar o NS2 a um AVA, proporcionando de forma automatizada, uma ferramenta gráfica para modelagem, simulação e animação das simulações de redes.

Para alcançar o objetivo principal do trabalho, os seguintes pontos foram levados em consideração:

- Pesquisar e selecionar o AVA a ser utilizado;
- Projetar e modelar uma interface gráfica;
- Implementar mecanismo de geração automática de *script*;

- Implementar o mecanismo de integração do AVA com o NS2;
- Implementar mecanismo gráfico de animação da simulação;
- Testar as funcionalidades da ferramenta;

1.4 METODOLOGIA

Para atingir o objetivo geral deste trabalho, as seguintes atividades foram necessárias:

1. Realizar uma pesquisa bibliográfica sobre AVAs mais utilizados, e dentre eles destacar o melhor a ser empregado;
2. Desenvolver uma ferramenta com interface gráfica para permitir a geração automática de *scripts* na linguagem OTCL;
3. Identificar a melhor forma de integração do NS2 com o AVA;
4. Desenvolver mecanismo que permite a visualização da simulação a partir do AVA;
5. Integrar a ferramenta desenvolvida com o AVA escolhido;
6. Testar as funcionalidades da ferramenta e a comunicação com os Agentes;

Na próxima seção, serão apresentados os principais trabalhos relacionados com a pesquisa desenvolvida.

1.5 ORGANIZAÇÃO DO TRABALHO

Para um melhor entendimento do trabalho, além da Introdução, o texto da dissertação está organizado da seguinte forma:

- Capítulo 2 – **TRABALHOS RELACIONADOS**: apresenta os trabalhos relacionados com a pesquisa;
- Capítulo 3 – **REFERENCIAL TEÓRICO**: apresenta as ferramentas NS2, HTML5, o AVA Moodle e uma contextualização sobre Agentes e Sistemas Multiagentes;

- Capítulo 4 - **ARQUITETURA E IMPLEMENTAÇÃO DA PROPOSTA**: descreve as principais contribuições deste trabalho, suas características e o processo de desenvolvimento;
- Capítulo 5 – **CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS**: apresenta as conclusões e as recomendações para trabalhos futuros.

2 TRABALHOS RELACIONADOS

O uso de ferramentas de geração de *scripts* e ferramentas de simulação inseridas em AVA já vem sendo exploradas há alguns anos. Assim, este Capítulo relaciona algumas dessas ferramentas.

2.1 Ferramentas com interface gráfica para geração de script

O uso de ferramentas para auxiliar na criação de cenários com o simulador NS2 já vem sendo explorado há alguns anos. Em um contexto aproximado à pesquisa apresentada neste trabalho, algumas iniciativas, listadas a baixo, foram realizadas para desenvolver uma interface gráfica com a função de gerar *scripts* na linguagem OTCL para NS2.

2.1.1 GENESI

O GENESI (COUTINHO et al., 2005) é um *software* que tem como proposta central a geração de *scripts* em OTCL, por meio de um gerenciador gráfico, com retorno de dados da simulação. Trabalha com três tipos de rede: WANs, LANs ou redes WAN/LAN, tendo como objetivo auxiliar no processo de criação e configuração de cenários de redes.

A partir do momento em que o usuário passa a utilizar o GENESI, muitos detalhes de código começam a ser abstraídos, uma vez que recursos de construção baseados em arrastar e soltar facilitam a concepção e visualização do modelo, sem a necessidade de preocupação com o código (COUTINHO et al., 2005).

O projeto GENESI tem como desvantagens a escassez de cenários que podem ser confeccionados, e não permite que a animação da simulação seja feita na própria ferramenta, tendo que recorrer a visualizadores externos do NS2. Ele também não foi disponibilizado para uso, sendo posteriormente descontinuado.

2.1.2 NSGRAPH

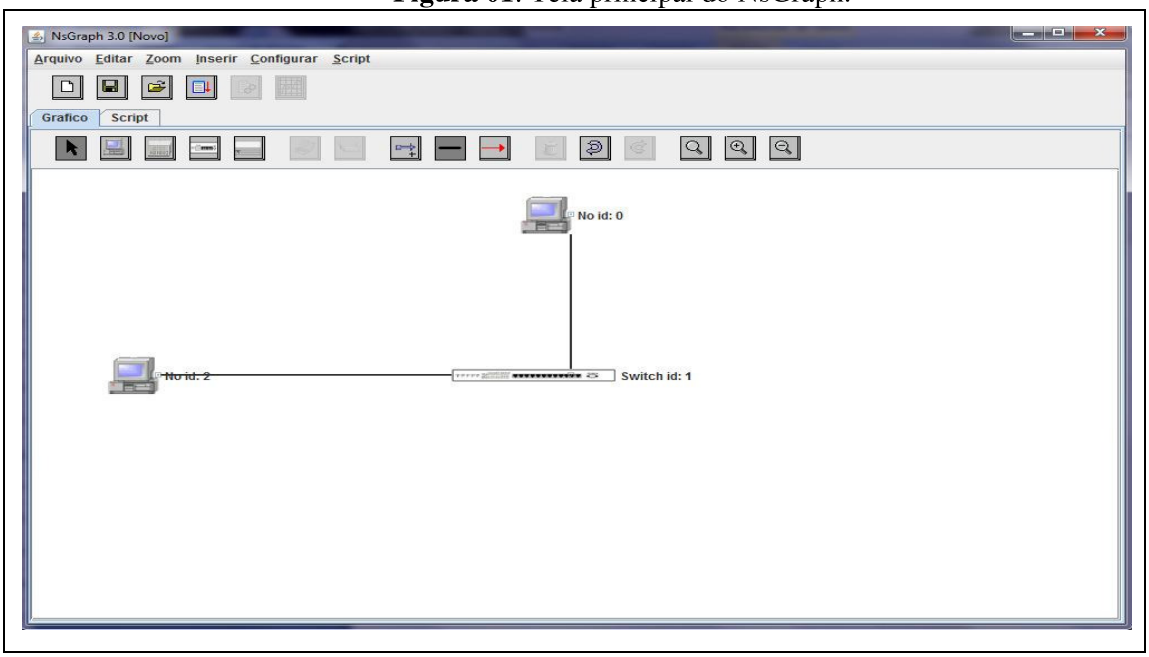
Desenvolvido na linguagem Java e com auxilio da biblioteca *Jgraph*, o NsGraph foi concebido com o propósito de facilitar a construção e configuração de cenários de simulação de forma automatizada (PONTES; LIMA, 2006). A Figura 01, ilustra a tela principal do NsGraph.

É possível observar na Figura 01 que a ferramenta possibilita o planejamento e modelagem da simulação por meio de uma interface gráfica que facilita a compreensão dos cenários a partir dos componentes da simulação. Com o NsGraph no processo de planejamento de uma simulação, a criação da topologia é o primeiro passo a ser dado (PONTES; LIMA, 2006).

O NsGraph oferece ao usuário uma área de desenvolvimento de cenários onde o mesmo pode modelar, configurar os parâmetros e executar o simulador. Além disso, possibilita a integração da simulação junto às demais ferramentas auxiliares ao NS2 (PONTES; LIMA, 2006).

Para executar uma simulação com o NsGraph primeiramente é confeccionado um cenário como ilustrado na Figura 01 e posteriormente o usuário escolhe a opção "Script/Executar NS", para que dessa forma o NS2 seja chamado e execute o arquivo OTCL gerado.

Figura 01: Tela principal do NsGraph.



Fonte: Manual do NsGraph (PONTES; LIMA, 2006)

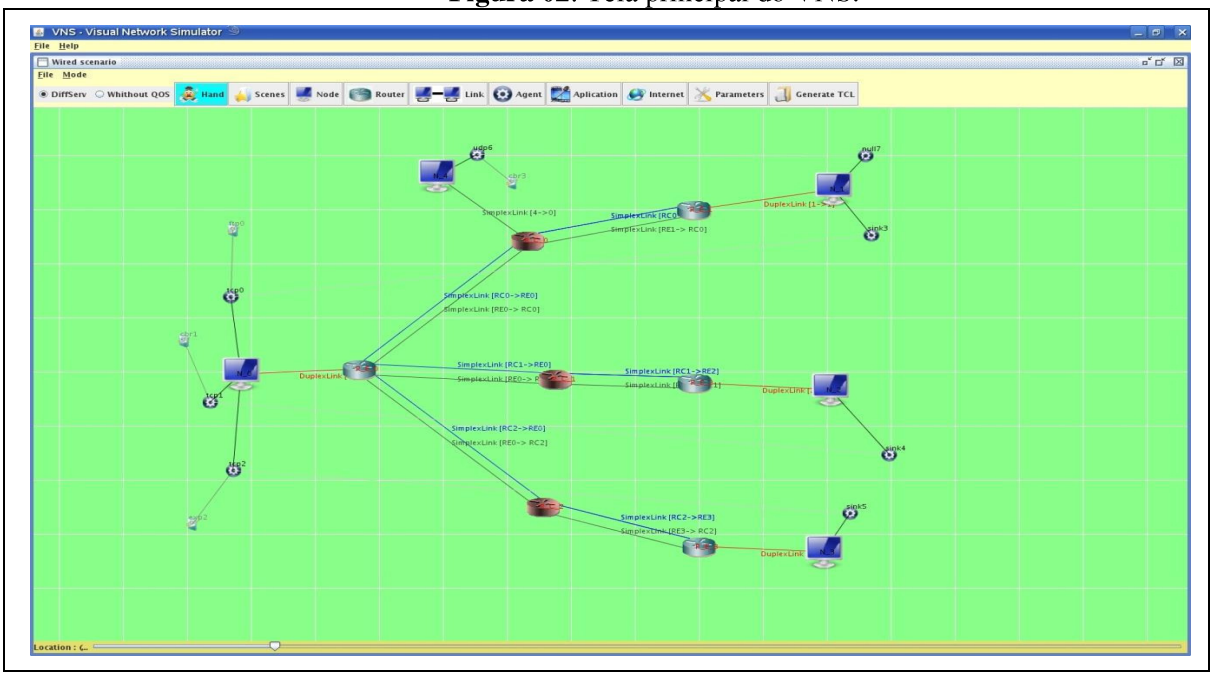
Assim como o GENESI, o NsGraph possui como desvantagem a falta de um módulo próprio de animação da simulação, sendo necessário a utilização do visualizador gráfico externo *Network Animator* (NAM) do NS2.

2.1.3 VISUAL NETWORK SIMULATOR (VNS)

Desenvolvido por Marques et al. (2009), o *Visual Network Simulator* (VNS), foi desenvolvido com o objetivo de facilitar a criação de cenários com suporte a QoS para ser executado com o NS2.

A modelagem da topologia da rede é realizada pela utilização do relacionamento dos objetos existentes entre os *hosts* e roteadores, tornando a simulação mais intuitiva e simples (Figura 02). Outra característica importante do VNS é o suporte a modelagem de cenários simples de redes, ou arquiteturas mais complexas tais como DiffServ com suporte ou não a QoS.

Figura 02: Tela principal do VNS.



Fonte: Manual do VNS (MARQUES et al., 2009)

Além de permitir a criação da simulação de cenários de redes com suporte a QoS o VNS também visa proporcionar um conjunto simples de ícones para representar os componentes de uma rede. Exemplos desses ícones são *hosts* ou roteadores representados por

símbolos diferentes como pode ser observado na Figura 02. No contexto de DiffServ, os roteadores de núcleo e de borda devem ser diferenciados.

A desvantagem do VNS é que este trabalha apenas com a simulação de *hosts* e roteadores, não permitindo a utilização e configuração de outros equipamentos que são de grande importância em redes de computadores, como *switchs*, *hubs*, dentre outros. Pode-se destacar também que como o GENESI e o *NsGraph*, para o VNS executar a animação da simulação, é necessário a utilização de módulos adicionais como o visualizador NAM.

2.2 Ferramentas de simulação integradas em AVA

Nesta subseção serão listadas algumas iniciativas no contexto do uso de ferramentas de simulação integradas a ambientes virtuais de aprendizagem, mostrando que o uso dessa integração contribui significativamente para o processo de ensino/aprendizagem.

2.2.1 STOP

Utilizado para simulação de sistemas do setor elétrico, o STOP (Sistema Simulador para Treinamento Presencial e a Distância da Proteção de Sistemas Elétricos) (SILVA et al., 2011) tem como objetivo permitir a capacitação e treinamento de funcionários e estudantes do setor elétrico por meio do ensino presencial e a distância via Internet. A tela inicial do STOP é ilustrada na Figura 03.

Figura 03: Tela principal do STOP.

Fonte: Manual do STOP (SILVA et al., 2011)

Sendo integrado ao *Modular Object-Oriented Dynamic Learning Environment* (Moodle) possibilita a simulações de situações reais proporcionando um ensino/aprendizagem mais eficaz. O STOP foi desenvolvido na forma de um bloco para ser utilizado dentro do Moodle como um laboratório virtual proporcionando ao usuário as seguintes funcionalidades:

- Especificação da relação nominal dos transformadores de corrente, permitindo ao aluno a escolha de equipamentos comercialmente utilizados nos sistemas elétricos;
- Verificação da correção do dimensionamento do equipamento utilizado;
- Implantação de critérios de ajuste e métodos de cálculo de cada função de proteção;
- Geração de coordenograma das proteções, após a definição das curvas e ajustes dos relés de sobrecorrente;

2.2.2 BOCA-LAB

Desenvolvida por Sirotheau et al. (2011), o BOCA-LAB surgiu da extensão de um sistema utilizado em maratonas de programação – o BOCA (CAMPOS; FERREIRA, 2004), tendo como objetivo contribuir para uma melhor compreensão do estudante no aprendizado de programação. Foi integrada ao Moodle propiciando uma maneira de visualizar e simular programas, permitindo a combinação de técnicas de avaliação da complexidade do código para dar um melhor *feedback* das atividades.

O BOCA-LAB é capaz de compilar e executar programas escritos em diversas linguagens de programação. Os programas submetidos são avaliados quanto a erros de compilação e execução em um processo automático, além de tratar de problemas de plágio.

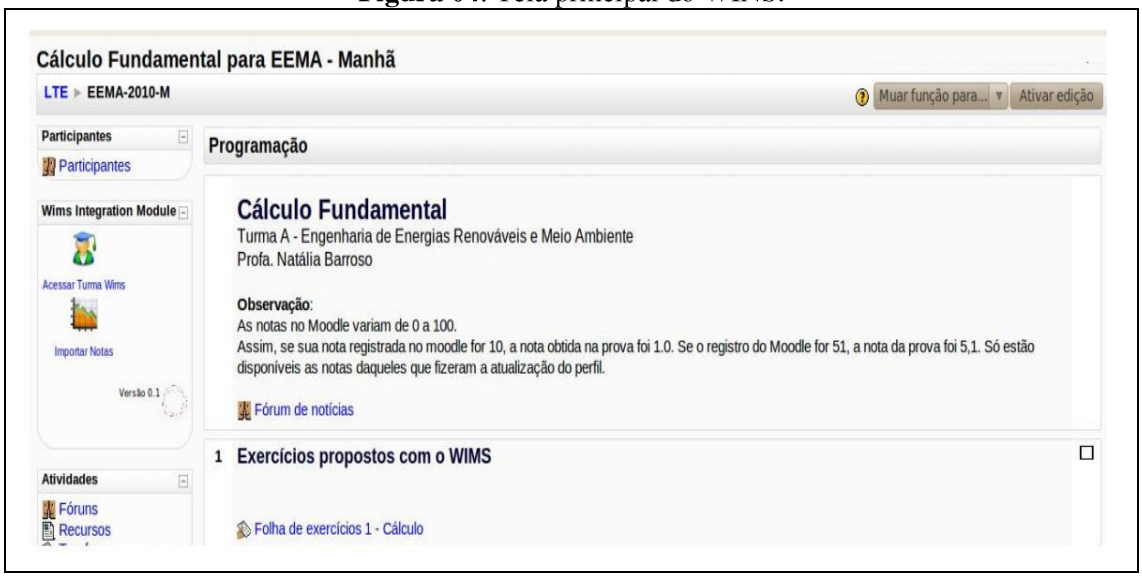
Outra importante iniciativa que pode ser destacada é a encontrada em Tavares et al. (2010), que trata da integração do ambiente WIMS ao Moodle para avaliações interativas, que são capazes de rastrear informações relevantes sobre o comportamento do aluno durante a realização de exercícios.

2.2.3 WINS

Utilizado para a disciplina de cálculo diferencial, a arquitetura de integração do WIMS ao Moodle é composta de três blocos, ao qual foi adicionado um módulo cliente permitindo ao professor acessar os serviços disponíveis em um componente chamado *Wims Integration Module* (TAVARES et al., 2010).

Os alunos por sua vez, acessam diretamente os exercícios propostos no WIMS como uma atividade disponibilizada no Moodle, sem a necessidade de fazer um segundo *login* ou navegar nas opções do ambiente externo. A Figura 04 ilustra a interface do WINS para uma disciplina de cálculo fundamental.

Figura 04: Tela principal do WINS.



Fonte: Manual do WINS (TAVARES et al., 2010)

2.3 CONSIDERAÇÕES FINAIS

O trabalho apresentado nesta pesquisa tem como vantagens, em relação às ferramentas apresentadas anteriormente, fornecer:

- Um maior número de cenários por meio da simulação de vários tipos de equipamentos de redes, não se restringindo apenas a *hosts* e servidores como o VNS;
- Modelar e configurar links de comunicação utilizando protocolos da camada de transporte;
- Utilizar diferentes tipos de aplicações nas simulações;
- Verificar o comportamento dos pacotes entre os equipamentos de redes;
- Criar um módulo próprio para visualizar a simulação na própria ferramenta.

Outro diferencial que pode ser destacado dos demais trabalhos relacionados é que como a ferramenta é apresentada por meio de um AVA, ela dispõe da utilização da interatividade e de recursos (chats, fóruns, lições) que este oferece.

A seguir serão apresentados alguns trabalhos que utilizam ferramentas de simulação integradas com AVA.

3 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico utilizado para o desenvolvimento deste trabalho. Para isso, o mesmo encontra-se organizado da seguinte forma: a seção 3.1 apresenta alguns exemplos de ferramentas de simulação de redes; na seção 3.2 é descrita a linguagem HTML5, utilizada no desenvolvimento do bloco; a seção 3.3 apresenta alguns ambientes virtuais de aprendizagem; a seção 3.4 faz uma breve descrição do mecanismo utilizado para integração do NS2 ao Moodle; e a seção 3.5 apresenta as considerações finais do capítulo.

3.1 FERRAMENTAS DE SIMULAÇÃO

Em redes de computadores, existem várias contribuições de autores no desenvolvimento de simuladores para dar apoio ao processo de aprendizagem. Dentre eles pode-se destacar:

- **CISCO PACKET TRACER** – ferramenta de aprendizado que suporta uma vasta gama de simulações físicas e lógicas em *hardware* cisco. Fornece ferramentas de visualização de enlaces, pacotes, tráfego de redes (CISCO, 2014);
- **OPNET** – ferramenta que fornece um ambiente virtual para modelagem, análise e previsão do desempenho das infraestruturas de TI, incluindo aplicações, servidores e tecnologias de rede (RIVERBED, 2014);
- **NETSIMK** – simulador que permite criar redes com roteadores e *switches* cisco, possuindo a função de Calculadora de IP integrada (NETSIMK, 2014);
- **NS2** (*Network Simulator* 2) – desenvolvido pela Universidade de *Berkeley* simulador para eventos discretos orientado para a pesquisa em redes de computadores (Oliveira et al., 2012);

Desta forma, para o desenvolvimento desta pesquisa de dissertação foi utilizado o simulador NS2 por sua grande utilização no meio acadêmico e por características que serão apresentadas na subseção seguinte.

3.1.1 NS2

Como discutido no Capitulo 1, o *Network Simulator* 2 (NS2), destaca-se por ser um dos simuladores mais utilizados no meio acadêmico, possuindo várias contribuições de pesquisadores da área de redes. É um simulador de eventos discreto que possibilita a simulação de vários tipos de cenários com diversas configurações e utilizando diferentes tipos de protocolos (OLIVEIRA et al., 2012).

Para exemplificar melhor essas características, esta seção está dividida em 4 subseções: na subseção 3.1.1 apresenta uma contextualização sobre o NS2; a subseção 3.1.2 apresenta os primeiros passos para criar uma simulação; na subseção 3.1.3 discute um pouco sobre roteamento; e na subseção 3.1.4 aborda o protocolo TCP.

3.1.1.1 CONTEXTUALIZAÇÃO

O NS2 foi desenvolvido pela Universidade de *Berkeley* utilizando as linguagens de programação OTCL e C++, ambas orientadas a objetos. Criado em 1989, o simulador surgiu a partir do *software* REAL *Network Simulator* (OLIVEIRA et al., 2012). Atualmente seu desenvolvimento e distribuição são mantidos pelo *Information Sciences Institute* (ISI) e financiado pela *Defense Advanced Research Projects Agency* (DARPA) e pela *National Science Foundation* (NFS) com o apoio de universidades como *University of Southern California* (USC) e os institutos *Xerox Palo Alto Research* (PARC) e *Lawrence Berkeley National Laboratory* (LBNL).

O NS2 é uma ferramenta dirigida a eventos que trabalha simulando vários tipos de redes e com suporte a diversos tipos de tráfegos. Foi inicialmente desenvolvido para ambiente *Unix*, sendo posteriormente portado para outras plataformas, inclusive sistema *Windows* por meio de emuladores (OLIVEIRA et al., 2012).

Uma grande vantagem do NS2 reside no fato de ser livre, ou seja, totalmente gratuito e com código fonte aberto, o que permite ao usuário proceder com os ajustes que julgar necessários. O simulador oferece suporte à simulação de um grande número de tecnologias de rede (com e sem fio), diferentes cenários, diversos escalonadores, políticas de fila e com diversas distribuições estatísticas (OLIVEIRA et al., 2012).

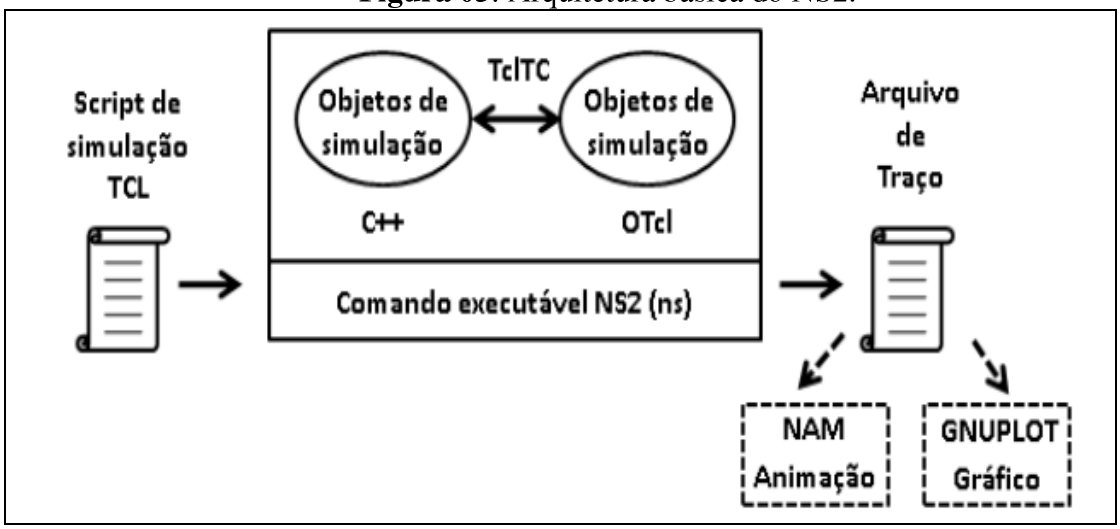
A programação do NS2 é feita com duas linguagens: C++ para a estrutura básica (protocolos, agentes, etc) e OTCL para uso como *frontend* (ISSARIYAKUL; HOSSAIN, 2008). O motivo para utilizar duas linguagens de programação baseia-se em duas diferentes necessidades:

- Existe a necessidade do uso de uma linguagem mais robusta para manipulação de *bytes*, pacotes e para implementar algoritmos que rodem um grande conjunto de dados. Nesse contexto a C++, que é uma linguagem compilada e de uso tradicional, mostra-se mais eficaz.
- Durante o processo de simulação, ajustes serão necessários com certa frequência. Dessa forma, para cada alteração feita é preciso testar todo projeto novamente, seja pela alteração de um parâmetro no tamanho do enlace, ou por acrescentar um novo nó a rede. O uso da linguagem OTCL, que é interpretada, evita esse desgaste por parte do usuário, pois há uma simplificação no processo interativo de alterar e reexecutar o modelo (OLIVEIRA et al., 2012).

Na Figura 05 ilustra a simplificação que a linguagem OTCL proporciona no desenvolvimento de uma simulação de redes. Na imagem é possível observar que o arquivo é repassado ao interpretador através do comando "*ns*", e posteriormente à sua execução, será gerado um arquivo de traço que contém todas as informações da simulação.

Esse arquivo de traço é utilizado por ferramentas como o NAM ou GNUPLOT para realizar a animação ou a criação de estatísticas referentes a simulação (OLIVEIRA et al., 2012).

Figura 05: Arquitetura básica do NS2.



Fonte: Adaptado de (ISSARIYAKUL; HOSSAIN, 2008).

A programação em OTCL requer conhecimentos sobre a sintaxe utilizada pelo NS2. A seguir são apresentadas as principais classes que são comuns à maioria das simulações (OLIVEIRA et al., 2012).

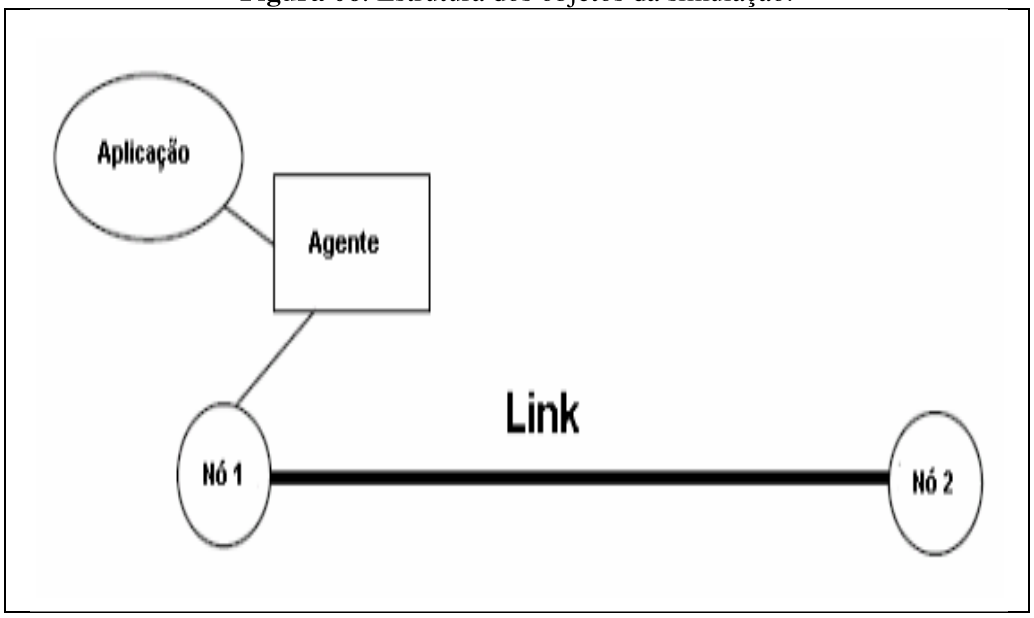
3.1.1.2 CRIANDO UMA SIMULAÇÃO

A criação da topologia da rede é um dos pontos primordiais de uma simulação. É preciso realizar um planejamento cuidadoso para que os nós e enlaces representem um cenário mais realístico (OLIVEIRA et al., 2012). No NS2 a topologia é construída utilizando nós que são interconectados por meio de enlaces.

Em cada enlace são configurados eventos escalonados (ISSARIYAKUL; HOSSAIN, 2008) para a comunicação entre os dispositivos da rede. Entre os nós e enlaces podem existir agentes que realizam a transmissão entre estes nós proporcionando a geração de diferentes pacotes para comunicação.

A fonte de tráfego, ilustrada na Figura 06, é uma aplicação associada a um agente particular em um nó, onde para cada agente é necessário possuir um tipo de receptor que receberá os pacotes enviados a este agente, podendo ser agentes do tipo TCP ou UDP (ISSARIYAKUL; HOSSAIN, 2008).

Figura 06: Estrutura dos objetos da simulação.



Fonte: Adaptado de (ISSARIYAKUL; HOSSAIN, 2008).

No caso do agente TCP (*Transmission Control Protocol*) esse receptor chama-se *sink* e tem a responsabilidade de gerar os pacotes de reconhecimento ACK (*Acknowledge*) (ISSARIYAKUL; HOSSAIN, 2008). Nos casos em que o tipo de agente é o UDP (*User Datagram Protocol*) o receptor será apenas configurado como *null* (OLIVEIRA et al., 2012).

Para criar uma simulação, como ilustrada na Figura 06, é preciso escrever um *script* na linguagem OTCL. Para isso é preciso executar os seguintes passos descritos no Algoritmo 01:

Algoritmo 01: Passos para iniciar uma simulação.

1	*criar objeto simulador*
2	*abrir arquivo de traço*
3	*criar topologia da rede*
4	*inserir agentes de transporte*
5	*conectar os agentes de transporte*
6	*gerar tráfego*
7	*criar eventos de simulação*
8	*fechar animação*
9	*executar animação*

Cada passo descrito no Algoritmo 01 será explicado, a seguir, na forma de *scripts* na linguagem OTCL.

Para iniciar qualquer simulação, primeiramente é preciso ajustar a variável que identifica o início da simulação, conforme a linha 1 do Algoritmo 02. Este comando define a geração de uma instância do objeto simulador e a associa com a variável "*ns*", realizando as tarefas de: inicializar o formato dos pacotes, criar um escalonador de eventos e selecionar o formato padrão de endereçamento.

Algoritmo 02: Criação dos objetos de simulação.

1	*set ns [new simulator]*
2	*set nf [open out.nam w]*
3	*$ns namtrace-all $tf*
4	*set tf [open out.tr w]*
5	*$ns trace-all $tf*

Com o NS2, é possível gerar um arquivo que servirá de base para a construção de uma animação gráfica de acordo com o cenário utilizado. No entanto, essa é uma etapa opcional e pode ser eliminada caso não seja necessário uma animação gráfica.

Caso exista a necessidade da animação da simulação, basta que no *script* seja adicionado o parâmetro de chamada de um dos módulos externos do NS2, conforme descrito na linha 2 da Algoritmo 02. Dessa forma será criado um arquivo ("*out.nam*") para salvar todos os passos da simulação (OLIVEIRA et al., 2012).

O próximo passo da configuração dos parâmetros de animação, conforme a linha 3 do Algoritmo 02, é especificar ao simulador que toda gravação da simulação deve ser feita no arquivo "*out.nam*" criado na linha anterior. Similarmente, pode-se usar a função "*trace-all*" para gravar arquivos de traço com informações utilizadas para análises da simulação, como é observado nas linhas 4 e 5 do Algoritmo 02.

Após realizada a criação dos objetos da simulação, é necessário definir a criação dos nós que farão parte da topologia da rede, conforme descrito nas linhas 1 e 2 do Algoritmo 03. Os parâmetros "*set n0[$ns node]*" e "*set n1[$ns node]*" definem a criação de dois nós para o objeto simulador "*ns*", criado no inicio da simulação, conforme o Algoritmo 02. Em seguida, é necessário configurar os enlaces para que os nós possam se comunicar, sendo que a sintaxe utilizada varia de acordo com a tecnologia de rede empregada.

Algoritmo 03: Criação da topologia da rede.

```
1  set n0 [$ns node]
2  set n1 [$ns node]
3  $ns duplex-link $n0 $n1 1 Mb 10ms DropTail
4  set udp0 [new Agent/UDP]
5  $ns attach-agent $n0 $udp0
6  set null0 [new Agent/Null]
7  $ns attach-agent $n1 $null0
8  $ns connect $udp0 $null0
```

Para o *script* do Algoritmo 03 é utilizado um enlace *full-duplex* (ISSARIYAKUL; HOSSAIN, 2008), conforme a linha 3, entre os nós "*n0*" e "*n1*". O comando da linha 3 define ainda a criação de um enlace com capacidade de 1 Mbit/s e retardos de 10 ms. O último parâmetro da especificado na linha 3, define o tipo de fila utilizado, onde *DropTail* (ISSARIYAKUL; HOSSAIN, 2008) representa o algoritmo FIFO (*First In, First Out*).

Posteriormente, é necessário a criação de agentes, que são componentes da arquitetura do NS2 responsáveis pela simulação dos protocolos de transporte TCP e UDP (OLIVEIRA et al., 2012), onde estes agentes utilizam um canal de comunicação entre o nó transmissor e receptor. No Algoritmo 03 na linha 4 é especificado que será utilizado o protocolo UDP, sendo este anexado ao nó "*n0*" conforme a linha 5.

Em seguida, deve-se criar um agente receptor cuja função é apenas receber os pacotes enviados ao nó "*n0*". O código representado na linha 6 mostra a configuração de um receptor do tipo *null*, sendo, posteriormente na linha 7, anexado ao nó "*n1*". Na linha 8 é efetivamente estabelecido o canal de comunicação entre o emissor e o receptor a nível de transporte.

Após o estabelecimento do canal de comunicação é necessário que um tráfego de uma determinada aplicação seja gerado e transmitido por este. O Algoritmo 04 exemplifica o *script* de criação dos geradores de tráfego e conexões com os agentes da camada de transporte. Nas linhas 1, 2 e 3, é configurado um tráfego CBR (*Constant Bit Rate*) (ISSARIYAKUL; HOSSAIN, 2008), que geralmente é utilizado para *streaming* de áudio e/ou vídeo.

Algoritmo 04: Geradores de tráfego e comunicação dos agentes da camada de transporte.

```
1   set cbr0 [new Application/Traffic/CBR]
2   $cbr0 set packetSize_ 50010
3   $cbr0 set interval_ 0.005
4   $cbr0 attach-agent $udp0
5   $ns at 0.5 "$cbr0 start"
6   $ns at 4.5 "$cbr0 stop"
7   $ns at 5.0 "finish"
8   proc finish {} {global ns nf $ns flush-trace close $nf exec nam out.nam & exit 0}
9   $ns run
```

Alguns parâmetros devem ser especificados para configuração desse tipo de aplicação, como o tamanho do pacote (em *bytes*), o intervalo de transmissão (em segundos), entre outros. Feito isso, é preciso anexar na aplicação criada um tipo de agente da camada de transporte, conforme a linha 4 do Algoritmo 04.

Uma das etapas mais importantes em uma simulação é definir o comportamento que esta terá durante a sua execução, ou seja, os eventos que ocorrerão. Como o NS2 é um simulador orientado a eventos, a definição da dinâmica na topologia pode ser feita mais facilmente (OLIVEIRA et al., 2012).

Para isso, atribui-se um tempo total de simulação e, durante este período, pode-se invocar eventos específicos, como início de tráfego, quedas ou perdas de dados nos enlaces, dentre outras características. No código das linhas 5, 6 e 7, a aplicação CBR é iniciada no tempo 0.5s e finalizada em 4.5s. Após 5.0s, é chamado o procedimento "*finish*", que encerra a simulação.

Concluída a etapa acima, é necessário declarar o fim da simulação, através do procedimento apresentado na linha 8, que será responsável por: fechar e limpar os arquivos de traços já existentes, abrir o NAM (caso este seja utilizado) e encerrar efetivamente a simulação. Por último, o comando "*$ns run*" conforme a linha 9 do Algoritmo 04, é responsável por executar a simulação e iniciar o escalonador de eventos do NS2 (OLIVEIRA et al., 2012).

3.1.1.3 ALGORITMOS DE ROTEAMENTO

Para que ocorra a transferência de pacotes de um nó emissor para um receptor existe a necessidade de determinar que caminho os pacotes deverão seguir. Essa tarefa é realizada na camada de rede, mais precisamente pelos protocolos de roteamento desta camada (OLIVEIRA et al., 2012).

O NS2 permite a simulação de diversos algoritmos de roteamento, como *unicat* e *multicast* (ISSARIYAKUL; HOSSAIN, 2008). Para o *unicast*, é utilizada uma comunicação ponto-a-ponto, quando um emissor cria um canal de comunicação separado para cada destinatário. Por exemplo, se a transmissão é feita entre um nó emissor e três nós destinatários, o emissor enviará três vezes o mesmo arquivo um para cada destinatário.

Analisando como esta comunicação é feita, percebe-se que há uma sobrecarga do emissor à medida que o número de receptores cresce, tornando o congestionamento e o atraso de dados mais intensos (OLIVEIRA et al., 2012). Para o *unicast* podem ser utilizados os seguintes tipos de algoritmos: *Static*, *Session*, *Distance Vector* (DV) e *multiPath* (ISSARIYAKUL; HOSSAIN, 2008).

No *multicast* um nó envia os dados para vários nós através de uma técnica conhecida como grupos. Os vários nós que desejam receber os dados de um nó transmissor ingressam neste grupo através de um endereço de grupo.

Assim a transmissão será recebida apenas pelos nós que fizerem parte do grupo de recebimento num dado instante. Para o *multicast* o NS2 possibilita os seguintes tipos de algoritmos: *ctrMcast* (*Multicast* Centralizado), *DM/pimdm* (Modo Denso usando protocolo *pimdm*), *DM/dvmrp* (Modo Denso usando protocolo *dvmrp*) e *BST* (Modo Árvore compartilhada bidirecional) (ISSARIYAKUL; HOSSAIN, 2008).

A seguir, serão apresentados mais detalhes sobre a utilização do protocolo TCP no NS2, mostrando sua sintaxe básica na linguagem OTCL.

3.1.1.4 PROTOCOLO TCP

O protocolo TCP foi projetado para oferecer um fluxo de *bytes* fim-a-fim confiável (ISSARIYAKUL; HOSSAIN, 2008). Ele realiza controle de fluxo e congestionamento,

retransmissão, entre outras funções (OLIVEIRA et al., 2012), garantindo ainda que os dados chegarão de forma correta e ordenada ao seu destino. Para criar um agente do tipo TCP, usa-se a sintaxe do Algoritmo 05.

Algoritmo 05: Utilizando protocolo TCP.

```
1   set tcp0 [new Agent/TCP]
2   $ns attach-agent $n0 $tcp0
3   set sink0 [new Agent/TCPSink]
4   $ns attach-agent $n1 $sink0
5   $ns connect $tcp0 $sink0
```

A linha 1 do Algoritmo 05, especifica a criação de um agente TCP referenciado pela variável "*tcp0*". A linha 2 associa o agente ao nó "*n0*", que será o nó emissor de pacotes. Em seguida, da mesma forma que no agente UDP, é necessária a configuração de um receptor para receber os pacotes enviados na rede.

No protocolo TCP o receptor é um agente do tipo TCPSink, responsável por enviar os pacotes de reconhecimento ACK (*Acknowledgement*) (ISSARIYAKUL; HOSSAIN, 2008). A sintaxe de configuração para este agente é mostrada nas linhas 3 e 4 do Algoritmo 05. Posteriormente é realizada a conexão entre os dois agentes (TCP e TCPSINK) conforme indicado na linha 5 do Algoritmo 05.

Como visto, o NS2 permite a simulação de diversas variações do protocolo TCP. A seguir, serão abordadas três das principais variações segundo Oliveira et al. (2012).

- TCP *Tahoe* - essa versão do TCP tem como característica trabalhar com a técnica de partida lenta, ou seja, quando há uma perda, ele começa a retransmitir a uma taxa inicial lenta e vai crescendo exponencialmente. Essa é a variação padrão adotada pelo agente TCP do NS2. A sintaxe utilizada para declarar essa variação é *Agent/TCP*;
- TCP *Reno* – é muito similar ao agente TCP *Tahoe*, exceto pelo fato de incluir uma recuperação rápida, onde a janela de congestionamento corrente é acrescida pelo número de ACK duplicado que o emissor TCP recebeu antes de receber um novo ACK. A sintaxe utilizada para declarar essa variação é *Agent/TCP/Reno*;
- TCP *Vegas* – possui uma política de reconhecimentos igual ao TCP RENO, porém foi desenvolvido para aumentar a taxa de transferência e reduzir a perda de segmentos em períodos de congestionamento. A sintaxe utilizada para declarar essa variação é *Agent/TCP/Vegas*;

A seguir, será apresentada a linguagem utilizada no desenvolvimento da ferramenta de integração do AVA com o NS2, destacando a motivação do seu uso e suas principais características.

3.2 A LINGUAGEM HTML5

A linguagem HTML5 tem como objetivo facilitar e melhorar a estruturação e a apresentação de conteúdos na *web* por meio de novas *tags*, incorporando funcionalidades como: uso de Canvas, executar vídeos, possibilitar o uso de *drag-and-drop* (SILVA JR; FIRMINO, 2010) que antes só era possíveis com o uso de *plugins* proprietários, dentre outras.

Neste sentido, como a pesquisa presente trata da integração de uma ferramenta de simulação em um AVA, e este sendo desenvolvido com uma linguagem *web*, a utilização do HTML5 possibilitará uma simplificação na integração da ferramenta desenvolvida com o AVA. Além disso, como um dos objetivos do trabalho apresentado é possibilitar a animação gráfica das simulações criadas, a utilização do elemento Canvas se mostra como uma boa alternativa para o desenvolvimento desta funcionalidade (SILVA JR; FIRMINO, 2010).

Para exemplificar melhor essas características, será apresentado a seguir na subseção 2.2.1 uma contextualização sobre a linguagem HTML5 e na subseção 2.2.2 um pouco mais sobre o elemento Canvas.

3.2.1 HTML5

Em maio de 2007, o *Word Wide Web Consortium* (W3C), reconsiderou sua decisão de encerrar o desenvolvimento da *Hypertext Markup Language* (HTML) em favor da *eXtensible HyperText Markup Language* (XHTML), tornando pública sua decisão de retornar os estudos para o desenvolvimento da HTML5, tendo como base o trabalho que já vinha sendo desenvolvido pela *Web Hypertext Application Tecnology Working Group* (WHATWG) (SILVA, 2011).

A essência do W3C tem sido melhorar a linguagem, dando suporte para os mais recentes recursos multimídia, enquanto a mantém facilmente legível por seres humanos e

consistentemente compreendidos por computadores e outros dispositivos (navegadores, *parsers*, etc.). De acordo com o W3C a *web* é baseada em 3 pilares: Um esquema de nomes para localização de fontes de informação na *web* (URI), um protocolo de acesso para acessar estas fontes (HTTP) e uma linguagem de hipertexto, para a fácil navegação entre fontes de informação (HTML).

Quando a versão anterior do HTML5 foi lançada, o W3C alertou os desenvolvedores sobre algumas boas práticas que deveriam ser seguidas ao produzir códigos *client-side*. Desde este tempo, assuntos como a separação de estrutura do código com a formatação e princípios de acessibilidade foram trazidos para discussões e à atenção dos fabricantes e desenvolvedores (SILVA, 2011).

Assim, um dos principais objetivos do HTML5 é facilitar a manipulação dos elementos possibilitando ao desenvolvedor modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final. Ao contrário das versões anteriores, o HTML5 fornece ferramentas para a *Cascading Style Sheets* (CSS) e o *JavaScript* (MORAZ, 2011) fazerem seu trabalho da melhor maneira possível (SILVA, 2011).

A linguagem permite, por meio de sua *Application Programming Interface* (API), a manipulação das características destes elementos, de forma que o *website* ou a aplicação continue leve e funcional. Ela também cria novas *tags* e possibilita modificações nas funções de outra. As versões antigas do HTML5 não continham um padrão universal para a criação de seções comuns e especificas como rodapé, cabeçalho, *sidebar*, *menus*, dentre outros. Não havia um padrão de nomenclatura de IDs, Classes ou *tags*, nem um método de captura automática das informações localizadas nos rodapés dos *websites*.

Antigamente, para que uma nova versão do HTML ou do CSS fosse lançada, todas as ideias listadas na especificação deveriam ser testadas e desenvolvidas para então serem publicadas para o uso dos *browsers* e para os desenvolvedores. Esse método foi mudado com o lançamento do HTML5 e do CSS3, sendo que agora, as duas tecnologias foram divididas em módulos. Isso quer dizer que a comunidade de desenvolvedores e os fabricantes de *browsers* não precisam esperar que todo o padrão seja escrito e publicado para utilizarem as novidades das linguagens.

A especificação para o HTML5 estabelece um Modelo de Conteúdo que vem a ser a descrição do tipo a ser inserido nos elementos, sendo que os conteúdos de cada elemento do HTML deverão estar de acordo com o que estabelece o modelo da W3C (MORAZ, 2011). Segundo Silva (2011), cada elemento pertence a uma ou mais categorias que agrupam

modelos de conteúdo, podendo ser dividida em: metadados, fluxo, seção, cabeçalhos, frase, incorporado, interativo, e formulário.

3.2.2 Elemento Canvas

O Canvas é um novo elemento do HTML5 que permite desenhar diretamente dentro de uma página *web*. Destina-se a delimitar uma área para criação dinâmica de: imagens, gráficos estáticos, gráficos dinâmicos, jogos, etc. A manipulação do elemento Canvas deve ser feita com uma linguagem de programação dinâmica, onde todo o trabalho de animação é proporcionado pelo *JavaScript* (SILVA, 2011).

O acesso ao conteúdo gráfico contido em Canvas, em navegadores que não suportam esse elemento, deve ser garantido com uso de conteúdo alternativo com a mesma função ou propósito do conteúdo gráfico. Tal conteúdo deve constar da marcação da página e ser inserido tendo o Canvas como seu *container* (SILVA, 2011).

Como dito anteriormente, o Canvas permite desenhar na página, dessa forma possibilita a atualização dinamicamente dos desenhos ou imagens inseridas, por meio de *scripts* ou atendendo às ações do usuário (SILVA JR; FIRMINO, 2010). As aplicações desenvolvidas com esta nova função do HTML5 vão de pequenos efeitos até grandes projetos como: jogos, efeitos dinâmicos em interfaces de usuário, editores de código, editores gráficos, efeitos 3D, etc. Na Figura 07 é possível observar um jogo bastante conhecido que foi desenvolvido utilizando os recursos da linguagem HTML5.

Figura 07: Jogo desenvolvido com HTML5.

Fonte: imagens do Google.
URL: https://www.singolo.com.br/blog/angry-birds/angry-birds-easter-ipad-2/

Em suma, o HTML5 é uma tecnologia que está em evolução e por isso existem diferentes implementações em diferentes navegadores (SILVA JR; FIRMINO, 2010). Pode ser destacado também que sua principal desvantagem está no fato de que o usuário precisa ter instalado em sua máquina versões recentes do navegador que este utiliza. Mesmo assim esta linguagem mostra-se promissora para o desenvolvimento de sistemas web que necessitam da utilização de recursos gráficos (SILVA JR; FIRMINO, 2010).

A seguir, será apresentado o ambiente virtual de aprendizagem (AVA) utilizado nesta pesquisa, destacando a motivação do seu uso e suas principais características.

3.3 AMBIENTES VIRTUAIS DE APRENDIZAGEM

É possível destacar vários ambientes virtuais de aprendizagem encontrados na Internet, que permitem a produção de conteúdos e canais variados de comunicação, bem como o gerenciamento de dados e controle total de informações, que podem comportar simuladores, como:

- **MOODLE** (*Modular Object-Oriented Dynamic Learning Environment*) – *software* de apoio à aprendizagem, executado em ambiente virtual, é utilizado principalmente num contexto de *e-learning* ou *b-learning*, o programa permite a criação de cursos *on-line*, páginas de disciplinas, grupos de trabalho e comunidades de aprendizagem (KUMAR et al., 2011);
- **AMADEUS** – é um sistema de gestão do aprendizado de segunda geração. Propõem canais para mediar à interação e colaboração entre tutores e aprendizes por meio de estilos de interação baseados em troca de artefatos (envio, visualização e entrega de matéria nas mais variadas mídias) e mensagens instantâneas ou assíncronas (fóruns, *chats*, e-mail). O projeto visa desenvolver um sistema de gestão de aprendizado de segunda geração, baseado no conceito de *blended learning* (AMADEUS, 2014);
- **TELEDUC** – é um ambiente de EaD pelo qual se pode realizar cursos por meio da Internet. Está sendo desenvolvido conjuntamente pelo Núcleo de Informática Aplicada à Educação e pelo Instituto de Computação da universidade Estadual de Campinas. Foi desenvolvido em PHP, *JavaScript* e *MySql* para ambientes *UNIX* e *LINUX*, possuindo licença livre (FRANCISCATO et al., 2008);

Assim, o trabalho presente optou pela utilização do AVA Moodle por características que serão apresentadas na subseção seguinte.

3.3.1 Moodle

O *Modular Object-Oriented Dynamic Learning Environment* (Moodle) é um sistema de código aberto destinado ao gerenciamento de cursos. Este AVA possibilita a disponibilização de conteúdos como textos, imagens, vídeos e áudios, além de permitir a criação de exercícios, questionários e uma série de outros recursos (KUMAR et al., 2011).

Em função de vantagens como estas, o Moodle foi escolhido como AVA para ser utilizado nesta pesquisa. Outra característica que colabora para o seu uso, reside no fato de que este AVA já vem sendo empregado como ferramenta de apoio nas disciplinas da graduação e de pós-graduação da Universidade do Estado do Rio Grande do Norte (UERN) para o qual este trabalho foi desenvolvido.

Assim, a finalidade desta seção é apresentar as principais características desse ambiente que justificam sua adoção como AVA, dessa forma as próximas subseções apresentam os principais elementos que dão suporte ao Moodle, começando por sua filosofia de desenvolvimento na subseção 3.3.1.1, em seguida e a subseção 3.3.1.2 descreve as suas características.

3.3.1.1 FILOSOFIA

O Moodle trabalha com uma perspectiva dinâmica da aprendizagem em que a pedagogia socioconstrutivista e as ações colaborativas ocupam lugar de destaque. Sendo assim, seu objetivo é permitir que processos de ensino-aprendizagem ocorram por meio não apenas da interatividade, mas, principalmente, pela interação, ou seja, privilegiando a construção/reconstrução do conhecimento, a autoria, a produção do conhecimento em colaboração com os pares e a aprendizagem significativa do aluno (KUMAR et al., 2011).

Essa linha de pensamento é baseada em quatro conceitos de aprendizagem: construtivismo, construcionismo, construtivismo social e comportamento conectado e

separado. O construtivismo sustenta que as pessoas constroem novos conhecimentos, ativamente, na medida em que interagem com o seu ambiente (MOODLE, 2012).

Já o construcionismo defende que a aprendizagem é particularmente efetiva quando constrói alguma coisa para outros utilizarem. No entanto, o construtivismo social estende as ideias do construtivismo e do construcionismo para um grupo social. Para as outras, criando, de forma colaborativa, uma pequena cultura de objetos compartilhados, com significados compartilhados (KUMAR et al., 2011).

Finalizando, o comportamento conectado e separado observa mais a fundo as motivações das pessoas numa discussão. O comportamento separado é quando alguém tenta permanecer objetivo, e tende a defender suas próprias ideias usando a lógica para encontrar falhas nas ideias dos seus oponentes.

Já o comportamento conectado é uma abordagem mais empática que aceita a subjetividade, tentando ouvir e fazer perguntas num esforço para entender o ponto de vista do outro (KUMAR et al., 2011). Além da filosofia que norteia o seu desenvolvimento, o Moodle possui várias particularidades que facilitam sua adoção como AVA. As Subseções a seguir apresentarão estas particularidades.

3.3.1.2 CARACTERÍSTICAS

O Moodle é uma ferramenta do tipo *Content Management System* (CMS), com finalidade educacional, sendo desenvolvida em *Hypertext Preprocessor* (PHP) (PHP, 2012), e há uma grande comunidade de profissionais que desenvolve novos módulos e também atua na correção de problemas que são encontrados por usuários.

Uma de suas principais vantagens sobre os outros AVA é o forte embasamento na Pedagogia Construcionista. A portabilidade do Moodle é outra de suas características fortes. Sem qualquer modificação, o ambiente pode ser utilizado nas plataformas: *Unix*, *Linux*, *Windows*, *Mac OS* e outros sistemas que suportem PHP (LIMA; FIALHO, 2009). Está disponível em 223 idiomas e disponibiliza uma API, com uma série de recursos para autenticação de usuário, manipulação de banco de dados, entre outros.

O Moodle também possui uma estrutura, em termos de programação, padronizada e muito definida. A interface *web* foi desenvolvida a partir das linguagens HTML e CSS, e a parte dinâmica (validação de formulários, conexão a banco de dados, armazenagem de

informações e outros) foi implementada com as linguagens PHP, *JavaScript* e *MySQL*. Sendo assim, conforme Lima (2009), as características do Moodle podem ser dividas em técnicas e pedagógicas. Ainda conforme o autor, de forma resumida as principais características são apresentadas na Tabela 01.

Tabela 01: Características técnicas e pedagógicas do Moodle.

Características Técnicas	Características Pedagógicas
Multiplataforma, podendo ser executado nos ambientes – *Unix*, *Linux*, *Windows*, *Mac OS* e qualquer outro sistema que suporte PHP;	Promove uma interação socioconstrutivista, que inclui colaboração e reflexão crítica, permitindo máxima interação e integração entre a comunidade virtual;
Instalação simples;	Adequado para cursos não presenciais bem como para complementar um curso presencial;
Software livre com código aberto;	Mostra descrição sumária dos recursos disponíveis, informando, inclusive, se estão disponíveis para acesso a provas, listas de exercícios e outras informações que requeiram segurança;
Desenvolvido de forma modular, permite flexibilidade para configurar ou remover funcionalidades em vários níveis;	

Fonte: Adaptado de (LIMA, 2009).

Por possuir uma estrutura modular, o Moodle disponibiliza diversas ferramentas que podem ser desenvolvidas na forma de módulos ou blocos de atividades (Calendário, *Login*, Usuários *online*, etc.) com conteúdos específicos que podem ser editados ou adaptados às necessidades do professor. Dessa forma esses blocos ou módulos possibilitam o desenvolvimento de novos recursos para o Moodle.

Para Lima (2009) estas ferramentas de atividades podem ser divididas em quatro grupos: comunicação e discussão (fóruns, chats e diálogos); construção coletiva (trabalhos, *workshops*, *wikis* e glossários); e pesquisa e opinião (teste, enquetes, referendos e questionários). As atividades listadas a seguir, são consideradas de uso mais comum em virtude de, em sua grande maioria, já virem incorporadas ao pacote de instalação do Moodle.

- Chats – é uma atividade que permite a interação on-line e simultânea (síncrona) entre os participantes de um curso;
- Fóruns – permite a criação de ferramentas de discussão, incluindo a possibilidade de classificar as mensagens;
- Glossários – destina-se a criação de dicionários de termos relacionados ao conteúdo exposto no curso;

- Lições – trata-se de uma atividade, em que perguntas e respostas são intercaladas com apresentações e arquivos de diferentes formatos;
- Questionários – viabiliza uma grande variedade de tipos de exercícios e avaliações on-line. Permite a criação de questões objetivas e dissertativas, além de fornecer opinião sobre erros e acertos;
- SCORM – conjunto de padrões que permite o desenvolvimento de OA, tendo como características principais a reusabilidade e a interoperabilidade;
- Testes – essa atividade possibilita a solicitação de atividades que devem ser realizadas on-line ou off-line;
- Wikis – possibilita que vários participantes construam coletivamente um hiperdocumento. Seu funcionamento se assemelha ao serviço disponibilizado na Wikipédia;

Além de todas essas características, o Moodle é utilizado para maximizar os espaços da aprendizagem, onde são realizadas ações pedagógicas numa lógica colaborativa, de interação e de ampliação, onde tanto o aluno como o professor poderá ter contato tanto no presencial como no virtual, dando uma dinâmica às relações de ensinar e aprender (PEREIRA; CHAVES, 2007).

Dando continuidade ao trabalho, na seção seguinte será apresentada as principais características para a utilização de agentes reativos como forma de realizar a comunicação entre o Moodle e o NS2.

3.4 AGENTES E SISTEMAS MULTIAGENTES

De acordo com Artero (2009) agentes são programas que executam um conjunto de operações no lugar de um usuário, utilizando uma representação do conhecimento que contém os objetivos desses usuários. Estes agentes podem ser classificados como:

- Agente Reativo Tabela – é uma estrutura mais simples de uma agente, na qual todas as percepções e ações possíveis estão relacionadas em uma tabela;
- Agente Reativo Simples – seleciona ações a serem executadas com base exclusivamente na percepção atual, não levando em consideração o histórico de percepções;

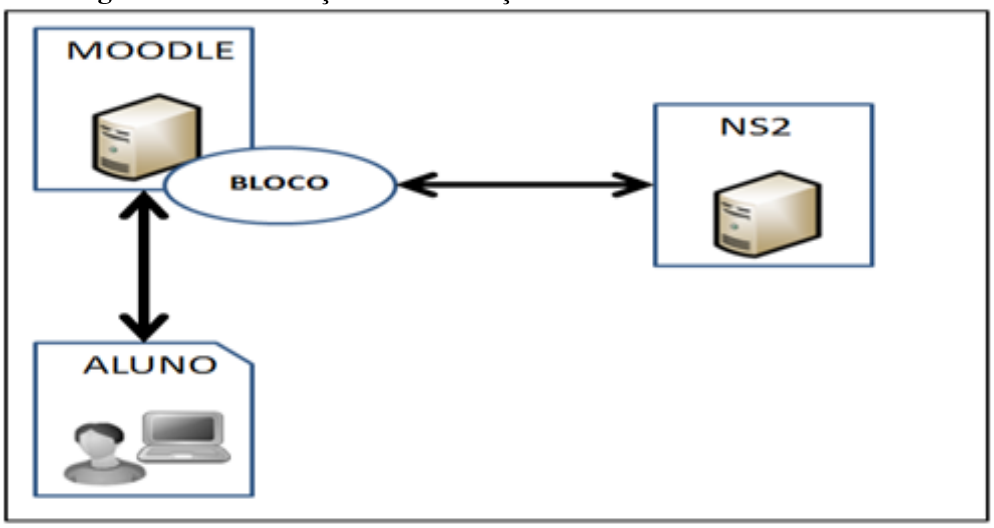
- Agente Reativo baseado em Modelo – controla o estado atual do mundo usando um modelo interno do ambiente a partir do seu histórico de percepções;
- Agente Reativo baseado em Objetivos – pondera suas ações levando em consideração a descrição do estado atual e os objetivos a serem alcançados; e
- Agente Reativo baseado em Utilidade – escolhe suas ações tentando sempre maximizar uma função de utilidade, mapeando um estado que descreve o grau de "felicidade" do agente caso aquele estado seja alcançado;

Uma vez tendo definido o conceito de agente, é possível definir o conceito de Sistemas Multiagentes (SMA). De acordo com Henderson-Sellers e Giorgini (2005), um SMA é um sistema composto de agentes cooperativos ou competitivos que interagem entre si para atingir um objetivo individual ou comum.

A utilização de SMA nesta pesquisa pode ser explicada com base na Figura 08. Na imagem é possível observar que as ferramentas utilizadas (Moodle e o NS2) não estão necessariamente instaladas no mesmo servidor, ou seja, fazem parte de um sistema distribuído.

Dessa forma a utilização de SMA possibilita alta interoperabilidade entre a comunicação dos agentes, proporcionando que as simulações realizadas sejam executadas e apresentadas de forma transparente ao usuário.

Figura 08: Visualização da distribuição das ferramentas NS2 e Moodle.



Para desenvolver os agentes foi utilizada a plataforma JADE, pois esta plataforma tem como principal característica a interoperabilidade existente na comunicação dos agentes e pela facilidade de desenvolvimento que esta fornece.

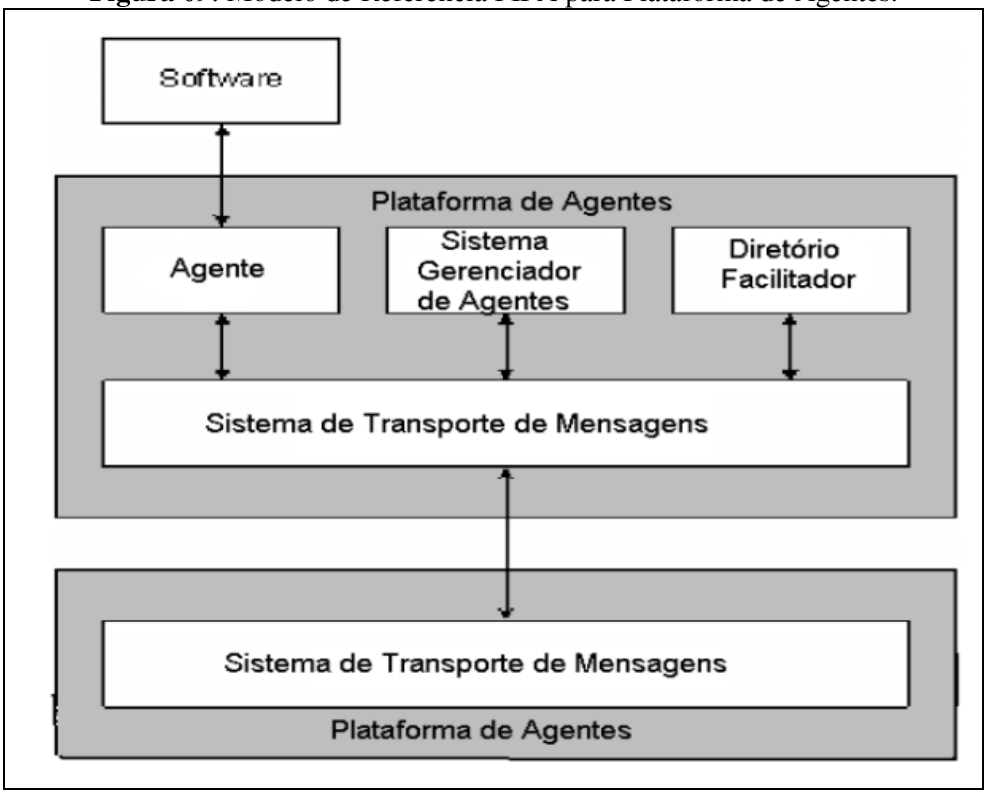
A seguir, será apresentada as principais características e funções utilizadas com a plataforma JADE para o desenvolvimento dos agentes desta pesquisa.

3.4.1 Plataforma JADE

O *Java Agent Development framework* (JADE) (BELLIFEMINE et al., 2007) é um ambiente para desenvolvimento de aplicações baseada em agentes conforme as especificações da *Foundation for Intelligent Physical Agents* (FIPA) (FIPA, 2011) para interoperabilidade entre SMA. Foi desenvolvido e suportado pelo CSELT (atual TILAB) da Universidade de Parma na Itália.

O principal objetivo do JADE (BELLIFEMINE et al., 2007) é simplificar e facilitar o desenvolvimento de SMA garantindo um padrão de interoperabilidade entre esses sistemas por meio de um conjunto de Agentes de Serviços, os quais tanto facilitam como possibilitam a comunicação entre Agentes, de acordo com as especificações da FIPA (Figura 09).

Figura 09: Modelo de Referência FIPA para Plataforma de Agentes.



Fonte: Adaptada de (BELLIFEMINI et al., 2007).

O Modelo de Referência FIPA para Plataforma de Agentes, conforme indicado na Figura 09, é composto de:

- *Agent* (Agente) - é o Agente propriamente dito, cujas tarefas serão definidas de acordo com o objetivo da aplicação. Encontra-se dentro de uma plataforma de Agentes (*Agent Platform*) e realiza toda sua comunicação com os demais agentes através de troca de mensagens e relaciona-se com aplicação externa (*software*);
- *Agent Management System* (AMS - Sistema Gerenciador de Agentes) - responsável por gerenciar o ciclo de vida dos Agentes na plataforma e prover um serviço de guia de endereços (*white pages*);
- *Directory Facilitator* (DF- Diretório Facilitador) - que funciona como um serviço de páginas amarelas com capacidades básicas de prover serviços desejados;
- *Message Transport System* (MTS - Sistema de Transporte de Mensagens) – também conhecido como canal de comunicação dos Agentes (*Agent Communication Channel-ACC*), é o agente responsável por prover toda a comunicação entre Agentes dentro e fora da plataforma;

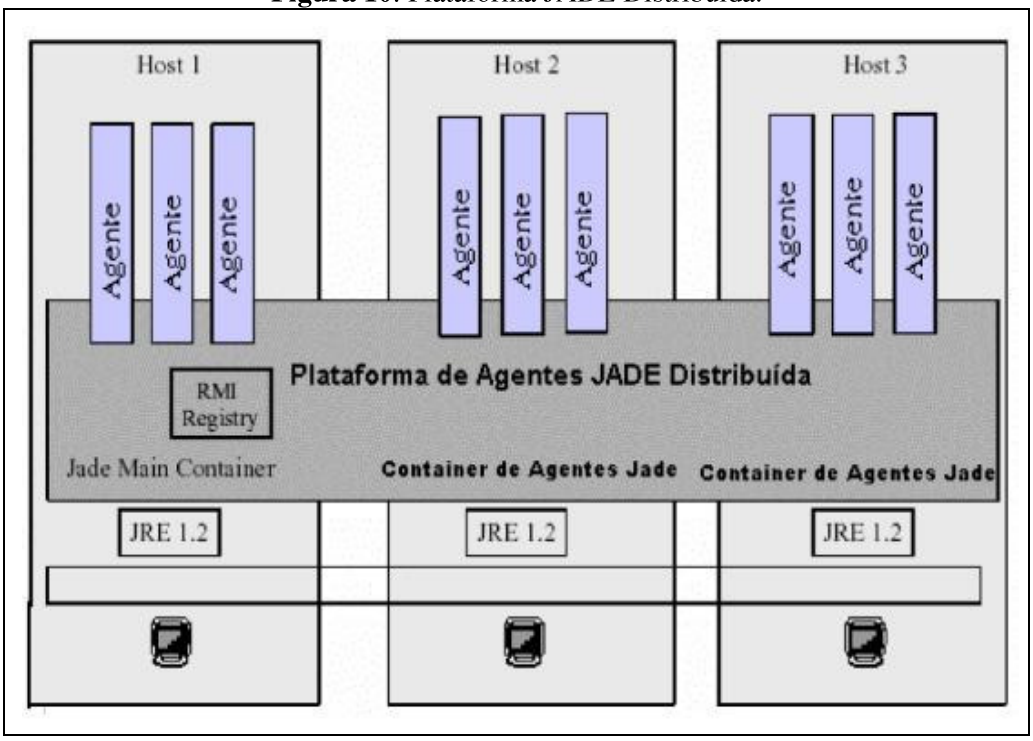
Segundo Bellifemini et al. (2007), o desenvolvimento de JADE se baseou nos seguintes princípios:

- Interoperabilidade – JADE é aderente ao padrão FIPA, desta forma os Agentes em JADE podem operar com quaisquer outros Agentes, desde que os mesmos também apresentem conformidade ao mesmo padrão;
- Uniformidade e portabilidade – JADE apresenta um conjunto homogêneo de APIs que são independentes da camada de rede e também do ambiente de execução Java.
- Facilidade de uso – A complexidade do *middleware* é ocultada através do uso de um conjunto de APIs simples e intuitivo;

Com relação à Filosofia, os programadores não precisam fazer uso obrigatório de todos os recursos providos pelo *middleware*. Recursos não utilizados não precisam ser considerados pelo programador e não implicam em sobrecarga computacional.

Uma das características da Plataforma JADE, conforme ilustrado pela Figura 10, é que ela pode ser distribuída por vários *hosts* (Host 1, Host 2 e Host 3) e, ainda assim, apresentar-se como uma única entidade para outro sistema aderente a FIPA.

Figura 10: Plataforma JADE Distribuída.



Fonte: Adaptada de (BELLIFEMINI et al., 2007).

Conforme a Figura 10, várias máquinas virtuais Java são agregadas numa mesma plataforma, e em cada uma delas um *Agent Container* (AC) será executado e este poderá, por sua vez, executar zero ou mais Agentes. A plataforma será então formada por um conjunto contendo um ou mais AC.

Para que um Agente possa localizar outros Agentes que oferecem um determinado tipo de serviço, obtendo assim seus identificadores para que possam se comunicar, a plataforma JADE implementa serviços de paginas amarelas em um Agente. Dessa forma os Agentes que desejam divulgar seus serviços, primeiramente, registram-se no DF, e os demais podem então buscar por um serviço específico.

Com base nas características apresentadas, foram desenvolvidos dois Agentes Reativos Simples (ARTERO, 2009), que são agentes que selecionam as ações a serem executadas com base exclusivamente na percepção atual do ambiente. Como não possuem uma memória interna, esses Agentes são incapazes de planejar ações futuras. Contudo, esse tipo de Agente pode ser convertido em Agentes de aprendizado que conseguem se adaptar as mudanças ocorridas no ambiente conforme elas vão acontecendo (SILVA, 2012).

3.5 CONSIDERAÇÕES FINAIS

Neste Capítulo, de forma geral, foram apresentados os principais conceitos utilizados como base para o desenvolvimento deste trabalho, sendo destacadas as principais características do NS2 que o torna uma das ferramentas de simulação mais utilizadas no meio acadêmico. Também foram abordadas as principais características do AVA Moodle que o torna uma boa escolha para ser utilizado para integração com o NS2.

Da mesma forma que o NS2 e o Moodle, neste capítulo foram abordados as principais características da linguagem HTML5, utilizada para desenvolver a ferramenta desta pesquisa; e uma contextualização sobre a definição de Agentes e Sistemas Multiagentes, mostrando algumas das características do uso da plataforma JADE para desenvolvimento de Sistemas Multiagentes.

Dando continuidade ao trabalho, o capítulo a seguir apresentará a arquitetura da ferramenta desenvolvida, a sua instalação no Moodle e a comunicação com NS2 por meio dos agentes desenvolvidos.

4 ARQUITETURA E IMPLEMENTAÇÃO DA PROPOSTA

Conforme apresentado na seção 3.3 do Capítulo 3, o Moodle oferece um espaço que reúne as principais informações e eventos relativos a uma disciplina em um determinado curso. Um dos benefícios que pode ser destacado é a possibilidade da criação de *plugins* na forma de módulos ou blocos para dar suporte a alunos e professores no desenvolvimento de suas atividades.

Com base nisso, este trabalho apresenta o NSMoo (*Network Simulator for Moodle*) uma ferramenta encapsulada em um bloco para integrar o NS2 ao Moodle. Assim, neste Capítulo será apresentada a sua arquitetura, detalhando as principais funcionalidades, bem como a forma utilizada na comunicação com o NS2.

Para isso, o Capítulo está organizado da seguinte forma: a seção 4.1 apresenta a arquitetura do NSMoo; a seção 4.2 apresenta algumas configurações possíveis de simular com o NSMoo; e por fim na seção 4.3 as considerações finais do Capítulo.

4.1 ARQUITETURA

Desenvolvido utilizando a linguagem HTML5, o NSMoo é uma ferramenta que permite a utilização do NS2 integrado ao Moodle, proporcionando a criação e configuração graficamente de cenários de redes. Como as ferramentas Moodle e NS2 não necessitam estar instaladas na mesma máquina, ou seja, podem estar distribuídas, tornou-se necessário desenvolver uma forma para realizar a comunicação entre o NSMoo integrado ao Moodle com o NS2.

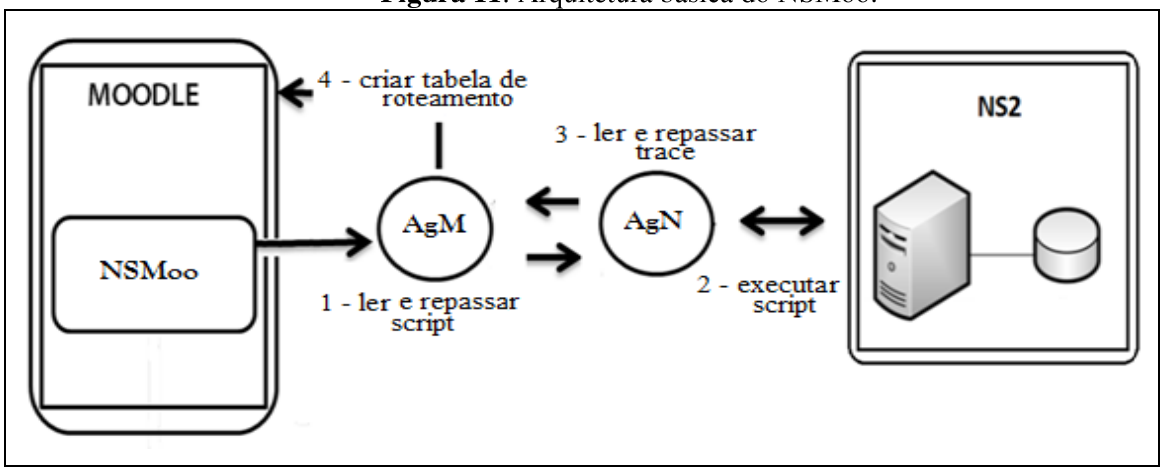
Assim, com base no que foi exposto sobre Agentes e Sistemas Multiagentes discutidos no Capítulo 2, optou-se por implementar dois Agentes Reativos Simples (BELLIFEMINE et al., 2007) para realizar a comunicação entre as ferramentas, como pode ser observado na Figura 11.

Os principais componentes visualizados na Figura 11 são:

- Moodle – AVA escolhido para integrar o NS2;

- NSMoo – ferramenta desenvolvida (encapsulada em um bloco) para o Moodle, responsável pela modelagem e configuração de cenários de redes, geração de *scripts* na linguagem OTCL e animação da simulação;
- AgM (Agente Moodle) – responsável por ler e repassar *scripts* na linguagem OTCL gerado pela ferramenta NSMoo, bem como montar a tabela de rotas para a animação da simulação;
- AgN (Agente NS) – responsável por executar os *scripts* no NS2;
- NS2 – ferramenta de simulação;

Figura 11: Arquitetura básica do NSMoo.



Como mencionado, para possibilitar a comunicação com o NS2 são utilizados dois agentes, um executando no servidor que contém o Moodle (AgM) e outro com o NS2 (AgN). Quando um cenário é configurado, é criado um arquivo temporário contendo o *script* em OTCL no diretório onde o AgM está executando.

Assim, a cada 5 segundos o AgM realiza a leitura desse arquivo a fim de verificar se foram realizadas alterações em sua estrutura. Posteriormente, caso alguma modificação seja percebida, o AgM, conforme descrito no passo 1 da Figura 11, inicia a comunicação com o agente reativo simples AgN.

Após o primeiro passo, o AgM repassa o *script* em OTCL para que o AgN o execute com o NS2 (passo 2). Ao executar o *script*, é gerado um arquivo de traço (*trace*) que contém todas as informações referentes a simulação do cenário proposto. Em seguida, o AgN lê esse arquivo e o repassa ao AgM conforme o passo 3.

Por fim, de posse do arquivo de traço, o AgM monta uma tabela de roteamento que contém todos os passos da simulação para que a animação da mesma seja executada dentro do Moodle (passo 4).

4.1.1 Implementação da comunicação entre os agentes

Para possibilitar a comunicação distribuída entre os agentes AgN e AgM como descrito na Figura 11, foi necessário a utilização de uma abordagem denominada Páginas Amarelas (BELLIFEMINE et al., 2007). A utilização de páginas amarelas especifica que cada agente ao se registrar no DF, deve também registrar o tipo de serviço que este oferece.

Com base nesta abordagem, o AgN ao ser iniciado, informa ao DF que provê serviços que necessitam do uso do NS2. Dessa forma qualquer agente que busque algo que esteja relacionado ao NS2, receberá o endereço do AgN. O Algoritmo 06 exibe o trecho de código responsável pelo registro do serviço no DF.

Na linha 2 do Algoritmo 06 é instanciada uma variável que receberá os serviços oferecidos pelo AgN. Em seguida, são associados a um nome especifico para ser disponibilizado no DF, conforme linha 3. Este nome é posteriormente adicionado ao endereço local do AgN (linha 4) e por fim é efetivamente registrado no DF (linha 5). Após o serviço ser registrado, o AgN estará pronto para atender as solicitações de outros agentes.

Algoritmo 06: AgN registrando serviço no DF.

```
1   ...
2       ServiceDescription servico = new ServiceDescription();
3       servico.setType("agente-ns");
4       servico.setName(this.getLocalName());
5       registraServico(servico);
6   try{...}
```

O processo de busca no DF é semelhante ao de registrar serviços, pois é criada uma descrição para que este seja retornado, porém sem a necessidade de informar a *Agent Identification* (AID) do agente. No Algoritmo 07 é possível observar como o AgM realiza a busca por um determinado serviço.

Na linha 2 do Algoritmo 07 é instanciada uma variável (*dfd*) responsável por retornar as descrições dos serviços oferecidos por um determinado agente registrado no DF. Posteriormente, conforme linha 3, é repassada a descrição do serviço ao DF (linha 4) e, em seguida, é instanciado um vetor (*resultado*) que receberá os endereços dos agentes que oferecem o serviço especificado na descrição repassada (linha 5).

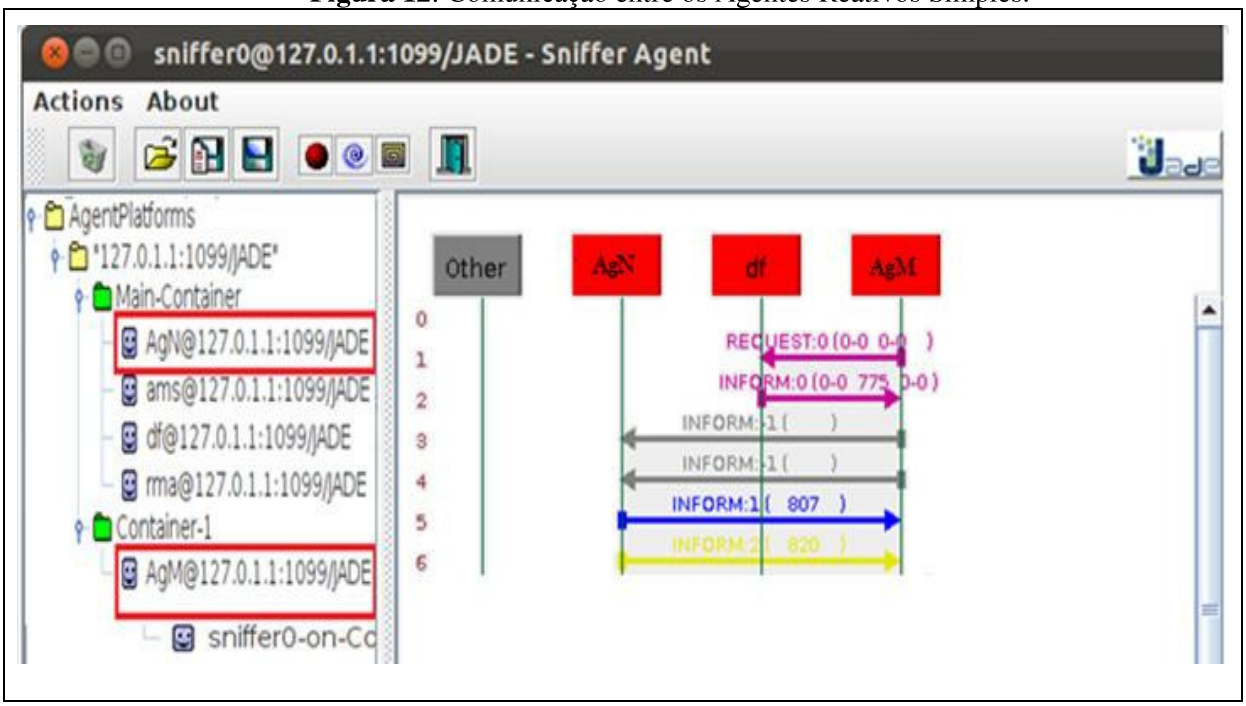
Se o vetor de registros não estiver vazio (linha 6), ou seja, no mínimo um endereço seja retornado, o AgM adiciona este endereço no cabeçalho da mensagem (linha 7) e a envia contendo o *script* em OTCL para o AgN, conforme linha 8 do Algoritmo 07.

Algoritmo 07: AgM buscando serviço.

```
1    protected void busca(final ServiceDescription sd, final ACLMessage msg) {
2    DFAgentDescription dfd = new DFAgentDescription();
3    dfd.addServices("agente-ns");
4    try {
5    DFAgentDescription[] resultado = DFService.search(myAgent, dfd);
6        if (resultado.length != 0) {
7            msg.addReceiver(resultado[0].getName());
8            myAgent.send(msg);
9    …
```

O estabelecimento da comunicação dos agentes pode ser observado de forma mais concreta na Figura 12. Utilizando o Agente *Snnifer* (BELLIFEMINE et al., 2007) da plataforma JADE, é possível capturar a troca de mensagens.

Figura 12: Comunicação entre os Agentes Reativos Simples.



Na Figura 12 é possível observar que o agente AgM realiza uma busca no DF (REQUEST:0(0-0 0-0)) a fim de encontrar um agente que realize a simulação com o NS2. O DF por sua vez responde ao AgM com a localização (INFORM:0(0-0 775 0-0)) do AgN, e a partir daí, os dois agentes estabelecem a comunicação entre si.

4.1.2 Funcionalidades do NSMoo

O NSMoo possibilita a utilização de equipamentos de redes, possuindo algumas funcionalidades para criação de cenários. Para instalar a ferramenta no Moodle o primeiro passo é fazer *download* do mesmo e posteriormente adicioná-lo a disciplina de redes de computadores (todo o processo de instalação será descrito no Apêndice A).

Após a ferramenta instalada e os agentes ativados, o NSMoo estará pronto para ser utilizado como ilustra a Figura 13. É possível observar que este possui alguns componentes e funcionalidades que são comuns no desenvolvimento de práticas de redes como: *host*, *switch*, *hub*, roteadores, nós móveis, criação de tráfego, inserção de falhas de enlace, geração de *scripts*, dentre outras.

Figura 13: NSMoo inserido no Moodle.

Para melhorar a visualização das características do NSMoo, a Tabela 02 faz uma descrição de suas funcionalidades e a Tabela 03 descreve seus principais componentes.

Tabela 02: Descrição das funcionalidades do NSMoo.

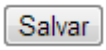
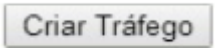
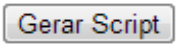
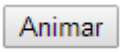
FUNCIONALIDADE	DESCRIÇÃO
	Criar um novo projeto
	Salvar o *status* atual do projeto
	Carregar um arquivo salvo
	Abrir arquivo carregado
	Criar tráfego entre os *hosts* conectados
	Adicionar uma falha de enlace
	Gerar *script* OTCL
	Testar cenários pré-configurados
	Iniciar Animação
	Parar Animação
	Informações sobre aplicações e pacotes
	Área para visualizar o *script* em OTCL gerado
	Obter ajuda

Tabela 03: Descrição dos componentes do NSMoo.

COMPONENTE	DESCRIÇÃO
	Inserir um *host*
	Inserir um nó móvel
	Inserir um *hub*

	Inserir um roteador
	Inserir um roteador sem fio
	Inserir um *switch* 24 portas
	Conectar componentes
	Desconectar componentes
	Excluir um equipamento

Assim, por meio dos componentes e funcionalidades descritos, o NSMoo permite por meio da simulação dos principais equipamentos de redes a criação de cenários, destacando-se as seguintes funcionalidades:

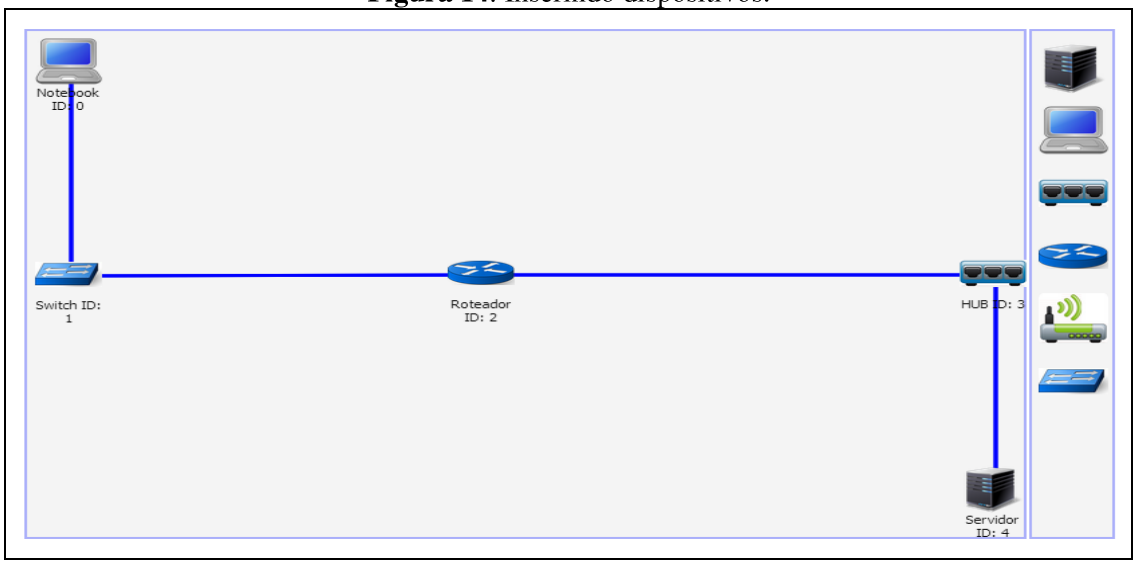
- Criação de topologias com inserção de nós e enlaces;
- Configuração de tráfego com os protocolos TCP e UDP;
- Configuração de aplicações CBR, FTP, PARETO e EXPONENTIAL;
- Geração automática de *script* na linguagem OTCL e simulação de falha de enlace;
- Simulação de redes LANs, MANs e WANs; e
- Animação da simulação sem utilizar módulos externos do NS2;

Na subseção a seguir será descrito com mais detalhes as funcionalidades destacadas acima.

4.1.3 Configurando tráfego e gerando script

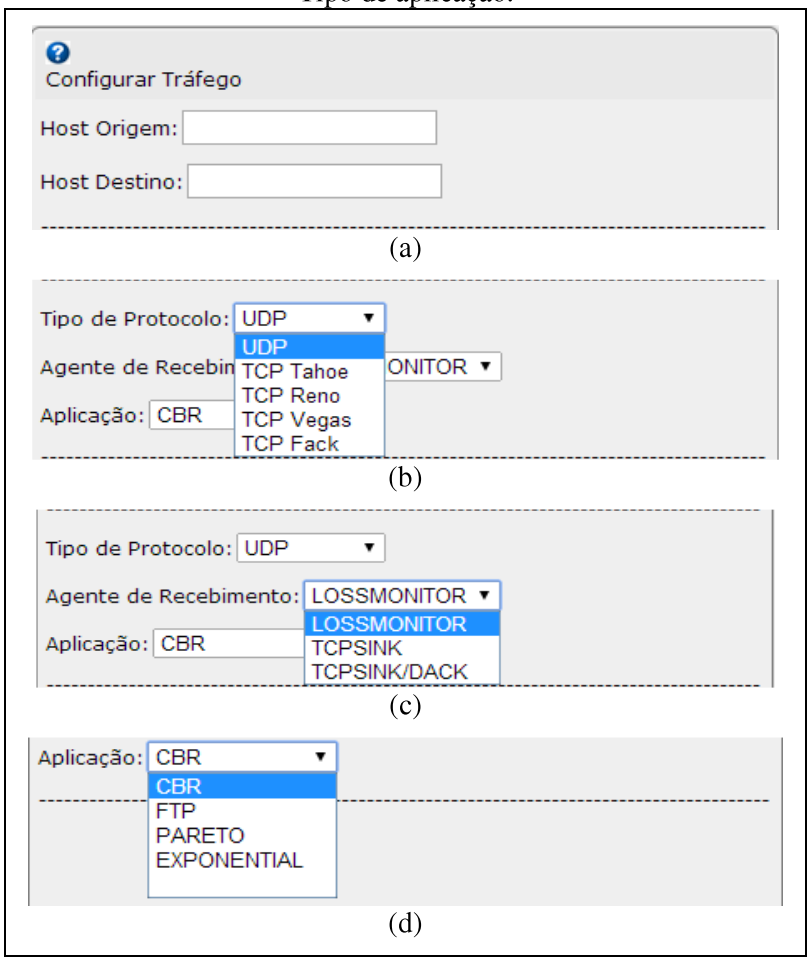
Para utilizar o NSMoo, primeiramente o usuário cria um cenário como ilustrado na Figura 14. Para adicionar um componente, é necessário clicar e segurar com o botão esquerdo do *mouse* em cima de um dos componentes e arrastá-lo para a área de criação. Feito isso, ao apontar o *mouse* em qualquer um dos componentes inseridos, este apresentará 3 opções: conectar, desconectar e excluir (descritos na Tabela 03), para que dessa forma as conexões entre os equipamentos possa ser trabalhada.

Figura 14: Inserindo dispositivos.



Após a criação do cenário, é necessário escolher a opção “Criar Tráfego” onde será informando o identificador do *host* de origem (Host Origem) e do *host* de destino (Host Destino), conforme Figura 15(a).

Figura 15: (a) Configuração inicial do tráfego. (b) Tipo de protocolo. (c) Agente de recebimento. (d) Tipo de aplicação.



O NSMoo permite a utilização de protocolos do tipo UDP e TCP, sendo este último nas variações: TCP *Tahoe*, TCP *Reno* e TCP *Vegas*. Para esta configuração basta escolher em "Tipo de Protocolo" o melhor a ser utilizado na simulação, conforme Figura 15(b).

Para cada protocolo, é necessário o uso de um "Agente de Recebimento" que será responsável por receber os pacotes enviados por um determinado *host*. Para o protocolo UDP, é utilizado o agente de recebimento do tipo LOSSMONITOR, e para as variações do TCP podem ser utilizados os tipos TCPSINK ou TCPSINK/DACK, Figura 15(c).

O NSMoo também possibilita à utilização de diversas classes que simulam protocolos da camada de aplicação no NS2. É possível observar no campo "Aplicação" na Figura 15(d) as principais aplicações que podem ser simuladas, classificadas em:

- CBR (*Constant Bit Rate*) – termo utilizado em telecomunicações referentes à qualidade de serviço é utilizado em aplicações de áudio e vídeo ou jogos interativos;
- FTP (*File Transfer Protocol*) – serviço aplicado à transferência confiável de arquivos, uma vez que requer garantias de entrega dos pacotes;
- PARETO – serviço que gera tráfego enviando pacotes a uma taxa fixa em vários períodos distintos;
- EXPONENTIAL – serviço que gera tráfego a uma taxa constante em períodos pré-determinados denominados em "*on*" e "*off*";

Como o NS2 simula várias classes de aplicações, com o NSMoo, para cada aplicação é utilizado um pacote contendo uma coloração diferenciada conforme a Tabela 04. Assim, é possível fazer distinção das várias aplicações que podem ser executadas.

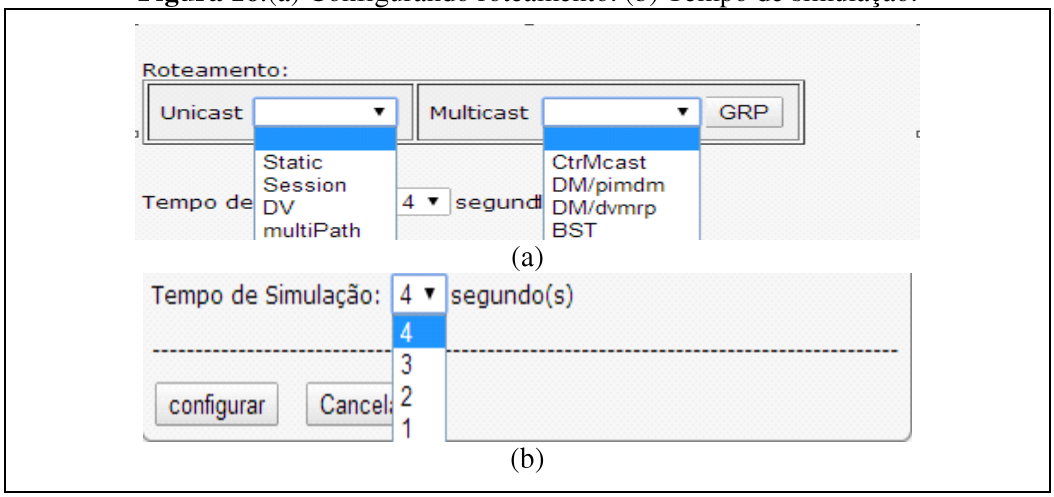
Tabela 04: Tipos de pacotes para cada aplicação.

PACOTE	SIGNIFICADO
	Pacote marrom: utilizado em aplicação CBR
	Pacote verde: utilizado em aplicação FTP
	Pacote roxo: utilizado em aplicação EXPONENTIAL
	Pacote amarelo: utilizado em aplicação PARETO

	Pacote azul: ACK
	Pacotes Perdidos

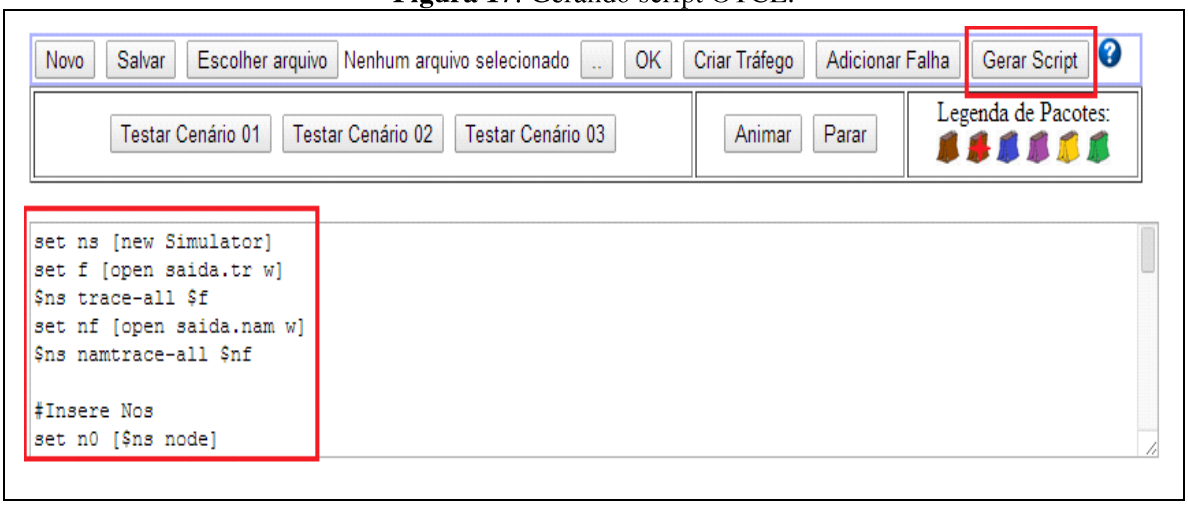
No NSMoo também é possível configurar algoritmos de roteamento de uma simulação. Esta informação deve ser adicionada no campo “Roteamento”, podendo ser do tipo *unicast* ou *multicast*, conforme Figura 16(a). Por fim, Figura 16(b) “Tempo de Simulação”, deve ser informado o tempo total que o NS2 executará a simulação, podendo variar entre 1 a 4 segundos.

Figura 16:(a) Configurando roteamento. (b) Tempo de simulação.



Após a configuração do tráfego, o usuário escolhe a opção “Gerar Script” (Figura 17), para que seja gerado o *script* na linguagem OTCL e repassado ao AgN para ser executado no NS2.

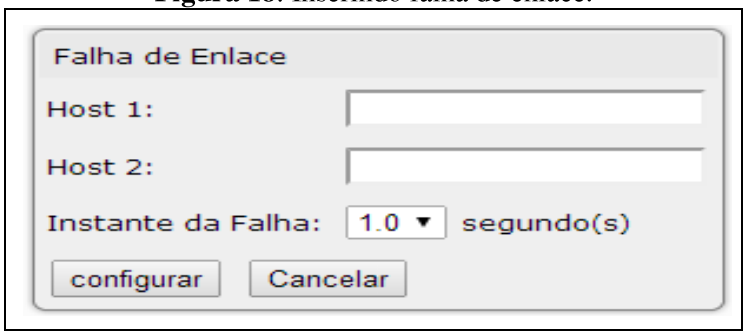
Figura 17: Gerando script OTCL.



O NSMoo possibilita ainda a inserção de falhas no enlace para tornar a simulação mais dinâmica. Para inserir uma falha de enlace, primeiramente é necessário pressionar o botão “Adicionar Falha”, em seguida aparecerá uma janela, como ilustrado na Figura 18, para que seja configurado o enlace entre os *hosts* (Host 1 e Host 2) e também o tempo em que a falha deverá ocorrer (Instante da Falha).

Assim, se estiver configurado um tempo de execução de 3 segundos para a simulação, e um tempo de 2 segundos para a falha, quando o NS2 completar 2 segundo de execução ele saberá que deverá simular uma falha de enlace no instante indicado.

Figura 18: Inserindo falha de enlace.



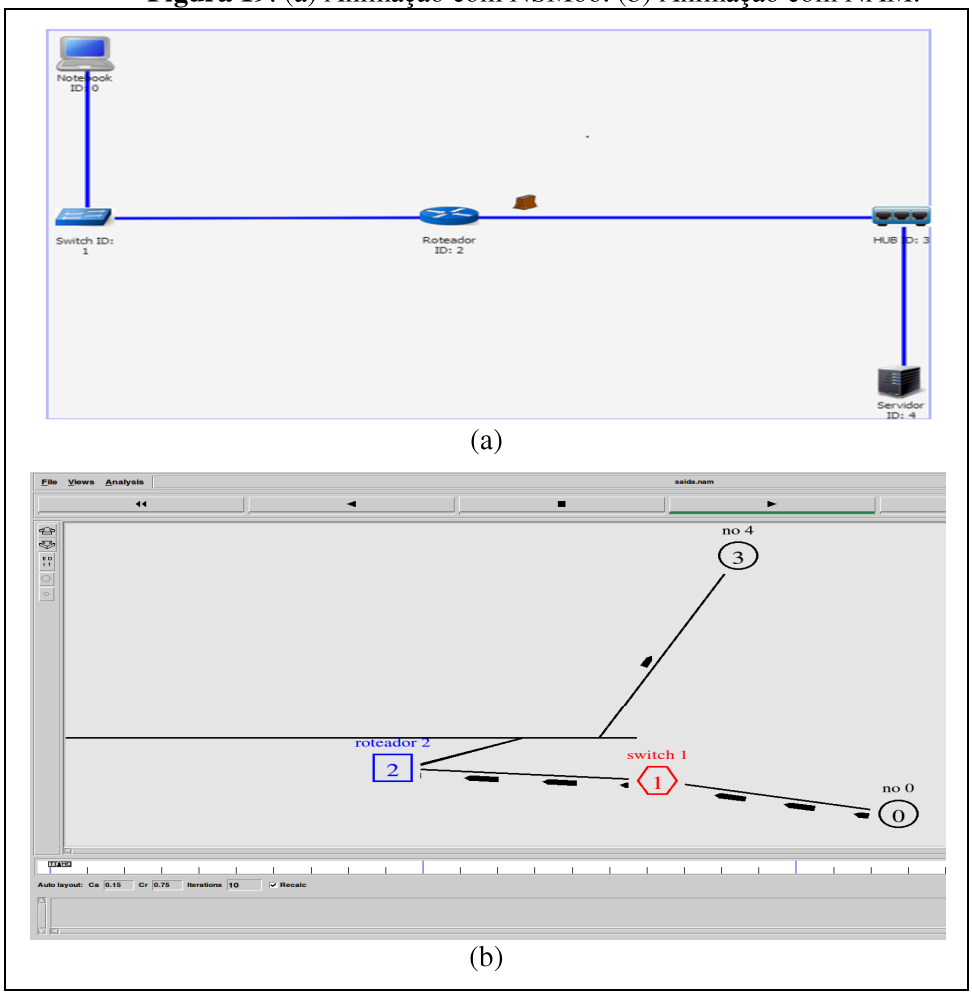
4.1.4 Animação

A necessidade de criação de um módulo próprio de animação para o NSMoo é explicada pela falta de interação existente no módulo externo utilizado pelo NS2, o NAM. Na Figura 19, é ilustrada uma comparação da animação entre as duas ferramentas.

A Figura 19(a) é representada pela animação com o NSMoo, enquanto que na Figura 19(b) ilustra a simulação com o NAM. É possível observar que na animação utilizando o NSMoo existe um maior nível de interação quando comparado ao NAM.

Enquanto que o NAM utiliza figuras geométricas para representar a animação dos componentes do cenário criado (na Figura 19(b) os círculos representam nós, o quadrado representa um roteador, a linha horizontal representa um hub e um hexágono representando o *switch*), o NSMoo utiliza as mesmas imagens que foram usadas para criar o cenário inicialmente, proporcionando melhor visualização da comunicação entre os equipamentos simulados.

Figura 19: (a) Animação com NSMoo. (b) Animação com NAM.



A próxima seção apresenta a interação com o NSMoo por meio da criação, configuração e simulação de cenários de redes.

4.2 INTERAÇÃO COM O NSMoo

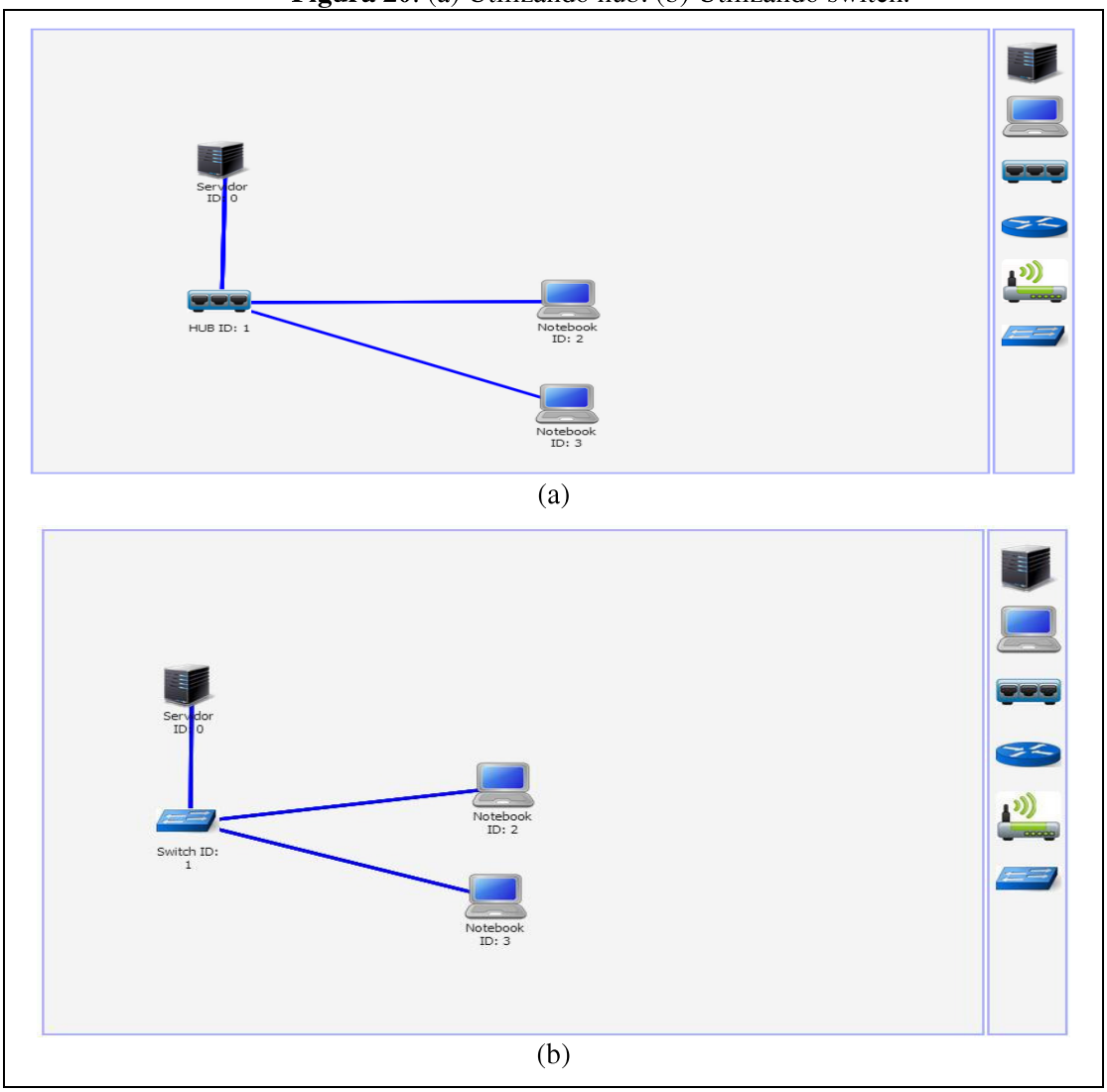
Na seção anterior foi possível ter uma visão geral da ferramenta NSMoo, suas principais funcionalidade, seus componentes e as configurações iniciais necessárias na criação e modelagem de cenários de redes.

Esta seção é responsável por apresentar a ferramenta em ação mostrando alguns dos possíveis cenários que podem ser configurados e animados utilizando o NS2 como ferramenta de simulação.

4.2.1 Cenário 1 – simulação com hub e switch

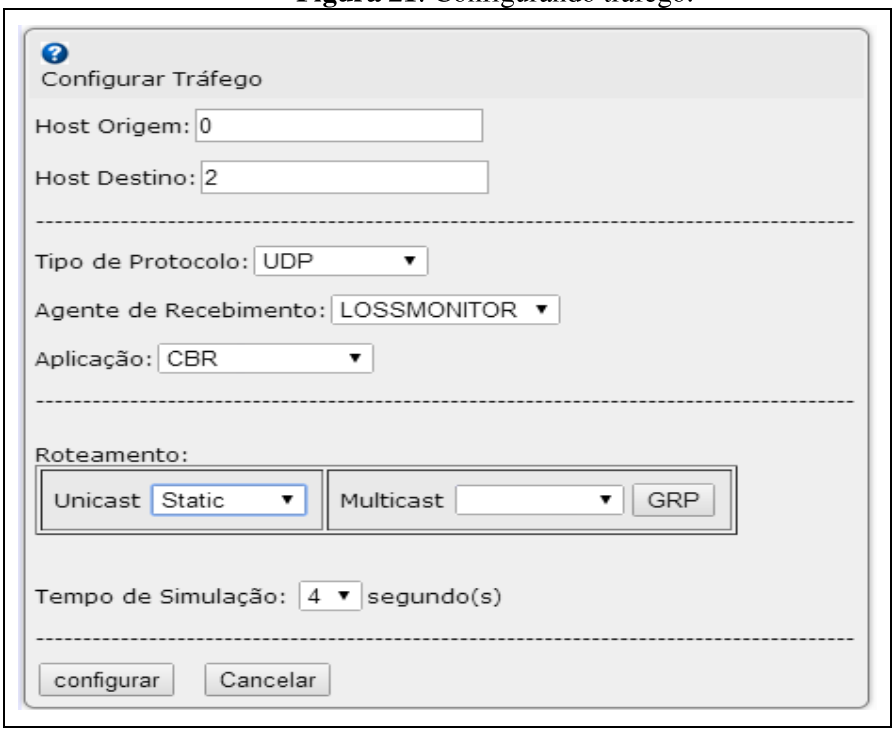
Nos cenários ilustrados na Figura 20, é possível observar dois dispositivos bastantes utilizados em redes de computadores, o *hub* (Figura 20(a)) e *switch* (Figura 20(b)). O objetivo principal desta subseção é mostrar as principais diferenças existentes entre estes equipamentos utilizando o NSMoo.

Figura 20: (a) Utilizando hub. (b) Utilizando switch.



Após adicionar os componentes na área de criação, o próximo passo é inserir um link de comunicação entre o *host* de origem e de destino, utilizando um protocolo (para o exemplo é utilizado o UDP) com um agente de recebimento (LOSSMONITOR), simulando uma determinada aplicação (CBR), conforme ilustrado na Figura 21.

Figura 21: Configurando tráfego.



O próximo passo é gerar o *script* na linguagem OTCL, e para isso o usuário deve escolher a opção "Gerar Script". Os Algoritmos 08 e 09, respectivamente, exemplificam a estrutura de configuração do tráfego na linguagem OTCL para o *hub* e para o *switch*.

Algoritmo 08: Script de criação do tráfego para o hub.

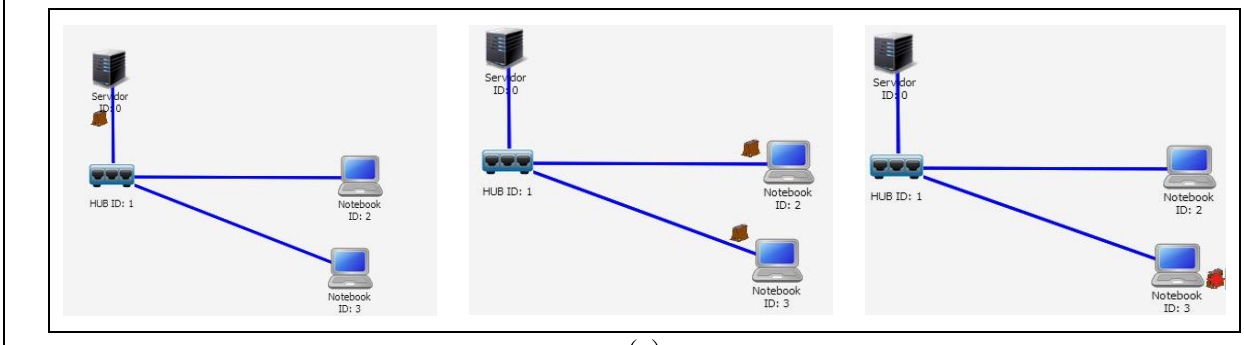
```
...
#Insere HUB
lappend nodelisth0 $n0
lappend nodelisth0 $n2
lappend nodelisth0 $n3
set h0 [$ns newLan $nodelisth0 10Mb 5ms LL DropTail Mac Channel]

#Agente de Transmissao
set udp4 [new Agent/UDP]
$udp4 set fid_ 4
$ns attach-agent $n0 $udp4

#Aplicacao
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4

#Agente de Recepcao
set lossmonitor4 [new Agent/LossMonitor]
$ns attach-agent $n2 $lossmonitor4
$ns connect $udp4 $lossmonitor4

#Escalonamento de transmissao
$ns at 0.0 "$cbr4 start"
```

Algoritmo 09: Script de criação do tráfego para switch.

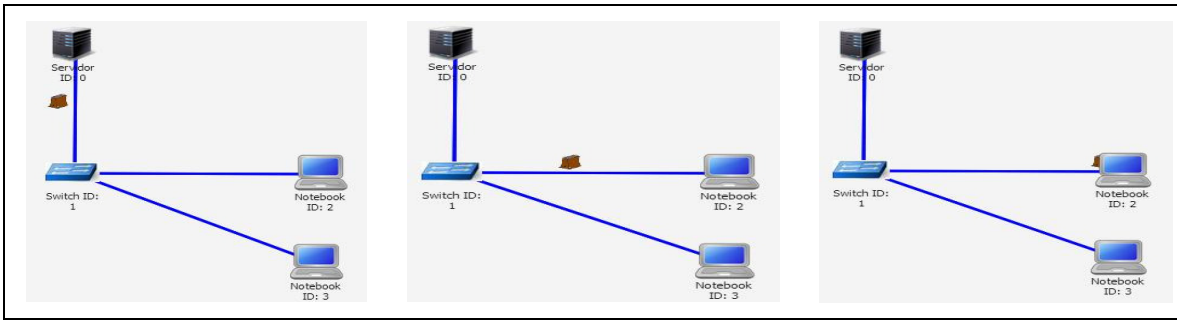
```
set s1 [$ns node]
$s1 color red
$ns at 0.0 "$s1 label \"Switch 1\""
$s1 shape box

#Insere Links
$ns duplex-link $n0 $s1 1.0Mb 10ms DropTail
$ns duplex-link $n2 $s1 1.0Mb 10ms DropTail
$ns duplex-link $n3 $s1 1.0Mb 10ms DropTail
#Agente de Transmissao
set udp4 [new Agent/UDP]
$udp4 set fid_ 4
$ns attach-agent $n0 $udp4
#Aplicacao
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
#Agente de Recepcao
set lossmonitor4 [new Agent/LossMonitor]
$ns attach-agent $n2 $lossmonitor4
$ns connect $udp4 $lossmonitor4
```

Após o *script* ser executado no NS2, a animação estará disponível. Para executar a animação o usuário deve pressionar o botão “Animar”. Na Figura 22 é possível observar a diferença existente entre a simulação entre *hub* (Figura 22(a)) e *switch* (Figura 22(b)).

Figura 22: (a) Animação da simulação com hub. (b) Animação da simulação com switch.



(a)



(b)

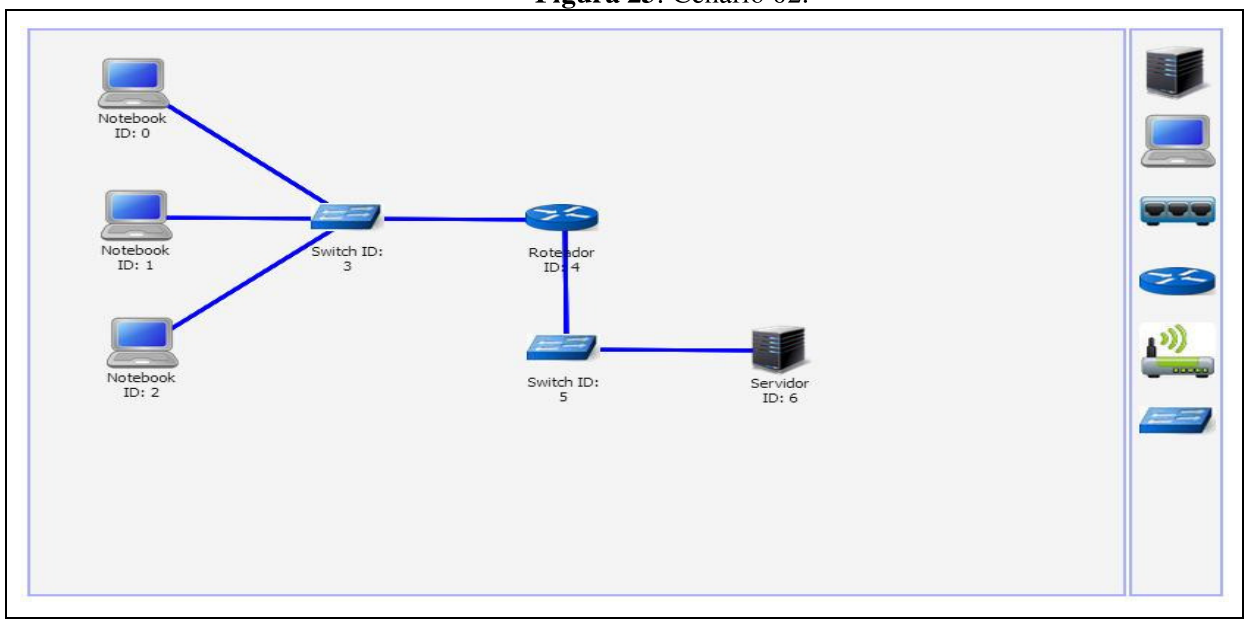
Os *switches* são dispositivos bastante semelhantes aos *hubs*, tendo como principal diferença a forma como transmitem dados entre os *hosts*. Enquanto *hubs* reúnem o tráfego em somente uma via, um *switch* cria uma série de canais exclusivos em que os dados do *host* de origem são recebidos somente pela máquina destino.

Os pacotes que são duplicados pelo *hub* são descartados pelas máquinas que não são destino desses pacotes.

4.2.2 Cenário 2 – aplicação FTP

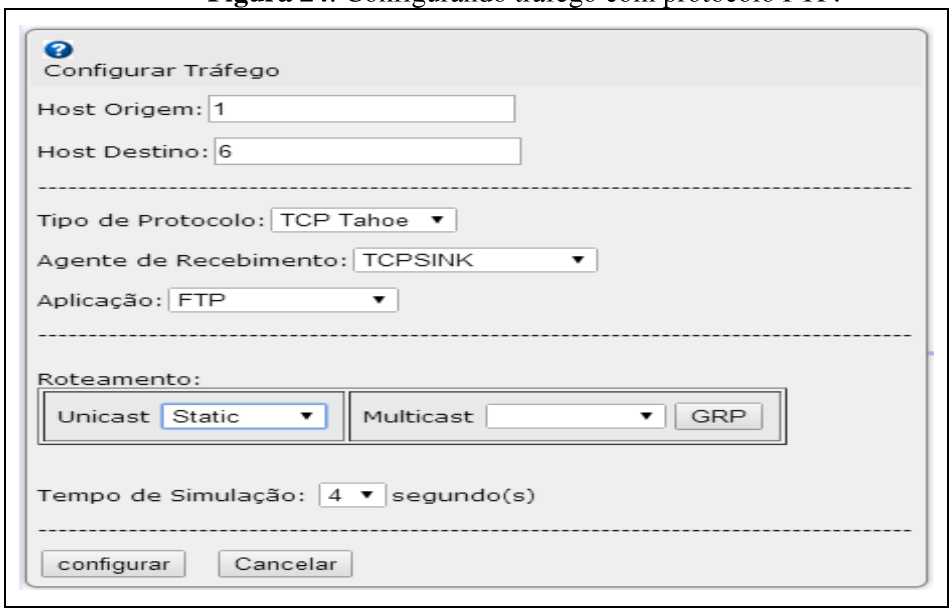
Como mencionado no Capítulo 3, o NS2 possibilita a configuração de diferentes tipos de aplicações. Nesta seção serão apresentados os passos para configurar uma aplicação FTP utilizando o NSMoo. A Figura 23 ilustra a configuração inicial para a simulação.

Figura 23: Cenário 02.



Após a inserção dos dispositivos, o próximo passo é criar o link de comunicação como ilustra a Figura 24. Na imagem é inserido um link entre os nós “ID: 1” e “ID: 6” utilizando o protocolo TCP *Tahoe*, com agente de recebimento TCPSINK.

Figura 24: Configurando tráfego com protocolo FTP.



O próximo passo após a criação do link de comunicação é gerar o *script* em OTCL para ser executado com o NS2. O Algoritmo 10 exemplifica a estrutura do *script* para o tráfego configurado na Figura 24.

Algoritmo 10: Script do tráfego para aplicação FTP.

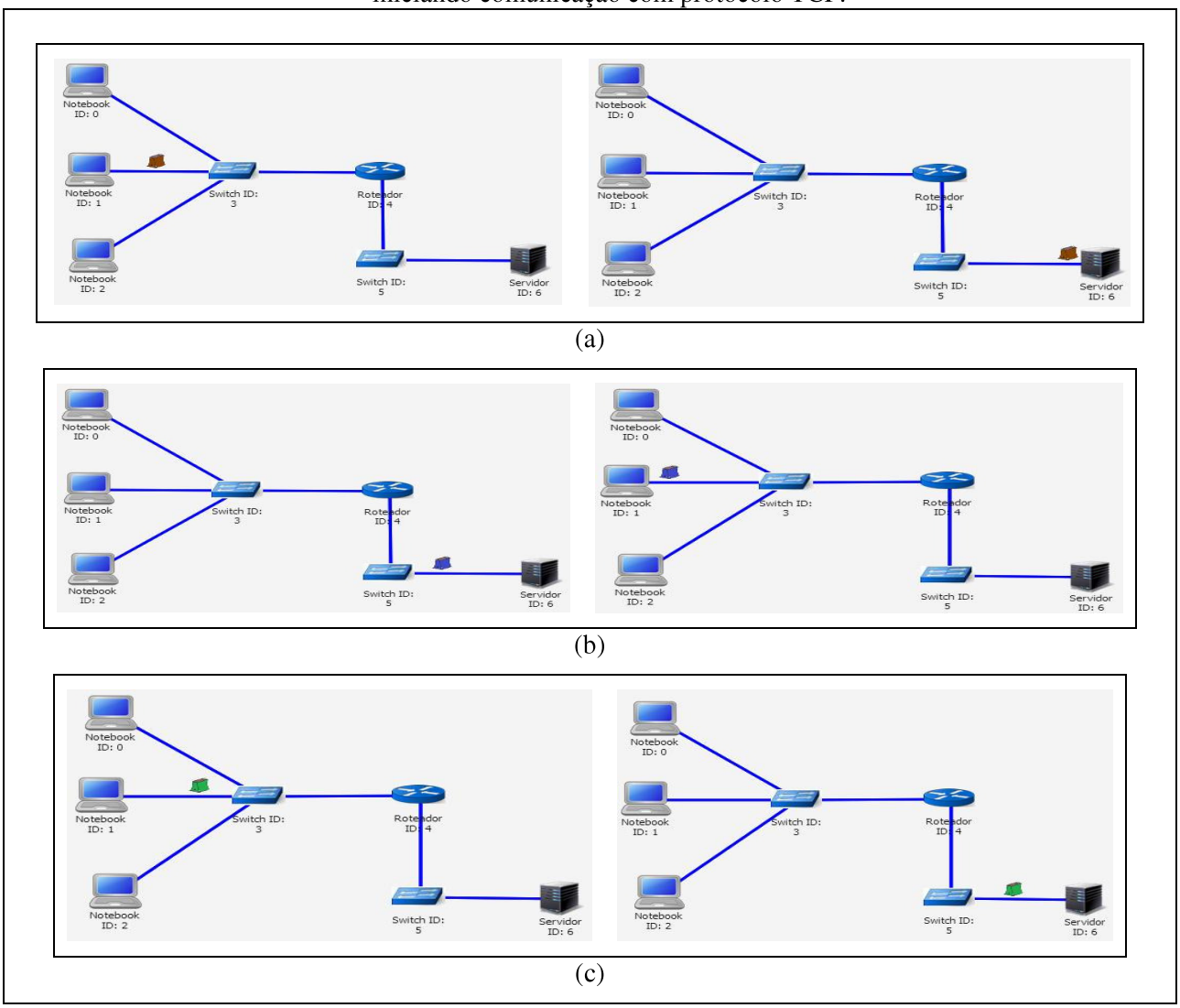
```
#Insere Links
$ns duplex-link $r4 $s3 1.0Mb 10ms DropTail
$ns duplex-link $r4 $s5 1.0Mb 10ms DropTail
$ns duplex-link $n0 $s3 1.0Mb 10ms DropTail
$ns duplex-link $n1 $s3 1.0Mb 10ms DropTail
$ns duplex-link $n2 $s3 1.0Mb 10ms DropTail
$ns duplex-link $n6 $s5 1.0Mb 10ms DropTail

#Agente de Transmissao
set tcp7 [new Agent/TCP]
$tcp7 set fid_ 7
$ns attach-agent $n1 $tcp7
#Aplicacao
set ftp7 [new Application/FTP]
$ftp7 attach-agent $tcp7
#Agente de Recepcao
set sink7 [new Agent/TCPSink]
$ns attach-agent $n4 $sink7
$ns connect $tcp7 $sink7
$ns at 0.0 "$ftp7 start"
```

Uma vez que o *script* seja executado com o NS2, a animação estará pronta para ser iniciada. Na Figura 25 é possível observar a animação da simulação. O FTP é um protocolo utilizado para transferência de dados da camada de aplicação, por meio de uma interface que o usuário realiza autenticação e envio/recebimento de arquivos para um servidor.

Como o protocolo HTTP, o FTP utiliza conexão TCP para se conectar com o servidor, porém com uma diferença importante, o FTP utiliza duas conexões TCPs paralelas para transmitir um determinado arquivo. Ao começar uma sessão FTP, o cliente primeiramente inicia uma conexão de controle com o servidor (Figura 25(a)) requisitando informações de usuário, senha e permissão para realizar alterações de diretório.

Com a verificação positiva (Figura 25(b)) o servidor mantém a conexão de controle aberta e aguarda as solicitações do cliente. Quando o cliente solicita um arquivo o servidor abre uma conexão de dados utilizando o protocolo TCP (Figura 25(c)).

Figura 25: (a) Conexão de controle. (b) Confirmação de estabelecimento de conexão. (c) iniciando comunicação com protocolo TCP.



4.2.3 Cenário 03 – aplicação EXPONENTIAL

Em aplicações *EXPONENTIAL* o tráfego é determinado por uma distribuição exponencial onde pacotes possuem tamanho constante, assumindo dois estados, *on* e *off*. Os pacotes nesse tipo de aplicação são enviados a uma taxa de tempo fixa quando assume o estado *on*.

No NSMoo a aplicação *EXPONENTIAL* pode ser configurada utilizando protocolo de transporte UDP ou TCP. A Figura 26 ilustra o cenário inicial para esse tipo de aplicação. Após a criação do cenário o passo seguinte, como descrito nas subseções anteriores, é o estabelecimento do link de comunicação. A Figura 27 ilustra a configuração do tráfego entre os nós "ID: 5" e "ID: 1".

Figura 26: Cenário 03.

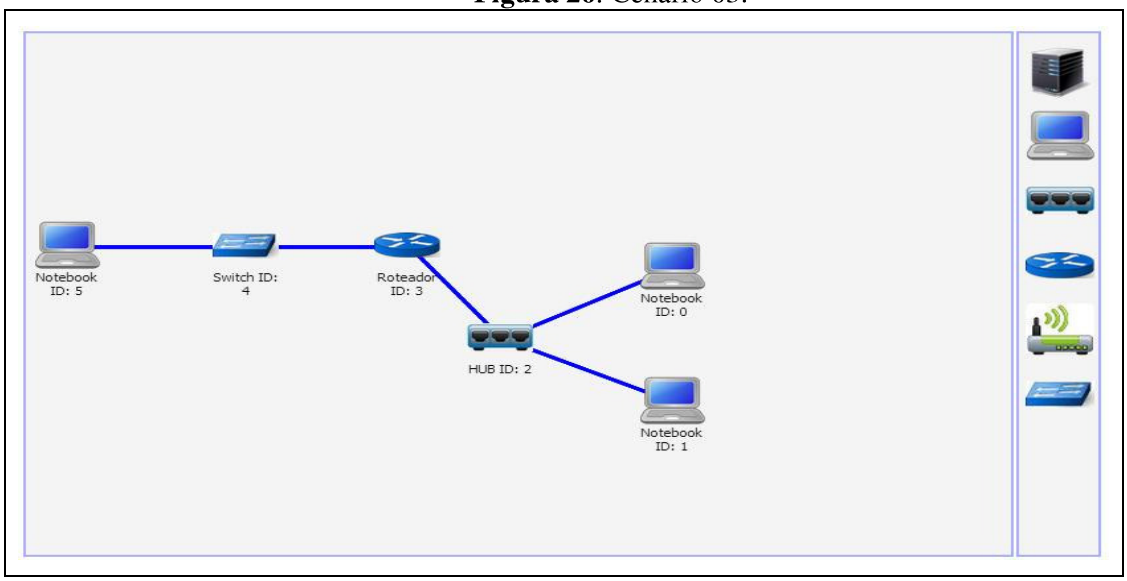
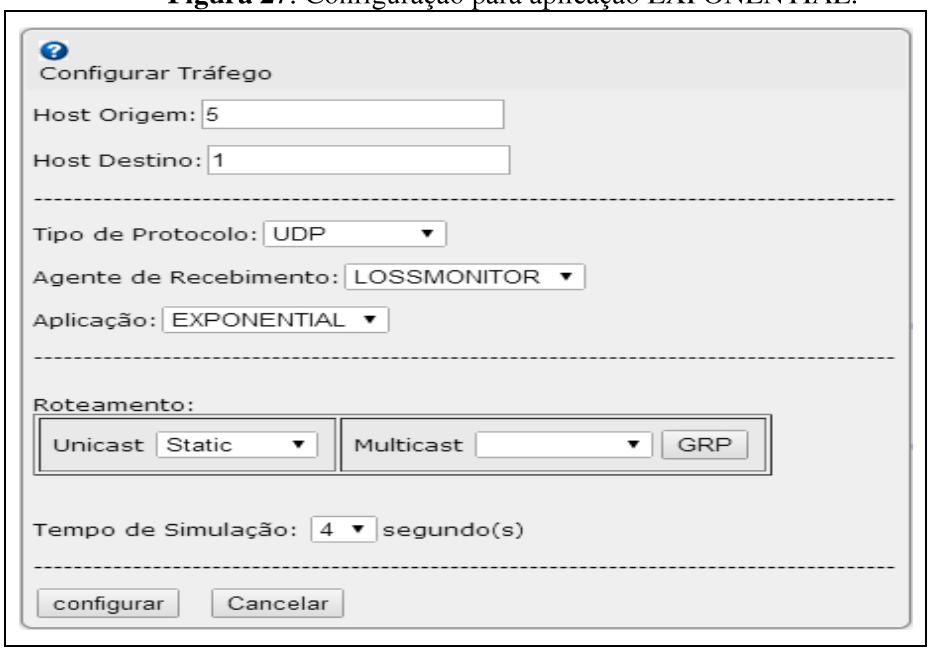


Figura 27: Configuração para aplicação EXPONENTIAL.

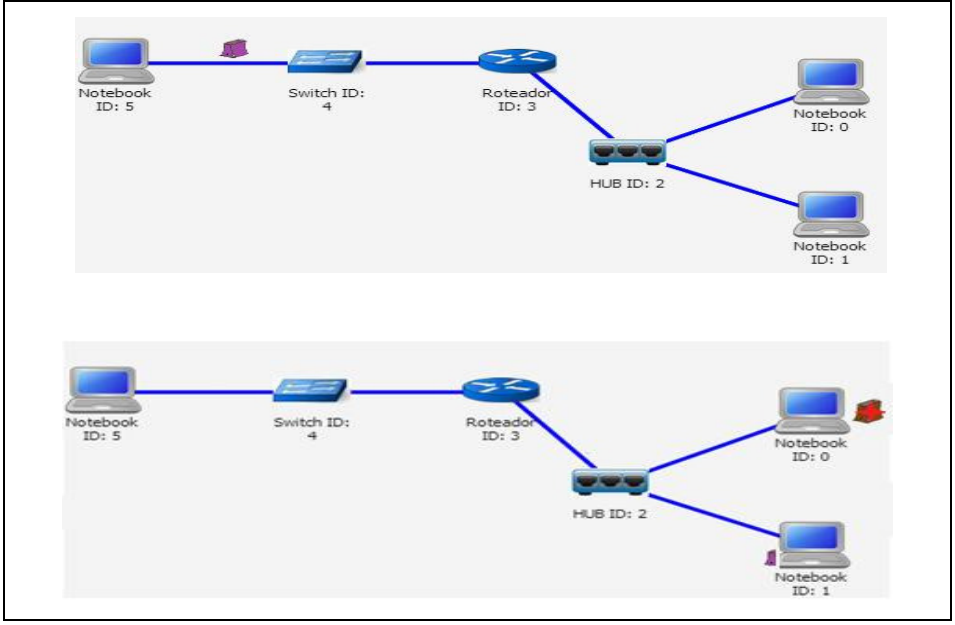


Após o estabelecimento do link, ao clicar na opção “Gerar Script” será apresentado o *script* em OTCL conforme Algoritmo 11. A animação para este tipo de aplicação pode ser observada na Figura 28.

Algoritmo 11: Script do tráfego para aplicação EXPONENTIAL.

```
...
lappend nodelisth0 $n0
lappend nodelisth0 $n1
lappend nodelisth0 $r3
set h0 [$ns newLan $nodelisth0 10Mb 5ms LL DropTail Mac Channel]
#Insere Links
$ns duplex-link $r3 $s4 1.0Mb 10ms DropTail
$ns duplex-link $n5 $s4 1.0Mb 10ms DropTail

#Transmissao
#Agente de Transmissao
set udp6 [new Agent/UDP]
$udp6 set fid_ 6
$ns attach-agent $n5 $udp6

#Aplicacao
set exponential6 [new Application/Traffic/Exponential]
$exponential6 attach-agent $udp6
#Agente de Recepcao
set lossmonitor6 [new Agent/LossMonitor]
$ns attach-agent $n1 $lossmonitor6
$ns connect $udp6 $lossmonitor6

#Escalonamento de transmissao
$ns at 0.0 "$exponential6 start"
...
```

Figura 28: Animação da aplicação EXPONENTIAL.



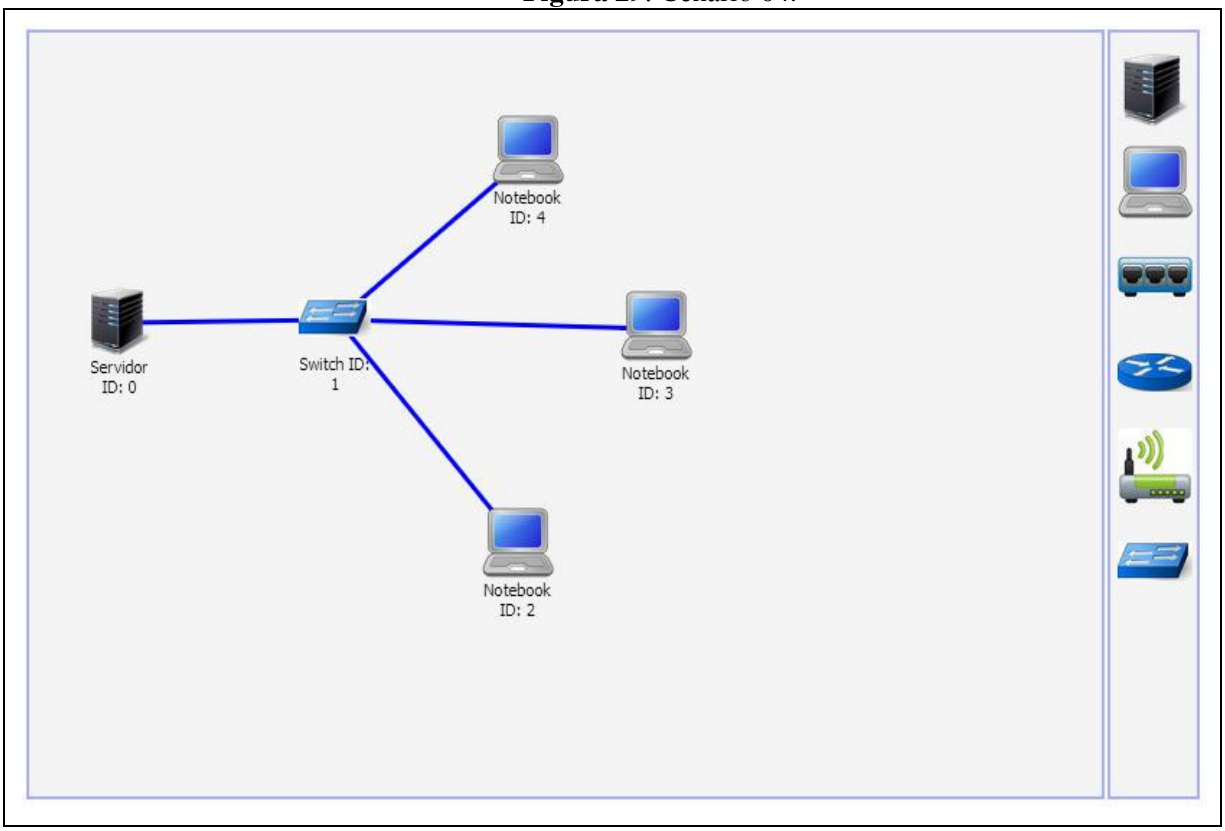
4.2.4 Cenário 04 – PROTOCOLO TCP

Como discutido no Capítulo 02, o NS2 permite a utilização do protocolo de transporte TCP em suas variações. Dessa forma por meio do NSMoo é possível configurar cenários e verificar as principais diferenças existentes no uso desse protocolo. Assim, nesta subseção será apresentada uma das variações do TCP, o Tahoe.

As primeiras implementações do TCP não incluíam qualquer mecanismo de controle de congestionamento, assim o TCP Tahoe foi a primeira proposta para tratar desse problema. Desde então, tem surgido muitas modificações no TCP para aumentar seu desempenho.

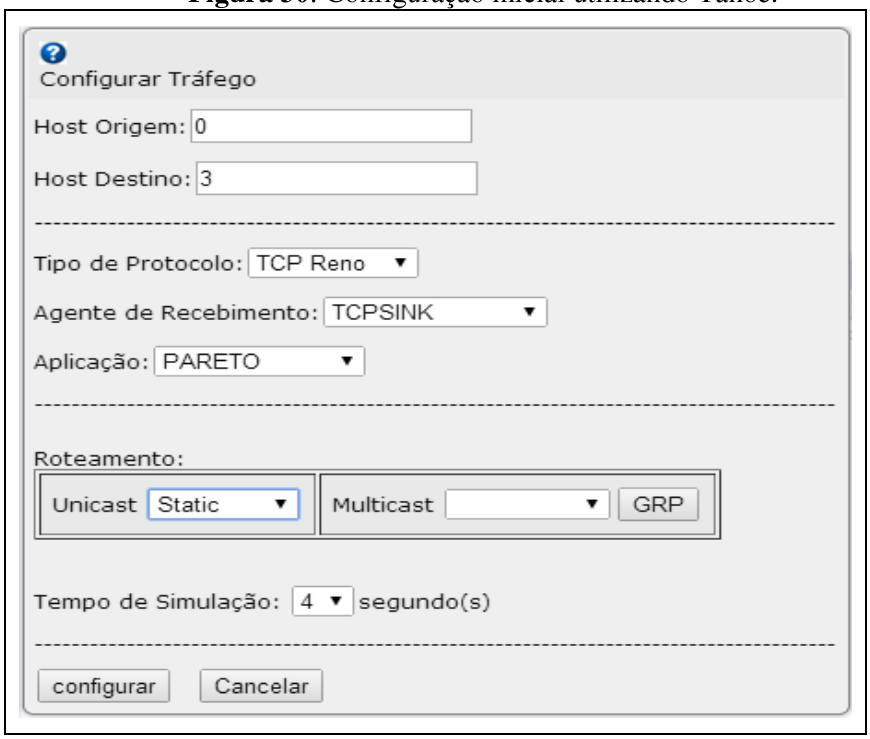
O TCP Tahoe utiliza o principio de *Slow Start* (partida lenta), onde a janela de congestionamento começa um segmento e vai aumentando exponencialmente até que ocorra uma perda. Ao ser detectada a perda do pacote, a janela de congestionamento volta a ser de um segmento e começa a crescer novamente. Dessa forma o Tahoe previne que seja enviado para a rede congestionada mais pacotes do que ela possa processar. Na Figura 29 e 30 são ilustradas as configurações iniciais para a exemplificação do uso do Tahoe.

Figura 29: Cenário 04.



Para este exemplo é utilizado uma aplicação PARETO para enviar pacotes do *host* “ID: 0” ao “ID:3”. Após a configuração do tráfego, o Algoritmo 12 resume os comandos em OTCL para criação do link de comunicação, e na Figura 31 a animação da simulação.

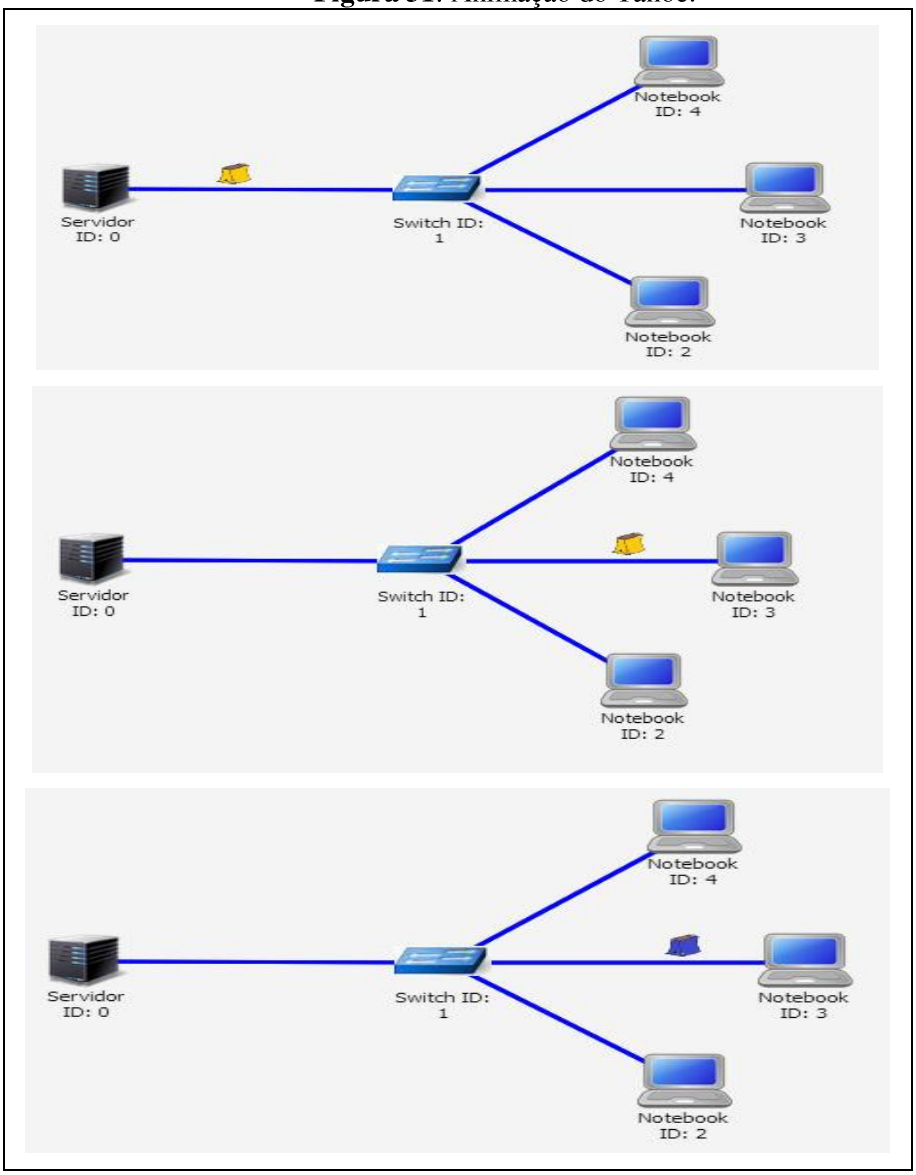
Figura 30: Configuração inicial utilizando Tahoe.



Algoritmo 12: Script do tráfego para Tahoe.

```
#Insere Links
$ns duplex-link $n0 $s1 1.0Mb 10ms DropTail
$ns duplex-link $n2 $s1 1.0Mb 10ms DropTail
$ns duplex-link $n3 $s1 1.0Mb 10ms DropTail
$ns duplex-link $n4 $s1 1.0Mb 10ms DropTail
#Agente de Transmissao
set tcp5 [new Agent/TCP]
$tcp5 set fid_ 5
$ns attach-agent $n0 $tcp5

#Aplicacao
set pareto5 [new Application/Traffic/Pareto]
$pareto5 attach-agent $tcp5

#Agente de Recepcao
set sink5 [new Agent/TCPSink]
$ns attach-agent $n3 $sink5
$ns connect $tcp5 $sink5

#Escalonamento de transmissao
$ns at 0.0 "$pareto5 start"
```

Figura 31: Animação do Tahoe.



4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este Capítulo apresentou a ferramenta NSMoo, uma interface desenvolvida para possibilitar a utilização do NS2 com o Moodle. O Capítulo também apresentou a arquitetura interna da ferramenta, seus principais componentes, funcionalidades e as características que tornam possível a integração.

O Capítulo também abordou o funcionamento do NSMoo como ferramenta para modelagem e configuração de cenários de redes utilizando o simulador NS2. Apresentou, por meio de suas funcionalidades, a facilidade que a ferramenta possibilita na visualização das

diferenças existentes entre os diversos tipos de equipamentos e aplicações que podem ser simuladas.

No próximo Capítulo, serão apresentadas as Considerações Finais e as perspectivas de Trabalhos Futuros.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

De acordo com a pesquisa desenvolvida neste trabalho, a utilização de simuladores para o ensino contribui bastante com o desenvolvimento das atividades laboratoriais (FERREIRA et al., 2013). Quando inserido em um AVA possibilita a realização de práticas mesmo quando os usuários não estão dentro das instituições de ensino, contribuindo assim tanto com o ensino presencial como ao ensino a distância (FERREIRA et al., 2013).

De forma geral, a pesquisa abordou os principais conceitos utilizados para o desenvolvimento da pesquisa, sendo destacadas as principais características do NS2, do AVA Moodle, da linguagem HTML5, utilizada para desenvolver a ferramenta desta pesquisa, bem como uma contextualização sobre a definição de Agentes e Sistemas Multiagentes, mostrando algumas das características do uso da plataforma JADE para desenvolvimento de agentes.

Foi apresentada a ferramenta NSMoo, sendo destacada sua arquitetura interna, seus componentes e funcionalidades, sendo possível observar a configuração da simulação dos principais tipos de cenários de redes e a verificação da comunicação entre os agentes AgM e AgN, bem como a importância da utilização de um módulo próprio de animação da simulação.

Assim, com o NSMoo é possível: verificar o funcionamento dos principais algoritmos de roteamento; definir diferentes topologias; configurar os equipamentos envolvidos na comunicação entre redes diferentes; diagnosticar e definir soluções para problemas comuns em ambientes reais.

Além disso, o NSMoo proporciona um ambiente atrativo e com baixo custo na implantação proporcionado pela escolha das ferramentas utilizadas (Moodle e NS2), facilitando o aprendizado e a utilização do NS2 por meio da geração de *scripts* na linguagem OTCL.

A ferramenta desenvolvida ainda possui como vantagens em relação aos trabalhos relacionados: confeccionar um maior número de cenários, não se restringindo apenas a *hosts* e servidores; modelar e configurar links de comunicação utilizando protocolos da camada de transporte; utilizar diferentes tipos de aplicações nas simulações; verificar o comportamento dos pacotes de redes; visualizar a animação da simulação.

Assim, o NSMoo é uma ferramenta que pode ser utilizada para dar apoio ao processo de aprendizagem em disciplinas de redes de computadores tanto na modalidade presencial quanto na modalidade a distância.

5.1 LIMITAÇÕES DA FERRAMENTA

O desenvolvimento da pesquisa encontrou algumas dificuldades. Apesar de ser possível no NSMoo a configuração e simulação de algoritmos de roteamento, o mesmo não consegue realizar a animação de todos estes algoritmos devido a um problema na leitura do arquivo *trace* gerado pelo NS2.

Dependendo do tempo configurado para o NS2 rodar uma simulação, o arquivo *trace* pode passar de 20 mil linhas em uma simulação de 8 segundos. Por esse motivo, se torna difícil exibir a animação da simulação.

Outra limitação a ser destacada (também por questões do tamanho do arquivo *trace*) é a animação de protocolos *wireless*. Apesar de ser possível a geração de *scripts* para esses protocolos, o NSMoo não suporta a animação de cenários sem fio.

5.2 TRABALHOS FUTUROS

O NSMoo ainda não foi liberado oficialmente para utilização prática. Por isso, para uma real avaliação do seu desempenho junto a disciplina de redes de computadores, é necessário realizar uma Avaliação prática da ferramenta.

Como trabalhos futuros esperam-se:

- Realizar testes no NSMoo com grupos de alunos na disciplina de redes de computadores;
- Realizar os ajustes necessários para suportar a animação dos diferentes tipos de algoritmos de roteamento e protocolos *wireless*;
- Utilizar mais recursos do simulador NS2 possibilitando a criação de cenários de redes de sensores sem fio.

5.3 PUBLICAÇÕES

Esta dissertação proporcionou algumas publicações, tais como:

1. FERREIRA, K. H. A.; LIMA, R. W.; CHAVES, J. O. M.; LIMA, M. V. A. **Inserindo um Laboratório Virtual para o Ensino de Redes de Computadores**. In: International Conference on Interactive Computer aided Blended Learning 2013 – ICBL 2013, Florianopolis – SC, 2013. Anais do International Conference on Interactive Computer aided Blended Learning 2013. ISBN: 978-3-86219-598-5.
2. FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M. **Laboratório Virtual para o Ensino de Redes de Computadores no Moodle**. In: XXIV Simpósio Brasileiro de Informática na Educação – SBIE 2013, Campinas – SP, 2013. Anais do XXIV Simpósio Brasileiro de Informática na Educação. DOI: 10.5753/CBIE.SBIE.2013.950.
3. FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M. **Utilizando um Laboratório Virtual para o Ensino de Redes de Computadores no Moodle**. In: VI Escola Potiguar de Computação e suas Aplicações – EPOCA 2013, Mossoró – RN, 2013. Anais da VI Escola Potiguar de Computação e suas Aplicações.
4. FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M.; ROCHA, A. **LVRC – Laboratório Virtual web para o ensino de Redes de Computadores no Moodle**. In: XVIII Conferência Internacional sobre Informática na Educação – TISE 2013, Porto Alegre – RS, 2013. Anais da XVIII Conferência Internacional sobre Informática na Educação. ISBN: 978-956-19-0836-9.
5. FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M.; ROCHA. **LARC – Laboratório Virtual para o auxílio de práticas na disciplina de Redes de Computadores integrado ao Moodle**. In: International Journal of Recent Contributions from Engineering, Science & IT – i-JES 2014, vol. 2 p. 17-21 eISSN: 2197-8581.
6. FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M.; ROCHA. **Integrando o Network Simulator 2.0 a um Ambiente Virtual de Aprendizagem**. In: XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014, Brasília – DF. Anais do XXXIV Congresso da Sociedade Brasileira de Computação.

REFERÊNCIAS

AMADEUS. SISTEMA LMS AMADEUS. 2014. **Portal do Software Público Brasileiro**. Disponível em <http://www.softwarepublico.gov.br/>. Acesso em: Jul. 2014.

AMARAL, E. M. H.; AVILA, B.; ZEDNIK, H.; TAROUCO, L. **Laboratório Virtual de Aprendizagem: Uma Proposta Taxonômica**; Revista Novas Tecnologias na Educação - RENOTE, Porto Alegre, v. 9 nº. 2, dezembro, 2011.

ARIEIRA, J. O.; ARIEIRA, C. R. D.; FUSCO, J. P. A.; SACOMANO, J. B.; BETTEGA, M. O. P. **Avaliação do aprendizado via educação a distância**: **a visão dos discentes**. *Ensaio: aval.pol.públ.Educ.* [online]. 2009, vol.17, n.63, pp. 313-340. ISSN 0104-4036.

ARTERO, A. O. **Inteligência Artificial – Teoria e Prática**. 1ª. ed. São Paulo: Livraria da Física, 2009.

BELLIFEMINE, F.; CAIRE, G., GREENWOOD, D. **Multi-Agent Systems with JADE**. Liverpool, Inglaterra: John Wiley & Sons, Ltd, 2007.

BELZERANA, P.; GONZALEZ-BARBONE, V. **Incorporacion de un Simulador Gráfico de Redes en un Objeto de Aprendizaje Reutilizable**. Disponível em <http://e-spacio.uned.es:8080/fedora/get/taee:congreso-2006-1124/SD104.pdf >. Acesso em: jul. 2012.

CALLAGHAN, M. J.; HARKIN, J.; MCGINNITY, T. M.; MAGUIRE, L. P. **Intelligent User Supportin Autonomous Remote Experimentation Environments**. 2008. Trabalho apresentado em IEE Transation Industrial Eletronics, 2008.

CAMPOS, C. P.; FERREIRA, C. E. **BOCA: Um sistema de apoio para competições de programação**. In: Workshop sobre educação em computação – WEI 2004, Salvador – BA.

CISCO. **Networking Academy**. Disponível em: <https://www.netacad.com/web/about-us/cisco-packet-tracer>. Acesso em: jul. 2014.

COUTINHO, M. M.; TELES, M. P. ; RIBEIRO JÚNIOR, M. A. S. **GENESI gerador gráfico de código para o Network Simulator**. In: Semana Paraense de Informática – Belém, 2005.

FERREIRA, K. H. A.; LIMA, R. W.; LIMA, M. V. A.; CHAVES, J. O. M. **Laboratório Virtual para o Ensino de Redes de Computadores no Moodle**. In: XXIV Simpósio Brasileiro de Informática na Educação – SBIE 2013, Campinas – SP, 2013. Anais do XXIV Simpósio Brasileiro de Informática na Educação. DOI: 10.5753/CBIE.SBIE.2013.950.

FIPA. Welcome to the Foundation for Intelligent Physical Agents. Site Oficial do Padrão FIPA, 2011. Disponível em: <http://www.fipa.org/>. Acesso em: out. 2013.

FRANCISCATO, F. T.; RIBEIRO, P. S.; MOZZAQUARTTO, P. M.; MEDINA, R. D. **Avaliação dos Ambientes Virtuais de Aprendizagem Moodle, TelEduc e Tidia - Ae: um estudo comparativo**. RENOTE: Revista Novas Tecnologias na Educação, v. 6 n. 2, 2008.

HASSAN, E. B. **Laboratório Virtual 3D para ensino de Redes de Computadores**; XIV Simpósio Brasileiro de Informática na Educação - NCE - IM/UFRJ 2003.

HENDERSON-SELLERS, B.; GIORGINI, P. **Agent-Oriented Methodologies**. Guernsey, Reino Unido: IGI Global, 2005.

ISSARIYAKUL, E. H.; HOSSAIN, E. **Introduction to Network Simulator NS2**. Springer, 2008.

KENSKI, V. M.; **Tecnologias do ensino presencial e à distância**. São Paulo, SP: Papirus, 2008.

KUMAR, S.; GANKOTIYA, A. K.; DUTTA, K. **A Comparative Study of Moodle with other e-Learning Systems**. In: International Conference on Electronics Computer Technology – ICECT, 3. Kanyakumari: IEEE, 2011.

LIMA, R. W. **Mapa de Conteúdos e Mapa de Dependências: ferramentas pedagógicas para uma metodologia de planejamento baseada em objetivos educacionais e sua implementação em um ambiente virtual de aprendizagem**. Tese (Doutorado), UFRN, 2009.

LIMA, R. W.; FIALHO, S. V. **Introducing assessment into the teaching-learning process of Distance Education using discipline planning**. In: 9th IFIP World Conference on Computers in Education, Bento Gonçalves – RS, 2009.

MARQUES, E. M. D; PLÁCIDO, R. A. S. A; SAMPAIO, P. N. M. **Visual Network Simulator (VNS): A GUI to QoS Simulation for the ns-2 Simulator**. In: Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on. ISBN: 978-1-4244-3806-8.

MOODLE. Modular Object-Oriented Dynamic Learning Environment. 2012. Disponível em: < https://moodle.org/>. Acesso em: mai. de 2013.

MORAZ, E. **Treinamento prático com ActionScritpt**. São Paulo, SP: Digerati Books, 2011.

NETSIMK**. The ultimate aid to teaching and learning Cisco™ Routers**. Disponível em: <http://netsimk.com/index.html>. Acesso em: jul. 2014.

OLIVEIRA, D. M.; CRUZ, R. S.; SALGUEIRO, R. J. P. B.; ROCHA, T. **An integrated development environment for the NS-2 Network Simulator**. Revista Scientita Plena vol. 8, num. 3, 2012.

PEREIRA, T. R. D. S.; CHAVES, D. A. R. **Moodle: Um Experimento On-Line para Potencializar um Ambiente de Apoio à Aprendizagem**. In: XVIII Simpósio Nacional de Geometria Descritiva e Desenho Técnico – GRAPHICA, Paraná – Curitiba, 2007.

PHP. Hypertext Preprocessor. 2012. Disponível em: <http://www.php.net>. Acesso em: set. 2013.

PINHEIRO, R. P.; LINS, F. A. A.; MELO, J. C. B. **A utilização de Simulação no Ensino de Redes de Computadores**. 2009. Disponível em: <http://www.eventosufrpe.com.br/epex2009/cd/resumos/R0311-1.pdf>. Acesso em: mai. 2013.

PONTES, A. A. A.; LIMA, R. W. **NsGraph: Interface Gráfica de Modelagem e Geração Automática de Scripts para o NS-2**. Anais do XXVI Congresso da SBC. Campo Grande: SBC, 2006.

RIVERBED. Disponível em: <http://www.riverbed.com/products/performance-management-control/>. Acesso em: jul. 2014.

SILVA, C. H. C.; SAMPAIO, R. F.; LEÃO, R. P. S.; BARROSO, G. C.; SOARES, J. M. **Desenvolvimento de um Laboratório Virtual para Capacitação Tecnológica à Distância em Proteção de Sistemas Elétricos**. XXXIX Congresso Brasileiro de Educação em Engenharia – COBENGE 2011, Blumenau – SC. Anais do XXXIX Congresso Brasileiro de Educação em Engenharia.

SILVA JR, J. M.; FIRMINO, E. C. M. **Desenvolvimento de Jogos em** HTML5. In: IX SBGames 2010, Florianópolis – SC. Anais do IX SBGames.

SILVA, M. S. **HTML5**; São Paulo: Novatec Editora, 2011.

SILVA, L. C. N. **MobiLE – Um Ambiente Multiagente de Aprendizagem Móvel para Apoiar a Remomendação Ubíqua de Objetos de Aprendizagem**. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Rio Grande do Norte, Universidade Federal Rural do Semi-Árido, Mossoró, 2012.

SIROTHEAU, S.; BRITO, S. R.; SILVA, A. S.; ELIASQUEVICI, M. K.; FAVERO, E. L; TAVARES, O. L. **Aprendizagem de iniciativas em algoritmos e programação: foco nas competências de autoavaliação**. In: Simpósio Brasileiro de Informática na Educação – SBIE 2011, Aracajú – SE.

TAVARES, D. A. B.; FRANÇA, A. B.; SOARES, J. M.; BARROSO, N. M. C.; MOTA, J. C. M. **Integração do ambiente WIMS ao Moodle usando Arquitetura Orientada a Serviços e Compilação Automática de Médias**. RENOTE: Revista Novas Tecnologias na Educação, v. 8 n. 3, 2010.

VOSS, G. B.; MEDINA, R. D.; AMARAL, E. M. H.; ARAÚJO, F. V.; NUNES, F. B.; OLIVEIRA, T. B. **Proposta de utilização de Laboratórios Virtuais para o ensino de redes de computadores: articulando ferramentas, conteúdos e possibilidades**. Revista Novas Tecnologias na Educação – RENOTE, v.10 nº 3, dezembro, 2012.

APÊNDICE A – INSTALAÇÃO DO NSMOO NO MOODLE

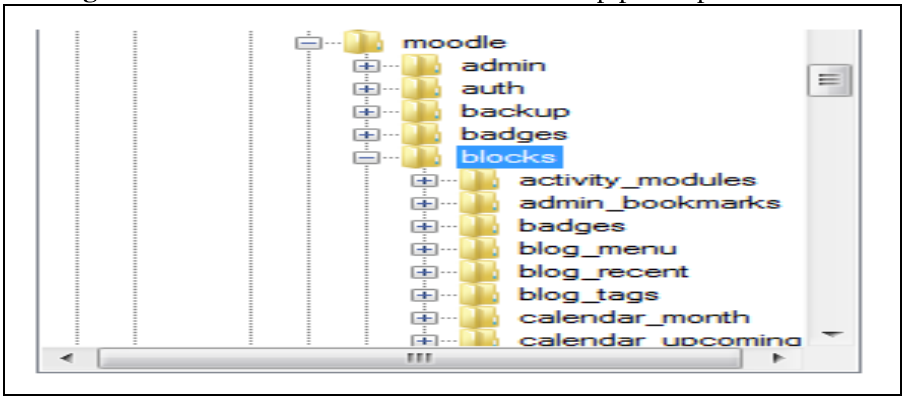
Como o NSMoo faz a utilização de Agentes Reativos desenvolvidos com a plataforma JADE, é necessário que algumas configurações sejam realizadas. A plataforma JADE foi desenvolvida em Java, e o requisito mínimo para que esta seja executada é o *run-time* do Java, na versão 1.4 ou superior. A documentação e todos os *softwares* JADE estão distribuídos sob as limitações da LGPL e disponíveis para *download* no sítio HTTP://jade.cselt.it.

Por se tratar de uma ferramenta de integração para o NS2, é necessário que este esteja instalado para realizar as simulações de redes. O NS2 e sua documentação podem ser encontrados no seguinte endereço eletrônico: http://nsnam.isi.edu/nsnam/index.php/Downloading_and_installing_ns-2.

INSTALAÇÃO DA FERRAMENTA NO MOODLE

Para instalar o NSMoo no Moodle, o primeiro passo é fazer *download* do arquivo block_nsmoo.zip no site https://dl.dropboxusercontent.com/u/67978034/meu_site/NSMOO/index.html#&panel1-1. Em seguida, é necessário fazer a extração do conteúdo do arquivo baixado para o diretório Moodle em "*moodle/blocks*", como ilustra a Figura 32.

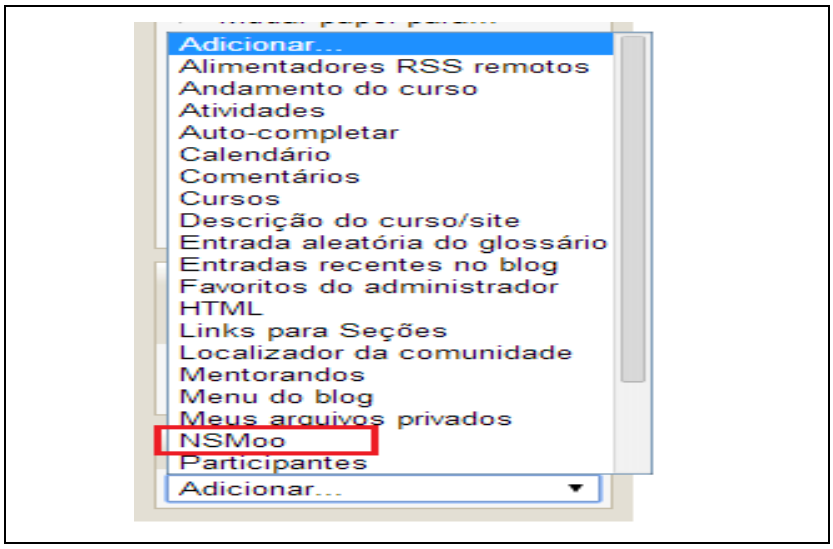
Figura 32: Extraindo o conteúdo do nsmoo.zip para a pasta blocks.



Após a extração do conteúdo para a pasta *blocks*, o usuário deve verificar as atualizações disponíveis no Moodle, e posteriormente clicar em "Ativar Edições" (Figura 33) e selecionar a opção "Adicionar..." Figura (34).

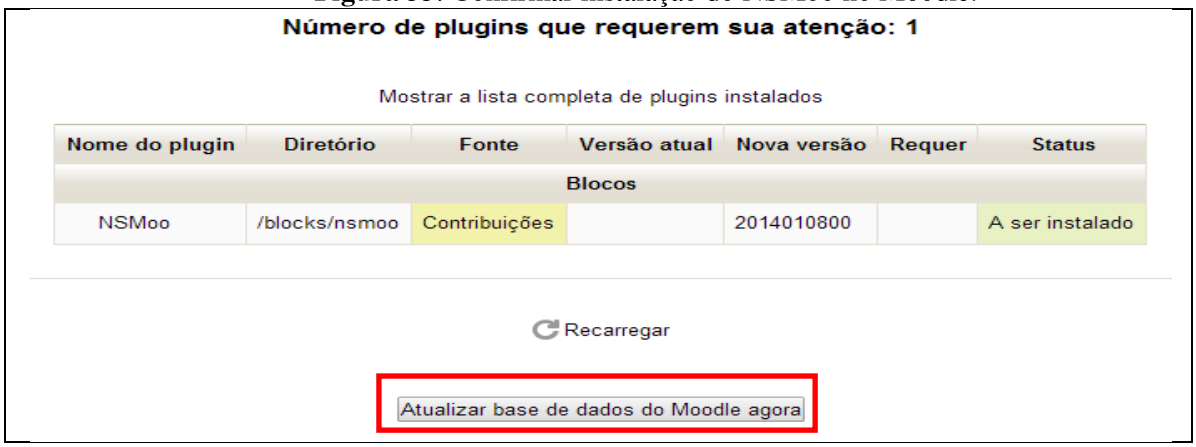
Figura 33: Ativando edições.

Figura 34: Adicionando bloco NSMoo ao Moodle



Em seguida, o usuário seleciona o bloco NSMoo (Figura 34), e pressiona o botão "Atualizar base de dados do Moodle agora" (Figura 35).

Figura 35: Confirmar instalação do NSMoo no Moodle.



Se os passos anteriores forem bem sucedidos, será apresentada a seguinte mensagem ilustrada na Figura 36.

Figura 36: Instalação realizada do NSMoo

INICIANDO AGENTES

Após a instalação do NSMoo no Moodle, o próximo passo é iniciar os Agentes Reativos que são responsáveis pela comunicação entre as ferramentas. Para isso no servidor que está instalado o NS2 crie uma pasta com o nome "*agente-NS*" e copie o arquivo *AgN.jar* para o diretório. Em seguida acesse o *shel*, localize a pasta criada e digite o seguinte comando para iniciar o AgN: *java -jar AgN.jar*.

O segundo passo é iniciar o Agente Moodle, para isso acesse pelo *shel* o diretório "Agentes" em "*blocks/nsmoo/lab/Agentes*". Em seguida digite o seguinte comando: *java -jar AgM.jar*.

Ao executar a linha de comando e pressionar "*Enter*" o agente AgM solicitará que seja informado o parâmetro de inicialização (Figura 37) e o nome do servidor que a *Agent Remot Management* está executando (Figura 38).

Figura 37: Parâmetro de inicialização.

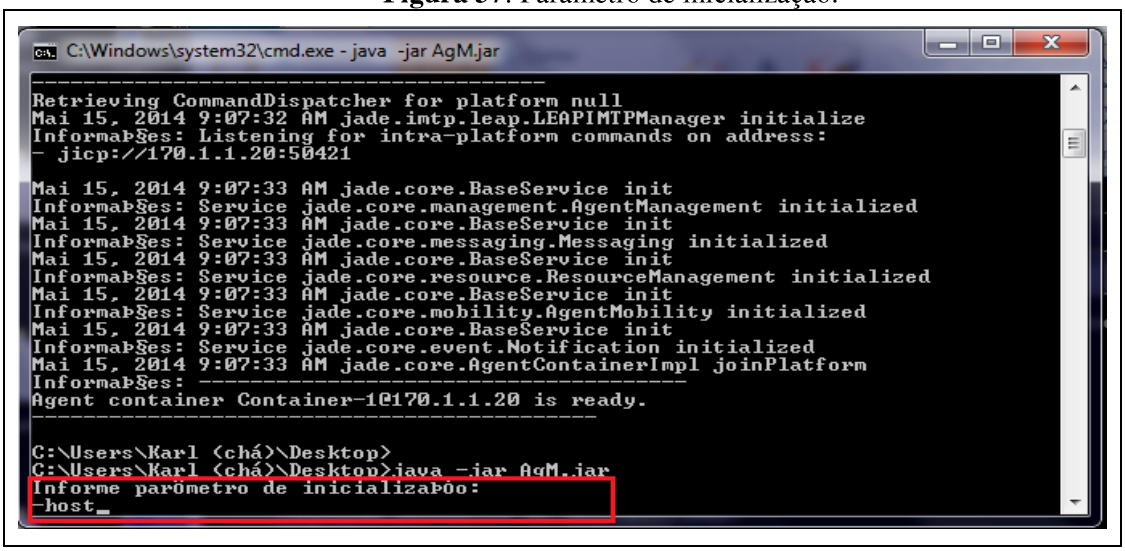
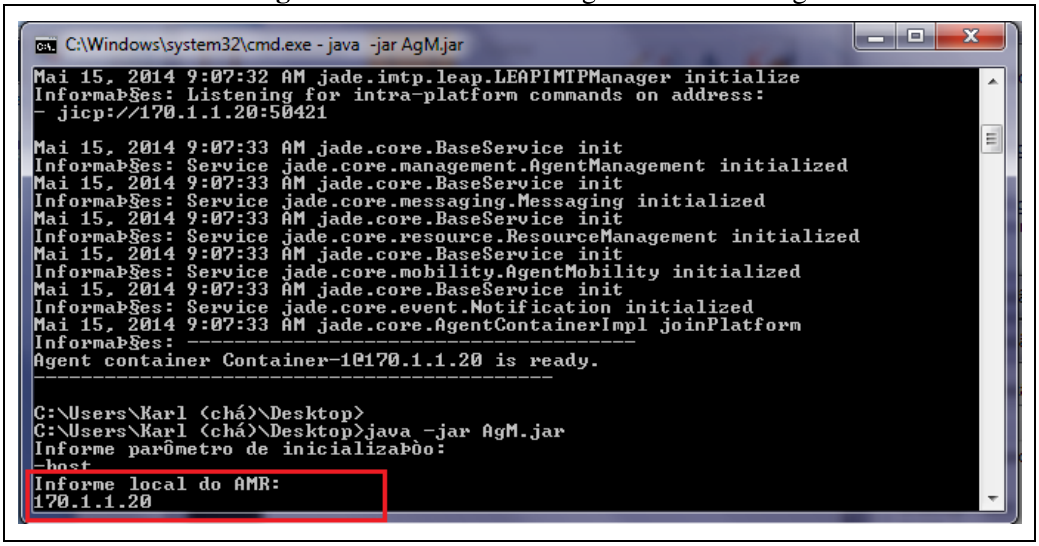


Figura 38: Localizando o Agent Remot Management.



Após os passos anteriores, será criado no ARM um *container* para que o AgM seja adicionado e a comunicação estabelecida entre os dois agentes.