

INSTITUTO FEDERAL DO PARANÁ

WELLITON FERNANDES LEAL

VERIFICAÇÃO DE IMPLEMENTAÇÃO DE PROTOCOLO DE REDE
POR MEIO DE TESTE BASEADO EM MODELO

LONDRINA

2015

INSTITUTO FEDERAL DO PARANÁ

WELLITON FERNANDES LEAL

VERIFICAÇÃO DE IMPLEMENTAÇÃO DE PROTOCOLO DE REDE
POR MEIO DE TESTE BASEADO EM MODELO

Trabalho de Conclusão de Curso
apresentado ao Curso Superior de
Tecnologia em Análise e
Desenvolvimento de Sistemas do Instituto
Federal do Paraná – Campus Londrina,
como requisito parcial de avaliação.

Orientador: Andréia Carniello

Co-orientador: Gilson Doi Junior

LONDRINA

2015

FOLHA DE APROVAÇÃO

WELLITON FERNANDES LEAL

VERIFICAÇÃO DE IMPLEMENTAÇÃO DE PROTOCOLO DE REDE

Trabalho aprovado como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas, ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Eixo de Comunicação e Informação, do Instituto Federal do Paraná, avaliado pela seguinte banca examinadora:

Orientador: Prof. Andréia Carniello
Instituto Federal do Paraná

Co-orientador: Prof. Gilson Doi Junior
Instituto Federal do Paraná

Prof. Adriana Carniello
Instituto Federal do Paraná

Prof. Marcel Toshio Omori
Instituto Federal do Paraná

Londrina, 02 de dezembro de 2015.

AGRADECIMENTOS

Agradeço a Deus por todas as conquistas em minha vida, pela força em momentos de dificuldade onde não me senti sozinho, pela superação onde aprendi a me desenvolver e ser melhor a cada dia. Obrigado por permitir que tudo isso seja possível, tornando-se realidade.

Muito obrigado aos meus orientadores Andréia Carniello e Gilson Doi Junior que sempre se colocaram à disposição para direcionar e solucionar dúvidas em todo o desenvolvimento do trabalho, por se preocuparem e tornarem possível o mesmo.

Aos professores que ao longo da graduação sempre estiveram empenhados em ensinar, ajudando não só em no desenvolvimento nas disciplinas, mas atendendo em todos os momentos de dificuldade e possibilitando crescer como discente e também como pessoa. Obrigado pela preocupação em ensinar da melhor forma. Muito obrigado em especial ao professor e orientador Gilson Doi Junior, que desde o início deste trabalho esteve sempre motivando, disposto e comprometido a ajudar, muito obrigado! Agradeço imensamente aos professores Flávio Navarro Fernandes, Diogo Roberto Olsen, Fernando Accorsi e Romualdo Rubens de Freitas, importantes em cada uma das etapas de minha formação.

Agradeço a minha família por tudo, obrigado pela educação, força e carinho, pelo apoio incondicional em todos os momentos. Muito obrigado em especial ao meu pai Valter Leal, minha mãe Maura C. Da Silva Leal e meu irmão Gustavo Henrique Leal. Vocês são a base de tudo para mim.

Ao grande amigo Emerson Pereira Alves, muito obrigado por todos os ensinamentos, pela confiança, amizade e respeito!

Aos meus colegas de sala que compartilharam todos os momentos de alegrias e sofrimentos, que passaram por todas as dificuldades e contribuíram ao longo desses anos, uns com os outros, obrigado por terem feito parte da minha história. Uma família se constituiu no início dessa graduação, uma família formada por alunos e professores, família que levarei para sempre, muito obrigado!

RESUMO

Através de simulações computacionais em redes de computadores, utilizando o *software Network Simulator 2*, foi gerado um arquivo de saída contendo os dados da simulação de rede. Com o arquivo de saída gerado foi constatada a viabilidade de automatizar a verificação de implementação de protocolo de rede. Dessa forma, o presente trabalho desenvolveu uma ferramenta, denominada MSRP, para automatização desse processo, no qual a verificação de implementação de protocolo de rede se dá através da análise do arquivo de saída do *software Network Simulator 2* e a comparação de um modelo comportamental, utilizando técnica de teste baseado em modelos, na qual é definindo uma modelo por meio de máquina de estados finitos e é comparado com os resultados da simulação. A ferramenta tem como objetivo realizar verificações em implementações de protocolo de redes de computadores.

Palavras-chave: Simulação de rede. Teste baseado em modelo. Protocolos de rede. *Network Simulator 2*.

LISTA DE ILUSTRAÇÕES

FIGURA 1 - Exemplo de representação textual de uma Máquina de estados finitos.	22
FIGURA 2 - Exemplo de uma representação gráfica de uma máquina de estados finitos.	22
FIGURA 3 - Exemplo de representação tabular de uma máquina de estados finitos.	23
FIGURA 4 - Máquina de estados finitos	23
FIGURA 5 - Múltiplas redes com dispositivos interconectados	25
FIGURA 6 - Rede de longa distância, WAN	25
FIGURA 7 - Imagem do modelo de referência OSI	27
FIGURA 8 - Modelo de referência OSI em comparação com o modelo TCP/IP	28
FIGURA 9 - Visão da comunicação de dados e das redes de computadores.	29
FIGURA 10 - Sequência de eventos do método de prototipação.	36
FIGURA 11 - Arquitetura do sistema MSRP.	38
FIGURA 12 - Interface gráfica, front-end do simulador GNS3.	40
FIGURA 13 - Interface <i>front-end</i> do <i>software</i> OPNET versão acadêmica.	41
FIGURA 14 - <i>Network Simulator 2</i> em execução via terminal.	43
FIGURA 15 - Estrutura dos objetos da simulação.	44
FIGURA 16 - Execução do arquivo Nam-Trace pela ferramenta Nam.	46
FIGURA 17 - Arquivo <i>trace</i> gerado pelo NS2.	48
FIGURA 18 - <i>Script</i> OTcl contendo as configurações de rede para a simulação pelo <i>software Network Simulator 2</i> .	51
FIGURA 19 - Diagrama de casos de uso da ferramenta MSRP.	53
FIGURA 20 - Diagrama de classes da ferramenta MSRP.	54
FIGURA 21 - Estrutura do código do projeto MSRP.	56
FIGURA 22 - Exibição da ferramenta MSRP desenvolvida.	57
FIGURA 23 - Exemplo de um modelo para verificação de implementação de protocolo.	58
FIGURA 24 - Modelo simples para checar protocolo TCP e suas confirmações de fluxo.	60
FIGURA 25 - Opção para importação do arquivo trace do NS2.	61

FIGURA 26 - Modelo de máquina de estados definida em XML para utilização na ferramenta MSRP.	62
FIGURA 27 - Seleção do arquivo trace pela ferramenta MSRP.	63
FIGURA 28 - Exibição de mensagem do arquivo trace carregado com êxito.....	64
FIGURA 29 - Arquivo trace corresponde ao modelo esperado e exibe mensagem.	65
FIGURA 30 - Mensagem de confirmação do relatório gerado pela ferramenta MSRP.	66
FIGURA 31 - Relatório com as informações ocorridas na verificação pela ferramenta MSRP.....	67
FIGURA 32 - Exemplo de um modelo que não corresponde ao funcionamento do protocolo.	68
FIGURA 33 - Opção para importação do arquivo <i>trace</i> do NS2.	69
FIGURA 34 - Seleção do arquivo <i>trace</i> pela ferramenta MSRP.	70
FIGURA 35 - Exibição de mensagem do arquivo <i>trace</i> carregado com êxito.....	71
FIGURA 36 - Arquivo <i>trace</i> não corresponde ao modelo esperado e exibe mensagem de atenção.....	72

LISTA DE TABELAS

TABELA 1 – Tabela com as informações do <i>trace</i> gerado pelo <i>Network Simulator</i>	
2.....	49

LISTA DE SIGLAS

ACK	- <i>Acknowledgement</i>
API	- <i>Application Programming Interface</i>
FTP	- <i>File Transfer Protocol</i>
GNS3	- <i>Graphical Network Simulator</i>
HTTP	- <i>Hyper Text Transfer Protocol</i>
IFPR	- Instituto Federal do Paraná
IP	- <i>Internet Protocol</i>
ISO	- Internacional Standards Organization
LBNL	- <i>Lawrence Berkeley National Laboratory</i>
MEF	- Máquina de estados Finitos
MSRP	- <i>Modeling System Routing Protocol</i>
MVC	- <i>Model View Controller</i>
NS2	- <i>Network Simulator 2</i>
OPNET	- <i>Optimized Network Engineering</i>
OSI	- <i>Open System Interconnection</i>
OTCL	- <i>Object Tool Command Language</i>
PC	- <i>Personal Computer</i>
TCL	- <i>Tool Command Language</i>
TCP	- <i>Transmission Control Protocol</i>
TTCN-3	- <i>Testing and Test Control Notation version 3</i>
UDP	- <i>User Datagram Protocol</i>
V&V	- Verificação e Validação
XML	- <i>Extensible Markup Language</i>

SUMÁRIO

1 INTRODUÇÃO	13
1.1 PROBLEMA	14
1.2 OBJETIVOS	14
1.2.1 Objetivo Geral	14
1.2.2 Objetivos Específicos	14
2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS	16
2.1 TESTE	16
2.1.1 Teste Baseado em Modelos.....	19
2.2 MÁQUINA DE ESTADOS FINITOS	20
2.3 REDE DE COMPUTADORES.....	24
2.3.1 Modelo OSI	26
2.3.2 Protocolo de Rede de Computadores	29
2.4 TESTE EM REDE DE COMPUTADORES.....	31
2.5 TRABALHOS RELACIONADOS	32
3 METODOLOGIA	36
4 RESULTADOS.....	39
4.1 DEFINIÇÃO DA FERRAMENTA DE SIMULAÇÃO DE REDES UTILIZADA	39
4.1.1 Dados da Simulação de Rede.....	47
4.2 CONFIGURAÇÃO DE SIMULAÇÃO	50
4.3 DESENVOLVIMENTO DA FERRAMENTA.....	52
4.4 EXEMPLO DE UTILIZAÇÃO DA FERRAMENTA MSRP	59
5 CONSIDERAÇÕES FINAIS	73
5.1 CONTRIBUIÇÕES.....	73
5.2 TRABALHOS FUTUROS	74
REFERÊNCIAS.....	75

1 INTRODUÇÃO

Para que seja possível estabelecer uma comunicação entre todos dispositivos interconectados em uma rede de computadores é necessário que regras sejam estabelecidas, cada dispositivo deve concordar com tais regras, estabelecendo um padrão de comunicação onde cada um dos dispositivos envolvidos é capaz de compreender uns aos outros. O conjunto de regras definidas para a comunicação entre dispositivos em uma rede de computadores é chamada de protocolo de rede.

É importante garantir que os protocolos de rede sejam implementados de forma correta e eficiente. No entanto, a verificação da implementação de protocolo de redes geralmente é feita manualmente. Para este fim, também existem *softwares* específicos, porém, eles não são utilizáveis na ferramenta *Network Simulator 2* (NS2). O NS2 é um simulador de eventos discreto que tem como objetivo a criação de simulações baseadas nos protocolos de rede, possibilita a geração de um arquivo *trace* que contem todos os dados da simulação realizada.

Visto a dificuldade de analisar o arquivo de *trace* do NS2 de forma eficiente, identificando os protocolos que estão ocorrendo em uma simulação de rede, sem a necessidade de o testador realizar uma verificação tupla a tupla de milhares de linhas, definiu-se a construção de uma ferramenta que se integre ao simulador de rede NS2. Esta possui o objetivo de verificar e validar uma implementação de protocolo, no qual seja possível, através do uso de teste baseado em modelos, utilizando a técnica de máquina de estados finitos, definir um modelo comportamental dos protocolos desejáveis de forma simples, através de arquivo XML ou a construção iterativa de um modelo comportamental por meio da própria ferramenta de verificação. Dessa forma, almeja-se que no processo de verificação de protocolos de uma simulação de redes haja uma diminuição da necessidade de um testador humano realizar uma verificação manual, pois eleva o gasto com recursos em teste.

Diante disso, este trabalho tem como objetivo principal o desenvolvimento da ferramenta *Modeling System Routing Protocol* (MSRP), que visa suprir essa demanda. Para desenvolvê-la, primeiramente foi feita uma pesquisa sobre os testes,

seus conceitos e possibilidades. Também foi feito um estudo sobre máquinas de estados. Em seguida, foi apresentado referencial teórico sobre redes de computadores, o modelo OSI e os protocolos de rede. Também foi discutido a aplicação de testes em redes de computadores. Por fim, foi disponibilizado um levantamento sobre as tecnologias existentes.

Dessa forma, desenvolveu-se neste trabalho uma ferramenta denominada MSRP para realizar a verificação de implementação de protocolos de rede.

1.1 PROBLEMA

O *software* de simulação de rede *Network Simulator 2* não contém uma opção de verificação de protocolo de rede que realize uma verificação a partir da simulação realizada ou, uma ferramenta que integre o sistema permitindo que um processo de verificação de protocolo de rede aconteça com fim de validar uma possível implementação de protocolo.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver uma ferramenta para validar simulações de protocolos de rede com relação à especificação de máquinas de estados finitos.

1.2.2 Objetivos Específicos

Os objetivos específicos para atingir o objetivo geral deste trabalho são:

- Analisar qual modelo de especificação comportamental já existente possibilita descrever uma simulação de rede.
- Analisar quais aspectos da rede são passíveis de serem mapeados em modelos formais.
- Analisar a possibilidade de integração a um *software* de simulações de rede de computadores.
- Aplicar conceitos de teste de *software* em simulações de redes de computadores.
- Desenvolver uma ferramenta que automatize o processo de verificação de implementação de protocolo de rede a partir de teste baseado em modelo.
- Aplicar a ferramenta de teste na área de redes de computadores a fim de verificar protocolos de rede desejáveis comparando-o com o modelo definido.
- A partir de simulações realizadas pelo *software Network Simulator 2* possibilitar a verificação de protocolo de rede desenvolvendo uma ferramenta que realize uma integração com o simulador a fim de verificar e validar uma implementação de protocolo de rede desejável, de maneira dinâmica, configurável, automatizando e facilitando a verificação do processo de verificação de protocolo de rede.

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

Verificações e testes manuais são mais suscetíveis a falhas, uma vez que estes processos são realizados por seres humanos. Testes baseado em modelos têm como benefício a redução das possibilidades de falhas no processo, elevam a eficácia do teste, além de proporcionar a redução de custo, tempo e recursos.

Estudos apresentados no capítulo 2.5 sobre teste baseado em modelos comprovam que estes conceitos da engenharia de *software* não estão sendo aplicados apenas em desenvolvimento de sistemas, são utilizados também em testes de redes de computadores no qual o objetivo pode ser analisar o seu comportamento em determinada configuração, identificar falhas ou possíveis ataques. Pesquisas na área de rede de computadores demonstram benefícios na utilização de ferramentas que utilizam modelos para a realização de diferentes tipos de análises e verificações em redes.

Neste capítulo são abordadas as áreas de aplicação do trabalho proposto, que são: teste de *software* e redes de computadores. Uma visão geral das áreas de aplicação é mostrada, bem como algumas especificidades das áreas tendo como objetivo delimitar o escopo deste trabalho na verificação de protocolos de redes através de teste baseado em modelos. Uma análise de trabalhos correlatos é desenvolvida e discutida neste capítulo, sendo esta análise utilizada para nortear o trabalho proposto.

2.1 TESTE

É de extrema importância a realização de teste em verificação e validação de sistemas (V&V). O processo de verificação e análise são atividades que ocorrem durante todas as etapas de construção de *software*, iniciando-se com as revisões de requisitos, continuando com o projeto e as inspeções de *software*. Desse modo, as etapas de verificação e validação de sistema são utilizadas para certificar o que foi desenvolvido, se está adequado ao problema e, ao mesmo tempo, atende os

requisitos do mesmo. Dessa forma, a verificação e validação e teste são processos totalmente diferentes, sendo que o processo de verificação tem como foco se está se construindo o *software* de forma correta, assim como foi especificado, enquanto o processo de validação está focado em se o *software* em questão está sendo construído corretamente. O teste é uma técnica que está contida em V&V e tem como finalidade descobrir falhas após o desenvolvimento do sistema (SOMMERVILLE, 2007).

A verificação possibilita uma melhor compreensão de todo o sistema que está sendo desenvolvido, se o mesmo está atendendo as especificações e se atende ou não os requisitos funcionais e não-funcionais. O processo de validação trata-se de uma abordagem mais ampla, cuja sua finalidade é garantir que o *software* atenda as expectativas do cliente.

O maior objetivo do processo de V&V é estabelecer a confiança de que o *software* ou o sistema verificado esteja adequado e coeso ao propósito para o qual foi desenvolvido. Este nível de confiabilidade depende do fim a que se presta o *software* e, também, das expectativas dos usuários que o utilizarão.

Dentro do processo de V&V é possível verificar duas abordagens para a verificação e análise de sistemas. Uma denominada de inspeção de *software* ou revisão por pares, nomeada também de V&V estática, na qual são analisadas e verificadas as representações do sistema como, por exemplo, um documento de requisitos, diagramas de projeto e código fonte do programa.

A segunda abordagem é a de testes de *software*, também conhecida como V&V dinâmica, como citado anteriormente. Envolve a execução do *software* com dados de testes a fim de checar as saídas do *software* de acordo com suas entradas, também o comportamento operacional do mesmo.

A verificação estática não envolve a execução do *software* e tem como atividades a análise de produtos desenvolvidos, a construção de protótipos e revisão dos documentos que devem ocorrer no final de cada fase do ciclo de vida do projeto do *software* (PRESSMAN, 2011).

A verificação dinâmica envolve a execução do código do *software*, o teste em tempo de execução do sistema desejado, com a finalidade de detectar defeitos ou erros. O teste de *software* pertence às atividades de análise dinâmica do *software*, sendo que na execução dessa atividade pode-se utilizar ferramenta de

automação de teste. Esta atividade de teste não é utilizada somente para os testes em *softwares* e também são aplicadas em diferentes problemas na computação, sendo possível realizar testes dinâmicos e obter resultados de forma eficaz como, por exemplo, em testes de redes de computadores onde são utilizados *softwares* que simulam e testam uma configuração desejável.

Além das técnicas de V&V e testes, outra técnica importante para a qualidade de *software* é a de Inspeção, que tem maior eficácia e, quando realizada de forma rigorosa pode remover até 90% dos defeitos antes que sejam executados os testes do respectivo material. Por outro lado, inspeções e testes, são formas complementares de apreciação, sendo que um não substitui o outro (PAULA FILHO, 2009).

O teste de *software* é o processo no qual o *software* é executado com a finalidade de encontrar falhas, utilizam-se técnicas de teste a fim de prover a qualidade do *software* ou produto desenvolvido. O teste é uma área na engenharia de *software* no qual seus conceitos são utilizados em outras áreas da computação que necessitam que sistemas sejam colocados a prova.

A técnica de teste de *software* é aplicada de forma diferente de acordo com o sistema e área na qual é utilizada. Para a devida correção, nesta fase devem-se descartar noções de utilização do *software* preconcebidas para elevar a eficácia do teste a ser aplicado.

Um teste de *software* bem-sucedido é aquele em que, de fato, encontram-se erros com uma baixa quantidade de tempo e o mínimo de esforços. Qualquer *software* pode ser testado de duas maneiras. A primeira é conhecendo-se a função específica que um *software* deve executar, dessa forma, testes podem ser realizados para demonstrar que cada função é totalmente operacional, ou seja, está funcionando de forma adequada, conforme o esperado. Esta técnica de teste é chamada de teste de caixa preta ou *Black box*, que foi utilizada no desenvolvimento deste trabalho. A expressão teste de caixa preta refere-se aos testes que são realizados quando não se conhece o funcionamento interno. Considerando *softwares* de computador, o teste de caixa preta pode, por exemplo, descobrir falhas mesmo que não se conheça o funcionamento interno do sistema.

Dessa forma, o teste de caixa preta é utilizado para demonstrar que as funções do *software* estão se comportando corretamente, que suas entradas são

adequadas e suas saídas são corretas, este tipo de teste se preocupa com os aspectos externos do sistema, sem se preocupar com as características lógicas e a estrutura interna do sistema (MYERS, 1979).

A segunda maneira de se realizar testes é conhecendo o funcionamento interno do produto, podem ser realizados para garantir que todas as partes do mesmo estejam se encaixando, que a operação interna do produto tenha um funcionamento de acordo com as especificações de desenvolvimento e que os componentes internos sejam adequadamente testados. Esta técnica de teste é chamada de teste de caixa branca.

O teste de caixa branca é realizado com base em um minucioso exame dos detalhes procedimentais. Os caminhos lógicos são testados, um número limitado de caminhos lógicos importantes pode ser selecionado e executado em busca de erros. Estes caminhos lógicos fornecem casos de teste nos quais põem a prova conjuntos específicos de condições e repetições. Este método é do projeto de casos de teste, que se utiliza da estrutura de controle do projeto procedimental para a derivação de casos de teste (PRESSMAN,1995).

A única maneira de se mostrar a correção de um *software*, ou de um processo ao qual este *software* se aplica, seria por meio da execução de um teste em que todas as combinações de valores de entrada do *software* sejam testadas, porém, esta prática é inviável, visto que o domínio dos dados de entrada é praticamente infinito ou, pelo menos, muito grande (MYERS, 1979).

2.1.1 Teste Baseado em Modelos

O teste baseado em modelos é uma técnica de teste funcional que utiliza informações sobre o comportamento funcional do *software*, descrito por um modelo comportamental para determinar como será realizado o teste. O teste baseado em modelos é definido como a automação do projeto de teste de caixa preta, o que difere do teste de caixa preta usual é que, ao invés de se escrever manualmente os casos de teste baseados em uma documentação de requisitos, é criado um modelo do comportamento do sistema em teste (OROZCO, 2009). O modelo

comportamental, construído utilizando os requisitos funcionais do mesmo, determina quais ações são possíveis em sua execução e quais as saídas são esperadas.

Existem diversas maneiras de se modelar o comportamento de um *software* ou processo, e para cada técnica de modelagem podem existir critérios distintos. Exemplos de critérios são todos os estados de uma máquina de estados finitos, todas as transições dessa máquina de estados e todos os seus caminhos, os quais exigem, respectivamente, que cada estado, cada transição e cada caminho do modelo comportamental seja executado pelo menos uma vez durante o teste.

A principal premissa por trás do teste baseado em modelos é a criação de um modelo, uma representação do comportamento do *software* a ser testado. Esse modelo deve descrever quais ações são possíveis e quais saídas são esperadas e, a partir dessa descrição podem ser aplicados critérios de seleção de casos de teste, os quais devem ser executados e avaliados.

O teste baseado em modelos não é a solução para todos os problemas da área, mas oferece uma promessa considerável na redução do custo de geração de teste, no aumento da eficiência dos testes e na redução deste ciclo. Esse tipo de teste pode ser eficaz para *softwares* que são alterados frequentemente, pois os testadores podem alterar o modelo comportamental e rapidamente gerar um novo conjunto de casos de teste.

Apesar de o teste baseado em modelos ser uma técnica funcional e ter sido desenvolvida principalmente para ser aplicada aos testes de sistema, é possível que a mesma seja aplicada a qualquer fase de teste e também a diferentes vertentes da computação que necessitam realizar comparações a fim de testar ou validar um comportamento. Dessa forma, ela pode ser aplicada aos testes de unidade, integração, validação e de sistema, desde que os modelos construídos representem o comportamento do *software* ou dos processos que serão colocados em teste no nível em que se deseja realizá-los (FANTINATO, 2003).

2.2 MÁQUINA DE ESTADOS FINITOS

A máquina de estados finitos, MEF, é um conceito matemático utilizado na

representação de programas ou circuitos lógicos possibilitando a visão geral dos estados que eles podem assumir, possibilitando uma visão de seu comportamento, dessa forma, é possível criar um modelo formal. As MEFs possibilitam uma grande variedade de representações de situações, utiliza informações de modelos comportamentais do *software* para a realização do teste, sendo também uma das técnicas de modelagem mais utilizadas no teste baseado em modelos, utilizadas, ainda, para a modelagem comportamental de *software*. Utilizam-se MEFs para facilitar o entendimento das definições que cada processo no sistema assume, os estados definem quais ações são permitidas aos processos e quais eventos são esperados pelo estado, também definindo como cada estado responderá a estes eventos (PASSOS, 2002).

Uma máquina de estados finitos possui uma quantidade finita de estados que pode assumir, sendo assumido apenas um por vez, que, quando ativo é chamado de estado atual. Para que os estados da máquina sejam alterados de um estado ativo para o próximo estado é necessário que ocorra alguma mudança, este acontecimento em uma máquina de estados é chamado de transição. As transições são as mudanças de estado que são compostas por eventos e ações. Um evento é correspondente a uma situação na qual permitirá que a transição ocorra, uma ação, por sua vez, é a consequência desse evento. Para cada estado podem existir zero ou mais transições, para cada transição existirão eventos que acarretarão em uma ação dentro da máquina de estados finitos (MASIN, 2013).

Os critérios de teste baseados nesta técnica analítica de modelagem podem considerar apenas aspectos de fluxo de controle do *software* modelado. Esta técnica é útil para modelar o comportamento de sistemas reativos, que são essencialmente dirigidos a eventos e dominados por controle.

Segundo Fantinato (2002), as máquinas de estados finitos possuem uma gama de aplicação bastante grande e genérica, podendo ser aplicada em diversos tipos de sistemas. Dessa forma as máquinas de estados são muito utilizadas em modelagens de diversos problemas como automação de design eletrônico, projeto de protocolos de comunicação, análise, entre outras situações em que uma máquina de estados poderá representar todo um comportamento.

MEFs podem ser classificadas como determinísticas ou não determinísticas, sendo que uma MEF determinística é determinada pelo estado, pois conhecendo o

estado atual é possível determinar o comportamento futuro do sistema para qualquer novo símbolo de entrada. O estado atual sumariza a história do sistema.

As MEFs não determinísticas são um modelo incompleto de um sistema, porém, mais compacto e cuja informação poderá ser suficiente para analisar o comportamento do mesmo.

Os elementos das MEFs podem ser representados de três formas:

Representação textual, no qual os elementos pertencentes à MEF são descritos textualmente, seguindo regras sintáticas, assim como representado na FIGURA 1.

Estados = {A, B, C}; Entradas = {Entrada 1, Entrada 2} Saídas = {Saída 1, Saída 2} Transição1 (A, Entrada 1) = (B, Saída 1) Transição2 (B, Entrada 2) = (C, Saída 2) Transição3 (C, Entrada 2) = (A, Saída 1)
--

FIGURA 1 - Exemplo de representação textual de uma Máquina de estados finitos.

Fonte: FANTINATO, 2002.

A FIGURA 2 é referente à representação gráfica de uma máquina de estados, chamada de grafo de estados, útil para a visualização dos elementos pertencentes às MEFs.

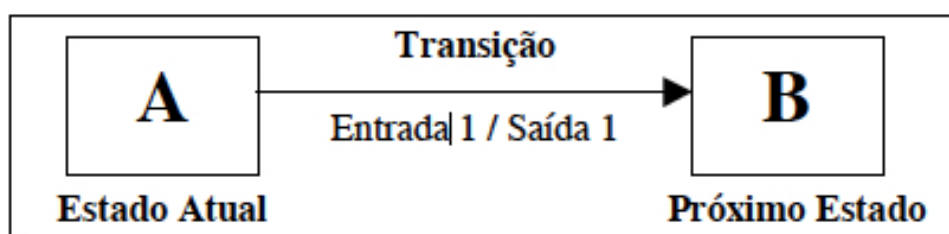


FIGURA 2 - Exemplo de uma representação gráfica de uma máquina de estados finitos.

Fonte: FANTINATO, 2002.

A representação tabular é uma tabela de transições de estados, úteis para o

armazenamento e manipulação computacional das informações que estão contidas na MEF, como mostrado na FIGURA 3.

	Entrada 1	Entrada 2
A	B / Saída 1	-
B	-	C / Saída 2
C	-	A / Saída 1

FIGURA 3 - Exemplo de representação tabular de uma máquina de estados finitos.

Fonte: FANTINATO, 2002.

Na FIGURA 4 é representada, de forma simples, uma máquina de estados finitos retratando a catraca de um ônibus. As setas que saem de um estado para o outro representam as transições, os estados possíveis nesta máquina de estados finitos são os estados de travada e destravada.

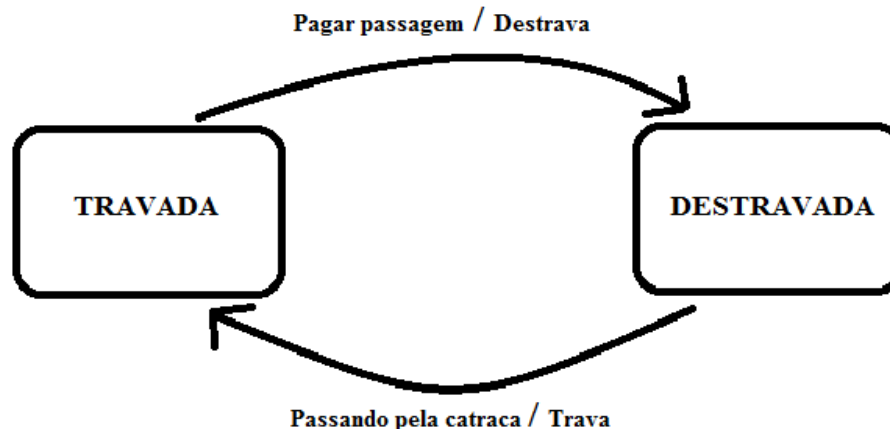


FIGURA 4 - Máquina de estados finitos

Fonte: MASIN, 2013.

Para que uma mudança de estado ocorra é necessário que uma ação aconteça, essa ação, na máquina de estados, refere-se ao evento, ou seja, de acordo com o evento ocorrido uma transição que atenda a condição levará ao próximo estado. Na FIGURA 4, por exemplo, o estado atual na máquina de estados é TRAVADA, para que o estado se altere para DESTRAVADA é necessário que a ação PAGAR PASSAGEM ocorra. Nesse momento a máquina de estados altera o

seu estado atual para DESTRAVADA. Novamente, para que o estado atual da máquina de estados se altere para o próximo estado, TRAVADA, é necessário que a ação PASSAR PELA CATRACA aconteça e, então, o estado será alterado para TRAVADA, que será o estado atual da MEF.

2.3 REDE DE COMPUTADORES

As redes de computadores são um conjunto de mecanismos que possibilita o transporte de dados de uma origem até um destino (MORAES, 2013).

Uma rede de computadores é a conexão entre dispositivos que podem estar em diferentes espaços geográficos, mas que se encontram interligados por um meio físico, trata-se de uma combinação de hardware e *software* que estabelece uma comunicação e envia dados entre entidades que podem estar executando diferentes sistemas operacionais, sendo que uma entidade pode ser qualquer dispositivo capaz de enviar ou receber estes dados na rede. (FOROUZAN, 2007).

Para que uma comunicação de fato aconteça na rede é necessário que exista um Emissor, também um canal de comunicação, o meio ao qual estes dispositivos enviarão e receberão os dados transmitidos, também o Receptor, fechando o ciclo do fluxo de dados.

Em uma rede de computadores, onde dispositivos compartilham dados o Emissor é aquele que no momento da comunicação está transmitindo estes dados, enviando-os pelo canal de comunicação ao qual encontra-se conectado aos demais dispositivos da rede. O canal é simplesmente o meio em que estes dispositivos se encontram interconectados, os canais podem ser por exemplo, cabo de rede, fibra ou ar. O Receptor é o dispositivo que no momento da comunicação está recebendo os dados transmitidos pelo Emissor.

É importante saber que em uma comunicação em rede ora os dispositivos são emissores ora são receptores, ao menos que sejam programados para somente enviar ou receber dados.

A FIGURA 5 representa a interconexão entre diferentes tipos de redes.

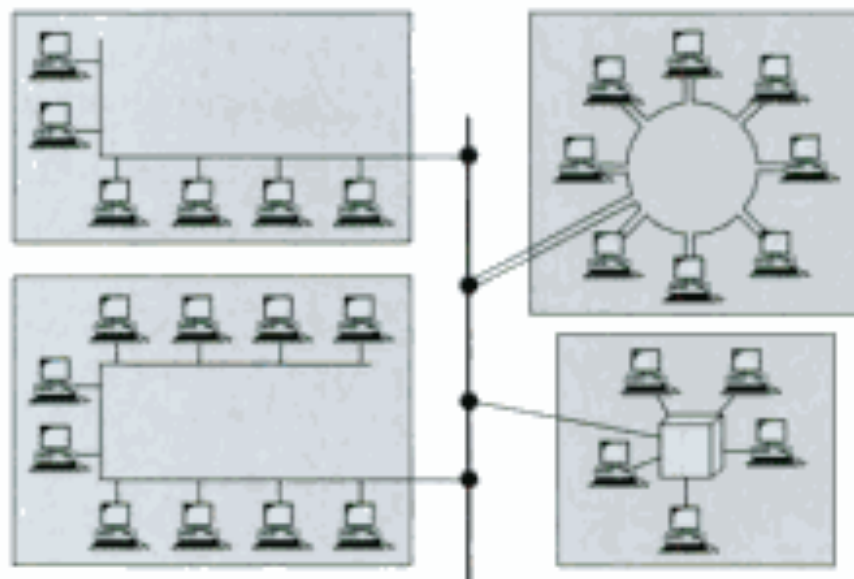


FIGURA 5 - Múltiplas redes com dispositivos interconectados.

Fonte: FOROUZAN, 2006.

A FIGURA 6 demonstra uma rede de longa distância entre computadores.



FIGURA 6 - Rede de longa distância, WAN.

Fonte: FOROUZAN, 2006.

2.3.1 Modelo OSI

O modelo OSI foi desenvolvido pela *Internacional Standards Organization*, a (ISO), como proposta para padronização das regras de comunicação aplicadas as diversas camadas em redes, pois tem como objetivo a interconexão de sistemas que sejam abertos à comunicação com outros sistemas, estabelecendo um padrão bem definido de regras onde dispositivos de diferentes fabricantes possam independentemente de seu sistema operacional, tipo e sua finalidade estabelecer uma conexão e comunicar-se de forma igualitária, para isso sendo apenas necessária a existência de uma conexão de rede e o padrão devidamente implementado.

O modelo OSI propriamente dito não se trata de uma arquitetura de rede na qual dita especificamente quais serviços e protocolos devem ser utilizados em cada uma das camadas, pelo contrário, o modelo trata apenas de como deve ser feita está comunicação entre cada uma das camadas existentes e o que cada uma delas deve fazer. No entanto, a *Internacional Standards Organization* produziu também padrões para cada uma das camadas do modelo OSI, cada padrão foi publicado como um padrão internacional distinto (TANENBAUM, 2003).

A ideia de divisão por camadas do modelo OSI tem como objetivo reunir informações relacionadas em grupos discretos, cada uma dessas divisões torna-se camadas neste modelo. Cada uma das camadas define um conjunto de funções distintas das demais camadas. Através dessa divisão foi criada a arquitetura em camadas, esta que define o funcionamento do modelo de referência OSI. As camadas estão interligadas pois o funcionamento deste modelo depende dos serviços específicos de cada uma das camadas, de maneira em que cada camada faz o uso dos serviços da camada inferior a ela e fornece seus serviços a camada superior. Por exemplo, a camada 3 utiliza os serviços providos pela camada 2 e consequentemente servindo serviços para a camada superior 4.

O modelo OSI é tido como referência em interconexão de dispositivos e é utilizado como base para criação de novas camadas, nas quais são criadas quando

há necessidade de outro grau de abstração, estas por exemplo, contendo regras específicas para a implementação de uma aplicação. O modelo de referência OSI é constituído por sete camadas, a sua ordem inicia-se pela camada inferior. A primeira camada é a física, seguida da camada de enlace, camada de rede, camada de transporte, camada de sessão, camada de apresentação e a sétima e última, camada de aplicação.

A FIGURA 7 representa a ordem das camadas no modelo OSI, a sequência na qual estão organizadas.



FIGURA 7 - Imagem do modelo de referência OSI.

Fonte: FOROUZAN, 2006.

Além do modelo OSI que conta com uma divisão entre sete camadas de rede, visto a cima neste capítulo, também existe um modelo específico para comunicação de dados e conectividade entre um emissor e um receptor em uma *internetworking*, este é reconhecido como o modelo da internet ou pilha de protocolos TCP/IP. Este modelo é composto por cinco camadas, assim como no modelo OSI a ordenação das camadas inicia-se pela camada mais inferior indo até a quinta e última camada deste modelo. O modelo TCP/IP tem como referência o modelo OSI e por este motivo a estrutura em camadas entre estes dois modelos são muito semelhantes. As diferenças entre o modelo OSI e o modelo de internet TCP/IP está na não utilização de algumas camadas. Assim como visto na imagem a cima do

modelo de referência OSI, as camadas de apresentação e sessão que compõem o modelo OSI não estão presentes no modelo voltado a internet TCP/IP (FOROUZAN, 2006; TANENBAUM, 2013).

Os modelos de referência OSI e TCP/IP baseiam-se no conceito de pilhas de protocolos independentes, são muito semelhantes em sua arquitetura, suas camadas são encarregadas de executar praticamente as mesmas funções, como visto a cima neste capítulo, o modelo TCP/IP está contido no modelo OSI.

A FIGURA 8 demonstra a estrutura do modelo OSI em comparação com o modelo TCP/IP.

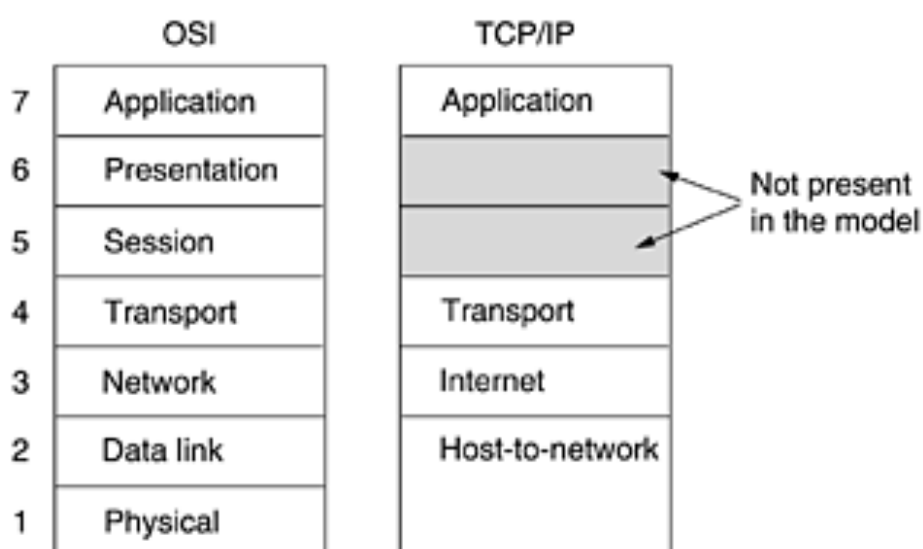


FIGURA 8 - Modelo de referência OSI em comparação com o modelo TCP/IP.

Fonte: TANENBAUM, 2003.

A FIGURA 9 demonstra uma visão da comunicação de dados e das redes de computadores, é possível visualizar a estrutura das camadas do modelo OSI e também do modelo TCP/IP em funcionamento conjunto, o nível onde cada uma das camadas destes modelos trabalham e quais camadas se comunicam entre si.

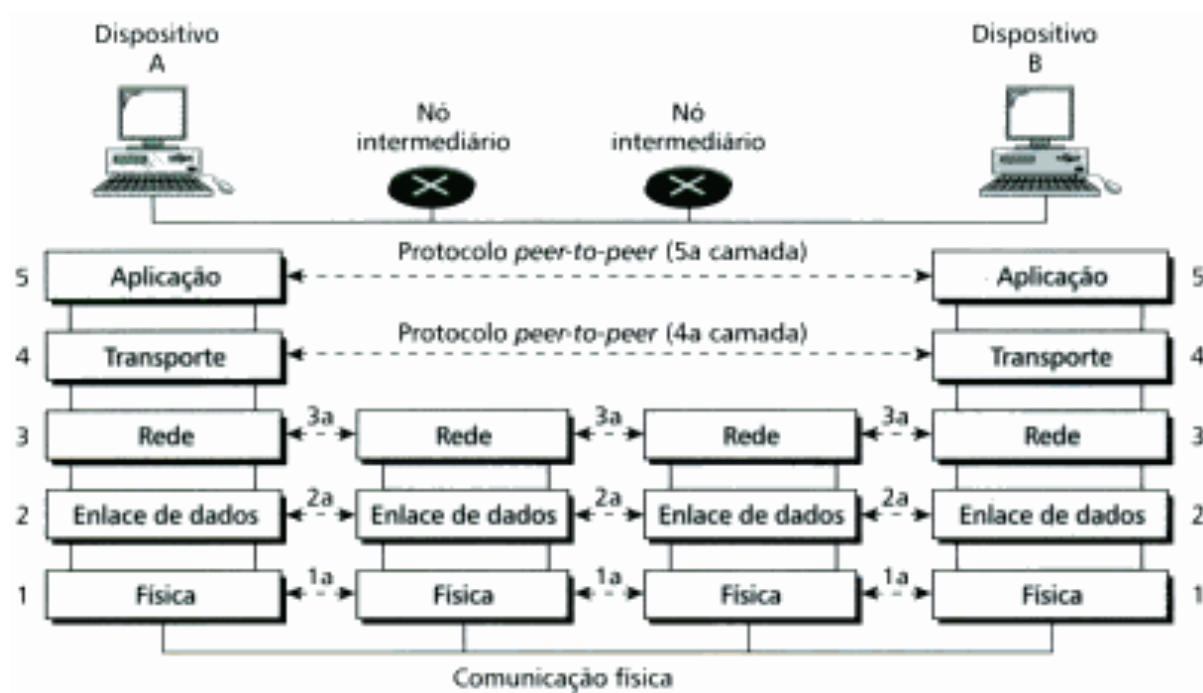


FIGURA 9 - Visão da comunicação de dados e das redes de computadores.

Fonte: FOROUZAN, 2006.

2.3.2 Protocolo de Rede de Computadores

Entidades em uma rede não podem simplesmente enviar fluxos de dados entre elas e esperarem que sejam compreendidas. Por este motivo é necessário que existam regras, que essas regras sejam estabelecidas, nas quais as entidades devem concordar com tais regras para que seja possível se estabelecer uma comunicação de forma em que todas entidades envolvidas nesta interconexão sejam capazes de compreenderem uma as outras. O protocolo de rede é o conjunto de regras que tem como objetivo viabilizar a conexão entre as entidades, é o meio lógico para a comunicação, nele contém definições de como esta comunicação deve funcionar e especifica como as entidades devem tratar o fluxo de dados.

Em rede de computadores a comunicação entre diferentes entidades interconectados utilizando diferentes sistemas operacionais é comum, por este motivo foi definido o conjunto de regras nas quais foram padronizadas. Esse conjunto de regras padronizadas e bem definidas formam um protocolo no qual

especificam o formato de dados e procedimentos a serem seguidos, também determina como um programa deve prepara-los e a maneira como deverão ser enviados para o estágio seguinte do processo de comunicação (MENDES, 2007).

Existem três elementos-chave em um protocolo, são eles a sintaxe, semântica e *timing* ou temporizador.

A sintaxe refere-se a forma, formato na qual o protocolo apresenta os dados, a estrutura, por exemplo, em um protocolo simples poderia ser especificado que em seu primeiro *byte* faz referência ao endereço de origem, o segundo *byte* ao endereço de destino, e o restante do fluxo de dados contidos no protocolo seriam referentes a mensagem propriamente dita que está sendo transmitida.

A semântica é referente ao significado, revelando o que de cada conjunto ou sessão de *bits* representa no protocolo. A semântica também define como um padrão particular será interpretado e qual ação será executada com base nessa interpretação.

O *timing* ou temporizador está relacionado ao tempo, este tempo é referente a duas características, sendo a primeira o tempo quando os dados devem ser enviados e a segunda característica ao tempo em o quão rápido esses dados podem ser enviados (FOROUZAN, 2006).

Há diversos tipos de protocolos utilizados para comunicação em redes de computadores e equipamentos eletrônicos. Quaisquer dispositivos que tenham capacidade de conexão a uma rede, interconexão, utilizam-se de um determinado tipo de protocolo de comunicação que promove comunicação com o restante da rede a qual está conectado. As implementações de protocolos costumam ser complicadas e por este motivo existem pesquisas com o objetivo de descobrir técnicas matemáticas formais para a especificação e verificação, tanto na camada de enlace de dados como para as demais camadas do conjunto (TANENBAUM, 2003).

Um protocolo específico de rede é implementado para tratar dados entre uma camada e outra de um modelo implementado. As camadas são responsáveis por receber segmentos de dados, encapsular e enviar para a camada seguinte, dessa forma possibilitando a transferência dos dados recebidos desde a camada mais inferior até a mais superior. Assim podemos considerar que a concepção de um protocolo tem como base a resolução dos dados de sua camada específica.

Cada protocolo tem diferentes responsabilidades conforme a camada em que está implementado.

Pela possibilidade de desenvolver aplicações distribuídas que sejam úteis é que surgiu a necessidade de projetar algoritmos, equipamentos e protocolos de rede. Dentro do conceito da rede mundial de computadores, mais especificamente, das redes TCP/IP, estes protocolos são os chamados protocolos da camada de aplicação. A internet, rede mundial de computadores, e outros tipos de redes TCP/IP, oferecem dois tipos de serviços às suas aplicações. Um serviço é o chamado de orientado a conexão, e o outro, não orientado a conexão. O Serviço orientado a conexão é implementado pelo TCP (*Transmission Control Protocol*), por sua vez este protocolo garante que os dados transmitidos do emissor, nó origem, ao receptor, nó destino sejam entregues na ordem a qual foram enviados e completos. O serviço da internet não orientado a conexão é fornecido pelo protocolo UDP (*User Datagram Protocol*), este por sua vez não faz a confirmação dos dados enviados, logo não é possível saber se os dados enviados pelo nó origem foram entregues ao nó destino. Estes são serviços da camada de transporte.

Grande parte das aplicações conhecidas na internet utilizam do TCP, como por exemplo o protocolo FTP (*File Transfer Protocol*) e HTTP (*Hyper Text Transfer Protocol*). O protocolo UDP por exemplo é utilizado principalmente para aplicações multimídia, como por exemplo *streaming* de áudio e vídeos.

2.4 TESTE EM REDE DE COMPUTADORES

Em rede de computadores também são utilizadas as técnicas de teste, assim como no desenvolvimento de *software* onde o produto deve ser constantemente testado. Redes de computadores necessitam da execução de técnicas de testes para comprovar ou validar o seu funcionamento.

Aplicações de teste em redes de computadores têm como objetivo testar desde sua estrutura, a conexão entre as entidades envolvidas na rede, os tipos de conexões, o fluxo de banda entre cada nó e também os protocolos que se fazem presente na comunicação. Os testes em redes de computadores podem ser

realizados empiricamente, onde toda a estrutura a ser testada é montada fisicamente e colocada a prova. O grande problema de testes empíricos ocorre quando uma configuração é extensa e, ou complexa, como por exemplo, quando composta por uma grande quantidade de nós, tipos de enlace diferentes e largura da banda que variam entre cada nó na rede. Um teste nestas configurações pode consumir muito tempo, além de contar com uma probabilidade de erros pois, o funcionamento correto de toda a estrutura depende também de cada componente físico envolvido no teste, o que não estão livres de defeitos e podem causar falhas durante um teste deste tipo.

Além de testes empíricos, uma técnica de teste comumente utilizada em testes em redes de computadores é a simulação de redes, esta proporciona benefícios como redução de tempo de teste, exclusão de falhas físicas e redução significativa dos gastos com recursos. A técnica de simulação será utilizada neste trabalho possibilitando realizar uma implementação de uma estrutura de rede de forma lógica sem a necessidade de executar fisicamente todos os componentes de uma configuração de rede desejada. As simulações de rede também são utilizadas para proporcionar uma visão aos usuários do comportamento de toda a rede, suas divisões e subdivisões até mesmo informações de tráfego, protocolos e possíveis falhas.

Os simuladores de redes são de total importância também para verificar o desempenho de uma determinada configuração. A simulação é um processo que visa o entendimento de um modelo real de uma rede de computadores, sendo possível realizar experimentos com este modelo com a finalidade de verificar e analisar o seu comportamento.

2.5 TRABALHOS RELACIONADOS

Um levantamento bibliográfico foi realizado buscando analisar teste de *software* aplicado à redes de computadores. Nesta seção são descritos os trabalhos correlatos e são apresentadas as análises sobre os mesmos, bem como

Teste Baseado em Modelo utilizando TTCN-3

No trabalho realizado por Wu-Hen-Chang et al (2011), foi proposto o desenvolvimento de uma ferramenta na qual realize a criação de modelos abstratos de sistemas a serem testados a partir da utilização da linguagem TTCN-3 por ser uma linguagem bem difundida na área de teste. O método de verificação utiliza-se da técnica de máquina de estados finitos estendidas para a criação de um modelo. A partir da criação de um modelo utilizando a TTCN-3 uma comparação poderá ser realizada.

Geração Automática de esquemas de Teste de Penetração e Modelo de Análise de Rede

No trabalho desenvolvido por Shen et al (2011), foi apresentado um novo esquema de teste de penetração em redes de computadores. Foi criado um modelo para geração de testes de penetração para redes de computadores de forma automática.

A ferramenta visa automatizar a geração de esquemas de teste de penetração, permitindo a geração automática de um modelo de análise de rede. Foi desenvolvido um protótipo para a verificação do modelo proposto e baseado em uma verificação protótipo do sistema o grafo de penetração pode ser gerado no qual pode ser utilizado para obter-se o caminho de penetração em uma rede de computadores.

VERA – Ferramenta de Teste Baseado em Modelo

No trabalho desenvolvido por Blome et al (2013), apresenta-se uma proposta de desenvolvimento de uma ferramenta chamada VERA na qual seu objetivo é realizar testes em resultados obtidos sobre cada caminho alcançado pelo modelo de máquina de estados finitos. A finalidade desta ferramenta é testar se há alguma anormalidade que possa causar uma vulnerabilidade ou um possível ataque.

QuickCheck – Gerador de Casos de Teste Automático para TCP

No trabalho desenvolvido por PARIS e ARTS (2009), foi apresentado um gerador de casos de teste automático para TCP utilizando a ferramenta QuickCheck. Este testador gera fluxos de pacotes para testar características específicas de uma pilha TCP, em seguida, faz o controle da pilha que se encontra em teste, possibilitando analisar e testar uma interface específica, como por exemplo um socket e os envios de respostas aos pacotes criados pela pilha sobre teste.

Neste trabalho de pesquisa são utilizados dados de trabalhos relacionados como base de comprovação da construção de uma ferramenta para verificação de protocolo de rede através da comparação por modelo.

As ferramentas e propostas apresentadas nesta seção, apesar de automatizarem parcialmente o processo de teste aplicado a redes de computadores acabam sendo complexos e específicos a um determinado protocolo ou linguagem para criação de um modelo.

No trabalho de Teste Baseado em Modelo utilizando TTCN-3 o testador deve ter conhecimento em uma linguagem específica, a TTCN-3, para a construção de um modelo, o que se torna um ponto negativo para testadores que não conheçam ou dominem esta linguagem. O trabalho de Geração Automática de Esquemas de Teste de Penetração e Modelo de Rede, apesar da automatização dos esquemas de teste e a geração de modelos com base nos esquemas gerados, requer a realização de duas tarefas para a verificação de possíveis invasões na rede, o que é trabalhoso, visto que para realizar as verificações é necessário gerar esquemas de teste desejados e os modelos para a análise. Em relação à verificação dos eventos esperados ainda é necessário a existência de um humano para avaliar os resultados.

A ferramenta VERA proposta é voltada para análise em serviços web, com características cliente-servidor, sua desvantagem encontra-se na utilização específica para comunicação neste tipo de estrutura de comunicação, na qual não se aplica a outras estruturas de rede.

Tendo em vista algumas características identificadas nos trabalhos correlatos que referem-se a considerar somente protocolos de rede específicos e utilizar linguagens de criação de modelos específicas e complexas, propõe-se, neste trabalho, uma arquitetura de verificação de protocolos de rede que se aplica à qualquer protocolo de rede, inclusive para validar a concepção de novos protocolos. Esta validação é realizada com relação à conformidade de um modelo descrito pela própria ferramenta utilizando a técnica de máquina de estados finitos ou através de um arquivo em XML que contenha as definições do modelo, viabilizando, assim, que a arquitetura proposta seja implementada.

3 METODOLOGIA

Para o desenvolvimento deste trabalho foi adotado o modelo de Prototipação. Este método de desenvolvimento consiste em criar um modelo do *software* que será implementado. O modelo pode assumir três formas: um protótipo em papel ou modelo baseado em PC no qual retrata a interação homem-máquina de forma que possibilite o usuário entender todas as interações possíveis; um protótipo de trabalho que implementa alguma parte, subconjunto, da função exigida do *software*; ou um programa já existente que executa partes ou todas as funções desejadas, mas que apresente outras características que serão melhoradas, desenvolvidas (PRESSMAN, 2006).

A Figura 10 ilustra a sequência de eventos para o desenvolvimento utilizando o paradigma de prototipação.

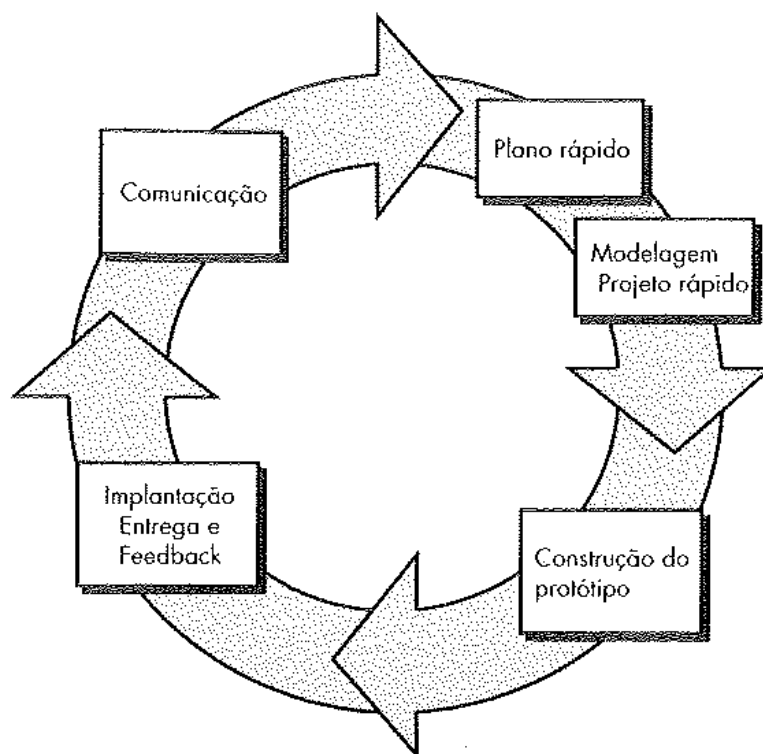


FIGURA 10 - Sequência de eventos do método de prototipação.

Fonte: PRESSMAN, 2006.

A prototipação inicia-se seu processo com a coleta dos requisitos. Os objetivos são definidos a partir de uma reunião entre o desenvolvedor e o cliente,

Através da discussão sobre os objetivos e exigências do sistema, um projeto rápido é elaborado com a finalidade de representar os aspectos de *software* que serão visíveis ao usuário, as abordagens de entrada e os formatos de saída. O projeto rápido leva à construção de um protótipo que será utilizado para uma avaliação pelo cliente, usuário, dessa forma é possível ajustar e refinar o sistema que está sendo desenvolvido.

As áreas de aplicação deste trabalho de pesquisa, que são: engenharia de *software* e redes de computadores, foram analisadas com o objetivo de identificar como a engenharia de *software*, por meio de teste de *software*, pode oferecer suporte à área de redes de computadores na verificação e validação de protocolos de rede. Pesquisou-se por trabalhos correlatos e estes foram analisados e discutidos, visando obter um direcionamento para este trabalho.

O levantamento bibliográfico deste trabalho foi realizado em bases de pesquisas como ACM Digital, IEEE, Google Scholar e também em livros.

Com base nas informações obtidas através do levantamento bibliográfico, definiu-se um simulador de redes de computadores que atendeu aos requisitos pré-definidos para o desenvolvimento do trabalho. A partir da definição do mesmo foi analisada as funcionalidades, comportamento e limitações do mesmo.

Foi definido um *framework* no qual possibilite a criação de um modelo de forma iterativa que permita uma manipulação descomplicada através da própria ferramenta, permitindo a criação e configuração do modelo de acordo com as necessidades de teste.

Utilizando a linguagem Java foi realizado o desenvolvimento de uma ferramenta com suporte a arquivos de saída referentes as simulações executadas pelo simulador NS2, nomeada *Modeling System Routing Protocol* (MSRP). Para a codificação desta ferramenta foi aplicado a metodologia de desenvolvimento de Prototipação. O padrão *Model View Controller* (MVC) foi utilizado no desenvolvimento da arquitetura do *software*.

Por meio da utilização dos padrões definidos pela UML, foi utilizado os diagramas que retratam a ferramenta proposta através do modelo de casos de uso, para definição das funcionalidades do sistema e, diagrama de classes para a modelagem da codificação do sistema proposto.

A FIGURA 11 retrata a arquitetura na qual será adotada para a construção

da ferramenta, contará com um modulo de comparação denominado Checker que realiza a checagem do arquivo de entrada *Trace*, gerado pelo simulador NS2 e, um módulo que permitirá a construção do modelo baseado em máquina de estados finitos denominado Mef Gráfica.

Os arquivos de entrada para o sistema serão arquivos de saída do NS2 e arquivos de definições de máquinas de estados finitos em XML.

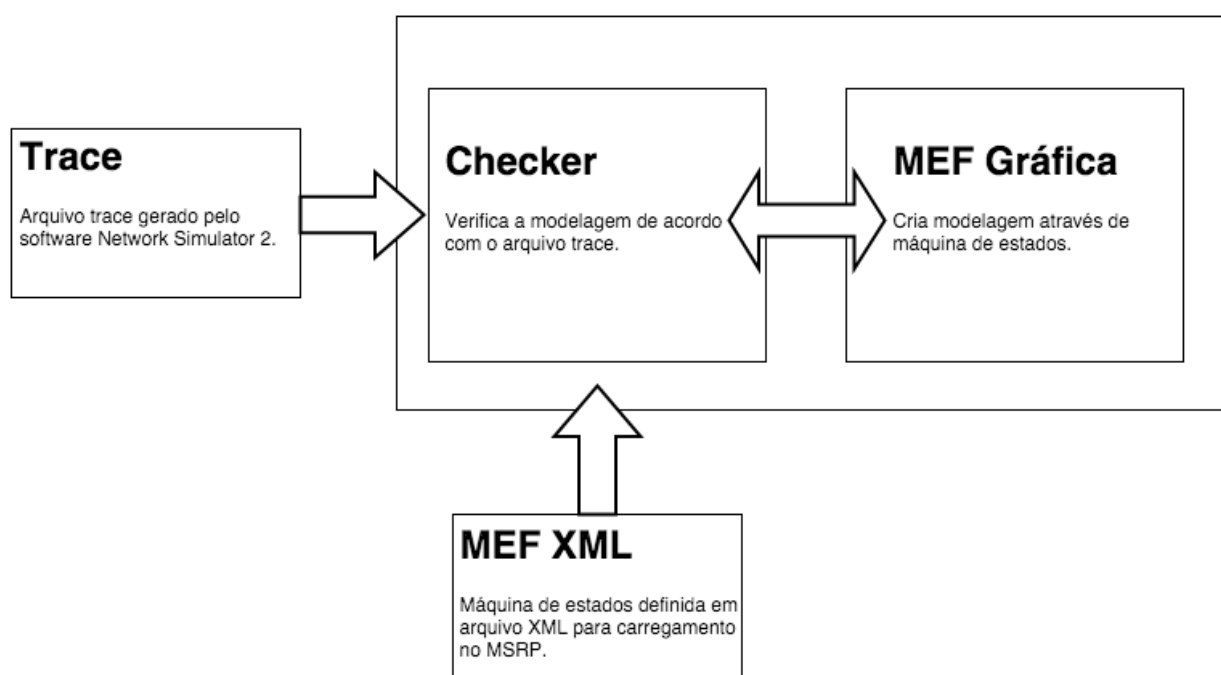


FIGURA 11 - Arquitetura do sistema MSRP.

Fonte: Autoria própria.

4 RESULTADOS

Com base no levantamento bibliográfico realizado, foram analisadas formas na qual a técnica de máquina de estados finitos é utilizada pela engenharia de *software*, analisando também a utilização deste modelo para resoluções de diferentes problemas em verificação e teste e como são aplicadas a área de redes de computadores.

4.1 DEFINIÇÃO DA FERRAMENTA DE SIMULAÇÃO DE REDES UTILIZADA

Existem diversos *software* desenvolvidos voltados para a realização de simulações de redes de computadores, para o desenvolvimento deste trabalho foram analisados três *softwares* para simulação de ambientes de redes utilizados em larga escala, são eles: GNS3 (*Graphical Network Simulator*, 2015), OPNET (*Optimized Network Engineering Tools*, 2011) e NS2 (*Network Simulator 2*, 2011). Essas ferramentas são utilizadas para a definição lógica de uma estrutura de redes de computadores e para simulação de tráfego de dados nessas redes. A partir da simulação, é possível realizar análises a respeito dos modelos de rede e seu funcionamento antes de uma implementação real, física.

A escolha para a utilização do *software* que mais se adeque ao trabalho foi realizada através de pesquisas com resultados de trabalhos correlacionados sobre testes em redes, estes com diferentes objetivos como por exemplo teste de análise de fluxo, teste de invasão de rede, entre outros. Dessa forma foi levada em consideração para a escolha do *software* características como licença de utilização, aplicabilidade, portabilidade e também se é gerado um arquivo de saída que possa ser utilizado para fins de análise das simulações realizadas.

O *software* GNS3 é um simulador *front-end* gráfico de licença gratuita que permite gerenciar simulações de redes complexas, para processadores MIPS, PowerPC e X86. Permite a partir do processador selecionado é executar imagens de sistemas operacionais destinadas a roteadores, como a IOS (*Internetwork Operation*

System) da fabricante Cisco, dos roteadores da série 1700, 2600, 3600, e 7200 ealém de possibilitar a execução de imagens destinadas ao *firewall* Cisco PIX e do JunOS destinadas aos roteadores Juniper.

A simulação pelo GNS3 possibilita a configuração dos dispositivos suportados, a ligação entre eles definindo os nós, seus enlaces, tráfego de conexão, os protocolos utilizados, toda configuração podendo ser realizada de maneira idêntica aos dispositivos suportados pelo *software*.

A FIGURA 12 exhibe a interface do *software* GNS3, através dessa interface são criadas e definidas as configurações das simulações.

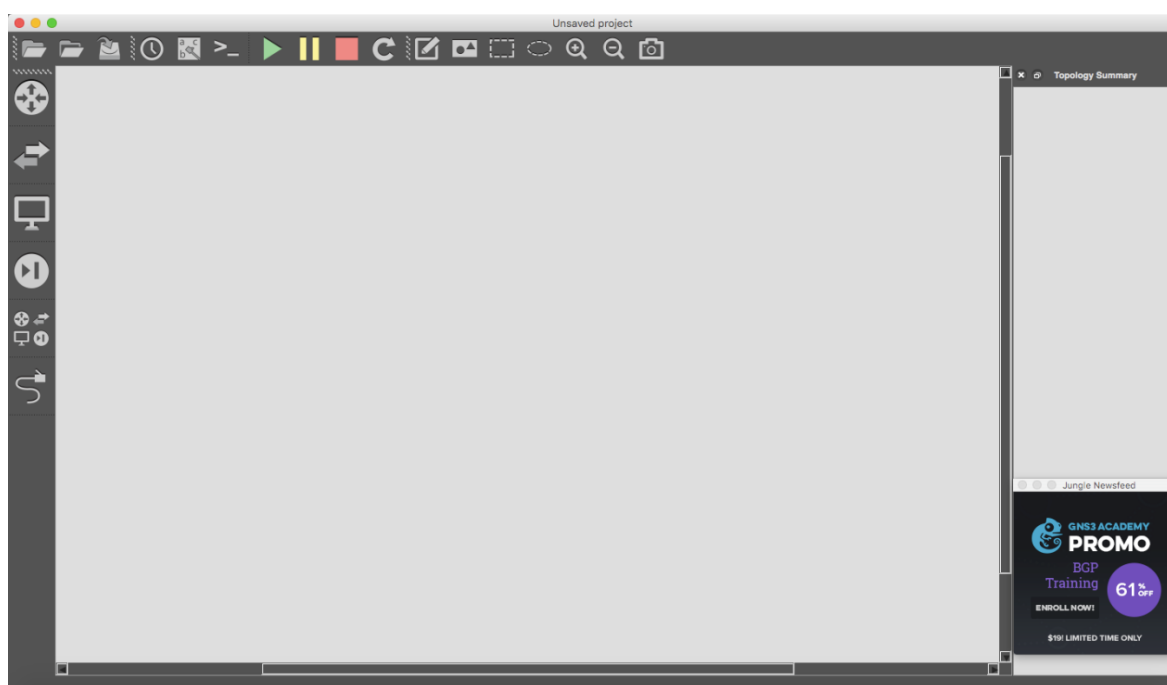


FIGURA 12 - Interface gráfica, front-end do simulador GNS3.

Fonte: Autoria própria.

Devido a possibilidade de simular roteadores e a execução de firmwares a gama de opções para simulações de topologia de redes e protocolos proprietário e também livres é ampla, o sistema é descomplicado tratando-se de uma interface simples e funcional, uma opção para simulações de redes onde não exista a necessidade de verificação de dados posteriores. O GNS3 não possibilita a criação de um arquivo de saída, algum tipo de dados de saída que possam ser utilizados para algum tipo de teste e verificação. Devido a ausência da opção de gerar dados

de saída a utilização do *software* GNS3 foi descartada para o desenvolvimento deste trabalho.

O segundo *software* analisado foi o OPNET, um simulador comercial, largamente utilizado em âmbito corporativo, devido a sua precisão nos resultados e também pela utilização fácil e descomplicada. Focada em uma interface gráfica simples que facilita o entendimento em tarefas de maior complexidade como, por exemplo, a configuração de cenários de rede e a visualização de resultados.

O *software* é de licença proprietária mas conta também com uma versão acadêmica a fins educacionais.

A FIGURA 13 exibe a interface do *software* OPNET, através desta interface são realizadas as configurações de simulação.

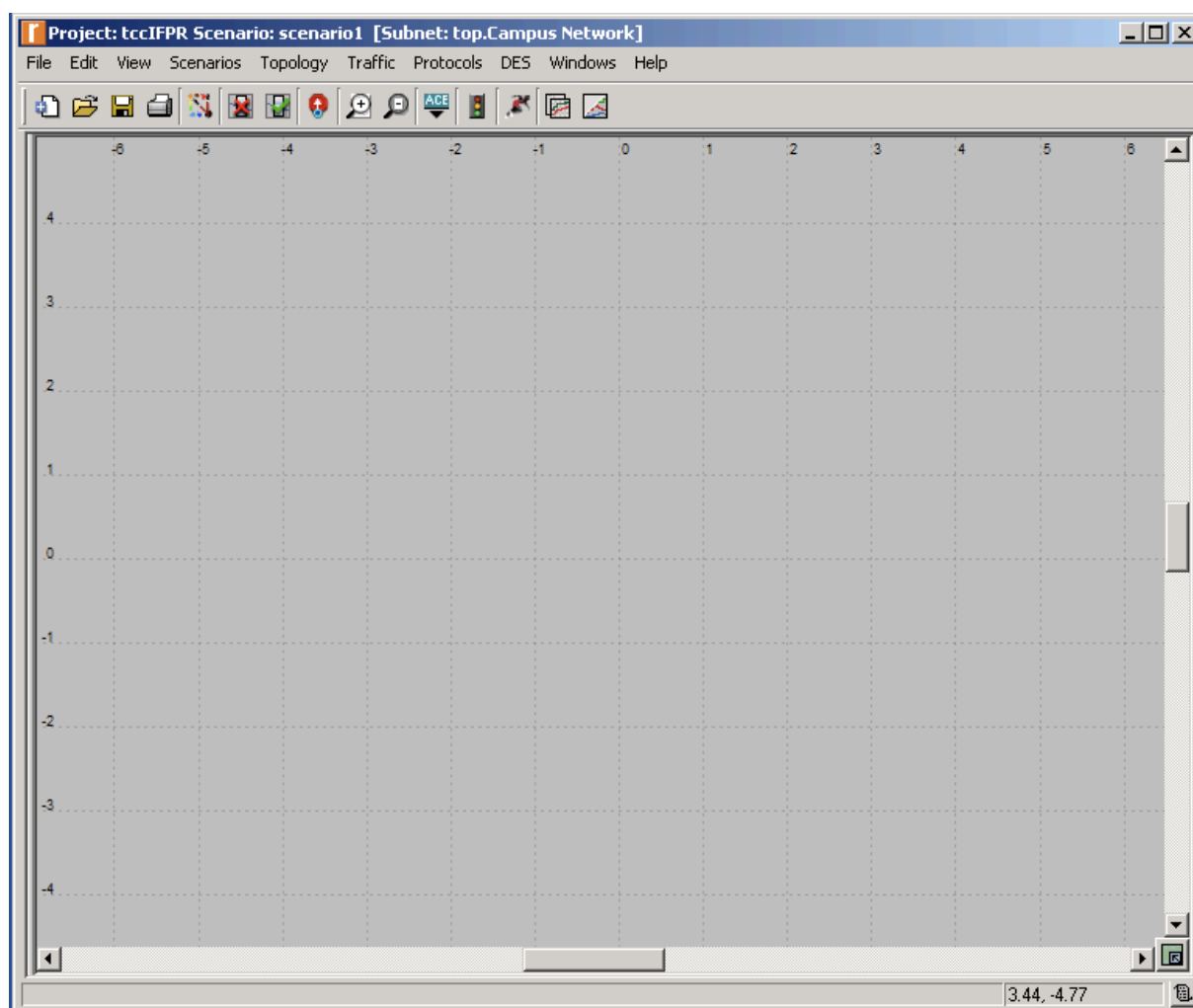


FIGURA 13 - Interface *front-end* do *software* OPNET versão acadêmica.

Fonte: Autoria própria.

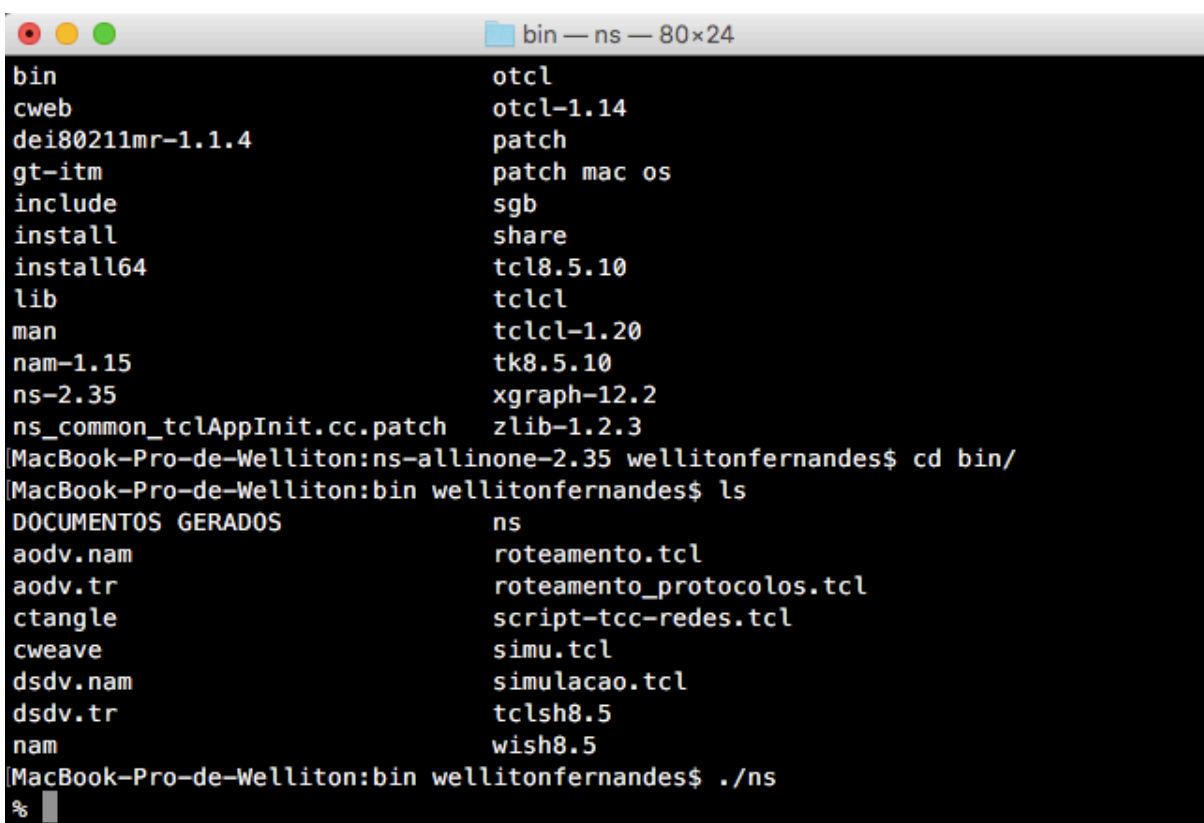
O OPNET conta com a possibilidade de geração de um arquivo de saída o que possibilita uma possível integração a uma outra ferramenta, mas não atende aos requisitos desejáveis para este trabalho por se tratar de um *software* proprietário e não multiplataforma. A versão mais completa do simulador OPNET é paga e também não atende aos critérios estabelecidos de escolha para este trabalho.

O *software Network Simulator 2*, (NS2), foi analisado com base nos critérios estabelecidos os quais todos foram atendidos. O *software* é largamente utilizado, contendo trabalhos de pesquisa e desenvolvimento já realizados com a utilização dessa ferramenta, estes dados se tornaram pontos positivos que definiram a escolha pelo *software* NS2.

O NS2 trata-se de um *software open source* de código aberto, livre para modificações e dá suporte a múltiplas plataformas como Unix e sistemas baseados em Unix, FreeBSD, Solaris e Windows.

O *Network Simulator 2* é um simulador de eventos discreto que tem como objetivo a criação de simulações baseadas nos protocolos de rede, foi construído através da utilização de duas linguagens, C++ em sua estrutura básica e OTcl para uso como *frontend*. O projeto no qual se deu origem ao *software* começou no ano de 1989 e surgiu inicialmente de um projeto concebido com título VINT (*Virtual InterNetwork Testbed*), seu código fonte é aberto e oferece suporte a um número significativo de tecnologias de rede, sua aplicação pode ser realizada em diferentes cenários baseados nos protocolos TCP e UDP, dentre outros compõem esse projeto a DARPA, USC/ISI, Xerox PARC, LBNL e a Universidade de Berkeley. A linguagem OTcl é interpretada, desenvolvida pelo MIT, é com esta linguagem que as simulações são escritas, o simulador de redes NS2 é executado via terminal e não faz o uso de interface gráfica para configuração ou execução dos cenários a serem simulados.

A FIGURA 14 demonstra a execução do *software* NS2 que acontece via terminal de um sistema operacional Mac Os X o qual foi utilizado no desenvolvimento deste trabalho.



```

bin
cweb
dei80211mr-1.1.4
gt-itm
include
install
install64
lib
man
nam-1.15
ns-2.35
ns_common_tclAppInit.cc.patch
otcl
otcl-1.14
patch
patch mac os
sgb
share
tcl8.5.10
tclcl
tclcl-1.20
tk8.5.10
xgraph-12.2
zlib-1.2.3
MacBook-Pro-de-Welliton:ns-allinone-2.35 wellitonfernandes$ cd bin/
MacBook-Pro-de-Welliton:bin wellitonfernandes$ ls
DOCUMENTOS GERADOS      ns
aodv.nam                 roteamento.tcl
aodv.tr                  roteamento_protocolos.tcl
ctangle                  script-tcc-redes.tcl
cweave                   simu.tcl
dsdv.nam                 simulacao.tcl
dsdv.tr                  tclsh8.5
nam                       wish8.5
MacBook-Pro-de-Welliton:bin wellitonfernandes$ ./ns
%

```

FIGURA 14 - Network Simulator 2 em execução via terminal.

Fonte: Autoria própria.

O NS2 possibilita ao usuário um maior controle do projeto a se desenvolver, pois toda sua estrutura é documentada e de codificação aberta, tornando possível a criação de *frameworks*, ferramentas que façam integração e complementem o sistema quando necessário, conforme a necessidade do projeto no qual irá ser aplicado.

A noção de tempo no NS2 é obtida através de unidades de simulação que podem ser associadas, para efeitos didáticos, a segundos. A rede é construída por nós nos quais se conectam através de enlaces, os eventos ocorridos são escalonados para que possam passar entre os nós através dos enlaces estabelecidos, os nós e enlaces podem ter propriedades associadas, agentes podem ser associados aos nós e estes são responsáveis pela geração de diferentes pacotes.

A fonte de tráfego, a criadora dos dados a serem enviados pelo emissor, é uma aplicação a qual é associado um agente particular, os agentes necessitam de

um receptor que receberá seus pacotes possibilitando a comunicação entre os nós, um agente TCP por exemplo, orientado a conexão, o receptor chama-se SINK e tem como tarefa a geração de pacotes de reconhecimento chamados de ACK – *Acknowledge*, no caso de uma agente UDP, não orientado a conexão, o receptor chama-se NULL.

A FIGURA 15 apresenta a estrutura dos objetos da simulação, na qual fazem parte a aplicação, os agentes, nós envolvidos e o link, enlace entre os nós estabelecidos.

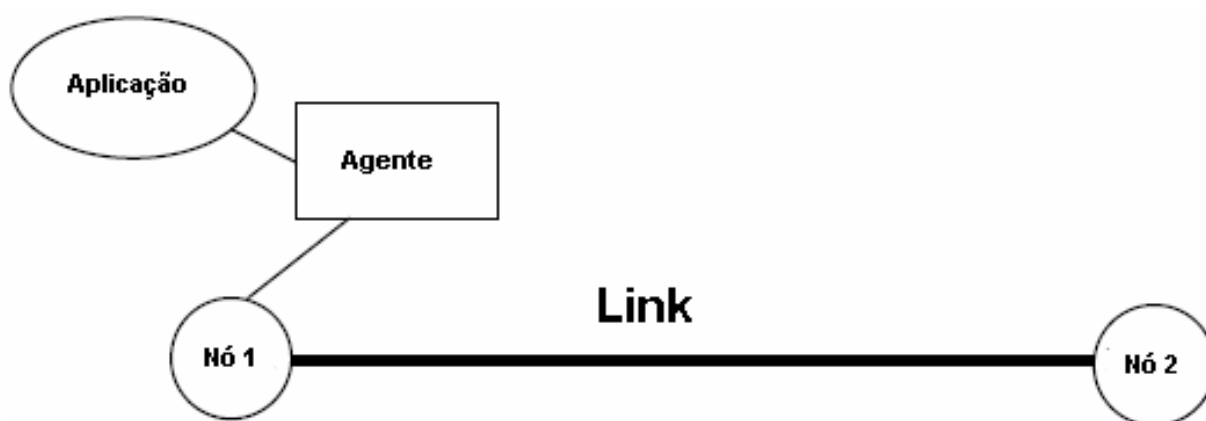


FIGURA 15 - Estrutura dos objetos da simulação.

Fonte: Tutorial NS2.

A execução de uma simulação possibilita a geração de um arquivo de saída em formato texto, chamado *trace* que contém os dados capturados pela simulação de rede executada pelo NS2. Este arquivo pode ser utilizado em outros *softwares* como dados de entrada para um detalhamento da simulação ou leitura para representação gráfica da simulação realizada. O processo de criação de uma simulação utilizando-se do NS2 é uma sequência bem definida de processos escrita em um *script* de formato OTcl, este arquivo em si é a simulação, arquitetura, na qual o *software* irá rodar e gerar seus respectivos dados, toda configuração de rede desejada a simular deve ser descrita neste *script*.

No *script* OTcl a sequência de processos definidos são a criação de um objeto simulador (escalonador de eventos), em seguida define-se a escrita para a

abertura de arquivos para análise posterior chamados de arquivos *trace*, então inicia-se a criação da topologia, arquitetura, dos nós e seus enlaces, definindo o tipo de ligação que estes nós terão entre eles, a criação dos agentes da camada de transporte e conexão com os nós, após estas configurações é necessário estruturar a topologia da rede, suas ligações e agentes, definido os geradores de tráfego e conexão com os agentes da camada de transporte, a partir daí é realizada a programação dos eventos da simulação e finalizando com o fechamento da simulação e, a possível geração de estatísticas. A sintaxe do *script* OTcl para a geração de simulações no NS2 é específica do *software* e requer conhecimento desta linguagem.

O NS2 conta com ferramentas de auxílio para análise dos dados extraídos com por exemplo a ferramenta Nam, na qual é responsável por fazer a leitura do arquivo Nam-trace e representar a topologia e comportamento de tráfego graficamente, sendo possível compreender melhor os nós, seus enlaces e como os dados estão sendo manipulados, possibilitando a identificação de gargalos e perda de dados na rede.

A ferramenta Nam é muito importante para uma análise detalhada e de fácil compreensão pois possibilita a visão geral de toda a estrutura planejada, visualizando o andamento da simulação, identificando além dos fluxos a formação de filas e o descarte de pacotes.

A FIGURA 16 exibe a ferramenta Nam em execução, utilizando os resultados obtidos de uma simulação do NS2 e exibindo os nós e fluxos entre eles.

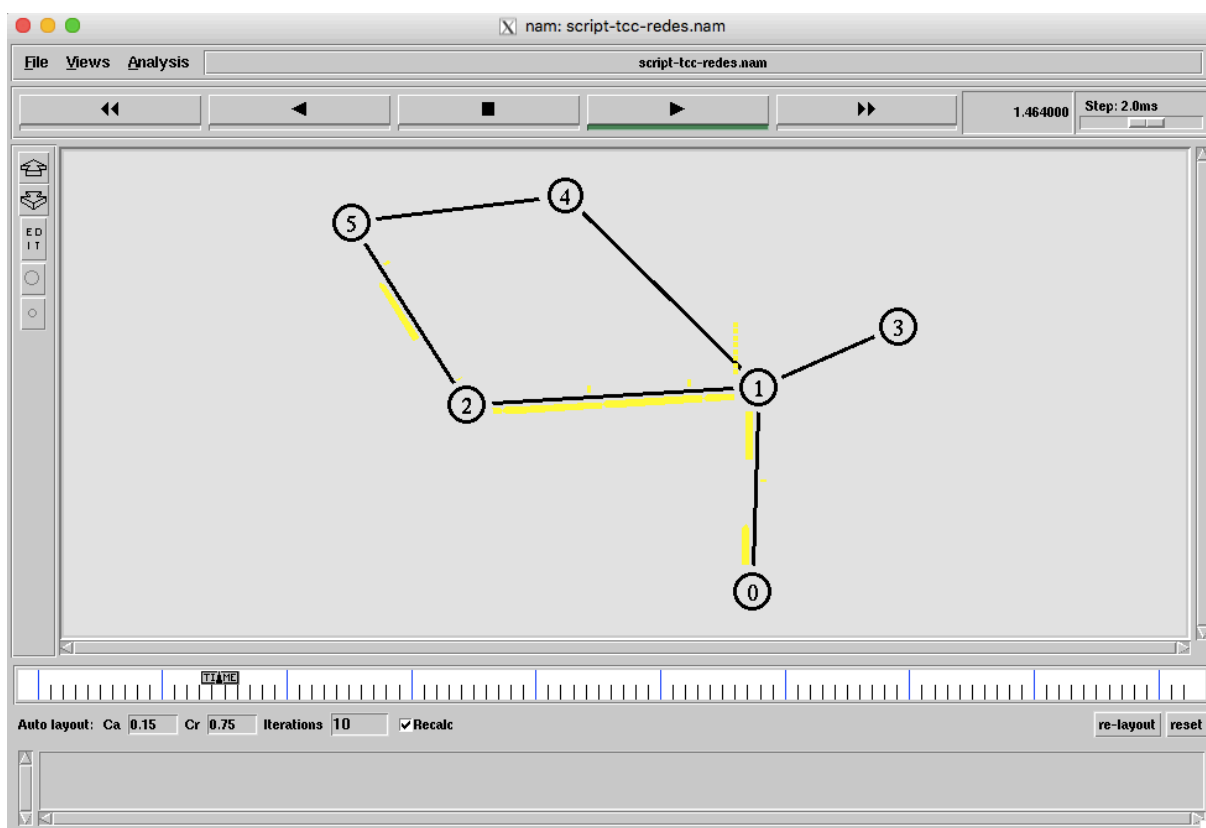


FIGURA 16 - Execução do arquivo Nam-Trace pela ferramenta Nam.

Fonte: Autoria própria.

Há também a ferramenta gráfica Xgraph e Gnuplot nas quais, assim como a ferramenta Nam são voltadas para análise gráfica do arquivo *trace*. Estas ferramentas possibilitam a geração de gráficos para verificação de vazão de cada um dos fluxos utilizados na simulação, contribuem para análise das estatísticas do NS2, contam com uma opção de plotagem de gráficos e visualização interativa para gerar a animação. Trabalham com arquivos colonados, utilizando-se da estrutura de linhas e colunas x e y.

O Gnuplot é uma ferramenta de *software* de domínio público destinado a construção de gráficos e superfícies, tem como uma de suas características fortes a possibilidade de execução de um arquivo *script* em diferentes sistemas operacionais.

Diferentes protocolos podem ser implementados nas simulações utilizando o NS2, como protocolos da camada de rede que nos permite definir regras de roteamento, toda transferência de pacotes entre um nó emissor e um nó receptor faz

uso da definição de qual o caminho deverá ser percorrido para que exista a comunicação entre os nós, o núcleo do protocolo de roteamento é o algoritmo de roteamento, este tem a função de determinar a rota adequada entre a origem e o destino dos datagramas. Diversos protocolos de roteamento podem ser implementados utilizando a simulação pelo NS2, como por exemplo o roteamento unicast, estático, dinâmico e vetor de distância.

Assim como apresentado pela pesquisa realizada por Blome et al (2013), existem diversos *softwares* que são utilizados para testes de penetração e testes de vulnerabilidades, mas ainda assim é possível se deparar com situações problemáticas nas quais o *software* para a realização destes testes não podem ajudar pois não contam com uma detecção de erros e validação de regras específicas.

Realizado um levantamento bibliográfico foi analisada e comprovada a viabilidade de utilização de modelos comportamentais em testes aplicados a redes de computadores. A primeira etapa do trabalho consistiu em realizar uma análise de três *software* conhecidos na área de simulação de redes. A análise entre os *softwares* consistiu-se em avaliar quais dos simuladores melhor se adequaria ao desenvolvimento deste trabalho.

As funcionalidades do NS2 foram analisadas, suas possibilidades, tipos de simulações de redes, os protocolos suportados, assim como o comportamento da ferramenta em diferentes plataformas. O NS2 foi testado em três sistemas operacionais distintos o Windows XP, Linux na distribuição Ubuntu e Mac Os X.

No desenvolvimento deste trabalho foi utilizado o sistema operacional Os X.

4.1.1 Dados da Simulação de Rede

As simulações a serem executadas pelo *software* NS2 são escritas na linguagem OTcl, em um arquivo com extensão Tcl, este contém todas as definições de conexões, tipos de protocolos, ligações entre dispositivos, tipo da rede e suas políticas de comunicação e transferência de dados.

Os dados de simulação gerados pelo NS2 são salvos em arquivos do tipo

texto e sua extensão é do tipo *tr*, o arquivo *trace* é formado por 12 colunas e uma sequência de tuplas referentes aos eventos ocorridos na simulação de forma ordenada. A quantidade de tuplas correspondentes aos eventos e suas características podem variar de acordo com as definições contidas no arquivo *tcl*, sendo que quanto mais extenso o projeto a ser simulado, a quantidade de nós e seus enlaces definidos no *script* de simulação, maior a quantidade de informações na qual serão armazenadas no arquivo *trace*, conseqüentemente será maior o número de tuplas no arquivo especificando cada evento ocorrido. Dessa forma um arquivo do tipo *trace* pode chegar a vários megabytes de tamanho.

A Figura 17 apresenta um exemplo de um arquivo *trace* gerado por uma simulação do NS2.

```

1  + 0.5 0 1 tcp 40 ----- 1 0.0 5.0 0 0
2  - 0.5 0 1 tcp 40 ----- 1 0.0 5.0 0 0
3  r 0.51016 0 1 tcp 40 ----- 1 0.0 5.0 0 0
4  + 0.51016 1 2 tcp 40 ----- 1 0.0 5.0 0 0
5  - 0.51016 1 2 tcp 40 ----- 1 0.0 5.0 0 0
6  r 0.53048 1 2 tcp 40 ----- 1 0.0 5.0 0 0
7  + 0.53048 2 5 tcp 40 ----- 1 0.0 5.0 0 0
8  - 0.53048 2 5 tcp 40 ----- 1 0.0 5.0 0 0
9  r 0.54064 2 5 tcp 40 ----- 1 0.0 5.0 0 0
10 + 0.54064 5 2 ack 40 ----- 1 5.0 0.0 0 1
11 - 0.54064 5 2 ack 40 ----- 1 5.0 0.0 0 1
12 r 0.5508 5 2 ack 40 ----- 1 5.0 0.0 0 1
13 + 0.5508 2 1 ack 40 ----- 1 5.0 0.0 0 1
14 - 0.5508 2 1 ack 40 ----- 1 5.0 0.0 0 1
15 r 0.57112 2 1 ack 40 ----- 1 5.0 0.0 0 1
16 + 0.57112 1 0 ack 40 ----- 1 5.0 0.0 0 1
17 - 0.57112 1 0 ack 40 ----- 1 5.0 0.0 0 1
18 r 0.58128 1 0 ack 40 ----- 1 5.0 0.0 0 1
19 + 0.58128 0 1 tcp 1040 ----- 1 0.0 5.0 1 2
20 - 0.58128 0 1 tcp 1040 ----- 1 0.0 5.0 1 2
21 + 0.58128 0 1 tcp 1040 ----- 1 0.0 5.0 2 3
22 - 0.58544 0 1 tcp 1040 ----- 1 0.0 5.0 2 3
23 r 0.59544 0 1 tcp 1040 ----- 1 0.0 5.0 1 2
24 + 0.59544 1 2 tcp 1040 ----- 1 0.0 5.0 1 2

```

Line 18018, Column 48 Tab Size: 4 Plain Text

FIGURA 17 - Arquivo *trace* gerado pelo NS2.

Fonte: Autoria própria.

A TABELA 1 apresenta um exemplo da estrutura e organização dos dados do arquivo *trace*.

Evento	Tempo	Nó Orig.	Nó Dest.	Tipo Pacote	Tam. Pacote	Flags	Id Fluxo	End. Fonte	End. Destino	Num Seq.	Id Pacote
r	1.3556	3	2	ack	40	----	1	3.0	0.0	15	201
+	1.3556	2	0	ack	40	----	1	3.0	0.0	15	201
-	1.3556	2	0	ack	40	----	1	3.0	0.0	15	201
r	1.35576	0	2	tcp	1000	----	1	0.0	0.0	29	199
+	1.35576	2	3	tcp	1000	----	1	0.0	3.0	29	199
d	1.35676	2	3	tcp	1000	----	1	0.0	3.0	29	199
+	1.356	1	2	cbr	1000	----	2	1.0	3.0	157	207
-	1.356	1	2	cbr	1000	----	2	1.0	3.1	157	207

TABELA 1 - Tabela com as informações do *trace* gerado pelo *Network Simulator 2*.

Fonte: Autoria própria

Para que seja possível a análise das simulações após a realização da fase de simulação, execução do *script*, é de extrema importância a compreensão do arquivo de saída, seus dados e como estes estão organizados no arquivo *trace* gerado pelo *Network Simulator 2*.

A primeira coluna da tabela do *trace* diz respeito ao evento ocorrido, nesta coluna é possível ter como representação dos eventos a letra (r) representando o evento de pacote recebido, o símbolo de adição (+) o evento de entrada de pacote, o símbolo de subtração (-) é a saída do pacote e a letra (d) na qual tem como objetivo representar um pacote descartado. A segunda coluna diz respeito ao evento de tempo, informando em qual tempo o tipo de evento descrito na primeira coluna aconteceu, este tempo é representado em segundos. A terceira coluna é referente a descrição do nó origem, o nó no qual enviou um determinado pacote na rede, e na sequência na quarta coluna o nó de destino, o qual receberá o pacote enviado pelo nó origem. A quinta coluna da tabela destina-se ao tipo de pacote a ser transmitido, a sexta coluna da tabela especifica o tamanho do pacote que está sendo enviado, na sétima coluna são especificadas as notificações antecipadas de congestionamento do enlace entre os nós, estas notificações são chamadas de

flags. A oitava coluna é responsável por registrar a identificação do fluxo, e em seguida, na nona coluna está contido o endereço do transmissor ou também chamado de endereço fonte, seguido da décima coluna o endereço de destinatário, décima primeira coluna o número de sequência do pacote, para referência da ordem dos pacotes recebidos no evento. Por fim, a décima segunda e última coluna da tabela, contém os dados de id do pacote, este tem a função de identificar de forma única o pacote na rede.

Apenas com a utilização do NS2 não é possível analisar o arquivo *trace*, dificultando o entendimento das ocorrências da simulação. Percebe-se uma frustração do usuário final ao realizar uma simulação e receber como resposta um arquivo de milhares de linhas contendo diversos dados que só fazem sentido no NS2. O arquivo de *trace* mesmo em simulações pequenas costuma ter dezenas de *kilobytes*, podendo chegar a *gigabytes* em simulações de grande porte (CAMPESTRINI, 2005).

Para a realização deste trabalho somente a quinta coluna foi utilizada, na qual a informação contida nesta coluna é referente ao protocolo ocorrido na simulação.

4.2 CONFIGURAÇÃO DE SIMULAÇÃO

Utilizando-se da linguagem OTcl foram desenvolvidos *scripts* contendo as configurações desejadas para as simulações de rede, que por meio do simulador NS2 foram executados e gerados arquivos de *trace* da simulação.


```

1  ##Welliton Fernandes Leal
2  ##Instituto Federal do Paraná
3  ##SCRIPT TCL - SIMULAÇÃO DE REDE
4
5  ##Variavel que indica o inicio da simulacao
6  ##Escalonador de eventos
7  set ns [new Simulator]
8
9  ##Define a cor do fluxo de dados que sera exibido
10 ##pela ferramenta grafica NAM
11 $ns color 1 Yellow
12
13 ##Procedure que gera o Nam-Trace para simulacao
14 ##grafica.
15 set nf [open tcc-ifpr.nam w]
16 $ns namtrace-all $nf
17
18 ##Procedure que gera o Trace da execucao da simulacao
19 set nf2 [open tcc-ifpr.tr w]
20 $ns trace-all $nf2
21
22 set f0 [open tcc-ifpr.tr w]
23
24 ## Fechamento da simulacao, animacao (nam)
25 ## e geracao de estatisticas
26 proc finish {} {
27     global ns nf
28     $ns flush-trace
29     close $nf
30     exec nam tcc-ifpr &
31     exec xgraph tcc-ifpr-bw.tr &
32     exit 0
33 }
34 proc record {} {
35     global sink f0
36     set ns [Simulator instance]
37     set time 0.100
38     set bw0 [$sink set bytes_]
39     set now [$ns now]
40     puts $f0 "$now [expr $bw0/$time*8/1000000]"
41     $sink set bytes_ 0
42     $ns at [expr $now+$time] "record"
43 }
44
45 ## Definicao dos nos da rede
46 set n0 [$ns node]
47 set n1 [$ns node]
48 set n2 [$ns node]
49 set n3 [$ns node]
50 set n4 [$ns node]

```

FIGURA 18 - Script OTcl contendo as configurações de rede para a simulação pelo software *Network Simulator 2*.

Fonte: Autoria própria.

A FIGURA 18 apresenta parte de uma configuração de rede na linguagem OTcl para a execução através do NS2. Todas as configurações desejáveis a serem simuladas são definidas neste arquivo de extensão Tcl.

Após a realização dos *scripts*, seguido da execução das simulações, foi realizada a verificação do arquivo *trace* com fim de identificar como é salvo internamente, como os dados estão estruturados e o significado de cada um deles. Foi identificado o padrão de linhas e colunas na estrutura do *trace*, onde cada coluna é correspondente a um tipo de dado específico da simulação, e cada tupla, contém todos os dados correspondentes a um determinado momento da simulação, assim

como mostrado no capítulo 4.1.1. Entretanto, este arquivo de *trace* não possui nenhuma informação tratada, somente os dados relacionados aos eventos ocorridos.

Com a identificação da estrutura do arquivo *trace* foi definido o escopo para o desenvolvimento do trabalho proposto. Neste trabalho somente dados da quinta coluna do *trace*, referente aos protocolos ocorridos na simulação, foram utilizados para testes de comparação através da ferramenta desenvolvida. Esta ferramenta, proposta neste trabalho de pesquisa, é descrita a seguir.

4.3 DESENVOLVIMENTO DA FERRAMENTA

Neste trabalho desenvolveu-se uma ferramenta que realiza uma integração ao simulador NS2 denominada de *Modeling System Routing Protocol* (MSRP). A codificação do *software* foi feita utilizando-se do paradigma de orientação a objetos com uso da linguagem Java. O ambiente de desenvolvimento utilizado para a codificação do *software* foi o Eclipse Luna.

O processo de desenvolvimento foi direcionado pelo uso da metodologia de desenvolvimento de Prototipação, esta metodologia conta com a capacidade de responder com mais eficiência a mudanças e alterações em *software* durante o desenvolvimento, permitindo uma análise constante do sistema que está sendo desenvolvido, adaptando, gerando uma evolução do sistema até atingir os objetivos definidos. Um esboço do sistema foi construído para estabelecer os diagramas e a implementação.

Por meio de diagrama de casos de uso foi descrito o sistema MSRP com as principais funcionalidades acessíveis ao usuário. A FIGURA 19 descreve as funcionalidades do sistema utilizando o diagrama de casos de uso.

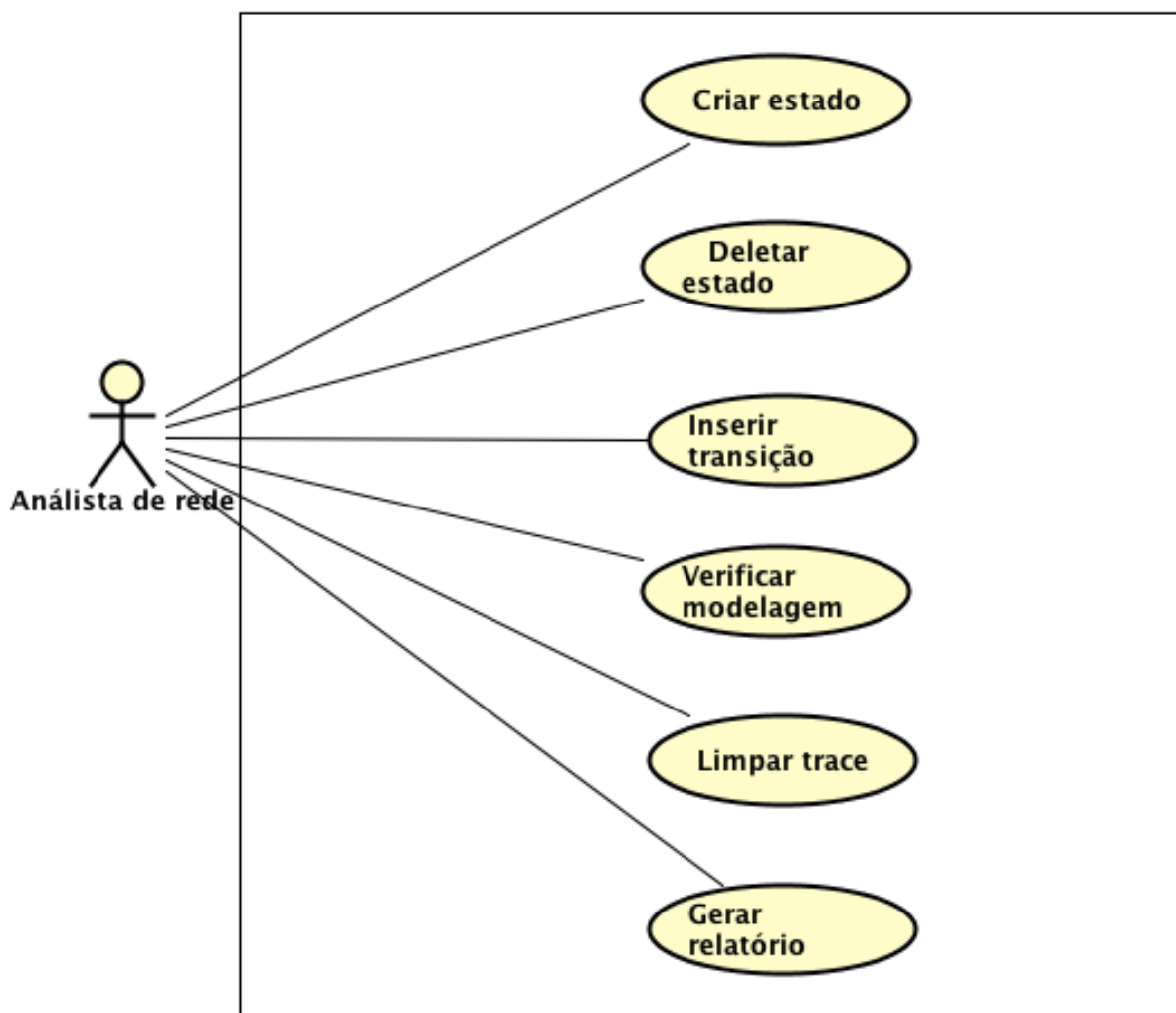


FIGURA 19 - Diagrama de casos de uso da ferramenta MSRP.

Fonte: Autoria própria.

Com base no diagrama de casos de uso foi definida a estrutura do código utilizando do padrão MVC, estruturando os pacotes e definindo as classes que foram utilizadas através da criação de um diagrama de classes. Com a definição do diagrama de classes foi possível detalhar a codificação de todo o sistema de forma simples, a relação entre as classes e seus atributos e métodos.

A FIGURA 20 retrata o sistema MSRP pelo diagrama de classes.

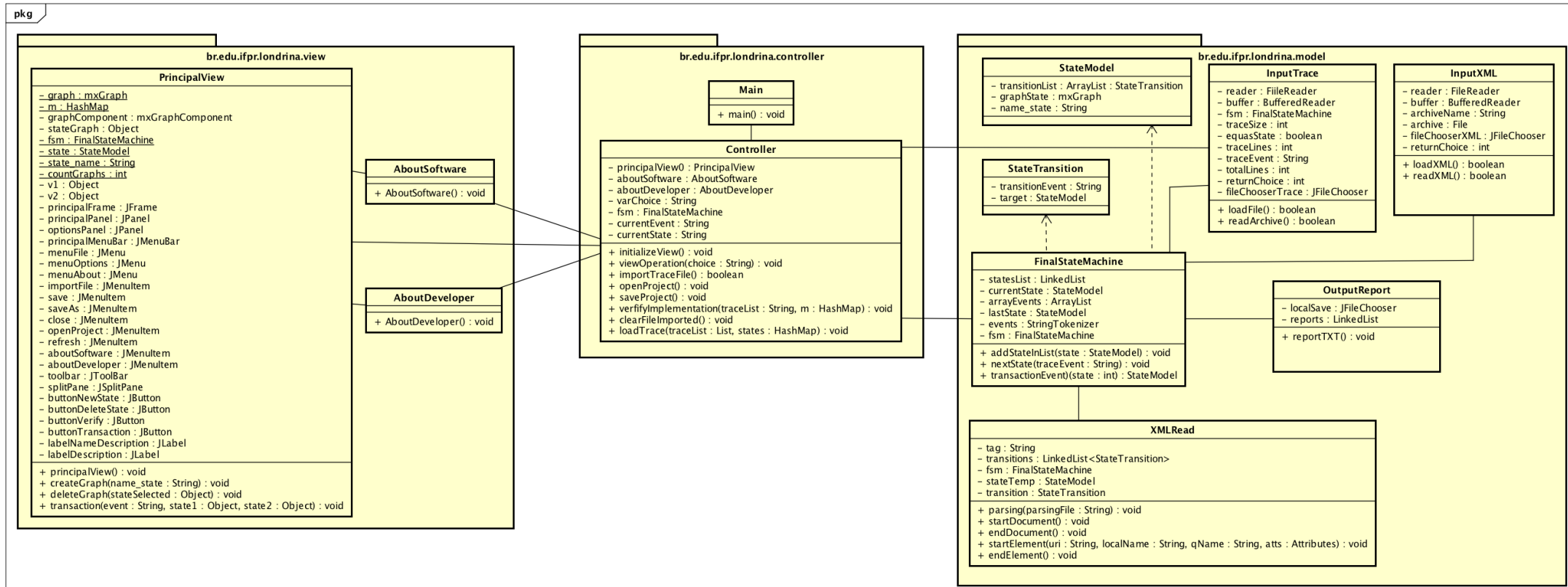


FIGURA 20 - Diagrama de classes da ferramenta MSRP.

Fonte: Autoria própria.

A classe Main faz parte do pacote `br.edu.ifpr.controller`, esta tem como única função realizar a primeira chamada a classe Controller.

A classe Controller realiza a chamada de inicialização da tela principal, contem também outros métodos como por exemplo para realizar a abertura do arquivo XML, outro responsável pela importação do arquivo *trace* e também o método responsável por realizar a verificação entre o *trace* e o *modelo* definido. A classe Controller faz a ligação entre as classes através da estrutura MVC.

No pacote `br.edu.ifpr.view` estão contidas as classes referentes as telas do sistema, fazem parte desde pacote as classes `PrincipalView`, na qual é responsável por apresentar a tela principal do sistema, a classe `AboutSoftware` responsável pela tela que exibe informações sobre a ferramenta e a classe `AboutDeveloper` que exibe informações sobre o desenvolvedor.

No pacote `br.edu.ifpr.model` estão contidas as classes para manipular as informações de forma mais detalhada, os modelos tem acesso a todas informações contidas no sistema. Três classes são tidas como principais para a criação de uma máquina de estados finitos, são elas: a classe `StateModel` é o modelo responsável pela estrutura dos estados gerados e seus atributos, a classe `StateTransition` que é responsável por gerar a estrutura de armazenamento das informações sobre as transições dos estados gerados e a classe `FinalStateMachine` que é a classe na qual trata os dados de toda a estrutura da máquina de estados finitos em memória, nela estão contidos objetos das classes `StateModel` e `StateTransition`.

Também estão presentes no pacote `br.edu.ifpr.model` a classe `InputTrace`, contém a estrutura para o carregamento do arquivo *trace* do NS2, a classe `InputXML`, estrutura responsável pelo carregamento do arquivo XML para a ferramenta, a classe `XMLRead` que faz a leitura do modelo XML carregado e trata os dados, por fim classe `OutputReport` é responsável por realizar a geração de um arquivo texto, relatório, e salvar os dados obtidos em formato texto de extensão `.txt`.

Após a definição da estrutura por meio de modelos da UML para o desenvolvimento da ferramenta, foi iniciada a construção do código. Além do padrão MVC, foi utilizado o padrão de projeto *Singleton*.

A FIGURA 21 demonstra a estrutura do padrão MVC, a divisão entre os pacotes e suas classes no ambiente de desenvolvimento Eclipse.

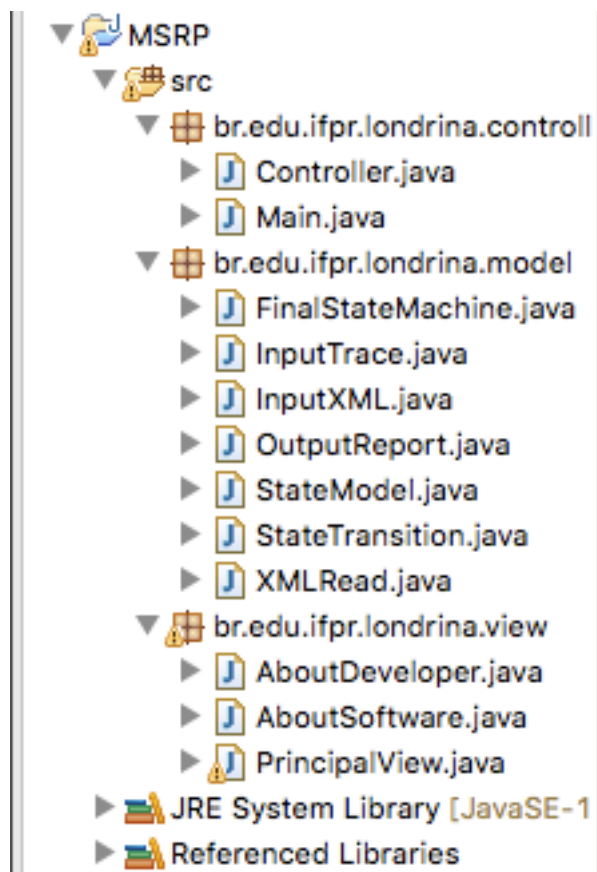


FIGURA 21 - Estrutura do código do projeto MSRP.

Fonte: Autoria própria.

Para o desenvolvimento gráfico da ferramenta, possibilitando a criação de um modelo de máquina de estados, foi utilizada a API JGraph, *framework* destinado a manipulação de grafos e diagramas. A escolha deste *framework* foi definida com base no estudo de análise e comparação. O *framework* JGraph tem como ponto forte a manipulação e criação de grafos visuais, esta API utiliza-se das bibliotecas *java.swing* e *java.awt* para a renderização dos elementos gráficos e de interação com o usuário (CASTRO, 2012).

O arquivo *trace* é utilizado como entrada para o MSRP, a verificação é realizada através da comparação do modelo comportamental esperado, o qual foi modelado na própria ferramenta ou definido e carregado por meio de um arquivo

XML, este então é comparado ao arquivo *trace*.

A FIGURA 22 apresenta a ferramenta MSRP desenvolvida. O menu encontra-se ao lado esquerdo da janela na ferramenta com as opções de criar um novo estado, deletar um estado selecionado, checar a modelagem com o *trace*, inserir as transições e eventos e gerar um relatório no formato txt.

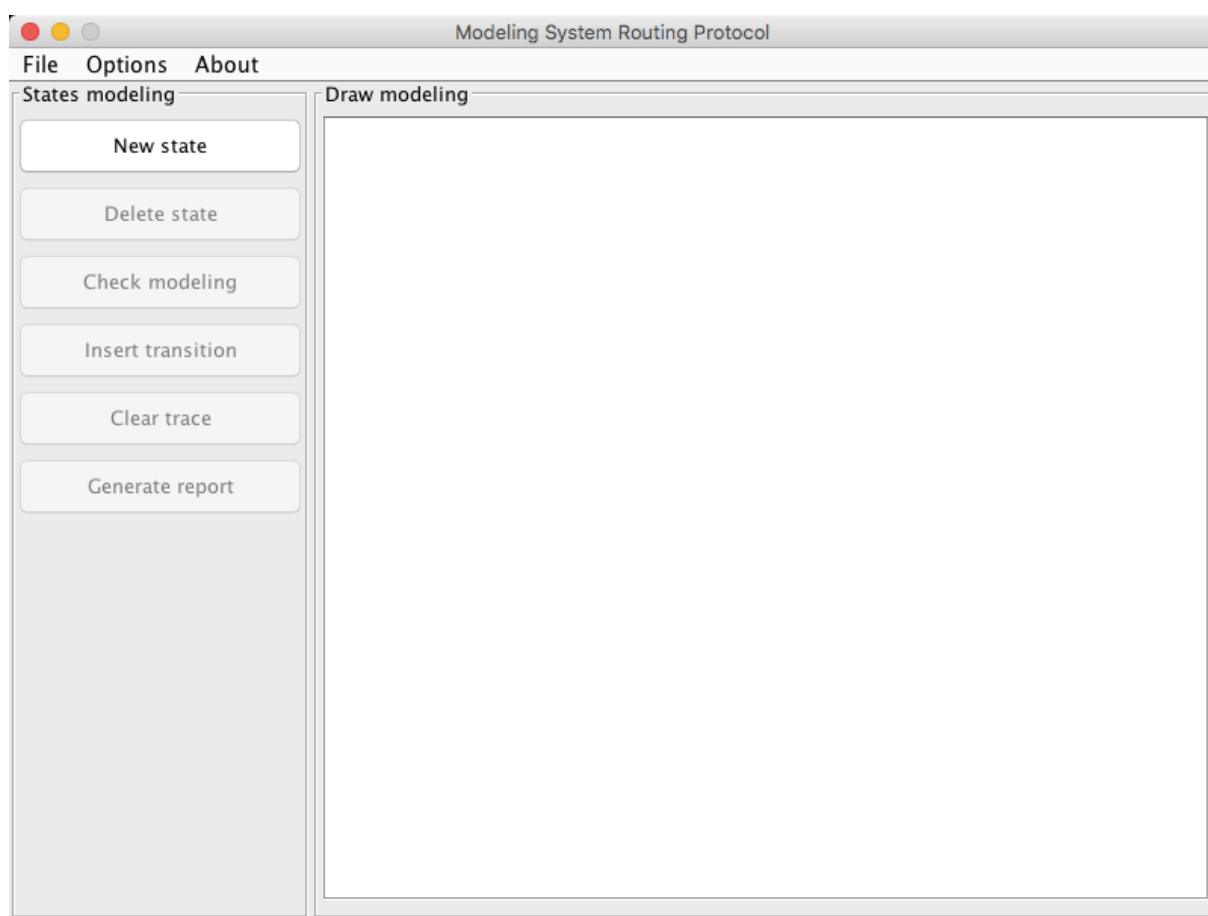


FIGURA 22 - Exibição da ferramenta MSRP desenvolvida.

Fonte: Autoria própria.

O MSRP permite importar o arquivo *trace* e percorre-lo através da opção *Checker modeling* realiza a verificação dos protocolos existentes comparando com a MEF definida a fim de verificar se o funcionamento dos protocolos de rede na simulação atende ao funcionamento esperado que foi expresso no modelo comportamental.

Após a verificação do *trace* a ferramenta exibe uma mensagem de acordo

com a verificação executada e, é possível gerar um relatório de estados alcançados em formato texto, de extensão txt, no qual exibe o estado atual, o número de transições do estado, o protocolo ocorrido e o próximo estado da MEF. Assim, é possível verificar além da existência do protocolo na simulação a ordem que os protocolos ocorreram e a quantidade de vezes que foram encontrados.

A FIGURA 23 exibe um modelo construído através da utilização da própria ferramenta MSRP.

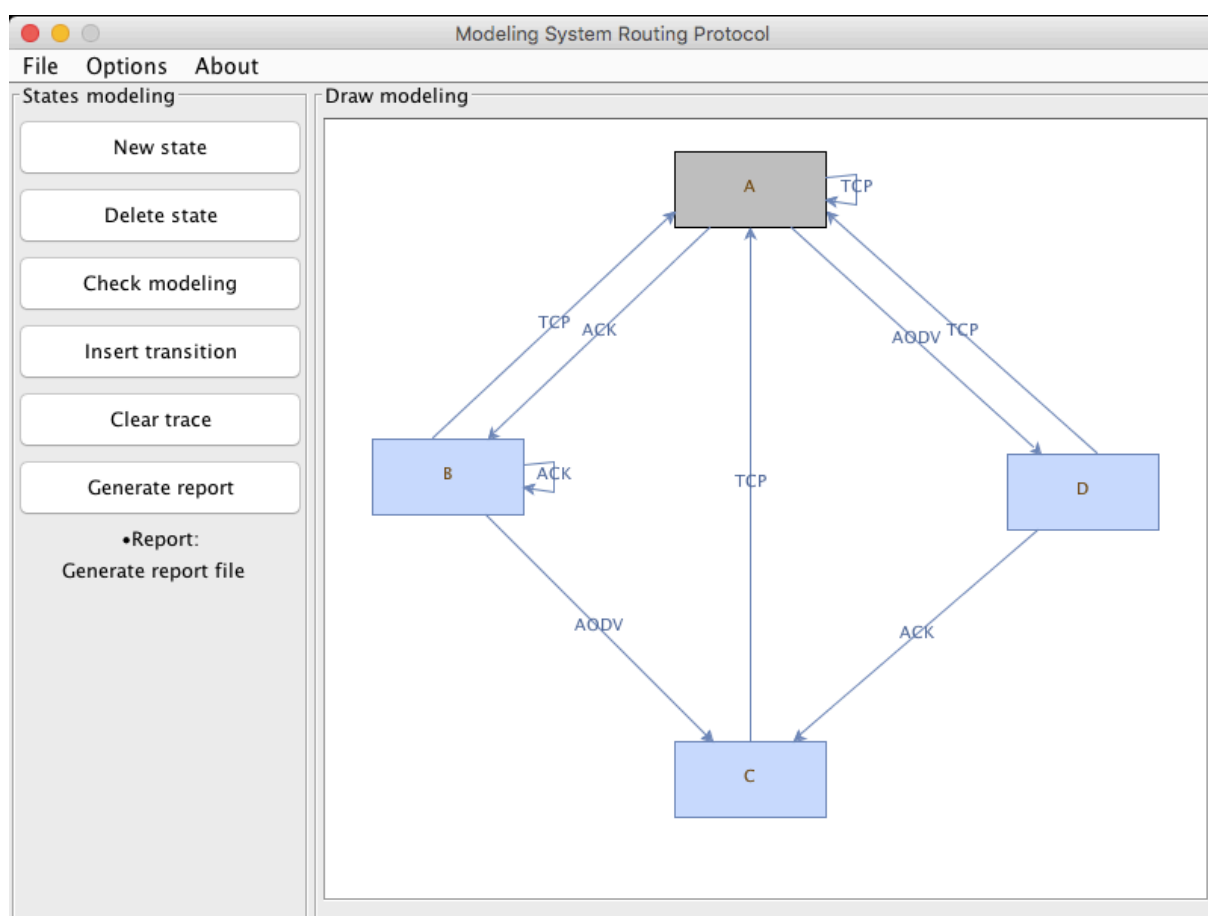


FIGURA 23 - Exemplo de um modelo para verificação de implementação de protocolo.

Fonte: Autoria própria.

A ferramenta MSRP atende os requisitos de ser multiplataforma podendo ser executada em diferentes sistemas operacionais, somente sendo necessário que o sistema operacional tenha suporte a uma máquina virtual Java.

4.4 EXEMPLO DE UTILIZAÇÃO DA FERRAMENTA MSRP

Com a utilização da ferramenta MSRP desenvolvida neste trabalho será demonstrado um exemplo de verificação de implementação do protocolo TCP através da modelagem de um modelo simples de máquina de estados e também por meio da utilização de um arquivo XML. O *trace* utilizado neste exemplo foi gerado pelo *script* descrito no capítulo 4.2 deste trabalho.

Com o *software* MSRP em execução é necessário que um modelo correspondente ao resultado esperado seja criado, utilizando a própria ferramenta através do painel *Draw Modeling* ou carregando o modelo através de um arquivo XML.

A FIGURA 24 a seguir retrata um modelo definido a partir da própria ferramenta. Este modelo tem o objetivo checar se o protocolo TCP, orientado a conexão, está sendo executado e em seus fluxos de dados estão corretos retornando pacotes de confirmação do tipo ACK.

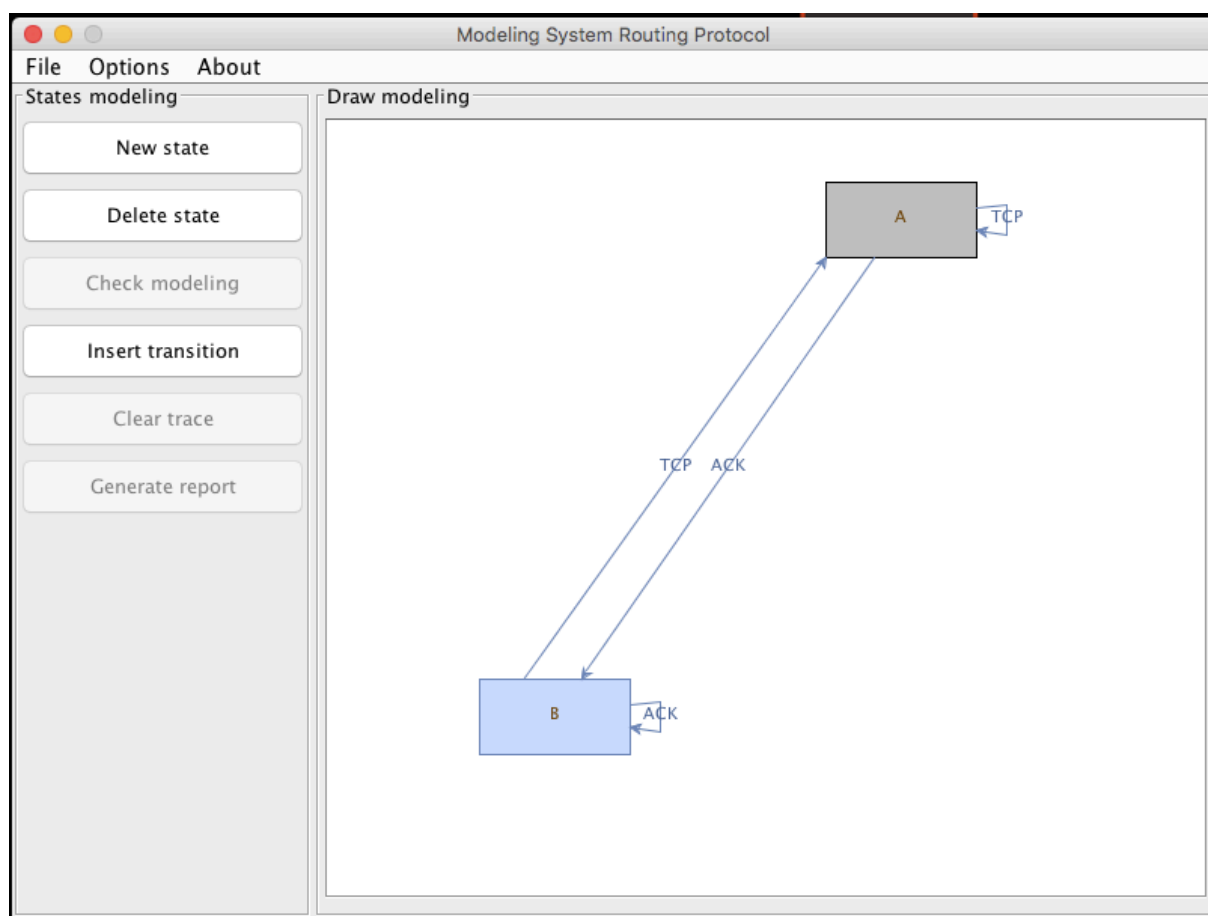


FIGURA 24 - Modelo simples para checar protocolo TCP e suas confirmações de fluxo.

Fonte: Autoria própria.

Após a criação do modelo é necessário importar o arquivo de *trace* para a ferramenta MSRP, no menu FILE na barra superior contem a opção IMPORT TRACE. Selecionada a opção de importação do arquivo uma tela de seleção será exibida.

A FIGURA 25 exhibe o menu FILE com a opção IMPORT TRACE.

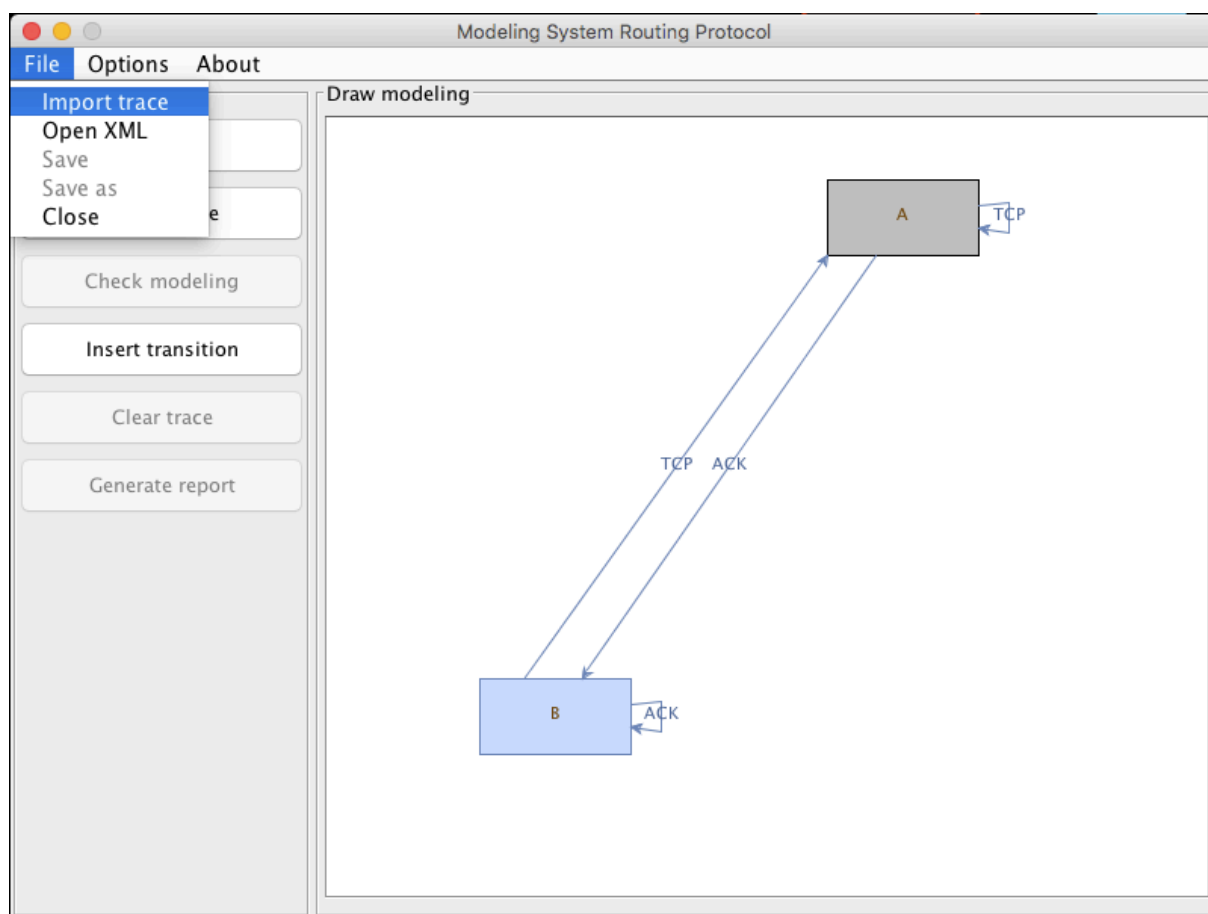


FIGURA 25 - Opção para importação do arquivo trace do NS2.

Fonte: Autoria própria.

Caso exista um arquivo com as definições do modelo em XML, não é necessário criar o modelo utilizando a ferramenta, basta selecionar a opção FILE no menu superior e em seguida a opção OPEN XML.

Os procedimentos seguintes são idênticos a importação do *trace*. A FIGURA 26 demonstra um arquivo XML correspondente a máquina de estados modelada pela ferramenta.

```

1  <!--
2  Arquivo XML - Sistema MSRP
3
4  Modelo de Máquina de Estados TCP
5  Utilizando dois estados (A,B)
6  -->
7  <fsm>
8      <state initial="true">A</state>
9      <state>B</state>
10     <transition>
11         <source>A</source>
12         <target>A</target>
13         <event>TCP</event>
14     </transition>
15     <transition>
16         <source>A</source>
17         <target>B</target>
18         <event>ACK</event>
19     </transition>
20     <transition>
21         <source>B</source>
22         <target>B</target>
23         <event>ACK</event>
24     </transition>
25     <transition>
26         <source>B</source>
27         <target>A</target>
28         <event>TCP</event>
29     </transition>
30 </fsm>

```

Line 2, Column 27 Spaces: 2 XML

FIGURA 26 - Modelo de máquina de estados definida em XML para utilização na ferramenta MSRP.

Fonte: Autoria própria.

A FIGURA 27 demonstra a exibição da tela de seleção após a opção IMPORT FILE ter sido selecionada para o carregamento do *trace*.

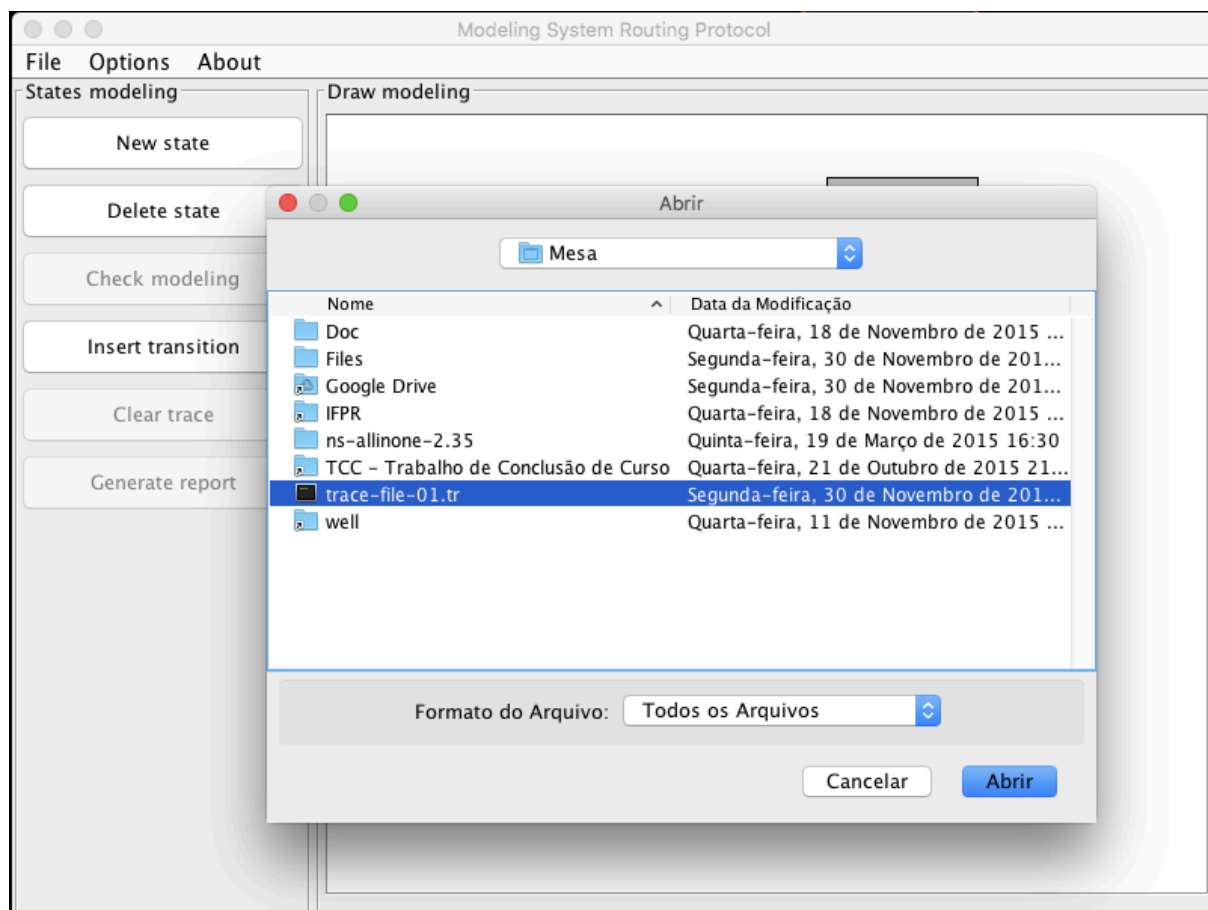


FIGURA 27 - Seleção do arquivo trace pela ferramenta MSRP.

Fonte: Autoria própria.

Após o arquivo *trace* ter sido selecionado e carregado pela ferramenta o seu conteúdo será analisado e caso esteja correto uma mensagem de confirmação será exibida. A FIGURA 28 demonstra a mensagem de confirmação do carregamento do arquivo bem-sucedido pelo MSRP.

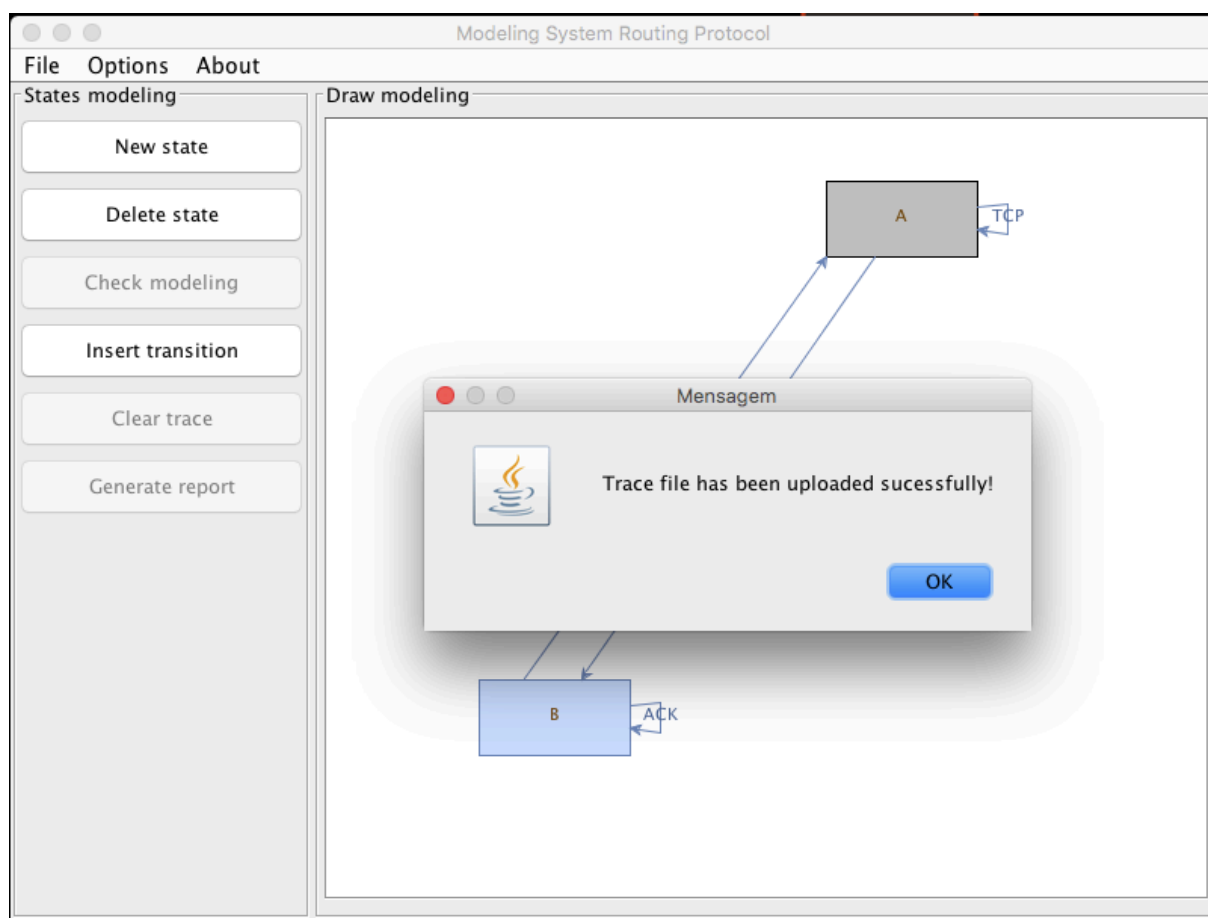


FIGURA 28 - Exibição de mensagem do arquivo trace carregado com êxito.

Fonte: Autoria própria.

Com o arquivo *trace* importado pela ferramenta, para realizar a verificação, comparação entre o *trace* e o modelo esperado, é necessário clicar em CHECK MODELING. Se o arquivo *trace* corresponder ao modelo uma mensagem de implementação correta será exibida, como exibida na FIGURA 29.

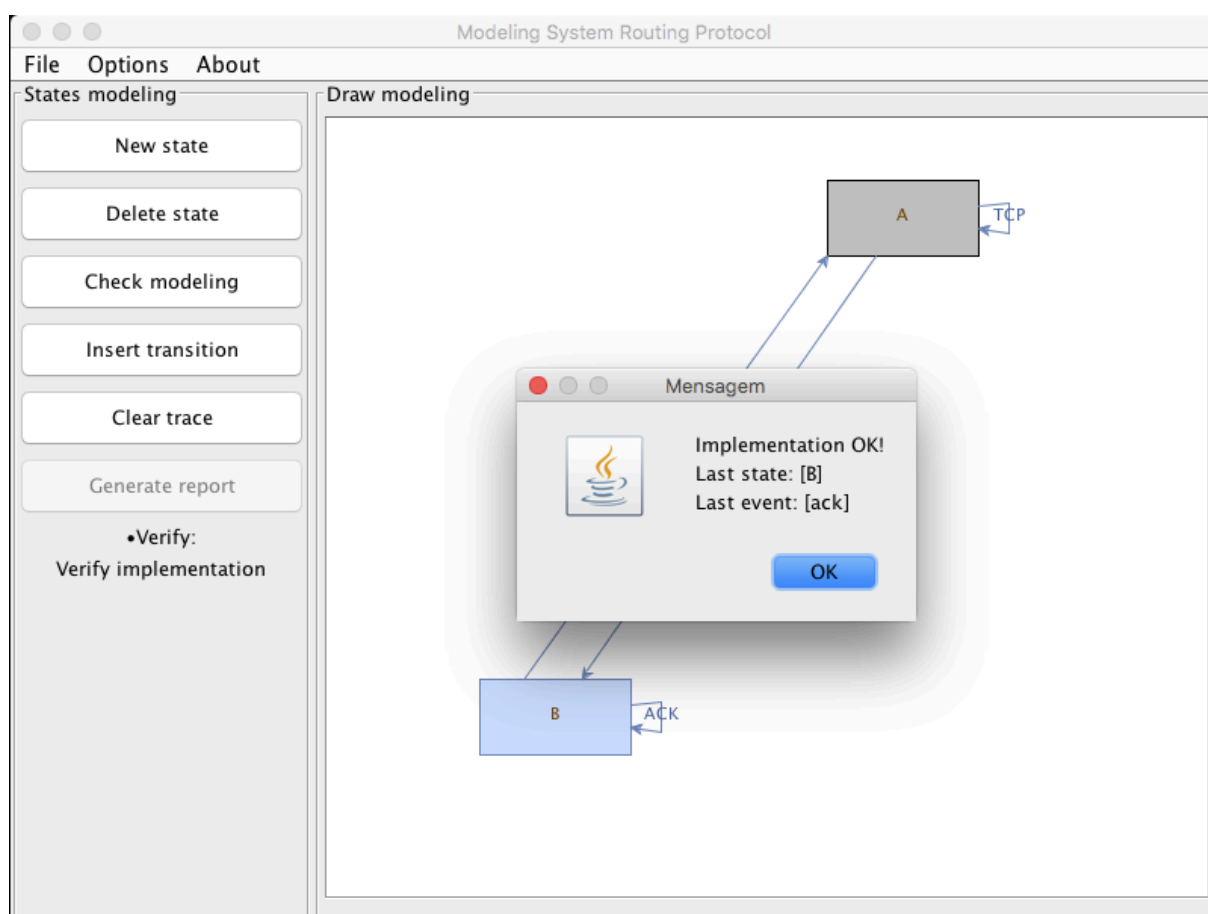


FIGURA 29 - Arquivo trace corresponde ao modelo esperado e exibe mensagem.

Fonte: Autoria própria.

A ferramenta possibilita a geração de um arquivo texto em formato .txt para a verificação das ocorrências, estados e transições alcançadas durante a comparação com a modelagem. Este arquivo pode ser criado somente depois de uma verificação entre modelo e *trace*. Para gerar o arquivo de relatório para análise basta clicar no botão GENERATE REPORT e escolher o local onde deseja salvar o arquivo. É necessário dar um nome ao arquivo e inserir a extensão .TXT ao final do nome atribuído.

Caso o local escolhido estiver correto o arquivo será gerado e salvo, em seguida, uma mensagem de confirmação será exibida como demonstrada na FIGURA 30.

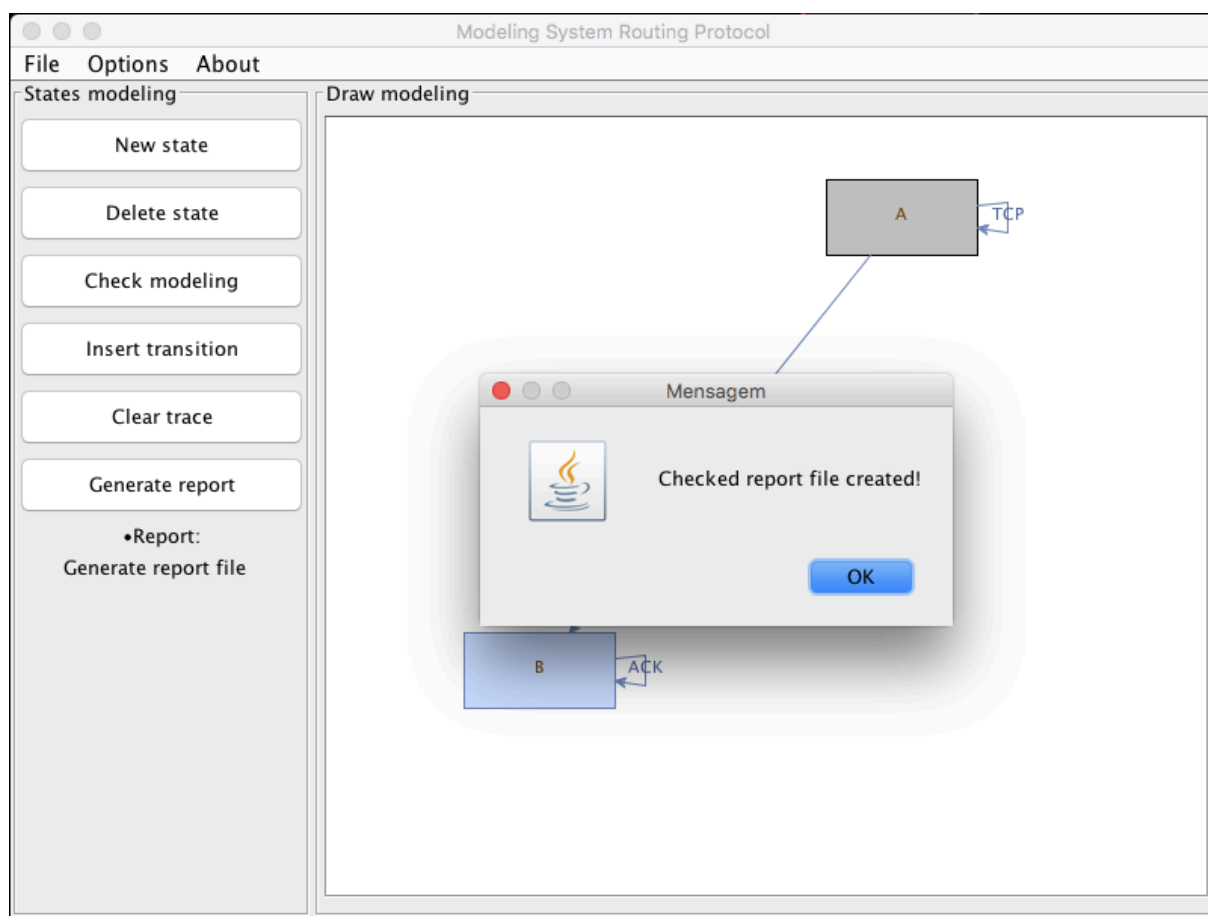
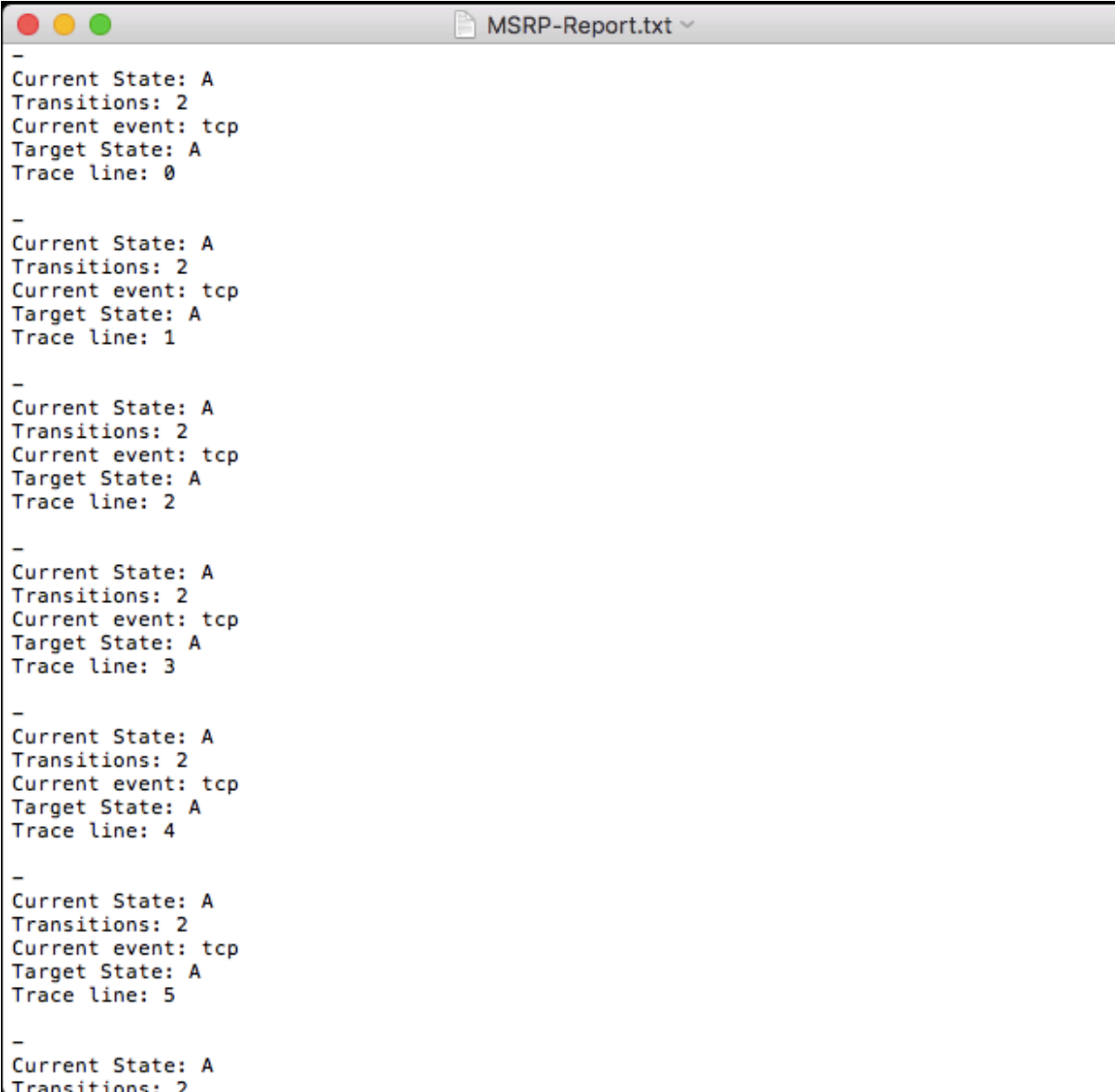


FIGURA 30 - Mensagem de confirmação do relatório gerado pela ferramenta MSRP.

Fonte: Autoria própria.

A FIGURA 31 apresenta parte de um relatório gerado pela ferramenta MSRP.



```
-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 0

-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 1

-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 2

-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 3

-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 4

-
Current State: A
Transitions: 2
Current event: tcp
Target State: A
Trace line: 5

-
Current State: A
Transitions: 2
```

FIGURA 31 - Relatório com as informações ocorridas na verificação pela ferramenta MSRP.

Fonte: Autoria própria.

O arquivo de relatório gerado pela ferramenta MSRP contém os dados dos estados alcançados, o número de transições que estes estados contêm, o evento atual, ocorrido no momento da verificação, o próximo estado e o número da linha do *trace* na qual a verificação se encontra.

Será apresentado na FIGURA 32 um modelo definido pela ferramenta no qual o arquivo *trace* não atende o modelo especificado.

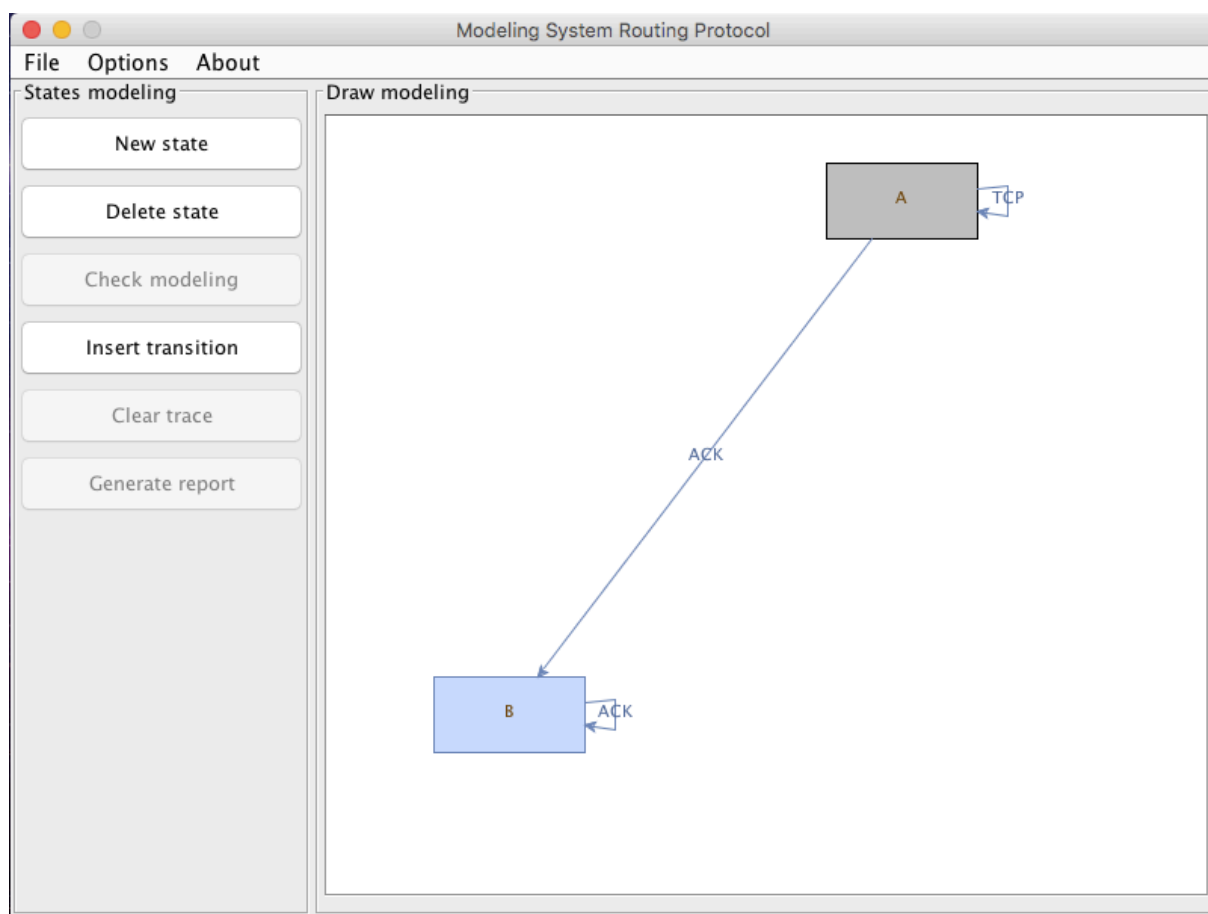


FIGURA 32 - Exemplo de um modelo que não corresponde ao funcionamento do protocolo.

Fonte: Autoria própria.

Após a criação do modelo é necessário realizar a importação do arquivo *trace* para a ferramenta MSRP, assim como apresentada na FIGURA 33.

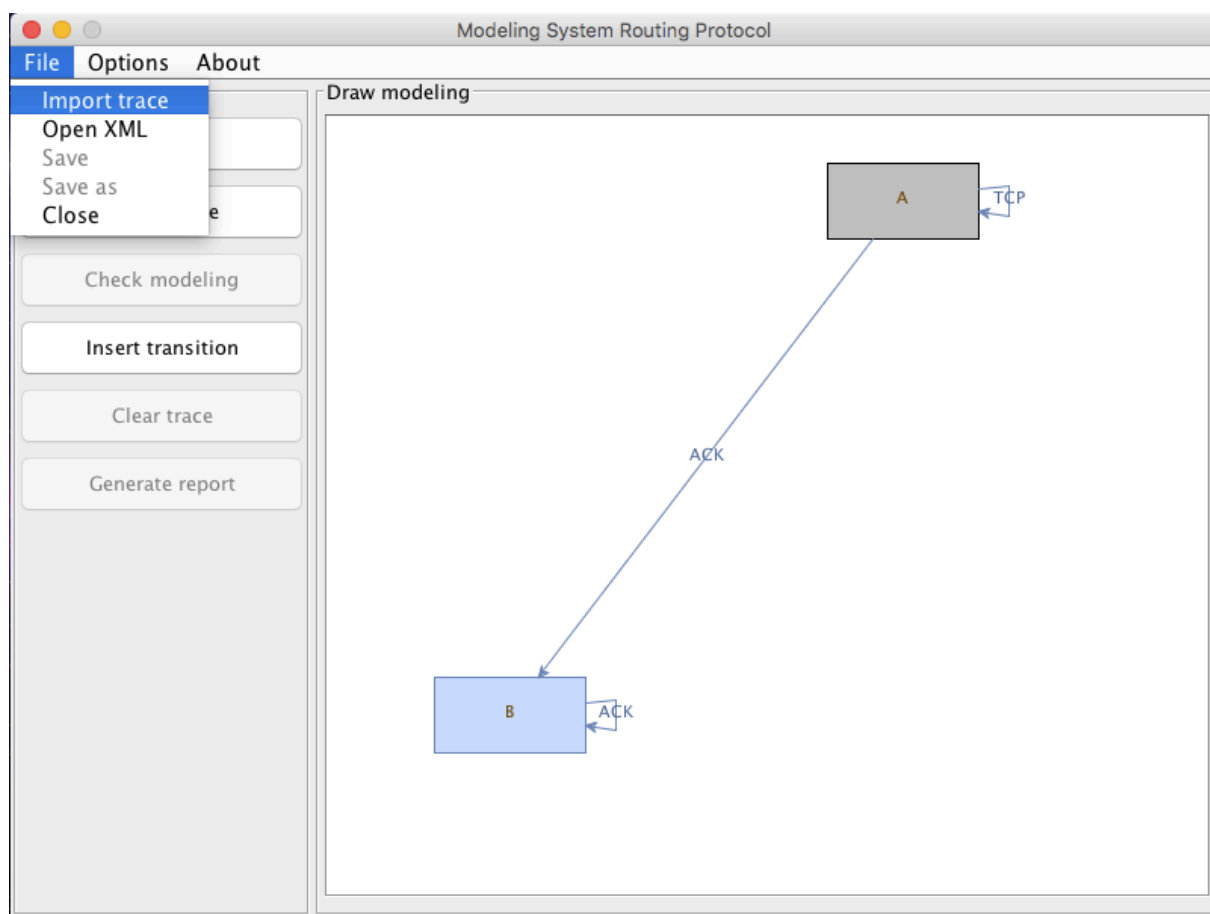


FIGURA 33 - Opção para importação do arquivo *trace* do NS2.

Fonte: Autoria própria.

Após a seleção de importação do *trace* uma tela de seleção é exibida onde será necessário selecionar o arquivo de *trace* desejado. Será utilizado o mesmo arquivo demonstrado na FIGURA 27.

A FIGURA 34 apresenta a seleção do arquivo de *trace* para a ferramenta MSRP.

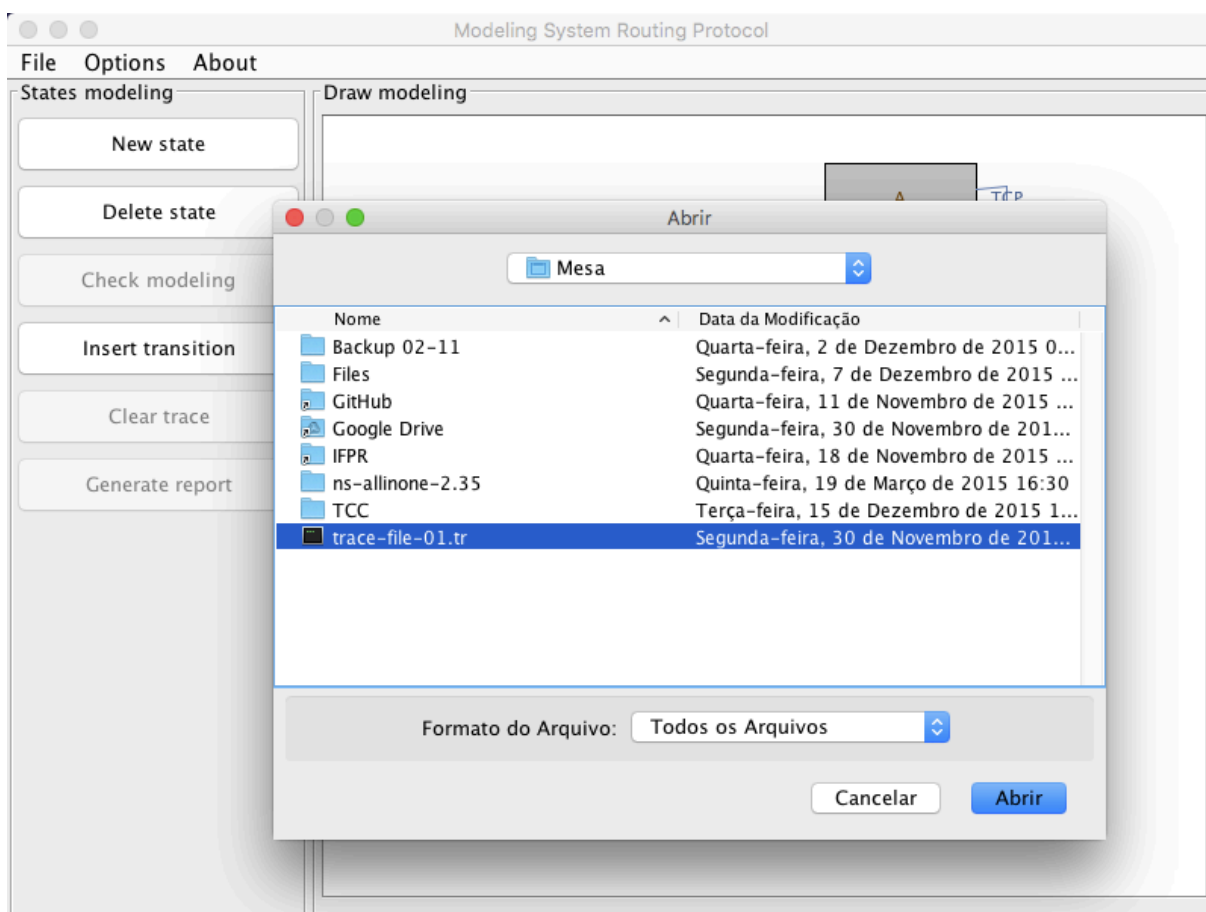


FIGURA 34 - Seleção do arquivo *trace* pela ferramenta MSRP.

Fonte: Autoria Própria.

Com o *trace* importado corretamente pela ferramenta, será exibida uma mensagem de confirmação do processo de importação assim como é demonstrada na FIGURA 35.

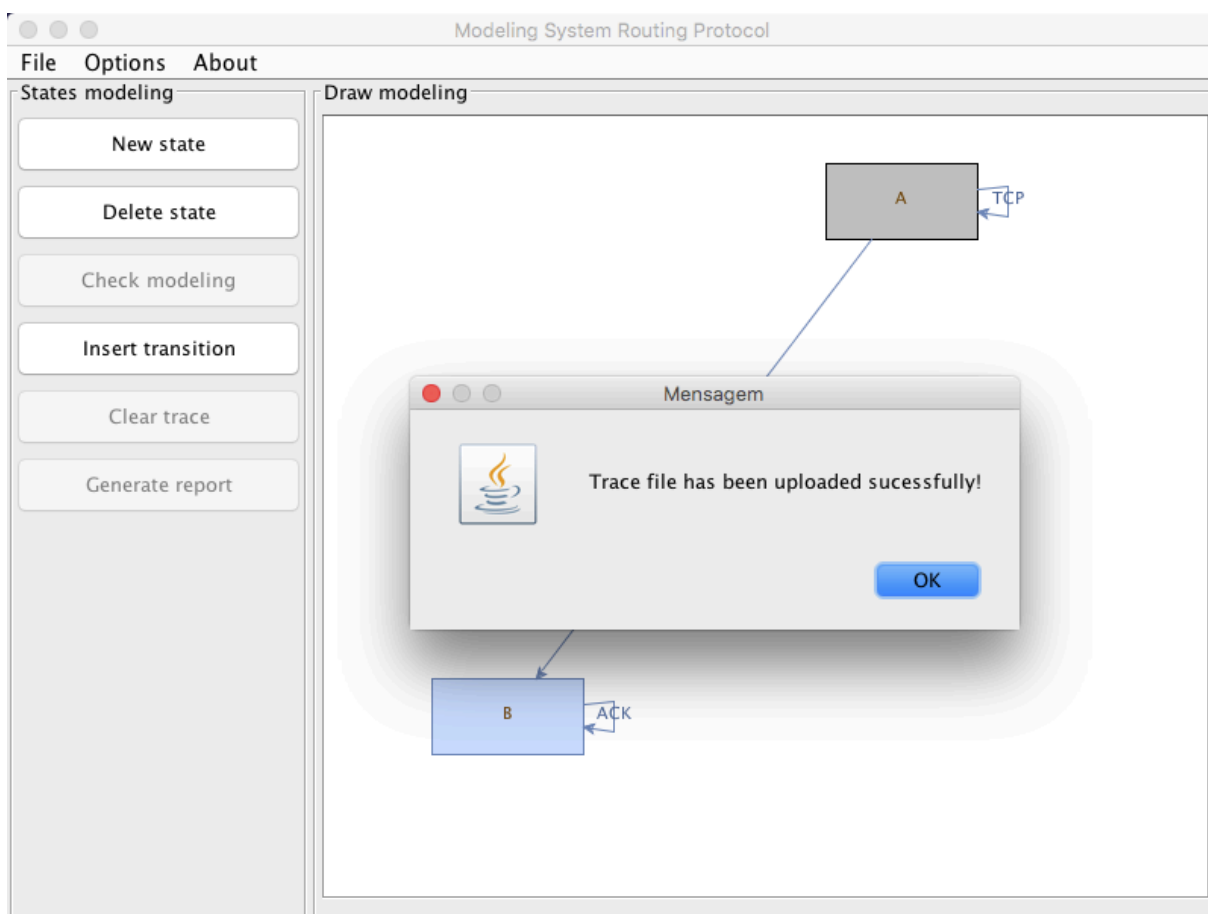


FIGURA 35 - Exibição de mensagem do arquivo *trace* carregado com êxito.

Fonte: Autoria própria.

Após a confirmação de importação do arquivo *trace* a verificação pode ser realizada através do botão *Check Modeling*.

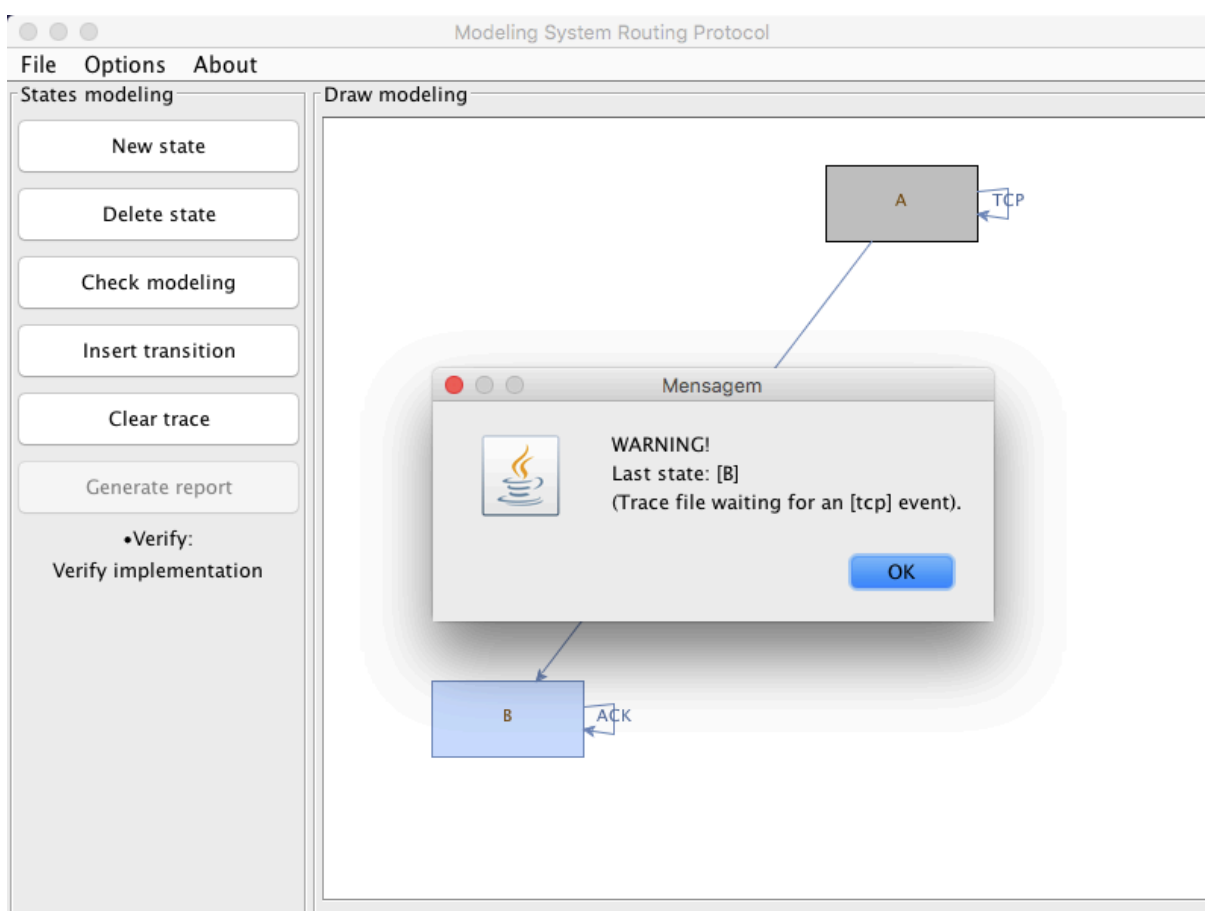


FIGURA 36 - Arquivo *trace* não corresponde ao modelo esperado e exibe mensagem de atenção.

Fonte: Autoria própria.

O modelo especificado, apresentado na FIGURA 32 não contém a transição de B para A com o evento TCP, assim como apresentada na FIGURA 24, por este motivo uma mensagem de atenção será exibida pois a verificação irá terminar no estado B visto que o evento esperado TCP não ocorre neste estado. Dessa a ferramenta exibe uma mensagem alertando que o *trace* não corresponde ao modelo especificado, apresentando em qual estado a verificação parou e qual o evento acontece no arquivo *trace* que não contem na especificação.

5 CONSIDERAÇÕES FINAIS

Este trabalho atingiu o objetivo de desenvolver uma ferramenta para validar simulações de protocolos de rede com relação à especificação de máquinas de estados finitos.

Dessa forma, pode-se concluir que é possível aplicar os conceitos da engenharia de *software* como verificação e teste na área de redes de computadores, automatizando o processo de verificação e validando uma possível implementação de protocolo. Esta solução simplifica o processo de verificação, o que descarta a necessidade de realizar uma verificação estática do arquivo de saída do *software* NS2.

A ferramenta MSRP possibilita analisar o comportamento dos protocolos do arquivo *trace* e comparar com a modelagem definida por meio de uma máquina de estados finitos. O sistema é configurável, permitindo que sejam selecionados quais protocolos de rede deseja-se rastrear no arquivo de saída do NS2. Isto possibilita a validação de uma simulação de rede com relação a protocolos de rede específicos.

5.1 CONTRIBUIÇÕES

Este trabalho apresentou as seguintes contribuições:

- Geração do conhecimento de como realizar a comparação entre modelos comportamentais (modelos de máquina de estados finitos) com os resultados de simulações de rede;
- Desenvolvimento da ferramenta MSRP que automatiza parcialmente o processo de verificação de implementação de protocolos de rede;
- Desenvolvimento de um módulo que permite a geração de modelos comportamentais integrado à ferramenta MSRP. Este módulo provê uma facilidade ao uso da ferramenta MSRP.

5.2 TRABALHOS FUTUROS

Este trabalho pode ser complementado com as seguintes atividades:

- Serializar os modelos comportamentais criados pela ferramenta;
- Permitir a verificação de outros dados ocorridos no arquivo trace, como por exemplo, o tempo de envio de pacotes;
- Aprimorar a interface gráfica do usuário.

REFERÊNCIAS

- ALMEIDA, Paulo Roberto de. **Validação de modelo matemático para verificação do comportamento do protocolo TCP em redes assimétricas**. 2013.
- BALDO, Nicola; MAGUOLO, Federico; ROSSI, Michele; ZORZI, Michele. **NS2-MIRACLE: A Modular framework for multi-technology and cross-layer support in Network Simulator 2**: 2007
- BLOME, Abian; OCHOA, Martin; LI, Keqin, PEROLI, Michele, DASHTI, Mohammad Torabi. **VERA: A flexible model-based vulnerability testing tool**: 2013 IEEE Sixth International Conference on *Software Testing, Verification and Validation*.
- BRITO, Roberta C. de; MARTENDAL, Diogo M; OLIVEIRA, Henrique Eduardo M. de. **Máquina de estados finitos de Mealy e Moore**: 2003.
- CAMPESTRINI, Adriano Orlando. **Uma ferramenta de suporte a simulações de rede com o NS-2**: 2005.
- CANDOLO, Marco Arthur Pereira; SIMÃO, Adenilso da Silva; MALDONADO, José Carlos. **MGASet - Uma ferramenta para apoiar o teste e validação de especificações baseadas em máquina de estado finito**: 2001 XV Simpósio Brasileiro de Engenharia de *Software*
- CARVALHO, Renata A. de; FABBRI, Sandra Camargo P. F; MALDONADO, José Carlos. **Um estudo sobre a avaliação de custo de aplicação da análise de mutantes na validação de máquinas de estados finitos**:1999.
- CASTRO, Lucas Fernando Souza de; ALVES, Gleifer Vaz. **Análise e comparação de frameworks para edição de grafos**. 2012 Sulcomp Congresso Sul Brasileiro de Computação.
- CHAVES, Marcelo Henrique Peixoto Caetano. **Análise de estado de tráfego de redes TCP/IP para aplicação em detecção de intrusão**: 2003.
- DAI, Zhiyong; et al. **Network penetration testing scheme description language**. 2011 International Conference on Computational and Information Sciences.
- DO, Nguyet Quang; et al. **Open-source testing tools for smart grid communication network**. 2013 IEEE Conference on Open Systems (ICOS).
- EL-FAR, Ibrahim k.; WHITTAKER, James A. **Model-based software testing**. Encyclopedia on *Software Engineering*. Florida: Wiley, 2001.
- FANTINATO, Marcelo. **Crêterios de teste funcional baseados em máquinas de estados finitos estendidas**: 2002.

FANTINATO, Marcelo. **Teste de software baseado em máquina de estados finitos – uma revisão**: 2002 VIII Congreso Argentino de Ciencias de la Computación.

FILHO, Wilson de Pádua Paula. **Engenharia de Software**. Fundamentos, métodos e padrões. 3. ed. Rio de Janeiro: LTC, 2011.

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores**. Local: Editora, 2006.

MASIN, David Teixeira; XAVIER, Christopher Breno Coelho; JÚNIOR, José Fernando de Almeida Teobaldo. **O uso de uma máquina de estados finitos para modelar um protocolo**: 2013.

MEETEI, Kh. Playtoni; et al. **Design and development of a handoff management system in LTE networks using predictive modelling**. 2009.

MENDES, Douglas Rocha. **Redes de computadores**. Teoria e Prática. 1. ed. São Paulo: Novatec, 2007.

MORAES, Alexandre Fernandes de. **Redes de computadores**. Fundamentos. 7. ed. São Paulo: Érica Ltda, 2013. 15 p.

MYERS, Glenford J.; BADGETT, Tom; SANDLER, Corey. **The art of software testing**. 3 ed. New Jersey: John Wiley & Sons, Inc., 2012.

NETO, Arilo Claudio Dias. **Seleção de técnicas de teste baseado em modelos**: 2009.

OLIVEIRA, D.M; CRUZ, R. dos S.; SALGUEIRO, R. J. P de B; ROCHA, T. **An integrated development environment for the NS-2 network simulator**. 2012 Scientia Plena. Vol 8. Num. 3

OLIVEIRA, D.M; CRUZ, R. dos S.; SALGUEIRO, R. J. P de B. **NS-Facilities - Uma ferramenta de apoio ao desenvolvimento de simulações**. 2010.

OROZCO, Alex Mulattieri Suarez. **Linha de produtos de testes baseados em modelos**: 2009.

PARIS, Javier; ARTS, Thomas. **Automatic testing of TCP/IP implementations using quickcheck**. 2009 ICFP International Conference on Functional Programming.

PASSOS, Rodrigo Marinho. **Modelagem e implementação de protocolos com e sem fio de um sistema integrado hardware/software em tempo real**: 2002.

PRESSMAN, Roger S. **Engenharia de Software**. Uma abordagem Profissional. 3 ed. São Paulo: Pearson, 1995.

PRESSMAN, Roger S. **Engenharia de Software**. Uma abordagem Profissional. 6 ed. São Paulo: Pearson, 2006.

PRESSMAN, Roger S. **Engenharia de Software**. Uma abordagem Profissional. 7 ed. São Paulo: AMGH, 2011.

SELEGUIM, Guilherme Cestarolli. **Máquina de estados finitos TCP e Aplicações em rede**. 2003.

SHEN, Lu; et al. **Automatic generation for penetration testing scheme analysis model for network**. 2011 International Conference on Computational and Information Sciences.

SILVA, J.C; SILVA, J.L; CAMPOS, J.C; SARAIVA, J.A. **Uma abordagem para a geração de casos de teste baseada em modelos**: 2013

SOMMERVILLE, Lan. **Engenharia de Software**. 8. ed. São Paulo: Person, 2007.

SOUZA, Arthur; MARUOKA, César Toshiya, RODRIGUES, Diego Alan, TEIXEIRA, Talisman. **Simulações de redes wireless**: 2005.

TANENBAUM, Andrew S. **Computer networks**. 4. ed. Amsterdam: Editora Campus, 2003. 45 p.

TONSING, Sérgio Luiz. **Engenharia de Software**. Análise de projeto de sistemas. 2ed. Rio de Janeiro: Ciência Moderna, 2008.

VIANA, Aline Carneiro; MAAG, STEPHANE; ZAIDI, Fatiha. **One step forward: linking wireless self-organizing network validation techniques with formal testing approaches**. 2011 ACM Computing Surveys (CSUR).

VICCARI, Leonardo Davi. **Automação de teste de software através de linhas de produto e testes baseados em modelos**: 2009.

WU-HEN-CHANG, Antal; et al. **A new approach in model-based testing: designing test models in TTCN-3**. 2011 international conference on Integrating System and Software Modeling

XIAO, Ming-ming; YU, Shun-Zheng; WANG, Yu. **Automatic network protocol automaton extraction**. 2009 Third International Conference on Network and System Security.

YI-CHANG, Liu; YING-HUI, Chen; FENG, Qi; XUE-SONG, Qiu. **A model-driven conformance testing method for 3G network management north bound interface**. 2010 Software engineering and Service sciences (ICSESS), IEEE International Conference