



UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Gabriel Covello Furlanetto

Geração de Simuladores de Filas para Diferentes Contextos
com Estudo de Casos para Redes de Computadores

São José do Rio Preto
2016

Gabriel Covello Furlanetto

Geração de Simuladores de Filas para Diferentes Contextos
com Estudo de Casos para Redes de Computadores

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Orientadora: Prof^ª. Dr^ª. Renata Spolon
Lobato

São José do Rio Preto
2016

Gabriel Covello Furlanetto

Geração de Simuladores de Filas para Diferentes Contextos
com Estudo de Casos para Redes de Computadores

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Comissão Examinadora

Prof^a. Dr^a. Renata Spolon Lobato
Universidade Estadual Paulista “Júlio de Mesquita Filho” – São José do Rio Preto
Orientadora

Prof. Dr. Alexandro José Baldassin
Universidade Estadual Paulista “Júlio de Mesquita Filho” – Rio Claro

Prof. Dr. Henrique Mongelli
Universidade Federal do Mato Grosso do Sul

São José do Rio Preto
2016

Aos meus pais Ana e Rogério
À minha avó Maria
Aos meus tios José Mário e Alexandre.

Agradecimentos

Neste momento, gostaria de agradecer a Deus e a todas as pessoas que sempre me deram e me dão forças para a continuidade de meu trabalho, principalmente à minha família.

À professora Renata Spolon Lobato e ao professor Aleardo Manacero Jr. pela dedicação, orientação, profissionalismo e também amizade.

Ao pessoal do Grupo de Sistemas Paralelos e Distribuídos, Lucas, Renan, Fernanda, Fernando, Mário, Victor Hugo, Brunno, Cássio, João, Diogo, Gabriel, Paulo e Victor pela amizade e companheirismo. Em especial ao Denison, que além da amizade e companheirismo inúmeras vezes também me auxiliou profissionalmente.

Aos companheiros de faculdade e profissão, Rafael, Danilo, Gabriel, Arthur, Daniel, Vinícius Galhardi, Vinícius Oliveira, Raphael, Guilherme, William, Eduardo e Matheus.

Aos companheiros de mestrado Diego e Renan.

Aos amigos Liliam, Marcelo, Ariéli, Priscila, Nayara e Victor.

À Jéssica pela compreensão, companheirismo e apoio em todos os momentos.

À Capes, pelo apoio financeiro.

"Seu trabalho vai preencher uma parte grande da sua vida, e a única maneira de ficar realmente satisfeito é fazer o que você acredita ser um ótimo trabalho. E a única maneira de fazer um excelente trabalho é amar o que você faz."

- Steve Jobs

Resumo

A busca por aprimorar o desenvolvimento de aplicações complexas tem aumentado gradativamente. Além disso, procura-se a redução de gastos ao implementar a aplicação, como também o aumento da segurança, o que torna muito importante a técnica de solução de modelos e a simulação de sistemas. Isso ocorre principalmente nas áreas comerciais, industriais e com finalidade de pesquisa. Visando facilitar a utilização da técnica de solução de modelos e a simulação de sistemas, neste projeto é apresentado o desenvolvimento de um gerador de simuladores baseado em filas, o Yasc (*Yes, a simulator's compiler*). Esta ferramenta possibilita a construção de aplicações capazes de solucionar os problemas ditos anteriormente de maneira simples e sem a necessidade de codificação. Assim, a partir de parâmetros informados pelo usuário como entrada, o gerador fornece, como saída, uma ferramenta de simulação. Apresenta-se também um estudo de caso de redes de computadores em que o Yasc, com sua implementação já finalizada, foi utilizado para gerar simuladores de redes e para realizar a solução de modelos e simulação de ambientes reais, cujos resultados foram comparados aos de outro simulador específico da área, o NS-3 (*Network Simulator*).

Palavras-chaves: Modelagem, simulação, padrões.

Abstract

The search to improve the development of complex applications has gradually increased. Besides that, it is aimed to reduce the expenses as much as implements regarding the building of the application, also its security increase, which makes the models solution technique and the simulation of systems very important. This occurs in the commercial, industrial and research-purposed areas. To facilitate the models solution technique and the simulation of systems, this project presented the development of a simulator generator based on queues, the Yasc (**Y**es, **a** simulator's compiler). This tool makes it possible to generate applications that can simply solve problems mentioned before without coding. Thus, based on parameters from the user's input, the generator provide, as output, a simulation tool. Furthermore, we present a case study of computer networks where the Yasc, with your finished implementation, was used to generate network simulators and to perform models solution and simulations of real enviroments, whose results were compared to other specific simulator area, the NS-3 (Network Simulator).

Keywords: Modeling, simulation, templates.

Lista de ilustrações

Figura 1 – Diagrama gráfico de uma fila e um servidor (SOARES, 1992)	21
Figura 2 – Diagrama gráfico de uma fila e vários servidores (SOARES, 1992)	21
Figura 3 – Diagrama da rede básica múltiplas filas e um servidor (SOARES, 1992)	22
Figura 4 – Diagrama para a rede básica com múltiplas filas e múltiplos servidores (SOARES, 1992)	22
Figura 5 – Diagrama para a rede básica de servidor infinito (SOARES, 1992)	23
Figura 6 – Fluxograma de simulação (BANKS et al., 2009) - Traduzido	24
Figura 7 – Algoritmos do evento de chegada e término de atendimento de clientes (SOARES, 1992)	26
Figura 8 – Algoritmo do fluxo de entidades para uma simulação orientada a processos (SOARES, 1992)	27
Figura 9 – Interface gráfica do Omnet++ para modelagem do usuário (extraído de (VARGA, 2016))	32
Figura 10 – Interface gráfica do Sinalgo ((DCG), 2016)	33
Figura 11 – Diagrama conceitual da ferramenta para um simulador	38
Figura 12 – Interface gráfica de criação solicitando o identificador e a imagem que representarão o objeto	40
Figura 13 – Interface gráfica de criação solicitando a forma de comportamento do objeto para a simulação	40
Figura 14 – Interface gráfica de criação solicitando o tipo de fila básica que irá reger o funcionamento do objeto criado	41
Figura 15 – Interface gráfica de criação solicitando sobre a presença ou não de um parâmetro no tipo básico de fila	41
Figura 16 – Interface gráfica para o desenvolvimento de uma função de transferência que rege o funcionamento do objeto	42
Figura 17 – Interface gráfica para descrever falhas e restrições de capacidade	42
Figura 18 – Interface gráfica para escolha das métricas	42
Figura 19 – Exemplo de arquivo de configuração para um simulador gerado	43
Figura 20 – Interface icônica	45
Figura 21 – Diagrama conceitual do motor de simulação - Adaptado de (MENEZES et al., 2012)	46
Figura 22 – Diagrama de classes representando as filas do motor de simulação do iSPD	47
Figura 23 – Diagrama de classes representando as filas do motor de simulação da ferramenta construída (Yasc)	47
Figura 24 – Resultado obtido para a primeira questão objetiva apresentada no questionário	51
Figura 25 – Resultado obtido para a segunda questão objetiva apresentada no questionário	51

Figura 26 – Resultado obtido para a terceira questão objetiva apresentada no questionário	51
Figura 27 – Resultado obtido para a quarta questão objetiva apresentada no questionário	52
Figura 28 – Resultado obtido para a quinta questão objetiva apresentada no questionário	52
Figura 29 – Resultado obtido para a sexta questão objetiva apresentada no questionário	53
Figura 30 – Resultado obtido para a sétima questão objetiva apresentada no questionário	53
Figura 31 – Ambiente de redes de computadores do primeiro modelo testado	55
Figura 32 – Representação da quantidade de testes sem perdas de pacotes	56
Figura 33 – Comparativo entre tempos simulados do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes	56
Figura 34 – Representação da quantidade de testes com resultados próximos ou não do NS-3 com perdas de pacotes	57
Figura 35 – Comparativo entre tempos simulados do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes	58
Figura 36 – Comparativo entre as quantidades de pacotes perdidos pelo NS3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc para o primeiro modelo	58
Figura 37 – Ambiente de redes do segundo modelo testado, representando o laboratório GSPD	59
Figura 38 – Representação da quantidade de testes com resultados próximos ou não do NS-3 sem perdas de pacotes	60
Figura 39 – Comparativo entre tempos simulados do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes	60
Figura 40 – Representação da quantidade de testes com resultados próximos ou não do NS-3 com perdas de pacotes	61
Figura 41 – Comparativo entre tempos simulados do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes	62
Figura 42 – Comparativo entre as quantidades de pacotes perdidos pelo NS3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc para o segundo modelo	62

Lista de tabelas

Tabela 1 – Comparativo entre os simuladores de redes	34
Tabela 2 – Tempos simulados (em segundos) do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes	56
Tabela 3 – Tempos simulados (em segundos) do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes	57
Tabela 4 – Quantidade de pacotes perdidos para o primeiro modelo do NS-3 e do resultado mais próximo obtido pelo simulador gerado pelo Yasc	58
Tabela 5 – Tempos simulados (em segundos) do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes	60
Tabela 6 – Tempos simulados (em segundos) do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes	61
Tabela 7 – Quantidade de pacotes perdidos para o segundo modelo do NS-3 e do resultado mais próximo obtido pelo simulador gerado pelo Yasc	61

Lista de abreviaturas e siglas

LEF	Lista de eventos futuros
FIFO	<i>First In, First Out</i>
LIFO	<i>Last In, First Out</i>
SJF	<i>Short Job First</i>
XML	<i>eXtensible Markup Language</i>
GPSS	<i>General Purpose Simulation System</i>
NED	<i>NEtwork Description</i>
Yasc	<i>Yes, a simulator's compiler</i>
iSPD	<i>iconic Simulator of Parallel and Distributed Systems</i>
DTD	<i>Document Type Definition</i>
GSPD	Grupo de Sistemas Paralelos e Distribuídos

Sumário

1	INTRODUÇÃO	15
1.1	Motivação e objetivo	15
1.2	Visão geral do Yasc	16
1.3	Organização do texto	17
2	TEORIA DE FILAS, MODELAGEM E SIMULAÇÃO DE SISTEMAS	18
2.1	Introdução	18
2.2	Teoria de Filas	19
2.2.1	Notação Kendall	20
2.2.2	Redes de Filas Básicas	20
2.3	Simulação de sistemas	22
2.4	Simulação a partir de modelos básicos (<i>templates</i>)	27
2.4.1	Utilização de modelos básicos (<i>templates</i>) na construção de ferramentas	28
2.5	Simuladores fundamentados em modelos básicos (<i>templates</i>)	29
2.5.1	KanbanSIM	29
2.5.2	PowerTrainSIM	30
2.5.3	RapidSim	30
2.6	Simuladores de redes de computadores	31
2.6.1	OMNeT++	31
2.6.2	Sinalgo	32
2.6.3	NS-3	33
2.7	Comparação entre os simuladores de redes	34
2.8	Métricas de Desempenho	34
2.9	Considerações finais	35
3	TRABALHO DESENVOLVIDO	37
3.1	Introdução	37
3.2	Visão geral sobre a arquitetura do Yasc	37
3.3	Construção das interfaces gráficas para descrição dos objetos que serão disponibilizados no simulador e seu comportamento	39
3.4	Módulos para geração e interpretação de bibliotecas	41
3.5	Interface Icônica	44
3.6	Motor de simulação	45
3.7	Considerações Finais	47

4	TESTES E VALIDAÇÕES	48
4.1	Introdução	48
4.2	Teste de usabilidade e o estudo de casos na área de redes de computadores	48
4.2.1	Procedimentos para a realização do teste	48
4.2.2	Análise do teste de usabilidade	50
4.3	Validação e resultados produzidos pelos simuladores gerados	54
4.3.1	Primeiro modelo simulado	55
4.3.2	Segundo modelo simulado	59
4.4	Considerações finais	62
5	CONCLUSÕES	64
5.1	Trabalhos futuros	64
5.2	Publicações	65
	Referências	66
	APÊNDICES	68
	APÊNDICE A – PLANO DE TESTE	69
	APÊNDICE B – QUESTIONÁRIO DE VALIDAÇÃO	71
	APÊNDICE C – MANUAL DE USO	72
	APÊNDICE D – VALIDAÇÃO DAS IMPLEMENTAÇÕES DE FILAS BÁSICAS	82

1 Introdução

No cenário atual, procura-se cada vez mais por aprimorar o desenvolvimento de aplicações complexas, não apenas de sistemas computacionais. Isso ocorre não só devido à busca por redução de gastos ao implementar-se ou construir-se a aplicação, mas também devido a questões como segurança e até mesmo a necessidade de alterar-se um sistema sem paralisá-lo, nas situações em que sua pausa pode levar a prejuízos ou até mesmo a riscos. Deste modo, a procura por modelagem e em especial pela simulação tem crescido, principalmente para a análise e desenvolvimento de sistemas, os quais podem ser comerciais, industriais, com fins de pesquisa, entre outros.

Assim, percebe-se que as aplicações de simulação são amplas podendo ser observadas (BANKS et al., 2009):

- Em manufatura;
- Em aplicações militares;
- Em simulações de tráfego e relacionadas a transporte;
- Em aplicações de saúde.

Neste trabalho será realizada uma fundamentação teórica sobre modelagem, simulação, modelos básicos (*templates*), teoria de filas e aplicações para tais conceitos. Temas estes essenciais para o objetivo principal deste projeto, o desenvolvimento do Yasc (*Yes, a simulator's compiler*), um gerador de simuladores de filas para diferentes contextos.

1.1 Motivação e objetivo

A observação do comportamento de sistemas é algo importante, uma vez que pode contribuir para sua melhoria em termos de eficiência e também para sua modificação, reduzindo custos e possibilitando sua avaliação sem a necessidade do sistema físico existir.

Maneiras que solucionem o problema de eficácia e redução de tempo no processo de desenvolvimento de *software* possuem como etapa gerar ferramentas que aumentem a eficiência do desenvolvimento ou até mesmo oferecer ao usuário aplicações que possibilitem com que ele desenvolva o seu próprio simulador (SPOLON et al., 1996).

Devido ao fato de não se ter encontrado, na literatura, uma ferramenta capaz de gerar qualquer tipo de simulador de filas para fazer tais avaliações e modificações já ditas e da pequena quantidade de *softwares* que utilizam modelos básicos em seu funcionamento, surge a motivação

para este trabalho. Ele trará contribuições para os usuários, uma vez que possibilitará que este crie sua própria ferramenta de maneira simples e fácil, além de que contará com uma interface gráfica intuitiva, a qual terá por finalidade fazer com que usuários leigos em programação e linguagens computacionais possam utilizar o *software* sem maiores problemas.

Este trabalho tem como objetivo o desenvolvimento do Yasc (*Yes, a simulator's compiler*), um gerador de simuladores de filas, em que o usuário especifica uma aplicação através de uma interface gráfica e, a partir dessa especificação, a ferramenta realiza a construção automática de um produto, ou seja, um simulador.

Tal ferramenta pode ser aplicada para determinados contextos, sendo eles representados por problemas que possam ser modelados por sistemas de filas, como por exemplo:

- Sistemas de processamento distribuídos contemplando *clusters*, *grids* e *clouds*;
- Transmissão de pacotes, contemplando elementos de chaveamento, *hosts* individuais, pacotes e falhas;
- Sistemas digitais, contemplando componentes funcionais básicos em diferentes níveis, como portas lógicas, subsistemas de memória, elementos de processamento entre outros, funcionalidade que está em desenvolvimento.

No protótipo desenvolvido, ou seja, na primeira versão implementada do Yasc, escolheu-se as redes de computadores para estudo de caso e validação.

1.2 Visão geral do Yasc

O Yasc foi desenvolvido de maneira a permitir que o usuário especifique quais redes de filas básicas devem ser utilizadas pela aplicação final, ou seja, o simulador gerado, de qual forma elas interagem e como um sistema por ele representado se comporta.

Todas essas informações são coletadas a partir de uma interface gráfica que preenche campos em um arquivo de configuração. A partir do arquivo de configuração, um interpretador de modelos reconhece as especificações feitas pelo usuário, as quais trabalham junto ao motor de simulação, componente responsável pela simulação propriamente dita, em que estão implementadas as filas.

O gerador de simuladores tem como função adicionar uma interface icônica ao motor que permite a criação de modelos para a classe de sistemas em questão e após a simulação de um destes modelos, fornece ao usuário as métricas obtidas.

Desta forma, o Yasc tem como base a utilização de modelos básicos (*templates*) com a finalidade de identificação de quais componentes são mapeados em quais centros de serviço.

Uma vez identificados os *templates*, é possível realizar a geração de um novo simulador, em que os ícones podem representar entidades distintas, porém com comportamentos semelhantes.

Neste trabalho é apresentado um estudo de caso relacionado com redes de computadores que, como será demonstrado, pode ser desenvolvido por um usuário de maneira simples.

1.3 Organização do texto

A estrutura deste trabalho divide-se em 5 capítulos. Após a introdução, no capítulo 2 são apresentados conceitos sobre teoria de filas, modelagem e simulação de sistemas, modelos básicos (*templates*) e aplicações relacionadas a simulação. No capítulo 3 discute-se o desenvolvimento do projeto. No capítulo 4 são descritas as atividades para validação e verificação da ferramenta. No capítulo 5 são apresentadas as conclusões e trabalhos futuros.

2 Teoria de filas, modelagem e simulação de sistemas

2.1 Introdução

Sistemas podem ser definidos como uma combinação de componentes que agem juntos para desempenhar uma função que não era possível para as partes individuais (RADATZ, 1997).

Com a finalidade de avaliação de sistemas, estando estes em desenvolvimento ou não, o conceito de análise de desempenho torna-se fundamental. Para coletar as informações do sistema em estudo, métodos como técnicas de avaliação existem.

Essas técnicas são separadas em dois tipos, as técnicas de aferição e as técnicas de solução de modelos (SPOLON et al., 1994):

- **Técnicas de aferição:** consiste na construção de protótipos do sistema, utilização de *benchmarks*, que submetem o sistema a uma determinada carga e assim examinam como ele se comporta e, técnicas de coletas de dados, em que a avaliação é feita no sistema real, sendo seu desempenho medido através de algum hardware ou software específico para tal fim.
- **Solução de modelos:** técnica de solução de modelos analíticos, ou seja, o sistema é avaliado através de cálculos matemáticos ou também por meio da simulação, alvo de estudo neste projeto.

A simulação consiste da imitação de uma operação que ocorre no mundo real ou de um processo ao longo do tempo. Para desenvolvê-la, é necessário um estudo do ambiente em que esta ocorre, construindo para ele um modelo que possui como função reproduzir o comportamento de um sistema (BANKS et al., 2009) (CASSANDRAS; LAFORTUNE, 2008). É uma maneira adequada de representação de um sistema caro e complexo. Isso ocorre uma vez que os demais métodos de avaliação de desempenho já citados são mais complexos e difíceis de implementar. Além disso, alguns deles fazem testes em ambientes reais, podendo assim danificar tais sistemas de maneira irreversível ou apresentar perigos aos usuários. Tem-se como exemplo destes problemas os testes realizados na linha de pesquisa aeronáutica e nuclear, em que avaliações no ambiente real podem causar catástrofes. Riscos estes que não existem em um sistema simulado.

Com isto, a simulação apresenta grande crescimento no cenário computacional devido às vantagens que oferece com relação a análises feitas com experimentação em sistemas reais. Dentre elas estão o uso de simulação quando esta permite estudo e experimentos em um sistema

complexo, sendo seus efeitos analisados através do comportamento do modelo. A possibilidade de verificação de quais variáveis possuem maior ou menor importância para tal sistema e como elas interagem, além de experimentar novos planos de projetos antes de sua implementação, de modo a se preparar para as consequências de tais alterações. A redução de custos com testes e a possibilidade de realizá-los mesmo quando não podem ocorrer interrupções do sistema real. E a oportunidade de apresentação ao usuário de uma perspectiva do modo como o sistema funciona (BANKS et al., 2009).

Desta forma, neste capítulo serão abordados conceitos de teoria das filas (seção 2.2), técnicas de solução de modelos e simulação (seção 2.3), a simulação fundamentada em modelos básicos ou padrões (*template based simulation*) (seção 2.4). Por fim, serão abordadas também aplicações para tais conceitos (seção 2.5), simuladores de redes de computadores (seção 2.6) e métricas para análise de desempenho (seção 2.8).

2.2 Teoria de Filas

A formação de filas deriva de um fenômeno causado quando a quantidade de serviços oferecidos torna-se menor do que o esperado para a demanda pelo serviço (HILLIER; LIEBERMAN, 1988).

Tal fenômeno ocorre com frequência, fazendo com que as pessoas deparem-se com filas nos mais diversos cenários, como por exemplo no trânsito, quando os automóveis utilizados para locomoção excedem a capacidade de ocupação das vias, nos caixas das lojas em que há maior número de consumidores do que de funcionários prontos para efetuar o atendimento, ou mesmo nos bancos enquanto aguarda-se para ser atendido nos caixas (KLEINROCK, 1975). Também depara-se com tal situação nos sistemas computacionais, como por exemplo na espera das tarefas por recursos para que possam ser processadas.

Um sistema de filas é definido por componentes e parâmetros, dentre eles estão clientes ou usuários da fila, que são entidades que recebem o serviço prestado pelos centros de serviço, que consistem nos recursos disponibilizados pelo sistema, ou seja, os servidores e as filas. Tais servidores prestam o serviço ao longo de um intervalo de tempo, que representa a quantidade necessária para executar o serviço solicitado por um cliente. Além disso, há servidores disponibilizados para trabalhar em um centro de serviço e também um número máximo de clientes suportado pelo sistema, sendo ele conhecido como a capacidade do sistema. Definem-se o intervalo de tempo entre a chegada de clientes ao sistema e um algoritmo de escalonamento, que controla a inserção e retirada dos clientes da fila que figuram como principais FIFO (*First in, first out*), LIFO (*Last in, first out*), SJF (*Short Job First*) e *Round-Robin* (TANENBAUM, 2009) (SOARES, 1992).

A população de chegada é definida pela distribuição da quantidade de clientes que chegam no sistema, sendo o mesmo válido para o serviço que é prestado, ou seja, ele é definido

pela distribuição dos tempos de prestação de serviço. Sendo assim, a quantidade de chegadas de clientes deve ser menor do que a taxa de atendimento do cliente, caso contrário o tamanho da fila poderá crescer indefinidamente (BANKS et al., 2009).

2.2.1 Notação Kendall

Kendall propôs uma notação para sistemas de filas (BANKS et al., 2009). Esta notação tem o formato $A/B/c/N/K$ (KLEINROCK, 1975). Tais letras representam:

- A: a taxa de chegada de clientes ao sistema.
- B: a taxa adotada para a distribuição de serviço.
- c: a quantidade de servidores do sistema.
- N: a capacidade do sistema.
- K: o número máximo de clientes que podem estar presentes no sistema.

Quando "N" e "K" possuem valores infinitos, a notação pode ser reduzida passando a ser representada por $A/B/c$.

Além disso, as distribuições de chegada e serviço podem ser representadas por símbolos específicos como por exemplo, "A" e "B" substituídos por "M", o qual indica uma distribuição exponencial ou de Markov. "D" pode simbolizar distribuição constante, "Ek" uma distribuição k-Erlang e "Hk" pode representar a distribuição hiperexponencial que possui "k" estados (BANKS et al., 2009).

2.2.2 Redes de Filas Básicas

As redes de filas básicas são formadas por centros de serviço que possuem uma ou várias filas e um ou vários servidores:

- **Uma fila e um servidor:** Constitui-se de um tipo de rede básica em que tanto a taxa de chegada quanto a taxa de serviço são distribuições exponenciais e a quantidade de servidores do centro de serviço é igual a um. Nela tem-se também capacidade e população infinitas. Quando um cliente chega até a fila, caso o servidor esteja livre, sua requisição de serviço será executada imediatamente. Caso contrário, o cliente entrará na fila e poderá ser atendido assim que receber permissão, através de um algoritmo de escalonamento (BANKS et al., 2009). Esta fila básica está ilustrada na Figura 1.

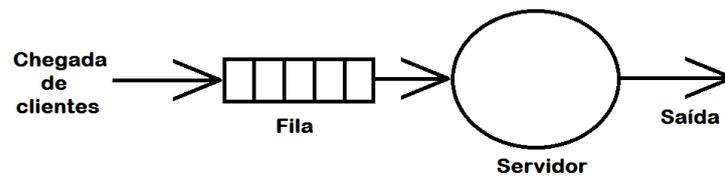


Figura 1 – Diagrama gráfico de uma fila e um servidor (SOARES, 1992)

- Uma fila, múltiplos servidores:** Esta rede básica consiste em um sistema com capacidade e população infinitas. Sua taxa de chegada e de serviço são também distribuições exponenciais. Possui k servidores iguais que compartilham a mesma fila, ou seja, é indiferente se um cliente é escalonado para um servidor ou para outro, uma vez que a maneira de atendimento é a mesma para todos. Assim, caso haja chegada de um cliente, e pelo menos um dos k servidores esteja livre, o cliente já é escalonado e passa a ser atendido. Caso contrário, a fila é formada (KLEINROCK, 1975). Este modelo está representado na Figura 2.

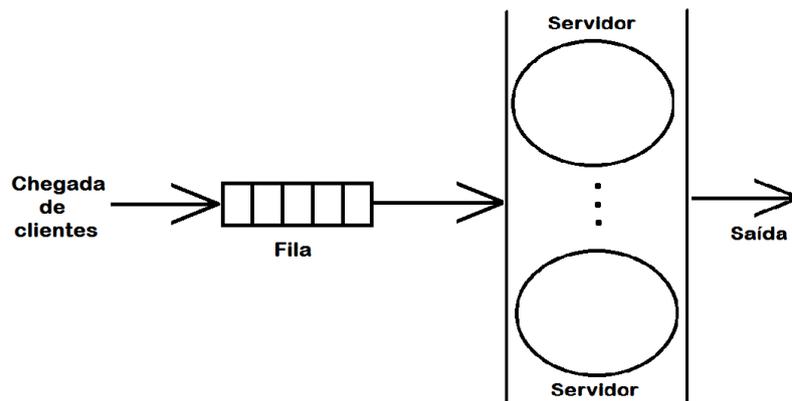


Figura 2 – Diagrama gráfico de uma fila e vários servidores (SOARES, 1992)

- Múltiplas filas e um servidor:** Segue o mesmo esquema de filas já tratado nos dois itens anteriores, com a diferença de que há k filas para um único servidor, como pode-se notar na Figura 3. Neste modelo, é necessário observar-se que deve haver um algoritmo de seleção no momento em que o atendimento de um cliente é finalizado, para decidir-se a qual fila pertence o próximo cliente que receberá o serviço. Esse algoritmo pode ser, por exemplo, um algoritmo de prioridade em que há uma categoria de prioridade para cada fila.
- Múltiplas filas e múltiplos servidores:** Esta fila trabalha com k servidores também. No entanto, ele diferencia-se da rede básica uma fila, vários servidores devido a quantidade de filas, a qual também é maior do que uma. Cada um destes k servidores pode atender a

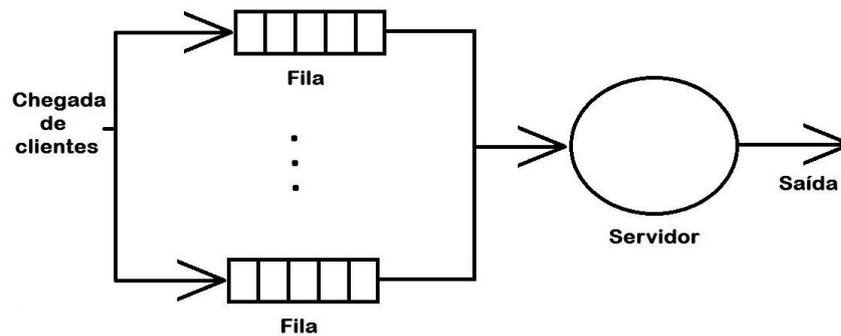


Figura 3 – Diagrama da rede básica múltiplas filas e um servidor (SOARES, 1992)

qualquer uma das m filas, necessitando novamente de um algoritmo de seleção para definir de qual fila será o próximo cliente a ser atendido, como ilustrado na Figura 4.

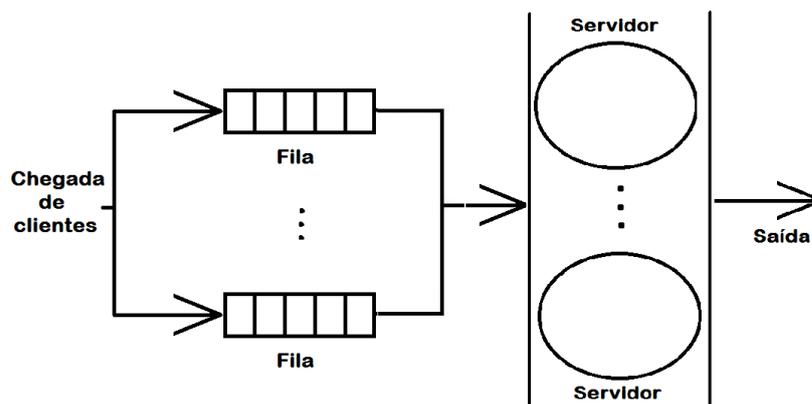


Figura 4 – Diagrama para a rede básica com múltiplas filas e múltiplos servidores (SOARES, 1992)

- **Centro de Atraso ou Servidor Infinito:** Consiste em um modelo em que não há filas devido a elevada quantidade de servidores que atende totalmente a demanda solicitada. Assim, um cliente que chega ao sistema já recebe imediatamente o serviço solicitado de um dos k servidores existentes. A representação deste tipo de modelo é apresentada na Figura 5

2.3 Simulação de sistemas

Técnicas para solução de modelos consistem em uma sequência de procedimentos nos quais devem ser realizados a especificação, parametrização e solução do modelo seguido da apresentação dos resultados obtidos. Neste trabalho, tais técnicas serão tratadas para sistemas discretos, ou seja, sistemas cujos estados das variáveis mudam apenas em pontos discretos ao longo do tempo, pontos estes denominados de eventos.

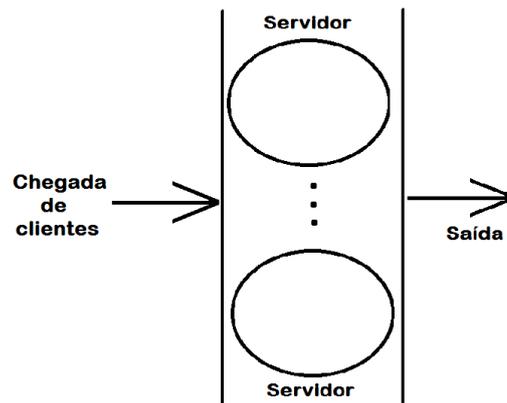


Figura 5 – Diagrama para a rede básica de servidor infinito (SOARES, 1992)

Quando ocorre um evento, há uma mudança do estado do sistema, e um conjunto de variáveis cujas informações descrevem o sistema em um determinado instante são alteradas. Para o armazenamento dos eventos, existe uma lista de notificações, denominada Lista de Eventos Futuros (LEF), a qual é ordenada pelo tempo de ocorrência de cada evento, tempo denominado de tempo simulado, que será medido através de um relógio, podendo assim registrar atraso entre eles (CARDOSO J. VALETTE, 1997) (BANKS et al., 2009).

No fluxograma da Figura 6 é apresentado como funciona o desenvolvimento de um estudo de desempenho por meio de simulação. Nele pode ser observado o desenvolvimento, a validação e a análise dos dados para um projeto genérico, que serão apresentados a seguir.

Formulação do problema, seus objetivos e planos de projeto

Consiste no primeiro passo do desenvolvimento da simulação. Nele, o usuário deve formular seu problema, ou seja, deixar bem claro qual a sua motivação e quais seus objetivos para o desenvolvimento do trabalho e defini-lo com a maior precisão possível. Este é um passo muito importante, uma vez que o tempo gasto para desenvolvê-lo será compensado devido aos erros futuros evitados (BANKS et al., 2009).

Conceitualização do modelo e coleta de dados

A partir dos objetivos já especificados, deve ser definida como será a representação do modelo e a escolha de suas métricas de avaliação. Geralmente, ele pode ser feito de várias formas, destacando-se entre as mais importantes os diagramas de fluxo, algoritmos e planos de projeto, com diagramas de casos de uso e de classe. Nesta parte do desenvolvimento serão gerados os requisitos para os dados de entrada do sistema, especificando quais parâmetros são úteis para a simulação dele (MACDOUGALL, 1989).

Tradução, verificação e validação do modelo

O próximo passo consiste na tradução do modelo para uma linguagem computacional

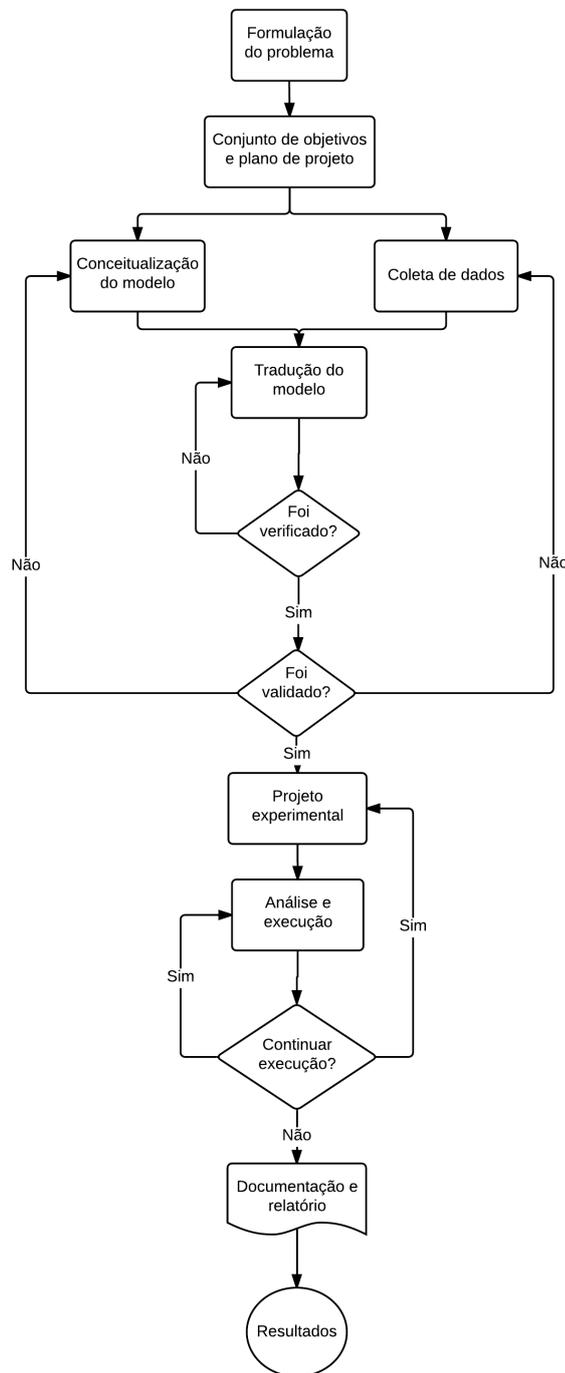


Figura 6 – Fluxograma de simulação (BANKS et al., 2009) - Traduzido

seguido de sua verificação e validação respectivamente, as quais devem ser satisfeitas para que a quarta etapa possa ser iniciada. Na fase de verificação é analisado se o programa de simulação trata-se realmente de uma implementação válida do modelo, enquanto que na fase de validação é avaliado se o programa atende os objetivos propostos a ele, ou seja, se reproduz o comportamento do ambiente real fielmente (SOARES, 1992).

Projeto experimental e análise de resultados

Após a verificação e validação, realiza-se a fase experimental e de análise de resultados. Nela, tem-se como questão fundamental o número de vezes, ou por quanto tempo, um programa deve ser executado. Devido a aleatoriedade das variáveis de entrada de uma simulação e por suas respectivas saídas serem dependentes destas tem-se que sua análise deve ser realizada de acordo com um problema de estatística. O programa de simulação deve ser reexecutado até que seus resultados possam convergir para o intervalo de confiança adotado, podendo ser adotado para isto métodos estatísticos (MACDOUGALL, 1989).

Ao final desta etapa, os resultados devem ser documentados em relatórios, para que possam ser consultados posteriormente, caso necessário, e investigados.

Alguns exemplos de uso para modelagem e simulação (CASSANDRAS; LAFORTUNE, 2008):

- Representação de sistemas de manufatura.
- Desenvolvimento de redes de comunicação e teste de protocolos para manuseio de mensagens.
- Desenvolvimento de aeroportos.
- Desenvolvimento de redes de metrô.
- Simulação de grades computacionais e de computação em nuvem.

Simulação orientada a eventos discretos

Os sistemas podem ser classificados como discretos, contínuos ou mistos. Esta seção será dedicada à simulação orientada a eventos discretos.

Na simulação orientada a eventos, o tempo da simulação só avança de maneira discreta, ou seja, em momentos estanques, quando tais eventos ocorrem. Desse modo, o tempo decorrido entre as alterações de estado do sistema não é relevante para obtenção de resultados (BANKS et al., 2009).

Por exemplo, um caixa de supermercado pode ser considerado um modelo de sistema discreto, em que uma das variáveis de estado consideradas consiste no número de clientes na fila, a qual muda apenas com a chegada de um novo cliente ao caixa ou com o término do serviço

que está sendo prestado a ele, ou seja, a contabilidade do preço dos produtos e a cobrança pelo pagamento destes. Estes eventos podem ser descritos pelos algoritmos da Figura 7 (SOARES, 1992).

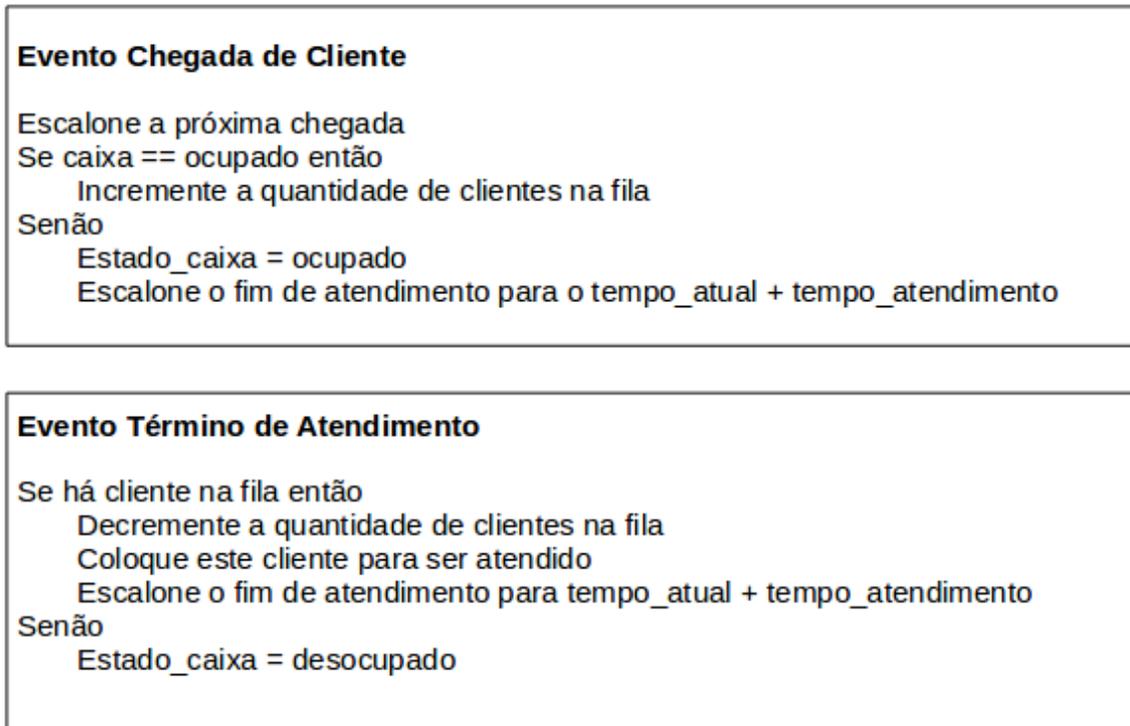


Figura 7 – Algoritmos do evento de chegada e término de atendimento de clientes (SOARES, 1992)

Simulação Orientada a Processos

Em simulação orientada a processos, cada entidade do sistema é descrita através de um processo, o qual geralmente é representado por uma sequência de funções, podendo ser estas do tipo lógico (representação de ações instantâneas realizadas pela entidade que executa essa função ao longo de seu desenvolvimento) ou de atraso de tempo (atraso provocado na entidade por um pequeno intervalo de tempo).

Esse tipo de simulação geralmente é utilizado para sistemas de filas em que as entidades são tomadas como clientes e passam pela rede em que há filas e servidores. Assim, os principais componentes desse tipo de orientação constituem-se de um conjunto de entidades, ou objetos que solicitam por um serviço, um conjunto de atributos que as caracterizam, as funções lógicas ou de atraso, um conjunto de recursos, que proveem serviços para as entidades e por fim, as filas, em que um conjunto delas aguardam para poder usar o recurso (CASSANDRAS; LAFORTUNE, 2008).

Uma simulação orientada a processos segue passos como a definição de entidades do sistema, criação de processos para elas junto da descrição de suas etapas e a execução concorrente

de tais processos. Eles chegam segundo determinada distribuição de probabilidade, recebem serviços requisitados e saem do servidor(algoritmo da Figura 8) (SPOLON et al., 1994).

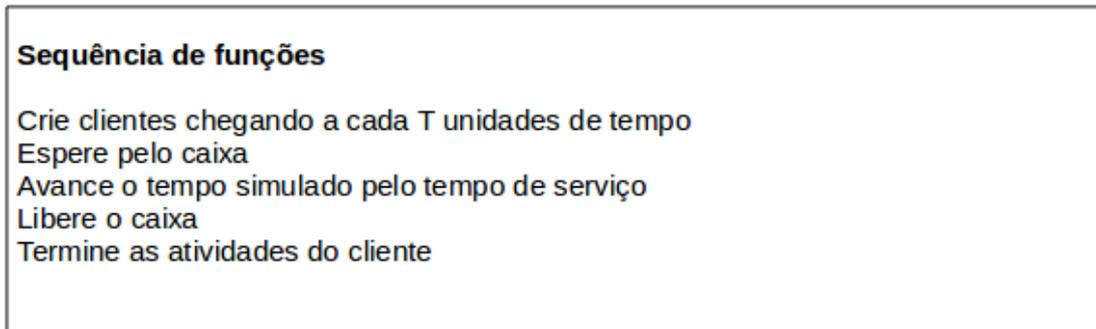


Figura 8 – Algoritmo do fluxo de entidades para uma simulação orientada a processos (SOARES, 1992)

2.4 Simulação a partir de modelos básicos (*templates*)

Uma simulação pode ser implementada de quatro maneiras diferentes:

- Por meio de pacotes de uso específico;
- Por meio de linguagens de programação de uso geral;
- Por meio de linguagens de programação específicas para simulação;
- Por extensões funcionais.

Os pacotes de simulação possuem uso específico, são desenvolvidos para sistemas em particular e apresentam pequena flexibilidade. Linguagens de programação de uso geral consistem de linguagens como por exemplo C, Java e C++, as quais o especialista em *software* deve conhecer para desenvolver um simulador. Linguagens de simulação são linguagens projetadas com a finalidade de solução de modelos para sistemas de vários tipos, podendo assim o modelo de simulação ser visto como um conjunto de eventos ou de processos do sistema, como exemplo delas tem-se a *General Purpose Simulation System (GPSS)* (GORDON, 1978), que é uma linguagem para simulação orientada a eventos discretos. Por fim, as extensões funcionais consistem de bibliotecas desenvolvidas para linguagens de programação de uso geral que quando atuam junto com estas linguagens compõem um ambiente completo de simulação (SPOLON et al., 1994) (SOARES, 1992) (ZHU et al., 2015) (LEI et al., 2007).

A partir dessas maneiras de implementação, surgem simuladores produzidos para uma aplicação específica e também simuladores genéricos. Os produzidos para uma aplicação específica após serem utilizados uma ou várias vezes para a mesma finalidade podem cair em desuso,

enquanto que os simuladores genéricos podem ser utilizados não apenas para um modelo de aplicação, mas também para outras de mesmo tipo, ou até mesmo com finalidades diferentes, possibilitando o reuso de software (DOSS; ULGEN, 2004).

A sequência da etapa de estudos realizada baseia-se em simuladores genéricos e busca apresentar aplicações em que é permitido ao usuário utilizar modelos básicos ou ferramentas genéricas de simulação. Para isso, tem-se como motivação que apesar de as ferramentas de simulação existentes não serem poucas, a maior parte delas não possuem a flexibilidade desejada para serem reutilizadas em diferentes contextos, uma vez que elas podem não oferecer ao usuário todas as funções por ele exigidas (FU; BECKER; SZCZEBICKA, 2015). Com isto, usuários leigos, sendo eles físicos, biólogos, matemáticos, entre outros, inúmeras vezes não encontram um ambiente de simulação que atenda todas as necessidades para seu trabalho, enfrentando dificuldades, já que nem sempre possuem conhecimentos aprofundados de linguagens de programação para escrever um simulador cujas funções sejam voltadas para o seu ambiente de pesquisa.

2.4.1 Utilização de modelos básicos (*templates*) na construção de ferramentas

Modelar é um dos passos mais importantes no estudo sobre simulação, já que o modelo fornece dados à ferramenta para que esta possa realizar a simulação. Retornando informações que possibilitam melhor compreensão dos aspectos de interesse de cada sistema. Assim, a prática de técnicas para construção do modelo em alguns casos, pode representar grande parte do total de tempo de estudo para que um modelo simulável construído seja bem feito e confiável.

É importante a reutilização de modelos em sua totalidade ou com pequenas alterações de componentes ou de maneira parcial na construção de ferramentas de simulação, uma vez que usuários que não são da área de computação e até mesmo pessoas desta área que não possuem conhecimento no ramo de simulação têm utilizado simuladores em suas aplicações. Com isto, o reuso de informações ou de um modelo básico facilitam o uso das ferramentas pelos usuários e as tornam mais eficazes e seguras, já que inserem dados já validados em sua construção (RAMPERSAD, 2012).

Este tipo de modelo básico traz consigo as seguintes vantagens (PATER; TEUNISSE, 1997):

- Diminui o tempo para construir um modelo, de maneira significativa, devido à reutilização.
- Há uma separação entre o desenvolvimento e a implementação.
- Uma vez que o modelo básico esteja elaborado e testado, a dificuldade do usuário passa a ser oculta, facilitando o processo, e deixando-o livre para trabalhar em problemas técnicos e funcionais da aplicação.

- Facilidades para validação uma vez que a reutilização de código faz com que não haja necessidade de novos testes sobre ele.

O conceito de reuso de modelos tem se tornado favorável para os usuários, com sua redução de custo, rápido desenvolvimento e manutenção fácil, esbarrando apenas no problema de limitações no sistema devido à necessidade do seguimento de padrões, ou seja, modelos que não se adequam ao padrão não podem ser reutilizados pelo sistema (RAMPERSAD, 2012).

2.5 Simuladores fundamentados em modelos básicos (*templates*)

Nesta seção dedica-se a mostrar ferramentas de simulação que utilizam modelos previamente construídos em suas abordagens. Entre elas encontram-se KanbanSIM, PowerTrainSim e RapidSim.

2.5.1 KanbanSIM

Kanban representa um termo japonês que significa "sinal", ou seja, o sinal disparado para que haja reposição de materiais para a continuação da produção (PATER; TEUNISSE, 1997).

A sinalização Kanban é usada pelo Sistema Toyota de Produção, controlando e nivelando a produção e minimizando estoques de produtos intermediários e finais. Esse sistema consiste em dividir o trabalho de produção em etapas, e assim, a partir delas fabricar apenas a quantidade de material utilizada pela etapa subsequente, impedindo que haja superprodução.

Pode-se exemplificar o sistema através de um supermercado e seu cliente, em que o cliente vai ao supermercado para comprar o que precisa e na quantidade necessária. Assim, quando realiza a compra, o processo de restituição de produtos do mercado recoloca nas prateleiras apenas o que foi comprado e repõe esta mesma quantidade no estoque a partir de seus fornecedores (SPÓSITO, 2003).

KanbanSIM é um simulador para auxiliar no correto fluxo de materiais para uma linha de produção, podendo ser utilizado para uma linha de produção de veículos ou de qualquer outro material.

O simulador auxilia o usuário a medir o desempenho do sistema de produção, verificando se esse sistema atende pontualmente a demanda de materiais, obtendo para isto, medidas de entrada e saída de materiais através de seu fluxo e também do tempo de entrega, para que assim seja possível otimizá-lo caso necessário.

Ele constitui-se de um simulador genérico, baseado em modelos previamente prontos com uma interface reconfigurável através do *software* Excel, o que garante a geração de novos modelos para a simulação de forma mais rápida.

O usuário do programa pode ter à sua disposição funcionalidades como a troca da taxa de consumo, o uso dos materiais na produção e alterações nas rotas de entregas dos produtos (se necessário, pode ocorrer um realocamento dos caminhos ferroviários percorridos) e, por fim, observar os custos da sobrecarga do sistema.

2.5.2 PowerTrainSIM

Powertrain é um termo em inglês utilizado para descrever a unidade motora de um veículo, ou seja, todo o sistema responsável por movimentá-lo, o qual tem em sua constituição as rodas, eixos, transmissão, motor, admissão e escapamento.

Deste modo, o PowerTrainSIM consiste em um simulador aplicado para a produção e a realização dos testes para este sistema. Para isto, ele é fundamentado em modelos básicos que descrevem o sistema facilitando a descrição de seus elementos.

A simulação de *powertrains* é voltada para a solução de problemas do sistema do seguinte tipo (DOSS; ULGEN, 2004):

- Detecção de gargalos.
- Distribuição de trabalho nos setores pontuais em que há maior necessidade.
- Avaliação das mais variadas situações operacionais.
- Avaliação do desempenho e da eficiência de produção.

Com isto, o sistema também se baseia em oferecer para o usuário rápida solução e facilidade para reuso de modelos, uma vez que permite tanto para leigos quanto para usuários que trabalham com *powertrains* a utilização de modelos básicos, não sendo necessário a criação completa de um novo modelo em cada caso específico.

2.5.3 RapidSim

O Rapsim é um simulador voltado para simulação em ambientes de manufaturas. É constituído de uma biblioteca de modelos básicos (*templates*). Para a construção de tal biblioteca, o desenvolvedor criou uma classificação analisando uma variedade de modelos da área de manufatura. Com este fim, utilizou-se métodos de cladística, os quais consistem em sistemas de agrupamento de modelos com características semelhantes, possibilitando também analisar suas diferenças (RAMBERSAD, 2012). Deste modo, os modelos são analisados e agrupados conforme a cladística e tais agrupamentos formavam as bibliotecas de *templates* para a categoria obtida com a classificação.

O simulador baseia-se em uma ferramenta desenvolvida em *Visual Basic* com interface no Excel, o que o torna amigável para o usuário, o qual pode até mesmo desconhecer técnicas de

linguagens de programação. Ele conta com uma vasta documentação, e sua estrutura é baseada na inserção de modelos pré-definidos (*templates*).

Como consequência de sua interface prática e suas facilidades para o usuário, é possível que a construção e execução do modelo a ser simulado seja mais rápida do que caso a técnica de solução de modelos e a simulação fossem feitas de maneira completa desde a especificação do modelo até a sua estimulação no processo de simulação.

2.6 Simuladores de redes de computadores

Devido a capacidade de proporcionar não apenas comunicação, mas também trocas de dados das mais variadas formas entre grandes áreas geográficas, as redes de computadores estão em contínuo desenvolvimento. Assim, a simulação e realização de testes para a criação de novas arquiteturas e protocolos faz-se necessária, já que permitem o estudo do comportamento do roteamento de pacotes em diferentes topologias (WEINGÄRTNER; LEHN; WEHRLE, 2009).

Nessa seção serão apresentadas ferramentas de simulação de redes de computadores, dentre elas o OMNeT++ que aparece na subseção 2.6.1, o Sinalgo, apresentado na subseção 2.6.2 e por fim o NS-3, apresentado em 2.6.3.

2.6.1 OMNeT++

OMNeT++ é uma biblioteca, desenvolvida em linguagem de programação C++, para simular ambientes de redes de comunicações com e sem fio, redes de filas, entre outros (VARGA; HORNIG, 2008).

A ferramenta possui suporte para Windows, Linux e Mac OS X, e atualmente encontra-se na versão 4.6. É bem documentada, com manuais de uso, porém possui instalação complexa via terminal do Linux, devido às inúmeras dependências de outros programas.

Permite a descrição de modelos tanto de forma textual, por meio da linguagem NED (*Network Description Language*), quanto por meio de interface gráfica para o usuário, a qual tem como propósito principal tornar visível os eventos que ocorrem no interior do modelo ao longo da execução da simulação além de permitir ao usuário realizar sua modelagem de forma gráfica, como pode ser visto na Figura 9 a qual retrata a interface gráfica do simulador junto de um modelo sendo construído, e visualizar os resultados da simulação do mesmo modo.

Embora o OMNeT++ não tenha sido desenvolvido com o intuito de simular redes de comunicação, atualmente vem se tornando popular para tal fim tanto na comunidade científica, como em ambientes industriais (VARGA, 2016).

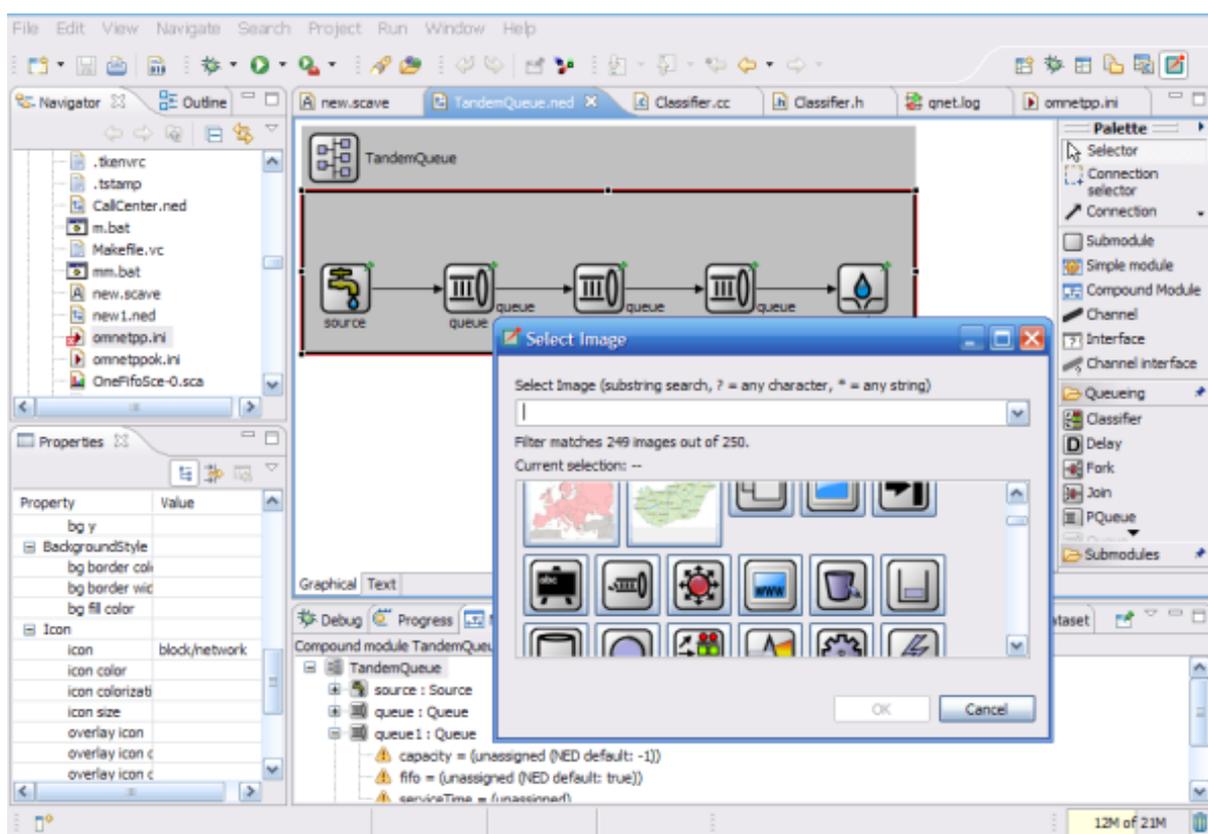


Figura 9 – Interface gráfica do Omnet++ para modelagem do usuário (extraído de (VARGA, 2016))

2.6.2 Sinalgo

Sinalgo é um ambiente de simulação desenvolvido para testar e validar algoritmos de rede, como por exemplo algoritmos de roteamento e transmissão de dados. Foi desenvolvida com a finalidade de simulação de ambientes de rede sem fio.

A ferramenta possui suporte para Windows, Linux e Mac OS X. Encontra-se na versão estável 0.75.3. É bem documentada, com manuais e exemplos de uso. Não requer instalação, sendo apenas necessário que seu código fonte seja baixado, executado ou modificado, sendo possível a utilização de ambientes de desenvolvimento para tais práticas como por exemplo o Netbeans e o Eclipse.

O simulador possui um algoritmo de prototipagem rápida em JAVA, além de um amplo conjunto de condições da rede, sob o qual podem ser realizados testes ((DCG), 2016).

Possui interface gráfica (mostrada na Figura 10), e exemplos de simulação disponível, porém obriga com que o usuário codifique seu ambiente de simulação praticamente todo em linguagem Java, exigindo para isso domínio de uma linguagem de programação, dificultando seu uso por usuários que não possuem experiência em programação ou que não estejam familiarizados com esta linguagem.

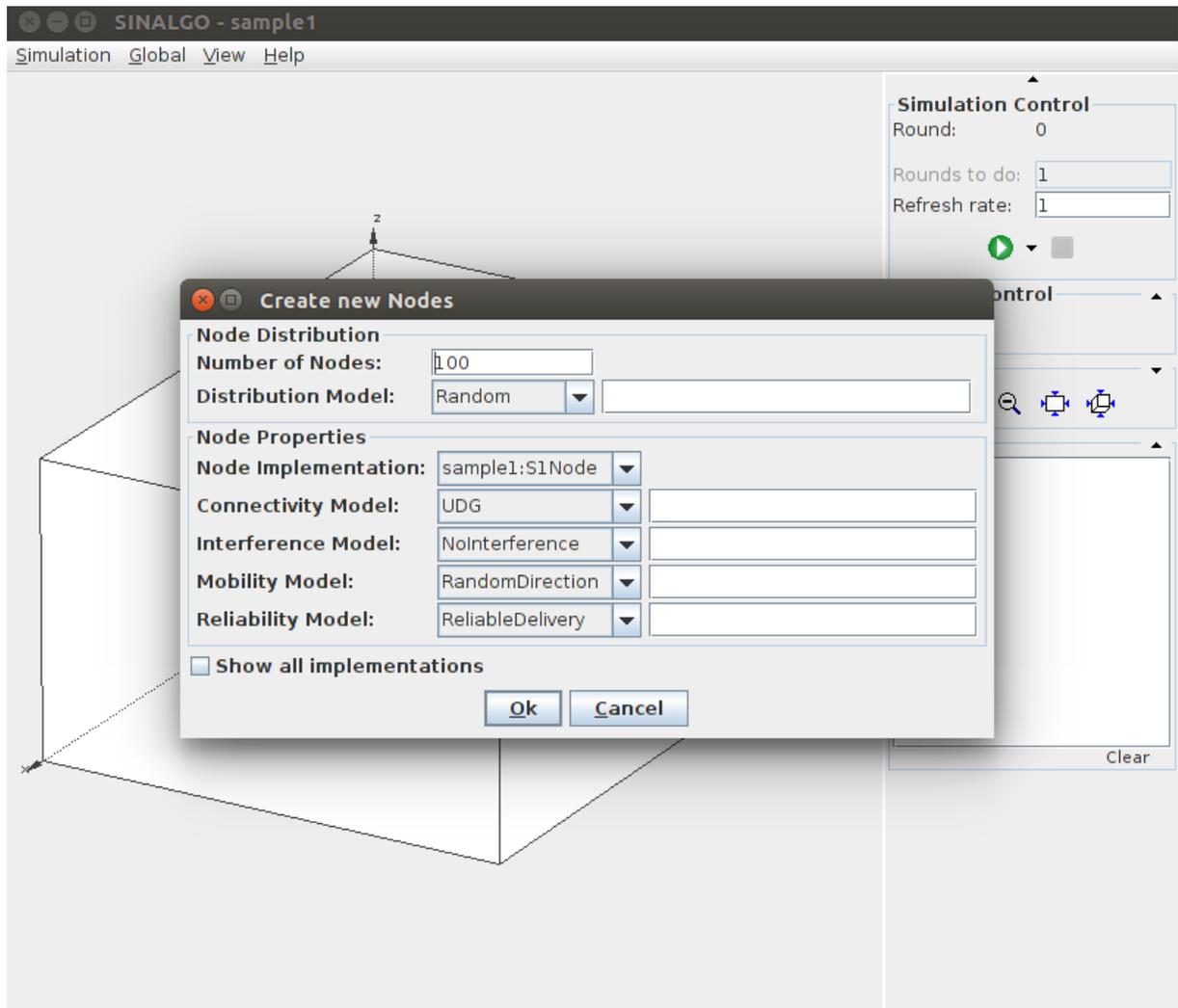


Figura 10 – Interface gráfica do Sinalgo ((DCG), 2016)

Apesar das vantagens oferecidas ao usuário, o simulador parece ter sido descontinuado, uma vez que seu domínio na internet não recebe atualizações desde 2008.

2.6.3 NS-3

O NS-3 é um simulador de redes baseado em eventos discretos, desenvolvido especialmente para pesquisa e uso educacional. Teve seu projeto iniciado no ano de 2006 com código fonte aberto e disponível.

A ferramenta possui suporte para Windows, Linux e Mac OS X, e atualmente encontra-se na versão 3.23. Possui documentação bem elaborada, porém exige do usuário conhecimento de linguagens de programação, no caso C++ ou Python, para que possa desenvolver seus modelos. É desenvolvido em C++, com código fonte aberto, o que facilita para pesquisadores que queiram contribuir com o projeto e também sua validação (ISI, 2016).

Desse modo, conta com contribuições da comunidade acadêmica para a correção de

erros e compartilhamento dos resultados obtidos com a simulação. Sua instalação é difícil, o que é amenizado devido a existência do manual do usuário. Possui uma quantidade razoável de exemplos, porém com modelos muito parecidos em seu conteúdo, o que pode limitar novas implementações.

Sua infraestrutura de software estimula o desenvolvimento de modelos de simulação realistas, podendo permitir que o simulador seja utilizado como um emulador de rede em tempo real. Além disso, possibilita com que protocolos de redes utilizados no mundo real sejam modelados e simulados, apresentando resultados sobre esta implementação ao usuário e viabiliza análises de otimização.

Utiliza como ferramenta de apoio para a visualização de resultados outro *software* denominado Wireshark e uma biblioteca denominada Tcpdump e não possui interface gráfica, fazendo com que as simulações e avaliações de resultados sejam feitas via terminal, algumas vezes trazendo dificuldades ao usuário iniciante.

2.7 Comparação entre os simuladores de redes

Com as informações obtidas no estudo de simuladores de redes pode-se elaborar uma tabela comparativa entre eles (Tabela 1). Nela são contrastadas características como a linguagem de programação em que os simuladores estão implementados, se existe ou não documentação de instalação e uso. Caso exista, quão bem elaboradas elas estão, o tipo de ambiente de execução de cada um deles, sendo gráfico ou textual, e observações importantes sobre cada ferramenta.

Tabela 1 – Comparativo entre os simuladores de redes

Simulador / Característica	Linguagem de implementação	Documentação	Tipo de ambiente de execução	Observações
Omnet++	C++	Código aberto e bem documentado	Gráfico	Possui difícil instalação
Sinalgo	Java	Código aberto e documentado	Gráfico	Projeto parado
NS-3	C++	Código aberto e bem documentado	Textual	Difícil instalação e uso, porém é a ferramenta mais completa e próxima de ambientes reais

Por ser a ferramenta mais completa, estar em constante desenvolvimento e aproximar-se de ambientes reais, o NS-3 será utilizado na validação deste projeto por meio de estudo de casos sobre redes de computadores.

2.8 Métricas de Desempenho

Com o objetivo de apresentar os dados que podem ser obtidos a partir da simulação, serão descritas nesta seção métricas de desempenho importantes para o desenvolvimento do Yasc. Dentre elas tem-se:

- **Taxa média de chegada:** Consiste na medida da quantidade de clientes que chegam ao sistema ao longo de um determinado período de tempo (MACDOUGALL, 1989).
- **Utilização do servidor:** Esta métrica representa a quantidade de tempo que o servidor permaneceu ocupado ao longo do funcionamento do sistema, sendo que taxas elevadas podem representar sobrecarga. (MACDOUGALL, 1989).
- **Tempo de atendimento médio:** Representa o tempo médio que cada cliente passa no servidor recebendo determinado serviço.
- **Taxa média de saída:** Representa a quantidade de clientes saindo do sistema de filas, sendo que uma elevada taxa pode significar grande eficiência dos servidores.
- **Tempo simulado:** Esta medida representa o intervalo de tempo que ocorre desde o momento de início do procedimento de simulação até o seu final.
- **Eficiência:** A eficiência resume a relação entre os resultados que puderam ser obtidos com uma simulação e os recursos que foram disponibilizados para que esta ocorresse. Desta forma, através dela pode-se verificar como foi o comportamento de sistema com relação às tarefas que lhe foram submetidas, ou seja, se este foi capaz de realizá-las em um tempo conveniente, ou se seria necessário maior quantidade de recursos para que tais tarefas fossem realizadas de maneira mais satisfatória.
- **Ociosidade:** A ociosidade consiste no período de tempo em que um sistema ou um de seus recursos ficou sem uso. Para que seja medida, são utilizados intervalos de tempo em que o recurso, apesar de ativo, não estava processando ou executando, mas apenas aguardando a chegada de algum tipo de tarefa a ser executada.

2.9 Considerações finais

A análise de desempenho é uma técnica fundamental para a avaliação de sistemas, podendo ser realizada por meio de aferição ou por solução de modelos. Uma das formas de solução de modelos é a simulação, um método que vem sendo amplamente utilizado devido ao seu baixo custo, menor dificuldade de implementação, além de não necessitar de pausas do ambiente real para testes e nem danificá-lo para isto.

Deste modo, um modelo que represente o sistema real deve ser criado com a finalidade de ser estimulado ao longo da simulação. Tal modelo pode ter sua constituição baseada em redes de filas.

Junto a isto, o crescente número de usuários leigos, a falta de flexibilidade, dificuldade do desenvolvimento de simuladores e por fim a necessidade da redução do tempo tanto da técnica de solução de modelos e simulação quanto de validação de softwares, tem feito com que também

aumente o interesse em simuladores baseados em padrões (*templates*). No entanto, ainda são poucas as ferramentas que permitem esta prática, e quando existem essas têm restrições, não possuem boa documentação e são de difícil acesso para os usuários.

3 Trabalho Desenvolvido

3.1 Introdução

Neste capítulo são apresentadas as atividades realizadas para a construção do gerador de simuladores para diferentes contextos, o Yasc. Na seção 3.2 é apresentada uma visão geral do gerador de simuladores, quais os módulos que o compõem, como eles interagem e qual ou quais suas funções. Na seção 3.3 é explicado o desenvolvimento da interface gráfica em que o usuário especifica como deve ser o simulador que está criando. Na seção 3.4 é descrito como são tratados os dados fornecidos pelo usuário para a geração da interface do simulador. Por fim, nas seções 3.5 e 3.6 são apresentadas a implementação da interface icônica do simulador gerado e do motor de simulação utilizado nele.

3.2 Visão geral sobre a arquitetura do Yasc

Desenvolvido em linguagem de programação Java, devido a sua portabilidade, o Yasc tem sua arquitetura baseada em módulos, sendo que cada um deles possui uma característica fundamental para o funcionamento da ferramenta. Na Figura 11 apresenta-se o diagrama conceitual desta ferramenta. Nela, pode-se perceber que os módulos são divididos entre os que pertencem à geração do simulador, como a interface gráfica para a descrição dos elementos, a interface gráfica para descrição do comportamento de cada elemento, o gerador e o interpretador. E também os módulos que representam o produto final, ou seja, o simulador criado, dentre os quais estão o motor de simulação e a interface icônica. Cada um deles terá sua funcionalidade descrita a seguir:

- **Interface gráfica para descrição de elementos e descrição comportamental:** Apesar de estarem divididas no diagrama conceitual (Figura 11), tais interfaces funcionam em conjunto. Por meio delas o usuário realiza a especificação na qual descreve como espera que seja o simulador que está criando.
- **Gerador:** Módulo em que os parâmetros fornecidos pelo usuário, por meio das interfaces gráficas, são utilizados para a geração de uma biblioteca cujos dados serão utilizados no produto final. Nesta biblioteca estão inclusos um arquivo de texto, em que são descritos os objetos do simulador e seus respectivos comportamentos, e as imagens que os representam.
- **Interpretador:** Módulo que realiza a conexão entre o Yasc e seu produto final. Ele recebe o arquivo de configuração da biblioteca criada pelo gerador, realiza sua leitura e a partir

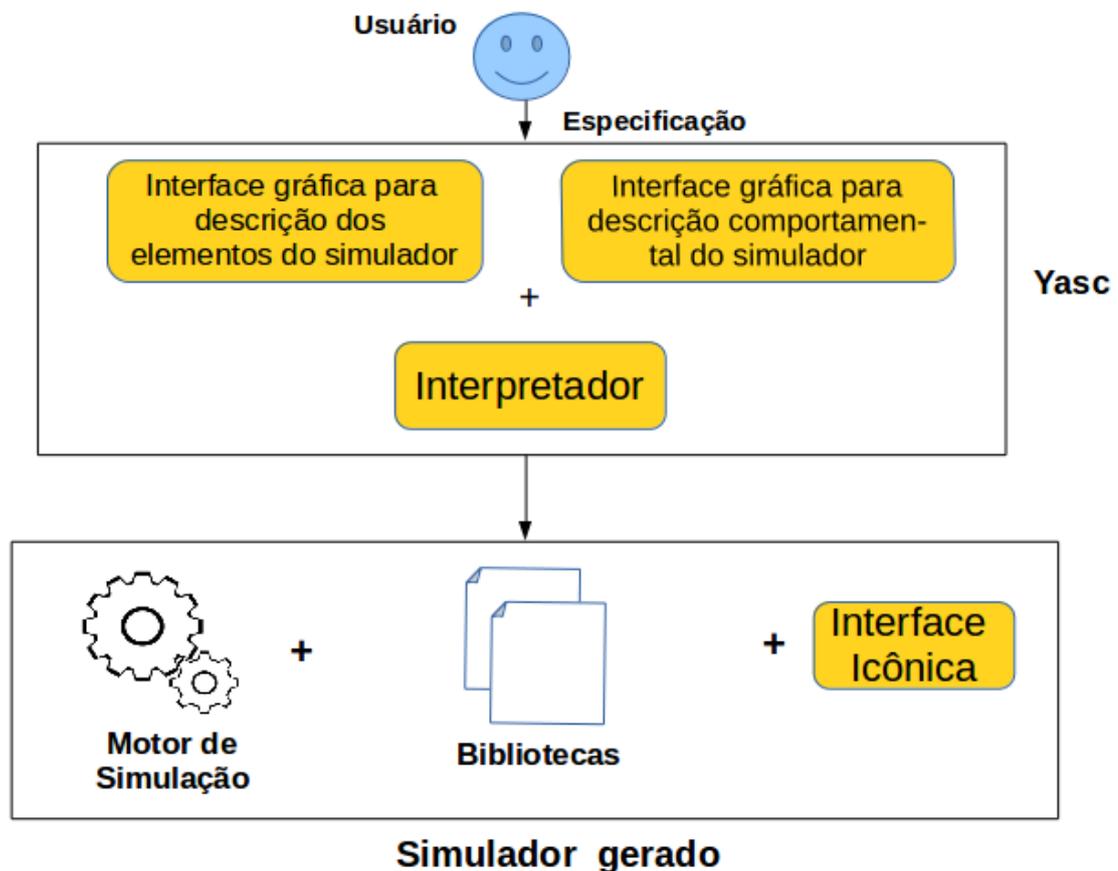


Figura 11 – Diagrama conceitual da ferramenta para um simulador

dos parâmetros obtidos junto com as imagens também pertencentes à biblioteca cria a interface icônica para o simulador e realiza seu acoplamento ao motor de simulação.

- **Motor de simulação:** É o responsável pela simulação propriamente dita. Nele estão implementadas o funcionamento das redes de filas básicas e dos demais objetos padrões dentre os quais o usuário pode optar para descrever seus objetos ao longo da especificação do simulador que está sendo criado. Desta forma, com o simulador já pronto, o motor é o responsável por utilizar os parâmetros descritos na interface icônica e a partir da simulação apresentar resultados significativos ao usuário.
- **Interface icônica:** Consiste na interface gráfica do simulador gerado. Nela, por meio de ícones, o usuário pode representar e configurar seu modelo. Ao final da simulação também é responsável por exibir as métricas de maneira simples e intuitiva.

Nas próximas seções serão detalhados o funcionamento de cada um dos módulos como também sua implementação.

3.3 Construção das interfaces gráficas para descrição dos objetos que serão disponibilizados no simulador e seu comportamento

O início do desenvolvimento do Yasc foi determinado pela construção de um componente de entrada capaz de armazenar todas as informações relativas ao contexto do simulador a ser gerado. Para isto, foi desenvolvida uma interface gráfica na qual o usuário deve descrever todas as características de funcionamento que deseja para sua ferramenta. Tal descrição pode ser feita em três etapas:

- **Etapa 1 - Criação dos objetos:** Nesta etapa, o usuário tem a possibilidade de criar todos os objetos pertencentes a simulação que serão disponibilizados na ferramenta gerada, como por exemplo, máquinas e canais de comunicação para simuladores de redes ou grades computacionais, portas lógicas para simuladores de circuitos sequenciais, filas e caixas para simuladores genéricos como os de bancos e mercados, entre outros. Para criar um objeto, em um primeiro momento deve-se fornecer a este um nome e uma imagem de representação, componentes solicitados conforme na Figura 12. Em um segundo momento, deve-se informar qual será seu comportamento, podendo este ser como o de uma fila básica, de forma instantânea ou durativa, regida por uma função de transferência, informados conforme mostrado na Figura 13. Em um último momento, caso o objeto possua comportamento de um tipo básico de fila, serão solicitadas informações sobre o tipo de fila desejado e os parâmetros pertencentes a esta, como pode-se observar nas Figuras 14 e 15 respectivamente. Caso possua função de transferência, será solicitada qual a função que rege seu funcionamento que poderá ser criada por meio da interface gráfica que aparece na Figura 16.

Realizadas tais subetapas de criação, o usuário será questionado sobre a necessidade de criação de outros objetos, caso responda positivamente, deverá repetir o passo a passo, podendo criar a quantidade de objetos necessária, sem limites.

- **Etapa 2 - Descrição de falhas e restrições de capacidade:** A segunda etapa de desenvolvimento do simulador consiste em selecionar entre a possibilidade ou não de ocorrerem falhas ao longo da simulação. Em caso afirmativo, deve-se escolher também se os centros de serviço utilizados no modelo deverão possuir restrições de capacidade, possibilitando a perda de eventos devido a lotação das filas ou não. A forma de inclusão destes parâmetros pode ser observada na Figura 17.
- **Etapa 3 - Escolha das métricas:** Na terceira e última etapa o usuário deverá informar qual o significado dos eventos em sua simulação, ou seja, o que os eventos representarão, podendo ser por exemplo pacotes, clientes e tarefas. Além disso, também deverá selecionar

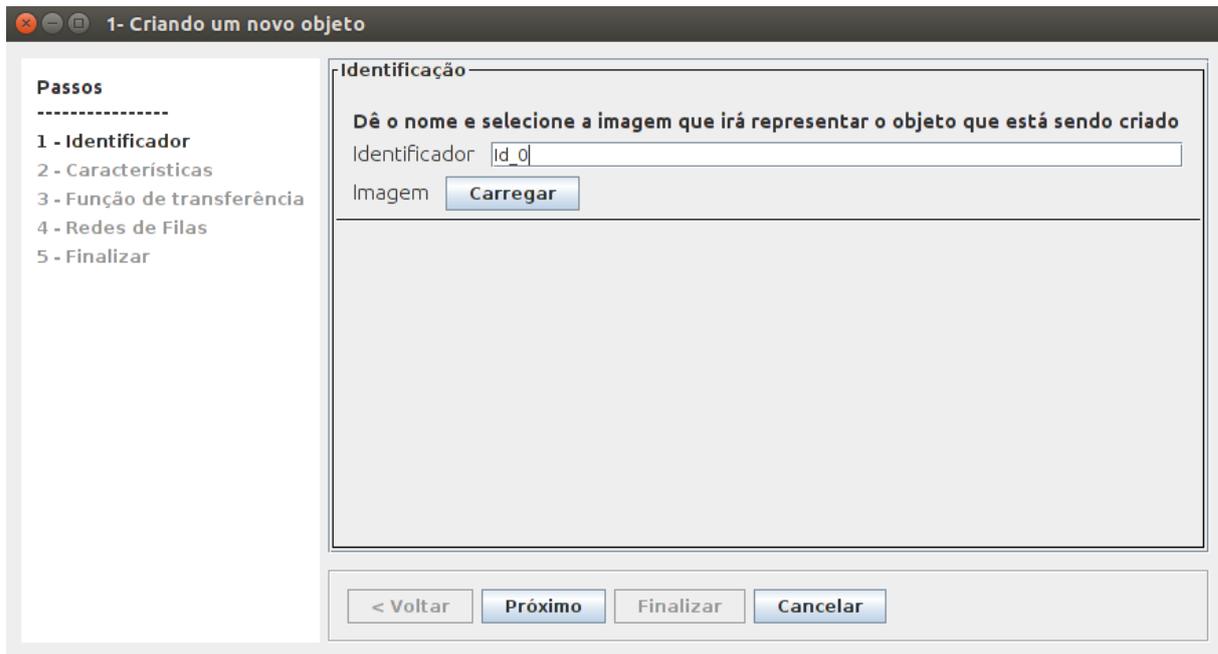


Figura 12 – Interface gráfica de criação solicitando o identificador e a imagem que representarão o objeto

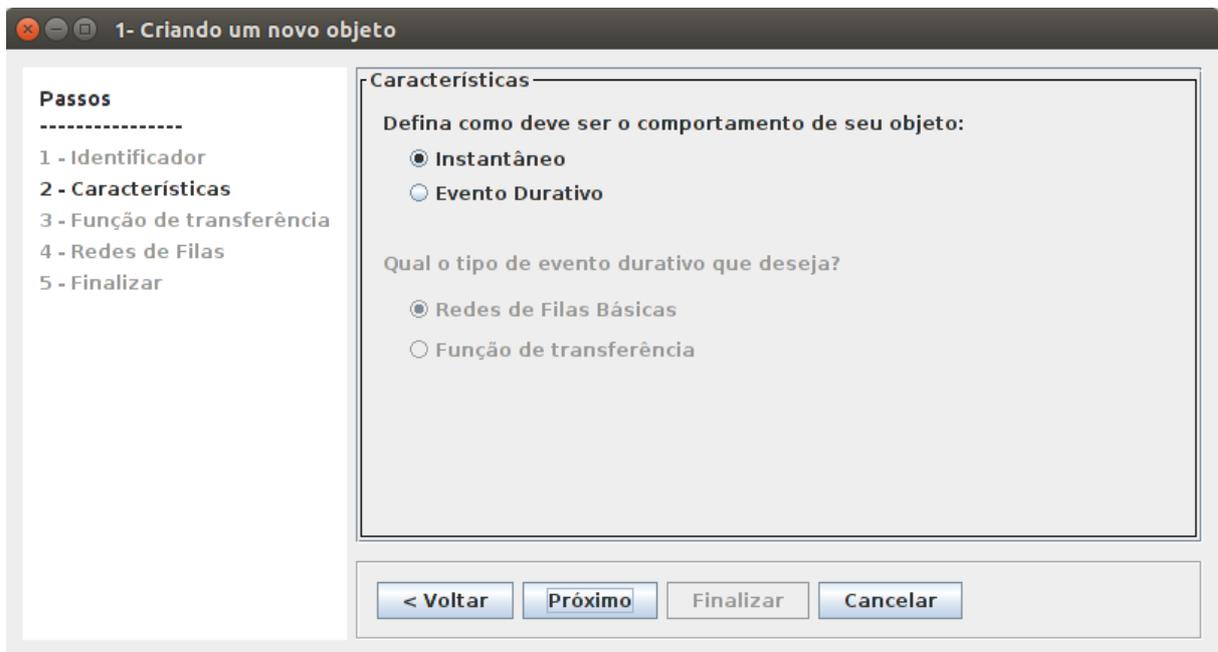


Figura 13 – Interface gráfica de criação solicitando a forma de comportamento do objeto para a simulação

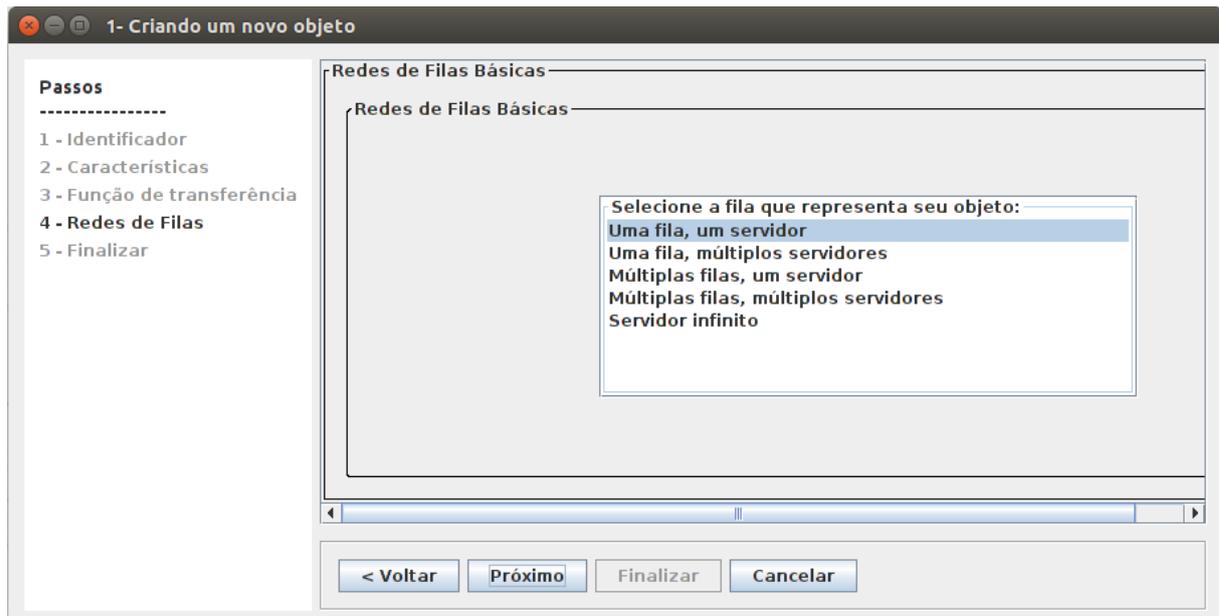


Figura 14 – Interface gráfica de criação solicitando o tipo de fila básica que irá reger o funcionamento do objeto criado

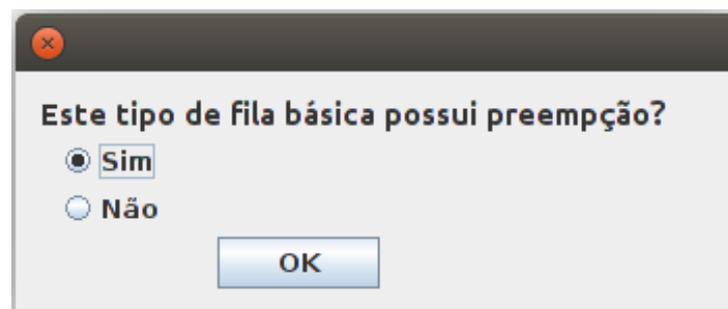


Figura 15 – Interface gráfica de criação solicitando sobre a presença ou não de um parâmetro no tipo básico de fila

quais as métricas que deseja ver como resultados para a simulação, podendo ser elas perda, tempo médio de espera em fila e tempo médio de atendimento dos eventos. Para isto, deverá preencher um formulário como o que aparece na Figura 18.

3.4 Módulos para geração e interpretação de bibliotecas

Ao fim da implementação da interface gráfica responsável pela entrada de dados foram criados dois módulos para tratamento dos dados recebidos, representados respectivamente por dois métodos implementados em Java.

O primeiro consiste no gerador que cria, a partir dos parâmetros recebidos do usuário, uma biblioteca de dados. Nela estão contidas as imagens disponibilizadas para os objetos e um arquivo texto de configuração, permitindo que o simulador gerado possa ser salvo e recarregado a qualquer momento após ter sido encerrado uma vez. O segundo consiste no interpretador,

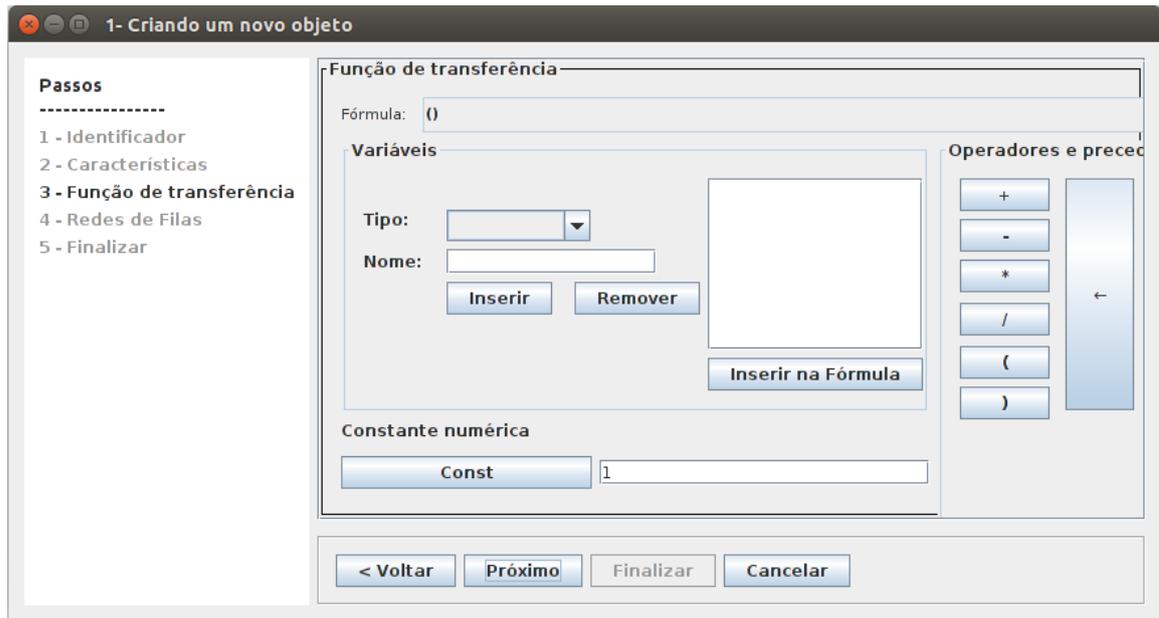


Figura 16 – Interface gráfica para o desenvolvimento de uma função de transferência que rege o funcionamento do objeto

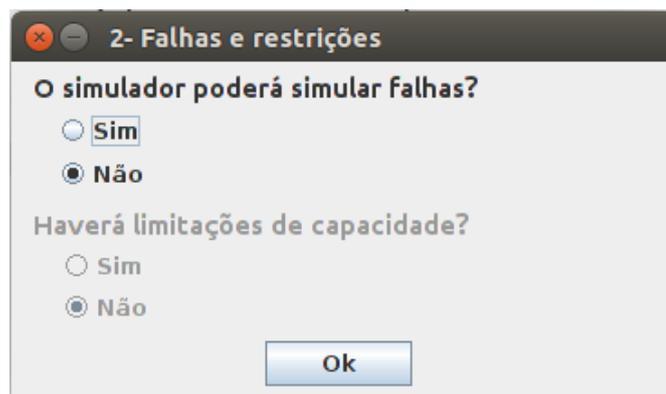


Figura 17 – Interface gráfica para descrever falhas e restrições de capacidade

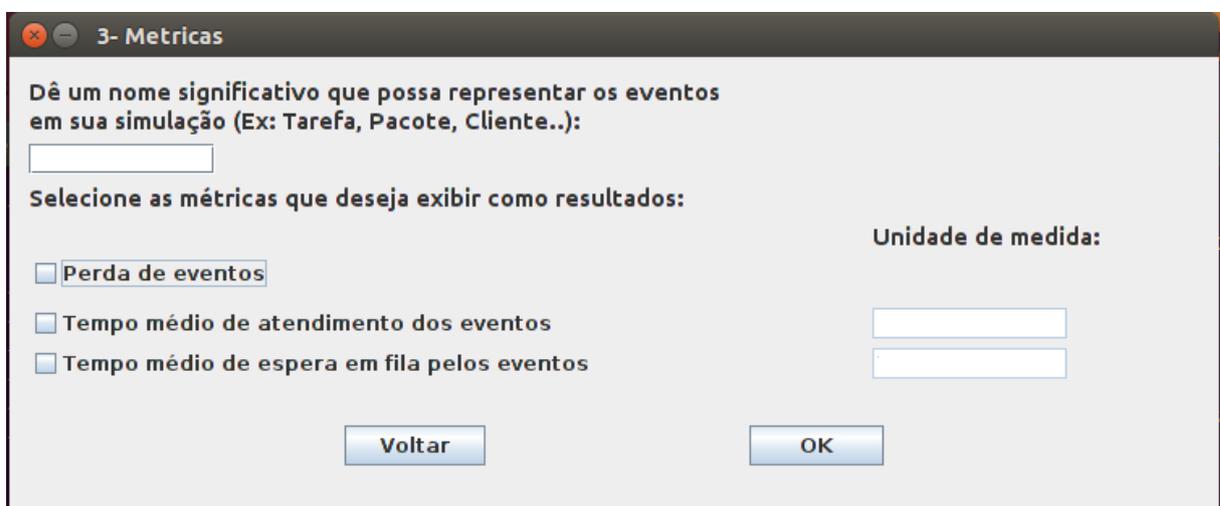


Figura 18 – Interface gráfica para escolha das métricas

módulo que faz a leitura de tal arquivo e carregamento das imagens, realizando a abertura de um simulador construído previamente.

Para o desenvolvimento da biblioteca foi elaborado um padrão de arquivo de configuração conforme o que segue na Figura 19.

```
1 Link simulatorlibrary/LinkComunicacao.png One_queue_one_server
a Preemption - -
a - Mbits/s -
3 Router simulatorlibrary/Roteador.png ; 0
1 Switch - One_queue_one_server
a Preemption - -
a - Mbits/s -
3 Node simulatorlibrary/No.png ; 0
4 Failure
5 Capacity
6 packets
7 Loss
7 Service seconds
7 Queue seconds
```

Figura 19 – Exemplo de arquivo de configuração para um simulador gerado

No arquivo de configuração, a primeira coluna indica o tipo do parâmetro que será descrito:

- "1": representa um objeto do tipo fila básica.
- "2": representa um objeto durativo, com função de transferência.
- "3": representa um objeto instantâneo.
- "4": representa que haverá simulação de falhas no simulador.
- "5": representa que além das falhas, também haverá restrição de capacidade das filas na simulação.
- "6": indica qual é o nome atribuído a um evento que está presente na simulação.
- "7": representa quais as métricas serão disponibilizadas ao usuário ao término da ação da ferramenta sendo estas seguidas por sua unidade de medida.
- "a": define parâmetros referentes ao numeral precedido mais próximo.

Além disso, os demais parâmetros descrevem outros dados indispensáveis para a criação do simulador. São eles:

- O nome dado ao objeto, que geralmente aparece logo após os números "1", "2" e "3" da primeira coluna.
- O endereço do diretório em que estão localizadas as imagens disponibilizadas para representar os objetos, os quais são apresentados logo após o nome do objeto.
- O tipo da fila básica que deverá estar presente caso o objeto seja do tipo "1", apresentado logo após o endereço do diretório em que se encontra a imagem que representa o objeto.
- A função de transferência do objeto e quantas variáveis esta possui que são respectivamente o penúltimo e último parâmetros da linha, caso a primeira coluna seja do tipo "2" ou "3".

O arquivo texto é gravado com as imagens em um diretório denominado *simulatorlibrary*. Este será aberto e seu conteúdo reconhecido e interpretado pelo módulo de leitura, possibilitando a execução da interface gráfica criada, que será discutida na seção 3.5), a qual funciona juntamente ao motor, que será discutido na seção 3.6, completando assim, o simulador criado.

3.5 Interface Icônica

Após a interpretação do arquivo de configuração criado durante a fase de coleta de dados, a interface icônica do simulador é aberta, permitindo ao usuário a modelagem de seu sistema e uma posterior solução de modelos através da simulação.

O desenvolvimento desta interface ocorreu com base no projeto do Grupo de Sistemas Paralelos e Distribuídos, denominado iSPD (*iconic Simulator of Parallel and Distributed Systems*) (MENEZES et al., 2012) (FURLANETTO et al., 2015c), que é um simulador de grades computacionais. Desta maneira, a interface original estática, a princípio idealizada apenas para realizar a modelagem de ambientes de grades computacionais foi reorganizada e reprogramada para tornar-se flexível e permitir sua adaptação aos mais diferentes tipos de simuladores que podem ser desenvolvidos pelo Yasc. Na Figura 20 pode-se ver um exemplo de como ela funciona.

Na parte identificada com o símbolo "I" são exibidos os objetos criados na primeira etapa da construção do simulador. Na parte identificada com o símbolo "II" são disponibilizados os parâmetros pertencentes aos ícones, aos quais o usuário deve fornecer valores referentes ao seu sistema, como por exemplo atribuir como capacidade do sistema dois mil clientes. Dentre eles estão os parâmetros relativos às falhas e restrições de capacidade. Na área quadriculada, também chamada de área de desenho, o usuário utiliza os ícones oferecidos no simulador, que aparecem em "I", para conceber o modelo que reproduz o ambiente real a ser simulado.

A interface icônica foi desenvolvida com a mesma finalidade que o Yasc, ou seja, possibilitar ao usuário com que faça a representação de seu sistema por meio de um modelo de maneira simples, sem necessidade de conhecimentos sobre linguagens de programação.

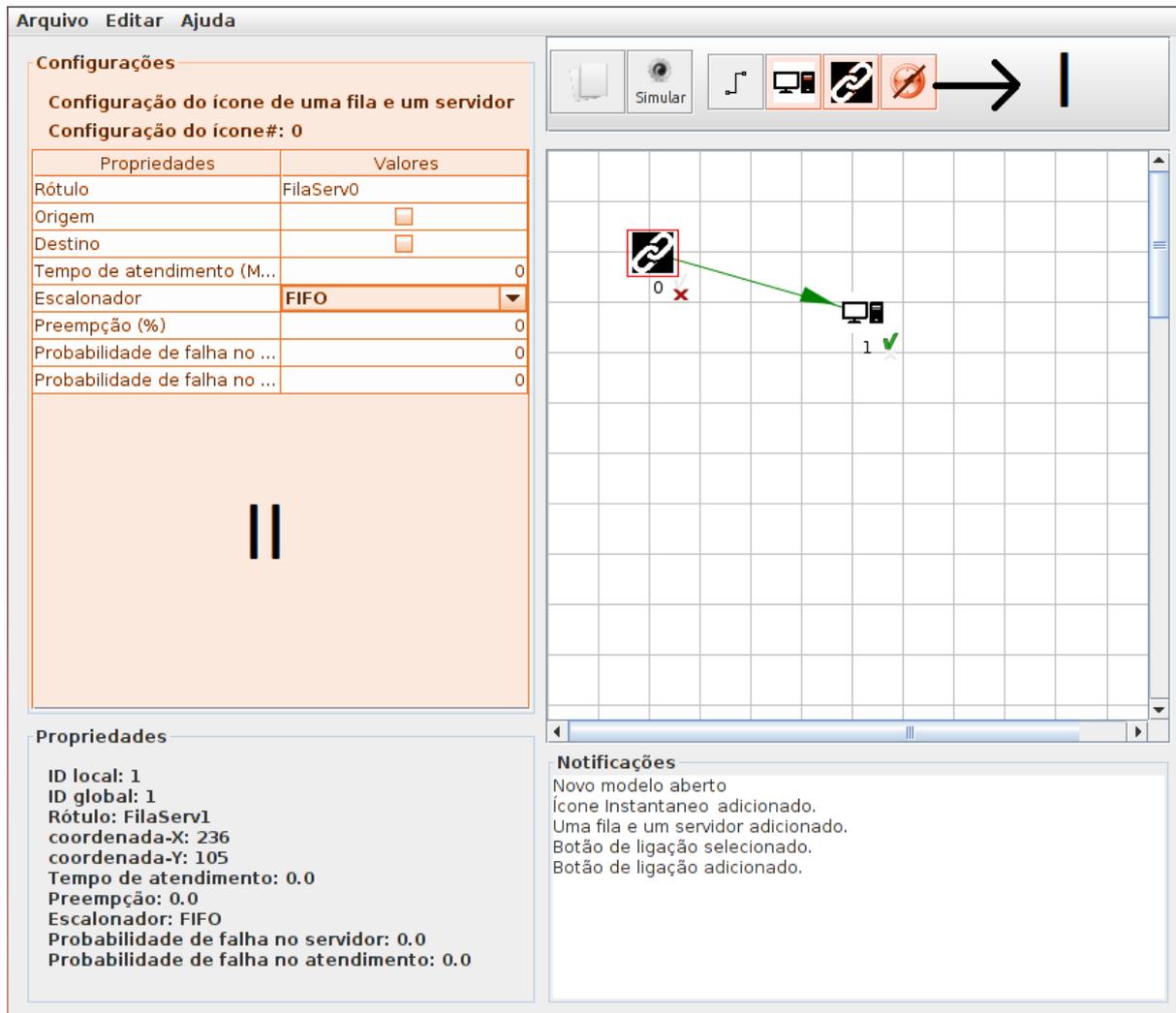


Figura 20 – Interface icônica

3.6 Motor de simulação

Com a conclusão da interface icônica pertencente ao simulador construído, foi desenvolvido o motor de simulação o qual também tomou como base o componente análogo pertencente ao iSPD.

O motor é o módulo em que ocorre a simulação propriamente dita. Para seu funcionamento, ele recebe da biblioteca, por meio do Yasc, as especificações do comportamento de cada um dos componentes com a finalidade de que, assim que um modelo descrito for repassado ao seu módulo de interpretação de modelos, ele possa verificar o tipo de cada objeto e conseguir processá-lo de maneira correta.

Terminado o preparo para o processamento, no momento em que o usuário completa sua especificação e seleciona a opção de simulação, o motor recebe um modelo icônico no formato XML, o qual é validado e interpretado pelo interpretador de modelos internos. Interpretador este pertencente ao iSPD, o qual foi adaptado para funcionar no simulador criado pelo Yasc e

teve sua função mantida que é transformar o modelo icônico no formato XML em um modelo simulável, ou seja, um modelo cujos parâmetros são significativos ao motor, permitindo que a simulação possa ser realizada retornando resultados significativos ao usuário. Tal esquema pode ser observado para melhor compreensão na Figura 21.

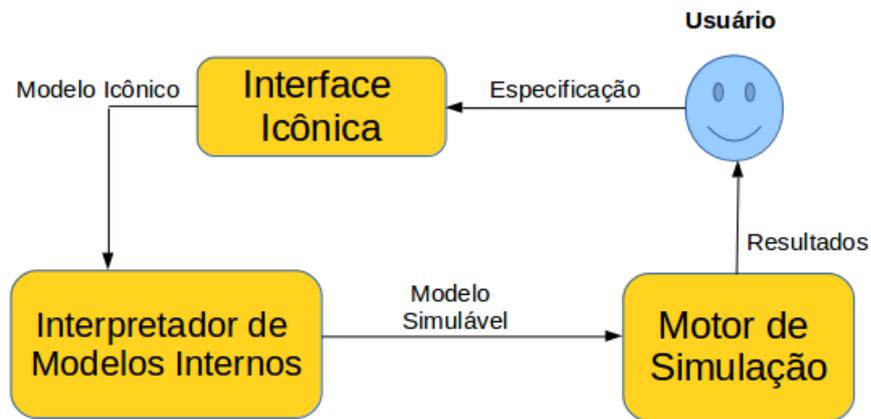


Figura 21 – Diagrama conceitual do motor de simulação - Adaptado de (MENEZES et al., 2012)

Deve-se destacar também que o módulo de métricas, responsável por efetuar cálculos a respeito destas e apresentá-las no formato de resultados para o usuário também está incluso no motor.

Com o objetivo de reutilizar o componente, duas importantes alterações foram necessárias no motor de simulação: a alteração da implementação de classes relativas ao tratamento das filas e a mudança da maneira de validar e interpretar os modelos XML gerados pela interface icônica.

A primeira alteração significativa pode ser observada entre os diagramas de classes nas Figuras 22 e 23. Ela foi feita com o intuito de suportar os diferentes tipos de filas adicionados. Desta forma, deixaram de existir as classes "CS_Comunicação" e "CS_Processamento", que dividiam tais filas, devido ao processamento distinto que era fornecido aos seus eventos, passando a existir apenas uma classe pai denominada "Centro de Serviço", que torna homogêneo tal processamento. Assim, permitiu-se que todos os modelos básicos de filas pudessem ser implementados e utilizados para a geração de simuladores, adicionando também dois novos centros de serviço que serão utilizados em trabalhos futuros, os quais permitem processamento instantâneo e durativo regido por função de transferência aos eventos que lhes são submetidos.

Enquanto isso, a segunda alteração teve como intuito permitir que a validação e interpretação de modelos XML fossem possíveis para qualquer tipo de simulador gerado pelo Yasc. Dessa maneira, foi construído uma definição genérica em *XML Schema*, documento que descreve como deve ser a estrutura de um XML, substituindo a maneira que estava implementada previamente, em que a validação e interpretação ocorria apenas para um tipo de arquivos padrão e era feita por meio de um DTD (*Document Type Definition*). Com isto, o motor tornou-se apto para realizar a simulação de qualquer modelo gerado pelo Yasc.

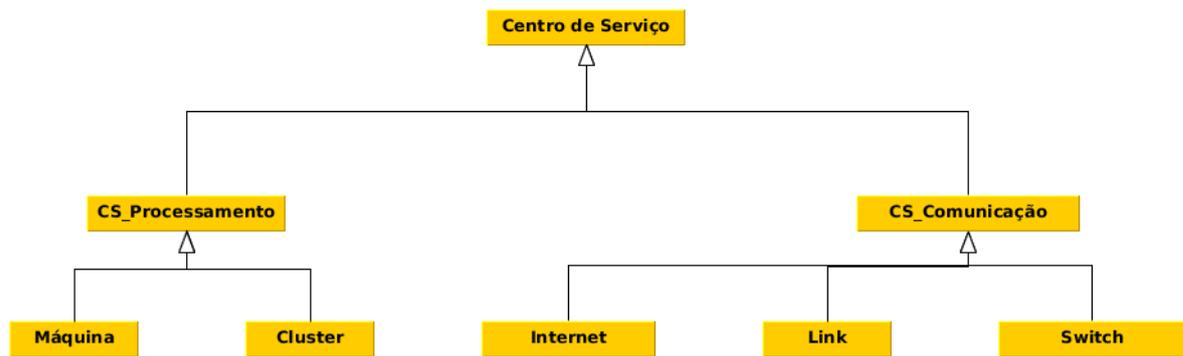


Figura 22 – Diagrama de classes representando as filas do motor de simulação do iSPD

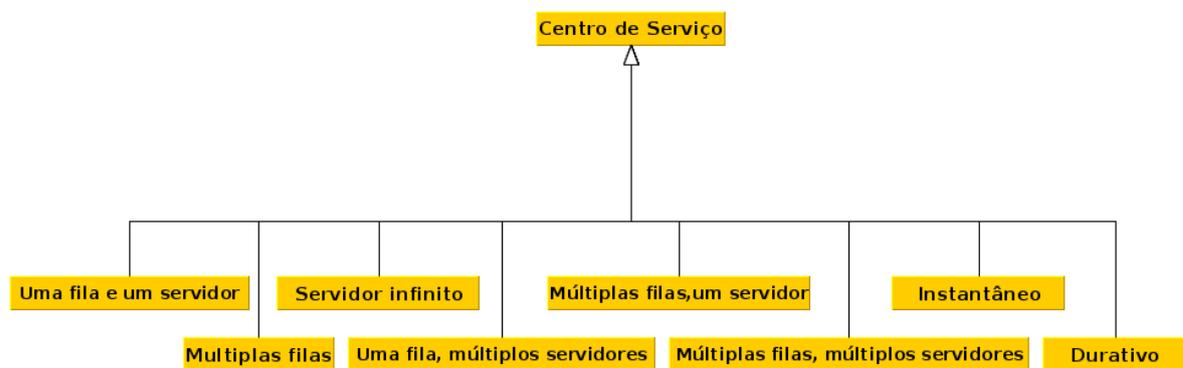


Figura 23 – Diagrama de classes representando as filas do motor de simulação da ferramenta construída (Yasc)

3.7 Considerações Finais

Neste capítulo foi descrita a arquitetura do Yasc. Por meio dela, pode-se observar que o gerador de simuladores é dividido em módulos, dentre os quais estão: a interface gráfica para descrição de elementos e descrição comportamental do simulador a ser criado, o gerador e interpretador de bibliotecas de configuração, a interface icônica gerada e o motor de simulação.

Além disso, foi apresentado como ocorreu o desenvolvimento de cada um de tais módulos e como foi feita adaptação do motor de simulação do iSPD, *software* do Grupo de Sistemas Paralelos e Distribuídos, para que este pudesse realizar a simulação de modelos de maneira genérica em todos os simuladores criados a partir do Yasc.

4 Testes e validações

4.1 Introdução

Neste capítulo são descritos os testes sobre o Yasc. Após a implementação das filas básicas ser validada por meio de testes, como o que pode ser visto no Apêndice D, em que comparamos o modelo de fila básica com uma fila e um servidor presente em (MACDOUGALL, 1989) com a implementação disponível no motor do Yasc, obtendo resultados satisfatórios. Foram executados testes com dois objetivos distintos. O primeiro consiste em analisar a eficácia de uso da ferramenta considerando críticas e sugestões para obter-se melhorias, como será exposto na seção 4.2. No segundo, busca-se validar o simulador gerado comparando-o com outro simulador, o NS-3, como será apresentado na seção 4.3. Serão feitas algumas considerações sobre os testes na seção 4.4.

4.2 Teste de usabilidade e o estudo de casos na área de redes de computadores

O teste de usabilidade permite avaliar sistemas interativos por meio da observação de usuários reais realizando tarefas pré-definidas. Com isto, avalia o grau de efetividade da interação e determina o nível de satisfação com que o sistema apoia as necessidades dos usuários (PRESSMAN, 2011). Visando qualificar o grau de eficácia oferecido pelo Yasc, submeteu-se a ferramenta a tal teste, com estudo de casos na área de redes de computadores.

4.2.1 Procedimentos para a realização do teste

O teste realizado para validação do Yasc foi dividido em etapas. Na primeira etapa, foi realizado o cálculo do tamanho do espaço amostral para o teste e definido o perfil dos usuários que participariam dele. Na segunda etapa, foi elaborado um plano de testes, no qual foi definido que o avaliador deveria fazer uso da ferramenta de maneira a criar um simulador de redes de computadores. Este simulador gerado foi usado posteriormente para reproduzir um modelo específico com nós terminais, canais de comunicação e roteadores, e também um questionário com perguntas sobre a eficácia do *software* que estava sendo testado. Procedimento que além de mostrar como criar um exemplo completo de um ambiente de redes, permite também que possa-se obter noção de funcionamento do programa para a criação de uma nova ferramenta de simulação. Na terceira etapa, após já ter feito uso de todas as funcionalidades da ferramenta, o usuário respondeu o questionário elaborado na etapa anterior e disponibilizou o simulador gerado para testes futuros. Na quarta etapa, os dados obtidos por meio do questionários foram

avaliados e os simuladores obtidos foram utilizados para realizar testes com dois diferentes modelos. Todas as etapas serão detalhadas a seguir.

Cálculo do espaço amostral e definição do perfil dos avaliadores

Em uma etapa preliminar ao início da execução dos testes, foi calculado o espaço amostral, ou seja, a quantidade de usuários necessária para a validação. Assim, a partir da expressão matemática $n = (Z^2 * p * (1 - p)) / e^2$, retirada de (BUSSAB; MORETTIN, 2013), em que:

- n = tamanho da amostra
- Z = nível de confiança escolhido
- p = proporção do evento na população, em que $0 < p < 1$
- $(1-p)$ = complemento de p
- e = erro amostral

calculou-se tal quantidade. Para isto, utilizou-se erro amostral e nível de confiança iguais a 10% e p com o valor de 50% (valor utilizado quando não tem-se uma noção da proporção do evento), obtendo-se como resultado, que um total de pelo menos 68 usuários deveriam fazer uso da ferramenta com a finalidade de verificar sua usabilidade.

Além disso, determinou-se que, como a validação foi feita com um estudo de caso de redes de computadores, os avaliadores deviam ter conhecimentos mínimos sobre o assunto, ou seja, o funcionamento de tais redes.

Elaboração do plano de testes e questionário

Conhecendo-se o tamanho do espaço amostral e os tipos de conhecimento que os usuários deveriam ter, foi elaborado o plano de testes e também o questionário a ser respondido após o final do procedimento.

Com o plano de testes, cuja descrição completa encontra-se no Apêndice A, teve início o estudo de caso, utilizando assim o Yasc para a geração de um simulador de redes de computadores e simulação de um modelo de rede específica.

Como últimas ações do cronograma de testes, cada usuário teve que responder a um questionário apresentado no Apêndice B, no qual relata como foi sua experiência, e em sequência entregar o simulador de redes criado. Esse simulador foi utilizado na comparação de resultados com o NS-3, conforme será apresentado na seção 4.3.

Execução dos testes

Ao término da elaboração do plano de testes, foi dado início a sua execução. Nela, os 68 avaliadores foram submetidos ao mesmo procedimento. Eles realizaram todas as etapas descritas, sendo estas responder ao questionário e entregar o simulador de redes junto com o modelo nele criado para procedimentos seguintes, descritos na seção 4.3.

4.2.2 Análise do teste de usabilidade

Após realizadas todas as avaliações programadas para o teste de usabilidade, os resultados obtidos foram analisados e serão apresentados nesta seção.

O questionário respondido por cada usuário foi composto por sete questões objetivas, nas quais cada resposta representava um grau de concordância com o que era afirmado, podendo este grau ser:

1. Discordo totalmente
2. Discordo em boa parte
3. Nem concordo nem discordo
4. Concordo em boa parte
5. Concordo integralmente

Ainda havia duas questões discursivas, em que o usuário respondia sobre os aspectos positivos e negativos do uso da ferramenta.

A partir das questões objetivas foram gerados gráficos para análise da avaliação, como pode ser visto nas Figuras 24, 25, 26, 27, 28, 29 e 30. Em todas elas pode-se observar qual foi o questionamento feito ao usuário, o qual aparece na parte de cima da figura, e as quantidades apresentadas para cada uma das cinco possíveis respostas.

Na figura 24, em que o questionamento feito foi "O Yasc é fácil de usar?", pode-se concluir que, apesar do Yasc ainda estar em sua primeira versão, obteve excelente qualificação com relação a sua facilidade de uso. Isto foi confirmado pelos 63.2% dos validadores que concordam em boa parte ou integralmente com a informação.

Na Figura 25, cujo questionamento era: "As informações na interface do Yasc estão bem organizadas?", 64.7% dos usuários responderam positivamente, concordando totalmente ou parcialmente com isso.

Na Figura 26, em que o questionamento feito foi: "A aparência das telas do Yasc é bastante clara?", 77.9% dos avaliadores confirmaram o que foi perguntado.

O Yasc é fácil de usar?

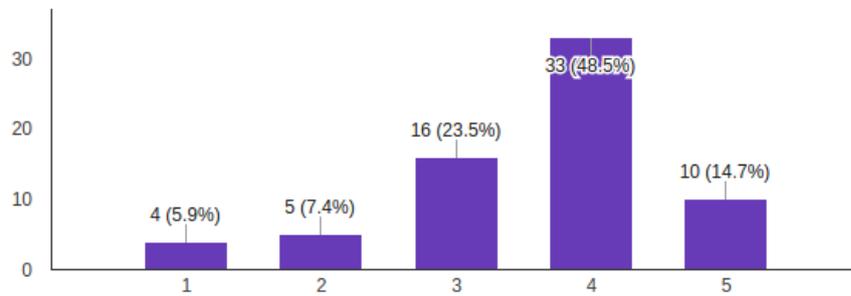


Figura 24 – Resultado obtido para a primeira questão objetiva apresentada no questionário

As informações na interface do Yasc estão bem organizadas?

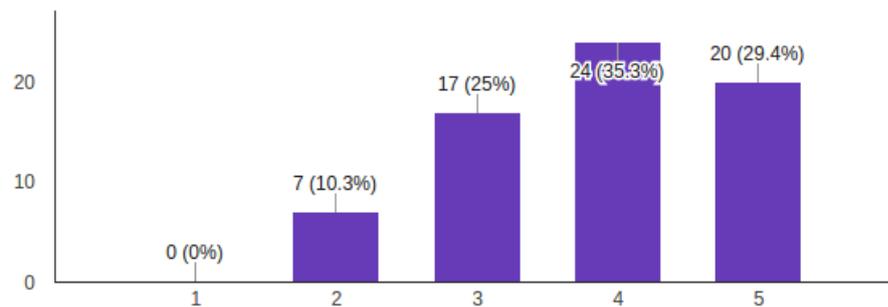


Figura 25 – Resultado obtido para a segunda questão objetiva apresentada no questionário

A aparência das telas do Yasc é bastante clara?

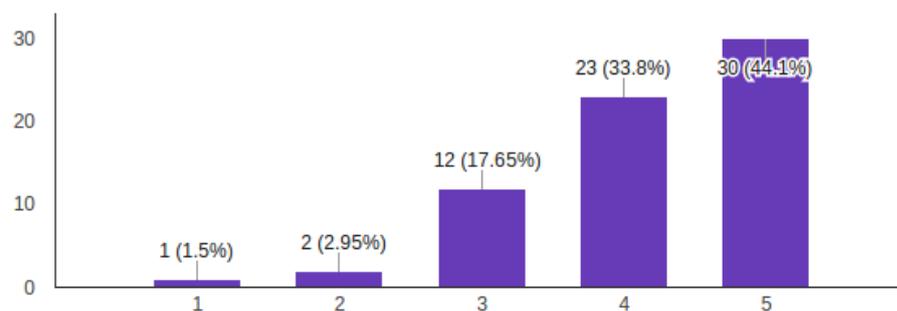


Figura 26 – Resultado obtido para a terceira questão objetiva apresentada no questionário

Na Figura 27, em que o questionamento foi: "A nomenclatura utilizada nas telas é fácil de compreender?", 64.7% dos usuários concordaram totalmente ou parcialmente com isso.

A nomenclatura utilizada nas telas (nome de comandos, títulos, campos, etc.) é fácil de compreender?

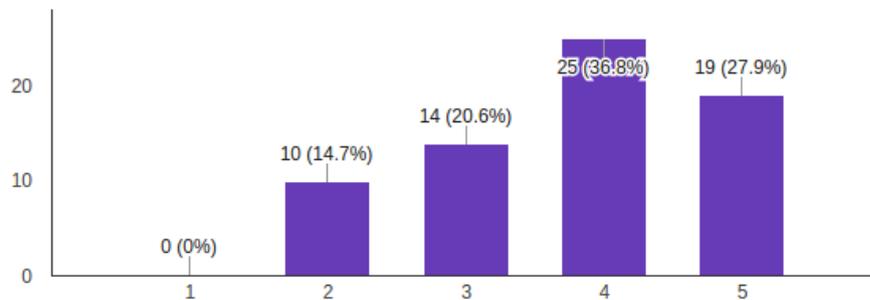


Figura 27 – Resultado obtido para a quarta questão objetiva apresentada no questionário

Encerrando as questões relacionadas com as interfaces gráficas do Yasc, na Figura 28, foi perguntado: "As mensagens do sistema são claras?". Questão que recebeu 63.2% das avaliações sendo positivas.

As mensagens do sistema são claras?

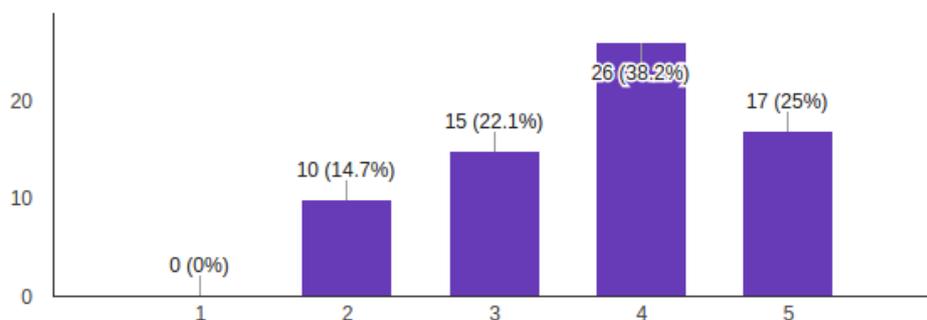


Figura 28 – Resultado obtido para a quinta questão objetiva apresentada no questionário

Desta forma, com as respostas das questões das Figuras 25, 26, 27 e 28, pode-se concluir que quanto a organização, clareza, nomenclatura e aparência, o Yasc foi eficaz e agradou os usuários.

Por fim, mais duas questões objetivas foram feitas. Na primeira, que aparece em 29, o enunciado foi: "No geral, a utilização do Yasc foi interessante?". Dentre as respostas, 88.4%

dos usuários confirmaram seu interesse pelo uso da ferramenta e responderam que concordam completamente ou parcialmente com o que foi perguntado.

No geral, a utilização do Yasc foi interessante?

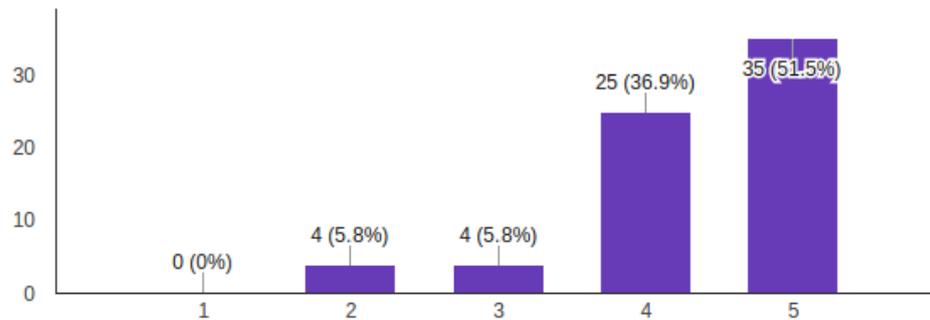


Figura 29 – Resultado obtido para a sexta questão objetiva apresentada no questionário

Na segunda, que aparece em 30, foi perguntado: "O Yasc é uma ferramenta interessante para geração de simuladores?", e 88.2% dos validadores responderam positivamente também concordando completamente ou parcialmente com a questão.

O Yasc é uma ferramenta interessante para geração de simuladores?

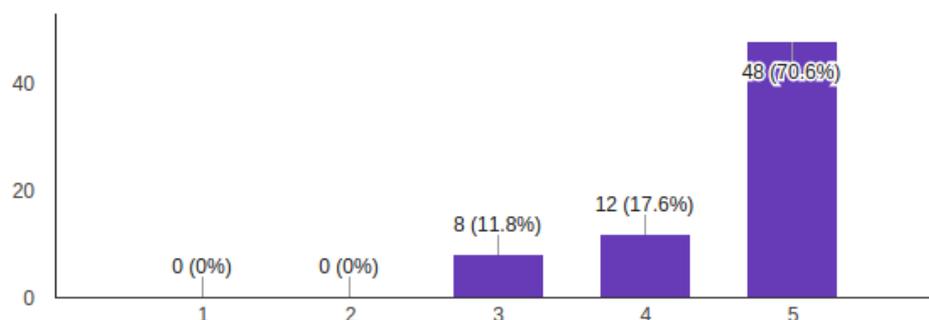


Figura 30 – Resultado obtido para a sétima questão objetiva apresentada no questionário

Com relação às questões discursivas, a primeira, "Aponte os pontos positivos que você encontrou ao utilizar o sistema", teve como propósito questionar os usuários sobre quais pontos positivos foram encontrados durante a utilização da ferramenta. Assim, diversas respostas foram obtidas, porém todas destacavam:

- A simplicidade do Yasc.

- O fato de suas interfaces gráfica e icônica serem intuitivas.
- A clareza nas informações.
- A portabilidade entre diferentes plataformas, incluindo Mac OS, Windows e Linux.
- A possibilidade de fornecer imagens para os ícones e facilidade de simulação.

Enquanto isso, na segunda, cujo enunciado era "Aponte os pontos negativos que você encontrou ao utilizar o sistema", o propósito era questionar os mesmos usuários sobre os pontos negativos encontrados. Dentre as respostas obtidas, estavam:

- Problemas na interface gráfica, como textos cortados ou que não tinham seu idioma modificado ao alterar-se a linguagem de inglês para português.
- A falta de uma opção para excluir objetos criados na etapa de geração do simulador e a necessidade de maiores explicações sobre o que cada opção proporciona ao ser selecionada.
- Sugestões de melhorias, como a adição de exemplos de simuladores ao Yasc e a confecção de um manual de uso.

Com base nos pontos negativos levantados pelos usuários, foram realizadas alterações na implementação da ferramenta, solucionando os problemas encontrados, principalmente os de interface gráfica, como ausência de maior detalhamento das opções disponíveis nas telas, e adicionando novas funcionalidades, como a possibilidade de exclusão de um objeto criado. Além disso, realizou-se o planejamento para que nas próximas versões do Yasc sejam disponibilizados, o manual confeccionado, encontrado no Apêndice C, e também exemplos para a criação de novos simuladores.

4.3 Validação e resultados produzidos pelos simuladores gerados

Finalizado o teste de usabilidade, obteve-se como produto 68 simuladores de redes de computadores gerados pelos usuários. Tais simuladores foram utilizados para dar continuidade ao procedimento de validação do Yasc, e foram empregados para realizar a simulação de modelos, também construídos em equivalência para outra ferramenta da área, o NS-3. Possibilitando deste modo, a extração e comparação de métricas.

Foram elaborados e testados dois modelos, discutidos nas seções 4.3.1 e 4.3.2.

4.3.1 Primeiro modelo simulado

O primeiro modelo, hipotético, é composto por quatro nós terminais que trocam pacotes entre si, ligando-se a três roteadores por meio de canais de comunicação, aos quais são atribuídas velocidades de 5 Mbit/s com atraso de 2 milissegundos, como é apresentado na Figura 31.

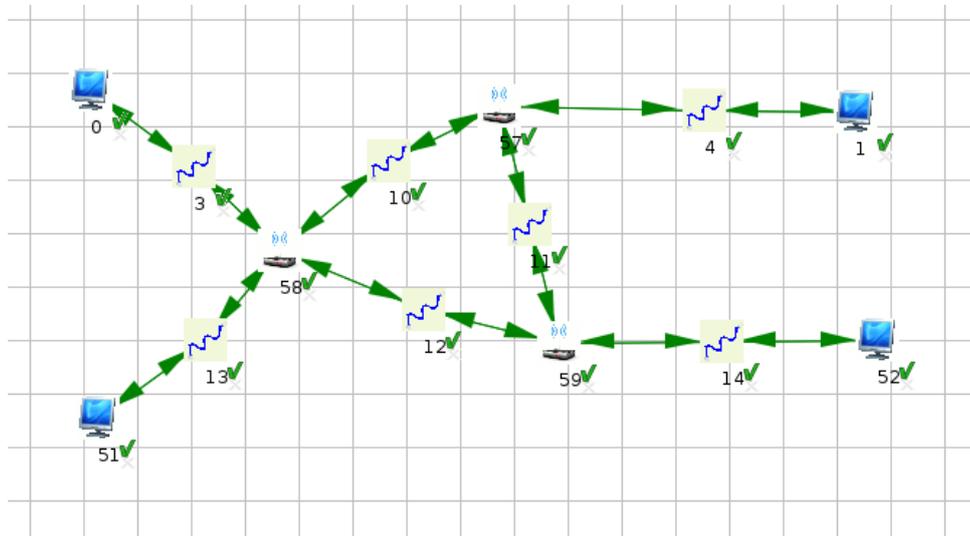


Figura 31 – Ambiente de redes de computadores do primeiro modelo testado

Nele, foram realizadas simulações com fluxos de 100.000, 200.000 e 400.000 pacotes, sendo que cada um destes possuía o tamanho de 1.000 *bytes*. Houve duas séries de testes, sendo que na primeira a capacidade das filas presentes nos elementos da rede era infinita, não acontecendo perdas de pacotes, e no segundo a capacidade para estas era de 30.000 pacotes por fila, ocorrendo perdas. As validações para este modelo foram realizadas nos 68 simuladores obtidos pelo teste de usabilidade e no NS-3, comparando-se o tempo simulado, ou seja, tempo avançado pelo relógio virtual durante a simulação, entre eles.

Primeira série de testes

A partir da primeira série de testes, em que a capacidade das filas eram infinitas, pode-se perceber que dentre os resultados obtidos para os simuladores gerados, a maioria era muito próxima dos obtidos para o NS-3 (até 6% de diferença). Outros divergiam completamente deste (mais que 90% de diferença), como pode-se observar na Figura 32.

Observa-se que a maioria (79.4%) dos 68 testes possuem pequena diferença (até 6%) para o tempo simulado apresentado pelo NS-3, enquanto uma minoria (20.6%) possui tempos com faixas entre 90 e 100% de divergência para este. A partir de tais resultados, concluiu-se que tal desigualdade entre os 14 (20.6%) dos resultados ocorreu devido à utilização inadequada do Yasc por parte dos validadores da ferramenta. Durante a etapa de criação dos simuladores, eles modelaram todos os objetos que seriam disponibilizados nestes como instantâneos, o que resultou em um tempo simulado de valor 0 para qualquer simulação realizada. Desta forma, tais simuladores foram excluídos da avaliação e comparou-se apenas os tempos simulados corretos

Teste 1 (Nº de testes/Intervalo de tempo)

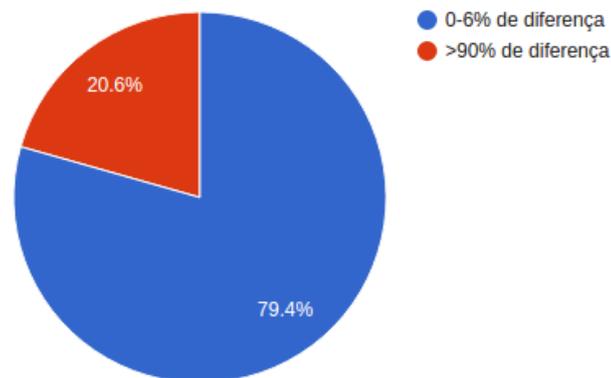


Figura 32 – Representação da quantidade de testes sem perdas de pacotes

entre ambas as ferramentas. Nesta série não havia perda de pacotes, como pode ser observado na Tabela 2 e no gráfico referente a ela presente na Figura 33. Estes descrevem tempos simulados do *Network Simulator* e do simulador gerado pelo Yasc que atingiu os melhores resultados, notando-se diferença praticamente inexistente entre os tempos.

Tabela 2 – Tempos simulados (em segundos) do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	164,807	329,607	659,207
Yasc	164,17	328,35	656,68

Teste 1 (sem perda de pacotes)

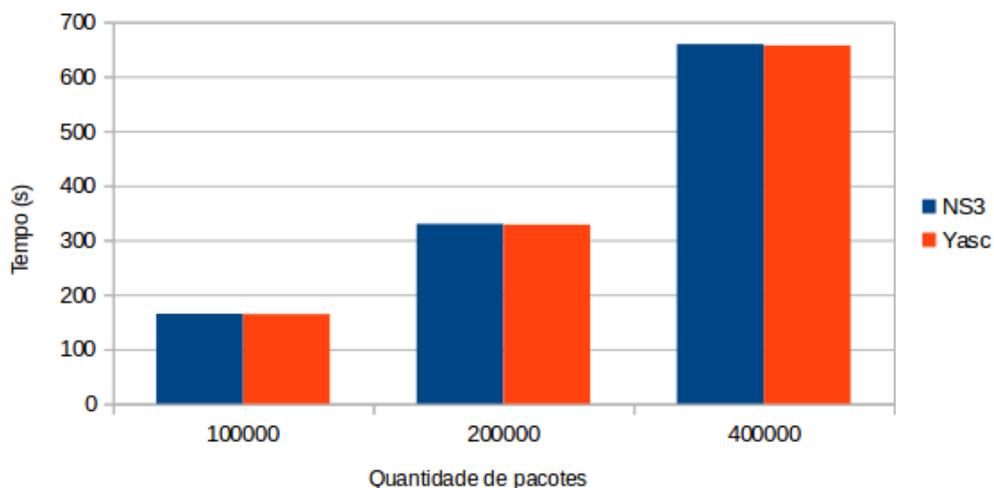


Figura 33 – Comparativo entre tempos simulados do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes

Segunda série de testes

Na segunda série de testes, em que a capacidade das filas foram definidas como de 30.000 pacotes, novamente foram notados entre os resultados dos simuladores gerados, o distanciamento para o NS-3, como é apresentado na Figura 34. Além dos tempos que divergiam totalmente, conforme ocorrido no teste sem perdas de pacotes, também houve o surgimento de uma faixa intermediária de simuladores cujos tempos possuíam uma diferença com valores entre 65 e 70% dos que foram obtidos pelo NS-3. Para estes simuladores, notou-se que a desigualdade vinha novamente de problemas em sua criação, sendo agora os nós terminais e *links* de comunicação descritos como instantâneos e os roteadores como o tipo básico uma fila e um servidor.

Desta forma, eles também foram descartados, sendo comparados apenas os que possuíam correta descrição. Assim, obtiveram-se os resultados apresentados nas Tabelas 3 e 4 e em seus respectivos gráficos, que aparecem nas Figuras 35 e 36, em que observa-se novamente a grande proximidade entre os tempos simulados e agora também entre as quantidades de perdas de pacotes do *Network Simulator* e do simulador gerado com resultado mais próximo.

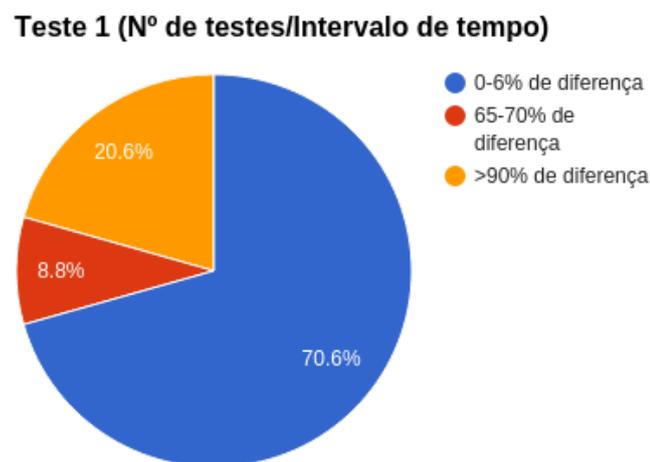


Figura 34 – Representação da quantidade de testes com resultados próximos ou não do NS-3 com perdas de pacotes

Tabela 3 – Tempos simulados (em segundos) do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	148,328	148,328	148,328
Yasc	147,76	147,76	147,76

Tabela 4 – Quantidade de pacotes perdidos para o primeiro modelo do NS-3 e do resultado mais próximo obtido pelo simulador gerado pelo Yasc

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	10000	110001	310001
Yasc	10001	110000	310001

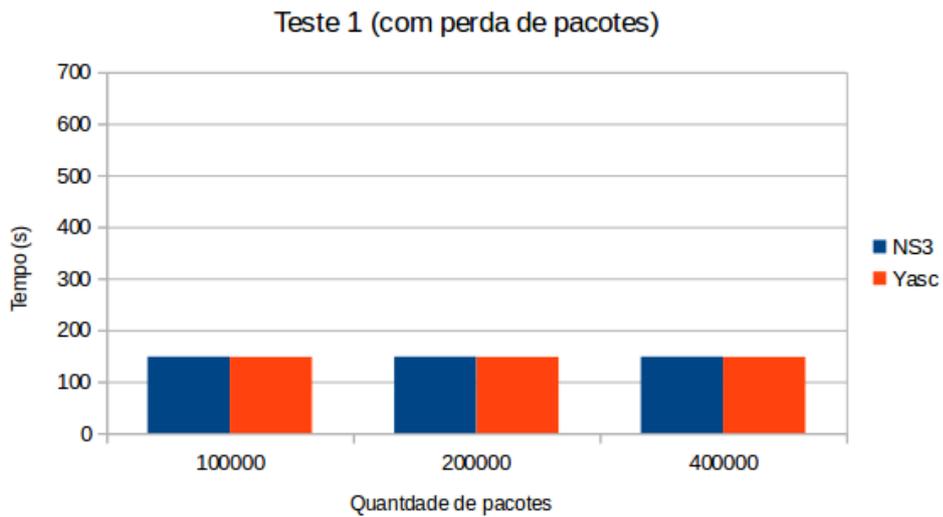


Figura 35 – Comparativo entre tempos simulados do primeiro modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes

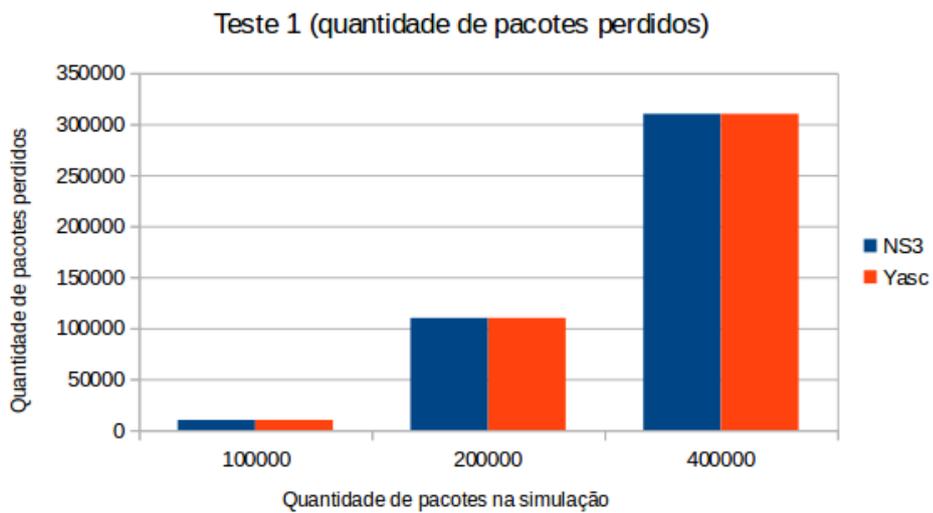


Figura 36 – Comparativo entre as quantidades de pacotes perdidos pelo NS3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc para o primeiro modelo

4.3.2 Segundo modelo simulado

O segundo modelo consiste na representação de uma rede real, pertencente ao Grupo de Sistemas Paralelos e Distribuídos (GSPD). Ele é composto por 21 nós terminais, dentre os quais 17 fazem parte do *cluster* do laboratório, 1 é o servidor e os demais são máquinas para uso pessoal, 2 *switches* e um roteador, todos interligados por canais de comunicação com velocidade de 1 Gbit/s e apenas o servidor ligado ao *switch* pertencente ao *cluster* por um canal de 10 Gbit/s, conforme mostrado na Figura 37.

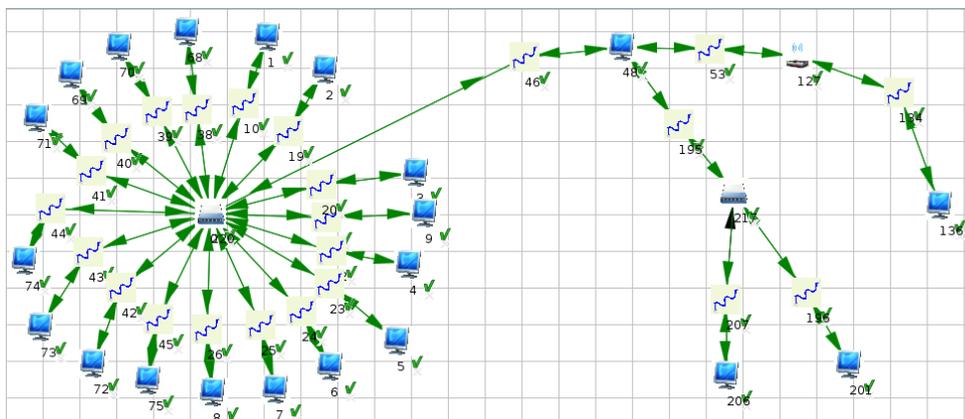


Figura 37 – Ambiente de redes do segundo modelo testado, representando o laboratório GSPD

Neste teste, as simulações também ocorreram com fluxos de 100.000, 200.000 e 400.000 pacotes, com 1.000 *bytes* cada. Os demais procedimentos foram todos repetidos, com duas séries de testes, sendo que na primeira a capacidade das filas era infinita, sem perdas de pacotes e na segunda, a capacidade para estas era de 30.000 pacotes por fila, ocorrendo perdas. As validações para este modelo também foram realizadas nos 68 simuladores obtidos pelo teste de usabilidade e no NS-3.

Primeira série de testes

A partir da primeira série de testes, em que a capacidade das filas eram infinitas, da mesma forma que no teste com o modelo hipotético, pode-se perceber que dentre os resultados obtidos para os simuladores gerados, a maioria era muito próxima dos obtidos para o NS-3 (até 6% de diferença). Outros divergiam completamente deste (mais que 90% de diferença), como pode-se observar na Figura 38.

Observa-se, novamente, que a maioria (79.4%) dos 68 testes possuem pequena diferença (até 6%) para o tempo simulado apresentado pelo NS-3, enquanto uma minoria (20.6%) possui tempos com faixas entre 90 e 100% de divergência para este. Tais resultados com desigualdade são causados devido à utilização inadequada do Yasc por parte dos validadores da ferramenta. Novamente, os simuladores criados de maneira errônea foram excluídos da avaliação e comparou-se apenas os tempos simulados corretos entre ambas as ferramentas. Nesta série não havia perda de pacotes, como pode ser observado na Tabela 5 e no gráfico referente a ela presente na Figura 39.

Teste 2 (Nº de testes/Intervalo de tempo)

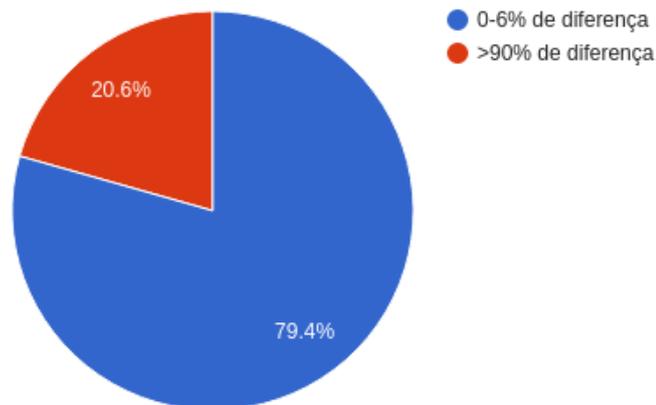


Figura 38 – Representação da quantidade de testes com resultados próximos ou não do NS-3 sem perdas de pacotes

Estes descrevem tempos simulados do *Network Simulator* e do simulador gerado pelo Yasc que atingiu os melhores resultados, notando-se diferença praticamente inexistente entre os tempos.

Tabela 5 – Tempos simulados (em segundos) do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	0,5753	1,1247	2,2233
Yasc	0,5472	1,0945	2,1889

Teste 2 (sem perda de pacotes)

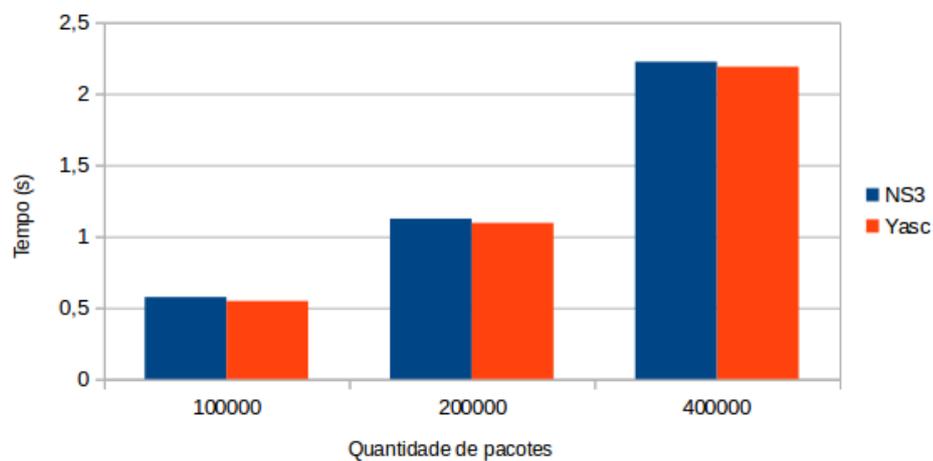


Figura 39 – Comparativo entre tempos simulados do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc sem perdas de pacotes

Segunda série de testes

Na segunda série de testes, em que a capacidade das filas foram definidas como de 30.000 pacotes, novamente foram notados entre os resultados dos simuladores gerados, o distanciamento para o NS-3, como é apresentado na Figura 40. Além dos tempos que divergiam totalmente, conforme ocorrido no teste sem perdas de pacotes, também houve o surgimento de uma faixa intermediária de simuladores cujos tempos possuíam uma diferença com valores entre 50 e 55% dos que foram obtidos pelo NS-3. Para estes simuladores, notou-se que a desigualdade vinha novamente de problemas em sua criação.

Eles também foram descartados, sendo comparados apenas os que possuíam correta descrição. Assim, obtiveram-se os resultados apresentados nas Tabelas 6 e 7 e em seus respectivos gráficos, que aparecem nas Figuras 41 e 42, em que observa-se novamente a grande proximidade entre os tempos simulados e agora também entre as quantidades de perdas de pacotes do *Network Simulator* e do simulador gerado com resultado mais próximo.

Teste 2 (Nº de testes/Intervalo de tempos)

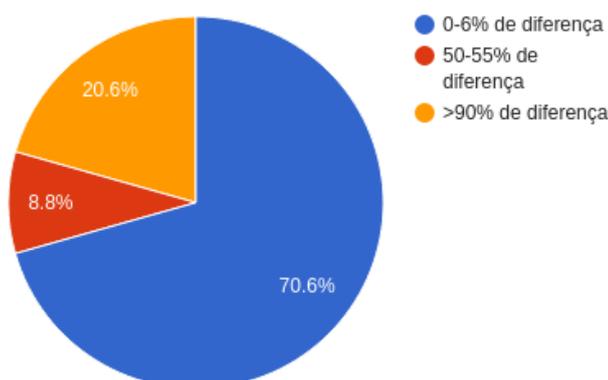


Figura 40 – Representação da quantidade de testes com resultados próximos ou não do NS-3 com perdas de pacotes

Tabela 6 – Tempos simulados (em segundos) do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	0,5204	0,5204	0,5204
Yasc	0,4925	0,4925	0,4925

Tabela 7 – Quantidade de pacotes perdidos para o segundo modelo do NS-3 e do resultado mais próximo obtido pelo simulador gerado pelo Yasc

Simulador/Nº de pacotes	100.000	200.000	400.000
NS-3	9998	110000	309998
Yasc	9998	109999	309998

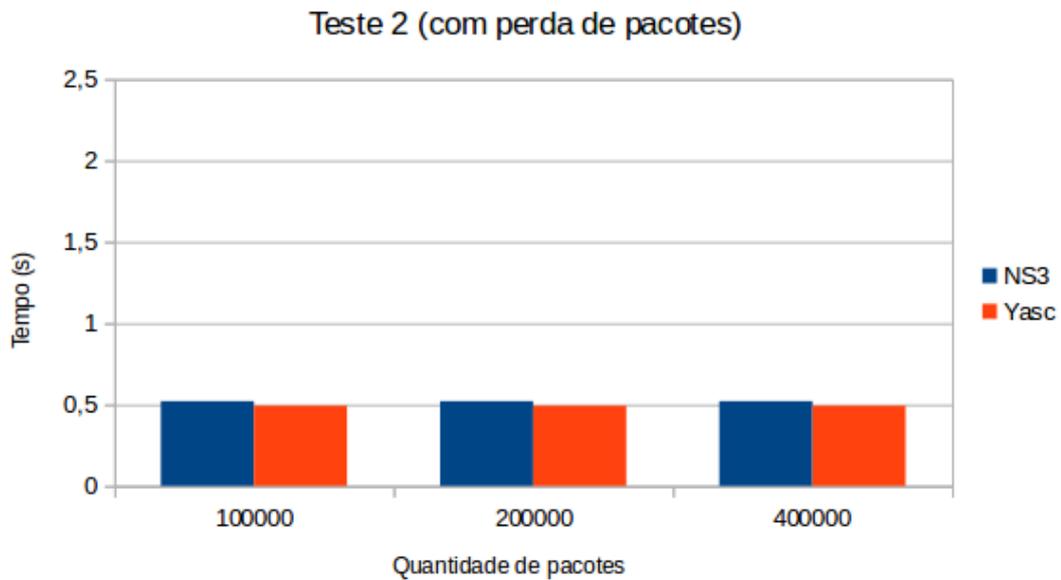


Figura 41 – Comparativo entre tempos simulados do segundo modelo para o NS-3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc com perdas de pacotes

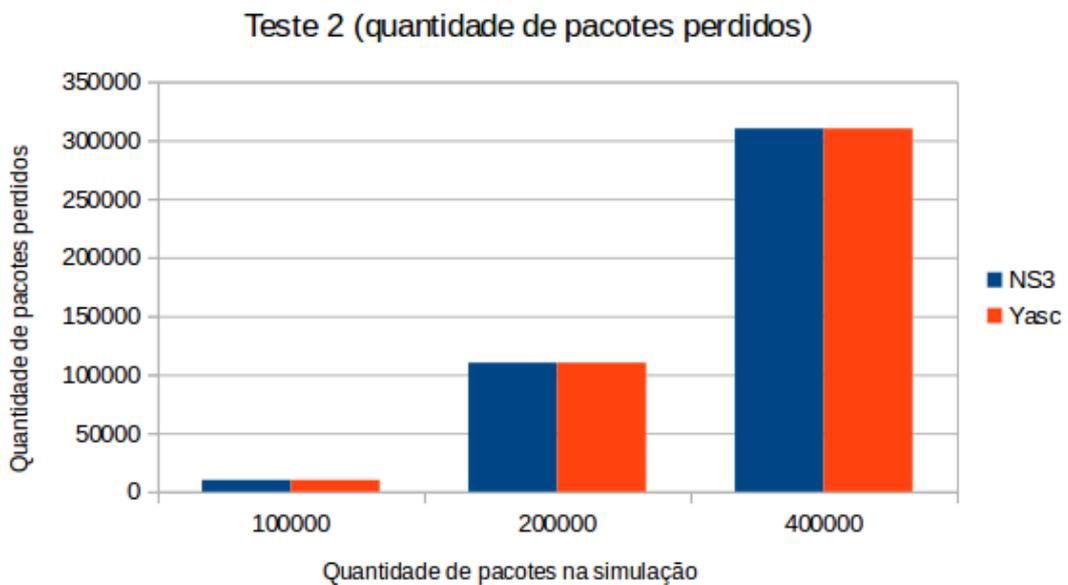


Figura 42 – Comparativo entre as quantidades de pacotes perdidos pelo NS3 e o resultado mais próximo obtido pelo simulador gerado pelo Yasc para o segundo modelo

4.4 Considerações finais

Neste capítulo foram apresentados resultados obtidos a partir da análise efetuada com a ferramenta implementada, o Yasc. Por meio dos testes e validações executados, cujos produtos foram satisfatórios, concluiu-se a simplicidade da utilização da ferramenta pelo usuário e a validade das métricas apresentadas por este em forma de resultado (desde que o usuário projete

seu simulador de maneira correta).

Ao longo da validação, o Yasc foi executado em três diferentes sistemas operacionais. O que consolidou o *software* como uma ferramenta multiplataformas.

5 Conclusões

Neste capítulo apresentam-se as conclusões sobre o trabalho e os próximos passos para a adição de novas funcionalidades à ferramenta.

Este trabalho teve como intuito o desenvolvimento de uma ferramenta para a geração de simuladores, de maneira simples e eficaz, com base em diferentes contextos em que se encaixam filas básicas, o Yasc (*Yes, a simulator's compiler*).

Para isto, foram feitos estudos dos mais diversos contextos de aplicações baseadas em filas a partir da modelagem e simulação. Esse estudo foi necessário devido a importância da observação do comportamento de sistemas que deve ser feita através de maneiras eficazes que permitam redução expressiva de tempo no processo de desenvolvimento de software. Tornando assim o uso de modelos básicos essencial, a fim de oferecer diretamente ao usuário um gerador de simuladores, ferramenta que possibilita a avaliação de sistemas.

As dificuldades de projeto envolveram a classificação de componentes dentro de padrões específicos, que permitem a aplicação dos modelos básicos em condições equivalentes dentro dos centros de serviço, ou seja, em como relacionar quais centros de serviços implementados no motor de simulação são equivalentes a quais atividades dos tipos de aplicação propostas.

Como principal contribuição, o Yasc possibilitará com que usuários leigos, que não conhecem linguagens de programação, ou possuem dificuldades ao trabalhar com essas linguagens, por não serem especialistas da área de computação, possam desenvolver simuladores que permitam avaliar o desempenho de seus trabalhos através de sua interface gráfica simples e amigável.

Deve-se ressaltar que todo o trabalho foi realizado com êxito conforme o cronograma que havia sido proposto.

5.1 Trabalhos futuros

Como trabalhos futuros destacam-se melhorias e funcionalidades que podem ser incorporadas ao Yasc sendo que as principais são mostradas a seguir:

- Reimplementação do motor com o intuito de torná-lo genérico. Tal reimplementação não visa obter melhores resultados, nem acertar falhas, uma vez que o motor do Yasc foi construído a partir de outro motor de simulação já validado, mas sim possibilitar a simulação de novos contextos, não sendo apenas estes baseados em filas. Dessa forma, será permitido também a simulação de modelos fechados.

- Acrescentar métricas apresentadas ao usuário na forma de resultados, tornando possível que adaptem-se a qualquer tipo de simulador.
- Permitir, no simulador gerado, o carregamento das cargas por meio de *traces*, ou seja, um arquivo com parâmetros descrevendo a carga para determinado sistema. Isso torna a descrição simples, não sendo necessário fazê-la de maneira manual, o que é inviável conforme sua quantidade aumenta.

Deve-se ressaltar que já encontram-se funcionalidades em desenvolvimento em outro trabalho de pesquisa realizado no GSPD, que envolve utilização do Yasc para a geração de simuladores de circuitos sequenciais, o qual, já está na etapa de implementação. Sua conclusão está prevista para o final do ano de 2016.

5.2 Publicações

Com a finalidade de expor as atividades de desenvolvimento do Yasc, resultados parciais foram publicados nos seguintes eventos:

- *Geração de simuladores de filas para diferentes contextos*. Apresentado na Escola Regional de Alto Desempenho de São Paulo (ERAD -SP), 2015, São José do Rio Preto (FURLANETTO et al., 2015a).
- *Geração de simuladores de filas para diferentes contextos*. Apresentado no V Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP (WPPGCC 2015) (FURLANETTO et al., 2015b).
- *A tool for model conversion between simulators of grid computing*. Apresentado no *Annual Simulation Symposium* (ANSS), 2015, Alexandria (Virgínia, Estados Unidos), conferência com qualis B1 em Ciência da computação. (FURLANETTO et al., 2015c).

Referências

- BANKS, J. et al. *Discrete-Event System Simulation*. Fifth edition. [S.l.]: Prentice-Hall, 2009.
- BUSSAB, W. d. O.; MORETTIN, P. A. *Estatística básica*. 8a edição. ed. [S.l.]: Saraiva, 2013.
- CARDOSO J. VALETTE, R. *Redes de Petri*. [S.l.: s.n.], 1997.
- CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer, 2008.
- (DCG), D. C. G. Sinalgo - simulator for network algorithms. Disponível em <<http://www.disco.ethz.ch/projects/sinalgo/>> Acessado em: 18 de janeiro. 2016.
- DOSS, D. L.; ULGEN, O. M. A case for generic, custom-designed simulation applications for material handling and manufacturing industries. In: CITESEER. *Brooks Automation's Worldwide Automation Symposium*. [S.l.], 2004. p. 1–5.
- FU, D.; BECKER, M.; SZCZEBICKA, H. Universal simulation engine (use) - a model-independent library for discrete event simulation. In: *Annual Simulation Symposium (ANSS)*. [S.l.: s.n.], 2015. p. 1–8.
- FURLANETTO, G. C. et al. Geração de simuladores de filas para diferentes contextos. In: *VI Escola Regional de Alto Desempenho de São Paulo*. São José do Rio Preto-SP: [s.n.], 2015. CD-ROM.
- FURLANETTO, G. C. et al. Geração de simuladores de filas para diferentes contextos. In: *V Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP*. Bauru-SP: [s.n.], 2015. CD-ROM.
- FURLANETTO, G. C. et al. A tool for model conversion between simulators of grid computing. In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL. *Proceedings of the 48th Annual Simulation Symposium*. [S.l.], 2015. p. 9–16.
- GORDON, G. The development of the general purpose simulation system (gps). In: ACM. *History of programming languages I*. [S.l.], 1978. p. 403–426.
- HILLIER, F. S.; LIEBERMAN, G. J. *Introdução a pesquisa operacional*. 1a edição. ed. [S.l.]: Editora da Usp, São Paulo, 1988. Tradução de Helena L. Lemos.
- ISI, I. S. I. Network simulator - ns3. Disponível em <<https://www.nsnam.org/>> Acessado em: 18 de janeiro. 2016.
- KLEINROCK, L. *Queueing systems*. First edition. [S.l.]: Wiley-Interscience, 1975.
- LEI, Y. et al. A metamodel-based representation method for reusable simulation model. In: IEEE. *Simulation Conference, 2007 Winter*. [S.l.], 2007. p. 851–858.
- MACDOUGALL, M. *Simulating Computer Systems Techniques and Tools*. 2nd. ed. [S.l.]: The MIT Press, 1989.

- MENEZES, D. et al. Scheduler simulation using ispd, an iconic-based computer grid simulator. In: *Computers and Communications (ISCC), 2012 IEEE Symposium on*. [S.l.: s.n.], 2012. p. 637 – 642.
- PATER, A. J.; TEUNISSE, M. J. The use of a template-based methodology in the simulation of a new cargo track from rotterdam harbor to germany. In: IEEE COMPUTER SOCIETY. *Proceedings of the 29th conference on Winter simulation*. [S.l.], 1997. p. 1176–1180.
- PRESSMAN, R. S. *Engenharia de software*. 11a edição. ed. [S.l.]: McGraw Hill Brasil, 2011.
- RADATZ, J. *The IEEE standard dictionary of electrical and electronics terms*. [S.l.]: IEEE Standards Office, 1997.
- RAMPERSAD, K. *The rapid design of simulation models using cladistics and template based modelling*. Tese (Doutorado) — Cranfield University, 2012.
- SOARES, L. F. G. *Modelagem e Simulação Discreta de Sistemas*. [S.l.: s.n.], 1992.
- SOLON, R. et al. Desenvolvimento de um gerador de aplicação para simulação de sistemas discretos. Notas do ICMSC-USP. 1994.
- SOLON, R. et al. Automatic generation of discrete-system simulation programs. In: *Summer Computer Simulation Conference*. [S.l.: s.n.], 1996. p. 133–138.
- SPÓSITO, T. G. *Sistema Toyota de Produção e Kanban: uma abordagem prática aos resultados esperados e às dificuldades inerentes à sua implantação*. Monografia (Trabalho de Conclusão de Curso) — Escola de Minas da Universidade Federal de Ouro Preto (UFOP), Ouro Preto, 2003.
- TANENBAUM, A. S. *Modern operating systems*. [S.l.]: Pearson Education, 2009.
- VARGA, A. Omnet ++ - discrete event simulator. Disponível em <<http://www.omnetpp.org/>> Acessado em: 18 de janeiro. 2016.
- VARGA, A.; HORNIG, R. An overview of the omnet++ simulation environment. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TELECOMMUNICATIONS ENGINEERING). *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. [S.l.], 2008. p. 60.
- WEINGÄRTNER, E.; LEHN, H. V.; WEHRLE, K. A performance comparison of recent network simulators. In: IEEE. *Communications, 2009. ICC'09. IEEE International Conference on*. [S.l.], 2009. p. 1–5.
- ZHU, N. et al. Generating domain-specific simulation environments from model frameworks based on smp2 for rapid development of simulation applications. In: *Annual Simulation Symposium (ANSS)*. [S.l.: s.n.], 2015. p. 1–4.

Apêndices

APÊNDICE A – Plano de Teste

O que é a ferramenta?

O Yasc é uma ferramenta para gerar simuladores. Nele o projetista, por meio de uma interface gráfica, fornece descrições dos os elementos que farão parte do simulador que será gerado e também detalha como estes se comportam.

A partir desta descrição a ferramenta gera um simulador com interface icônica. O objetivo de usar interface icônica é facilitar a tarefa de criar modelos dos sistemas a serem simulados, evitando a necessidade de codificação dos ambientes.

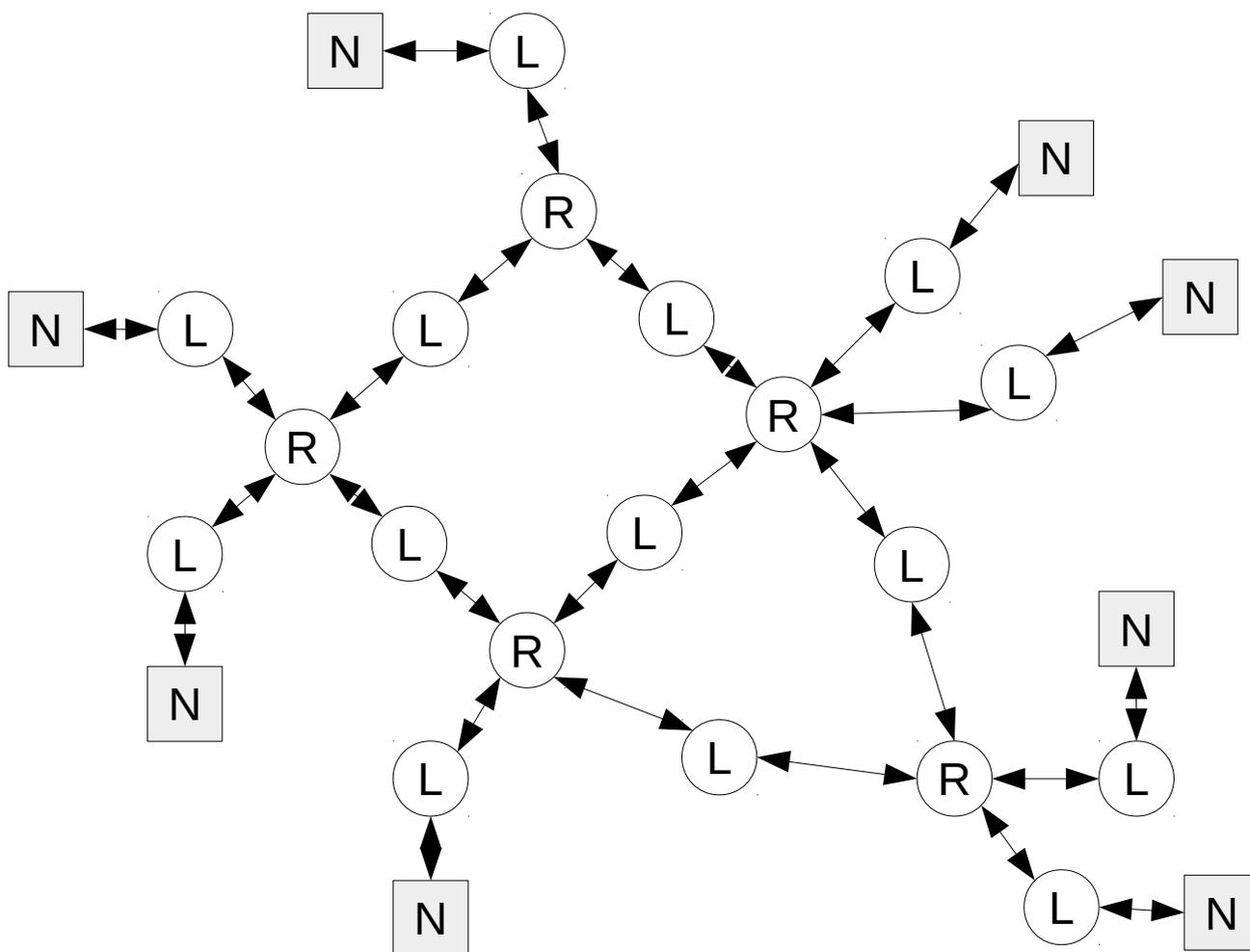
Proposta de simulador a ser gerado para validação

Com a finalidade de validar o sistema e testar sua usabilidade, pedimos que você utilize o Yasc e construa um simulador de redes de computadores. Esse simulador deverá permitir a modelagem, isto é deve ter ícones representativos, de canais de comunicação, elementos de conexão (roteadores, switches, etc), além de nós representando estações de trabalho (Pcs, etc). Cabe a você indicar que parâmetros são importantes nos elementos criados (tipos, nomes, unidades de medida, etc).

Para facilitar a criação da interface icônica são oferecidos ícones para canais de comunicação (figura1.gif), elementos de conexão (figura2.gif e figura3.gif) e nós terminais (figura4.gif)

Por fim, para o término da configuração do simulador, eventos deverão ser denominados como pacotes. Haverá simulações de falhas e limitações de capacidade.

Após a geração do simulador, crie nele o seguinte modelo:



Em que:

R – Roteador

L – Link de comunicação

N - Nó terminal marcado como início e fim

Para configurar todos os ícones do desenho, adicione o *Service Time* como 2 e a preempção como 0.2 (20%).

Por fim, crie 1000 tarefas na distribuição *Stage Two Uniform*, sendo o parâmetro mínimo igual a 1000, o máximo igual a 100000 e o ponto de quebra igual a 70000.

Obs: é possível copiar e colar ícones que já foram configurados

Desenvolvido por Gabriel Covello Furlanetto, aluno do programa de pós graduação em ciência da computação da Unesp
Obrigado por participar da validação e por favor, não se esqueça de preencher o questionário.

APÊNDICE B – Questionário de validação

O objetivo deste questionário é coletar informações sobre a opinião do usuário a respeito do protótipo do Yasc (Yes, a simulator's compiler). As informações fornecidas serão de extrema importância para a validação e o aprimoramento do sistema. Por favor, leia com atenção as questões a seguir e em caso de dúvida, solicite esclarecimento ao avaliador.

Nas questões a seguir, indique seu grau de concordância com a afirmação feita, sendo:

- 1-Discordo totalmente
- 2-Discordo em boa parte
- 3-Nem concordo nem discordo
- 4-Concordo em boa parte
- 5-Concordo integralmente

a)	O Yasc é fácil de usar	1	2	3	4	5
b)	As informações na interface do Yasc estão bem organizadas	1	2	3	4	5
c)	A aparência das telas do Yasc é bastante clara	1	2	3	4	5
d)	A nomenclatura utilizada nas telas (nome de comandos, títulos, campos, etc.) é fácil de compreender	1	2	3	4	5
e)	As mensagens do sistema são claras	1	2	3	4	5
f)	No geral, a utilização do Yasc foi interessante	1	2	3	4	5
g)	O Yasc é uma ferramenta interessante para geração de simuladores	1	2	3	4	5

Aponte os pontos positivos que você encontrou ao utilizar o sistema:

Aponte os pontos negativos que você encontrou ao utilizar o sistema:

**Desenvolvido por Gabriel Covello Furlanetto, aluno do programa de pós graduação em ciência da computação da Unesp
Obrigado por participar da validação e por favor, não se esqueça de preencher o questionário.**

APÊNDICE C – Manual de uso

1-Introdução

O Yasc é uma ferramenta para geração de simuladores. Nele o projetista, por meio de uma interface gráfica, fornece descrições dos elementos que farão parte do simulador que será gerado e também detalha como estes se comportam.

A partir desta descrição a ferramenta cria um simulador com interface icônica. O objetivo de usar interface icônica é facilitar a tarefa de criar modelos dos sistemas a serem simulados, evitando a necessidade de codificação dos ambientes.

Este texto tem como finalidade orientar o usuário durante a geração e a utilização de simuladores. Para isto, na sequência serão apresentados como exemplo os passos que devem ser seguidos para a criação e utilização de um simulador de redes de computadores.

2- Como utilizar a interface gráfica para criar um simulador

Nesta seção do manual, será descrito como criar um simulador de redes de computadores contendo roteadores, *switchs* e nós terminais e *links* de comunicação. Além disso, ele ainda deverá permitir a simulação de falhas como perdas de pacotes e apresentá-las como métricas junto ao tempo médio de fila e o tempo médio de atendimento para cada pacote.

Para isso, os seguintes passos devem ser seguidos:

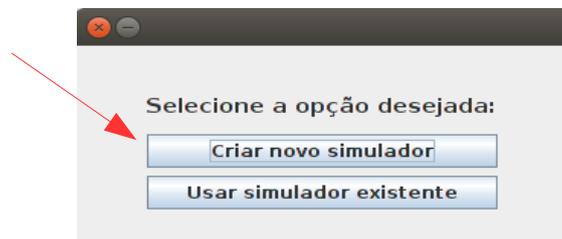
1º Execute a ferramenta

Windows e Mac Os → basta dar dois cliques sobre o ícone da ferramenta

Linux → abra um terminal e digite “`java -jar Yasc.jar`”

Após realizado o primeiro passo, será solicitado ao usuário qual linguagem deseja e também aparecerá a tela de boas vindas ao simulador, bastando clicar “OK” em ambos para prosseguir.

2º Selecione a opção “Criar novo simulador”



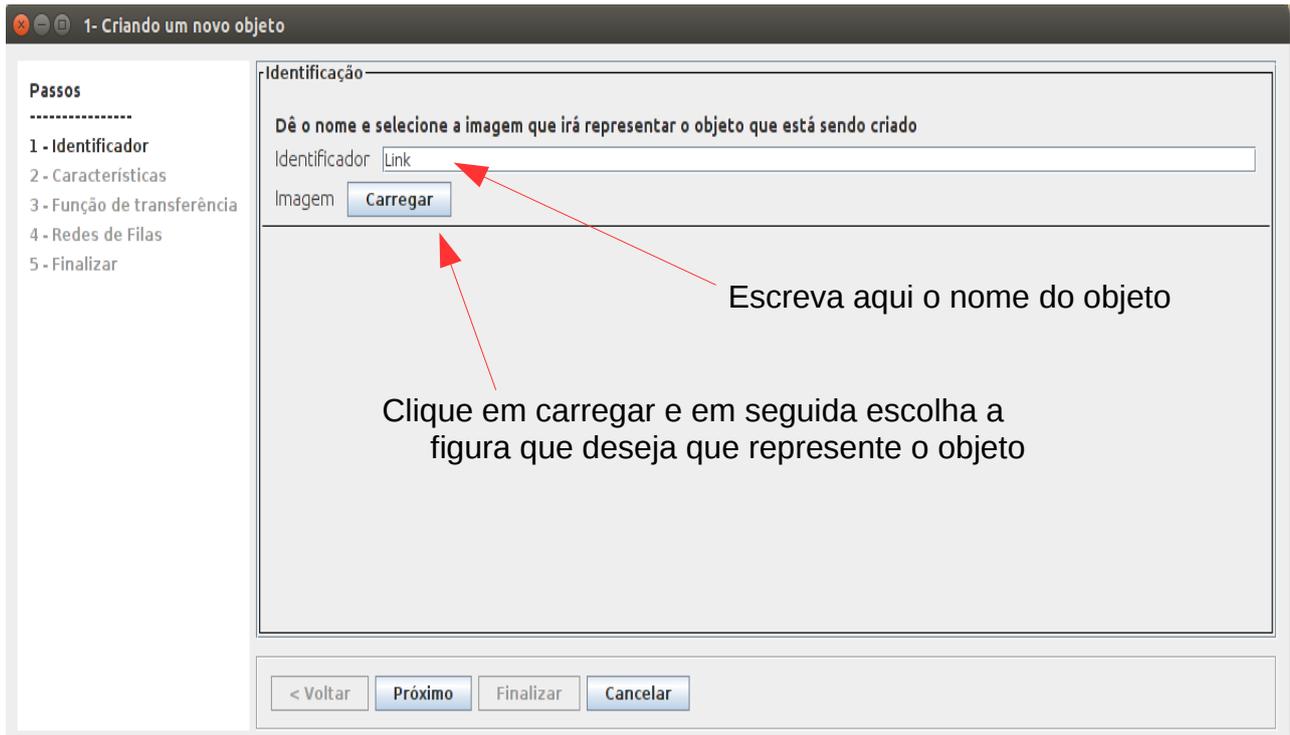
3º Crie os objetos

No segundo passo da etapa de criação, deverão ser projetados os objetos a serem disponibilizados no simulador criado. Neste caso, serão projetados o roteador, *switch*, *link* de comunicação e nós terminais da seguinte forma:

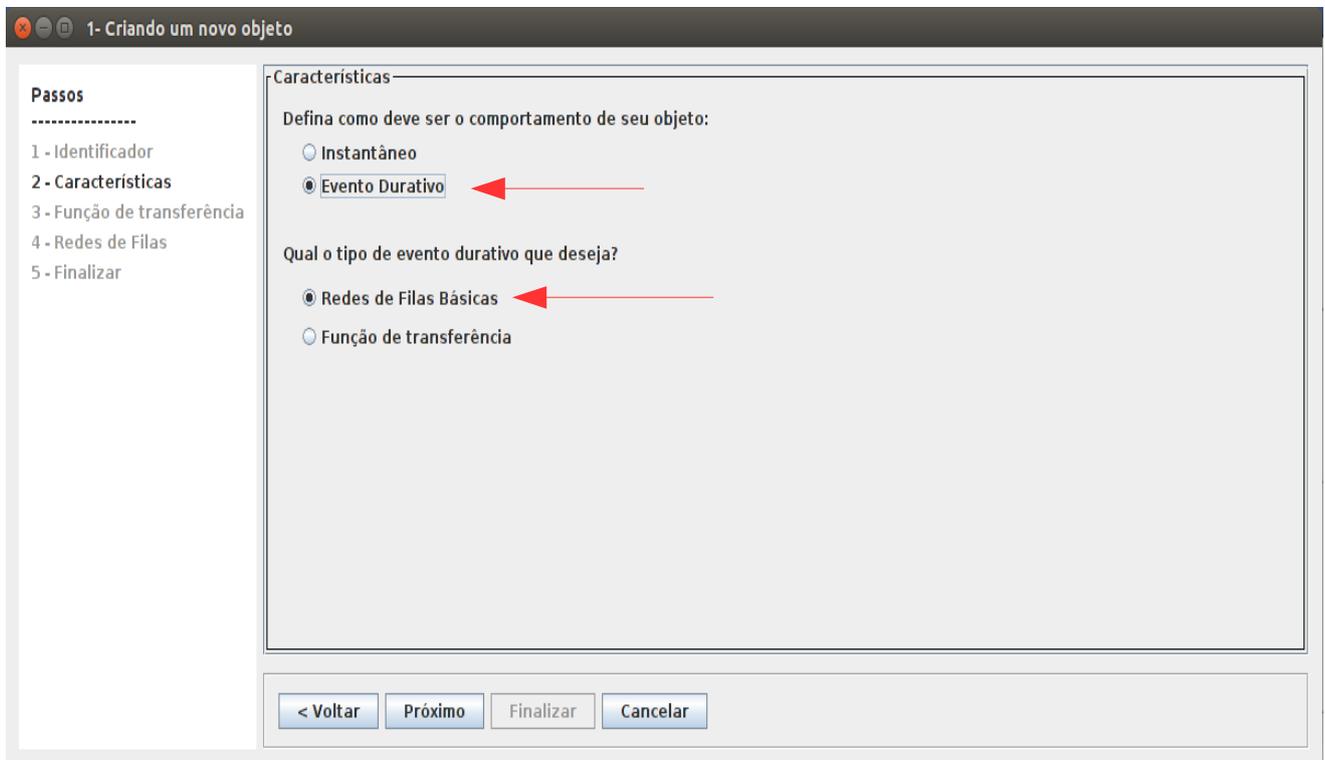
a) *Link de comunicação*

1) Coloque o nome do objeto como *Link*, selecione a imagem que o representará e clique em

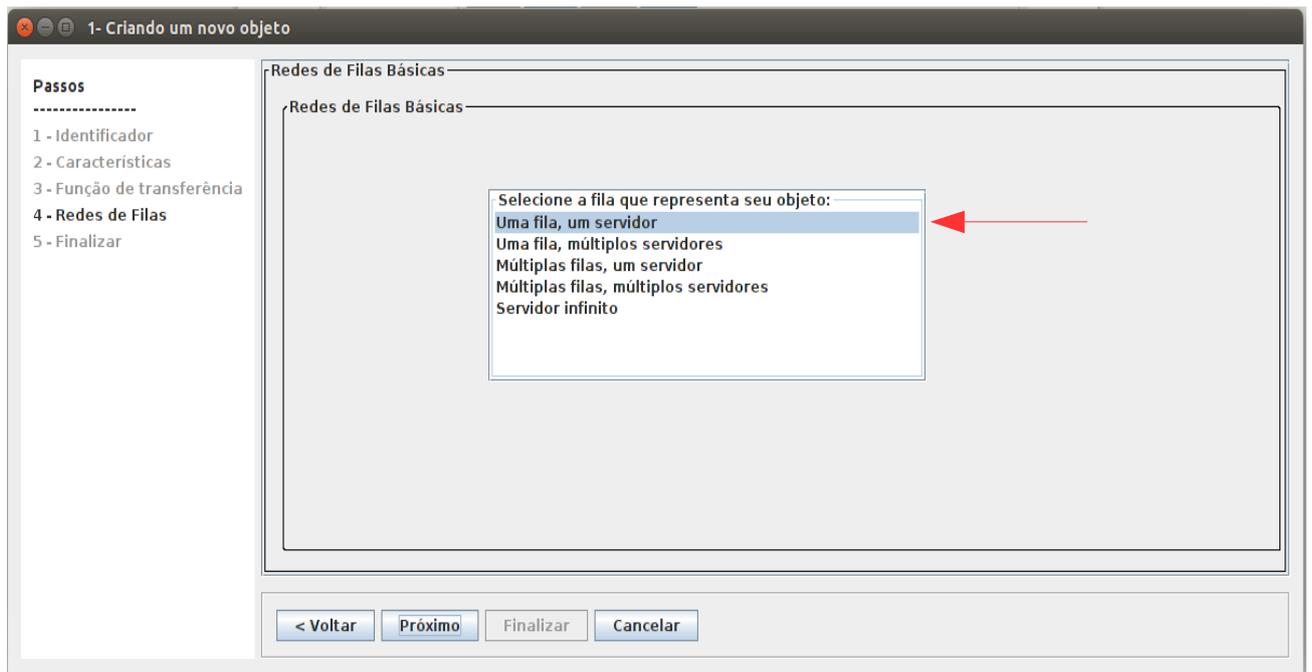
“Próximo”.



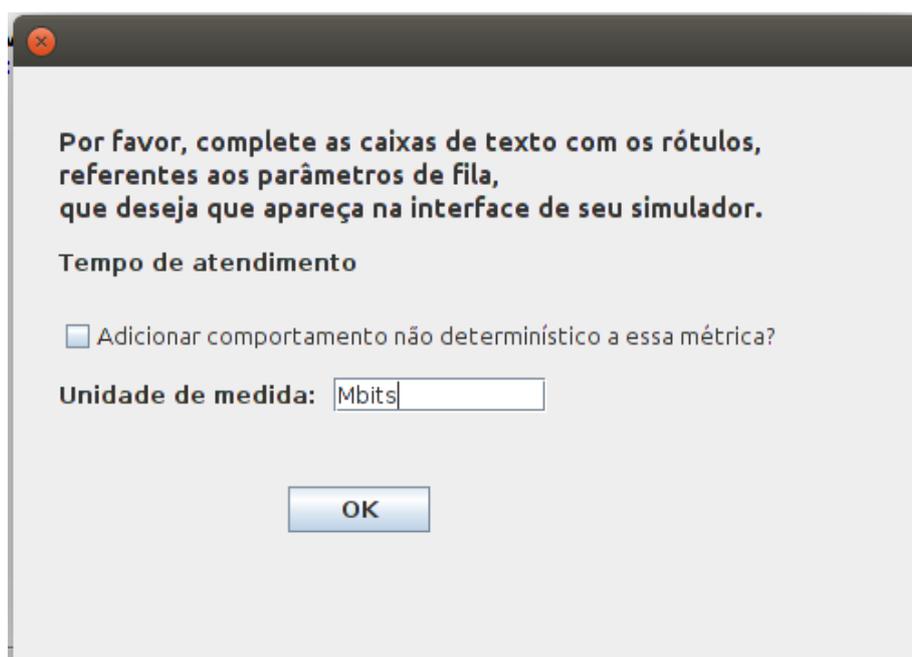
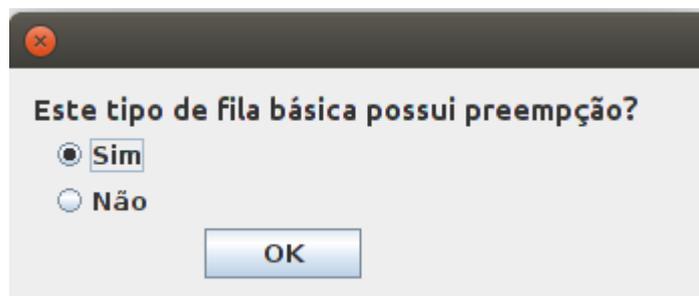
II) Selecione o tipo de comportamento que o objeto segue e clique em “Próximo”. (Para o *link*, selecionaremos um comportamento durativo, do tipo fila básica).



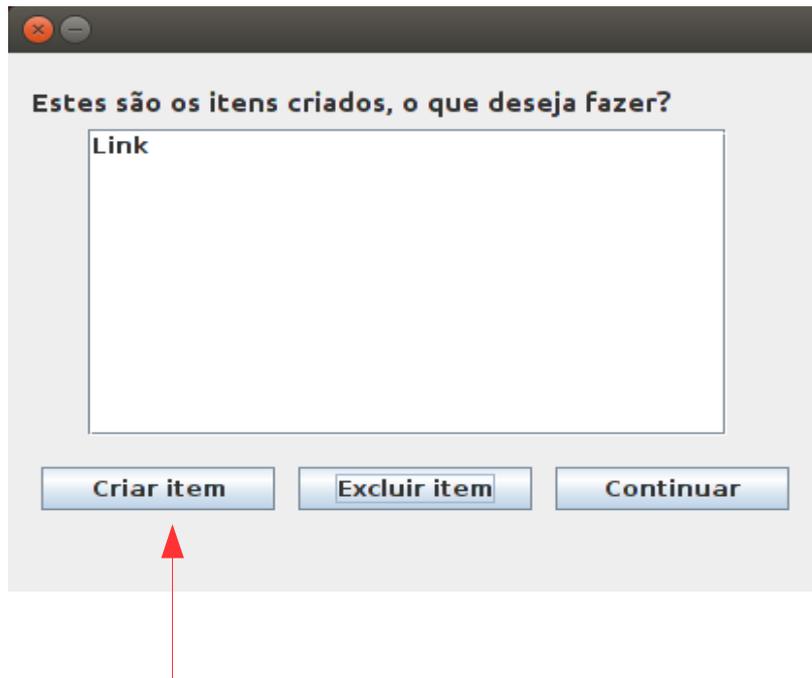
III) Selecione como tipo de fila básica “uma fila e um servidor” e clique novamente em “Próximo”.



IV) Aparecerá uma tela com o resumo do que foi feito. Clique em finalizar e será questionado sobre haver ou não preempção no simulador e o nome da métrica que representa o tempo de atendimento do servidor, preencha da seguinte maneira e clique em “OK”.



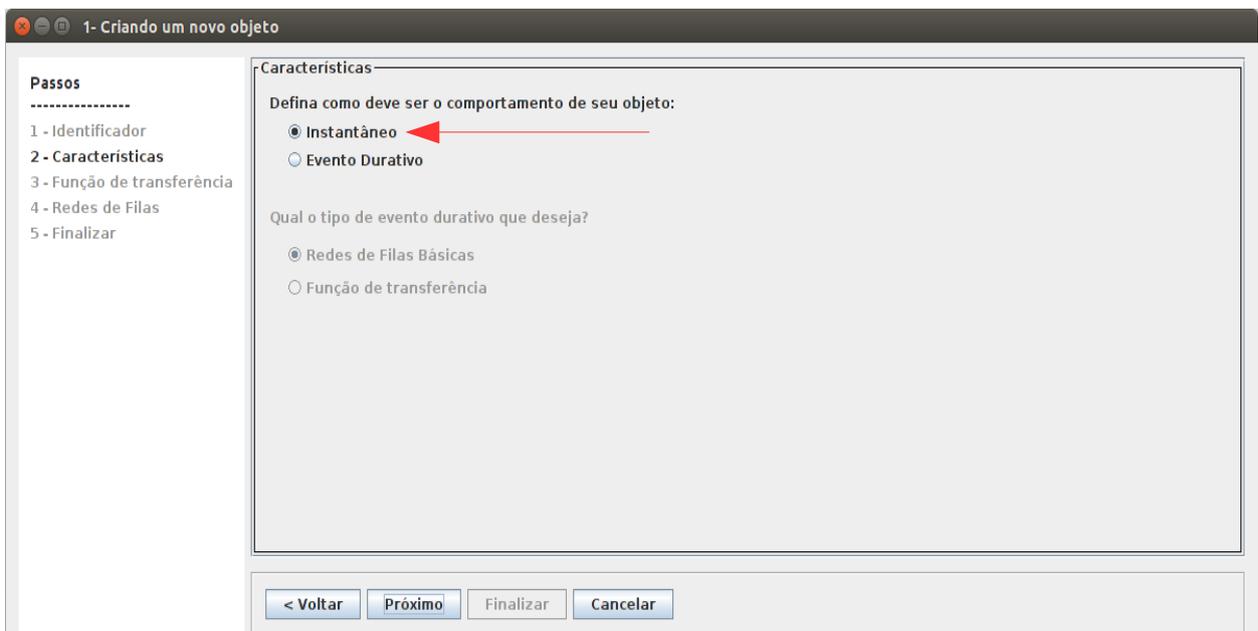
V) Ao finalizar, aparecerá uma interface como a mostrada abaixo, a qual realiza o gerenciamento dos ícones criados. Clique em “Criar item” e inicie a criação dos demais itens.



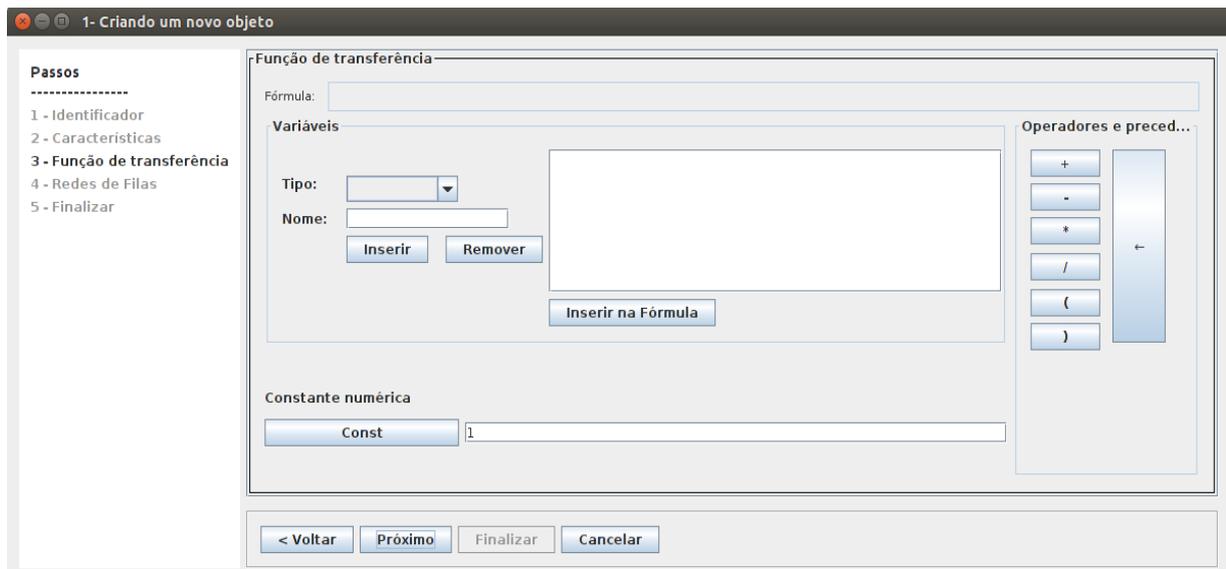
b) Roteador, *switch* e nós terminais

I) Novamente forneça um nome e escolha uma imagem para representar o roteador.

II) Selecione agora o comportamento como instantâneo e clique em “Próximo”.

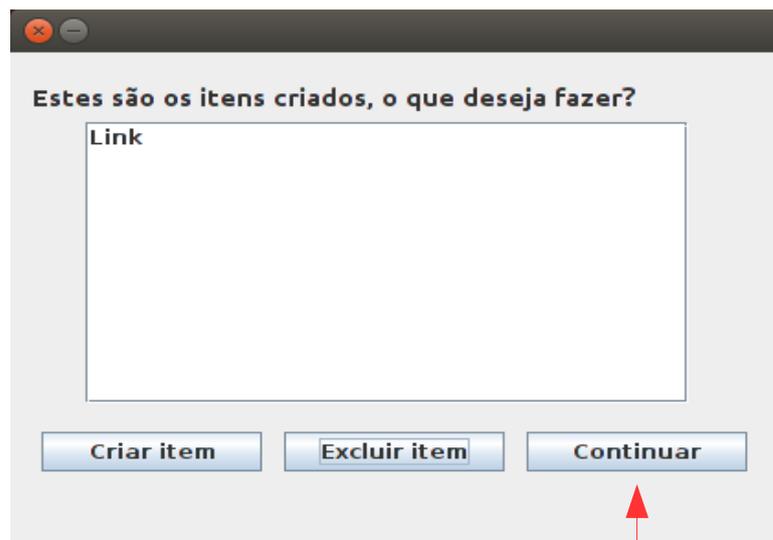


III) Deixa a função de transferência em branco, pois ela não é útil para este exemplo e clique novamente em “Próximo”

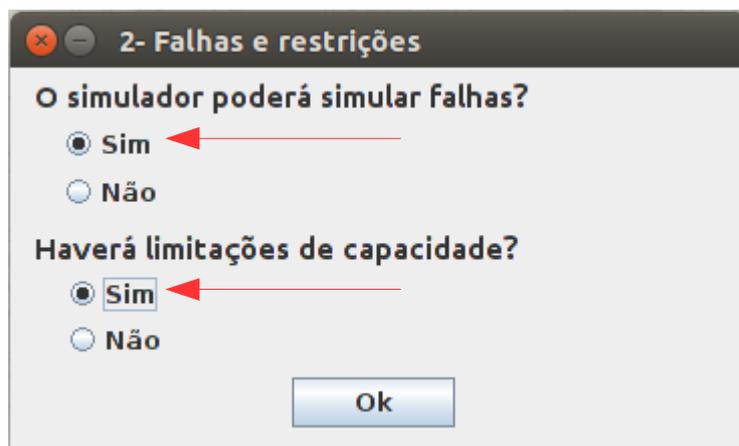


IV) A próxima tela conterá o resumo do objeto criado, apenas clique em “Finalizar”

V) Clique em criar objeto e repita todos os passos feitos para o *switch* e nó terminal até chegar novamente ao gerenciador de objetos. Desta vez clique em “Continuar”.



4º Selecione que deseja simular falhas e que o simulador deve ter restrições de capacidade e clique em “OK”



5º Forneça o nome dos eventos como “pacote”, selecione que deseja como métricas as perdas e os tempos médios, informe que a unidade de medida deles é em segundos e clique em “OK”.

3- Metricas

Dê um nome significativo que possa representar os eventos em sua simulação (Ex: Tarefa, Pacote, Cliente..):

Selecione as métricas que deseja exibir como resultados:

Perda de eventos

Tempo médio de atendimento dos eventos

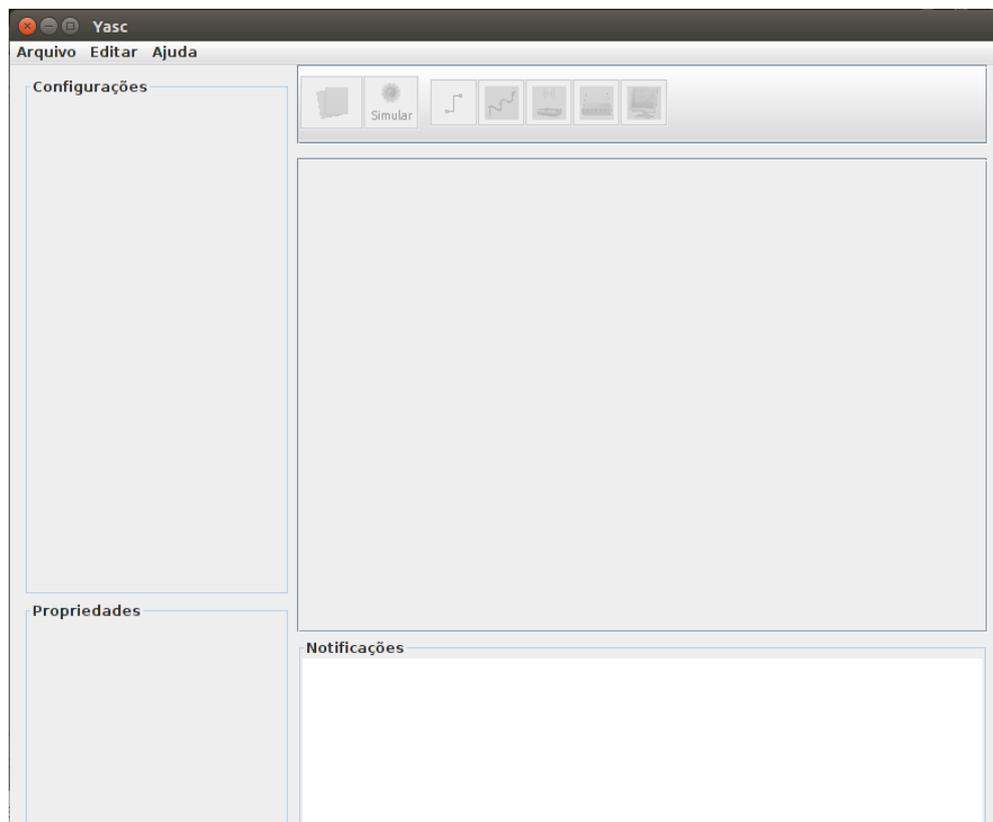
Tempo médio de espera em fila pelos eventos

Unidade de medida:

6º Por fim, confirme que sua descrição de simulador está completa e este será carregado em sua tela.

3- Como utilizar a interface icônica para criar um modelo

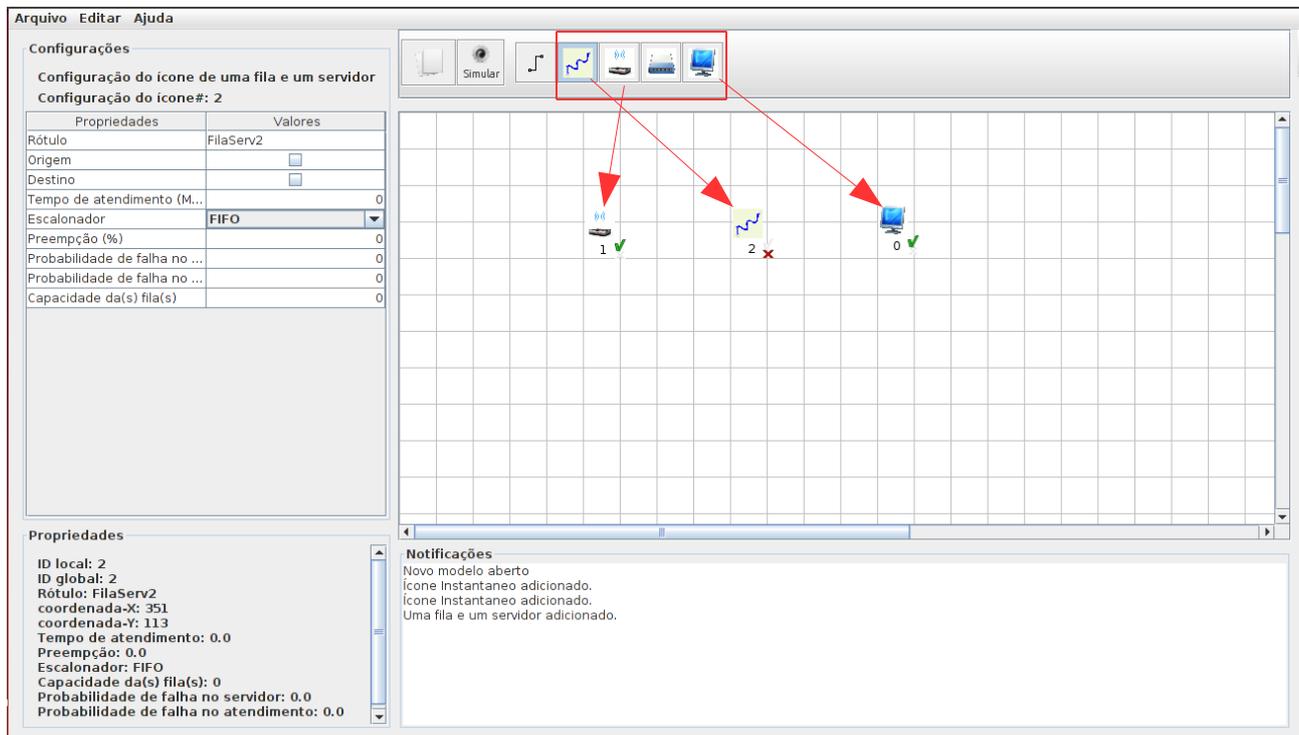
Com o simulador já pronto a seguinte tela será apresentada:



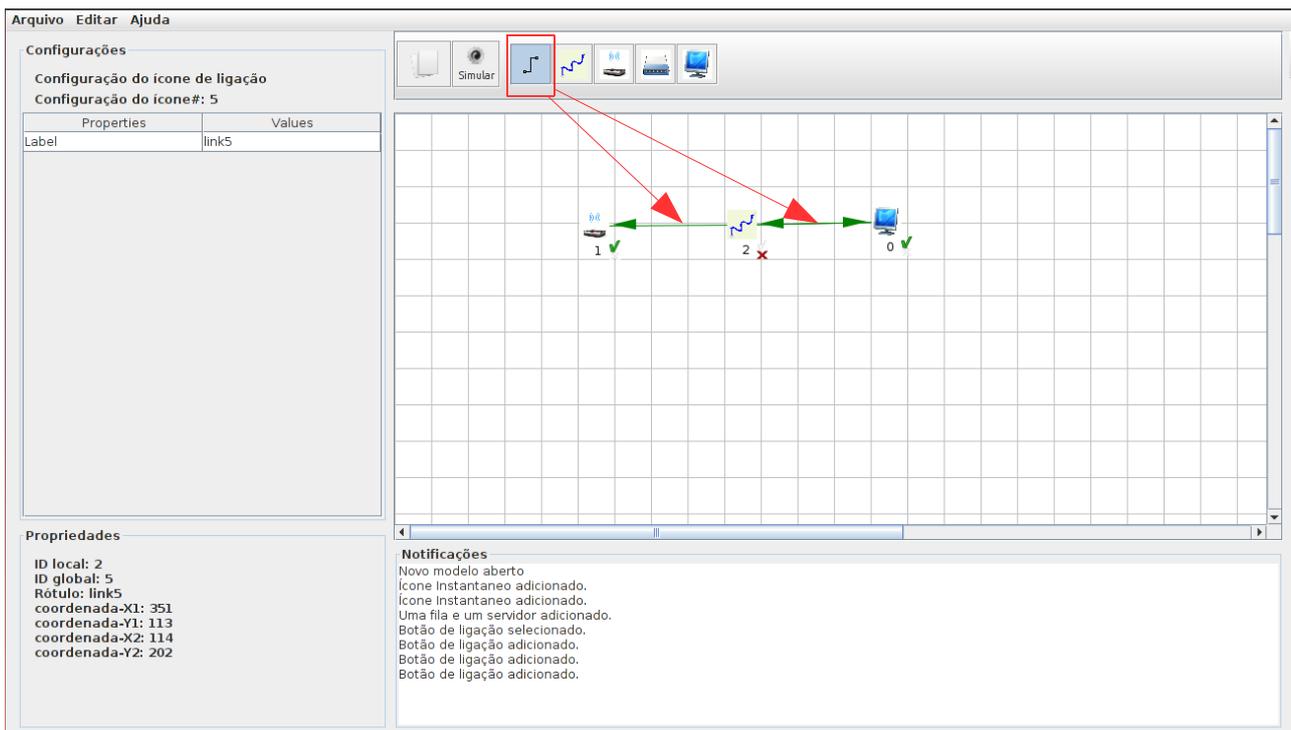
Ela representa a interface icônica do simulador criado, em que o usuário retratará o sistema real desejado por meio de um modelo, o qual será estimulado e retornará resultados relativos ao comportamento de tal sistema.

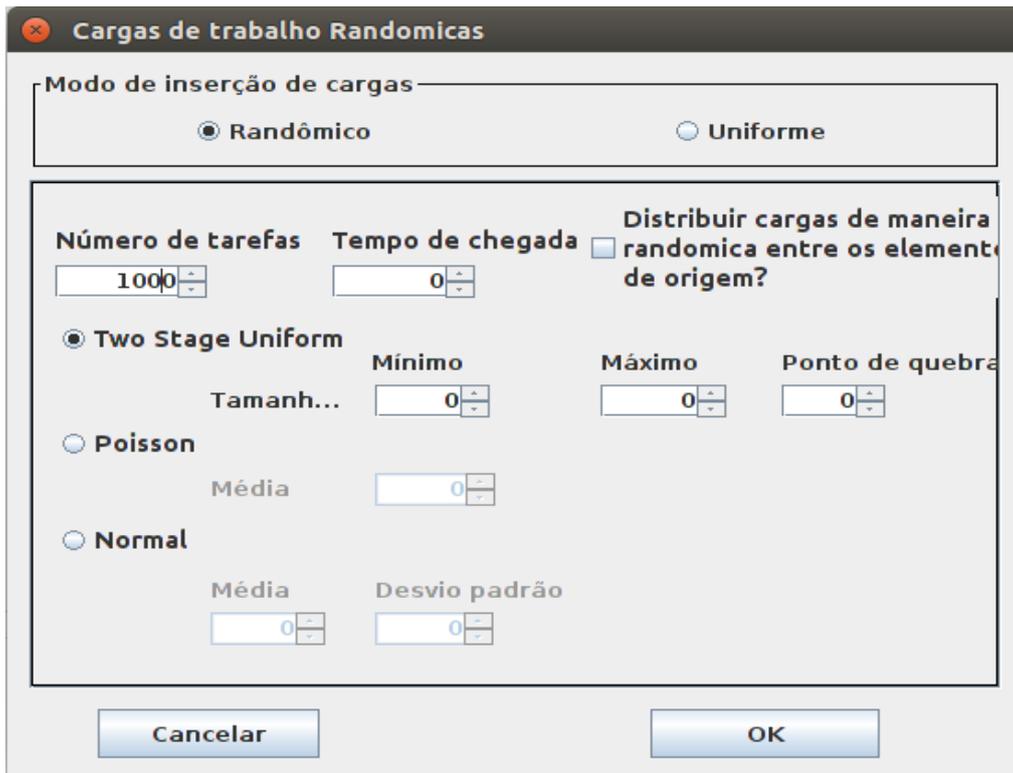
Para uso da interface icônica os seguintes passos devem ser seguidos:

1º Com um modelo novo aberto selecione o ícone desejado dentre os que estão no interior do retângulo em vermelho e os distribua na área quadriculada (Área de Desenho) da maneira desejada para melhor representar seu sistema

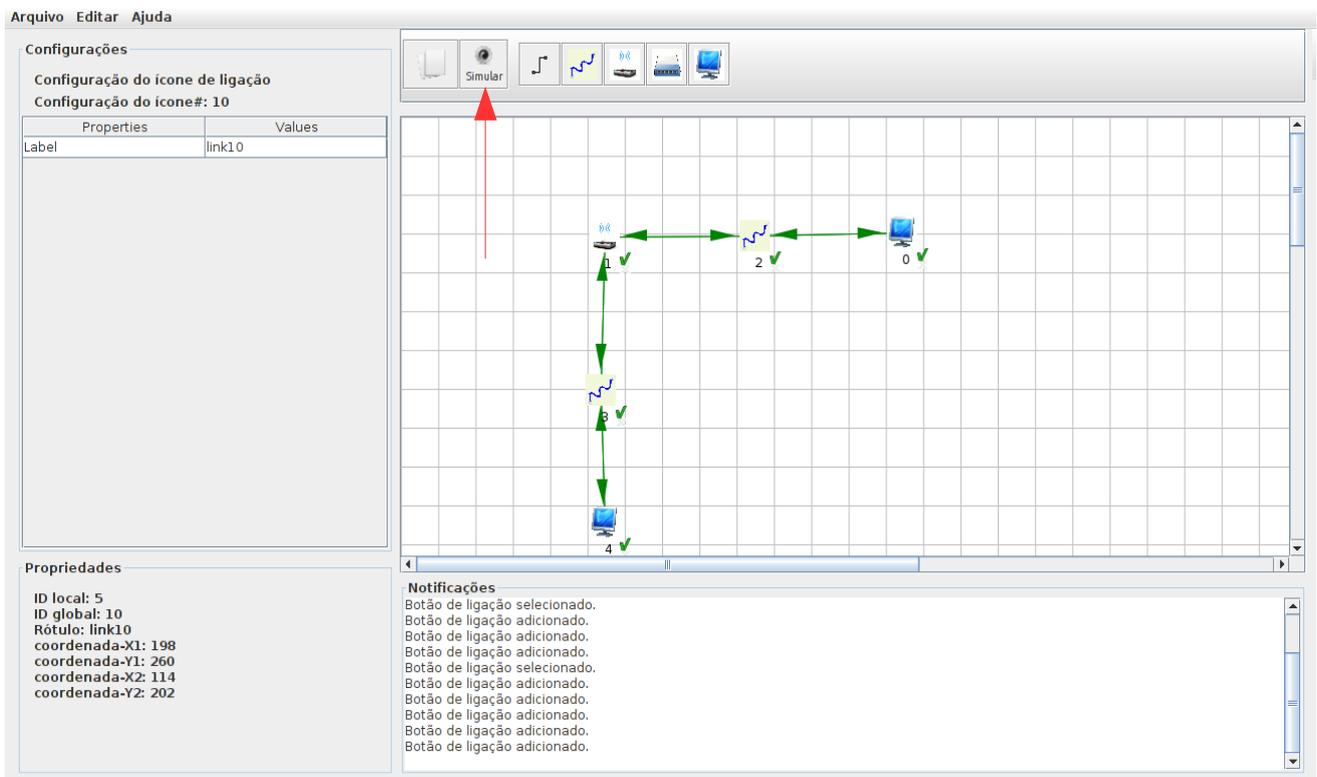


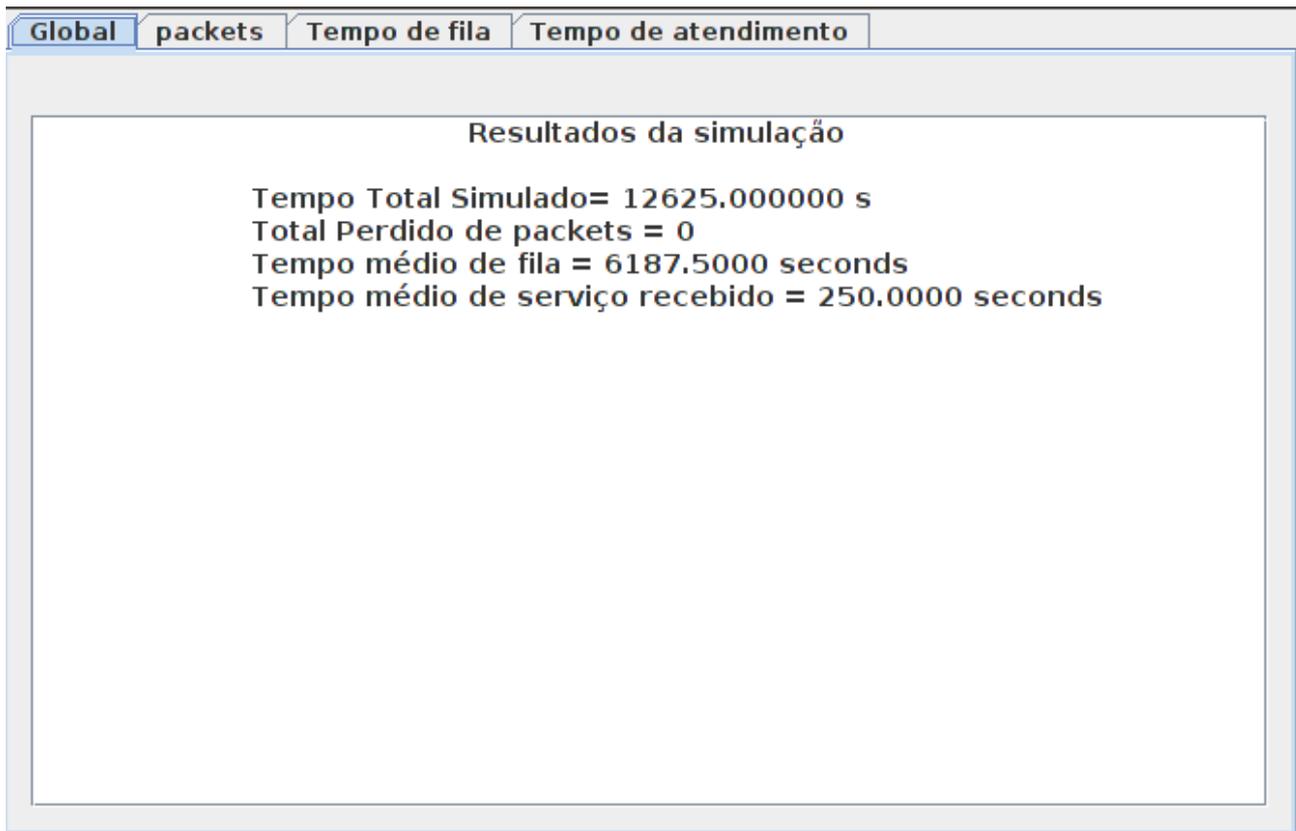
2º Ligue os ícones para indicar o sentido de propagação dos eventos, adicionando setas duplas caso haja mais de um sentido.





5º Por fim, clique em “Simular” e então ao término da simulação, caso não seja registrado nenhum erro no modelo fornecido, será apresentada uma tela com os resultados da simulação





4- Conclusão

O Yasc é um sistema que possui várias funcionalidades apesar de ainda estar em sua primeira versão. Por isto, torna-se uma tarefa muito complexa cobrir todos os seus casos de uso em um simples manual.

Espera-se que com este texto explicativo, seja apresentado ao usuário uma introdução à ferramenta, sendo esta suficiente para a compreensão de como aplicar a ferramenta as mais diversas possibilidades, direcionando e facilitando o trabalho e a pesquisa.

Equipe de Desenvolvimento do GSPD (Grupo de Sistemas Paralelos e Distribuídos).

APÊNDICE D – Validação das implementações de filas básicas

1-Validação do funcionamento das redes de filas básicas

Ao término da implementação do Yasc (Yes, a simulator's compiler), com o intuito de verificação e validação das redes de filas básicas codificadas, foi gerado um simulador para ser submetido a testes. Para a realização destes testes, utilizou-se uma aplicação presente em (MACDOUGALL, 1989), na qual é descrita uma rede básica de fila com uma fila e um servidor, conforme o código fonte em linguagem de programação C, apresentado a seguir:

```

1. #include "stdio.h"

2. main()
3. {
4.     //Ts= média de tempo de serviço por cliente
5.     //Ta=Tempo entre as chegadas
6.     //te=tempo simulado
7.     //qtd de clientes = te/ts;
8.     //B = total de tempo de servidor ocupado
9.     double Ta=200.0, Ts=100.0, te=200000.0, t1, t2,time;
10.    double B, C, L, s, tb, tn, U, W, X;
11.    int n;
12.    n = 0;
13.    t1 = 0.0;
14.    t2 = te;
15.    time = 0.0;
16.    B = s = 0.0;
17.    C = 0;
18.    tn = time;
19.    while(time<te){
20.        if(t1<t2){
21.            /* evento1: chegada*/
22.            time = t1;
23.            s = s+n*(time-tn);
24.            n++;
25.            tn=time;
26.            t1=time+randomico(Ta);
27.            if(n==1){
28.                tb=time;
29.                t2=time+ randomico(Ts);
30.            }
31.        }
32.        else{
33.            /*evento2: conclusão*/
34.            time=t2;
35.            s=s+n*(time-tn);
36.            n--;

```

```

37.         tn=time;
38.         C++;
39.         if(n>0){
40.             t2=time+randomico(Ts);
41.         }
42.         else{
43.             t2=te;
44.             B=B+time-tb;
45.         }
46.     }
47. }
48. U= B/time; //U = Utilização do servidor
49. printf("Utilização do Servidor = %lf \n", U);
50. printf("Média de tempo no servidor = %lf \n", B/C);
60. printf("Total de tempo de servidor ocupado = %lf \n", B);
61. }

```

Neste código, pode-se observar que nas linhas que vão de 9 até 18 são estabelecidos os parâmetros que foram utilizados na simulação. Dentre eles estão:

- O tempo entre as chegadas de clientes a fila, com valor de 200 unidades de tempo.
- O tempo médio de serviço prestado à cada cliente, com valor de 100 unidades de tempo.
- O tempo simulado, com valor de 200000 unidades de tempo.

Entre as linhas 19 e 47 está o laço de repetição responsável por realizar toda a simulação até que o tempo simulado de 200000 unidade finalize. Em seu interior são tratados três tipos de eventos: a chegada, o atendimento e a saída dos clientes da fila de espera, para os quais é incrementado o relógio de simulação ao longo de sua ocorrência.

Por fim, nas linhas 48 até 60 estão presentes comandos responsáveis pela impressão dos dados obtidos ao longo da simulação, os quais foram comparados com as informações obtidas no Yasc.

O mesmo modelo retirado de (MACDOUGALL, 1989) foi construído em um simulador gerado pelo Yasc para simulação de redes básicas de filas. Ele também foi simulado e teve suas métricas comparadas com as obtidas a partir do código fonte em C, como será visto na seção 2.

2-Resultados

Após a realização da simulação de ambos os modelos descritos na seção anterior as seguintes métricas foram obtidas:

- **Utilização do Servidor:** definida pela razão entre o total de tempo em que o servidor esteve ocupado e o tempo simulado obtido.
- **Média de tempo no servidor:** quantidade média de tempo que os clientes passaram recebendo serviços.
- **Total de tempo do servidor ocupado:** quantidade total do tempo que os clientes passaram no servidor.

Para a utilização do servidor, tanto o modelo retirado de (MACDOUGALL, 1989), quanto o

simulado pelo Yasc obtiveram como valor aproximado 0.5. Como valor de tempo médio no servidor, o primeiro modelo citado obteve valor 99.958 enquanto o segundo obteve valor 99.873. Por fim, como valor do total de tempo do servidor ocupado, o primeiro obteve valor de 100058 unidades de tempo contra 100060 do segundo.

As diferenças de tempo que podem ser notadas ao comparar os resultados ocorrem devido a diferença no critério de parada entre os programas de teste. O primeiro teste tem como critério de parada o tempo simulado, ou seja, o teste prossegue até que o tempo simulado chegue a 200000 unidades de tempo, restando clientes na fila no final da execução. Enquanto isso, o segundo teste tem como critério de parada o atendimento da quantidade total de clientes definidas para ele.