EDUARDO ASSIS DA SILVA

UMA FERRAMENTA PARA EXTRAÇÃO DE MEDIDAS DE DESEMPENHO NO SIMULADOR NS-3.

LAVRAS - MG

2014

EDUARDO ASSIS DA SILVA

UMA FERRAMENTA PARA EXTRAÇÃO DE MEDIDAS DE DESEMPENHO NO SIMULADOR NS-3.

Monografia apresentada ao Colegiado do Curso de Sistemas de Informação da Universidade Federal de Lavras como parte das exigências para obtenção do título de Bacharel em Sistemas de Informação.

Orientador

Prof. Dr. Neumar Costa Malheiros

LAVRAS - MG

2014

EDUARDO ASSIS DA SILVA

UMA FERRAMENTA PARA EXTRAÇÃO DE MEDIDAS DE DESEMPENHO PARA O SIMULADOR NS-3.

Trabalho de Conclusão de Curso de Graduação apresentado ao Colegiado do Curso de Bacharelado em Sistemas de Informação, para obtenção do título de Bacharel.

APROVADA em 28 de novembro de 2014.

Drª Marluce Rodrigues Pereira

José Henrique Santos Andrade

Dr. Neumar Costa Malheiros (Orientador)

LAVRAS-MG
Novembro/2014

AGRADECIMENTOS

Agradeço em primeiro lugar à minha família, em especial minha mãe, Graça de Lourdes, que sempre esteve ao meu lado desde sempre me dando amor e apoio em todas as minhas escolhas. Agradeço também aos meus irmãos e sobrinhos que foram muito importantes para que eu nunca desistisse dos meus objetivos. Agradeço também a minha namorada, Ana Flávia, pelo companheirismo, me apoiando nos momentos de dificuldades. Agradeço aos amigos de curso pelo caminho percorrido juntos, com muitas dificuldades, mas também cheio de alegrias.

RESUMO

O desempenho de sistemas computacionais é importante para que seja possível medir o rendimento de uma aplicação. Para medir o desempenho em redes de computadores são utilizadas simulações. O Network Simulator 3 (NS-3) é um dos principais softwares utilizados para realizar essas simulações afim de conseguir avaliar o desempenho de diversos arranjos de tecnologias e aplicações de redes. Entretanto as simulações do NS-3 oferecem como saída arquivos chamados de traces, que contém somente com os resgistros dos eventos ocorridos na simulação. Esses arquivos geralmente são muito densos e com muitas informações sendo inviável inferir medidas de desempenho manualmente. Sendo assim, é apresentada nesse trabalho uma ferramenta de apoio para a realização da análise desses traces, afim de calcular medidas de desempenho em redes de computadores. Ao final são apresentados os resultados da análise da ferramenta, com um estudo de caso, que mostram que a ferramenta realiza o cálculo das medidas e auxilia na comparação de diferentes experimentos.

Palavras-chave: Avaliação de desempenho; Análise de Traces; NS-3; Simulação de redes; Redes de computadores

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Medir o desempenho de sistemas de computadores é essencial pois diferentes aplicações exigem requisitos distintos. Essas medidas servem para que os desenvolvedores possam projetar suas aplicações de forma a obter o maior proveito possível da infra-estrutura disponível e realizar os tratamentos necessários para garantir um desempenho satisfatório de suas aplicações. Segundo (JAIN, 1991), "Os usuários de sistemas de computadores, analistas, engenheiros e cientistas são todos interessados em avaliação de desempenho uma vez que o seu objetivo é obter ou fornecer o mais alto desempenho com o menor custo". Portanto, para a avaliação de um sistema computacional é necessário que seja medido o desempenho desse sistema afim de se apurar a qualidade e o desempenho do serviço disponibilizado.

Para a realização de estudos em redes de computadores e sistemas distribuídos não é diferente: é necessário também a avaliação de métricas de desempenho. As medidas de desempenho podem ser calculadas, de acordo com métricas bem definidas, a partir de dados obtidos através de simulações. As simulações são muito utilizadas nas áreas de redes de computadores e sistemas distribuídos pois é muito difícil a reprodução de cenários reais para a realização dos testes.

O Network Simulator 3 (NS-3) é um dos principais softwares utilizados para realizar essas simulações afim de conseguir avaliar o desempenho de diversos arranjos de tecnologias e aplicações de redes. No entanto, o NS-3 oferece como saída arquivos chamados de *traces* que não possuem as medidas calculadas, possuem somente os registros das ocorrências dos eventos da simulação. Nesse contexto, surge a necessidade de ferramentas de apoio para a realização da análise desses *traces*.

O problema consiste em analisar os dados disponibilizados pelo simulador, organizá-los e, então realizar cálculos sobre esses dados afim de gerar informações de desempenho de acordo com métricas bem definidas como, por exemplo, a quantidade e o tamanho médio dos pacotes enviados, recebidos e descartados, a quantidade de dados enviados, recebidos e descartados, o atraso fim-a-fim por pacote, dentre outras.

O objetivo do presente trabalho é desenvolver uma ferramenta para gerar medidas de desempenho de acordo com métricas bem definidas a partir dos resultados das simulações realizadas pelo software NS-3. Os arquivos de *traces* serão analisados para que se possa inferir as medidas de interesse utilizadas pela maior parte dos pesquisadores das áreas de redes de computadores e sistemas distribuídos e gerar gráficos para apoiar a compreensão dos resultados da simulação.

O NS-3 foi escolhido pois, além de ser uma ferramenta gratuita, é também uma ferramenta nova que veio para substituir a versão anterior do software (NS-2 disponível em `http://www.isi.edu/nsnam/ns/`) e ainda não existem muitas ferramentas de apoio a ela. A necessidade de uma ferramenta de análise de desempenho é justificada pela complexidade de se analisar os arquivos (*traces*) gerados pelo simulador.

A motivação para a realização desse trabalho é a possibilidade de aprofundar os conhecimentos nos conceitos de métricas de desempenho de redes de computadores e assim poder entender melhor quais os fatores que influenciam tais métricas e também o fato de produzir uma ferramenta que poderá ser utilizada por outros pesquisadores afim de facilitar o seu trabalho na análise de novas soluções propostas. É também uma motivação poder desenvolver uma ferramenta que poderá ser

utilizada por uma empresa que queira avaliar o desempenho de sua rede interna e de suas aplicações distribuídas afim de encontrar possíveis gargalos.

Os demais capítulos deste trabalho estão estruturados da seguinte forma. O Capítulo 2 discorre sobre o Refencial Teórico que servirá de base para o entendimento do escopo do trabalho. O Capítulo 3 apresenta os trabalhos relacionados. No Capítulo 4 é apresentada a metodologia utilizada no desenvolvimento do trabalho. O Capítulo 5 descreve a ferramenta proposta. No Capítulo 6 é apresentado um estudo de caso para testar a ferramenta proposta. O Capítulo 7 é apresentada a conclusão do trabalho.

2 Referencial Teórico

O desempenho é muito importante para a utilização de um sistema computacional. É importante que um sistema ofereça alta performance a um baixo custo. Um profissional responsável por um sistema computacional deve ser capaz de avaliar requisitos de desempenho de seu sistema e comparar diferentes opções para escolher a melhor alternativa de acordo com os requisitos correspondentes. Nesse contexto, os desenvolvedores necessitam de conhecimento sobre técnicas e metodologias para avaliação de desempenho afim de apurar o desempenho do sistema.

Com relação às aplicações distribuídas, o desempenho da rede é importante para que a eficiência da aplicação possa ser assegurada. Neste capítulo, são descritos conceitos sobre avaliação de desempenho de sistemas computacionais e, em particular, avaliação de desempenho de redes de computadores.

2.1 Avaliação de Desempenho de Sistemas Computacionais

Pode-se considerar um sistema computacional qualquer conjunto de hardware, software e componentes de firmware utilizado para processar informações. Desempenho é a medida de capacidade de resposta de um sistema. Essa medida serve para que os desenvolvedores de sistemas computacionais possam encontrar possíveis gargalos e assim executar as melhorias necessárias, ou até mesmo avaliar um serviço de um cliente que necessita de um sistema computacional com garantias de performance. Como explicado em (JAIN, 1991), as principais motivações para avaliação de desempenho podem ser classificadas nos tópicos a seguir:

- Comparação de sistemas;

- Identificação de gargalos;
- Caracterização de cargas de trabalhos;
- Configuração de sistemas; e
- Estimativa de desempenho.

2.2 Como Medir o Desempenho

São discutidas, na literatura, três formas de avaliar o desempenho de um sistema, cada uma dessas possuindo suas vantagens e desvantagens com realação a outra. Essas abordagens são descritas a seguir:

- *Modelagem analítica* - conjunto de equações matemáticas que determinam o desempenho de um sistema baseando-se nos dados de entrada, chamados de parâmetros de carga. Existem várias aborgagens para se criar modelos analíticos de um sistema, entre as quais podemos destacar: leis operacionais, probabilidade e estatística, teoria das filas, teoria dos jogos, otimização e modelos matemáticos de forma geral.
- *Simulação* - consiste em imitar um comportamento preaviamente modelado, aplicando métodos matemáticos e estatísticos.
- *Medição* - é um processo de quantificação de métricas efetuado sobre um sistema já implementado. Essa quantificação é realizada sobre o sistema real, utilizando códigos fontes, software de medição dedicados e hardware de medição dedicado;

A escolha da técnica de avaliação deve considerar o nível de exatidão dos resultados e os recursos disponíveis para a avaliação do sistema, porém a principal

consideração a ser feita para decidir qual técnica de avaliação deve ser utilizada é a fase do ciclo de vida que o sistema está.

Se o sistema proposto apresentar um novo conceito e não existir sistema semelhante, então deve-se utilizar a modelagem analítica e a simulação, entretanto deve-se levar em conta que essas duas técnicas devem ser utilizadas quando não for possível utilizar a técnica de medição e saber que em geral os seus resultados seriam mais convincentes se a modelagem analítica e a simulação fossem comparadas a medições anteriores.

O desempenho de um sistema deve ser medido de forma objetiva, a partir de métricas bem definidas. Métricas de desempenho são um conjunto de parâmetros que quantificam a performance de um sistema computacional. Uma métrica é a base para que se possa chegar à medida de determinado recurso, serviço ou qualquer que seja o objeto de estudo.

2.3 Avaliação de Desempenho de Redes

Antes de falarmos sobre avaliação de desempenho em redes de computadores, vamos enteder um pouco sobre o funcionamento da comunicação que é oferecida pela Internet.

De forma sucinta, em uma comunicação entre dois sistemas computacionais utilizando uma rede de computadores, as informações trocadas entre esses dois sistemas são fragmentadas pelo sistema final de origem formando assim dados menores que são chamados de pacotes. Um pacote inicia em um sistema final (origem) e através da rede esse pacote percorre enlaces e roteadores até o sistema final (destino).

Para que seja possível que um pacote partindo da origem chegue ao seu destino são necessários comutadores. Existem dois tipos principais de comutadores: roteadores e comutadores de camada de enlace (switches). Um comutador de pacotes pode estar ligado a vários enlaces. Para cada enlace, que o comutador estiver ligado, há um buffer de saída (fila), que armazena pacotes que estejam prontos para serem enviados pelo comutador para aquele enlace. Os buffers são fundamentais, pois com eles é possível que seja mantida uma fila de pacotes enquanto um enlace estiver ocupado. No entanto, isso pode trazer problemas com atraso e perda de pacotes.

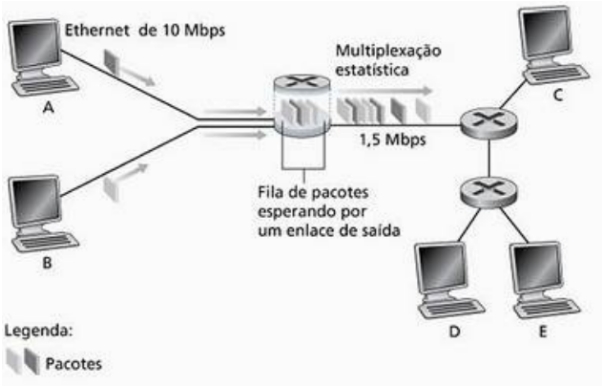

Figura 1: Comutação de pacotes

Medir o desempenho em redes de computadores é importante pois o desempenho é fundamental para que o funcionamento e eficiência de algumas aplicações sejam mantidos ou otimizados. O fato da Internet não oferecer garantias com relação aos recursos de redes nem informações sobre o desempenho da rede, medir e analisar o desempenho de redes torna-se fundamental para algumas aplicações, para que os fornecedores de aplicações possam encontrar possíveis gargalos de desempenho de uma aplicação.

Seria ideal que os serviços de Internet transferissem dados entre dois sistemas finais de modo que não houvessem nenhuma perda de informação e que essa transferência fosse instantânea. Mas isso não é possível atualmente. Ao contrário do que se considera o ideal, as redes de computadores restrigem a quantidade de dados por segundo trocados entre dois sistemas finais, possuem atrasos e também perda de pacotes. E é isso que motiva muitos pesquisadores pelo mundo a fora. Nessa seção serão abordadas algumas métricas de desempenho de redes de computadores e sistemas distribuídos.

2.3.1 Métricas de Desempenho

Métricas de desempenho de redes de computadores têm como finalidade permitir que a performance de uma rede possa ser quantificável e expressa em unidades possibilitando assim que o desempenho da rede possa ser avaliado de forma objetiva. Adiante são apresentadas algumas métricas de desempenhos, consideradas as principais, segundo (ROCHA, 2010):

Atraso É uma medida de desempenho que diz respeito ao intervalo de tempo para um pacote ser transmitido e propagado pela rede. Para mensurar o atraso na rede, são necessárias as seguintes medidas:

1. *Atraso de transmissão* é a quantidade de tempo requerida para que o reoteador transfira o pacote para o enlace, ou seja, para que o roteador empure para fora o pacote. É dado pela razão entre o tamanho do pacote e a taxa de transmissão do enlace.

   O atraso de transmissão pode ser calculado da seguinte forma:

$$at = \frac{L}{R} \tag{1}$$

, onde at é o atraso de transmissão, L é o tamanho do pacote (em bits) e R é a taxa de transmissão do enlace (em bits por segundo).

2. *Atraso unidirecional* equivale ao tempo gasto por um pacote para percorrer um caminho de rede da origem até o destino. Também chamado de atraso de propagação, é dado pela razão entre a distância de dois roteadores e a velocidade de propagação no enlace;

   O atraso unidirecional (propagação), pode ser calculado da seguinte forma:

$$ap = \frac{d}{v} \tag{2}$$

   , onde ap é o atraso de propagação, d é o comprimento do link físico e v é a velocidade de propagação no meio.

3. *Atraso de processamento* é o tempo necessário para que um roteador examine o cabeçalho do pacote e então possa determinar qual o destino desse pacote. Pode-se incluir também o tempo despendido para a verificação de erros nos bits dos pacotes que ocorrem durante a transmissão dos pacotes. Atrasos de processamento em roteadores de alta velocidade normalmente são da ordem de microssegundos ou menos.

4. *Atraso de fila* é o tempo que o pacote espera para ser transmitido no enlace. Esse atraso pode ser variável, pois depende de quantos pacotes chegaram antes de determinado pacote, ou que já estiverem na fila esperando pela transmissão. No melhor caso, quando a fila estiver vazia

e nenhum pacote estiver sendo transmitido, o atraso de fila será zero, entretanto se o tráfego estiver congestionado esse atraso será longo.

Variação de atraso (Jitter) É a diferença entre tempo de chegada de dois pacotes enviados consecutivamente e o intervalo das respectivas transmissões.

Capacidade É considerada também uma classe de medidas de desempenho e está vinculada à capacidade de transmissão de dados do sistema pela rede. Várias medidas de desempenho estão relacionadas com essa classe, como: *Largura de banda disponível*, que é a fração não utilizada da capacidade de um enlace, ou de todos os enlaces ao longo do caminho; *Throughput, ou Vazão*, que equivale à capacidade total de um canal de comunicação em processar e transferir dados entre dois sistemas finais. Em outras palavras, é o número total de pacotes enviados em um determinado intervalo de tempo; *Goodput* é a medida de dados úteis transmitidos em um canal de comunicação em um intervalo de tempo. Dados úteis são os pacotes que saíram da origem até o destino sem perdas; dentre outras.

Perda Corresponde à porcentagem de pacotes perdidos em relação ao total de pacotes enviados. A perda de pacotes em redes de computadores está associada a:

1. *Taxa de perda* é obtida pelo fração entre o número de pacotes perdidos e o número de pacotes enviados.
2. *Distribuição de perdas consecutivas* é uma estimativa para o número total de pacotes perdidos sequencialmente.

Disponibilidade É a porcentagem do tempo em que um serviço fica disponível em um intervalo de tempo de observação.

2.3.2 NS-3

O NS-3[1] é um simulador para sistemas de Internet baseado em eventos discretos, ou seja, considera-se somente os instantes que ocorrem alguma mudança no sistema causada por um evento e é focado para o desenvolvimento de pesquisas em redes de computadores. É desenvolvido como uma biblioteca C++ que fornece um conjunto de modelos/classes que auxiliam nas simulações Criado em 1989 a partir do REAL Network Simulator, projeto da Cornell University, desde então o NS-3 tem evoluído com o auxílio de algumas organizações e de desenvolvedores da comunidade de redes por ser de código aberto. Atualmente o NS-3 é mantido pela DARPA (Defense Advanced Research Projects Agency) dos EUA através do projeto AMAN e pela NSF (National Science Foundation) também dos EUA através do projeto CONSER.

Tratando-se de um simulador de eventos discretos, o NS-3 oferece como resultado após uma simulação, os registros desses eventos. Há no NS-3 vários tipos de eventos que podem ocorrer durante uma simulação. Esses eventos variam conforme o tipo de canal utilizado para realizar a comunicação entre os nós da simulação. A seguir são apresentados alguns dos tipos de eventos, retirados em (NSNAM.ORG, 2010), possíveis no NS-3 durante uma simulação:

- +: significa que uma operação de enfileiramento ocorreu na fila do dispositivo. Ou seja, um pacote foi colocado na fila da interface de transmissão;
- -: significa que uma opereração de remoção ocorreu na fila do dispositivo. Ou seja, um pacote foi retirado da fila da interface de transmissão;

[1] É possível realizar o download gratuitamente no site `http://www.nsnam.org/`.

- *d*: significa que um pacote foi perdido, descartado. Normalmente isso ocorre quando a fila estiver cheia;
- *r*: significa que um pacote foi recebido por um dispositivo de rede;

Na figura 2 é apresentado um exemplo de *trace* gerado pelo NS-3 retirado do site da ferramenta. Como já discutido, *traces* são arquivos gerados pelo NS-3 durante uma simulação que não possuem as medidas calculadas, possuem somente os registros das ocorrências dos eventos da simulação.

```
1  +
2  2
3  /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue
4  ns3::PppHeader (
5    Point-to-Point Protocol: IP (0x0021))
6    ns3::Ipv4Header (
7      tos 0x0 ttl 64 id 0 protocol 17 offset 0 flags [none]
8      length: 1052 10.1.1.1 > 10.1.1.2)
9      ns3::UdpHeader (
10       length: 1032 49153 > 9)
11       Payload (size=1024)
```

Figura 2: Exemplo de arquivo trace

Há muitas informações de forma muito densa, como pode ser visto na figura 2. No entanto vale notar que o arquivo é dividido em linhas, e que cada uma corresponde a um evento. No exemplo aqui citado as informações são apresentadas de forma que sua visualização seja mais organizada, no entanto em um arquivo de saída do NS-3, todas essas informações estão contidas em uma única linha.

São listados no arquivo os eventos da fila de transmissão entre os dispositivos de rede ponto-aponto utilizados na simulação. A fila de transmissão é uma fila em que todos os pacotes com destino a um canal ponto-a-ponto devem passar.

Para melhor entendimento, a seguir são detalhadas cada linha do exemplo da figura 2:

1. A primeira linha do arquivo corresponde à operação que ocorreu.

2. A segunda linha é tempo da simulação expresso em segundos

3. A terceira linha especifica qual a fonte de rastreamento que originou o evento. No exemplo, o nó responsável pelo evento é o nó 0 (zero), e isso é identificado pela string /NodeList/0. A string seguinte, que é /DeviceList/0, indica qual interface do nó foi usada no evento.

4. A partir da quarta linha são apresentados os cabeçalhos dos pacotes em questão. A linha 4 e 5 indicam que o pacote é encapsulado no protocolo ponto-a-ponto. Entre as linhas 6 e 8 são apresentadas informações sobre qual versão do IP o pacote possui e ainda os IP de origem e destino do pacote. As linhas 9 e 10, mostram em qual protocolo de transporte o pacote está encapsulado. No exemplo que está sendo discutido, o protocolo de transporte do pacote é o UDP. Por fim a linha 10 mostra qual o tamanho da carga da transferência é esperada.

3 Trabalhos Relacionados

Com o objetivo de entender e levantar os requisitos necessários para desenvolver a ferramenta, foi realizada uma pesquisa afim de encontrar ferramentas similares à ferramenta proposta nesse trabalho. O NS-3 oferece algumas ferramentas para auxiliar no monitoramento, apresentação de dados e estatísticas para análise das simulações de redes. A seguir são listadas algumas ferramentas relacionadas ao trabalho proposto:

- FlowMonitor

  O FlowMonitor é um módulo e também um framework que faz parte do próprio simulador para oferecer uma estrutura de monitoramento das simulações do NS-3. O FlowMonitor pode ser utilizado para coletar e armazenar dados de desempenho da rede simulada pelo NS-3.

  A seguir é mostrado um exemplo de utilização do FlowMonitor:

```
1 flowmon_helper = ns3.FlowMonitorHelper()
2 monitor = flowmon_helper.InstallAll()
3 monitor.SetAttribute( DelayBinWidth , ns3.
    DoubleValue(0.001))
4 monitor.SetAttribute( JitterBinWidth , ns3.
    DoubleValue(0.001))
5 monitor.SetAttribute( PacketSizeBinWidth , ns3.
    DoubleValue(20))
6 ns3.Simulator.Run()
7 monitor.SerializeToXmlFile( results .xml ,True,True)
```

Explicando o funcionamento do código acima:

1. Cria um objeto FlowMonitorHelper;
2. Chamada do método InstallAll do objeto flowmon_helper. O resultado dessa chamada será a criação do objeto flow monitor que é configurado para monitorar o IPv4 em todos os nós da simulação;
3. Configuração de atributos que serão usados no histograma do objeto flow monitor;
4. Idem à linha anterior;
5. Idem à linha anterior;
6. Executar a simulação;
7. Escrever os resultados monitorados para um arquivo XML chamado "results.xml". O segundo parâmetro do método SerializeToXmlFile indica se queremos também salvar o conteúdo do histograma, e o terceiro parâmetro indica se queremos salvar as estatísticas de fluxo.

A análise realizada pelo FlowMonitor (CARNEIRO; P.; RICARDO, 2009) ocorre durante a simulação, ou seja, os dados são coletados durante a execução. Todas a medidas oferecidas como saída por essa ferramenta são calculadas durante a simulação o que dá um ganho por realizar a análise de forma rápida. Entretanto o cálculo durante a simulação pode trazer desvantagens pois algumas medidas só podem ser calculadas conhecendo-se todos os eventos monitorados durante a simulação além de existir a necessidade de alterações no código do script da simulação;

- Framework Estatístico

O Framework Estatístico[2] trabalha basicamente com a manipulação de dois coletores de dados: um contador e um observer de min/max/avg/total. Assim como o FlowMonitor, ele procura um melhor desempenho e evita trabalhar com arquivos *trace*. No entanto o Framework estatístico requer mais alterações no código do script da simulação;

- TraceMetrics

  TraceMetrics (SAGGIORO; GONZAGA; RIBEIRO, 2012) é uma ferramenta de apoio ao Network Simulator 3 (NS-3) que analisa arquivos de *traces* gerados nas simulações do NS-3 e calcula algumas medidas úteis para a medição e pesquisa de performance. O TraceMetrics trabalha analisando linha a linha do arquivo de *trace* gerado pelo NS-3.

Após analisar as ferramentas já existentes nota-se que ainda existe a possibilidade de melhoria na realização de tais tarefas como calcular outras medidas, como a taxa de perda de pacatoes e o goodput, ambas agregadas por experimento, e apresentar essas medidas utilizando gráficos, com informações estatísticas e informações por fluxos de rede e facilitar a comparação entre diferentes experimentos e isso não é feito por nenhuma das ferramentas similares aqui citadas.

[2]Disponível em: `http://www.nsnam.org/docs/release/3.12/models/html/statistics.html`.

4 Metodologia

Esse projeto pode ser classificado como um trabalho de Pesquisa e Desenvolvimento (P&D) experimental. Esse tipo de P&D tem por objetivo gerar produtos e processos. Para alcançar esse objetivo é necessário utilizar conhecimento científico e prático para o desenvolvimento de novos produtos ou serviços, ou a otimização dos existentes. O trabalho proposto consiste no desenvolvimento de um novo software: uma ferramenta para extração de medidas de desempenho a partir de arquivos traces do NS-3. Essa ferramenta foi desenvolvida através da aplicação dos conhecimentos sobre avaliação de desempenho em sistemas computacionais.

Para atingir o objetivo principal do trabalho, foi necessário a realização das seguintes etapas:

1. Estudo sobre avaliação de desempenho de sistemas computacionais, redes de computadores e sistemas distribuídos.
2. Estudo da ferramenta NS-3 para compreender como são implementadas as simulações e quais são os possíveis formatos dos *traces* gerados pelo simulador.
3. Estudo de outras ferramentas relacionadas ao NS-3 e que realizam o cálculo de medidas de desempenho a partir das saídas geradas pelo NS-3.
4. Definição dos requisitos da ferramenta que foi desenvolvida.
5. Projeto da ferramenta proposta. Durante essa etapa ocorreram as decisões sobre o que, como e quando deveria ser implementada a ferramenta.
6. Implementação da ferramenta. Nessa etapa a ferramenta foi desenvolvida e para isso foi necessário o levantamento de requisitos definindo assim o

escopo do projeto para saber o que deveria ser feito e em quanto tempo. Foi durante essa etapa que foi decidida a arquitetura da ferramenta como: qual linguagem seria utilizada, quais frameworks e bibliotecas, etc.

7. Teste. Essa foi uma etapa relaizada com o intuito de avaliar a ferramenta proposta a partir de estudo de caso. O estudo de caso consistiu em realizar uma simulação no NS-3 para gerar os arquivos que são a entrada da ferramenta e assim a ferramenta realiza a análise desses arquivos gerando os gráficos com os resultados das medidas de desempenho.

5 Ferramenta Proposta

A ferramenta foi desenvolvida com o objetivo de analisar os dados (*traces* das simulações) gerados pele simulador NS-3 e inferir as informações de interesse dos usuários da mesma. É importante salientar que dentre os arquivos de *trace* gerados pelo simulador a ferramenta proposta foca no *log* do tipo rastreamento ASCII da classe PointToPointHelper, mas isso não impede que possa ser feita uma implementação para atender outros tipo de *logs* oferecidos pelo própio NS-3.

A ferramenta é capaz de realizar os cálculos de métricas de desempenho de redes de computadores e ainda apresentar os reultados aos usuários de forma simples, utilizando-se de gráficos, para a extração de informações relevantes e comparações entre simulações.

5.1 Requisitos

A análise e o levantamento de requisitos é o ponto inicial do desenvolvimento de um projeto, sendo muito importante para o sucesso das próximas etapas. Essa etapa do desenvolvimento tem como objetivo gerar um cenário propício para a obtenção de produtos de softwares de qualidade, que atendam às necessidades dos clientes com prazos e orçamentos esperados, pois facilita a comunicação entre a equipe do desenvolvimento e tem ainda a função de estreitar a comunicação dos desenvolvedores com o cliente. Nessa etapa, foram definidos quais os requisitos o sistema deveria atender para que o projeto pudesse resultar na implementação de um software que atingisse as expectativas dos potenciais usuários. Os requisitos foram divididos em funcionais e não funcionais. Os requisitos funcionais são aqueles que especificam as funcionalidades que os usuários desejam que o sistema

possua, ou seja, definem funções que o software deve executar. Eles descrevem as transformações do software de entradas em saídas. Já o requisitos não funcionais declaram características de qualidade que o sistema deve possuir e que estão relacionadas às suas funcionalidades. Referem-se às funcionalidades implícitas de um sistema, geralmente os usuários só as percebem quando estão ausentes ou com problemas. Como o presente projeto diz respeito ao desenvolvimento de uma ferramenta de software essa seção é dedicada ao levantamento dos requisitos.

5.1.1 Prioridade dos requisitos

Para estabelecer uma prioridade entre os requisitos levantados, foram adotadas as denominações: essencial, importante e desejável. Abaixo segue a descrição de cada uma:

Essencial	É o requisito sem o qual o sistema não irá funcionar. Esse é o requisito que deve ser implementado impreterivelmente.
Importante	É o requisito sem o qual o sistema funcionará, entretanto não será de forma satisfatória. Esse tipo de requisito deve ser implementado, mas sem eles o sistema poderá ser implantado e usado mesmo assim.
Desejável	É o requisito sem o qual o sistema funciona de forma satisfatória. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

Tabela 1: Prioridade de Requisitos

5.1.2 Requisitos Funcionais

A seguir são listados os requisitos funcionais da ferramenta:

RF 01– Leitura de arquivo com os *traces*: O sistema oferece a funcionalidade de carregar os arquivos que contenham os registros dos eventos gerados durante a simulação do NS-3. O usuário poderá carregar mais de um arquivo afim de realizar comparativos entre diferentes simulações. Para isso os arquivos devem ser organizados em pastas com o nome do experimento, sendo que cada arquivo corresponde a uma execução de determinado experimento.

Prioridade: *Essencial*

RF 02– Cálculo de Goodput: O sistema oferece como uma das saídas o cálculo do goodput da rede. O goodput é a medida da quantidade de dados transferidos em um determinado arranjo de rede durante um determinado período de tempo, mas diferente do throughput, essa medida desconsidera as perdas de pacotes ocorridas durante a transfrência. Sendo assim, o goodput é uma métrica para calcular a quantidade real de pacotes que saíram da origem e que efetivamente chegaram no destino. O cálculo do goodput deverá ser realizado por execução ou agregado por execuções, ou seja, que é calculado somando os resultados de todas as execuções e calculando a média e também o desvio padrão.

Prioridade: *Importante*

RF 03– Cálculo da Taxa de perda: O sistema oferece também como uma das saídas o cálculo da taxa de perda dos pacotes ocorrida durante a simulação. A perda corresponde à porcentagem de pacotes perdidos em relação ao total de pacotes enviados. O cálculo da taxa de perda deverá ser realizado considerando os valores agregados das execuções, ou seja, somando os resultados de todas as execuções e calculando a média e também o desvio padrão.

Prioridade: *Importante*

RF 04– Cálculo do Atraso: O sistema oferecerá ainda o atraso, medida de desempenho já discutida no Capítulo 2.

Prioridade: *Importante*

RF 05– Variação de atraso (Jitter): Será também calculado a variação do atraso (jitter) que é a diferença entre tempo de chegada de dois pacotes enviados consecutivamente e o intervalo das respectivas transmissões. **Prioridade:** *Importante*

RF 06– Geração de Relatórios: São gerados como saídas relatórios gráficos para cada experimento, levando-se em conta todas as suas execuções, possibilitando assim a realização de comparações entre diferentes experimentos realizados.

Prioridade: *Importante*

Obs.: Todas as métricas são calculadas por fluxo de comunicação, ou seja, são considerados nos cálculos dessas métricas somente os eventos ocorridos na origem e no destino.

5.1.3 Diagrama de casos de uso

A seguir é apresentado um diagrama de casos de uso utilizado para auxiliar no levantamento dos requisitos funcionais:

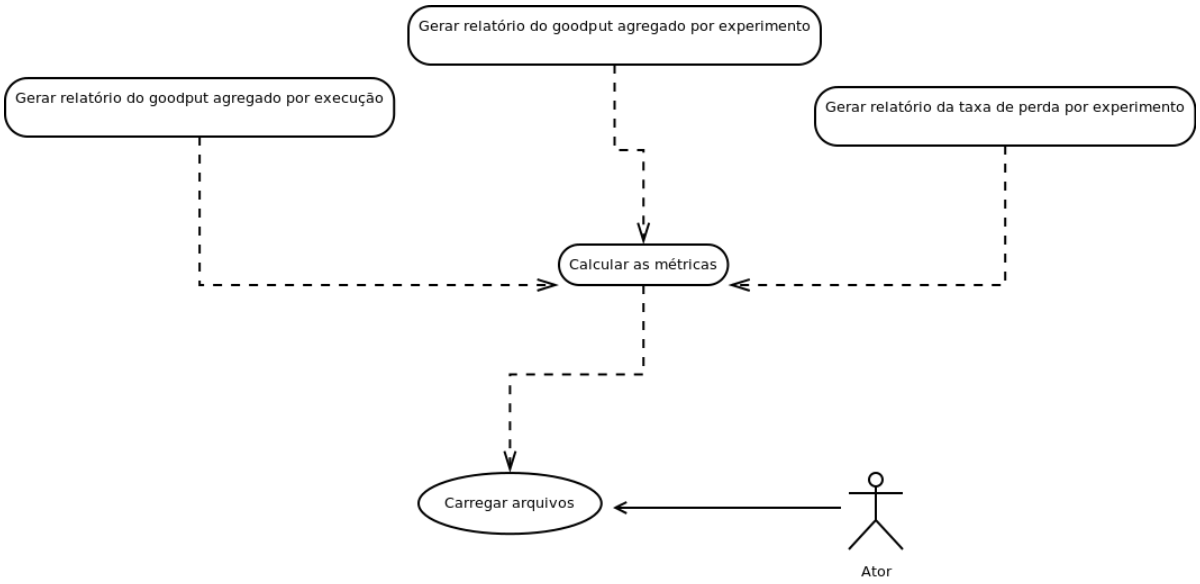

Figura 3: Diagrama de casos de uso

5.1.4 Requisitos Não Funcionais

RNF 01– Tempo de Resposta: A análise dos arquivos deve ser realizada em um tempo aceitável. Usa-se aqui a palavra aceitável pois o tempo da análise irá depender do tamanho do arquivo que é variável, vai depender de cada simulação.

Prioridade: *Desejável*

RNF 02– Interface Amigável: A ferramenta deve possuir uma interface simples e de fácil usabilidade, beneficiando a navegação do usuário pelo menu oferecido.

Prioridade: *Importante*

Dos requisitos listados acima, somente os requisitos *RF 04* e *RF 05* não foram implementados. Entretanto nota-se que há a necessidade de algumas melhorias com relação a interface, a entrada dos arquivos no sistema e ao cálculo de um número maior medidas estatísticas como, por exemplo, o intervalo de confiança. A interface da ferramenta deve melhorar no sentido estético, devem ser implementados um mecanismo de upload dos arquivos e o cálculo de mais métricas de desempenho.

5.2 Arquitetura

Após o levantamento dos requisitos foi iniciado o desenvolvimento da ferramenta. Para isso foram utilizadas tecnologias em que o autor já possuia algum conhecimento.

A linguagem de programação escolhida foi o JavaScript tendo como plataforma de desenvolvimento o node.js[3], que permite o desevolvimento de aplicações de rede que podem ser acessadas via interface web. É uma aplicação que roda localmente e que disponibiliza serviços para um cliente que também é local. Entretanto, mesmo tendo uma aplicação provendo serviços locais, e a ferramenta sendo de uso local, nada impede disponibilizá-la de forma distribuída, ou seja, de forma que possa ser acessada de uma máquina diferente como em uma arquitetura cliente/servidor. A interface é toda desenvolvida utilizando HTML, portanto para renderizar as páginas HTML é necessário um browser para realizar tal tarefa. Tinha-se duas opções: executar em navegador comum, já instalado na máquina de utilização ou a utilização do TopCube[4] que é uma aplicação baseada no Chromium e node.js. Essa abordagem foi adotada pois assim é possível desenvolver uma fer-

[3]Disponível em: `http://nodejs.org/`.
[4]Disponível em: `https://github.com/creationix/topcube`.

ramenta desktop utilizando-se as facilidades das tecnologias do desenvolvimento WEB. Para geração dos gráficos foi utilizada a biblioteca Chart.js[5]

[5]Disponível em: http://www.chartjs.org/

6 Estudo de Caso

Com a finalidade de testar e assim poder avaliar os resultados das saídas da ferramenta desenvolvida foi realizado um estudo de caso. Para isso foi necessário a implementação de uma simulação para que fosse possível gerar os arquivos que serviriam de entrada para a análise. Apartir de um exemplo do NS-3, foi implementada uma simulação para avaliar três variações do protocolo TCP:

- Tcp New Reno
- Tcp Tahoe
- Tcp Westwood

Esses algoritmos diferem entre si no que diz respeito aos mecanismos de controle e prevenção de congestionamento.

É importante salientar a dificuldade encontrada durante essa etapa, pois a ferramenta visa extrair as métricas por fluxos de rede, e para tal é necessário capturar durante a simulação somente os eventos da origem e do destino do fluxo. Como o objetivo do projeto não era a implementação de simulações no NS-3, e sim a sua análise, a simulação foi alterada para conter somente um fluxo com somente dois nós, origem e destino, e um roteador (comutador), interligando esses dois nós, pois assim fica mais evidente os atrasos e perdas de pacotes. No entanto, os eventos do roteador não poderiam ser considerados na realização dos cálculos das métricas, pois senão os eventos seriam considerados mais vezes do que devem ser de fato. Então foi realizado um pré-processamento dos arquivos de traces para que fossem removidos os eventos do roteador. Porém antes de realizar esse pré-processamento, foi realizada uma pesquisa para verificar se é possível através da

simulação no NS-3 capturar somente os eventos dos nós origem e destino e foi constado que é possível sim, porém não é uma implementação simples e trivial e visto que o objetivo do projeto é a ferramenta de análise foi considerado aceitável tal pré-processamento dos arquivos de traces. Esse pré-processamento é descrito logo a seguir.

Vale reforçar que sendo assim qualquer um que queria utilizar a ferramenta desenvolvida poderá utilizá-la desde que realize sua implementação considerando somente os eventos ocorridos nas origens e destinos de cada fluxo de sua simulação.

O código utilizado para a simulação segue anexado a esta trabalho. A simulação foi executada 15 (quinze) vezes para cada uma das implementações do Tcp citadas anteriormente a uma taxa de erro de 10%. Para isso foi criado um script, utilizando-se a linguagem shell script, para automatizar a chamada da execução. O código shell segue anexado a este trabalho.

Pode-se notar o processamento realizado no arquivo de saída da simulação com o comando “sed -i ’/NodeList/0/DeviceList/d”’ que retira do arquivo todas as linhas com a ocorrência desse padrão especificado na regex. Através de uma análise nos arquivos conclui-se que o roteador é o NodeList 0, portanto ele deveria ser descartado do arquivo antes de análise ser realizada.

O script de execução da simulação move todos os arquivos gerados para uma pasta na área de trabalho do computador e é essa pasta que será a entrada para a ferramenta. Atualmente não foi implementada a funcionalidade de realizar o upload dos arquivos portanto está configurado para que a ferramenta leia os arquivos a partir dessa pasta. É uma configuração feita no código da aplicação e se for de interesse colocar os arquivos em outro diretório é necessário ter em mente

que isso acarretará uma mudança no código. Também deve ser montada a mesma estrutura de pasta: o nome da pasta é o nome do experimento e todas as execuções desse experimento devem ir para essa pasta.

6.1 Resultados da Análise

Após executar a simulação, foi executada a análise dos arquivos a partir da ferramenta desenvolvida. A unidade de medida utilizada nos gráficos para demonstrar os resultados foi MB/s (megabyte por segundo). Na ferramenta, é possível visualizar essa informação ao passar o mouse sobre os gráficos. A seguir serão apresentados os resultados dessa análise:

- *Goodput agregado por execução* é o somatório do goodput de todos os fluxos de uma determinada execução.

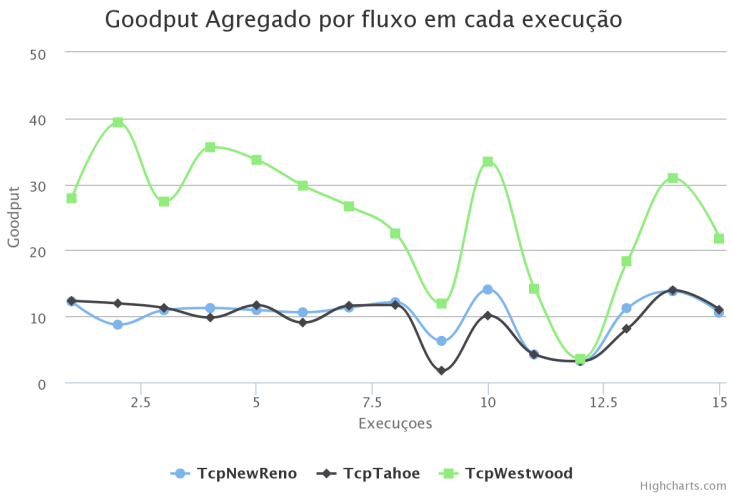

Figura 4: Relatório de goodput agregado por execução do experimento

- *Média do goodput agregado por experimento* é o somatório do goodput de todas as execuções do experimento dividido pelo número de execuções.

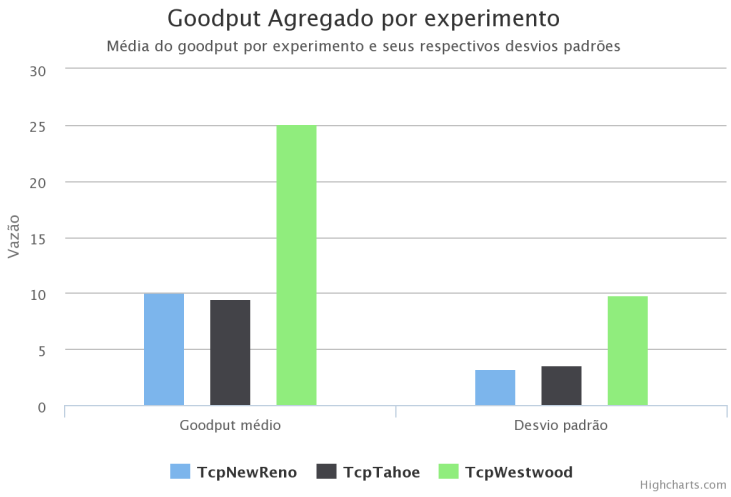

Figura 5: Média do goodput agregado por experimento

- *Taxa de perda por experimento* é o somatório da taxa de perda de todas as execuções do experimento dividido pelo número de execuções.

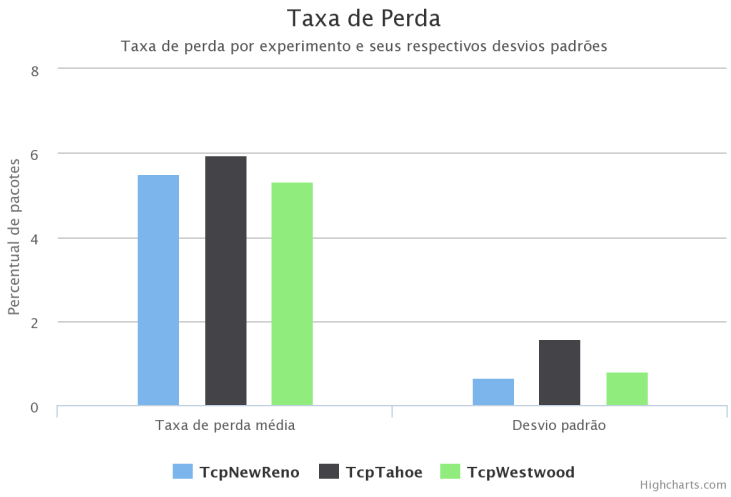

Figura 6: Taxa de perda por experimento

Pode-se notar que os requisitos **RF 02** (Cálculo do goodput), **RF 03** (Cálculo da taxa de perda) e **RF 06** (Relatórios dos resultados) foram implementados e com o estudo de caso foi possível visualizar as saídas geradas pela ferramenta desenvolvida.

7 Conclusão

Com o presente trabalho foi possível aprender como avaliar um sistema computacional com relação ao seu desempenho aplicando métricas definidas relacionadas à um determinado sistema e inferindo assim as medidas de desempenho de interesse. A ferramenta desenvolvida consegue realizar esses cálculos para a avaliação de desempenho de redes de computadores a partir de dados gerados por simulações do simulador NS-3. Foi possível ao final do desenvolvimento da ferramenta apresentar algumas medidas de desempenho realizando comparativos entre diferentes experimentos. As maiores difuculdades encontradas foram com relação ao desenvolvimento do cenário necessário para realização do estudo de caso aqui apresentado no que diz respeito à implementação da simulação no simulador NS-3.

A ferramenta desenvolvida vem complementar a ferramenta similar, no caso o TraceMetrics, citado no capítulo de trabalhos relacionados. O TraceMetrics é uma ferramenta de análise de *traces* do NS-3 que calcula métricas de desempenho entretanto realiza a análise de um arquivo de saída por vez. Ele não necessita realizar nenhum processamento nos *traces* do NS-3 porém não gera gráficos com base na análise realizada, disponibiliza somente os dados tabulados sendo necessário a utilização de outra ferramenta, por exemplo o gnuplot, para gerar os gráficos. Ao contrário do TraceMetrics, a ferramenta desenvolvida necessita realizar um processamento nos *traces* para depois analisá-los, entretanto, realiza a análise de múltiplos arquivos, organizando-os por execuções de um experimento afim de proporcionar dados comparativos entre experimentos o que torna os resultados muito utéis para validação e comparação de simulações, como no estudo de caso descrito

neste trabalho, e apresenta os resultados utilizando gráficos, sem a necessidade de utilizar outra ferramenta.

Para um trabalho futuro, seria válido a implementação de um formato de trace que possa ser utilizado em qualquer simulação do NS-3 para que somente os eventos ocorridos na origem e destino (por fluxo) sejam considerados. Assim, não seria mais necessário um pré-processamento dos *traces*. Há ainda a possibilidade de calcular um maior número de medidas de desempenho como os requisitos **RF 04** que é o Atraso e **RF 05** que é a Variação do atraso ou *Jitter*, que não foram implementados durante o desenvolvimento desse projeto, e também outras medidias estatísticas, como cálculo do intervalo de confiança, que facilitarão ainda mais a avaliação e comparação dos experimentos realizados.

Referências

CARNEIRO, G.; P., F.; RICARDO, M. Flowmonitor - a network monitoring framework for the network simulator 3 (ns-3). 2009.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* [S.l.]: Wiley Computer Publishing, John Wiley & Sons, Inc., 1991. 714 p.

NSNAM.ORG. *Using the Tracing System.* [S.l.], August 2010. Disponível em: <www.nsnam.org/docs/release/3.9/tutorial/tutoriale 23.html>.

ROCHA, A. A. de A. Sobre medidas de desempenho da internet para o uso em aplicações de redes. 2010. Disponível em: <https://www.lume.ufrgs.br/bitstream-/handle/10183/15972% -/000695256.pdf?sequence=1>.

SAGGIORO, L. F. Z.; GONZAGA, F. B.; RIBEIRO, R. R. Tracemetrics - uma ferramenta para a obtenção de medidas de interesse no ns-3. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2012.

8 Anexos

8.1 Código utilizado para automatização da simulação

```
#!/bin/bash

cd repos/ns-3-allinone/ns-3-dev/

LIMITE=15

for ((i=1; i <= $LIMITE ; i++))
do

        ## TcpTahoe

        ./waf --run "examples/tcp/tcp-variants-comparison --transport_prot=TcpTahoe
        --tr_name=TcpTahoe_$i.tr --run=$i --error_p=0.10


        mv TcpTahoe_"$i".tr ~/Área\ de\ Trabalho/results/TcpTahoe/

        sed -i '/NodeList\/0\/DeviceList\//d'
        ~/Área\ de\ Trabalho/results/TcpTahoe/TcpTahoe_"$i".tr


        ## TcpNewReno

        ./waf --run "examples/tcp/tcp-variants-comparison --transport_prot=TcpNewReno
        --tr_name=TcpNewReno_$i.tr --run=$i --error_p=0.10
```

```
        sed -i '/NodeList\/0\/DeviceList\//d'
        ~/Área\ de\ Trabalho/results/TcpNewReno/TcpNewReno_"$i".tr


        ## TcpWestwood

        ./waf --run "examples/tcp/tcp-variants-comparison --transport_prot=TcpWestwood
        --tr_name=TcpWestwood_$i.tr --run=$i --error_p=0.10


        mv TcpWestwood_"$i".tr ~/Área\ de\ Trabalho/results/TcpWestwood/

        sed -i '/NodeList\/0\/DeviceList\//d'
        ~/Área\ de\ Trabalho/results/TcpWestwood/TcpWestwood_"$i".tr

done
```

8.2 Código da implementação da simulação

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4
5 #include "ns3/core-module.h"
6 #include "ns3/network-module.h"
7 #include "ns3/internet-module.h"
8 #include "ns3/point-to-point-module.h"
```

```
9  #include "ns3/applications-module.h"
10 #include "ns3/error-model.h"
11 #include "ns3/tcp-header.h"
12 #include "ns3/udp-header.h"
13 #include "ns3/enum.h"
14 #include "ns3/event-id.h"
15 #include "ns3/flow-monitor-helper.h"
16 #include "ns3/ipv4-global-routing-helper.h"
17
18 using namespace ns3;
19
20 NS_LOG_COMPONENT_DEFINE ("TcpVariantsComparison");
21
22 double old_time = 0.0;
23 EventId output;
24 Time current = Time::FromInteger(3, Time::S);
25 bool first = true;
26
27 static void
28 OutputTrace ()
29 {
30 }
31
32 static void
33 CwndTracer (Ptr<OutputStreamWrapper>stream, uint32_t oldval
      , uint32_t newval)
34 {
35   double new_time = Simulator::Now().GetSeconds();
36   if (old_time == 0 && first)
37   {
```

```
38      double mycurrent = current.GetSeconds();
39      *stream->GetStream() << new_time << " " << mycurrent <<
            " " << newval << std::endl;
40      first = false;
41      output = Simulator::Schedule(current,&OutputTrace);
42    }
43    else
44    {
45      if (output.IsExpired())
46      {
47        *stream->GetStream() << new_time << " " << newval <<
              std::endl;
48        output.Cancel();
49        output = Simulator::Schedule(current,&OutputTrace);
50      }
51    }
52  }
53
54  static void
55  SsThreshTracer (Ptr<OutputStreamWrapper>stream, uint32_t
      oldval, uint32_t newval)
56  {
57    double new_time = Simulator::Now().GetSeconds();
58    if (old_time == 0 && first)
59    {
60      double mycurrent = current.GetSeconds();
61      *stream->GetStream() << new_time << " " << mycurrent <<
            " " << newval << std::endl;
62      first = false;
63      output = Simulator::Schedule(current,&OutputTrace);
```

```
64    }
65    else
66    {
67      if (output.IsExpired())
68      {
69        *stream->GetStream() << new_time << " " << newval <<
              std::endl;
70        output.Cancel();
71        output = Simulator::Schedule(current,&OutputTrace);
72      }
73    }
74  }
75
76  static void
77  TraceCwnd (std::string cwnd_tr_file_name)
78  {
79    AsciiTraceHelper ascii;
80    if (cwnd_tr_file_name.compare("") == 0)
81       {
82         NS_LOG_DEBUG ("No trace file for cwnd provided");
83         return;
84       }
85    else
86      {
87        Ptr<OutputStreamWrapper> stream = ascii.
              CreateFileStream(cwnd_tr_file_name.c_str());
88        Config::ConnectWithoutContext ("/NodeList/1/$ns3::
              TcpL4Protocol/SocketList/0/CongestionWindow",
              MakeBoundCallback (&CwndTracer, stream));
89      }
```

```
90  }
91
92  static void
93  TraceSsThresh(std::string ssthresh_tr_file_name)
94  {
95    AsciiTraceHelper ascii;
96    if (ssthresh_tr_file_name.compare("") == 0)
97      {
98        NS_LOG_DEBUG ("No trace file for ssthresh provided");
99        return;
100     }
101   else
102     {
103       Ptr<OutputStreamWrapper> stream = ascii.
              CreateFileStream(ssthresh_tr_file_name.c_str());
104       Config::ConnectWithoutContext ("/NodeList/1/$ns3::
              TcpL4Protocol/SocketList/0/SlowStartThreshold",
              MakeBoundCallback (&SsThreshTracer, stream));
105     }
106 }
107
108 int main (int argc, char *argv[])
109 {
110
111   std::string transport_prot = "TcpWestwood";
112   double error_p = 0.0;
113   std::string bandwidth = "2Mbps";
114   std::string access_bandwidth = "10Mbps";
115   std::string access_delay = "45ms";
116   bool tracing = true;
```

```
117   std::string tr_file_name = "teste.tr";
118   std::string cwnd_tr_file_name = "cwnd_tr_name.tr";
119   std::string ssthresh_tr_file_name = "teste_sss_";
120   double data_mbytes = 1000;
121   uint32_t mtu_bytes = 1500;
122   uint16_t num_flows = 1;
123   float duration = 30;
124   uint32_t run = 0;
125   bool flow_monitor = true;
126   std::string flow_monitor_file = "flowMonitor";
127
128   CommandLine cmd;
129   cmd.AddValue("transport_prot", "Transport protocol to use
          : TcpTahoe, TcpReno, TcpNewReno, TcpWestwood,
          TcpWestwoodPlus ", transport_prot);
130   cmd.AddValue("error_p", "Packet error rate", error_p);
131   cmd.AddValue("bandwidth", "Bottleneck bandwidth",
          bandwidth);
132   cmd.AddValue("access_bandwidth", "Access link bandwidth",
           access_bandwidth);
133   cmd.AddValue("delay", "Access link delay", access_delay);
134   cmd.AddValue("tracing", "Flag to enable/disable tracing",
           tracing);
135   cmd.AddValue("tr_name", "Name of output trace file",
          tr_file_name);
136   cmd.AddValue("cwnd_tr_name", "Name of output trace file",
           cwnd_tr_file_name);
137   cmd.AddValue("ssthresh_tr_name", "Name of output trace
          file", ssthresh_tr_file_name);
```

```
138   cmd.AddValue("data", "Number of Megabytes of data to
          transmit", data_mbytes);
139   cmd.AddValue("mtu", "Size of IP packets to send in bytes"
          , mtu_bytes);
140   cmd.AddValue("num_flows", "Number of flows", num_flows);
141   cmd.AddValue("duration", "Time to allow flows to run in
          seconds", duration);
142   cmd.AddValue("run", "Run index (for setting repeatable
          seeds)", run);
143   cmd.AddValue("flow_monitor", "Enable flow monitor",
          flow_monitor);
144   cmd.AddValue("flow_monitor_file", "Name of output
          flowmonitor", flow_monitor_file);
145   cmd.Parse (argc, argv);
146
147   SeedManager::SetSeed(1);
148   SeedManager::SetRun(run);
149
150     // Calculate the ADU size
151   Header* temp_header = new Ipv4Header();
152   uint32_t ip_header = temp_header->GetSerializedSize();
153   NS_LOG_LOGIC ("IP Header size is: " << ip_header);
154   delete temp_header;
155   temp_header = new TcpHeader();
156   uint32_t tcp_header = temp_header->GetSerializedSize();
157   NS_LOG_LOGIC ("TCP Header size is: " << tcp_header);
158   delete temp_header;
159   uint32_t tcp_adu_size = mtu_bytes - (ip_header +
          tcp_header);
160   NS_LOG_LOGIC ("TCP ADU size is: " << tcp_adu_size);
```

```
161
162   // Set the simulation start and stop time
163   float start_time = 0.1;
164   float stop_time = start_time + duration;
165
166   // Select TCP variant
167   if (transport_prot.compare("TcpTahoe") == 0)
168     Config::SetDefault("ns3::TcpL4Protocol::SocketType",
          TypeIdValue (TcpTahoe::GetTypeId()));
169   else if (transport_prot.compare("TcpReno") == 0)
170     Config::SetDefault("ns3::TcpL4Protocol::SocketType",
          TypeIdValue (TcpReno::GetTypeId()));
171   else if (transport_prot.compare("TcpNewReno") == 0)
172     Config::SetDefault("ns3::TcpL4Protocol::SocketType",
          TypeIdValue (TcpNewReno::GetTypeId()));
173   else if (transport_prot.compare("TcpWestwood") == 0)
174     {// the default protocol type in ns3::TcpWestwood is
          WESTWOOD
175       Config::SetDefault("ns3::TcpL4Protocol::SocketType",
            TypeIdValue (TcpWestwood::GetTypeId()));
176       Config::SetDefault("ns3::TcpWestwood::FilterType",
            EnumValue(TcpWestwood::TUSTIN));
177     }
178   else if (transport_prot.compare("TcpWestwoodPlus") == 0)
179     {
180       Config::SetDefault("ns3::TcpL4Protocol::SocketType",
            TypeIdValue (TcpWestwood::GetTypeId()));
181       Config::SetDefault("ns3::TcpWestwood::ProtocolType",
            EnumValue(TcpWestwood::WESTWOODPLUS));
```

```
182       Config::SetDefault("ns3::TcpWestwood::FilterType",
              EnumValue(TcpWestwood::TUSTIN));
183     }
184   else
185     {
186       NS_LOG_DEBUG ("Invalid TCP version");
187       exit (1);
188     }
189
190   NodeContainer gateways;
191   gateways.Create (num_flows);
192   NodeContainer sources;
193   sources.Create(num_flows);
194   NodeContainer sinks;
195   sinks.Create(num_flows);
196
197   // Configure the error model
198   // Here we use RateErrorModel with packet error rate
199   Ptr<UniformRandomVariable> uv = CreateObject<
          UniformRandomVariable>();
200   uv->SetStream (50);
201   RateErrorModel error_model;
202   error_model.SetRandomVariable(uv);
203   error_model.SetUnit(RateErrorModel::ERROR_UNIT_PACKET);
204   error_model.SetRate(error_p);
205
206   PointToPointHelper UnReLink;
207   UnReLink.SetDeviceAttribute ("DataRate", StringValue (
          bandwidth));
```

```
208   UnReLink.SetChannelAttribute ("Delay", StringValue ("0.01
          ms"));
209   UnReLink.SetDeviceAttribute ("ReceiveErrorModel",
          PointerValue (&error_model));
210
211
212   InternetStackHelper stack;
213   stack.InstallAll ();
214
215   Ipv4AddressHelper address;
216   address.SetBase ("10.0.0.0", "255.255.255.0");
217
218   // Configure the sources and sinks net devices
219   // and the channels between the sources/sinks and the
          gateways
220   PointToPointHelper LocalLink;
221   LocalLink.SetDeviceAttribute ("DataRate", StringValue (
          access_bandwidth));
222   LocalLink.SetChannelAttribute ("Delay", StringValue (
          access_delay));
223   Ipv4InterfaceContainer sink_interfaces;
224
225   for (int i=0; i<num_flows; i++)
226     {
227
228       NetDeviceContainer devices;
229       devices = LocalLink.Install(sources.Get(i), gateways.
              Get(i));
230       address.NewNetwork();
231
```

```
232       Ipv4InterfaceContainer interfaces = address.Assign (
              devices);
233       devices = UnReLink.Install(gateways.Get(i), sinks.Get
              (i));
234       address.NewNetwork();
235
236       interfaces = address.Assign (devices);
237       sink_interfaces.Add(interfaces.Get(1));
238     }
239
240   NS_LOG_INFO ("Initialize Global Routing.");
241   Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
242
243   uint16_t port = 50000;
244   Address sinkLocalAddress (InetSocketAddress (Ipv4Address
          ::GetAny (), port));
245   PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory",
          sinkLocalAddress);
246
247   for(uint16_t i=0; i<sources.GetN(); i++)
248     {
249       AddressValue remoteAddress (InetSocketAddress (
              sink_interfaces.GetAddress(i, 0), port));
250
251       if (transport_prot.compare("TcpTahoe") == 0
252           || transport_prot.compare("TcpReno") == 0
253           || transport_prot.compare("TcpNewReno") == 0
254           || transport_prot.compare("TcpWestwood") == 0
255           || transport_prot.compare("TcpWestwoodPlus") ==
                  0)
```

```
256         {
257           Config::SetDefault ("ns3::TcpSocket::SegmentSize"
                 , UintegerValue (tcp_adu_size));
258           BulkSendHelper ftp("ns3::TcpSocketFactory",
                 Address());
259           ftp.SetAttribute ("Remote", remoteAddress);
260           ftp.SetAttribute ("SendSize", UintegerValue (
                 tcp_adu_size));
261           ftp.SetAttribute ("MaxBytes", UintegerValue (int(
                 data_mbytes*1000000)));
262
263           ApplicationContainer sourceApp = ftp.Install (
                 sources.Get(i));
264           sourceApp.Start (Seconds (start_time*i));
265           sourceApp.Stop (Seconds (stop_time - 3));
266
267           sinkHelper.SetAttribute ("Protocol", TypeIdValue
                 (TcpSocketFactory::GetTypeId ()));
268           ApplicationContainer sinkApp = sinkHelper.Install
                  (sinks);
269           sinkApp.Start (Seconds (start_time*i));
270           sinkApp.Stop (Seconds (stop_time));
271         }
272       else
273         {
274           NS_LOG_DEBUG ("Invalid transport protocol " <<
                 transport_prot << " specified");
275           exit (1);
276         }
277     }
```

```
278
279   // Set up tracing if enabled
280   if (tracing)
281     {
282       std::ofstream ascii;
283       Ptr<OutputStreamWrapper> ascii_wrap;
284       if (tr_file_name.compare("") == 0)
285         {
286           NS_LOG_DEBUG ("No trace file provided");
287           exit (1);
288         }
289       else
290         {
291           ascii.open (tr_file_name.c_str());
292           ascii_wrap = new OutputStreamWrapper(tr_file_name
                  .c_str(), std::ios::out);
293         }
294
295       stack.EnableAsciiAll (ascii_wrap);
296
297       Simulator::Schedule(Seconds(0.00001), &TraceCwnd,
              cwnd_tr_file_name);
298       Simulator::Schedule(Seconds(0.00001), &TraceSsThresh,
               ssthresh_tr_file_name);
299     }
300
301   Ptr<FlowMonitor> flowMonitor;
302   FlowMonitorHelper flowHelper;
303   flowMonitor = flowHelper.InstallAll();
304
```

```
305    Simulator::Stop (Seconds(stop_time));
306    Simulator::Run ();
307
308    flowMonitor->SerializeToXmlFile(flow_monitor_file, true,
           true);
309
310    Simulator::Destroy ();
311    return 0;
312  }
```