

# Relatório: Utilização do Simulador de Redes NS-3

João Duarte, Lorhan Lima, Lucas Moreira.

Dezembro de 2024

## Sumário

Sumário	1
1 INTRODUÇÃO	3
2 PREPARAÇÃO DO AMBIENTE DE PROTOTIPAGEM	4
2.1 Hands-on 01: GitHub	4
2.1.1 Objetivos	4
2.1.2 Resultados	4
2.2 Hands-on 02: Jupyter Notebook	5
2.2.1 Objetivos	5
2.2.2 Resultados	5
3 PROTOTIPAGEM COM O NS-3	8
3.1 Atividade 2.2: first.cc	8
3.1.1 Objetivos	8
3.1.2 Resultados	8
4 ATIVIDADE 2.3: SECOND.CC	10
4.0.1 Objetivos	10
4.0.2 Resultados	10
5 ATIVIDADE 2.4: THIRD.CC	12
5.0.1 Objetivos	12
5.0.2 Resultados	12

6	ATIVIDADE 2.5: FIFTH.CC . . . . .	15
6.0.1	Objetivos . . . . .	15
6.0.2	Resultados . . . . .	15
7	ATIVIDADES 2.6 E 2.7: SIXTH.CC E SEVENTH.CC . . . . .	16
7.0.1	Objetivos . . . . .	16
7.0.2	Resultados . . . . .	16
8	ATIVIDADE 2.8: HANDS-ON 7 - RATE-ADAPTATION-DISTANCE.CC . . . . .	18
8.0.1	Objetivos . . . . .	18
8.0.2	Resultados . . . . .	18
8.0.3	Conclusão . . . . .	19
9	CONCLUSÃO . . . . .	20

# 1 Introdução

Este relatório tem como objetivo mostrarmos os resultados obtidos pela nossa equipe, junto com as etapas de utilização e dificuldades enfrentadas no uso do software de simulação de redes NS-3. Também foram utilizadas as ferramentas GitHub e Jupyter Notebook na realização do trabalho. Obtivemos sucesso em todas as etapas necessárias para a realização das simulações, tendo problemas apenas na integração com o VSCode. Contudo, isso não afetou o resultado final.

## 2 Preparação do Ambiente de Prototipagem

### 2.1 Hands-on 01: GitHub

#### 2.1.1 Objetivos

Nessa etapa, realizamos a criação e clonagem do repositório, submissão de arquivos e mudanças nos arquivos existentes do repositório, juntamente com a edição do `Readme.md`. Não foi necessário realizar o cadastro, pois todos os membros da equipe já possuíam conta na plataforma.

#### 2.1.2 Resultados

Nesta etapa aprendemos a fazer o clone de um repositório via terminal utilizando o comando:

```
git clone "endereço do repositório"
```

Após isso, criamos uma pasta no mesmo diretório em que estávamos rodando o terminal, com o nome do repositório em seu título.

Para criar um arquivo de teste, utilizamos o comando:

```
touch "nomedoarquivo.formato"
```

Depois, agendamos o envio para o repositório com o comando:

```
git add teste.txt
```

Para validar a alteração, utilizamos o comando:

```
git commit -m "att"
```

O login e senha do GitHub foram solicitados. Porém, enfrentamos um problema devido a uma atualização da plataforma, que não aceita mais login por senha. Foi necessário criar um token nas opções de segurança do GitHub. Após criar o token, conseguimos fazer o commit novamente. Para finalizar, enviamos o arquivo com o comando:

```
git push origin main
```

Também ajustamos o comando de `master` para `main`, como instruído no material, e concluímos a etapa. O arquivo de teste foi então confirmado no repositório.

Para editar o `Readme`, utilizamos a interface da plataforma online, simplesmente clicando no botão "editar" e inserindo o conteúdo desejado.

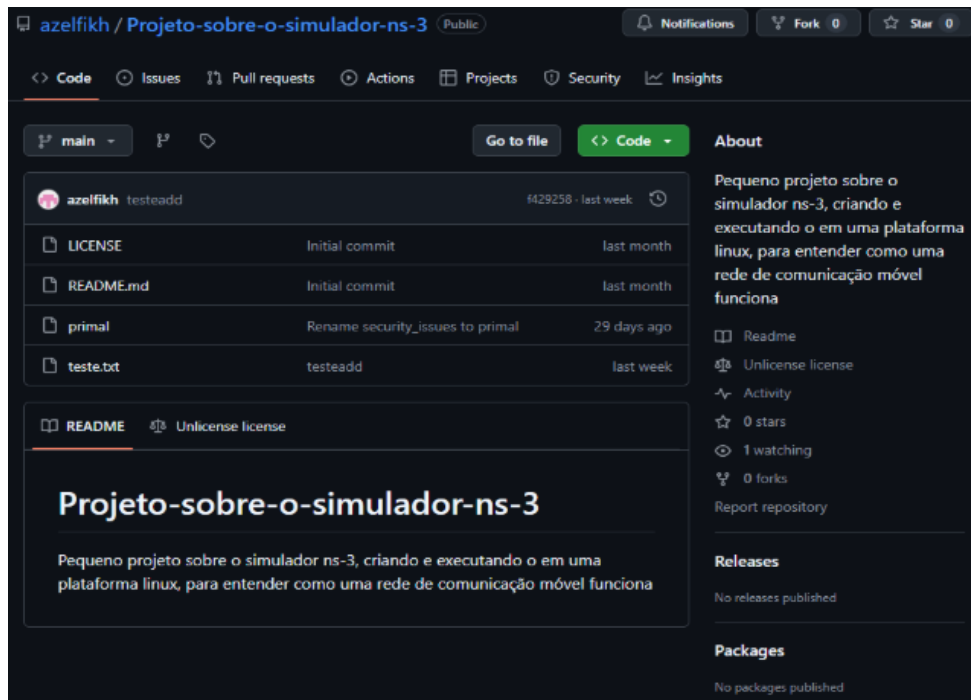


Figura 1 – Repositório no GitHub

## 2.2 Hands-on 02: Jupyter Notebook

### 2.2.1 Objetivos

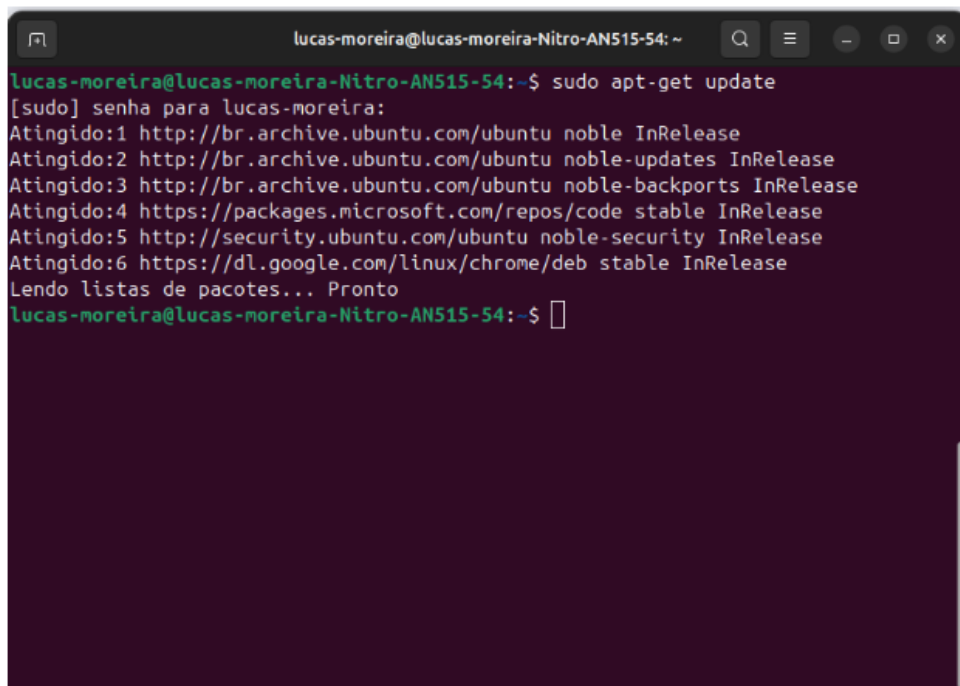
Essa etapa teve como finalidade baixar o repositório do curso, criar um notebook e convertê-lo para PDF.

### 2.2.2 Resultados

O primeiro passo foi realizar a preparação da máquina para receber a instalação do Jupyter Notebook. Para isso, atualizamos o gerenciador de pacotes utilizando o comando:

```
sudo apt-get update
```

Como mostrado na Figura 2, o gerenciador foi atualizado com sucesso.

A terminal window with a dark background and light green text. The window title is 'lucas-moreira@lucas-moreira-Nitro-ANS15-54: ~'. The command 'sudo apt-get update' has been executed. The output shows the system checking for updates from various sources, including Ubuntu archives and Microsoft repositories. The process is complete, and the prompt is ready for the next command.


```
lucas-moreira@lucas-moreira-Nitro-ANS15-54: ~  
lucas-moreira@lucas-moreira-Nitro-ANS15-54:~$ sudo apt-get update  
[sudo] senha para lucas-moreira:  
Atingido:1 http://br.archive.ubuntu.com/ubuntu noble InRelease  
Atingido:2 http://br.archive.ubuntu.com/ubuntu noble-updates InRelease  
Atingido:3 http://br.archive.ubuntu.com/ubuntu noble-backports InRelease  
Atingido:4 https://packages.microsoft.com/repos/code stable InRelease  
Atingido:5 http://security.ubuntu.com/ubuntu noble-security InRelease  
Atingido:6 https://dl.google.com/linux/chrome/deb stable InRelease  
Lendo listas de pacotes... Pronto  
lucas-moreira@lucas-moreira-Nitro-ANS15-54:~$
```

Figura 2 – Prompt de atualização do gerenciador de pacotes apt-get

Na sequência, executamos o comando:

```
sudo apt-get install jupyter
```

Isso instalou o Jupyter Notebook, conforme demonstrado na Figura 3.

A terminal window with a dark background and light green text. The window title is 'lucas-moreira@lucas-moreira-Nitro-ANS15-54: ~'. The command 'sudo apt-get install jupyter' has been executed. The output shows the system downloading the package, checking dependencies, and installing it. The process is complete, and the prompt is ready for the next command.

```
lucas-moreira@lucas-moreira-Nitro-ANS15-54:~$ sudo apt-get install jupyter  
Lendo listas de pacotes... Pronto  
Construindo árvore de dependências... Pronto  
Lendo informação de estado... Pronto  
Pacotes sugeridos:  
  jupyter-qtconsole  
Os NOVOS pacotes a seguir serão instalados:  
  jupyter  
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 127 não  
atualizados.  
É preciso baixar 0 B/2.174 B de arquivos.  
Depois desta operação, 10,2 kB adicionais de espaço em disco serão usados.  
A seleccionar pacote anteriormente não seleccionado jupyter.  
(Lendo banco de dados ... 227504 ficheiros e diretórios atualmente instalados.)  
A preparar para desempacotar .../jupyter_5.3.2-1ubuntu1_all.deb ...  
A descompactar jupyter (5.3.2-1ubuntu1) ...  
Configurando jupyter (5.3.2-1ubuntu1) ...  
lucas-moreira@lucas-moreira-Nitro-ANS15-54:~$
```

Figura 3 – Instalação do Jupyter Notebook

Depois, clonamos o repositório disponibilizado no roteiro da atividade com o comando:

```
git clone "endereço do repositório"
```

Esse comando nos forneceu todos os arquivos necessários para o desenvolvimento do trabalho. Por fim, inicializamos o Jupyter Notebook no terminal utilizando:

```
jupyter notebook
```

Após alguns instantes, uma aba no navegador foi aberta contendo os arquivos do diretório onde o Jupyter foi inicializado, como mostra a Figura 4.

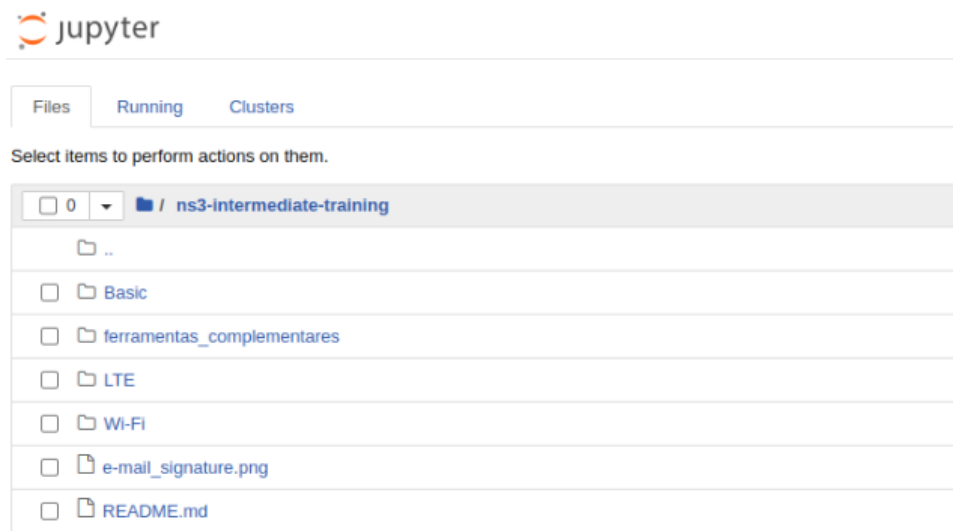


Figura 4 – Interface do Jupyter Notebook

Essa etapa confirmou que o Jupyter oferece uma abordagem interativa e portátil para projetos científicos e de programação, permitindo análises dinâmicas e participativas dos resultados.

## 3 Prototipagem com o NS-3

### 3.1 Atividade 2.2: first.cc

#### 3.1.1 Objetivos

Essa atividade teve o objetivo de realizar, via barramento, uma comunicação ponto a ponto (P2P). Enviamos dois pacotes com tamanhos diferentes do cliente para o servidor, que estava configurado para responder reenviando os pacotes recebidos pela porta 9.

#### 3.1.2 Resultados

Inicialmente, realizamos a leitura do código no guia do *hands-on* para compreender sua estrutura. Identificamos os scripts inclusos, os logs exibidos, e as linhas responsáveis por criar, instalar e conectar os nós por meio do objeto `NetDeviceContainer` com o método `install`. Para a comunicação, criamos uma pilha utilizando `InternetStackHelper` e atribuímos IPs válidos aos nós.

Compreendemos também as configurações do servidor, que recebia pacotes pela porta especificada, e do cliente, que enviava pacotes com diferentes tamanhos. Após entender o código, executamos o `first.cc`, que resultou na saída esperada: o envio de dois pacotes simultaneamente, com um deles tendo o dobro do tamanho do outro. O servidor recebeu e reenviou os pacotes ao cliente pela porta 9.

**Primeiro Desafio:** Enviar dois pacotes de tamanhos diferentes. Alteramos os parâmetros no código, criando dois clientes configurados com diferentes tamanhos de pacotes, conforme o guia. Utilizamos o comando:

```
./ns3 run "examples/tutorial/first.cc"
```

O resultado mostrou que os pacotes foram enviados corretamente, e o servidor os recebeu e reenviou ao cliente.

**Segundo Desafio:** Alterar a taxa de transmissão. Modificamos a linha de código responsável pelo parâmetro `DataRate`, reduzindo a taxa de 5Mbps para 1Gbps. Executamos o programa novamente, resultando em um tempo de execução significativamente menor.



```
[0/2] Re-checking globbed directories...  
[1/1] Linking CXX executable ../build/examples/tutorial/ns3.43-first-default  
At time +2s client sent 1024 bytes to 10.1.1.2 port 9  
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153  
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153  
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

Figura 5 – Saída first.cc

```
At time +2s client sent 1024 bytes to 10.1.1.2 port 9  
At time +2s client sent 2048 bytes to 10.1.1.2 port 9  
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153  
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153  
At time +2.00705s server received 2048 bytes from 10.1.1.1 port 49154  
At time +2.00705s server sent 2048 bytes to 10.1.1.1 port 49154  
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9  
At time +2.01241s client received 2048 bytes from 10.1.1.2 port 9
```

Figura 6 – Saída desafio 01 first.cc

```
At time +2s client sent 1024 bytes to 10.1.1.2 port 9  
At time +2s client sent 2048 bytes to 10.1.1.2 port 9  
At time +2.00201s server received 1024 bytes from 10.1.1.1 port 49153  
At time +2.00201s server sent 1024 bytes to 10.1.1.1 port 49153  
At time +2.00203s server received 2048 bytes from 10.1.1.1 port 49154  
At time +2.00203s server sent 2048 bytes to 10.1.1.1 port 49154  
At time +2.00402s client received 1024 bytes from 10.1.1.2 port 9  
At time +2.00404s client received 2048 bytes from 10.1.1.2 port 9
```

Figura 7 – Saída desafio 02 first.cc

## 4 Atividade 2.3: second.cc

### 4.0.1 Objetivos

Essa atividade teve como objetivo compreender o funcionamento do exemplo `second.cc` e realizar três desafios:

1. Enviar pacotes de diferentes tamanhos sem criar novos objetos.
2. Aumentar o número de nós CSMA pelo terminal.
3. Utilizar a funcionalidade `tcpdump` para analisar os arquivos `pcap`.

### 4.0.2 Resultados

Após analisar o código, identificamos as bibliotecas e mensagens de registro incluídas. Com a variável `verbose`, determinamos os componentes habilitados, e com `nCsma`, definimos o número de nós na rede CSMA.

O exemplo cria dois nós conectados por um barramento P2P, onde um deles se comunica com uma rede CSMA composta por vários nós. Utilizamos os helpers para configurar os canais e os endereços IP. A coleta de informações foi habilitada com a funcionalidade `pcap`.

**Primeiro Desafio:** Configurar dois pacotes para que o segundo tenha o dobro do tamanho do primeiro. Criamos dois objetos `UdpEchoClientHelper` para configurar os tamanhos e os tempos de envio. Utilizamos o comando:

```
./ns3 run "examples/tutorial/second.cc"
```

Os resultados mostraram que os pacotes foram enviados e recebidos corretamente pelas portas e tempos estipulados.

**Segundo Desafio:** Alterar o número de nós CSMA pelo terminal. Modificamos o código para incluir uma variável `nCsma`, configurável pelo terminal. Executamos o programa com:

```
./ns3 run "examples/tutorial/second.cc -nCsma=5"
```

Isso gerou uma saída semelhante à anterior, mas com o número ajustado de nós.

**Terceiro Desafio:** Analisar os arquivos `pcap`. Utilizamos o comando `tcpdump` para analisar as conexões entre os nós e os protocolos usados. A análise confirmou que as informações coletadas estavam de acordo com o esperado.



## 5 Atividade 2.4: `third.cc`

### 5.0.1 Objetivos

O objetivo dessa atividade foi explorar o código `third.cc`, realizando desafios que incluíram:

1. Envio de três pacotes de diferentes *STAs*.
2. Uso do `NetAnim` para observar a movimentação dos usuários.
3. Configuração do exemplo `rate-adaptation-distance.cc` para uma rede Wi-Fi simulada.

### 5.0.2 Resultados

Após analisar o código, identificamos uma nova biblioteca adicionada: `ns3/wifi-module.h`. Variáveis auxiliares foram configuradas para permitir ajustes via linha de comando, sem necessidade de alterar o código original.

Três nós Wi-Fi foram criados, e o primeiro nó da rede P2P foi designado como ponto de acesso. A comunicação foi realizada utilizando pacotes UDP e rastreamentos habilitados para análise posterior.

**Primeiro Desafio:** Enviar três pacotes de diferentes *STAs*. Cada pacote foi configurado com tamanhos diferentes e enviado em intervalos distintos. Ajustamos os tempos de início e fim das aplicações no código, garantindo que as mensagens fossem transmitidas corretamente.

**Segundo Desafio:** Utilizar o `NetAnim`. O `NetAnim` deveria ser usado para visualizar a movimentação dos usuários. Contudo, não conseguimos gerar o arquivo `.xml` necessário, o que impediu a realização completa do desafio. Tentamos contornar o problema usando a função `-vis` para ter alguma visualização do que está sendo simulado.

**Terceiro Desafio:** Configuração do exemplo para plotagem do gráfico. Simulamos uma rede Wi-Fi com movimentação dos nós e também modificamos o código configurando o movimento dos usuários e definindo o salvamento das informações para controlar e observamos que, à medida que a distância entre os nós variava, o throughput também mudava. A movimentação foi registrada e os resultados plotados com o `GNUPLOT`.

```
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.43-third-default
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.00926s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.00926s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.02449s client received 1024 bytes from 10.1.2.4 port 9
```

Figura 12 – Saída third.cc

```
gerson@netsimulator:~/ns3/ns-allinnone-3.43/ns-3.43$ ./ns3 run "examples/tutorial/third.cc"
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/examples/tutorial/ns3.43-third-default
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.00921s server received 1024 bytes from 10.1.3.1 port 49153
At time +2.00921s server sent 1024 bytes to 10.1.3.1 port 49153
At time +2.01085s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01085s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.01254s server received 1024 bytes from 10.1.3.2 port 49153
At time +2.01254s server sent 1024 bytes to 10.1.3.2 port 49153
At time +2.02449s client received 1024 bytes from 10.1.2.4 port 9
At time +2.0331s client received 1024 bytes from 10.1.2.4 port 9
At time +2.03477s client received 1024 bytes from 10.1.2.4 port 9
```

Figura 13 – Saída desafio 01 third.cc

```
gerson@netsimulator:~/ns3/ns-allinnone-3.43/ns-3.43$ ./ns3 run "examples/tutorial/third.cc"
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/examples/tutorial/ns3.43-third-default
At time +10s client sent 2048 bytes to 10.1.2.4 port 9
At time +10s client sent 1024 bytes to 10.1.2.4 port 9
At time +10.0093s server received 1024 bytes from 10.1.3.3 port 49153
At time +10.0093s server sent 1024 bytes to 10.1.3.3 port 49153
At time +10.013s server received 2048 bytes from 10.1.3.2 port 49153
At time +10.013s server sent 2048 bytes to 10.1.3.2 port 49153
At time +10.0244s client received 1024 bytes from 10.1.2.4 port 9
At time +10.034s client received 2048 bytes from 10.1.2.4 port 9
At time +15s client sent 4096 bytes to 10.1.2.4 port 9
At time +15.0132s server received 4096 bytes from 10.1.3.1 port 49153
At time +15.0132s server sent 4096 bytes to 10.1.3.1 port 49153
At time +15.0273s client received 4096 bytes from 10.1.2.4 port 9
```

Figura 14 – Saída desafio 02 third.cc

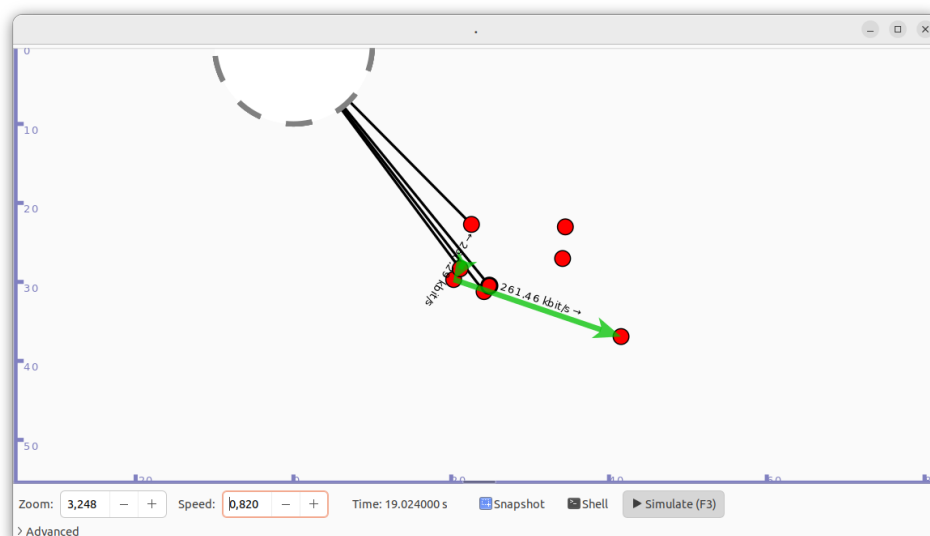


Figura 15 – Saída desafio 02 -vis third.cc

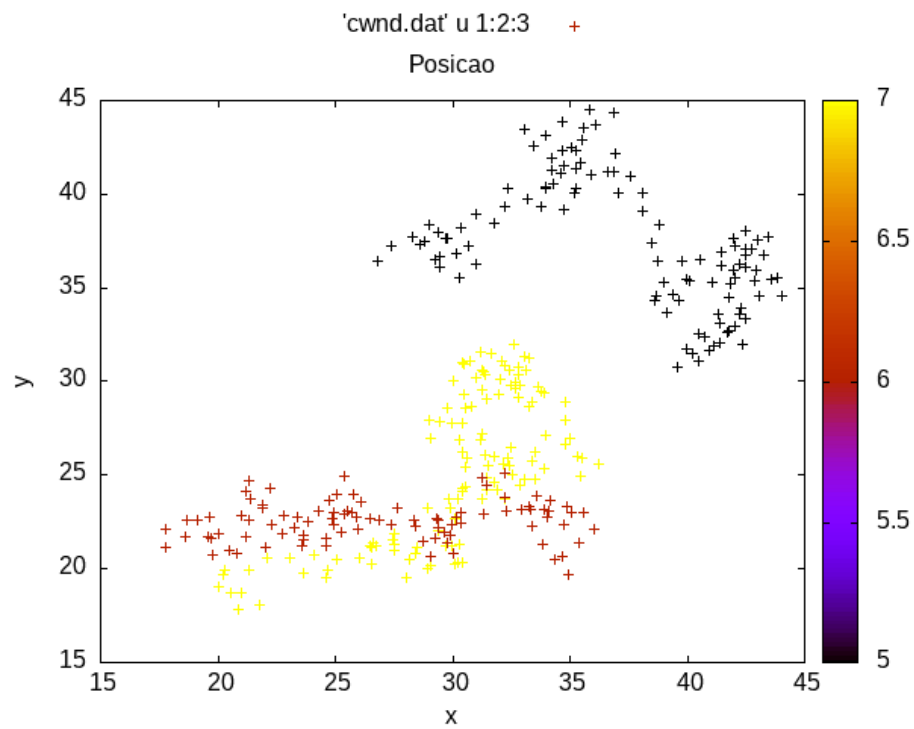


Figura 16 – Saída grafico posição third.cc

## 6 Atividade 2.5: fifth.cc

### 6.0.1 Objetivos

Nesta atividade, utilizamos o exemplo `fifth.cc` para visualizar a janela de congestionamento (`cwnd`) ao longo do tempo e interpretar os resultados.

### 6.0.2 Resultados

Executamos o código original, que exibía mensagens no terminal sobre o tempo e tamanho da janela de congestionamento. Para realizar a plotagem correta do gráfico, comentamos as linhas de código responsáveis por essas mensagens e salvamos os dados em um arquivo `.dat`.

A visualização foi gerada utilizando o comando:

```
gnuplot grafico5.p
```

O resultado foi um gráfico que mostrou como a janela de congestionamento variava ao longo do tempo, confirmando os comportamentos esperados.

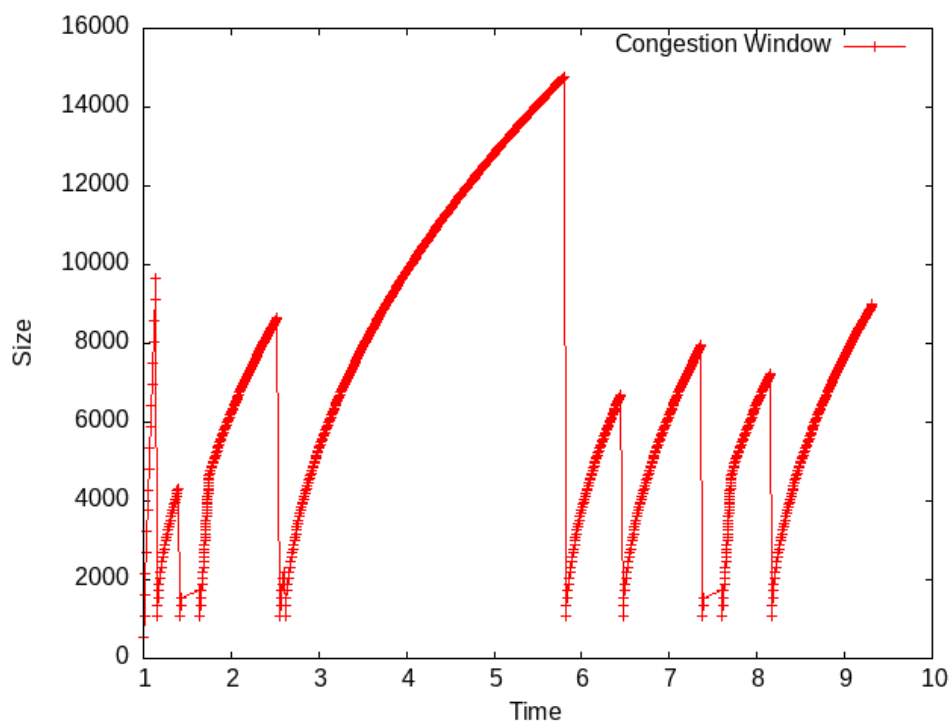


Figura 17 – Gráfico da janela de congestionamento

## 7 Atividades 2.6 e 2.7: sixth.cc e seventh.cc

### 7.0.1 Objetivos

Essas atividades introduziram o uso de **helpers** para automação da geração de arquivos `.pcap` e gráficos no `GNUPLLOT`. Além disso, passamos a trabalhar com o protocolo IPv6.

### 7.0.2 Resultados

No exemplo `sixth.cc`, alteramos os parâmetros das funções `CwndChange` e `RxDrop`, adicionamos os **helpers** `AsciiTraceHelper` e `PcapHelper`, e geramos arquivos de rastreamento em ASCII e `pcap`. Esses arquivos foram analisados utilizando ferramentas como `tcpdump`.

Rodamos o código com sucesso e geramos dois arquivos: um com o rastreamento de congestionamento (`.cwnd`) e outro com o tráfego de pacotes (`.pcap`). Ambos os arquivos foram validados com o comando:

```
tcpdump -nn -tt -r sexto.pcap
```

O exemplo `seventh.cc` envolveu a configuração do protocolo IPv6. Alteramos a variável correspondente de `false` para `true`, conforme o guia, e o programa passou a usar IPv6 nas simulações.

Além disso, automatizamos a geração de gráficos com o `GnuplotHelper`. O `FileHelper` foi utilizado para exportar os dados, que foram posteriormente usados para gerar um gráfico no `GNUPLLOT`. O gráfico final mostrou o desempenho da rede sob o novo protocolo. O gráfico resultante é um pouco diferente do mostrado no guia, porém os valores são parecidos, o que nos leva a crer ter sido causado apenas um erro de configuração de escala..



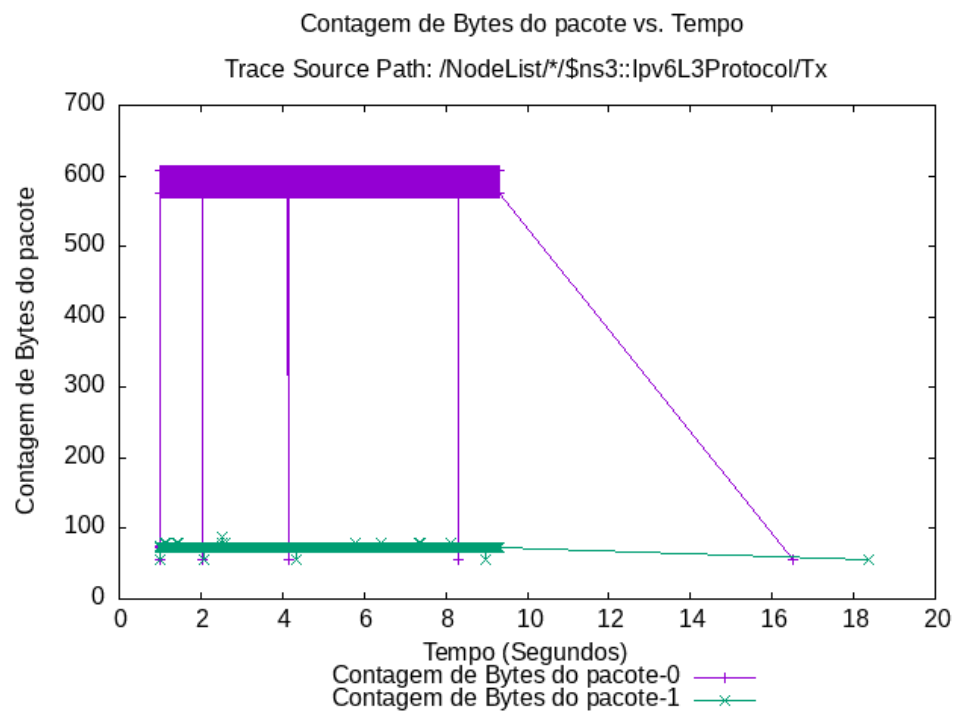


Figura 18 – Gráfico gerado pelo GNUPLOT

## 8 Atividade 2.8: Hands-on 7 - rate-adaptation-distance.cc

### 8.0.1 Objetivos

Nesta atividade, configuramos algoritmos de adaptação de taxa para redes Wi-Fi, avaliando o impacto da distância e da movimentação dos usuários na rede. Utilizamos o exemplo `rate-adaptation-distance.cc` para simular a movimentação de um *STA* (Station) em relação ao ponto de acesso (AP) e ajustamos a taxa de transmissão dinamicamente.

### 8.0.2 Resultados

A simulação foi realizada com dois nós: um ponto de acesso (AP) e um *STA* em movimento. A cada 2 segundos, o *STA* se movia 2 metros. O algoritmo `Minstrel` foi configurado para ajustar a taxa de transmissão com base nas perdas de pacotes e no desempenho da rede.

No primeiro desafio, configuramos o *STA* para mover-se aleatoriamente em relação ao ponto de acesso. A movimentação foi registrada com a função `AdvancePosition`, que foi associada a um modelo de movimentação utilizando o alocador `RandomDiscPositionAllocator`.

Ao longo da simulação, a taxa de transmissão foi ajustada automaticamente pelo algoritmo `Minstrel`, com base na qualidade do link entre o *STA* e o AP. A taxa foi aumentada ou diminuída conforme a necessidade, buscando garantir uma transmissão eficiente e minimizando perdas.

Os dados foram coletados e exportados para um arquivo `.dat`, que foi utilizado para gerar gráficos de desempenho. Utilizamos o `GNU PLOT` para gerar o gráfico de throughput, e os resultados confirmaram que, à medida que a distância aumentava, a taxa de transmissão diminuía, com o algoritmo `Minstrel` ajustando a taxa de forma eficiente.

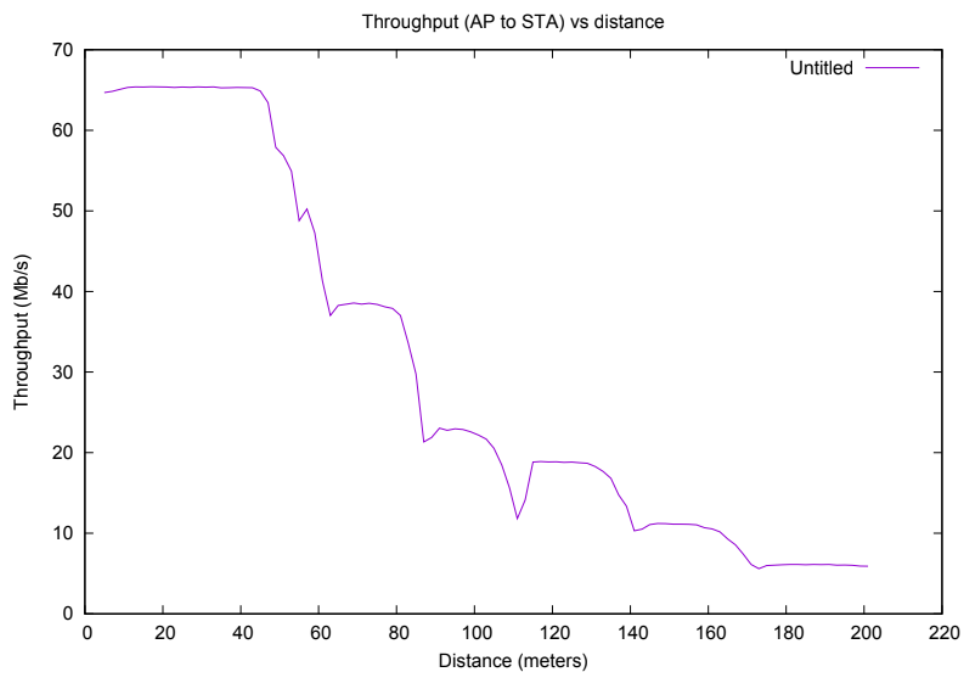


Figura 19 – Gráfico de throughput da adaptação de taxa

### 8.0.3 Conclusão

A atividade mostrou como a adaptação de taxa pode melhorar a eficiência das redes Wi-Fi, ajustando dinamicamente a taxa de transmissão conforme a distância e as condições do canal. A utilização do `GNUPLOT` para visualização dos resultados proporcionou uma compreensão clara dos impactos dessas mudanças na rede.

## 9 Conclusão

Durante as atividades propostas neste relatório, foi possível obter uma compreensão sólida do funcionamento do simulador NS-3 e das ferramentas associadas, como GitHub e Jupyter Notebook. A realização dos hands-ons permitiu que experimentássemos a configuração de redes complexas, análise de desempenho e adaptação dinâmica de taxa em redes sem fio.

Apesar de alguns desafios enfrentados, como problemas de integração do NS-3 com o VSCode e dificuldades na visualização com **NetAnim**, conseguimos alcançar os objetivos propostos nas atividades, ajustando os parâmetros e realizando as simulações com sucesso.

O uso das ferramentas de rastreamento (**pcap**, **cwnd**) e a geração de gráficos com o **GNUPLLOT** possibilitaram uma análise detalhada do desempenho da rede, permitindo observar os efeitos das mudanças de parâmetros na transmissão de pacotes e na taxa de congestionamento.

Em resumo, o trabalho foi bem-sucedido, com a equipe superando as dificuldades técnicas e cumprindo os objetivos de aprendizado, o que resultou em uma compreensão mais profunda sobre a simulação de redes com o NS-3.