



**Escola de Engenharia**  
**Curso de Bacharelado em Engenharia de Sistemas**

**DCC218 - Introdução a Sistemas Computacionais**  
**Relatório do TP 1 - Parte 2**

Raphael Henrique Braga Leivas - 2020028101

Belo Horizonte  
4 de novembro de 2025

## SUMÁRIO

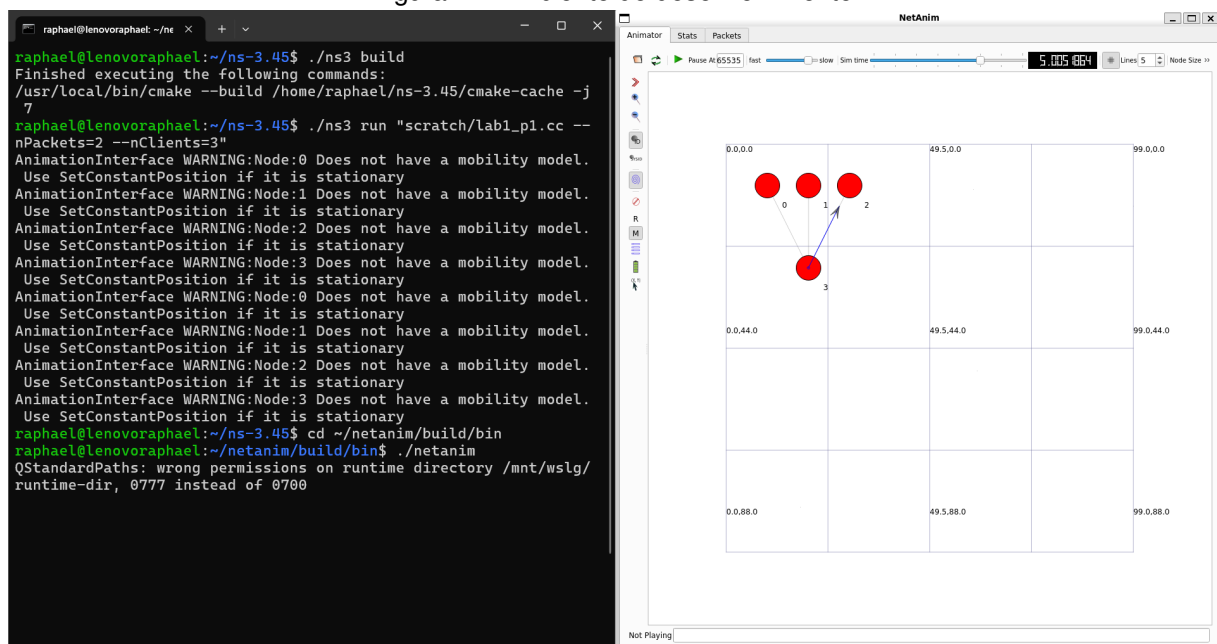
<b>1</b>	<b>AMBIENTE DE DESENVOLVIMENTO . . . . .</b>	<b>3</b>
<b>2</b>	<b>PARTE 1: TOPOLOGIA COM GARGALO . . . . .</b>	<b>4</b>
2.1	Janela de Congestionamento . . . . .	4
2.2	Parte 1a: Comparação entre o TCP CUBIC e o NewReno . . . . .	4
<b>3</b>	<b>PARTE 2: TOPOLOGIA COM GARGALO + FLUXOS HETEROGÊNEOS . .</b>	<b>6</b>

## 1 AMBIENTE DE DESENVOLVIMENTO

O projeto é desenvolvido em um notebook Lenovo com processador Intel-Core i5 de 11<sup>o</sup> geração, com arquitetura x64. A frequência do processador é 2.4 GHz e possui 4 cores. O processador possui 8 GB de memória RAM disponível. O sistema operacional do computador é Windows 11, mas o ambiente NS-3 é instalado através de WSL (Windows Subsystem for Linux) Ubuntu 22.04.1 LTS.

O ambiente NS-3 é instalado na pasta `home` do WSL. A Figura 1 ilustra o ambiente de desenvolvimento utilizado no projeto.

Figura 1 – Ambiente de desenvolvimento.



Fonte: Elaboração própria.

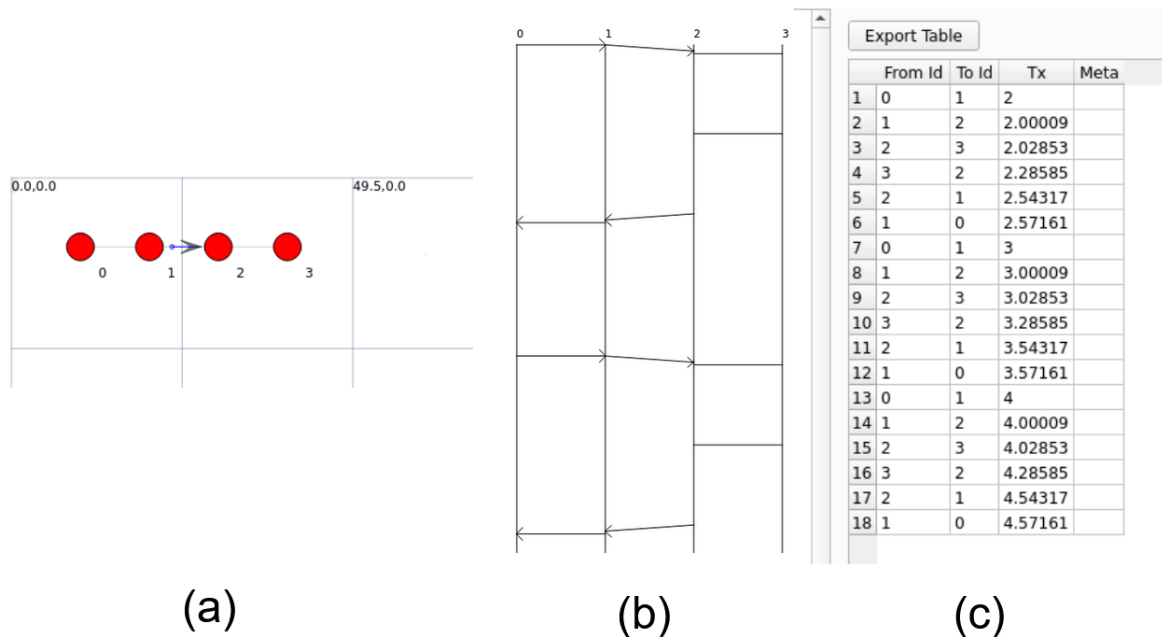
Essencialmente, é adotado um procedimento de duas etapas:

1. O comando `./ns3 build`, seguido de `./ns3 run` com o programa e argumentos de linha de comando desejados, é usado para executar o script C++ com as funções do NS-3. A saída dos comandos será um arquivo XML;
2. O comando `./netanim` é usado para visualizar o arquivo XML exportado na etapa 1, obtendo a animação exibida no lado direito da Figura 1.

## 2 PARTE 1: TOPOLOGIA COM GARGALO

O primeiro passo é obter uma topologia em que há um link de gargalo entre os nós que estão se comunicando. A Figura 2 (a) mostra a topologia e os tempos de envio e recepção de um pacote (b) e (c).

Figura 2 – Topologia base e resultados iniciais da parte 1.



Fonte: Elaboração própria.

Na topologia, os links entre os nós 0 - 1 e 2 - 3 possuem largura de banda de 100 Mbps. O link entre 1 - 2 representa o gargalo, com apenas 1Mbps de banda. Isso se reflete nas Figuras 2 (b) e (c), nas quais os pacotes gastam poucos microsegundos para realizar a transmissão entre os nós 0 - 1 e 2 - 3, e mais de 200 milissegundos entre os nós de gargalo da rede.

### 2.1 Janela de Congestionamento

A ação da janela de congestionamento, quando aplicamos um Bulksend com 5 fluxos simultaneamente pode ser vista na Figura ???. Vemos que quando o gargalo tem banda de 1 Mbps, o Goodput (taxa de dados que efetivamente são transmitidos entre o recipiente e o destinatário na rede) fica na ordem de 0.1 a 0.2 Mbps. Quando aumentamos a banda do gargalo para 10 Mbps, eles se estabiliza em 1 a 2 Mbps, cerca de 10% do total do gargalo. Essas variações nos valores do Goodput refletem as variações na janela de congestionamento do TCP.

### 2.2 Parte 1a: Comparação entre o TCP CUBIC e o NewReno

Figura 3 – Ação da janela de congestionamento na topologia com o Bulksend.

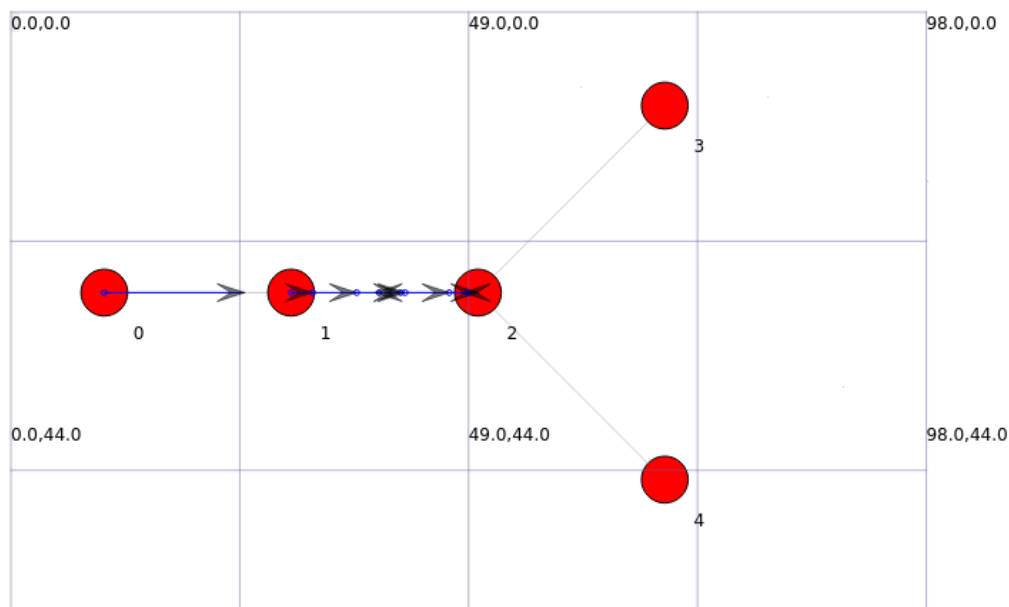
```
raphael@lenovoraphael:~/ns-3.45$ ./ns3 run "scratch/lab2_pl.cc --dataRate="1Mbps" --delay="20ms" --errorRate=0.00001 --nFlows=5"
=== Goodput (Mbps) ===
Flow 0: 0.122637 Mbps
Flow 1: 0.108915 Mbps
Flow 2: 0.195962 Mbps
Flow 3: 0.188243 Mbps
Flow 4: 0.16423 Mbps
raphael@lenovoraphael:~/ns-3.45$ ./ns3 run "scratch/lab2_pl.cc --dataRate="10Mbps" --delay="20ms" --errorRate=0.00001 --nFlows=5"
=== Goodput (Mbps) ===
Flow 0: 1.10931 Mbps
Flow 1: 1.62419 Mbps
Flow 2: 1.00897 Mbps
Flow 3: 1.5141 Mbps
Flow 4: 1.13761 Mbps
raphael@lenovoraphael:~/ns-3.45$
```

Fonte: Elaboração própria.

### 3 PARTE 2: TOPOLOGIA COM GARGALO + FLUXOS HETEROGÊNEOS

A topologia usada nesse exercício está exibida na Figura 4. O nó 0 (remetente) deseja enviar vários pacotes para os nós 3 e 4 (destinatários). Entre os nós 1 e 2, há um gargalo com banda 1 Mbps e 20 ms de delay. A banda das demais conexões é 100 ms com delay 0.01 ms, exceto o ramo entre os nós 2 e 3 que possui delay consideravelmente maior de 50 ms.

Figura 4 – Topologia da parte 2.



Fonte: Elaboração própria.

Note na Figura 4 que os pacotes vão se acumulando no gargalo. O objetivo é verificar no experimento como isso afeta a transmissão de dados de entre o remetente e os destinatários, inclusive para diferentes algoritmos de controle de congestionamento. Para isso, será adotado o seguinte procedimento:

1. Calcula o goodput (taxa de dados efetivamente enviada) dos nós 3 e 4;
2. Repete 10 vezes e toma a média;
3. Repete para os algoritmos TCP New Reno e TCP CUBIC;
4. Faz isso para o número de fluxos de 2, 4, 6 e 8.
5. Exporta tudo em csv e plota gráficos com os dados coletados em python com a ferramenta `matplotlib`.

Calculando o goodput na topologia da Figura 4 com 4 fluxos, obtemos os resultados da Figura 5. Vemos que o servidor B (nó 4) recebe menos dados devido ao

maior delay no seu ramo; contudo, devido ao congestionamento no ramo central, ele acaba recebendo uma quantidade de dados não muito menor que o nó 3, que possui um delay mais de 100 vezes menor.

Além disso, note que a banda de ambos os ramos é 100 Mbps. No entanto, com o gargalo de 1 Mbps entre o remete e o destinatário, a banda efetivamente usada é menos que 1% disso.

Figura 5 – Goodput calculados na topologia da parte 2.

```
raphael@lenovoraphael:~/ns-3.45$ ./ns3 run "scratch/lab2_p2.cc --dataRate="1Mbps" --delay="20ms" --errorRate=0.00001
--transport_prot="ns3::TcpNewReno" --nFlows=4"
[ 0%] Building CXX object scratch/CMakeFiles/scratch_lab2_p2.dir/lab2_p2.cc.o
[ 0%] Linking CXX executable /home/raphael/ns-3.45/build/scratch/ns3.45-lab2_p2-debug
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
----- Calcula o Goodput -----
Fluxo 0 recebeu: 439520 bytes, Goodput = 0.351616 Mbps (Servidor A (3))
Fluxo 1 recebeu: 241200 bytes, Goodput = 0.19296 Mbps (Servidor A (3))
Fluxo 2 recebeu: 173128 bytes, Goodput = 0.138502 Mbps (Servidor B (4))
Fluxo 3 recebeu: 153296 bytes, Goodput = 0.122637 Mbps (Servidor B (4))
raphael@lenovoraphael:~/ns-3.45$
```

Fonte: Elaboração própria.

Repetindo esse experimento 10 vezes, obtemos a distribuição de Goodput calculados para cada destinatário exibido na Figura 6. Calculando a média de cada um, obtemos

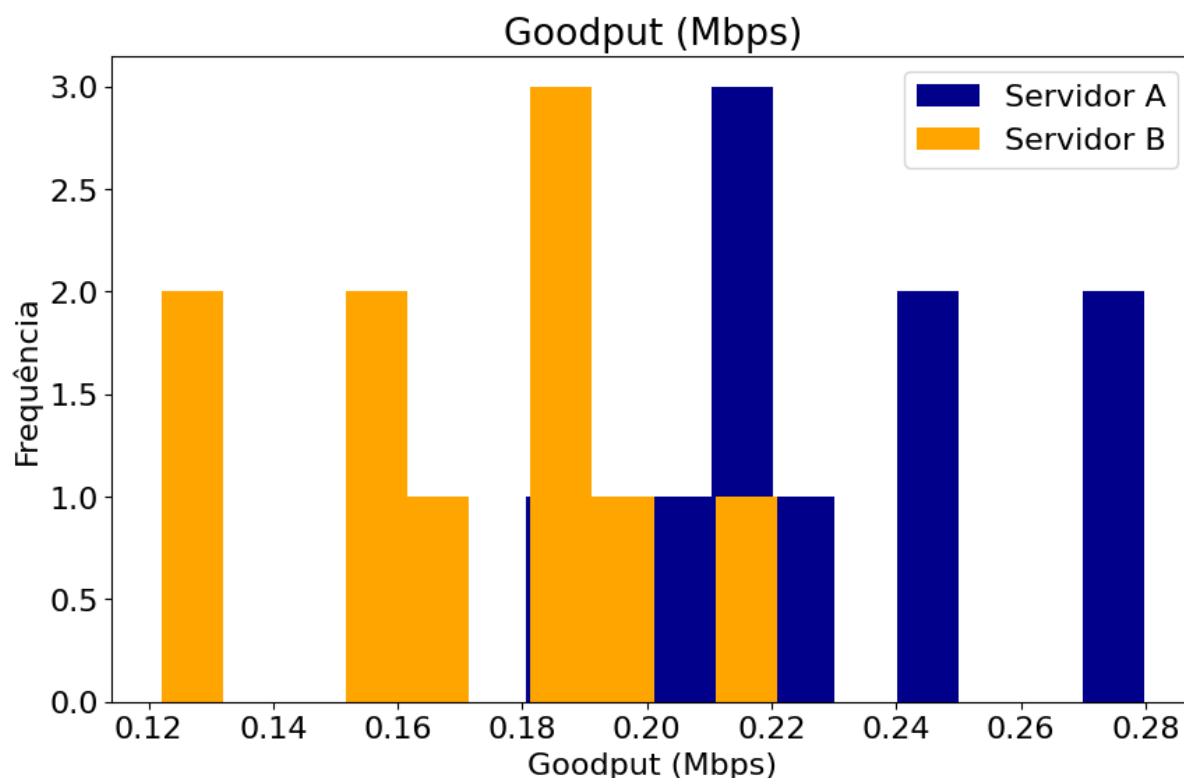
$$\mu_A = 0.23 \text{ Mbps} \quad , \quad \mu_B = 0.17 \text{ Mbps}$$

O próximo passo é comparar o goodput médio obtido entre cada servidor para diferentes algoritmos de congestionamento, para diferentes números de fluxos. O resultado obtido pode ser visto na Figura 7.

Analisando as curvas, podemos chegar às seguintes conclusões:

- Ambos algoritmos de congestionamento apresentam Goodput similar entre os mesmos servidores;
- O servidor B, em ambos protocolos, apresenta menor Goodput devido ao fato de possuir maior delay que o servidor A;
- Quanto maior o número de fluxos menor o Goodput em todos os casos. Isso ocorre devido ao aumento do congestionamento no ramo de gargalo, reduzindo a quantidade de pacotes que conseguem chegar aos destinatários.

Figura 6 – Histograma do goodput obtido por servidor executando 10 vezes.



Fonte: Elaboração própria.

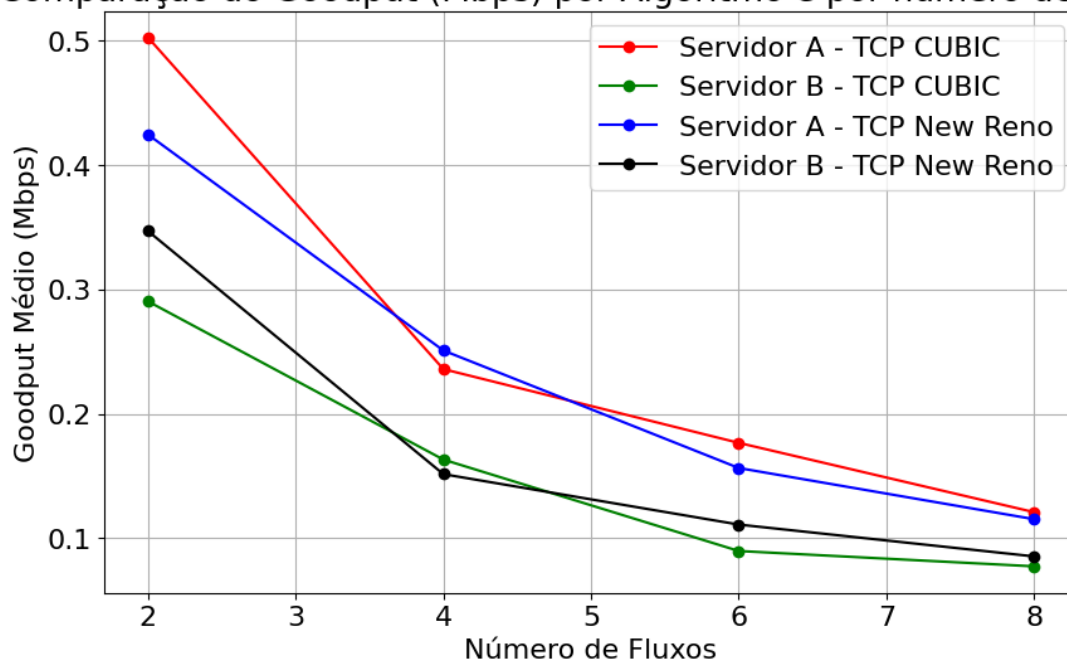
Um outra conclusão que podemos tomar a partir do gráfico se refere à como ambos algoritmos tentam ser "justos"(fairness) com ambos servidores, tentando fazer com que ambos tenham o mesmo RTT mesmo que o delay dos ramos sejam consideravelmente diferente. Para isso, podemos tomar a diferença entre os Goodputs dos servidores para cada algoritmo e plotar o resultado, exibido na Figura 8.

Em média, o New Reno apresenta diferença de 0.06 Mbps, enquanto o CUBIC apresenta diferença média de 0.1 Mbps. Assim, concluímos que o New Reno é mais justo que o TCP CUBIC, por apresentar menor diferença entre os Goodputs de cada servidor.



Figura 7 – Diferença de Goodput (Mbps) entre servidores para os diferentes algoritmos.

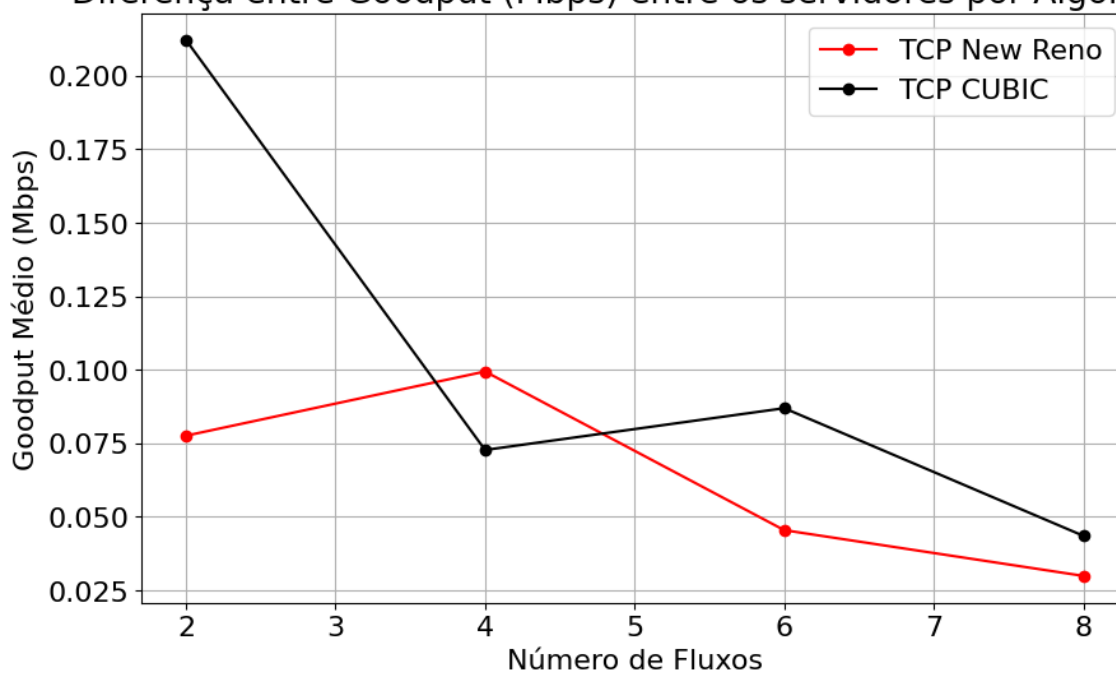
### Comparação do Goodput (Mbps) por Algoritmo e por número de fluxos



Fonte: Elaboração própria.

Figura 8 – Histograma do goodput obtido por servidor executando 10 vezes.

### Diferença entre Goodput (Mbps) entre os servidores por Algoritmo



Fonte: Elaboração própria.