

# MODELAGEM DO ROTEADOR

**Autor: Nélio Pagani Neto**

---

Professor: Jayor Nesi Teixeira

Disciplina: DEC7523 Modelagem e Simulação

Data: 04/12/2025



# SUMÁRIO

## 1 Introdução

- 1.1 Apresentação do objetivo do relatório
- 1.2 Contextualização do tema (roteadores e simulação de redes)
- 1.3 Breve descrição das etapas realizadas

## 2 Validação do Modelo do Roteador Simples

- 2.1. Descrição do Modelo Passado pelo Professor
- 2.2 Validação e correção do modelo

## 3 Modelagem de um Roteador Alterado

- 3.1. Proposta de Novo Modelo de Roteador ou Cenário com Múltiplos Roteadores
  - 3.1.1 Justificativa da escolha
  - 3.1.2 Descrição do cenário modelado (incluindo destinos e rotas)
- 3.2. Bases Teóricas (referências do KUROSE ou outras fontes)
  - 3.2.1 Resumo da seção 4.3 do livro sobre funcionamento interno do roteador

## 4 Configuração e Simulação de Cenários de Carga Diferentes

- 4.1 Justificar os cenários escolhidos e apresentar o modelo e alterações que foram testadas

## 5 Título Resultados e Análise de resultados

- 5.1 Dados coletados dos cenários

## 6 Considerações Finais

- 6.1 Conclusões gerais sobre as simulações e validações realizadas

6.2 Pontos de atenção e possíveis melhorias

## 7 Referências

7.1 **Livros, artigos, e recursos utilizados**

# 1 INTRODUÇÃO

O presente relatório tem como objetivo analisar e aprimorar um modelo simplificado de roteador desenvolvido no software Arena, conforme solicitado no Trabalho T2 da disciplina de Modelagem e Simulação. O estudo envolve três etapas principais:

- Verificação e validação do modelo original fornecido;
- Desenvolvimento de um novo modelo de roteador ou ampliação do cenário com múltiplos roteadores;
- Simulação do modelo sob diferentes condições de carga e análise dos resultados obtidos.

A simulação de redes é uma ferramenta amplamente utilizada para avaliar o desempenho de equipamentos de comunicação e prever comportamentos sob condições controladas. Roteadores, por sua vez, são dispositivos essenciais na camada de rede, responsáveis por encaminhar pacotes entre diferentes segmentos, aplicando técnicas de buffer, filas de entrada, escalonamento e gerenciamento de congestionamento.

Este relatório descreve em detalhes cada etapa do trabalho, com fundamentação teórica baseada no livro *Redes de Computadores e a Internet* de Kurose e Ross, além dos resultados obtidos com diferentes configurações de carga.

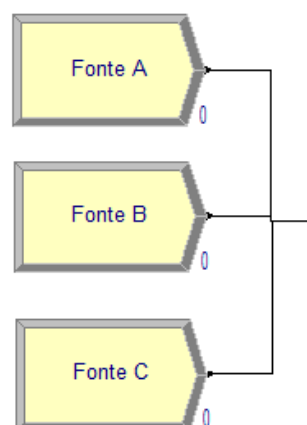
## 2 VALIDAÇÃO DO MODELO DO ROTEADOR SIMPLES

### 2.1 Descrição do Modelo Passado pelo Professor

O modelo original fornecido representa um roteador simples composto por três fontes de pacotes, um módulo de atribuição de destino, um mecanismo de teste de buffer, um módulo de processamento e um roteamento final para três destinos possíveis. A topologia lógica é linear e simplificada.

#### Fontes de Entrada

O sistema possui três entradas independentes:



- **Fonte A** – chegada determinística, tempo constante de 0,1 segundos entre pacotes.
- **Fonte B** – chegada imprevisível com distribuição **Exponencial(10)**.
- **Fonte C** – chegada imprevisível com distribuição **Exponencial(10)**.

Assim, a carga total do sistema é composta por uma fonte de alta frequência (Fonte A) combinada com duas fontes de tráfego esporádico.

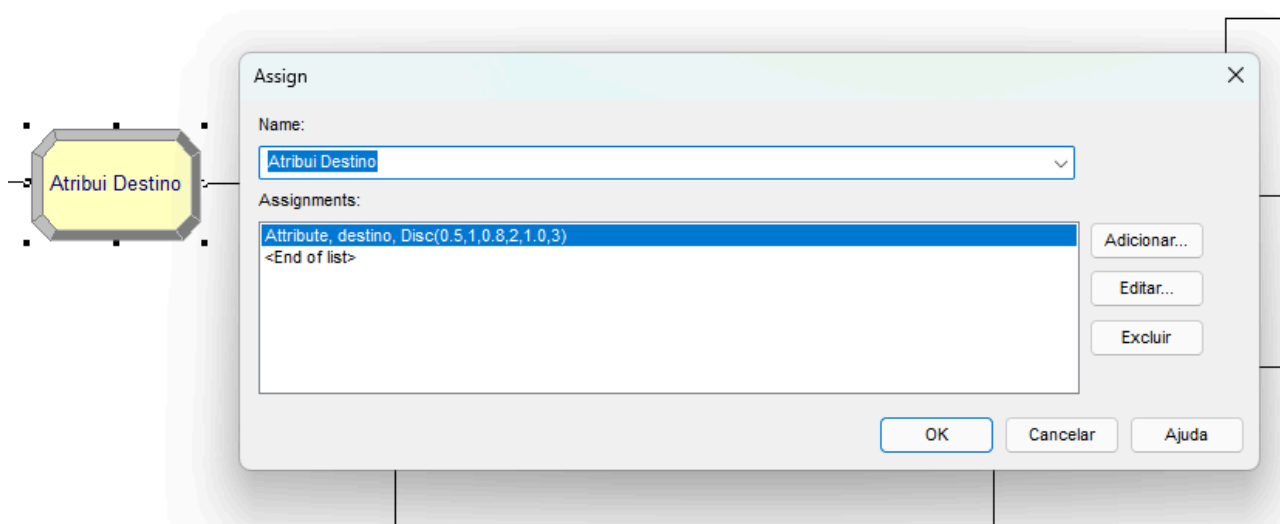
## Atribuição de Destino

O módulo **Assign** – “**Atribui Destino**” define o atributo “Destino” usando a distribuição discreta:

**Disc(0.5,1, 0.8,2, 1.0,3)**

Interpretando:

- 50% dos pacotes são enviados ao Destino 1.
- 30% ao Destino 2.
- 20% ao Destino 3.



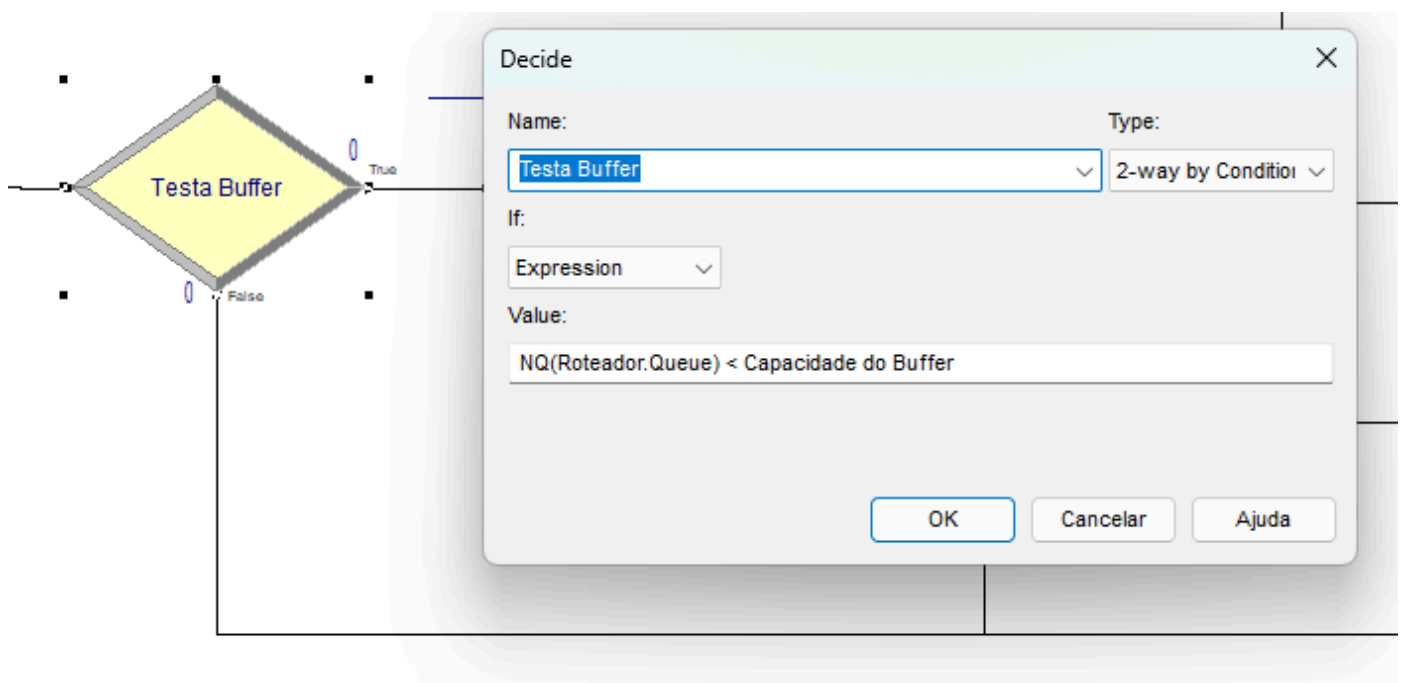
Esse atributo seria posteriormente utilizado para determinar o encaminhamento final do pacote.

## Teste de Buffer

O módulo Decide está configurado com a expressão

**$NQ(\text{Roteador.Queue}) < \text{Capacidade do Buffer}$  (valor inicial definido como 10)**

A intenção é verificar se o número de pacotes atualmente na fila interna do módulo Process é menor do que o tamanho máximo de buffer permitido, caso a fila seja maior, descarta os que estariam esperando.



## Processamento (Roteador)

O módulo representa o tempo gasto pelo roteador para processar um pacote e possui a seguinte configuração:

Process

Name: Roteador Type: Standard

Logic

Action: Seize Delay Release Priority: Medium(2)

Resources:

Resource, Processador, 1  
<End of list>

Adicionar...  
Editar...  
Excluir

Delay Type: Uniform Units: Seconds Allocation: Value Added

Minimum: 2 Maximum: 4

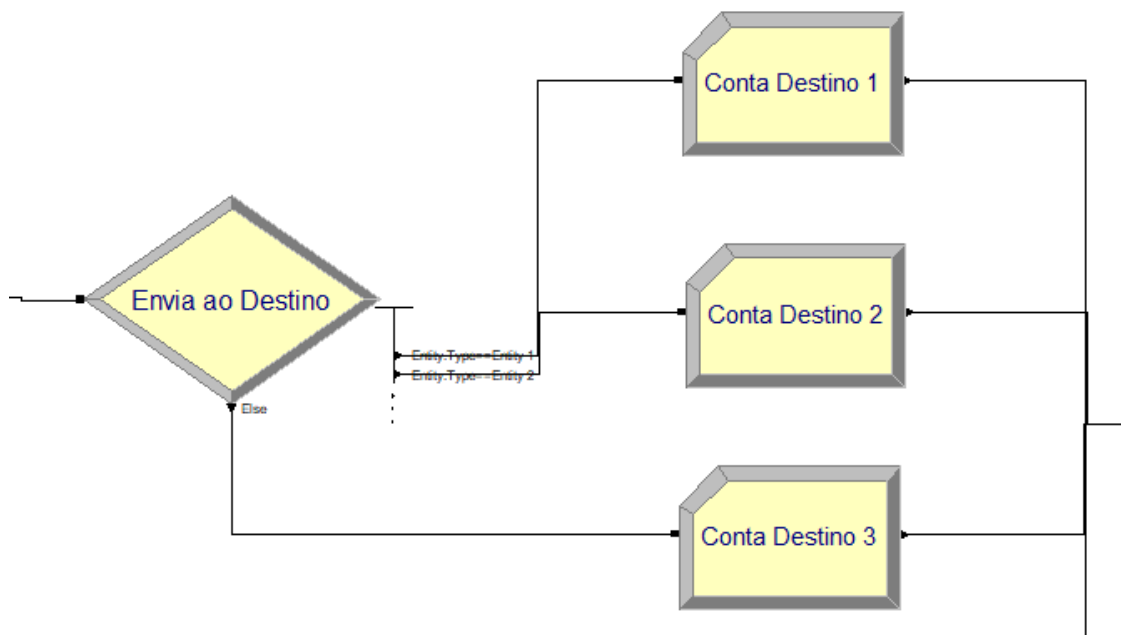
☒ Report Statistics

OK Cancelar Ajuda

- **Ação:** Seize–Delay–Release
- **Recurso:** Processador, capacidade 1
- **Fila interna:** Roteador.Queue (fila automática gerada pelo Arena)
- **Tempo de processamento:** Uniform(2, 4) segundos



## Encaminhamento Final



Após o processamento no módulo Process, o modelo utiliza um módulo Decide denominado **“Envia ao Destino”**. Esse Decide verifica o atributo “Entity Type” atribuído nos Creates e direciona o pacote para uma das rotas previstas:

- **Conta Destino 1**
- **Conta Destino 2**
- **Conta Destino 3**

## 2.2 Validação e Correção do Modelo

A validação revelou uma inconsistência severa no funcionamento lógico do roteador.

### Problema – Destino não é decidido pelo atributo destino

O módulo “Envia ao Destino” deveria verificar o destino atribuído no Assign. Porém, utiliza:

- Entity Type = 1 - Destino 1

- Entity Type = 2 - Destino 2
- Else - Destino 3

Isso significa que a distribuição  $\text{Disc}(0.5, 0.8, 1.0)$  não tem qualquer efeito no encaminhamento final, tornando o módulo Assign irrelevante.

Consequências diretas:

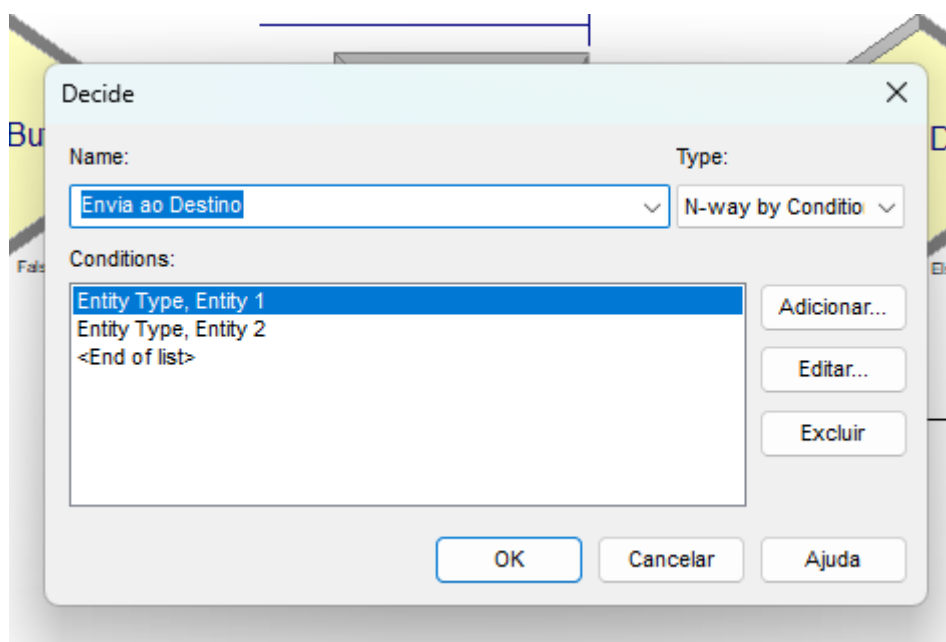
- A distribuição esperada 50/30/20% não ocorre.
- Quase todos os pacotes acabam indo para destinos inconsistentes com o Assign.

### Correção – Atribuir as condições corretas ao decide “Envia ao Destino”

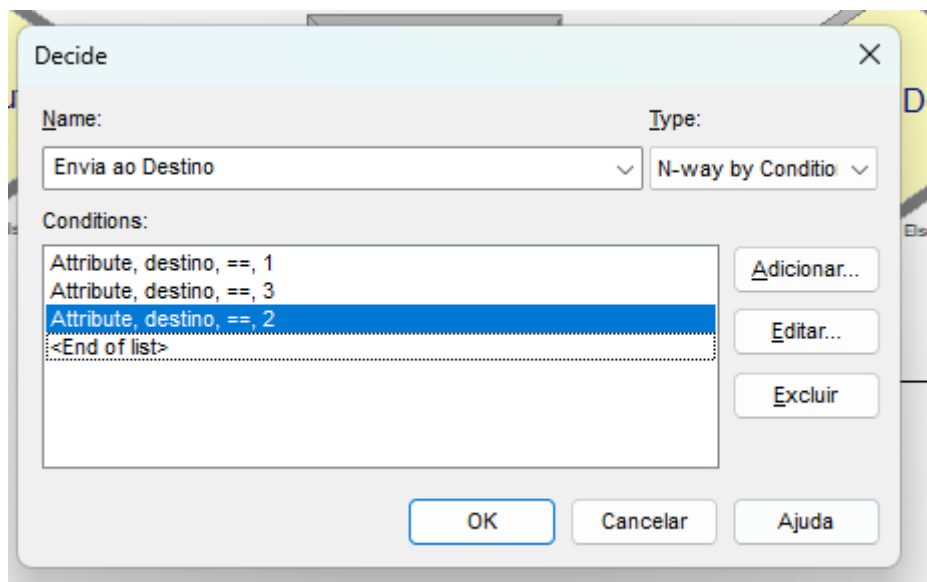
Como podemos observar ao realizar os testes do modelo, os Record's não estão distribuídos como esperado (50%, 30%, 20%) e sim contando a quantidade de entidades não descartadas.

Sumário para Replicação 1 de 1		
Projeto: Unnamed Project		
Analista: pf		
Replicação terminada no tempo de : 5000.0 Segundos		
Unidade de Tempo Base: Segundos		
CONTADORES		
Identificador	Limite Contagem	
Conta 2	12	Infinite
Conta 3	9	Infinite
Conta 1	1663	Infinite
descartes	49309	Infinite
Tempo executando a simulação: 0.00 minutos.		
Simulação completada.		

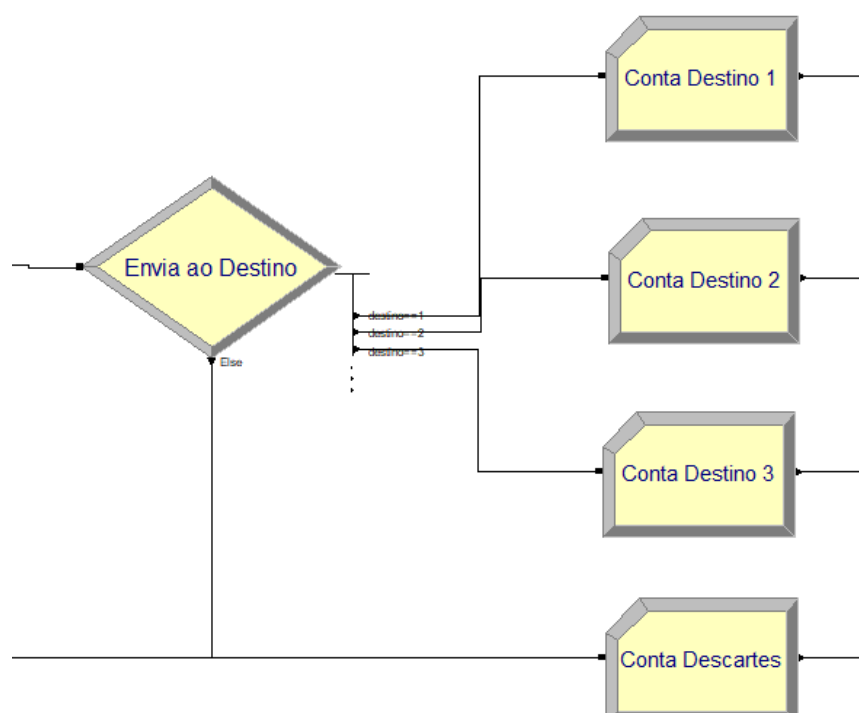
Após as seguintes alterações podemos observar uma grande melhora nos resultados, apesar do alto número de descarte.



Primeira mudança - Alteração das condições do Decide.



Segunda Mudança - Alteração da modelagem para ser compatível com a quantidade de condições e conexão do “Else” com “Descarte” para evitar erro de porta sem conexão de saída.



```
Sumário para Replicação 1 de 1

Projeto: Unnamed Project
Analista: pf

Replicação terminada no tempo de      : 5000.0 Segundos
Unidade de Tempo Base: Segundos

                                CONTADORES

Identificador                    Limite Contagem
-----
Conta 2                        503  Infinite
Conta 3                        371  Infinite
Conta 1                        810  Infinite
descartes                     49309 Infinite

Tempo executando a simulação: 0.02 minutos.
Simulação completada.
```

Resultado final após mudanças

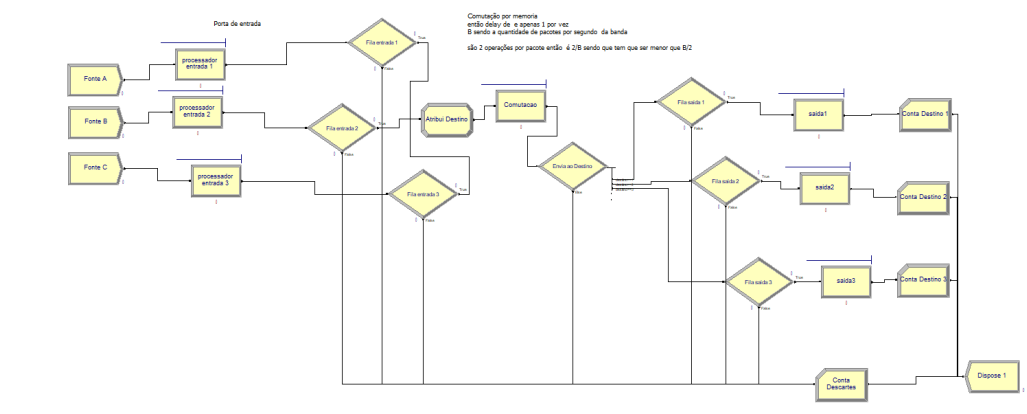
# 3 MODELAGEM DE UM ROTEADOR ALTERADO

## 3.1 Proposta de Novo Modelo de Roteador

### 3.1.1 Justificativa da escolha

A modelagem baseada em comutação por memória foi escolhida porque reproduz fielmente a arquitetura clássica descrita por Kurose e Ross, onde um único barramento limita a taxa de repasse dos pacotes a aproximadamente  $B/2$ . Esse mecanismo cria naturalmente um gargalo central, permitindo analisar de forma clara os efeitos de congestionamento, formação de filas, descarte e atraso — exatamente os fenômenos solicitados no trabalho. Além disso, mantém a estrutura do modelo original, facilitando comparação e implementação no Arena.

### 3.1.2 Descrição do modelo



#### Fontes de Entrada

Cada uma das três fontes (A, B e C) gera pacotes de acordo com distribuições de chegada configuradas para os cenários de simulação. Ao chegar ao roteador, cada pacote passa por um processador de entrada, representando, verificação inicial do pacote, etapas rápidas de pré-processamento e a checagem da integridade básica do cabeçalho.

Quando o processador de entrada libera o pacote, ele segue para uma fila de entrada, que representa o buffer de curto prazo da porta de entrada. Se a fila está cheia, o pacote é descartado.

### Lookup e Atribuição de Destino

Após entrar no roteador, o pacote passa por um módulo de **atribuição de destino**, que simula o “lookup” da tabela de encaminhamento, usando a distribuição discreta:

**Disc(0.5,1, 0.8,2, 1.0,3)**

Interpretando:

- 50% dos pacotes são enviados ao Destino 1.
- 30% ao Destino 2.
- 20% ao Destino 3.

Esse atributo é posteriormente utilizado para determinar a porta de saída escolhida após a comutação.

### Buffer Central e Comutação por Memória

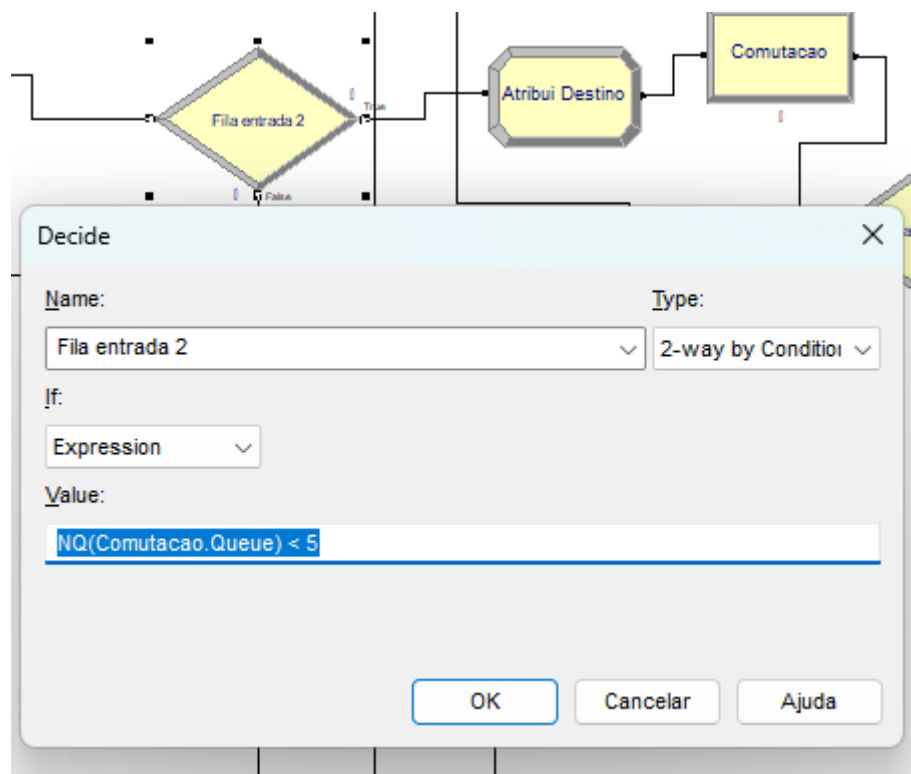
Todos os pacotes convergem para um único buffer central (Comutacao.Queue), que representa a memória compartilhada do roteador. Esse buffer tem capacidade limitada, configurada conforme o valor de B, escolhido como 20, com um processo de **Uniform(0.09, 0.11)** para ter variação, e o número permitido de operações de acesso por segundo.

O núcleo da comutação é um recurso com capacidade 1, simbolizando o barramento de memória. Apenas um pacote pode ser lido ou escrito por vez, exatamente como na comutação por memória descrita no livro.

O tempo de comutação é dado por:

$$\text{Tempo} = 2/B.$$

Sendo B a largura de banda da memória, representando: 1 operação para copiar o pacote para a memória (entrada); 1 operação para copiá-lo da memória para a porta de saída.



Caso  $NQ(\text{Comutacao.Queue}) > \text{Limite}$ , o pacote é descartado antes mesmo de tentar acessar o barramento, simulando saturação de buffer.

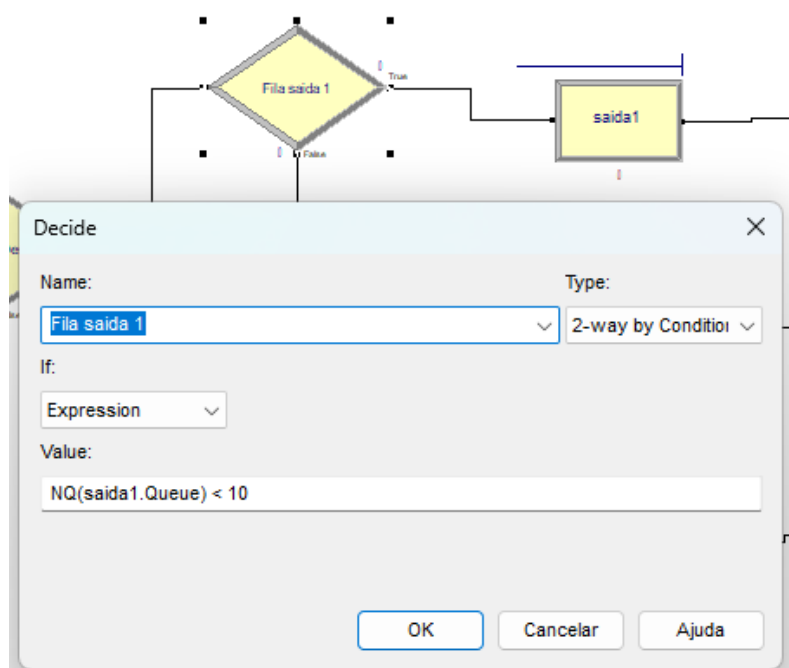


## Roteamento para as Portas de Saída

Após a comutação, o pacote passa pelo módulo “**Envia ao Destino**”, que verifica o atributo “Destino” e o direciona para uma das três portas de saída. Cada porta de saída é composta por:

- Uma fila de espera, representando o buffer da porta de saída;
- Um processador de porta de saída, simulando:
  - verificação de cabeçalhos,
  - controle de taxa,
  - envio físico pelo enlace.

Se a fila da porta de saída está cheia,  $NQ(saida1.Queue) > 10$ , o pacote é descartado.



Process

Name:  Type:

Logic

Action:  Priority:

Resources:

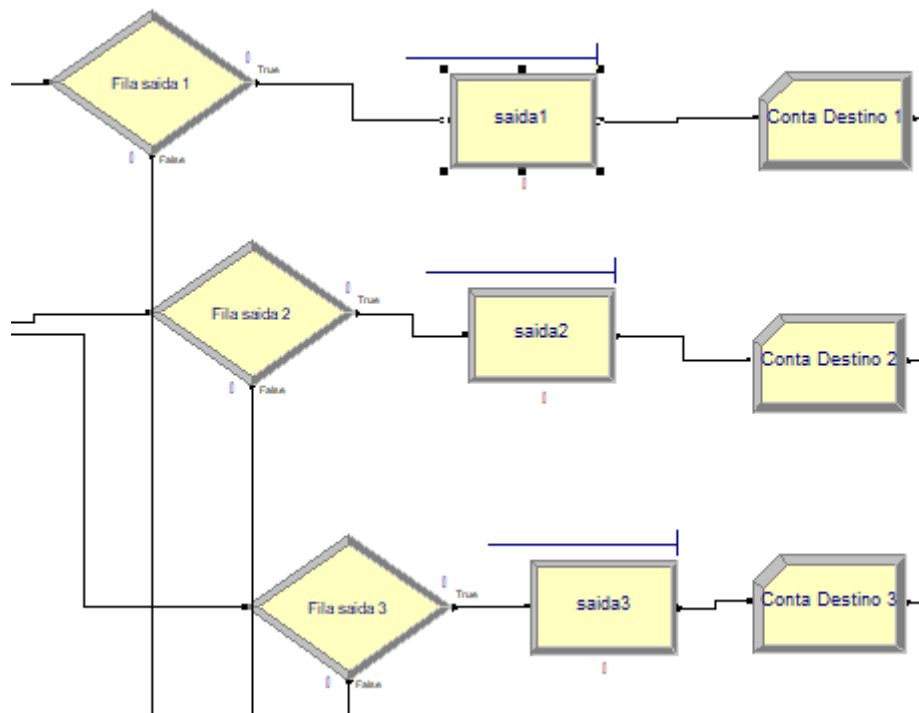
<End of list>

Delay Type:  Units:  Allocation:

Minimum:  Maximum:

☒ Report Statistics

## Encaminhamento Final



Os pacotes que completam o processamento saem do sistema em três módulos de destino:

- Conta Destino 1
- Conta Destino 2
- Conta Destino 3

e os descartados são enviados para **Conta Descartes**, registrando perdas por buffer de entrada e saída cheios.

## 3.2 Resumo da Seção 4.3 – “O que há dentro de um Roteador?” (Kurose & Ross)

A seção 4.3 descreve detalhadamente a arquitetura interna de um roteador e seus principais componentes. O livro divide o roteador em quatro subsistemas fundamentais, cada um responsável por uma etapa do processamento de pacotes dentro do equipamento.

### Portas de Entrada

O processamento começa nas portas de entrada, responsáveis por receber os pacotes da rede física, realizar funções de camada de enlace (como verificação de erros) e executar o *lookup* na tabela de encaminhamento para determinar a porta de saída apropriada. Nessa etapa também existe o buffer de entrada, utilizado quando os pacotes chegam mais rápido do que podem ser encaminhados para o elemento de comutação; se o buffer é excedido, os pacotes são descartados.

### Elemento de Comutação

O pacote então segue para o elemento de comutação, que faz a transferência interna entre as portas de entrada e de saída. O livro apresenta três arquiteturas possíveis: comutação via memória, comutação por barramento e comutação através de redes de interconexão. No caso da comutação por memória, usada em roteadores mais simples e também no modelo desenvolvido neste trabalho, a transferência ocorre através de leituras e escritas em uma memória compartilhada. Apenas uma operação pode ser executada por vez, o que limita a vazão máxima, que passa a ser menor que  $B/2$  pacotes por segundo, sendo  $B$  a taxa de leitura/escrita do barramento de memória.

### Portas de Saída

Após atravessar o elemento de comutação, o pacote chega às portas de saída, onde pode haver outro buffer, utilizado quando o enlace físico não está imediatamente disponível para transmissão. Esse buffer pode gerar filas caso muitos pacotes precisem

ser enviados para a mesma porta de saída, tornando-se um ponto crítico de congestionamento. As portas de saída também realizam funções de enlace, como formatação dos quadros e controle de taxa.

### **Processador de Roteamento**

Presidindo todo o funcionamento lógico do roteador está o processador de roteamento, responsável pela execução dos protocolos de roteamento, manutenção das tabelas de encaminhamento, tratamento de mensagens de controle e gerenciamento geral do equipamento. Embora não esteja diretamente no caminho de dados dos pacotes, ele é fundamental para o funcionamento do roteador como um todo.

A seção conclui destacando que o desempenho do roteador é influenciado pelo tamanho dos buffers, pela capacidade da arquitetura de comutação e pelas políticas de escalonamento adotadas. A ocorrência de congestionamento, tanto na entrada quanto na saída, e as consequentes perdas de pacotes fazem parte do funcionamento esperado em condições de tráfego elevado

# 4 CONFIGURAÇÃO E SIMULAÇÃO DE CENÁRIOS DE CARGA DIFERENTES

Simulação	Constantes das fontes	Limite de vazão (B/2)	Criação total de pkts
Baixa carga	0.5	10 pkt/s	1.5 pkt/s
Média carga	2	10 pkt/s	6 pkt/s
Altíssima carga	10	10 pkt/s	30 pkt/s

## Dados para testes de volume de carga

Em todos os testes foram utilizados os seguintes dados

1. Banda de memória (B) = 20 packets/s
2. Delay Comutacao =  $0.1 \text{ s} - 2/B$
3. Memoria\_Comutacao Capacity = 1
4. Processador\_Entrada Delay =  $\text{Uniform}(0.005, 0.02) \text{ s}$  Capacity = 1
5. Processador\_Saida Delay =  $\text{Uniform}(0.08, 0.12) \text{ s}$  Capacity = 1
6.  $\text{NQ}(\text{Comutacao.Queue}) < 5$
7.  $\text{NQ}(\text{Processador_SaidaX.Queue}) < 10$

# 5 RESULTADOS E ANÁLISES DE RESULTADOS

## 5.1 Dados fluxo baixo

Para este valor foi testado com intervalo de tempo nas fontes de 2 segundos para criação de 1 pacote. Foram realizadas 10 simulações de 5000.0 segundos, Com valores aproximados ao:

Identificador	Limite Contagem
---------------	-----------------

Conta 2	2146 Infinite
Conta 3	1559 Infinite
Conta 1	3795 Infinite
descartes	0 Infinite

## 5.2 Dados fluxo médio

Foram realizadas 10 simulações de 5000.0 segundos, Com valores aproximados ao de:

Identificador	Limite Contagem
---------------	-----------------

Conta 2	8861 Infinite
Conta 3	6229 Infinite
Conta 1	14910 Infinite
descartes	0 Infinite

## 5.3 Dados fluxo alto

Foram realizadas 10 simulações de 5000.0 segundos, Com valores aproximados ao de:

Identificador	Limite Contagem
---------------	-----------------

Conta 2	13599 Infinite
Conta 3	9061 Infinite
Conta 1	22675 Infinite
descartes	104662 Infinite



## 6 CONSIDERAÇÕES FINAIS

As simulações realizadas permitiram avaliar de forma detalhada o comportamento de um roteador modelado com arquitetura de comutação por memória e, ao mesmo tempo, evidenciar as limitações e inconsistências presentes no modelo original fornecido. A etapa inicial de validação mostrou problemas significativos, como a utilização incorreta do atributo “Destino”, e o desbalanceamento extremo entre a taxa de chegada e a capacidade de processamento. Com as correções implementadas, o novo modelo passou a refletir com maior fidelidade o funcionamento descrito por Kurose e Ross, permitindo análises mais consistentes.

Ao testar diferentes condições de carga — baixa, média e alta — observou-se que o roteador opera de forma eficiente em regimes de baixa e média entrada e entra em estado de congestionamento quando submetido a tráfego intenso. A arquitetura de comutação por memória, representada pelo barramento de capacidade limitada, demonstrou ser um ponto crítico do sistema, limitando diretamente a taxa máxima de repasse e influenciando o crescimento das filas internas. Da mesma forma, o tamanho do buffer revelou-se determinante para o desempenho: valores reduzidos levam a aumento de descartes, enquanto valores elevados reduzem perdas, mas introduzem maiores atrasos médios.

### Pontos de Atenção e Possíveis Melhorias

Alguns aspectos do modelo podem ser aprimorados em trabalhos futuros:

- A política de escalonamento utilizada foi FIFO em todas as filas. A inclusão de algoritmos de prioridade poderia representar cenários mais realistas de tráfego.
- O modelo assume um único processador por porta. Extensões com múltiplos processadores ou capacidades variáveis permitiriam simular roteadores de maior porte.

Em síntese, o modelo final desenvolvido é coerente, robusto e adequado para avaliação de desempenho em roteadores simples. Ele também estabelece uma base sólida para futuras extensões, permitindo a análise de estruturas mais complexas e de diferentes estratégias de gerenciamento de tráfego.

## 7 REFERÊNCIAS

**Kurose, J. F.; Ross, K. W.** *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. 6. ed. São Paulo: Pearson Addison Wesley, 2013.

**Rockwell Automation.** *Arena Simulation Software – User’s Guide*. Rockwell Automation, 2019.

**Freitas Filho, P. J.** *Introdução à Modelagem e Simulação de Sistemas com Aplicações em Arena*. 2. ed. Florianópolis: Visual Books, 2008. 372 p. ISBN 9788575022283.