Análise do RED e sua Influência na Autossimilaridade do Tráfego de Rede

Jorge Magno Lopes Moraes[1], **Arthur de Castro Callado**[2]

[1]Departamento de Computação
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Campus de Quixadá
Universidade Federal do Ceará (UFC) – Quixadá, CE – Brasil

jorge.pierrot@alu.ufc.br, arthur@ufc.br

Abstract. *With the discovery that network traffic has the characteristic of self-similarity, some studies have sought to decrease it because this characteristic causes some negatives effects, such as increased queuing delay and traffic congestion. Among the factors addressed in the last decades are queue management algorithms, such as Random Early Detection (RED). However, the researchers have not studied the influence of RED on self-similarity more deeply. After all, this algorithm has four configurable parameters (the queue weight, the maximum probability of drop, and the minimum and maximum thresholds), which, when modified, can lead to a change in performance and, consequently, in self-similarity. Therefore, this paper intends to verify the influence of RED on self-similarity. For this, we developed a standard to help in the setting of RED thresholds. Also, we show the impact of different threshold arrangements on self-similarity and the performance of network traffic. Besides, we compared some of the best settings of RED with Droptail.*

Resumo. *Com a descoberta de que o tráfego da rede possui a característica de autossimilaridade, alguns estudos buscaram diminuí-la, pois essa característica causa alguns efeitos negativos, como maior atraso na fila e congestionamento do tráfego. Entre os fatores trabalhados nas últimas décadas estão algoritmos de gerenciamento de filas, como Random Early Detection (RED). No entanto, os pesquisadores não estudaram a influência do RED na auto-similaridade mais profundamente. Afinal, esse algoritmo possui quatro parâmetros configuráveis (o peso da fila, a probabilidade máxima de queda e os limiares mínimo e máximo), que, quando modificados, podem levar a uma alteração no desempenho e, consequentemente, na autossimilaridade. Portanto, este artigo pretende verificar a influência do RED na auto-similaridade. Para isso, desenvolvemos um padrão para auxiliar na configuração dos limiares de RED. Também, mostramos o impacto de diferentes arranjos de limite na autossimilaridade e no desempenho do tráfego de rede. Além disso, comparamos algumas das melhores configurações de RED com Droptail.*

1. Introdução

Antes da pesquisa realizada por [Leland et al. 1994] acreditava-se que o tráfego de pacotes seguia o modelo de Poisson, o qual define que o tráfego se torna mais suave conforme

o número de fontes aumenta. No entanto, em redes de pacotes, o fluxo é formado por uma pluralidade de fontes de solicitações para o fornecimento de uma rede de serviços e aplicativos de rede que fornecem vídeo, dados, voz e outros serviços [Lozhkovskyi 2019]. Sendo assim, cada fonte gera seu próprio fluxo, diferindo desde a intensidade da carga ao número de aplicativos servidos. Nesse caso, o tráfego não é mais uma mera soma do número de fluxos independentes estacionários e comuns, mas uma combinação de fluxos caracterizado pelo chamado *burstiness* de tráfego com frequência aleatória e duração de picos e recessões [Lozhkovskyi 2019]. Dessa forma, [Leland et al. 1994] identificou que o tráfego de pacotes é autossimilar.

Com o aumento do grau de autossimilaridade do tráfego de pacotes, as características de qualidade do sistema se deterioram significativamente em comparação com a manutenção de tráfego de intensidade semelhante [Lozhkovskyi 2019], o que pode significar o aumento no atraso das filas e no congestionamento. Com isso, no início do século, alguns trabalhos foram realizados buscando a diminuição da autossimilaridade do tráfego e, consequentemente, de seus efeitos prejudiciais. Entre tais pesquisas estão [Sikdar et al. 2002a] e [Sikdar et al. 2002b], que buscaram diminuir a autossimilaridade a partir dos algoritmos de gerenciamento de fila, focando no *Random Early Detection* (RED). No entanto, ambos se concentraram em realizar modificações no funcionamento do RED, deixando de explorar seus parâmetros configuráveis, a saber, o peso da fila (w_q), o limiar mínimo (min_{th}), o limiar máximo (max_{th}) e a probabilidade máxima de descarte (max_p).

O RED é um algoritmo de *Active Queue Management* (AQM) proposto para implantação em redes com tráfego majoritariamente *Transport Control Protocol* (TCP) [Chandra et al. 2010], visando melhorar o desempenho em relação ao *Droptail*. Os objetivos iniciais do RED eram detectar o congestionamento, para alcançar a justiça entre os fluxos com diferentes níveis de rajadas, de modo a minimizar atraso de fila, evitar a sincronização global (quando o congestionamento faz com que várias fontes TCP reduzam significativamente suas taxas de transmissão ao mesmo tempo), minimizar a perda de pacotes e fornecer altas utilizações de enlaces [Adams 2013]. Assim, para sustentar o desempenho do RED para obter os resultados desejados, geralmente se ajustam os parâmetros de entrada, principalmente a probabilidade máxima de descarte do pacote [Abdel-Jaber 2020]. No entanto, RED tem alguns problemas de ajuste de parâmetro que precisam ser tratados com cuidado para que ele tenha um bom desempenho em diferentes cenários de rede [Chandra et al. 2010].

Diante do que foi comentado, fica evidente haver uma lacuna de pesquisa na relação autossimilaridade e os parâmetros do RED, pois os mesmos são importantes no desempenho do tráfego, o que, consequentemente, pode vir afetar a autossimilaridade. Além disso, notamos que os limiares mínimo e máximo do RED são pouco explorados, quando comparados ao parâmetro max_p. Sendo assim, diante da preocupação com os efeitos negativos que a autossimilaridade do tráfego de rede causa, o presente trabalho tem o objetivo de utilizar o RED e avaliar o impacto de diferentes configurações de seus limiares (min_{th} e max_{th}), além de criar um padrão para auxiliar na avaliação de tais parâmetros do RED.

O restante do artigo está organizado como segue: na seção 2, falamos um pouco mais em detalhes sobre os conceitos base do trabalho; na seção 3, apresentamos os traba-

lhos que tratam do RED e da autossimilaridade; na seção 4, comentamos sobre o padrão de configuração proposto para o RED; na seção 4.1 mostramos como as simulações foram projetadas e feitas; a seção 5 sumariza os resultados obtidos e as respectivas análises; e a seção 6, traz as considerações finais do trabalho e perspectivas futuras.

2. Fundamentação Teórica

Nessa seção, apresentamos os conceitos básicos de autossimilaridade aplicados a tráfego de rede e os principais aspectos referentes aos objetivos e funcionamento do RED, os quais são os principais pontos para o entendimento do trabalho.

2.1. Autossimilaridade

Autossimilaridade é um termo intimamente relacionado à invariância de escala, ou seja, as características não mudam em diferentes escalas. Portanto, se um objeto é semelhante a si mesmo, suas partes, quando ampliadas, lembram a forma do objeto inteiro [Kaur et al. 2020]. Dessa forma, a essência da autossimilaridade é que o mesmo padrão se repete em diferentes escalas sequenciais de tempo.

O primeiro trabalho a identificar a autossimilaridade no tráfego foi [Leland et al. 1994]. Usando certa quantidade de dados e uma análise estatística, a pesquisa demonstrou que o tráfego apresenta dependência de longo alcance (LRD - *Long Range Dependence*), que é um fenômeno estatístico mostrado em processos autossemelhantes [Yu et al. 2016], contrastando com os processos de dependência de curto alcance (SRD - *Short Range Dependence*), adotados anteriormente, que perdem sua explosão e se achatam quando as escalas de tempo são alteradas [Lokshina et al. 2020]. De fato, o tráfego da rede apresenta autossimilaridade estatística ao manter suas explosões em diferentes escalas de tempo (LRD), conforme visto na Figura 1. Em particular, o tráfego não suaviza rapidamente ao longo do tempo, mantendo uma tendência a explosões (rajadas).

Como a autossimilaridade é um processo aleatório, o grau de autossimilaridade pode ser determinado pelo coeficiente de Hurst, sendo capaz de analisar as séries temporais durante as quais o tráfego da rede foi coletado [Lysenko et al. 2020]. O expoente de Hurst, denotado por H, é a característica numérica mais simples de autossimilaridade estocástica e dependência de longo alcance [Jeong et al. 2017], sendo assim caracterizado os seus valores:

- Quando um valor de H está no intervalo de 0 até $0,5$ indica que os eventos são aleatórios e não há dependência de longo prazo entre eles, ou seja, SRD;
- Quando um valor de H está no intervalo de $0,5$ até 1 significa que o intervalo de tempo observado é uma série contínua de tempo com dependência de longo prazo, ou seja, LRD.

Portanto, o cálculo do coeficiente de Hurst deve estar no intervalo de $0,5$ à 1 para o tráfego ser denotado como autossimilar, sendo que quanto mais próximo de 1, maior a autossimilaridade [Lysenko et al. 2020]. O tráfego autossimilar foi identificado em diferentes sistemas como tráfego *Wide Area* [Paxson and Floyd 1995], *World Wide Web* [Crovella and Bestavros 1997] e redes ATM (*Asynchronous Transfer Mode*) [Tsybakov and Georganas 1998]. Em [Gong et al. 2005] é apontado que a autosimilaridade do tráfego da *Internet* é atribuída a uma mistura das ações de vários usuários

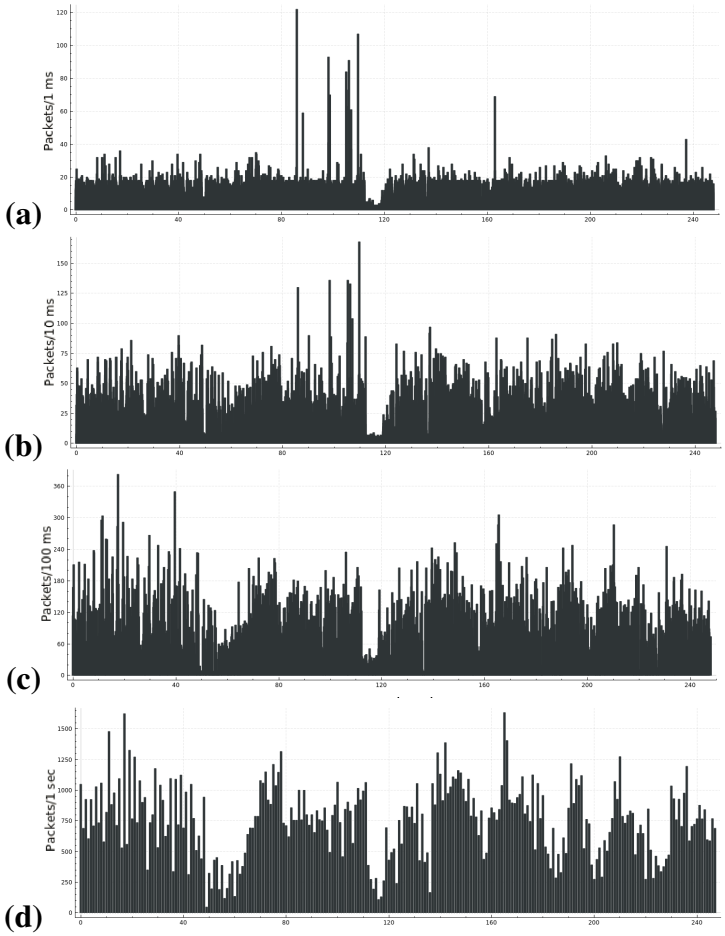

Figura 1. Tráfego em 4 escalas de tempo diferentes: (a) 1ms - (b) 10ms - (c) 100ms - (d) 1s.

individuais, comportamentos de *hardware* e *software* em seus *hosts* de origem, multiplexados através de uma rede de interconexão. Em outras palavras, a autossimilaridade sempre existe independentemente da rede, topologia, tamanho, protocolo ou serviços que a rede está transportando.

Por fim, conforme dito anteriormente, as características de qualidade do sistema se deterioram com o aumento do grau de autossimilaridade do tráfego de pacotes [Lozhkovskyi 2019], pois a rede fica sujeita a atrasos, congestionamentos e ineficiência no uso dos enlaces. Sendo assim, idealmente o tráfego deve apresentar o Hurst próximo a $0,5$ para não afetar o desempenho da rede.

2.2. Random Early Detection (RED)

O RED é um algoritmo de gerenciamento de fila ativo que foi criado há mais de duas décadas para resolver os problemas de injustiça e problemas de sincronização global gerados no *Droptail* [Hamdi et al. 2018]. Este algoritmo foi projetado com os objetivos de minimizar a perda de pacotes e o atraso de enfileiramento, evitar a sincronização global de fontes, manter alta utilização do enlace e remover vieses contra fontes em rajadas [Gharegozi 2010]. O interesse de nosso trabalho ao usar o RED é saber o impacto do algoritmo sobre a autossimilaridade do tráfego de rede e, consequentemente, sobre o de-

sempenho.

Em sua operação, o RED utiliza quatro parâmetros fixos, a saber: o peso da fila (w_q), a probabilidade máxima de descarte (max_p) e os limiares mínimo e máximo (min_{th} e max_{th}). O valor de w_q é utilizado para calcular o tamanho médio da fila. Os valores max_{th} e min_{th} são usados como parâmetros de controle para detecção de congestionamento e são determinados pelo tamanho médio da fila desejável naquele dispositivo. Conforme explicado em [Santhi and Natarajan 2011], quando um pacote chega à fila ocorrem três casos para o descarte de pacotes:

- Quando o tamanho médio da fila é menor que o min_{th}, nenhum pacote é descartado;
- Quando o tamanho médio da fila é maior que o max_{th}, todos os pacotes são descartados;
- Quando o tamanho médio da fila está entre esses dois limites (min_{th}, max_{th}), os pacotes são descartados com base na probabilidade de máxima de descarte (max_p).

Portanto, o RED possui duas funções. Uma para calcular o tamanho médio da fila para determinar o grau de rajada que irá ser permitida na fila e outra para calcular a probabilidade com que os pacotes serão descartados dado um determinado nível de congestionamento.

A possibilidade de modificações dos valores do RED pode ser útil, quando consideramos que cada rede possui uma realidade diferente em relação à quantidade de dados, ao tamanho dos conjuntos nos quais são transmitidos, categorias de protocolos, dentre outros aspectos. Assim, adaptar os parâmetros do algoritmo a cada caso, pode trazer ganhos como otimizar o gerenciamento de filas.

3. Trabalhos Relacionados

Devido aos seus impactos sobre a rede, existem trabalhos na literatura que se dedicaram a pesquisar e estudar a autossimilaridade com o RED. Além disso, há trabalhos que estudaram a melhora no desempenho da rede com o uso do RED. Nessa seção, expomos alguns desses trabalhos.

Em [Sikdar et al. 2002a], os autores investigaram o impacto dos algoritmos de gerenciamento de filas na autossimilaridade do tráfego. Além disso, eles propuseram uma modificação do algoritmo RED para reduzir tempos limite e reversões exponenciais em fluxos TCP e mostram que isso pode levar a reduções significativas na autossimilaridade de tráfego em uma ampla gama de condições de rede. Porém, [Sikdar et al. 2002a] modifica o RED sem explorar diferentes configurações de parâmetros, limitando o estudo.

Seguindo a mesma linha, em [Sikdar et al. 2002b], o algoritmo RED também foi modificado para reduzir os limites de tempo e reversões exponenciais no fluxo TCP, reduzindo assim a autossimilaridade do tráfego. Em relação ao nosso trabalho, ele é semelhante ao foco em diminuir a autossimilaridade e trabalhar com o algoritmo RED, porém partindo do princípio de modificar o RED.

Objetivando atingir baixa taxa de perda e alto rendimento, bem como alta utilização do enlace, [Patel 2014] modificou o RED, alterando a probabilidade de

marcação do pacote, para alcançar a estabilização no comprimento da fila em roteadores. Posteriormente, o desempenho do RED modificado é comparado ao RED original. Sendo assim, este trabalha busca a melhora do RED, porém modificando-o e explorando apenas um parâmetro do RED, a probabilidade de marcação.

[Rastogi and Zaheer 2016] apresenta uma comparação de três mecanismos de enfileiramento, ou seja, *Droptail*, RED e *Nonlinear Random Early Detection* (NLRED) com base em diferentes métricas de desempenho. A comparação é feita para saber qual mecanismo é adequado, de acordo com a rede e o tráfego. O estudo mostrou que conforme o congestionamento aumenta, o RED e o NLRED se tornam melhores opções. No entanto, a comparação acaba se limitando ao não variar os valores dos parâmetros do RED e do NLRED.

Em [Mohammed et al. 2017] um novo algoritmo baseado em RED é desenvolvido, chamado *Dynamic Queue RED* (DQRED), para garantir os requisitos de qualidade de serviço (QoS) para as aplicações de multimídia em tempo real. Além disso, evita a privação do tráfego de melhor esforço (aplicativos baseados em TCP) na presença de tráfegos não baseados em TCP (aplicativos em tempo real), especialmente em cargas de tráfego pesadas. O DQRED é comparado com o RED, porém nos experimentos altera-se apenas o número de fontes, não explorando os parâmetros do RED.

[Alkharasani et al. 2017] apresenta um novo algoritmo baseado no RED. o *Fair Weighted Multi-level Random Early Detection (FWMRED)*. Esta pesquisa compara o desempenho de fluxos de longa duração na classificação de redes de tráfego usando o RED, o *Adaptive Random Early Detection* (ARED) e o FWMRED. Para esta comparação, novas definições para os valores de peso da fila (w_q) e probabilidade máxima de descarte (max_p) dos três algoritmos são feitas para um tratamento otimizado para as compensações entre descarte de pacotes, atraso e taxa de transferência com rede altamente carregada. Apesar disso, nos experimentos não há uma variação dos valores dos limiares mínimo e máximo dos algoritmos.

Também voltado para o desempenho das métricas de rede, [Hamdi et al. 2018] compara vários algoritmos de gerenciamento de fila, entre os quais está o RED e o ARED. Mesmo fazendo uma avaliação de vários algoritmos e seus respectivos impactos no desempenho da rede, a pesquisa acaba se limitando ao não variar nas configurações dos parâmetros no RED e no ARED.

Em [Abdel-Jaber 2020] é proposto um novo método AQM exponencial denominado RED-Exponencial (RED-E). A técnica proposta difere de RED e seus sucessores por descartar os pacotes que chegam de uma maneira exponencial ao invés de ajustar a probabilidade máxima de descarte (max_p). Este processo exponencial de descarte de pacotes visa minimizar a dependência de parâmetros pré-ajustados. O RED-E, então, é comparado com o RED e o NLRED e na avaliação são explorados diferentes valores para os limiares (min_{th} e max_{th}) e para o peso da fila (w_q).

A análise dos trabalhos expostos mostra a relação existente entre os parâmetros de configuração do RED e o desempenho da rede. No entanto, poucos trabalhos estudaram a variação nos valores dos limiares do RED e o seu impacto no nível de autossimilaridade do tráfego. Assim, torna-se importante analisar esses parâmetros e verificar o impacto de diferentes configurações no desempenho da rede e na autossimilaridade do tráfego.

4. Padrão Proposto

Nesta seção falaremos do padrão proposto para a avaliação de diferentes configurações do RED, o qual será aplicado nas simulações.

Conforme dito anteriormente, os limiares do RED são usados como parâmetros de controle para detecção de congestionamento, sendo assim, min_{th} e max_{th} podem afetar diretamente a autossimilaridade do tráfego. Dessa forma, o padrão proposto se concentra na variação dos valores dos limiares.

A criação do padrão de configuração dos limiares do RED foi definida a partir das recomendações de [Floyd and Jacobson 1993], a saber: (i) os valores de min_{th} e max_{th} devem ser suficientemente altos; (ii) a diferença entre os limiares deve ser grande e (iii) max_{th} tem que ser no mınimo duas vezes maior que min_{th}. Seguindo esses preceitos, criamos o seguinte padrão:

- min_{th} deve corresponder no mínimo a 10% em relação ao tamanho total da fila e no máximo 45%;
- max_{th} deve corresponder no mínimo a 60% em relação ao tamanho total da fila e no máximo 90%;
- A diferença entre min_{th} e max_{th} deve ser no mínimo 40%.

Como pode ser notado, o padrão se refere a porcentagem em relação ao tamanho da fila e não a valores exatos. Optamos por assim fazer dado que os tamanhos das filas podem variar. Para criar um padrão geral, o uso da porcentagem é mais interessante.

4.1. Execução dos Experimentos

Nessa subseção, apresentamos o cenário modelado, a metodologia de execução dos experimentos e os recursos utilizados.

Com o objetivo de avaliar os impactos das configurações do RED sobre a autossimilaridade do tráfego de rede, realizamos experimentos divididos em dois propósitos: (i) identificar as melhores configurações min_{th} e max_{th} em diferentes cenários de carga de tráfego e (ii) comparar os resultados obtidos com outro algoritmo de gerenciamento de filas, o *Droptail*. O *Droptail* foi escolhido por ser um mecanismo simples utilizado em roteadores e ter uma abordagem diferente da adotada pelo RED. Nele, sempre que a fila está cheia, os pacotes que chegam são descartados até que seja gerado um espaço adequado para aceitar novos pacotes. Logo, além de simples, é o mais utilizado por apresentar um alto grau de eficiência [Patel and Patel 2019].

Para fins de modelagem dos cenários, foi escolhida a topologia *em halteres (dumbbell)*, conforme mostrado na Figura 2. Como essa topologia é formada por dois roteadores centrais conectados por um enlace (gargalo) e vários outros roteadores a eles conectados (folhas), é possível configurar os enlaces de modo a gerar congestionamentos no gargalo.

Com relação aos parâmetros, utilizamos as mesmas taxas de transmissão e *delays* que em [Sikdar et al. 2002b], por se adequarem aos objetivos aqui delineados. O tamanho da fila nos roteadores foi definido em 100 pacotes para facilitar a aplicação do padrão proposto neste trabalho. Além disso, foi usado o tráfego TCP, visto que o RED foi projetado para lidar com um protocolo da camada de transporte que realize controle de congestionamento. Por fim, os valores do peso da fila (w_q) e probabilidade máxima de descarte (max_p) do RED foram estabelecidos segundo recomendações

de [Floyd and Jacobson 1993], sendo valores replicados em outros trabalhos, como em [Abdel-Jaber 2020]. Na Tabela 1 pode-se visualizar os parâmetros definidos.

Tabela 1. Lista de parâmetros para a simulação.

Parâmetro	**Valor**
Taxa de transmissão do gargalo	1 Mbps
Taxa de transmissão das folhas	10 Mbps
Delay do gargalo	100 ms
Delay das folhas	10 ms
Tamanho da fila de pacotes	100
Peso da fila (w_q)	0,002
Probabilidade máxima de descarte (max_p)	0,1
Tipo de tráfego	TCP

Utilizando a topologia e os parâmetros indicados, criamos três cenários com o intuito de simular diferentes volumes de tráfego: leve, com 30 fontes TCP; moderado, com 150 fontes, e pesado, com 240. A escolha do número de fontes foi feita empiricamente através de testes preliminares. Outro fator definido diz respeito aos valores de min_{th} e max_{th} do RED, os quais foram escolhidos conforme o padrão proposto na Seção 4. Dessa forma, foram estabelecidas as seguintes configurações (os valores abaixo também podem ser vistos como porcentagem devido o tamanho da fila ser de 100 pacotes):

- Para max_{th} duas vezes maior que min_{th}: 40-80, 41-82, 43-86, 44-88, 45-90;
- Para max_{th} três vezes maior que min_{th}: 20-60, 21-63, 22-66, 23-69, 24-72, 25-75, 26-78, 27-81, 28-84, 27-87, 30-90;
- Para max_{th} quatro vezes maior que min_{th}: 15-60, 16-64, 17-68, 18-72, 19-76, 20-80, 21-84, 22-88
- Para max_{th} cinco vezes maior que min_{th}: 12-60, 13-65, 14-70, 15-75, 16-80, 17-85, 18-90;
- Para max_{th} seis vezes maior que min_{th}: 10-60, 11-66, 12-72, 13-78, 14-84, 15-90;
- Para max_{th} sete vezes maior que min_{th}: 10-70, 11-77, 12-84;
- Para max_{th} oito vezes maior que min_{th}: 10-80, 11-88;
- Para max_{th} nove vezes maior que min_{th}: 10-90.

Mediante os objetivos propostos, foram escolhidas três métricas: *nível de autossimilaridade* do tráfego, mensurado pelo parâmetro de *Hurst*; o *goodput*, diz respeito à

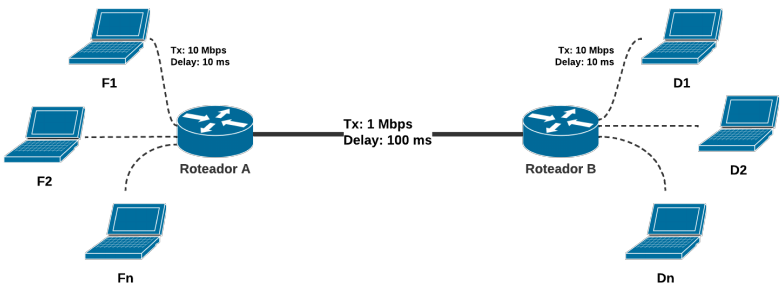

Figura 2. Topologia usada nos experimentos. Adaptada de [Sikdar et al. 2002b]

quantidade de dados úteis em uma conexão desprezando descartes e retransmissões e a *taxa de descarte*, porcentagem de pacotes descartados pelo roteador de gargalo configurado com o RED. Tais métricas foram escolhidas por possibilitar uma visão dos impactos das variações dos valores de min_{th} e max_{th} no tráfego e foram usadas tanto na identificação dos melhores arranjos de configuração do RED quanto na comparação com o *Droptail*.

A topologia escolhida foi implementada no *Network Simulator 3* (NS-3) e o tráfego de rede foi gerado em sentido único, de modo a evitar congestionamento no sentido contrário. Sendo assim, os nós do lado esquerdo enviavam pacotes para os do lado direito. A carga de trabalho nas simulações foi gerada utilizando o *Poisson Pareto Burst Process* (PPBP), um gerador de tráfego de rede simples e preciso que corresponde às propriedades estatísticas de redes reais [Ammar et al. 2011].

Nesse arranjo, o ponto de análise escolhido foi a fila de saída no roteador A para o enlace de gargalo (Figura 2). Nesse ponto da rede, foram coletados os dados utilizando o sistema de rastreamento ASCII do NS-3. Em seguida, foi utilizado um *script* para contar o número de pacotes recebidos pela fila do roteador escolhido a cada $0,1$ segundo. Para estimar o parâmetro de *Hurst*, usamos a Análise R/S, um dos métodos mais usados e simples [Patil et al. 2011]. É importante frisar que cada simulação foi executada por 320 segundos, porém as coletas só foram realizadas entre os instantes 10 e 310, de modo a evitar a captação de dados transitórios. Assim, foi obtido um total de 3000 amostras por execução.

Por fim, o cálculo do *goodput* do enlace de gargalo foi feito por meio do módulo *PacketSink* do NS-3, o qual recebe e consome tráfego para um endereço e porta [NS-3 2020a]. Já a taxa de descarte foi obtida recorrendo ao *QueueDiscStats*, uma estrutura que mantém as estatísticas da fila do roteador [NS-3 2020b].

5. Resultados

Com a finalidade do melhor entendimento dos resultados das simulações, esta seção está dividida em quatro partes. A primeira com os resultados das simulações com carga de tráfego leve (Cenário A), a segunda com carga de tráfego moderada (Cenário B), terceira com carga de tráfego pesada (Cenário C) e quarta parte contendo as considerações gerais sobre os resultados dos cenários. Por questão de espaço apresentamos apenas os resultados das configurações do RED com melhores e piores desempenho, exemplificando um contraste intuitivo, juntamente com os resultados do Droptail. Por fim, os resultados exibidos nas tabelas correspondem a uma execução do cenário, sendo assim, não há variabilidade nos experimentos.

5.1. Cenário A

Neste cenário as simulações foram realizadas contendo 30 fontes TCP na topologia em halteres (*dumbbell*) com o objetivo de gerar tráfego com carga leve.

Na Tabela 2, podemos observar que entre as configurações do RED, a diferença nos resultados de *goodput* e descarte foram pequenas, havendo maiores diferenças nos resultados no parâmetro de Hurst. Dessa forma, duas configurações RED se sobressaíram, a 45-90, que obteve a menor taxa de descarte, e a 11-88, que obteve o melhor Hurst. Porém, entre ambas, consideramos que a configuração 45-90 apresenta uma melhor performance,

pois além de apresentar a menor taxa de descarte dentre todas, mostra o *goodput* tão bom quanto os demais e o Hurst tão baixo quanto a configuração 11-88.

Tabela 2. Resultados do RED e Droptail no Cenário A.

Algoritmo	Hurst	Taxa de Descarte (%)	Goodput (Mbps)
Droptail	0,7072	1,37	0,8940
RED (11-88)	0,6505	1,16	0,8900
RED (21-63)	0,7075	1,84	0,8959
RED (29-87)	0,6561	1,13	0,8884
RED (45-90)	0,6533	1,12	0,8936

Ainda na Tabela 2, quando comparamos as configurações do RED com o *Droptail*, é possível notar que a autossimilaridade (Hurst) e a taxa de descarte do *Droptail* são piores que quase todas as configurações do RED, sendo melhor apenas que a 21-63, no que se refere ao descarte. Em relação ao *goodput*, o desempenho do *Droptail* foi equivalente aos demais, sendo melhor apenas que a 29-87. De modo geral, o RED obteve melhor desempenho que o *Droptail* quando o tráfego se mostra leve, uma vez que seja bem configurado.

5.2. Cenário B

Neste cenário as simulações foram realizadas contendo 150 fontes com a finalidade de gerar tráfego com carga moderada.

Na Tabela 3, podemos observar que a diferença do parâmetro de Hurst nas configurações do RED foram pequenas (variando entre $0,64$ e $0,65$), porém no *goodput* e no descarte as diferenças foram mais significativas. Diante disso, destacamos três configurações RED, a saber: 10-90, 21-63 e 25-75. A configuração 10-90 apresentou o melhor Hurst e taxa de descarte, porém seu *goodput* foi o pior dentre todos, obtendo um desempenho até 10% inferior que outras configurações. A 21-63 teve o melhor *goodput*, sendo bem superior à maioria das configurações, porém o seu descarte é elevado, além de que o Hurst esteve em torno de $0,65$, o que neste cenário representa estar entre as piores. Por fim, definimos a configuração 25-75 como a mais indicada, pois o seu desempenho em todas as métricas está próximo das melhores. No entanto, vale ressaltar que dependendo da rede, as configurações 10-90 e 21-63 podem ser opções a serem consideradas diante de seus ganhos em descarte e *goodput*, respectivamente.

Quando observamos o desempenho do *Droptail* na Tabela 3 em detrimento as configurações do RED, nota-se que a taxa de descarte é mais baixa que quase todas do RED. Apesar disso, o seu Hurst ainda é ruim, quando comparado aos melhores resultados do RED, assim como o *goodput*. Sendo assim, no âmbito geral, o RED obtém melhor desempenho com tráfego moderado, novamente, desde que corretamente configurado.

5.3. Cenário C

No Cenário C, as simulações foram realizadas contendo 240 fontes TCP para gerar tráfego com carga pesada.

Na Tabela 4, verifica-se que o Hurst de todas as configurações do RED ficaram por volta de $0,64$ sendo mínima a diferença. No entanto, no descarte e *goodput* houve

Tabela 3. Resultados do RED e Droptail no Cenário B.

Algoritmo	Hurst	Taxa de Descarte (%)	Goodput (Mbps)
Droptail	0,6503	11,38	0,7650
RED (10-90)	0,6455	10,58	0,7511
RED (13-78)	0,6474	11,91	0,7732
RED (15-60)	0,6567	14,31	0,8520
RED (21-63)	0,6543	14,17	0,8523
RED (25-75)	0,6474	12,80	0,8077

diferenças maiores. Sendo assim, ressaltamos duas configurações RED, a 21-63, com melhor *goodput*, e a 30-90, com menor taxa de descarte. A escolha entre ambas depende do que se busca, se um melhor *goodput* do enlace ou menor descarte no roteador, afinal a diferença entre os resultados destas configurações está em torno de 3%.

Agora, comparando o *Droptail* com o RED na Tabela 4, vemos que o *Droptail* apresentou a melhor taxa de descarte, tendo um descarte 3% inferior ao 30-90, que obteve melhor resultado nessa métrica no RED. Porém, o *goodput*, assim como no cenário anterior, teve um desempenho menor, sendo melhor apenas que a configuração 14-84 do RED. Vale destacar ainda, que apesar de ter o Hurst em $0,64$, a autossimilaridade do tráfego com *Droptail* foi a pior (mais alta), mesmo que por pouca diferença. Assim, no contexto geral, com tráfego intenso e pesado, a opção por *Droptail* ou RED também dependerá da preferência entre o melhor aproveitamento do enlace (RED) ou menor descarte nas filas dos roteadores (*Droptail*).

Tabela 4. Resultados do RED e Droptail no Cenário C.

Algoritmo	Hurst	Taxa de Descarte (%)	Goodput (Mbps)
Droptail	0,6491	16,95	0,7247
RED (10-60)	0,6443	23,80	0,7661
RED (12-72)	0,6425	22,42	0,7480
RED (14-84)	0,6445	21,48	0,7202
RED (21-63)	0,6474	23,16	0,7680
RED (30-90)	0,6460	20,17	0,7309
RED (41-82)	0,6489	21,75	0,7303

5.4. Discussão dos Resultados

Tomando com base os resultados obtidos nos três cenários, podemos destacar alguns pontos importantes. O primeiro deles refere-se à diminuição da autossimilaridade e do *goodput* e aumento da taxa de descarte da fila do roteador conforme o tráfego aumenta. Isso se explica pelo aumento do fluxo de dados na rede que reduz a autossimilaridade, já que os fluxos se tornam mais constantes, o que gera congestionamento, piorando o *goodput* e resultando em mais descartes.

Com relação às configurações min_{th} e max_{th} do RED, notou-se que diferentes valores nos limiares influenciam a autossimililaridade da rede, mesmo que de maneira mínima, como ocorreu nos cenários de tráfego moderado e pesado. Além disso, viu-se

que as diferentes configurações RED também influenciam o *goodput* do enlace e na taxa de descarte do roteador, mostrando que ao optar pelo RED é necessário avaliar diferentes configurações de min_{th} e max_{th}, pois o desempenho pode variar significativamente dependendo da escolha.

Quando comparado ao *Droptail*, observou-se que o RED é uma opção melhor quando o tráfego de rede é leve ou moderado, porém, à medida que o tráfego aumenta, ambos os algoritmos de gerenciamento de fila tornam-se equivalentes, pois o *Droptail* descarta menos pacotes enquanto o RED apresenta melhor *goodput*. Isso se deve ao modo de funcionamento de cada algoritmo, pois, como dito antes, o RED foi desenvolvido para lidar melhor com o congestionamento que o *Droptail*, o que se mostra verdadeiro pelo melhor *goodput* do RED em todos os cenários. No entanto, tem como consequência o aumento do descarte no roteador em relação ao *Droptail*. Além disso, o *Droptail* apresentou em todos os cenários o nível de autossimilaridade pior que as melhores configurações do RED, mesmo que minimamente. Com isso, e levando em conta que o *Droptail* está sujeito a alguns problemas como a sincronização global, consideramos que o RED bem configurado é uma opção melhor.

6. Conclusão

Nesse trabalho, abordamos a autossimilaridade no tráfego de rede, estudando o impacto da variação dos limiares do RED sobre esse fenômeno através de simulações no NS-3, técnica de avaliação que foi escolhida pela limitação de recursos e por haver um simulador de rede bem desenvolvido disponível que atendia as necessidades do trabalho. Além disso, observamos como essa variação influencia em outros aspectos de desempenho. A avaliação foi realizada em diferentes cenários de volume de tráfego e criando uma topologia para gerar congestionamento.

Os resultados obtidos apontaram que diferentes configurações de min_{th} e max_{th} do RED influenciam na autossimilaridade do tráfego, assim como nos índices de *goodput* e taxa de descarte de pacotes. Observamos também que diferentes configurações apresentaram melhor desempenho em cada caso, o que mostra a necessidade de analisar e escolher bem os limiares. Por fim, vimos que o RED apresenta melhor desempenho em praticamente todos os cenários *Droptail*, sendo que com tráfego pesado cada algoritmo apresenta um aspecto melhor em relação ao outro.

Assim, ao optar pela utilização do RED, é preciso estudar bem o tráfego de rede e avaliar diferentes configurações dos limiares, pois, como foi observado, se mal configurado, o desempenho da rede é afetado negativamente em relação a alguma métrica. Para isso, o padrão que propomos na Seção 4 se apresenta válido, afinal ele auxilia na escolha nos valores dos limiares a serem usados.

Como desdobramentos deste trabalho, seria relevante ampliar a análise aqui executada adicionando variabilidade aos experimentos, uma vez que alguns valores se apresentaram próximos. Além disso, pretendemos simular ou emular uma topologia real, como a Rede Nacional de Pesquisa (RNP). Também pretendemos ampliar a análise para outros algoritmos de gerenciamento de fila que foram desenvolvidos a partir do RED, como o Adaptive RED e o Nonlinear RED, isso permitiria uma observação mais ampla e concreta em relação a autossimilaridade do tráfego, além de podermos aplicar nosso padrão proposto nesses outros algoritmos.

Referências

Abdel-Jaber, H. (2020). Can exponential active queue management method based on random early detection. *Journal of Computer Networks and Communications*, 2020:1–11.

Adams, R. (2013). Active queue management: A survey. *IEEE Communications Surveys and Tutorials*, 15(3):1425–1476.

Alkharasani, A. M., Othman, M., Abdullah, A., and Lun, K. Y. (2017). An improved quality-of-service performance using red's active queue management flow control in classifying networks. *IEEE Access*, 5:24467–24478.

Ammar, D., Begin, T., and Guerin-Lassous, I. (2011). A new tool for generating realistic internet traffic in ns-3. In *SIMUTools '11: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 81–83. ACM.

Chandra, H., Agarwal, A., and Velmurugan, T. (2010). Analysis of active queue management algorithms their implementation for tcp/ip networks using opnet simulation tool. *International Journal of Computer Applications*, 6(11):12–15.

Crovella, M. and Bestavros, A. (1997). Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846.

Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413.

Gharegozi, A. (2010). Experimental evaluation of red queue management mechanism. In *2010 International Conference on Intelligent Network and Computing (ICINC 2010)*, pages 205–209.

Gong, W.-B., Liu, Y., Misra, V., and Towsley, D. (2005). Self-similarity and long range dependence on the internet: a second look at the evidence, origins and implications. *Computer Networks*, 48(3):377–399.

Hamdi, M. M., Rashid, S. A., Ismail, M., Altahrawi, M. A., Mansor, M. F., and AbuFoul, M. K. (2018). Performance evaluation of active queue management algorithms in large network. In *2018 IEEE 4th International Symposium on Telecommunication Technologies (ISTT)*, pages 1–6, Selangor, Malásia.

Jeong, H.-D. J., Ahn, W., Kim, H., and Lee, J.-S. R. (2017). Anomalous Traffic Detection and Self-Similarity Analysis in the Environment of ATMSim. *Cryptography*, 1(3):24.

Kaur, G., Saxena, V., and Gupta, J. (2020). Detection of TCP targeted high bandwidth attacks using self-similarity. *Journal of King Saud University - Computer and Information Sciences*, 32(1):35–49.

Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V. (1994). On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on networking*, 2(1):1–15.

Lokshina, I., Zhong, H., and Lanting, C. J. M. (2020). Self-similar Teletraffic in a Smart World. In Kryvinska, N. and Greguš, M., editors, *Data-Centric Business and Applications*, volume 30, pages 137–160. Springer International Publishing, Cham.

Lozhkovskyi, A. (2019). Calculation the service waiting probability with self-similar network traffic. *Journal of Engineering Science*, 26(2):36–40.

Lysenko, S., Bobrovnikova, K., Matiukh, S., Hurman, I., and Savenko, O. (2020). Detection of the botnets' low-rate DDoS attacks based on self-similarity. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(4):3651.

Mohammed, H., Attiya, G., and El-Dolil, S. (2017). Active queue management for congestion control: Performance evaluation, new approach, and comparative study. *International Journal of Computing and Network Technology*, 5(2):37–49.

NS-3 (2020a). ns3::packetsink class reference. https://www.nsnam.org/doxygen/classns3_1_1_packet_sink.html. 22 de Setembro, 2020.

NS-3 (2020b). ns3::queuedisc::stats struct reference. https://www.nsnam.org/doxygen/structns3_1_1_queue_disc_1_1_stats.html. 22 de Setembro, 2020.

Patel, N. and Patel, R. (2019). Performance analysis of network with different queuing mechanisms in tcp/ftp and udp/ftp scenario. In *Smart Systems and IoT: Innovations in Computing*, pages 13–21. Springer.

Patel, S. (2014). Performance analysis of red for stabilized queue. In *2014 Seventh International Conference on Contemporary Computing (IC3)*, pages 306–311, Noida, Índia.

Patil, G., McClean, S., and Raina, G. (2011). Drop tail and red queue management with small buffers: stability and hopf bifurcation. *ICTACT Journal on Communication Technology*, 2(2):339–344.

Paxson, V. and Floyd, S. (1995). Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244.

Rastogi, S. and Zaheer, H. (2016). Comparative analysis of queuing mechanisms: Drop-tail, red and nlred. *Social Network Analysis and Mining*, 6(70).

Santhi, V. and Natarajan, A. M. (2011). Active queue management algorithm for tcp networks congestion control. *European Journal of Scientific Research*, 54(2):245–257.

Sikdar, B., Chandrayana, K., Vastola, K., and Kalyanaraman, S. (2002a). Queue management algorithms and network traffic self-similarity. In *Workshop on High Performance Switching and Routing, Merging Optical and IP Technologie*, pages 319–323. IEEE.

Sikdar, B., Vastola, K. S., and Kalyanaraman, S. (2002b). On reducing the degree of self-similarity in network traffic. In *Proc. IEEE International Conference on Network Protocols*, pages 171–180.

Tsybakov, B. and Georganas, N. D. (1998). Self-similar traffic and upper bounds to buffer-overflow probability in an ATM queue. *Performance Evaluation*, 32(1):57–80.

Yu, S. J., Koh, P., Kwon, H., Kim, D. S., and Kim, H. K. (2016). Hurst parameter based anomaly detection for intrusion detection system. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 234–240, Nadi, Fiji.