UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

TÉCNOLOGO EM REDES DE COMPUTADORES

JORGE MAGNO LOPES MORAES

AVALIAÇÃO DE DIFERENTES CONFIGURAÇÕES DO ALGORITMO RED PARA A DIMINUIÇÃO DA AUTO-SIMILARIDADE DO TRÁFEGO DE REDE

QUIXADÁ

2018

JORGE MAGNO LOPES MORAES

AVALIAÇÃO DE DIFERENTES CONFIGURAÇÕES DO ALGORITMO RED PARA A DIMINUIÇÃO DA AUTO-SIMILARIDADE DO TRÁFEGO DE REDE

Monografia apresentada no curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de tecnólogo em Redes de Computadores. Área de concentração: Computação.

Orientador: Prof. Dr. Arthur de Castro Callado.

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M821a Moraes, Jorge Magno Lopes.
Avaliação de diferentes configurações do algoritmo RED para a diminuição da auto-similaridade do tráfego de rede / Jorge Magno Lopes Moraes. – 2018.
46 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2018.
Orientação: Prof. Dr. Arthur de Castro Callado.

1. Autossimilaridade. 2. Detecção precoce aleatória. 3. Queda da cauda. I. Título.

CDD 004.6

JORGE MAGNO LOPES MORAES

AVALIAÇÃO DE DIFERENTES CONFIGURAÇÕES DO ALGORITMO RED PARA A DIMINUIÇÃO DA AUTO-SIMILARIDADE DO TRÁFEGO DE REDE

Monografia apresentada no curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de tecnólogo em Redes de Computadores. Área de concentração: Computação.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Arthur de Castro Callado
Universidade Federal do Ceará (UFC)

_____
Prof. Me. Marcos Dantas Ortiz
Universidade Federal do Ceará (UFC)

_____
Prof. Me. Michel Sales Bonfim
Universidade Federal do Ceará (UFC)

Aos meus pais, Maria Luzilene Bezerra e Raimundo Jorge Lima Moraes.

AGRADECIMENTOS

Aos meus pais, Maria Luzilene e Raimundo Jorge Lima Moraes, que sempre estiveram ao meu lado me dando força, apoio incondicional e não me deixando desistir.

Ao Prof. Dr. Arthur de Castro Callado, pela excelente orientação e, principalmente, acreditar em mim mesmo quando eu não acreditava.

Aos professores participantes da banca examinadora Marcos Dantas Ortiz e Michel Sales Bonfim pelo tempo, pelas valiosas colaborações e sugestões.

A todos os professores que contribuíram para a minha formação acadêmica.

Aos colegas que estiveram comigo neste caminho me ajudando em trabalhos e pesquisas.

“Impossível é uma palavra muito grande que gente pequena usa para tentar nos oprimir.”

(Pregador Luo)

RESUMO

Com o conhecimento de que o tráfego de rede apresenta a característica de auto-similaridade, muitos estudos passaram a ser realizados buscando a sua diminuição, uma vez que a auto-similaridade no tráfego causa efeitos negativos, como aumento na taxa de buffer, atraso nas filas e congestionamento da rede. Um dos fatores que é mais abordado pelos pesquisadores, são as políticas de gerenciamento de filas, entre elas o RED e o droptail, devido à facilidade em configurar os algoritmos nos roteadores. Neste trabalho, com o auxílio das ferramentas NS3 e R, desenvolvemos um padrão para ajudar na configuração do RED e mostramos o impacto de diferentes configurações do RED na rede, sendo possível observar que algumas configurações são melhores, dependendo da carga de tráfego da rede, tendo a auto-similaridade e a taxa de perda de pacotes inferiores as demais. Além disso, realizamos comparações do RED com o droptail mostrando que o desempenho de ambos é bastante parecido.

Palavras-chave: Auto-similaridade. RED. Droptail.

ABSTRACT

With the knowledge that the network traffic presents the characteristic of self-similarity, many studies have been carried out seeking its decrease, since the self-similarity in the traffic causes negative effects, like increase in the buffer rate, delay in the queues and network congestion. One of the factors most addressed by researchers is queue management policies, including RED and droptail, due to the ease of configuring the algorithms in the routers. In this work, with the help of the NS3 and R tools, we developed a standard to help in the configuration of RED and show the impact of different configurations of the RED in the network, being possible to observe that some configurations are better, depending on the load of traffic of the network, having the self-similarity and packet loss rate lower than the others. In addition, we perform RED comparisons with the droptail showing that the performance of both is quite similar.

Keywords: Self-similarity. RED. Droptail.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No início as redes de telecomunicações eram projetadas para suportar um único tipo de tráfego, o de voz, utilizando a comutação de circuitos. No entanto, com o avanço constante da tecnologia, as redes começaram a sofrer mudanças. Com isso, elas passaram a ser projetadas para suportarem diferentes tipos de tráfego (voz, dados, vídeo, etc.) fazendo a comutação de pacotes crescerem de importância, pois proporciona uma maior flexibilidade aos serviços de comunicações (RUAS, 2010).

Antes do surgimento da internet, os modelos predominantes na engenharia de tráfego eram baseados em distribuições de *Poisson*. Esses modelos definem que as chegadas de clientes obedecem a uma distribuição de *Poisson* e o tempo de serviço possui distribuição exponencial, esses modelos foram denominados *Markovianos* (RUAS, 2010). Com os estudos realizados por Leland et al (1993) foi visto que o tráfego de pacotes difere bastante dos modelos de natureza de *Poisson* propostos, no qual o tráfego se torna mais suave conforme o número de fontes aumenta. Na verdade, com o aumento de fontes de tráfego ativas, o grau de auto-similaridade (parâmetro de Hurst) se intensifica. Com a identificação do tráfego de natureza auto-similar, percebeu-se que o mesmo pode ocasionar alguns efeitos negativos, como taxas de *buffer*, grandes atrasos e congestionamento da rede.

Os fatores que causam a auto-similaridade na rede são diversos, variando desde o comportamento de aplicativos na origem e destino, escolha das fontes de tráfego ou fatores humanos (SIKDAR; VASTOLA; KALYANARAMAN, 2002). Muitos trabalhos que abordam o tema, tratam principalmente um fator da auto-similaridade, o gerenciamento de filas. Isso ocorre devido à facilidade de modificar os algoritmos de gerenciamento de fila (*First In, First Out* (*FIFO*), *Random Early Detection* (*RED*), etc.) nos roteadores, tornando-se um caminho mais prático. Um dos trabalhos foi o de Sikdar et al (2002b) que propôs duas técnicas para reduzir a auto-similaridade, no qual era feita uma modificação no algoritmo *RED*.

Em relação aos algoritmos de gerenciamento de fila podemos destacar o *RED* que é um algoritmo que descarta pacotes antes que a fila fique cheia e que teve seu funcionamento detalhado em Floyd e Jacobson (1993), no qual afirmam que o *RED* evita a sincronização global do *TCP* (quando as fontes TCP diminuem e aumentam a taxa de transmissão simultaneamente) algo que pode ocorrer em *FIFO*, quando configurado corretamente. O *RED* apresenta quatro configurações que podem ser modificadas:

a) A constante de tempo de filtro passa baixa (w_q);
b) O limiar mínimo do número de pacotes (min_{th});

c) O limiar máximo do número de pacotes (max_{th});

d) A probabilidade máxima de descarte (max_p).

Floyd e Jacobson (1993) recomendam alguns valores, definidos após uma série de simulações, para essas configurações:

a) Na constante de tempo de filtro (w_q) o valor recomendado é 0,002;

b) Na probabilidade máxima de descarte (max_p) o valor recomendado é de 50%;

c) Na configuração do limiar mínimo (min_{th}) e do limiar máximo (max_{th}), recomenda-se que o limiar máximo seja pelo menos duas vezes maior que o limiar mínimo.

É perceptível que em Floyd e Jacobson (1993) foi definido um valor padrão para w_q e max_p, não ocorrendo o mesmo com os valores de min_{th} e max_{th}, pois a configuração de ambos depende da rede.

Diante da preocupação dos efeitos negativos que a auto-similaridade da rede causa, o presente trabalho tem o objetivo de utilizar o algoritmo de gerenciamento de filas *RED* e avaliar o impacto de diferentes configurações de min_{th} e max_{th}, além de criar um padrão para auxiliar na configuração do *RED*.

1.1 Objetivos

O objetivo principal deste trabalho é avaliar se diferentes configurações de min_{th} e max_{th} do algoritmo *Random Early Detection* (*RED*) diminuem a auto-similaridade no tráfego de uma rede com diferentes cargas de tráfego.

A fim de atingir o objetivo geral foram definidos os seguintes objetivos específicos:

a) Criar um padrão de configuração min_{th}-max_{th} do *RED*;

b) Encontrar a melhor (ou melhores) configuração min_{th}-max_{th};

c) Comparar a melhor (ou melhores) configuração do *RED* com o *droptail*;

2 TRABALHOS RELACIONADOS

Como foi citado anteriormente, existem alguns trabalhos que procuram reduzir a auto-similaridade, neste capítulo vamos falar dos principais trabalhos e o que eles têm em comum com este, apontando as principais semelhanças e diferenças.

Em Sikdar et al (2002b) foram apresentadas duas técnicas para reduzir a auto-similaridade do tráfego de rede. A primeira técnica busca reduzir a incidência de tempos limite e reversão exponencial em fluxos *TCP* (*Transmission Control Protocol*), isso é feito analisando e propondo novas políticas de gerenciamento de *buffer* para reduzir as perdas correlacionadas. A segunda técnica é eliminar a explosão inerente e as transmissões de pacotes *back-to-back* nos fluxos *TCP*, foi analisado o impacto da redução dos efeitos da compressão *ACK* (*Acknowledgment*) e da explosão das fontes *TCP* através do *TCP pacing*, que visa espalhar uniformemente a transmissão de uma janela de pacotes através de um *RTT* (*Round Trip Time*), na auto-similaridade do tráfego. Além disso, foi feita uma modificação no algoritmo *RED* com o objetivo de reduzir os tempos limites e reversões exponenciais no fluxo *TCP*, ocasionando uma diminuição da auto-similaridade do tráfego. Com relação ao presente trabalho, Sikdar et al (2002b) se assemelha no fato de ter foco em diminuir a auto-similaridade, além de trabalhar com o algoritmo *RED* e utilizar a topologia *dumbbell*. Porém o trabalho se difere em utilizar técnicas específicas para o tráfego *TCP*, em realizar uma modificação no algoritmo *RED* e cenários com poucas fontes de tráfego, enquanto o trabalho aqui desenvolvido busca diminuir a auto-similaridade de formal mais global, encontrando uma melhor configuração do algoritmo *RED* nos roteadores e não modificá-lo, além de utilizar cenários com mais fontes de tráfego.

Em Suresh e Göl (2005) foi proposto um algoritmo *RED* modificado para gerenciamento de congestionamento de tráfego em redes *IP* (*Internet Protocol*) com entrada auto-similar. A pesquisa utilizou redes *IP*, uma vez que o protocolo *IP* é o método mais utilizado para o transporte de dados dentro e entre a comunicação de redes. Além disso, foi feito um breve estudo sobre o algoritmo *RED* e o parâmetro de Hurst. A modificação do algoritmo tomou em consideração os valores de probabilidade correspondentes aos comprimentos médios da fila para calcular a probabilidade *de marking/dropping*, uma vez que o tráfego *IP* é tomado como auto-similar. Contudo Suresh e Göl (2005) se preocupam apenas com o tráfego *IP* e em criar um modelo *RED* modificado, sendo essas as principais diferenças

com relação ao presente trabalho, que irá trabalhar com fluxos *TCP* e sem modificar o algoritmo *RED*.

Em Amin et al (2013) foi elaborado um algoritmo de gerenciamento de filas ativo que leva em consideração a auto-similaridade do tráfego, utilizando uma técnica baseada no método *wavelet* para estimar o parâmetro Hurst. Além disso, o algoritmo elaborado foi comparado com o *RED*. Para o atual trabalho, Amin et al (2013) se torna importante para o melhor entendimento do parâmetro de Husrt, além de ajudar na configuração do algoritmo *RED*, porém se difere no método para a estimativa de Hurst, pois aqui será usado a análise R/S.

Em Tan e Huang (2009) estuda-se a configuração dos parâmetros *RED* com entrada de tráfego dependente de longo alcance (*LRD – Long Range Dependence*), onde a característica *LRD* do tráfego de rede é levada em consideração na configuração dos parâmetros de *RED*, ou seja, o peso do filtro, o limiar mínimo e máximo da fila e a taxa de *marking/dropping* de pacotes. A probabilidade de *marking/dropping* de pacotes é calculada com base no resultado analítico do modelo *Fractional Brownian Motion* (*FBM*). O limiar de fila mínimo e máximo é derivado com base no período de ocupação máximo, que faz com que o comprimento da fila atinja seu valor máximo em um período de explosão, com base em um processo de *FBM*. Além disso, as simulações são feitas usando uma topologia *dumbbell*. Tan e Huang (2009) é muito importante no estudo para a configuração dos parâmetros *RED*, principalmente no limiar mínimo e máximo da fila que serão as configurações a serem usadas neste trabalho, tendo como diferença o fato de não se preocupar com a diminuição da auto-similaridade, mas em otimizar o algoritmo *RED*.

Em Sikdar et al (2002a) é investigado o impacto de vários algoritmos de gerenciamento de fila na auto-similaridade do tráfego de rede e o impacto das políticas de gerenciamento de filas ativas (*RED*) e passivas (*Droptail*) usadas nos roteadores do tráfego *TCP* com auto-similaridade. Também é proposto uma modificação do algoritmo *RED*, com o objetivo de reduzir os tempos limite e as reversões exponenciais nos fluxos *TCP*, e mostrar que isso pode levar a reduções significativas na auto-similaridade do tráfego em uma ampla gama de condições de rede. Diante disso, Sikdar et al (2002a) se difere em estudar o impacto das políticas de gerenciamento de filas (*Droptail* e *RED*) e propor uma modificação no algoritmo *RED*, além de usar poucas fontes na geração de tráfego na topologia *dumbbell*. Enquanto este trabalho realiza uma comparação entre *Droptail* e *RED* mais aprofundada, pois usamos um número maior de configurações *RED* e de fontes de tráfego.

Na Tabela 1 podemos observar a comparação dos trabalhos relacionados com o trabalho aqui desenvolvido para o melhor entendimento das diferenças entre os trabalhos.

Tabela 1 - Comparação dos trabalhos relacionados e o presente trabalho

	RED	**Droptail**	**Auto-similaridade**	**TCP**	**IP**
Sikdar et al (2002b)	X	X	X	X	
Suresh e Göl (2005)	X		X		X
Amin et al (2013)	X		X		
Tan e Huang (2009)	X				
Sikdar et al (2002a)	X	X	X	X	
Presente Trabalho	X	X	X	X	

Fonte: Elaborada pelo autor

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão definidos os principais conceitos para o melhor entendimento deste trabalho. Primeiramente será explicado o que é a auto-similaridade, abordando como caracterizar um tráfego auto-similar e os conceitos matemáticos e estatísticos da auto-similaridade. Depois será explicado de uma maneira mais clara do que se trata o algoritmo *Random Early Detection* (*RED*) e logo depois o funcionamento do algoritmo *droptail*. Por fim, serão abordadas as ferramentas que serão utilizadas para o desenvolvimento do trabalho, o *Network Simulator 3* (*NS3*) e *R*.

3.1 Auto-similaridade

Nesta subseção, para a definição de tráfego auto-similar e os conceitos matemáticos e estatísticos da auto-similaridade será tomado como base o artigo de Leland et al (1993).

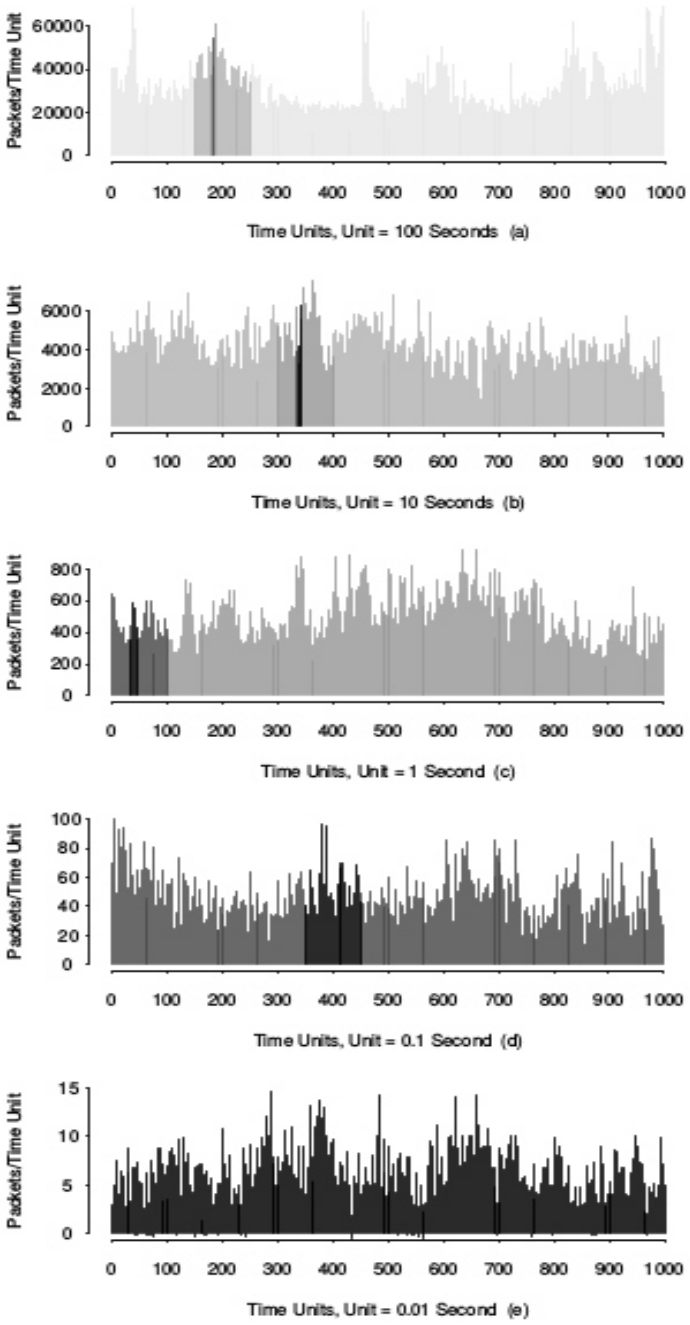
3.1.1 Tráfego auto-similar

Desde o início dos anos 90, uma série de pesquisadores publicaram artigos sobre a modelagem do tráfego auto–similar com uma grande variedade de situações reais em redes de computadores. O primeiro artigo a desenvolver um estudo de tráfego auto-similar foi Leland et al (1993), sendo expandido posteriormente em Leland et al (1994). Este artigo demostrou que a análise das filas, usando a distribuição de *Poisson*, não é adequada para modelar todos os tráfegos presentes em redes de computadores. Usando certa quantidade de dados e uma análise estatística, o artigo demonstrou que para o tráfego Ethernet, um novo modelamento e análise de aproximação são necessários. O artigo original relata resultados de medições de tráfego Ethernet transmitido entre 1989 e 1992. Os dados consistem em 4 grupos de medições de tráfego, cada um composto de 20 e 40 horas consecutivas de tráfego Ethernet. Os dados foram coletados em redes locais em *BellCore* (*Network BellCore Morris Research and Engineering Center*).

A Figura 1 mostra gráficos de número de pacotes por unidades de tempo, de um conjunto de dados de 1989, o qual consiste de 27 horas de monitoração contínua de tráfego Ethernet. O primeiro gráfico mostra 27 horas contínuas, usando a unidade de tempo de 100 segundos, para um gráfico com 1000 pontos de dados. Cada gráfico seguinte é obtido vindo

do anterior, incrementando pela resolução de tempo por um fator de 10. Consequentemente o segundo gráfico cobre um período de aproximadamente 2.7 horas, o terceiro de 0,27 horas e assim por diante.

Figura 1 - Tráfego Ethernet em 5 escalas diferentes



Fonte: Leland et al (1993).

Algumas observações interessantes foram feitas sobre estes dados. Todos os gráficos envolvem 1000 pontos de dados, isto é, quantidade de rajadas, consequentemente, o tráfego Ethernet tende a ser percebido da mesma maneira para grandes escalas (horas e minutos) e para pequenas escalas (segundos e milissegundos). Vemos que não há um comprimento fixo para as rajadas de tráfego. Em cada escala de tempo, existem rajadas que consistem em subperíodos de rajadas, estas separadas por pequenos subperíodos de rajadas. Esta característica auto–similar difere do tráfego de voz pelo telefone e dos modelos estocásticos tradicionalmente usados em análise de rede de dados.

3.1.2 Conceitos matemáticos da auto-similaridade

Para o melhor entendimento matemático da auto-similaridade é preciso saber os conceitos de processos estocásticos estacionários, função de autocorrelação e covariância estacionária. Um processo estocástico é dito ser estacionário quando as características das probabilidades do processo não variam como uma função de tempo. Um processo é estacionário, se o valor esperado é uma constante e a função de autocorrelação depende somente das diferenças de tempo (SILVA, 2001).

Em Leland et al (1993) é utilizado o seguinte exemplo. Seja $X = (X_t : t = 0, 1, 2, ...)$ um processo estocástico estacionário por possuir uma média constante $\mu = E[X_t]$, uma variância finita $\sigma^2 = E[(X_t - \mu)^2]$ e uma função de autocorrelação $r(k) = E[(X_t - \mu)(X_{t+k} - \mu)]/E[(X_t - \mu)^2]$ $(k = 0, 1, 2, ...)$ que depende somente de k. É assumido que X tem uma função de autocorrelação com forma:

$$r(k) \sim ak^{-\beta}, quando\ k \rightarrow \infty \quad (1);$$

onde $0 < \beta < 1$ (nesta função a letra a denota uma constante positiva). Para cada m = 1, 2, 3, ..., considera–se $X^{(m)} = (X_k^{(m)}: k = 1, 2, 3, ...)$ que denota uma nova série de tempo obtida pela média da série original X sobre os blocos não sobrepostos de tamanho m, isto é, para cada m = 1, 2, 3, ..., $X^{(m)}$ é dado por $X_k^{(m)} = 1/m(X_{km-m}+...+X_{km}), (k \geq 1)$. Nota-se que para cada m, a série de tempo agregado $X^{(m)}$ define um processo de covariância estacionária, onde, $r^{(m)}$ denota a função de autocorrelação correspondente.

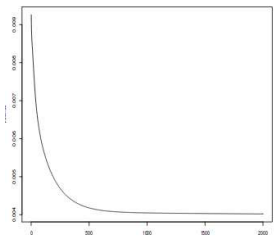
O processo X é chamado de exatamente (segunda ordem) auto–similar com parâmetro de auto–similaridade (*Hurst*) $H = 1 - \beta/2$, se o processo agregado correspondente $X^{(m)}$ tiver a mesma estrutura de correlação que X, ou seja, $r^{(m)}(k) = r(k)$ para todos m = 1, 2, ... (k = 1, 2, 3, ...). Isto é, X é auto–similar se os processos agregados $X^{(m)}$ forem vindos de X, em relação às suas propriedades estatísticas de segunda ordem. Um processo estacionário de

covariância X é chamado assintoticamente (segunda ordem) auto-similar com o parâmetro de auto-similaridade $H = 1 - \beta/2$ se $r^{(m)}(k)$ concorda assintoticamente (ou seja, para grandes m e grandes k) com o estrutura de correlação $r(k)$ de X dada pela equação (1) (LELAND, 1993).

A principal característica dos processos auto–similares (exatamente ou assintoticamente), é que seus processos agregados $X^{(m)}$ possuem estruturas de correlações que não se alteram quando $m \rightarrow \infty$. Esse comportamento é precisamente ilustrado na sequência de gráficos na Figura 1; se a série de tempo original X representa o número de pacotes Ethernet por 10 milissegundos (gráfico (e)), os gráficos de (a) a (d) representam segmentos das séries temporais agregadas $X^{(10000)}$, $X^{(1000)}$, $X^{(100)}$ e $X^{(10)}$, respectivamente. Todas os segmentos parecem "semelhantes", sugerindo uma função de autocorrelação quase idêntica para todos os processos agregados.

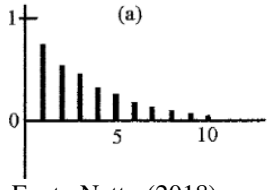
Matematicamente, a auto-similaridade manifesta-se por si mesma por meio das autocorrelações que decaem hiperbolicamente (Figura 2) e não exponencialmente rápido (Figura 3), como ocorre na análise clássica, ocorrendo em uma função de autocorrelação não somável, $\sum_k r(k) = \infty$.

Figura 2 - Decaimento hiperbólico



Fonte: Laurini (2002)

Figura 3 - Decaimento exponencial rápido



Fonte:Netto (2018)

Esta característica nos traz a percepção de uma estrutura de correlação que não se altera entre os instantes de tempo presentes na série temporal analisada. Esta característica é chamada de dependência de longa duração (*LRD*). A existência de uma estrutura de correlação

que não se altera para com os processos agregados $X^{(m)}$ quando $m \rightarrow \infty$, é o principal contraste em comparação com os modelos de tráfego de pacotes considerados na literatura clássica, onde, todos estes modelos, possuem uma propriedade que seus processos agregados $X^{(m)}$ tendem a uma estrutura de correlação que tende a zero quando m tende a infinito, ou seja, $r^{(m)}(k) \rightarrow 0$ quando $m \rightarrow \infty$ $(k = 1, 2, ...)$. Equivalentemente, eles podem ser caracterizados por uma função de autocorrelação que decai exponencialmente rápido, implicando em uma função de autocorrelação somável, $\sum_k r(k) < \infty$, ao não satisfazer à equação (1) caracteriza-se uma correlação que se altera entre os instantes de tempo presentes nas séries temporais analisadas, a característica é chamada de dependência de curta duração (*SRD – Short Range Dependence*) (SILVA, 2001).

Em resumo, para que o tráfego seja considerado auto-similar, ele precisa apresentar a característica de dependência de longa duração (*LRD*).

3.1.3 Conceitos estatísticos da auto-similaridade

Quando é estudada a estatística da auto-similaridade, pretende-se detectar as correlações existentes em séries temporais que possuem características auto-similar. A detecção destas características consiste em estimar o parâmetro de Hurst. Esta estimativa pode ser feita por meio da análise do período reescalado (Análise *R/S*), pelo Estimador Gradual (*Whittle's Estimator*) e pela análise por *Wavelet* (*Wavelet Analysis*) (SILVA, 2001). A seguir discutiremos mais sobre a Análise R/S, que será utilizada neste trabalho para estimar o parâmetro de Hurst, por ser a principal e mais utilizada para a estimativa devido a sua baixa complexidade matemática (RUAS, 2011).

3.1.3.1 Análise R/S

Em Leland et al (1993) é utilizado a análise *R/S*, devido a sua relativa robustez contra as mudanças da distribuição marginal esta característica permite investigações praticamente separadas da propriedade de auto-similaridade de um determinado registro empírico e de suas características de distribuição, que é baseada em uma aproximação gráfica heurística que tenta explorar a informação sobre a estimativa de quão a auto-similaridade se faz presente em uma dada série temporal. Se os dados forem fractais, então as correlações em diferentes escalas de tempo poderão ser vistas estando relacionadas umas com as outras. A

presença destas correlações é expressa pela estimativa do Parâmetro de Hurst, o qual segue o seguinte comportamento:

a) Quando Husrt < 0,5, não existem correlações, isto é, incrementos nos valores das séries temporais são imparciais a incrementos ou reduções no grau de correlações presentes na série temporal (*SRD – Short Range Dependence*);
b) Quando 0,5 < Hurst < 1, as correlações são persistentes, isto é, incrementos nos valores das séries temporais são mais prováveis de serem seguidos por incrementos no grau de correlações presentes na série temporal (*LRD – Long Rang Depedence*).

Para um processo estocástico *X(t)* definido em instantes discretos de tempo *{X_t, t = 0,1, 2, ...}*, o período reescalado de *X(t)* durante um intervalo de tempo *N* é definido com a razão R/S :

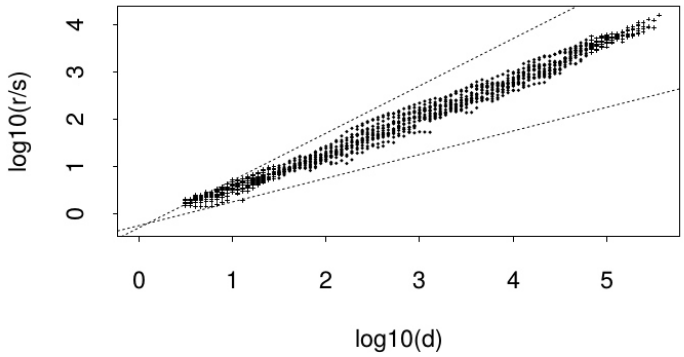
$$\frac{R}{S} = \frac{\max\limits_{1 \le j \le N} \sum_{k=1}^{j} X_k - M(N) - \min\limits_{1 \le j \le N} \sum_{k=1}^{j} X_k - M(N)}{\sqrt{\frac{1}{N} \sum_{j=1}^{N} \left(X_k - M(N)\right)^2}} \quad (2);$$

onde *M(N)* é a média da amostra durante o período de tempo *N* :

$$M(N) = \frac{1}{N} \sum_{j=1}^{N} X_j \quad (3);$$

O numerador desta razão (equação 2) é obtido somando, seguindo a indicação do índice *j*, os valores de X_k subtraídos da média (equação (3)). O denominador (equação (2)) é o desvio padrão da amostra de tráfego. A estimativa do parâmetro de Hurst é dada pela inclinação do gráfico *log* [*R/S*] por *log* [*N*] em um gráfico *log* por *log*, como pode ser observado na Figura 4.

Figura 4 - Grafico log[R/S] por log[N]



Fonte: Leland et al (1993)

3.2 Random Early Detection (RED)

O *RED* é um algoritmo de gerenciamento de filas ativo que descarta aleatoriamente os pacotes antes que uma fila fique cheia, de modo que os nós finais possam responder ao congestionamento antes que os *buffers* transbordem (SIKDAR; VASTOLA; KALYANARAMAN, 2002). O interesse desse trabalho ao usar o *RED* é saber o impacto do algoritmo sobre à auto-similaridade do tráfego de rede.

O algoritmo *RED* descarta pacotes probabilisticamente mesmo antes que a fila esteja cheia com base em uma média ponderada (*avg*) do comprimento da fila. O valor da média ponderada é comparado a dois limiares, o limiar do número mínimo de pacotes (min_{th}) e o limiar do número máximo de pacotes (max_{th}) da fila. Quando um pacote chega à fila ocorrem 3 casos para o descarte de pacotes:

a) Quando o tamanho médio da fila é menor do que o limiar do número mínimo de pacotes, nenhum pacote será descartado;
b) Quando o tamanho médio da fila é maior do que o limiar do número máximo de pacotes, todos os pacotes que chegam no momento que o tamanho médio foi determinado serão descartados ou marcados;
c) Quando o tamanho médio da fila está entre os dois limiares, a probabilidade (p_a) com que um pacote que acaba de chegar na fila venha a ser marcado ou descartado é uma função do tamanho médio da fila.

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 (p_a) com que os pacotes serão marcados ou descartados dado um determinado nível de congestionamento. O cálculo do tamanho médio da fila é determinado usando um

filtro passa baixa, que nada mais é do que a função *Exponential Weighted Moving Average* (*EWMA*) (SANTOS, 2005):

$$avg_{(t)} = \left(1 - w_q\right)avg + w_q q \quad (4);$$

onde *avg* é o tamanho médio da fila, w_q é a constante de tempo de filtro passa baixa e q é o valor atual da fila (em pacotes).

Caso o valor para w_q seja muito baixo, o cálculo do valor do tamanho médio da fila irá refletir o valor atual da fila (q) com um retardo muito grande. Ou seja, o tamanho médio da fila não refletirá as mudanças repentinas no tamanho atual da fila. Durante o período em que a fila está vazia o algoritmo *RED* estima o número (m) de pequenos pacotes que poderia ser transmitido pelo roteador durante esse período. Após o período ocioso o roteador calcula o tamanho médio da fila como se (m) pacotes tivessem chegado a essa fila durante o período ocioso (SANTOS, 2005).

Como o tamanho médio da fila (*avg*) varia entre min_{th} e max_{th}, a probabilidade de descarte ou de marcação de pacotes varia linearmente entre 0 ao maior valor dessa probabilidade (max_p). Portanto, a probabilidade de descarte ou de marcação de pacotes quando o tamanho médio da fila (*avg*) está entre os limiares min_{th} e max_{th} é dada por:

$$P_a = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})} \quad (5);$$

Para evitar a sincronização global da rede é necessário que a diferença entre max_{th} e min_{th} seja grande o suficiente para que o tamanho médio da fila (*avg*) não oscile frequentemente até max_{th}. Caso essa diferença seja muito pequena o tamanho médio da fila irá variar até max_{th} em uma frequência maior, logo com maior frequência todos os pacotes serão descartados com probabilidade máxima de descarte (max_p) (SANTOS, 2005). Floyd e Jacobson (1993) afirmam que uma regra prática para definir a configuração min_{th}-max_{th} é colocar o max_{th} pelo menos duas vezes maior que min_{th}. Além disso, eles recomendam que os valores de min_{th} e max_{th} sejam suficientemente altos para garantir o melhor uso da rede.

3.3 Droptail

Um método tradicional de gerenciamento de filas é definido pelo algoritmo *droptail*, pois seu funcionamento consiste em descartar as requisições quando ultrapassar o valor máximo da fila (FIGUEIREDO, 2011). Em Braden et al (1998) é mencionado que o *droptail* apresenta dois pontos negativos. O primeiro ponto é permitir que uma simples

conexão monopolize o espaço da fila, enquanto outras conexões têm seus pacotes descartados pela à falta de espaço. O segundo ponto é que o *droptail* permite que a fila fique cheia por longos períodos causando a redução da vazão.

Nota-se que o funcionamento do *droptail* é simples, afinal o descarte ocorre sempre quando o tamanho da fila alcança 100% de ocupação, mesmo assim entendê-lo bem é importante para o prosseguimento do trabalho, pois posteriormente ele é comparado ao algoritmo *RED*. Além disso, é importante salientar que diferente do *RED*, o droptail necessita apenas de uma configuração, que no caso é o tamanho total da fila.

3.4 Ferramentas

Nesta seção, apresentamos as ferramentas utilizadas para o desenvolvimento do trabalho explicando suas funcionalidades e importância para o uso no trabalho.

3.4.1 Network Simulator 3 (NS3)

O *NS3* foi desenvolvido para fornecer uma plataforma de simulação de rede aberta e extensível para pesquisa e educação em rede. Em resumo, o *NS3* fornece modelos de como as redes de dados de pacote funcionam e executam, e fornece um mecanismo de simulação para os usuários conduzirem experimentos de simulação. Algumas das razões para usar o *NS3* incluem realizar estudos que são mais difíceis ou impossíveis de realizar com sistemas reais, estudar o comportamento do sistema em um ambiente altamente controlado e reproduzível e aprender sobre como as redes funcionam (NS3d, 2018).

A escolha do *NS3* se deve, principalmente, por apresentar uma documentação muito vasta, que auxilia e facilita na criação de tráfego de rede e topologias. Além disso, apresenta boas opções para a obtenção dos resultados das simulações, como, por exemplo, o sistema de rastreamento ASCII e PCAP, e os pacotes do *RED* e *droptail* disponíveis sendo simples sua implementação. Por fim, trata-se uma plataforma bastante completa e de código aberto, que conta com uma comunidade de desenvolvedores participativos, os quais ajudam a melhorar e criar tutoriais de uso do *NS3*.

3.4.2 Software R

O *R* é uma linguagem orientada a objetos criada em 1996 por Ross Ihaka e Robert Gentleman que aliada a um ambiente integrado permite a manipulação de dados, realização de

cálculos e geração de gráficos. Semelhante à linguagem S desenvolvida pela *AT&T's Bell Laboratories* e que já é utilizada para análise de dados, mas com a vantagem de ser de livre distribuição. Vale ressaltar que o R não é um programa estatístico, mas que devido às suas rotinas permite a manipulação, interpretação e avaliação de procedimentos estatísticos aplicados a dados (SOUSA; PETERNELLI; MELLO, 2018).

O *R* torna-se, portanto, importante na análise de manipulação de dados que serão extraídos das simulações por apresentar uma facilidade na análise de séries temporais e, consequentemente, nos cálculos para estimar o parâmetro de Hurst, contendo vários pacotes em seu repositório que auxiliar na estimativa. Além de possuir uma grande comunidade de usuários nas mais diversas áreas de conhecimento.

4 METODOLOGIA

Neste capítulo falaremos rapidamente da metodologia usada para alcançar os objetivos desejados, além de falar de alguns detalhes de cada etapa. A execução do trabalho seguiu as seguintes etapas:

a) A primeira etapa deste trabalho consistiu em fazer um levantamento bibliográfico referente à auto-similaridade e ao *RED*, em busca do estado da arte. Este levantamento foi feito sendo identificados os principais artigos relacionados ao assunto;

b) Na segunda etapa foi realizado um estudo mais aprofundado das ferramentas (*NS3* e *R*) a serem utilizadas a fim saber a melhor maneira de utilizá-las em busca da melhor desempenho nas simulações e resultados. Com isso, o *NS3* foi usado na criação dos cenários e execução das simulações enquanto o *R* foi utilizado para estimar o parâmetro de Hurst;

c) Na terceira etapa foram criados os cenários usados neste trabalho com o auxílio do *NS3*. Foram desenvolvidos três cenários com o objetivo de simularem diferentes tipos de carga de tráfego (leve, moderada e pesada), todos na topologia *dumbbell*;

d) Na quarta etapa foi desenvolvido um padrão de configuração min_{th}-max_{th} do *RED*, esse padrão foi feito de acordo com recomendações de Floyd e Jacobson (1993);

e) Na quinta etapa foram realizadas as simulações no *NS3* com cada configuração min_{th}-max_{th} sendo executada em cada um dos três cenários, além de uma execução usando *droptail* nos cenários. Nesta etapa, para cada simulação executada foi criado um arquivo de texto através do sistema de rastreamento ASCII, além de utilizar uma função do *NS3* (*ns3::QueueDisc::Stats*) que permitiu saber a taxa de descarte;

f) Na sexta etapa foi feita a captura das amostras de tráfego de cada simulação. Para isso foi criado um *script* com a função de capturar o fluxo de dados do gargalo da topologia;

g) Na sétima etapa foi estimado o parâmetro de Hurst para cada simulação. Nesta etapa foi utilizado o *R*, que possui um pacote (*fractal*) que calcula o parâmetro de Hurst usando a análise *R/S*;

h) Na última etapa foi feita a análise dos resultados comparando as diferentes configurações min_{th}-max_{th} em cada cenário. Além de realizar, posteriormente, comparação entre as melhores configurações min_{th}-max_{th} e *droptail*.

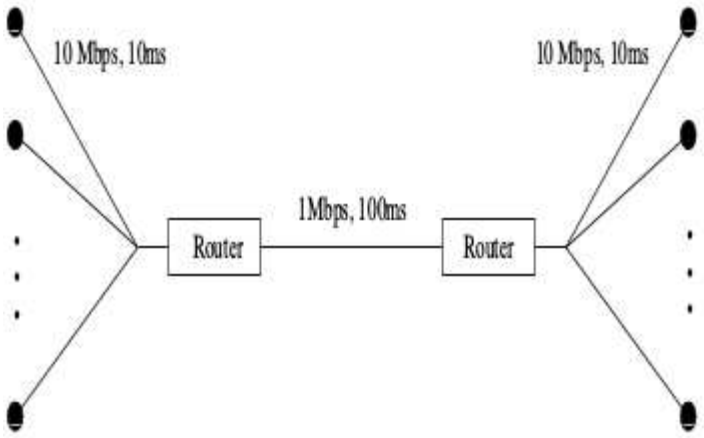
5 DESENVOLVIMENTO

Neste capítulo será apresentado o desenvolvimento do trabalho, ou seja, como foi posto em prática a sua execução. Para isso, o capítulo foi divido em duas partes, que são simulações (fase de preparação e execução das simulações) e resultados (análise e comparação dos resultados).

5.1 Simulações

A execução das simulações foi feita utilizando a topologia *dumbbell* (ou haltere). Essa topologia apresenta dois roteadores conectados por um *link*, chamados de roteadores do gargalo, e vários outros roteadores, chamados de folhas, conectados a um dos roteadores do gargalo (Figura 5). Os *links* da topologia foram configurados para gerarem congestionamento nas filas dos roteadores do gargalo, onde os *links* das folhas foram configurados com velocidade de *10Mbps* e *delay* de *10ms*, enquanto o link do gargalo foi configurado com velocidade de *1Mbps* e *delay* de *1ms*.

Figura 5 - Topologia dumbbell usada nas simulações



Fonte: Sikdar et al (2002b)

A partir de então foram criados três cenários com diferentes números de fontes *TCP* comuns, ou seja, iniciar uma transmissão *TCP* e não parar de enviar até que tudo seja enviado. Estes cenários tinham o objetivo de simular três diferentes tipos de carga de tráfego

(leve, moderada e pesada, onde cada lado da topologia transmitia dados para o outro lado (lado esquerdo enviando dados para o lado direito e vice-versa). O primeiro cenário utilizou 60 fontes *TCP* (30 fontes em cada lado da topologia *dumbbell*), o segundo utilizou 300 fontes *TCP* (150 fontes em cada lado da topologia *dumbbell*) e o último cenário utilizou 480 fontes *TCP* (240 fontes em cada lado da topologia *dumbbell*). A escolha para o número de fontes em cada cenário foi feita empiricamente a partir de testes preliminares.

Para as simulações, a fila escolhida para ser analisada foi a do roteador de gargalo do lado direito e o tamanho total das filas *droptail* e *RED* foram definidos em 100 pacotes (no *NS3* o tamanho de fila foi definido em pacotes), esse tamanho foi escolhido para facilitar no entendimento da porcentagem de ocupação das filas, ou seja, 100 pacotes na fila correspondem a 100% de ocupação e também por ter sido o mesmo tamanho de fila usado em Sikdar et al (2002b). O restante das configurações do *RED* foi definido seguindo as recomendações de Floyd e Jacobson (1993), sendo assim, os valores de w_q (0,002) e max_p (50%) foram mantidos. Para a configuração dos valores de min_{th} e max_{th} foi criado um padrão de configuração min_{th}-max_{th} que seguiu outras três recomendações de Floyd e Jacobson (1993), os valores de min_{th} e max_{th} tem que ser suficientemente altos, max_{th} tem que ser no mínimo duas vezes maior que min_{th} e a diferença entre max_{th} e min_{th} tem que ser grande. Com isso foi definido o seguinte padrão:

a) $10\% \leq min_{th} \leq 45\%$, onde o valor de min_{th} varia entre 10% e 45% do tamanho total da fila;
b) $60\% \leq max_{th} \leq 90\%$, onde o valor de max_{th} varia entre 60% e 90% do tamanho total da fila;
c) $max_{th} - min_{th} \geq 40\%$, onde a diferença entre max_{th} e min_{th} tem que ser no mínimo 40%.

Com a implantação desse padrão ficaram definidas as seguintes configurações min_{th}-max_{th} do *RED* para as simulações (os valores abaixo também podem ser vistos como porcentagem uma vez que o tamanho total da fila é de 100 pacotes):

a) 40-80, 41-82, 43-86, 44-88, 45-90 (max_{th} duas vezes maior que min_{th});
b) 20-60, 21-63, 22-66, 23-69, 24-72, 25-75, 26-78, 27-81, 28-84, 27-87, 30-90 (max_{th} três vezes maior que min_{th});
c) 15-60, 16-64, 17-68, 18-72, 19-76, 20-80, 21-84, 22-88 (max_{th} quatro vezes maior que min_{th});
d) 12-60, 13-65, 14-70, 15-75, 16-80, 17-85, 18-90 (max_{th} cinco vezes maior que min_{th});

e) 10-60, 11-66, 12-72, 13-78, 14-84, 15-90 (max_{th} seis vezes maior que min_{th});

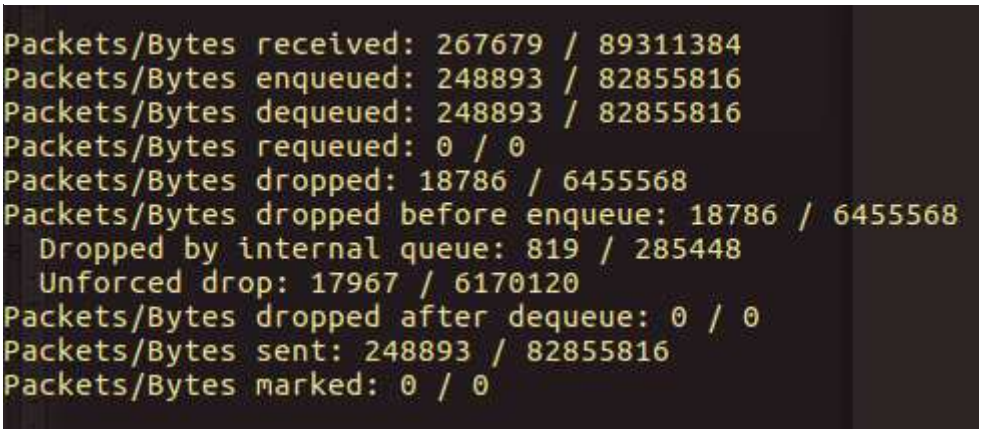
f) 10-70, 11-77, 12-84 (max_{th} sete vezes maior que min_{th});

g) 10-80, 11-88 (max_{th} oito vezes maior que min_{th});

h) 10-90 (max_{th} nove vezes maior que min_{th}).

O tempo de execução de cada simulação foi de 320 segundos, sendo que a coleta de dados foi feita após 10 segundos até 310 segundos a fim de evitar a coleta de dados transitórios. A coleta de dados foi feita a cada 0,1 segundos, com isso foram obtidas 3000 amostras. A geração do tráfego em cada simulação foi feita utilizando o *BulkSendApplication*, que é um gerador de tráfego que simplesmente envia dados até que o mesmo seja interrompido (NS3a, 2018), além disso, o tamanho dos pacotes na simulação foi mantido o padrão do *NS3*. Para essa coleta de dados foi utilizado o sistema de rastreamento ASCII do *NS3*, no qual cada simulação gerou um arquivo *.tr* e depois foi utilizado um *script* responsável por contar o número de pacotes recebidos pela fila do gargalo do roteador escolhido a cada 0,1 segundos, gerando um arquivo *.txt*. O arquivo *.txt*, então, foi usado para estimar o parâmetro de Hurst através da função *RoverS* do pacote *fractal* do *R*, o qual usa o método de análise *R/S* para estimar Hurst (R, 2018). Juntamente com a geração do arquivo *.tr* era feito o cálculo do *throughput* médio, no qual foi utilizado o *flow monitor* (que é um módulo que fornece um sistema flexível para o desempenho dos protocolos de rede (NS3b, 2018)), e a contagem da perda de pacotes na simulação e do número total de pacotes recebidos na simulação, com o objetivo de se calcular a taxa de descarte da simulação. Para isso, foi usado o *ns3::QueueDisc::Stats*, que é uma estrutura que mantém as estatísticas da fila (NS3c, 2018). Na Figura 6 podemos observar a saída dessa estrutura, onde na primeira linha temos o número de pacotes recebidos na fila (no exemplo são 267679) e na sexta linha temos o número de pacotes descartados antes do enfileiramento (no exemplo são 18786). Para o cálculo da taxa de descarte, então, dividimos o número de pacotes descartados antes do enfileiramento pelo o número de pacotes recebidos na fila e depois multiplicando o resultado por 100. Seguindo o exemplo da Figura 6 o cálculo seria *(18786/267679) x 100*, onde o resultado seria em torno de 7%. Lembrando que o resultado da taxa de descarte refere-se ao intervalo de tempo da coleta de dados da simulação, ou seja, de 10 segundos até 310 segundos.

Figura 6 - Saída do ns3::QueueDisc::Stats



Fonte: Elaborada pelo autor

A Tabela 2 apresenta um resumo das configurações das simulações.

Tabela 2 - Tabela de configuração das simulações

Protocolo	Topologia	Replicações	Tempo de simulação (s)
TCP	dumbbell	1	320

Fonte: Elaborada pelo autor.

5.2 Resultados

Com a finalidade do melhor entendimento dos resultados das simulações, esta seção está dividida em quadro 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. Cada cenário apresenta a comparação dos resultados entre as configurações min_{th}-max_{th} do *RED* e uma comparação das melhores configurações min_{th}-max_{th} com a fila *droptail*.

5.2.1 Cenário A

Neste cenário as simulações foram realizadas contendo 60 fontes *TCP* na topologia *dumbbell*, sendo 30 fontes no lado esquerdo e 30 no lado direito, com o objetivo de gerar tráfego com carga leve.

Com as simulações realizadas no Cenário A, observou-se que as melhores configurações min_{th}-max_{th} foram a 14-84, que apresentou o Hurst mais baixo e o melhor throughput médio, e a 21-84, que obteve a menor taxa de descarte (1,77%).

Na Tabela 3 temos a comparação das melhores configurações min_{th}-max_{th} com a fila *droptail*. Nota-se na tabela que os valores das três filas ficaram próximos no valor de Hurst e na taxa de descarte, porém a configuração 14-84 do *RED* apresentou o *throughput* médio bem mais alto comparado aos outros dois.

Tabela 3 - Comparação das filas RED e droptail no Cenário A

60 Fontes TCP			
Fila	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
Droptail	0.6775342	1,93%	0.870345
21-84	0.6864014	1,77%	0.877949
14-84	0.6793804	1,89%	0.90041

Fonte: Elaborada pelo autor

Diante dos resultados analisados, é possível dizer que a configuração 14-84 do *RED* levou boa vantagem, pois apresentou um Hurst praticamente igual ao *droptail*, uma taxa de descarte mais baixa e apresentou o melhor *throughput*. A configuração 21-84 *do RED* obteve a taxa de descarte mais baixa, mas com o Hurst mais alto que os demais, tornando-o uma opção menos interessante para quem busca ter uma auto-similaridade mais baixa.

5.2.2 Cenário B

Neste cenário as simulações foram realizadas contendo 300 fontes *TCP* na topologia *dumbbell*, com 150 fontes em cada lado, com a finalidade de gerar tráfego com carga moderada.

Neste cenário os valores de Hurst ficaram mais próximos em comparação aos resultados do cenário anterior, com diferenças bem pequenas, porém a taxa de descarte de algumas configurações ficou bem acima de outras. No cenário com carga moderada outras três configurações min_{th}-max_{th} obtiveram os melhores resultados, a configuração 22-66, que obteve o melhor *throughput* médio (0.939384 Mbps), a configuração 18-90, que apresentou o

Hurst mais baixo (0.6416019), e a configuração 11-88, que apresentou a taxa de descarte mais baixa (4,62%), sendo uma das poucas que ficou abaixo de 5%.

Na Tabela 4 podemos observar a comparação das melhores configurações min_{th}-max_{th} deste cenário com a fila *droptail*. Observa-se que os valores das quatro filas ficaram parecidos no que se refere ao Hurst, sendo que configuração 22-66 apresentou um Hurst pior em relação aos outros três, além disso a configuração 22-66 obteve a taxa de descarte mais alta, chegando a quase 8%.

Tabela 4 - Comparação das filas RED e droptail no Cenário B

300 Fontes TCP			
Fila	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
Droptail	0.6404869	3,53%	0.927865
18-90	0.6416019	5,13%	0.921854
11-88	0.6478391	4,62%	0.932022
22-66	0.6518602	7,93%	0.939384

Fonte: Elaborada pelo autor

Com os resultados deste cenário, pode-se afirmar que a configuração 11-88 do *RED* apresentou o melhor desempenho na média, afinal teve um Hurst bem próximo do mais baixo (*droptail*), além de possuir a taxa de descarte bem baixa sendo apenas maior que a do *droptail* e *throughput* médio apenas pior que a configuração 22-66.

5.2.3 Cenário C

No Cenário C, as simulações foram realizadas contendo 480 fontes *TCP* na topologia *dumbbell*, com 240 fontes em cada lado para gerar tráfego com carga pesada.

Com os resultados deste cenário notou-se que os valores de Hurst em todas as configurações min_{th}-max_{th} ficaram em torno de 0,64, não havendo quase nenhuma diferença entre o valor de Hurst mais alto e o mais baixo das configurações, porém houve diferenças significativas na taxa de descarte. Com isso, podemos destacar duas configurações min_{th}-max_{th} do Cenário C, as configurações 27-81 e 30-90, que apresentaram o menor Hurst (0.6423202) e melhor *throughput* médio (0.966264 Mbps), respectivamente. Vale salientar que a

configuração 43-86 apresentou a menor taxa de descarte (14,07%), porém também apresentou o pior *throughput* médio, sendo assim não foi considerada como uma das melhores configurações min_{th}-max_{th}.

Na Tabela 5 temos a comparação das melhores configurações min_{th}-max_{th} do *RED* com o *droptail*. Notamos que o valor de Hurst em *droptail* ficou bem parecido com as demais configurações, porém obteve uma taxa de descarte inferior e também um *throughput* pior.

Tabela 5 - Comparação das filas RED e droptail no Cenário C

480 Fontes TCP			
Fila	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
Droptail	0.643709	13,57%	0.914572
27-81	0.6423202	14,24%	0.926943
30-90	0.6441976	14,70%	0.966264

Fonte: Elaborada pelo autor

Com os resultados do Cenário C, é possível afirmar que a configuração 30-90 do *RED* obteve o melhor desempenho, pois apresentou um Husrt e taxa de descarte bem próximos das outras configurações, porém apresentou um *throughput* médio bem mais alto, sendo esse o grande diferencial desta configuração.

5.2.4 Considerações gerais

Com a execução dos três cenários percebeu-se que com o aumento do número de fontes, em relação ao Cenário A, houve uma diminuição da auto-similaridade, além de um melhor *throughput*. Também notou-se que quando o max_{th} foi abaixo de 70%, os resultados sempre estiveram entre os piores em cada cenário, principalmente no que se refere ao valor de Hurst e da taxa de descarte.

Por fim, foi possível criar uma tabela de recomendação (Tabela 6) da configuração min_{th}-max_{th} do RED de acordo com as possibilidades de Hurst (apenas quando for menor do que 0,7) e taxa de descarte.

Tabela 6 - Tabela de recomendação min_{th}-max_{th} do RED

Possibilidades Husrt/Descarte	**Configuração min_{th}-max_{th} do RED indicada**
H < 0,7 / descarte < 2%	14%-84%
H < 0,7 / 2% < descarte < 10%	11%-88%
H < 0,7 / descarte > 10%	30%-90%

Fonte: Elaborada pelo autor

6 CONCLUSÃO

Neste trabalho, falamos sobre a auto-similaridade no tráfego de rede e do algoritmo de gerenciamento de fila *RED*, além de buscar observar o impacto desse algoritmo na auto-similaridade da rede.

Com a execução das simulações e análise de seus respectivos resultados foi observado que diferentes configurações min_{th}-max_{th} influenciam na diminuição da auto-similaridade da rede. Além disso, foi possível criar um padrão de configuração min_{th}-max_{th}, que sofreu apenas uma modificação em relação ao que foi definido primeiramente. O padrão ficou definido da seguinte maneira:

a) $10\% \leq min_{th} \leq 45\%$;
b) $70\% \leq max_{th} \leq 90\%$ (modificação feita após ser observado que em todas as simulações o max_{th} abaixo de 70% ficou entre os piores);
c) $max_{th} - min_{th} \geq 40\%$, onde a diferença entre max_{th} e min_{th} tem que ser no mínimo 40%.

Além disso, foi realizada uma comparação, em cada cenário, do *RED* com o *droptail*. Os resultados mostraram que a utilização do *RED* em todos os cenários é mais recomendada, principalmente pelo melhor aproveitamento do enlace por parte do *RED* (*throughput* médio). Isso torna o *RED* uma melhor opção do que o *droptail* mesmo em redes de carga mais pesada. Sendo assim, o presente trabalho pode servir de ajuda para quem optar pelo o *RED* e podendo utilizar o padrão aqui criado para quando for configurá-lo.

REFERÊNCIAS

AMIN, Farnaz; MIZANAIN, Kiarash; MIRJALILY, Ghasem. Active queue management for self-similar network traffic. In: IRANIAN CONFERENCE ON ELECTRICAL ENGINEERING (ICEE), 21., 2013, Mashhad, Iran. **2013 21st Iranian Conference on Electrical Engineering (ICEE).** Piscataway, NJ: IEEE, 2013. p. 961 - 965.

BRADEN, B. et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. **RFC 2309**, [s.l], p.1-16, abr. 1998.

FIGUEIREDO, Ricardo Nogueira de. **Avaliação de algoritmos de controle de congestionamento como controle de admissão em um modelo de servidores Web com diferenciação de serviço.** 2011. 92 f. Dissertação (Mestrado) - Curso de Ciências de Computação e Matemática Computacional, ICMC/USP, São Carlos, 2011.

FLOYD, Sally; JACOBSON, Van. Random early detection gateways for congestion avoidance. **IEEE/ACM Transactions On Networking**, [s.l.], v. 1, n. 4, p.397-413, ago. 1993.

LAURINI, Márcio Poletti. **Medidas de Persistência a Choques e Eficiência de Mercado para a Taxa de Câmbio R$/US$.** 2002. 140 f. Dissertação (Mestrado) - Curso de Economia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

LELAND, W.e. et al. On the self-similar nature of Ethernet traffic (extended version). **IEEE/ACM Transactions On Networking**, [s.l.], v. 2, n. 1, p.1-15, fev. 1994.

LELAND, Will E. et al. On the Self-Similar Nature of Ethernet Traffic. In: CONFERENCE SIGCOMM '93 SYMPOSIUM, COMMUNICATION, ARCHITECTURES AND PROTOCOLS, 7., 1993, San Francisco. **ACM SIGCOMM '93 Conference.** New York: Acm, 1993. p. 183 - 193.

NETTO, Maria Aparecida Cavalcanti. **A Previsão com a Metodologia de Box-Jenkins.** Disponível em: <http://www.ie.ufrj.br/download/APrevisaoComMetodologiadeBox-Jenkins.pdf>. Acesso em: 02 dez. 2018.

NS3. **BulkSendApplication.** Disponível em: <https://www.nsnam.org/doxygen/group__bulksend.html>. Acesso em: 11 dez. 2018.

NS3. **Flow Monitor.** Disponível em: <https://www.nsnam.org/docs/models/html/flow-monitor.html>. Acesso em: 11 dez. 2018.

NS3. **Ns3::QueueDisc::Stats Struct Reference.** Disponível em: <https://www.nsnam.org/doxygen/structns3_1_1_queue_disc_1_1_stats.html#a90a1735f9d859c9520b5a948246159ff>. Acesso em: 27 nov. 2018.

NS3. **Ns-3 Tutorial.** Disponível em: <https://www.nsnam.org/docs/release/3.13/tutorial-pt-br/singlehtml/index.html#ns-3-tutorial>. Acesso em: 16 nov. 2018.

R. **RoverS {fractal}.** Disponível em: <http://finzi.psych.upenn.edu/R/library/fractal/html/RoverS.html>. Acesso em: 29 out. 2018.

RUAS, Weldisson Ferreira. **Análise de Desempenho de Redes de Filas com Tráfico Modelado por Distribuições de Cauda Pesada.** 2010. 101 f. Dissertação (Mestrado) - Curso de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações – INATEL, Santa Rita do Sapucaí, 2010.

SANTOS, Luciano da Silva. **Tolerância à falha dos agentes de mobilidade do protocolo mobile IP.** 2005. 204 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

SIKDAR, Biplab et al. Queue management algorithms and network traffic self-similarity. In: IEEE HIGH PERFORMANCE SWITCHING AND ROUTING, 7., 2002, Kobe, Japan. **HPSR2002 - Workshop on High Performance Switching and Routing, Merging Optical and IP Technologie - Proceedings.** [s.l]: IEEE, 2002. p. 319 - 323.

SIKDAR, Biplab; VASTOLA, Kenneth S.; KALYANARAMAN, S.. **On Reducing the Degree of Self-Similarity in Network Traffic.** 2002.

SILVA, Leandro Alves da. **Metodologia para a Utilização da Análise de Tráfego Auto–Similar em Redes de Computadores.** 2001. 57 f. Dissertação (Mestrado) - Curso de Engenharia da Computação, Instituto de Pesquisas Tecnológicas do Estado de São Paulo, São Paulo, 2001.

SOUZA, Emanuel Fernando Maia de; PETERNELLI, Luiz Alexandre; MELLO, Márcio Pupin de. **Software Livre R: aplicação estatística.** Disponível em: <http://www.de.ufpb.br/~tarciana/MPIE/ApostilaR.pdf>. Acesso em: 9 nov. 2018.

SURESH, S.; GÖL, Özdemir. Congestion management of self similar IP traffic - application of the RED scheme. In: INTERNATIONAL CONFERENCE ON WIRELESS AND OPTICAL COMMUNICATIONS NETWORKS, 2., 2005, Dubai, United Arab Emirates. **2005 International Conference on Wireless and Optical Communications Networks : March 6 - 8, 2005, Dubai, United Arab Emirates.** Piscataway, NJ: IEEE, 2005. p. 372 - 376.

TAN, Xianhai; HUANG, Yuanhui. Parameters Setting Scheme of RED with Long-Range Dependent Traffic Input. In: INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND MOBILE COMPUTING, 5., 2009, Beijing, China. **Proceedings - The 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2009).** Piscataway: IEEE, 2009. p. 3867 - 3870.

APÊNDICE A – TABELAS DOS RESULTADOS DAS CONFIGURAÇÕES RED COMPLETAS

Tabela 7 - Tabela de resultados das configurações RED no Cenário A

60 Fontes TCP			
RED Min-Max	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
40-80	0.7026598	2,10%	0.83747
41-82	0.6916014	1,98%	0.869804
42-84	0.685075	1,90%	0.897568
43-86	0.685519	1,90%	0.889863
44-88	0.6819808	1,82%	0.892538
45-90	0.6869007	1,87%	0.893388
20-60	0.7070645	2,64%	0.834891
21-63	0.6923675	2,62%	0.84999
22-66	0.6998035	2,59%	0.851652
23-69	0.6926457	2,52%	0.858345
24-72	0.6935525	2,23%	0.867507
25-75	0.70143	2,35%	0.858128
26-78	0.6872071	2,07%	0.889271
27-81	0.6935226	1,92%	0.869062
28-84	0.6850129	1,83%	0.896982
29-87	0.6860649	1,99%	0.887429
30-90	0.6820371	1,84%	0.891814
15-60	0.6989152	2,66%	0.849033
16-64	0.706712	2,60%	0.843005
17-68	0.7050745	2,62%	0.846131
18-72	0.704233	2,35%	0.842538
19-76	0.6903354	2,08%	0.877236
20-80	0.6958242	2,00%	0.857595

21-84	0.6864014	1,77%	0.877949
22-88	0.6862677	1,85%	0.890735
12-60	0.7125922	2,72%	0.813595
13-65	0.7107554	2,51%	0.830449
14-70	0.7095508	2,30%	0.827899
15-75	0.6977283	2,09%	0.848184
16-80	0.6929245	2,01%	0.875844
17-85	0.6906765	1,85%	0.888242
18-90	0.6802764	1,80%	0.896354
10-60	0.7151756	2,62%	0.822621
11-66	0.7151276	2,39%	0.81614
12-72	0.7139835	2,20%	0.814647
13-78	0.6916999	2,02%	0.868961
14-84	0.6793804	1,89%	0.90041
15-90	0.6844097	1,88%	0.894572
10-70	0.7071991	2,29%	0.838314
11-77	0.6892366	2,21%	0.881101
12-84	0.6827979	1,84%	0.891814
10-80	0.6836255	1,91%	0.8948
11-88	0.6829999	1,89%	0.894303
10-90	0.6850535	1,88%	0.89481

Fonte: Elaborada pelo o autor

Tabela 8 - Tabela de resultados das configurações RED no Cenário B

300 Fontes TCP			
RED Min-Max	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
40-80	0.6464704	6,15%	0.923696
41-82	0.6477191	5,71%	0.922801
42-84	0.6425256	5,20%	0.923708
43-86	0.6449197	5,36%	0.923415

44-88	0.6443984	4,97%	0.923181
45-90	0.6430655	4,83%	0.924467
20-60	0.6533527	9,19%	0.929806
21-63	0.6516885	9,41%	0.930319
22-66	0.6518602	7,93%	0.939384
23-69	0.6465086	8,39%	0.925889
24-72	0.6476334	8,00%	0.926875
25-75	0.6477165	6,40%	0.924954
26-78	0.6451987	5,78%	0.923962
27-81	0.6440779	5,41%	0.923402
28-84	0.6430334	5,40%	0.924785
29-87	0.642183	5,15%	0.923075
30-90	0.6441108	4,76%	0.922625
15-60	0.6544548	9,58%	0.931674
16-64	0.6487779	8,95%	0.939097
17-68	0.647635	8,31%	0.927743
18-72	0.6464083	7,93%	0.924914
19-76	0.6465515	6,47%	0.925914
20-80	0.6433835	5,83%	0.925858
10-60	0.6435293	5,99%	0.922794
22-88	0.6460203	4,89%	0.921942
12-60	0.6565959	9,21%	0.927744
13-65	0.6510203	7,62%	0.92755
14-70	0.6533993	8,59%	0.928642
15-75	0.649817	6,35%	0.924482
16-80	0.6476049	6,30%	0.924248
17-85	0.6451491	5,88%	0.922418
18-90	0.6416019	5,13%	0.921854

10-60	0.65667	9,60%	0.923682
11-66	0.651234	8,55%	0.927796
12-72	0.6478608	6,90%	0.938902
13-78	0.6443373	6,26%	0.923362
14-84	0.6458713	5,87%	0.923075
15-90	0.6436545	5,25%	0.920986
10-70	0.6478453	7,02%	0.928209
11-77	0.6451125	6,52%	0.936179
12-84	0.6479913	5,99%	0.932914
10-80	0.6448173	6,12%	0.924534
11-88	0.6478391	4,62%	0.932022
10-90	0.6430881	5,14%	0.921129

Fonte: Elaborada pelo autor

Tabela 9 - Tabela de resultados das configurações RED no Cenário C

480 Fontes TCP			
RED Min-Max	**Hurst**	**Taxa de Descarte (%)**	**Throughput (Mbps)**
40-80	0.6451853	15,57%	0.927368
41-82	0.646402	15,00%	0.927557
42-84	0.6445537	14,40%	0.929289
43-86	0.6455704	14,07%	0.925437
44-88	0.6450191	14,58%	0.926147
45-90	0.6428534	14,22%	0.933458
20-60	0.6450818	15,41%	0.928276
21-63	0.646061	15,79%	0.938574
22-66	0.6467945	15,87%	0.940341
23-69	0.6454753	15,53%	0.946769
24-72	0.6472835	15,09%	0.937642
25-75	0.6449814	15,36%	0.927466
26-78	0.6455083	15,25%	0.927941

27-81	0.6423202	14,24%	0.926943
28-84	0.6443833	14,37%	0.935459
29-87	0.6454559	15,08%	0.927263
30-90	0.6441976	14,70%	0.966264
15-60	0.645015	15,52%	0.940635
16-64	0.6463083	15,67%	0.941832
17-68	0.644785	14,93%	0.926668
18-72	0.6447816	15,01%	0.940123
19-76	0.6478366	15,28%	0.928397
20-80	0.6427979	15,20%	0.92577
21-84	0.6446182	14,90%	0.935391
22-88	0.6470883	14,23%	0.926786
12-60	0.645429	16,48%	0.92743
13-65	0.6487653	15,47%	0.928472
14-70	0.6465724	15,69%	0.931491
15-75	0.646877	14,84%	0.927264
16-80	0.6457209	14,88%	0.936303
17-85	0.6427784	14,86%	0.926856
18-90	0.6432936	14,40%	0.927148
10-60	0.6452269	16,14%	0.929219
11-66	0.6455721	15,38%	0.929575
12-72	0.6455667	15,51%	0.930777
13-78	0.64392	15,50%	0.93134
14-84	0.6427838	14,75%	0.928739
15-90	0.6430137	14,09%	0.932211
10-70	0.644923	15,46%	0.928038
11-77	0.6453102	15,34%	0.927578
12-84	0.6440732	15,13%	0.928506

10-80	0.6450777	14,41%	0.95101
11-88	0.6430122	14,33%	0.927619
10-90	0.6434227	14,48%	0.925934

Fonte: Elaborada pelo o autor