Special Conference Edition, November, 2018

http://dx.doi.org/10.4314/bajopas.v11i1.66S

Bayero Journal of Pure and Applied Sciences, 11(1): 420 - 426
ISSN 2006 – 6996

ANALYSIS OF CONGESTION CONTROL ALGORITHM FOR FOG COMPUTING SYSTEMS WITH INTERNET OF THINGS

[1]Adamu Aminu and [2]Zainab Dahiru Kaita
[1]Umaru Musa Yar'adua University, Katsina State, Nigeria.
Mathematics and Computer Science Department.
[1]aminu@mail.ru: +2348031581658; [2]kaitazainab@gmail.com: +2348035930018

ABSTRACT
With rapid deployment of Internet of Things (IoT) applications which demand real-time, low-latency services, fog computing was proposed as a promising distributed computing paradigm to support the IoT systems. Unlike the centralized cloud computing architecture, fog computing deploys network devices with various degrees of computational and storage capabilities at the edge of the network closer to the IoT devices in order to meet the real-time, low-latency service demands generated by these devices. Considering the rapid growth of IoT systems and the huge traffic generated by billions of IoT devices, there is the need to have an effective algorithm implemented at the fog gateways for congestion control. In this paper a queuing model was proposed to analyze the effectiveness of the Random Early Discard (RED) algorithm with generalized nonlinear loss function for edge gateways of fog computing architecture which supports the IoT applications. Results of the analysis have shown that in high traffic load situations, the probability of service requests' drop increases as the index value of the generalized nonlinear loss function decreases. For light traffic load situations, the throughput of service requests increases as the index value of the generalized nonlinear loss function increases.
Keywords: RED, Delay, Congestion Control, IoT, Fog Computing.

INTRODUCTION

Cloud computing has gained wider acceptance and many organizations worldwide have subscribed for cloud services (Armbrust, *et. al.,* 2010). The cloud computing was popularly known as a promising computing paradigm due to its ability to remarkably reduce computing cost, increase flexibility and scalability (Rimal, *et. al.*, 2009). However, with the invention of IoT systems, the Internet computing is now taking a new dimension (Hong, *et. al.,* 2013; Preden, *et. al.,* 2015; Bonomi, *et. al.,* 2012; Bonomi, *et. al.,* 2014). It was forecasted by Cisco that by the year 2020 over 50 billion smart devices resulted from the deployment of large scale wireless sensor networks, smart vehicles, smart metering, wearable computing, smart home etc., will be added to the Internet and will generate over 43 trillion gigabytes of data (Cisco, 2014; Subhadeep & Sudip, 2016; Blesson, *et. al.,* 2017). The Internet of Things is considered to be the future Internet and is gaining much attention, on its invention, due to the limited storage and computational capabilities of smart devices, cloud computing was chosen as its supportive computing paradigm, since the cloud can provide elastic storage and computational resources to the smart devices on demand (Cisco, 2014; Subhadeep & Sudip, 2016; Blesson, *et. al.,* 2017; Atzori, *et. al.,* 2010). However, the cloud computing architecture is completely centralized in nature, and most of the cloud data centers are located far away from these smart devices, making the cloud computing to be not the right candidate to augment the IoT systems, because most of IoT applications require low latency response, location awareness, geo-distribution and mobility support (Subhadeep & Sudip, 2016). To overcome the cloud computing shortcomings considering the rapid growth of Internet of Things, Cisco proposed a new computing paradigm known as fog computing (Cisco, 2014).

The idea of fog is to transfer some core cloud services towards the edge of the network closer to the smart devices in a decentralized fashion. The fog computing was defined by Blesson, *et. al.,* (2017) as "model to complement the cloud by decentralizing the concentration of computing resources (for example, servers, storage, applications and services) in data centers towards users for improving the quality of service and their experience". The fog computing's target is to harness the untapped computational capabilities of edge nodes such as routers, mobile base stations, switches etc., situated closer to the end devices (Blesson, *et. al.,* 2017; Atzori, *et. al.,* 2010; Subhadeep, *et. al.,* 2015; Yannuzzi, *et. al.,* 2014; Stojmenovic, 2014; Yi, *et. al.,* 2015; Krishnan, *et. al.,* 2015).

The advantage of fog computing paradigm over the conventional cloud computing for low-latency services was justified and its suitability for IoT applications was also recently established (Blesson, *et. al.,* 2017; Subhadeep, *et. al.,* 2015). Subsequently, billions of devices will be located at edge of the network requesting for various services. Most of these devices generate real-time traffic which is high latency and data loss intolerable. Clearly, managing such huge traffic in fog systems especially with IoT applications deployed at large scale is not easy and requires very efficient congestion control algorithm, in order to markedly reduce the service latency and data loss.

This paper analyzes the effectiveness of RED (Random Early Discard) algorithm with generalized nonlinear loss function as congestion control algorithm for edge gateways of fog computing architecture which supports the IoT applications, unlike the traditionally known RED algorithm (Floyd & Jacobson, 1993; Firoiu & Borden, 2000) which was designed with linear drop function. In traditional RED with linear drop function the service request drop is constant regardless of the nature of the traffic, however, RED with generalized nonlinear loss function is very flexible, i.e. if the traffic load is high, the index value of the drop function is decreased to control the congestion, and when the traffic load becomes light, the index value of the loss function is increased to increase the throughput of service requests in the system. For the analysis of RED algorithm with generalized nonlinear drop function, a queuing model was formed and formulas were obtained for the computation of system's key Quality of Services (QoS) parameters (Jun, *et. al.,* 2014).

The rest of the paper is organized as follows. Section II provides the detailed description of fog computing architecture which supports the IoT applications. In section III, a model of the system was presented and mathematical formulas were obtained for the computation of system's QoS parameters. Section IV presents the results of the analysis and section V concludes the paper.

FOG COMPUTING ARCHITECTURE

Rapid deployment of IoT systems and cloud computing limitations to support the real-time latency-sensitive service requests generated by IoT applications raise the need for a new computing paradigm. As a solution, Cisco proposed a new computing paradigm termed fog computing. Fog computing was designed as a distributed computing architecture that is capable of handling billions of IoT devices with various degrees of QoS demands. The basic idea of fog computing was driven from the concept of edge computing, where some services are hosted within the system's edge devices such as gateways, switches, routers and access points, these devices are termed as fog computing devices (Blesson, *et. al.,* 2017; Atzori, *et. al.,* 2010; Subhadeep, *et. al.,* 2015; Yannuzzi, *et. al.,* 2014; Stojmenovic, 2014; Yi, *et. al.,* 2015; Krishnan, *et. al.,* 2015). However, it's paramount to note that fog computing paradigm was designed to augment the cloud computing architecture in the context of IoT systems by extending some services to the edge of the network.

The fog computing is currently at its infant stage, however, some realistic assumptions were made with respect to all entities involved in fog computing architecture by Subhadeep & Sudip, (2016). As mentioned earlier, fog computing was proposed to remedy the shortcomings of cloud computing in supporting IoT systems, hence the two architectures work hand in hand with each other, the fog computing overcomes the cloud's shortcoming via decentralization of computing resources. The fog computing comprises of three (3) tier computing architecture, i.e. lower, middle and upper tiers. The lower tier also known as bottom-most tier consists of smart devices termed Terminal Nodes (TNs). Geographically closed TNs are grouped together to form a virtual cluster (VC). The middle tier is called the fog tier and it consists of edge devices capable of providing various computational and storage services. Due to geographical location issues associated to virtual clusters, the fog tier is also divided into several fog instances (FIs), each fog instance (FI) handles a given VC (Figure 1). Service requests from a VC are directed to its associated FI. The mapping between VCs and FIs is one-to-one mapping; however, due to mobility issues of TNs, hence mapping between TN to FI is flexible and non-static.

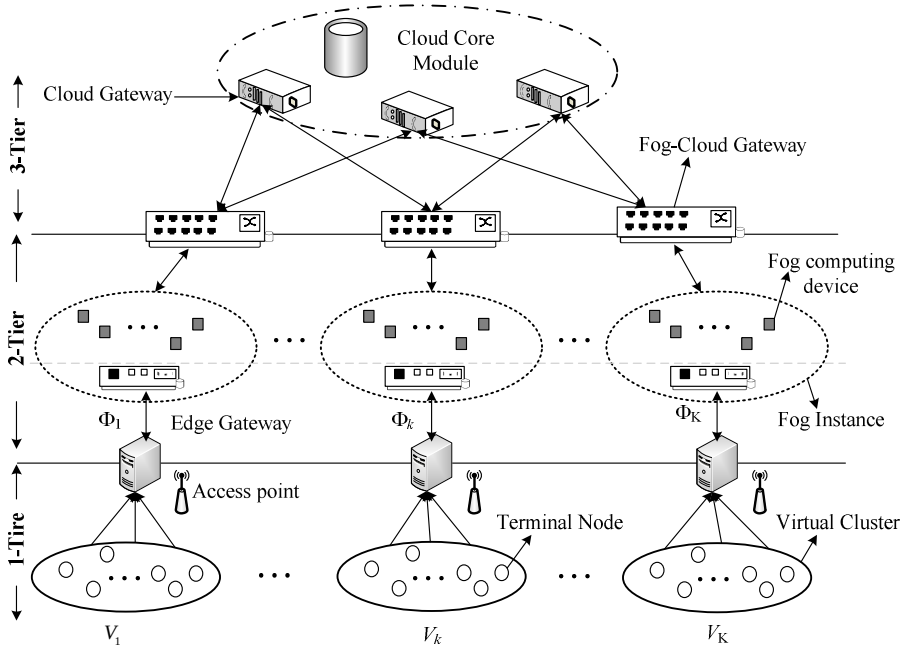

Figure 1. Fog computing architecture

The upper layer consists of the cloud core module, unlike the pure cloud architecture, with fog computing architecture introduced, not all service requests are rendered to the cloud core, instead real-time latency sensitive service requests are hosted at the fog layer, and as such access to the cloud core is done periodically in controlled manner. Considering the vast number smart devices at the lower tier, enormous services requests will be subjected to the fog instances; subsequently it's very crucial to have an effective congestion control at the fog gateways of fog instances to prevent data loss and to satisfy the service demands of real-time latency sensitive service requests.

THE MODEL FOR THE ANALYSIS OF CONGESTION CONTROL ALGORITHM

Let's consider fog instance Φ_k with its associated virtual cluster V_k as in Figure 1. The gateway of the above system is model as M/M/1 queue as shown in Figure 2.

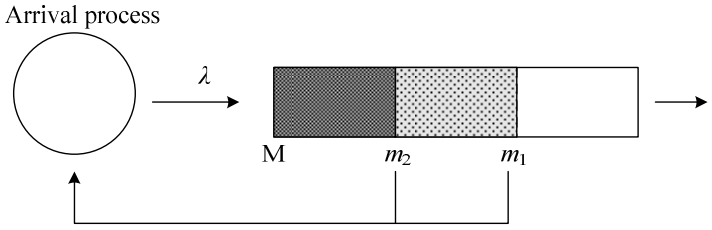

Figure 2. Queuing model of the system

The system consists of a queue with maximum capacity M. As contained in the traditional RED algorithm (Floyd & Jacobson, 1993, Firoiu & Borden, 2000) the queue is divided into three (3) sections via two demarcation values m_1 and m_2, where m_1 is the minimum threshold and m_2 is the maximum threshold. If a service request is currently being served, the other incoming service requests are kept in the queue and served according to FIFO service discipline. Let m be the instantaneous queue length, $m = 0,\mathrm{L}\ ,\mathrm{M}$. When $m < m_1$ no service request will be dropped, however, when $m_1 \le m < m_2$ then service requests will be dropped with probability which increases nonlinearly as a function of m given by (1), unlike the traditional RED this probability increases linearly as a function of average queue length. If $m_2 \le m < \mathrm{M}$, then service requests will be dropped with constant probability p, when $m \ge \mathrm{M}$ then all incoming service requests will be dropped. Due to the feedback between the queue and the arrival process, whenever a service request drop occurs, the arrival is decreased. This work introduces into the traditional RED algorithm a generalized nonlinear loss function with index value $n > 0$ presented in equation (1).

$$d(m)=\begin{cases}0, & 0\le m<m_1\\ \left(\dfrac{m-m_1}{m_2-m_1}\right)^n\cdot p, & m_1\le m<m_2\\ p, & m_2\le m<\mathrm{M}\end{cases} \quad (1)$$

Where $n>0$.

To analyze the performance of RED algorithm with generalized nonlinear drop function presented in (1), let $X(t)$, $t>0$ be the number of all service requests present in the system, and the state space of the system is $\Omega=\{0,\mathrm{L}\ ,\mathrm{M}\}$. Assume that service requests arrive at the system with rate λ distributed according Poisson distribution and service requests are served at rate μ distributed according exponential distribution, where $\rho=\lambda/\mu$. Denote by $q=1-p$, $\pi(m)=\left(\dfrac{m-m_1}{m_2-m_1}\right)^n\cdot p$ and $\gamma(m)=1-\pi(m)$. The state transition diagram of the described system is presented in Figure 3.

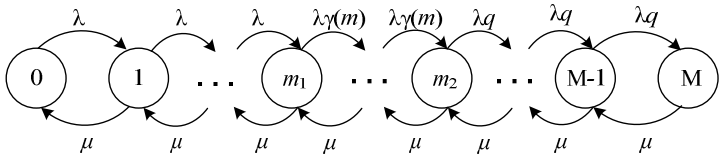

Figure 3. State transition diagram

To obtain the steady state distribution p_m that in the system there are *m* service requests, let's consider the steady state equations presented in (2).

$$\begin{cases}\lambda p_0=\mu p_1, & m=0\\ \lambda p_{m-1}+\mu p_{m+1}=(\lambda+\mu)p_m, & 0<m\le m_1\\ \lambda\gamma(m)p_{m-1}+\mu p_{m+1}=(\lambda\gamma(m)+\mu)p_m, & m_1\le m<m_2\\ \lambda q p_{m-1}+\mu p_{m+1}=(\lambda q+\mu)p_m, & m_2\le m<\mathrm{M}\\ \lambda q p_{\mathrm{M}+1}=\mu p_{\mathrm{M}}, & m=\mathrm{M}\end{cases} \quad (2)$$

By solving the steady state equations we have,

$$p_m=\begin{cases}\rho p_0, & m=1\\ \rho^m p_0, & 1<m\le m_1\\ \prod_{k=m_1}^{m-1}\gamma(k)\rho^m p_0, & m_1<m\le m_2\\ \prod_{k=m_1}^{m_2-1}\gamma(k)\left(\prod_{k=m_2}^{m-1}q\right)\rho^m p_0, & m_2<m\le \mathrm{M}\end{cases} \quad (3)$$

From the law of total probability $\sum_{m=0}^{\mathrm{M}}p_m=1$, we have,

$p_0=\dfrac{1}{G_1+G_2+G_3}$, where $G_1=\sum_{m=0}^{m_1}\rho^m$, $G_2=\sum_{m=m_1+1}^{m_2}\prod_{k=m_1}^{m-1}\gamma(k)\rho^m$, and $G_3=\sum_{m=m_2+1}^{\mathrm{M}}q^{m-m_2}\prod_{k=m_1}^{m_2-1}\gamma(k)\rho^m$

The average number of service requests in the system is given by

$$N = \sum_{m=0}^{\mathrm{M}} m p_m \tag{4}$$

The average waiting time of a service request in the system is obtained from Little's law, i.e.

$$W = \frac{N}{T}, \tag{5}$$

where $T = \sum_{m=1}^{\mathrm{M}} \mu p_m$ is the system mean throughput.

Let Δ be the total drop probability of service requests, and it consists of three (3) components Δ_1, Δ_2 and Δ_3 defined as $\Delta_1 = \sum_{m=m_1}^{m_2-1} p_m \pi(m)$, $\Delta_2 = \sum_{m=m_2}^{\mathrm{M}-1} p_m p$ and $\Delta_3 = p_{\mathrm{M}}$.

$$\Delta = \Delta_1 + \Delta_2 + \Delta_3. \tag{6}$$

I. SIMULATION RESULTS

The simulation experiment was conducted with OMNet++ simulator (Buzura, *et. al.*, 2013). The values of the model parameters used in the experiment are $\mu = 0.02$, $m_1 = 10$, $m_2 = 30$ and $\mathrm{M} = 40$. The input values for the experiment were obtained from (Jun, *et. al.*, 2014; Bonald, *et. al.*, 2000).

Firstly, the drop probability of service requests was analyzed for different index values *n* of the nonlinear loss function under different traffic loads. The results of the analysis are presented in Figure 4.

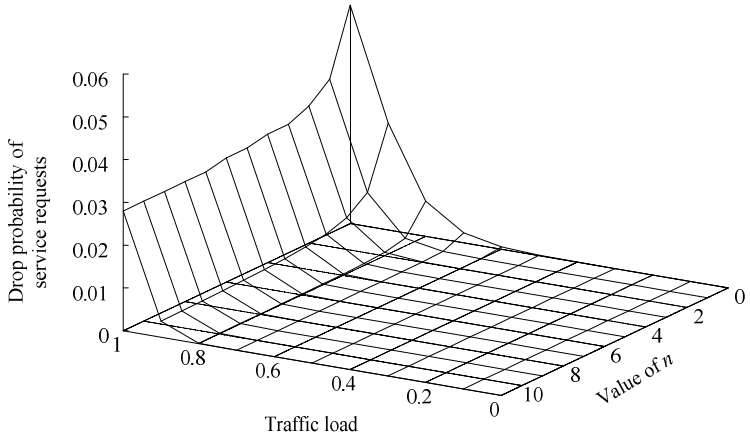

Figure 4. Drop probability of service requests vs. traffic load and index values *n*

It can be seen from the graph in Figure 4, for large traffic load, the drop probability of service requests increases as the index value *n* of the loss function decreases.

The average number of waiting service requests in the system was also analyzed for different index values *n* of the nonlinear loss function. The results of the analysis are presented in Figure 5, from the graph it can be observed that with a fixed value of *n*, the average number of waiting service requests increases as the traffic load increases, however, when the traffic load becomes high, the average number of waiting service request decreases as the index value *n* of loss function decreases. These results can be connected to the results found in Figure 4, whenever a service request is dropped, the flow of incoming service requests is reduced.

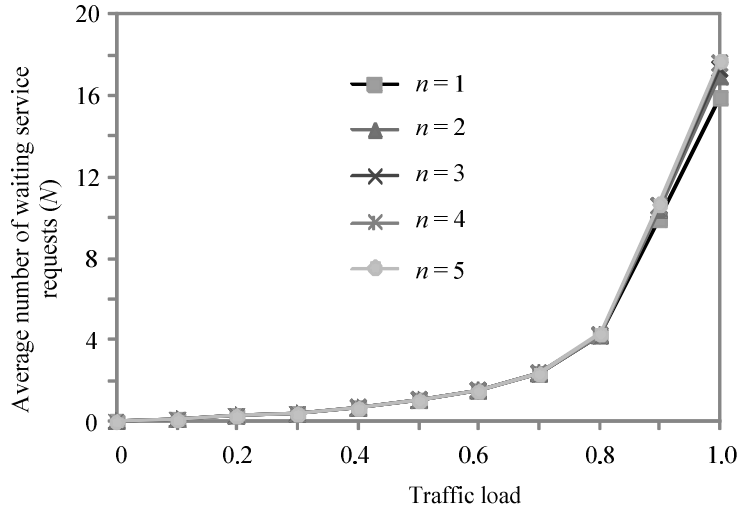

Figure 5. Average number of waiting services requests vs. traffic load and index value *n*

Further, the average waiting time of a service request in the system was also analyzed for the considered system, similarly it can be observed from the graph presented in Figure 6 that for a fixed index value *n*, the average waiting time decreases as traffic load decreases, however, for large traffic load, the average waiting time decreases as the index value *n* of the loss function decreases.

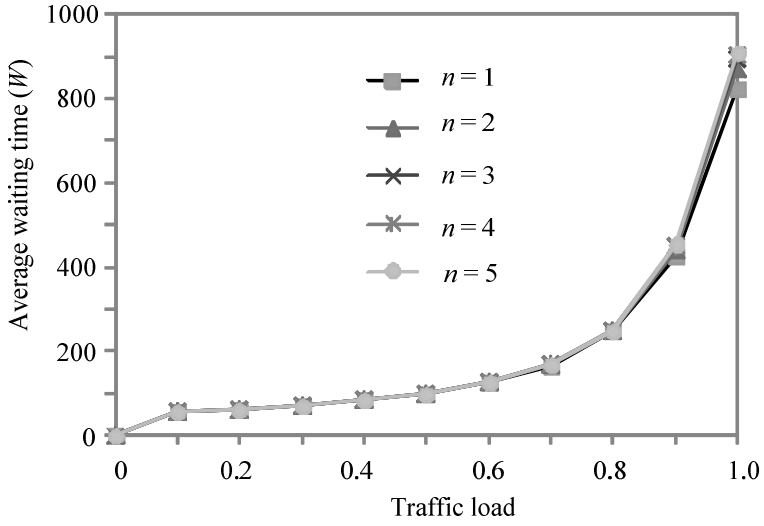

Figure 5. Average waiting time vs. traffic load and index value *n*

CONCLUSION

In this paper a model was presented for the analysis of congestion control algorithm for the edge gateways of fog computing architecture which supports the IoT systems. A generalized nonlinear drop function was introduced into the RED algorithm which makes the algorithm to be flexible for high and light traffic load situations. Queuing model was developed and formulas for computation of key QoS parameter for the considered system were obtained. The key QoS parameters used were the average number of waiting service requests in the system, the average waiting time of a service request in the system and the drop probability of service requests. The results obtained have shown that for high traffic load, the drop probability of service requests decreases as the index value *n* of nonlinear loss function increases. The results have also shown that the waiting time and the average number of service requests decrease as traffic load increases but with decrease in the index values *n* of the loss function.

REFERENCES

Armbrust, M., et al. (2010). A view of cloud computing. ACM Commun. Mag, 53, (4), 50–58.

Rimal, B.P., et. al. (2009). A taxonomy and survey of cloud computing systems. 5th Int. Joint Conf. on INC, IMS and IDC, Seoul, South Korea, 44–51.

Hong, K., et al. (2013). Mobile fog: A programming model for large-scale applications on the internet of things. Proc. of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, 15–20.

Preden, J., et al. (2015). Data to decision: pushing situational information needs to the edge of the network. IEEE Int. Inter-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support, Orlando, USA, 158–164.

Bonomi, F., et al. (2012). Fog computing and its role in the internet of things. Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing (ACM), Helsinki, Finland, 13–16.

Bonomi, F., et al. (2014). Fog Computing: A platform for internet of things and analytics', in Bessis, N., Dobre, C. (Eds.): 'Big data and internet of things: a roadmap for smart environments – part I. Springer International Publishing, Switzerland, vol. 546, 169–186.

MarketWatch (2014). Cisco delivers vision of fog computing to accelerate value from billions of connected devices. http://www.theiet.org/resources/ journals/research/index.cfm.

Subhadeep, S., Sudip, M. (2016). Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. IET Networks, vol. 5, issue 2, 23 – 29.

Blesson, V., et al. (2017). Feasibility of Fog Computing. arXiv:1701.05451v1 [cs.DC].

Atzori, L., et al. (2010). The internet of things: A survey. Computer Networks, vol. 54, no. 15, 2787 – 2805.

Subhadeep, S., et al. (2015). Assessment of the Suitability of Fog Computing in the Context of Internet of Things. IEEE Transactions on Cloud Computing, DOI 10.1109/TCC.2015.2485206, 2168-7161.

Yannuzzi, M., et al. (2014). Key ingredients in an IoT recipe: Fog Computing. Cloud computing, and more Fog Computing. Athens, Greece, 325–329.

Stojmenovic, I. (2014). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. Australasian Telecommunication Networks and Applications Conf., Southbank, Australia, 117–122.

Yi, S., et al. (2015). A survey of fog computing: concepts, applications and issues. ACM Proc. of the 2015 Workshop on Mobile Big Data, Hangzhou, China, 37–42.

Krishnan, Y.N., et al. (2015). Fog computing - Network based cloud computing. 2nd Int. Conf. on Electronics and Communication Systems, Coimbatore, India, 250–251.

Jun, Hu., et al. (2014). Modeling and Analysis on Congestion Control in the Internet of Things. IEEE ICC 2014, Ad-hoc and sensor networking symposium, 434 – 439.

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

Firoiu, V and Borden M. (2000). A study of active queue management for congestion control. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, INFOCOM 2000, 3, 1435–1444.

Buzura, S., et al. (2013). Simulations framework for network congestion avoidance algorithms using the omnet++ ide. Roedunet International Conference (RoEduNet), 1–8.

Bonald, T., et al. (2000). Evaluation of RED performance. Proc. of the IEEE INFOCOM 2000, 1415 – 1424.