A Discrete Event Simulation of Fair Bandwidth Share Mechanism CHOKe-FS

Gamil Gardan[1], Rohaya Latip[1,2]
[1]*Faculty of Computer Science and Information Technology Universiti Putra Malaysia, Serdang, Malaysia.*
[2]*Institute for Mathematical Research, Universiti Putra Malaysia.*
j.jar49n@gmail.com

***Abstract*—CHOKe-FS is a partial state Active Queue Management (AQM) of fair bandwidth share mechanism among different flows in the same link. There is no public general programming language (GPL) available for this mechanism. This paper focused on the development of a Discrete Event Simulation (DES) for the proposed partial state Active Queue Management (CHOKe-FS) to simulate this mechanism with different simulation environment. CHOKe-FS uses RED algorithm probability to match the incoming packet with the selecting packets from the queue to decide either to drop packet or allow it to enter the queue. CHOKe-FS uses same CHOKe technique with three main differences. In this research, we focused on the development of discrete event simulator to implement one of active queue management mechanisms which is called CHOKe-FS and compare it with other three active queue management mechanisms which are called RED, CHOKe, CHOKeD. The results gained from this research showed that the developed simulators have produced almost the same results as previous simulators. CHOKe-FS, CHOKeD and CHOKe maintain fairness in the share link, identify and penalize non-responsive flows while RED fails.**

***Index Terms*—AQM; CHOKe-FS; CHOKe; Congestion Control; Discrete Event Simulation.**

I. INTRODUCTION

CHOKe-FS is a partial state Active Queue Management (AQM) technique and it is a packet-dropping mechanism for classifying and restricting unresponsive or misbehaving flows during congestion. The number of active flows directly calculated in the queue buffer to offer the fair-share. CHOKe-FS depends on two mechanisms on [1], it uses RED technique to manage the buffer and CHOKe to match the packet from unresponsive flow to be dropped.

In this paper, we will simulate the original work of CHOKe-FS: CHOKe with Fair Bandwidth Share by Raza et al [12]. We use Discrete Event Simulation (DES) to demonstrate movement of packets from multiple sources to a single destination (sink) through a single queue. The traffic in the queue may be exhausted by only one type off low which is unresponsive or misbehaving flows. This Discrete Event Simulation (DES) will simulate how to address the problem of fair-bandwidth allocation among those flows. The proposed work is inherited from CHOKe and RED but it differs from CHOKe in three points 1) Queue region is divided into four regions. 2) The drawing factor is adjusted automatically by using the average queue occupancy, and choosing multiple packets from the buffer. 3) Offering fair share by estimating the number of active flow and drop the same type of incoming packets. Network overall performance will be measured in terms of Goodput.

II. RELATED WORK

In IP-Network with best effort, Active Queue Management enables the router to: 1) detect early congestion, 2) give an early alert by dropping or reducing the sending rate of the packets before the overflow happens in the queue. Nevertheless, some sources might ignore an early warning signals because it wants to receive higher bandwidth or because of their unresponsive nature. This leads congestion and bottleneck at the router causes unfairness on the flow. The responsive flows will suffer and the drooping rate will be increase, if the AQM scheme does not provide an efficient treatment to unresponsive and responsive flows.

Many researches have been conducted to solve the congestion in router based scheme. [1] proposed RED (Random Early Detection) algorithm to solve the problem. The algorithm proposed two threshold values: Min and Max. Every time the new packet arrives, a new average for the queue length will be calculated. If the average queue length lesser than Min, the packet placed in the buffer. Otherwise, if the average of the queue length is greater than Max, the packets will be dropped.

CHOKe mechanism uses RED concept in [2], where they used the Min and Max threshold, the research work also added the probability mechanism where if the average queue length larger than the Min threshold, one packet will be chosen randomly the victim (CHOKe victim), this packet will be compared with the new arrived packet, if both comes from same stream both packet will be dropped (CHOKe hit), otherwise the new arrived packet will be replaced in the queue with P probability. Further studies have been conducted to demonstrate CHOKe properties [3].

An enhanced CHOKe was proposed by [4] named xCHOKe used a table and named it as Lookup table, this Lookup table store the CHOKe hit to check with the new arrival packet, if arrived packet have same stream id then the packet will be dropped and xCHOKe will scan the table again and increase the hit counter every time makes the drop. If the packet is not in the table, then the algorithm will create a new row with *counter =1*, where every time the new packet coming with same stream id the hit counter will increase to count the dropped packets.

[5] add a new parameter *maxcomp*, this parameter will have value 2 or more. This parameter determines the max number of successful comparison, if the number of successful comparison equal to *maxcomp* then the packets will be dropped. The authors made a comparison between RED, CHOKe and gCHOKe. It shows from the results that gCHOKe has a better result in term of queuing latency.

CHOKeW algorithm, which used CHOKe technique in dropping the packet to achieve bandwidth allocation, the main difference that the CHOKeW excludes RED technique (Min and Max threshold) where it will depend on the priority of arrived packets and the congestion status to determine maximum number of packets stored in the buffer of router. CHOKeW adjusted the number of packets comparison for drop purpose depending on congestion level. CHOKeW simulation shows its capability to have higher bandwidth share with the larger priority with good fairness and protecting TCP against high speed unresponsive flows. However, there is a few glitches that cannot solve by CHOKeW: 1) the bandwidth differentiation with various priorities becomes smaller when the flow number increases. 2) CHOKeW has a poor performance with a bursty traffic. 3) with increasing in network congestion, CHOKeW cannot cope with the bandwidth allocation with nonresponsive flows. CHOKeW proposed by [6] concerned with bandwidth differentiation and TCP protection in order to improve the quality of service(QOS)for TCP/IP networks authors claim that no previous research conducted combining both tasks.

[7], [8] algorithms are an extension of CHOKeW and used the matching technique of CHOKe with multistep increasing and single step decreasing. The authors conducted the extension to solve CHOKeW limitation. [9] proposed CHOKe with recent drop history CHOKe-RH with same CHOKe principle but with different matching comparison technique, which consist of two parts: first, the basic CHOKe comparison and secondly penalty for unresponsive flow, where the number of comparison edited dynamically depending on the average buffer size. CHOKe-RH keeps the recent dropped packet history to store it as flow ID to use the history for punishing the unresponsive flow. The method simulated with NS-2 and showed better flow fairness comparing with RED, CHOKe and CHOKeR.

High bandwidth likely to have more unresponsive flow with more packets in the queue. In order to solve this problem CHOKeD proposed by [10] where their algorithm have same CHOKe concept but, it increases the number of dropped packet in matching comparison, as the queue occupancy increases, the CHOKeD increases the dropping process. Another researches have been conduct to improve the protection from router congestion, the Queue rate management (QRM) proposed by [11] protecting the router from overflow by checking with the allowing rate. And consequentially checking whether drop or keep the incoming packets. Mathematical model and NS-3 simulation shows that there is no way to exceed the allowing rate. The proposed algorithm provides sits efficiency comparing with CoDel, RED and GREEN, in term of throughput, quality of service and performance.

III. ALGORITHMS

There are four algorithms implemented in this paper. The performance of those four algorithms will be compared in terms of Goodput or overall arriving data.

A. RED algorithm

Random early detection (RED), also known as random early discard or random early drop is a queuing discipline for a network scheduler suited for congestion avoidance. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. It maintains an exponentially moving average queue size that indicates the level of congestion in the router and drops incoming packets with a certain probability dependent on the queue size.

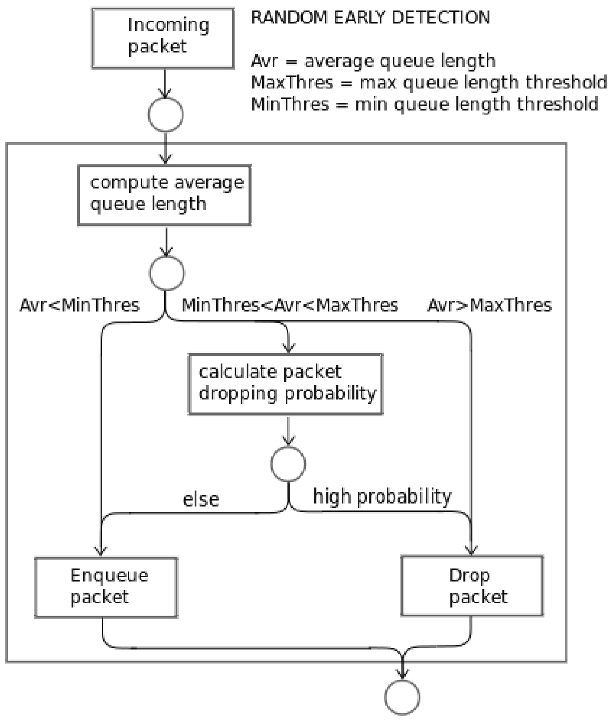
RANDOM EARLY DETECTION
Avr = average queue length
MaxThres = max queue length threshold
MinThres = min queue length threshold
Incoming packet
compute average queue length
Avr<MinThres
MinThres<Avr<MaxThres
Avr>MaxThres
calculate packet dropping probability
else
high probability
Enqueue packet
Drop packet

Figure 1: RED Flowchart

RED algorithm process is presented in Figure 1. The flowchart of how RED calculates packet dropping probability is illustrated in Figure 1. The probability of dropping packet is likely to be high if the average queue length is near to max queue length threshold. Incoming packet will be dropped if that probability is high otherwise it will be entered to the queue.

B. CHOKe algorithm

CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) [2] is a queue management algorithm that used to prevent non-responsive flows using the flow information of queue buffer occupancy of each flow. CHOKe calculates the average occupancy of the FIFO buffer using an exponential moving average, just as RED does. It marks and used two thresholds called minimum threshold and maximum threshold. If the average queues size is less than *min-th*. The arriving packets queued in FIFO buffer. If the total arrival rate of UDP is smaller than the output capacity of link, the average queue the packets are not always dropped directly and queue size should not build up to *min-th*.

Every arriving packets dropped when the average queue size is greater than *max-th*. This makes the occupancy of queue back to below *max-th*. If the average queue size is bigger than *min-th* randomly selected packets from FIFO buffer compared with arriving packets from. These packets called drop candidate packets. If they have the same flow ID (IP address of source and destination, source and destination port address, etc.) they are both dropped. Otherwise, if both packets do not have the same flowID, the candidate packet stay at the buffer and the arriving packets dropped that depends on the average queue size. The drop probability is computed exactly as in RED. In particular, this means that packets are dropped with probability 1 if they arrive when the average queue size exceeds max-th.

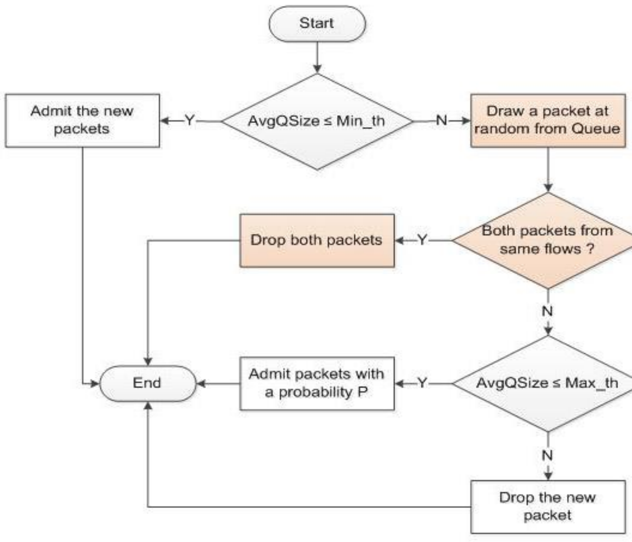

Figure 2: CHOKe Algorithm Flowchart

Figure 2 clarifies the CHOKe algorithm steps. The flowchart on Figure 2 describes CHOKe algorithm behavior which is differing from RED in the middle region where the average queue size is between min-th and max-th threshold. CHOKe in this region choose a random packet to compare it with the incoming packet and drop both of them if they are from the same flow.

C. CHOKeD algorithm

CHOKeD is a stateless Active Queue Management (AQM) scheme, proposed by [10] to protect responsive flows from unresponsive flows and provide fairness between these flows in the Internet. In CHOKeD, match-drop comparisons have been used to keep the responsive flows safe. The number of packets which has been selected as a drop candidate packet is increased depend on the ratio of queue occupancy by using match-drop comparisons. When the packet arrives to the queue CHOKeD examines the queue which has been divided into two regions the front and rare regions and choose a drawing factor and draws number of drop candidate packets dynamically from the rear queue region based on the queue region in contrast with CHOKe which is based on the average queue size.

CHOKeD is differ from CHOKe and CHOKe-FS on the middle region of the queue as show in Figure 3. It divides this region into two equal regions (front and rear) and CHOKe deal with it as one region while CHOKe-FS divides it to four regions. The problem with CHOKeD its complexity which is considered high in compare with other mechanisms and that is because it calculated number of drop candidate packets from the rare region at the end matchs them with incoming packet if any one of them is matching with that incoming packet it will drop all candidate packets and incoming packet. At the second step if there are no packets matches, it will calculate another number of drop candidate packets from the front region.

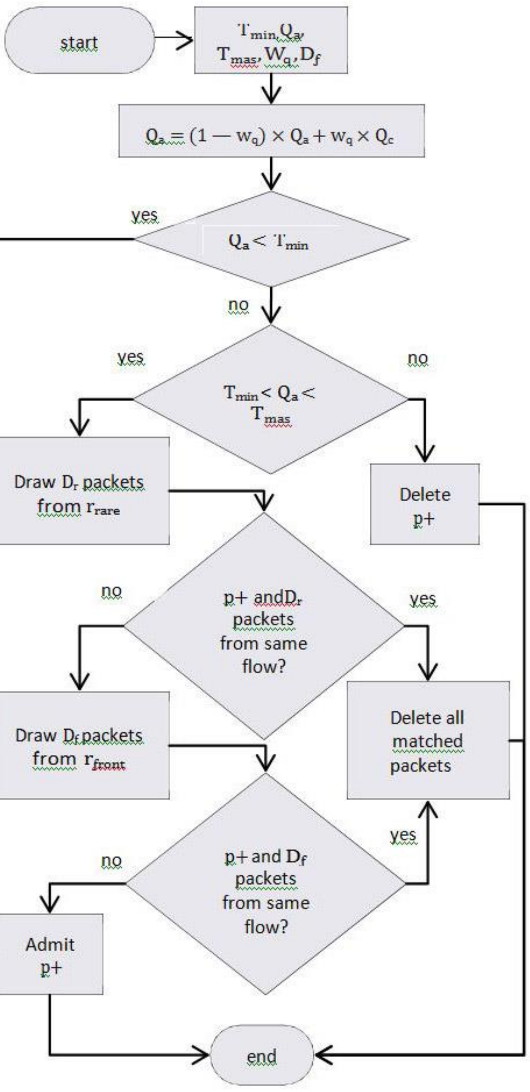
start
$T_{min}, Q_a, T_{mas}, W_q, D_f$
$Q_a = (1 - w_q) \times Q_a + w_q \times Q_c$
yes
$Q_a < T_{min}$
no
yes
no
$T_{min} < Q_a < T_{mas}$
Draw D_r packets from r_{rare}
Delete p+
no
p+ and D_r packets from same flow?
yes
Draw D_f packets from r_{front}
Delete all matched packets
yes
no
p+ and D_f packets from same flow?
Admit p+
end

Figure 3: CHOKeD Algorithm Flowchart

It is clear in Figure 3 that CHOKeD divides the queue into four regions. CHOKeD behaves with the first region and last region similarly to RED and CHOKe. It admits all incoming packets while the average queue size less than min threshold and drops all incoming packets while the average queue size greater than max threshold.

D. CHOKe-FS algorithm

The main target of CHOKe-FS is to avoid the shortcomings of CHOKe by using flow state information to enhance the fairness at router. CHOKe does not consider the level of congestion at the router, it divides queuing region into only three regions and it works at the middle region which lies above the lower threshold min-th and less than the higher threshold max-th and whatever the current queue size has reach in this region it will deal with it by same mechanism. CHOKe- FS overcome this problem by dividing this region into four regions to classify the level of congestion and change the way of matching process to avoid the congestion. In CHOKe it is assumed that unresponsive flows will send more packets than responsive flows which make unresponsive flows overflow the queue which leads to unfair sharing for the queue. However, CHOKe-FS provides a mechanism to calculate the fair share and gives every flow its' fair chance to send its packets.

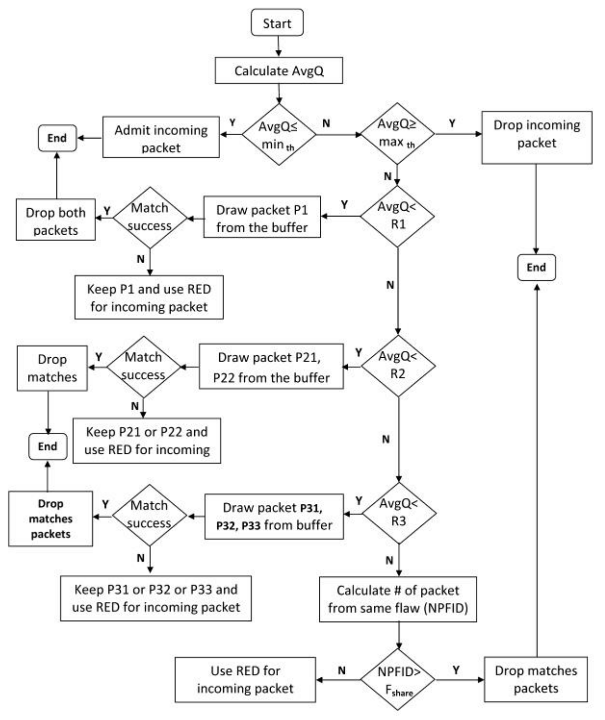
Start
Calculate AvgQ
AvgQ≤ min th
Y
N
Admit incoming packet
End
AvgQ≥ max th
Y
Drop incoming packet
N
AvgQ< R1
Y
Draw packet P1 from the buffer
Match success
Y
Drop both packets
N
Keep P1 and use RED for incoming packet
End
N
AvgQ< R2
Y
Draw packet P21, P22 from the buffer
Match success
Y
Drop matches
N
Keep P21 or P22 and use RED for incoming
End
N
AvgQ< R3
Y
Draw packet P31, P32, P33 from buffer
Match success
Y
Drop matches packets
N
Keep P31 or P32 or P33 and use RED for incoming packet
N
Calculate # of packet from same flaw (NPFID)
NPFID> Fshare
N
Use RED for incoming packet
Y
Drop matches packets

Figure 4: CHOKe-FS Algorithm Flowchart

The flowchart in Figure 4 describes CHOKe-FS. It divides the queue into three regions and it divides the middle region to four regions. The behavior of CHOKe-FS with the first and last region similar to the previous three AQMs. For the first middle region where the average queue size less than R1 as shown in Figure 5 it acts like CHOKe by drawing only one packet to compare it with the incoming packet. CHOKe-FS behaves like RED in all other three middle regions when the drawing packets are not from the same flows of incoming packet flow as shown in Figure 4.

IV. Simulation Model

In this paper, the basic model of packet simulation will be used, which is made up of multiple transmitting nodes (sources), single queue, and single destination (sink). The model diagram is as shown in Figure 5.

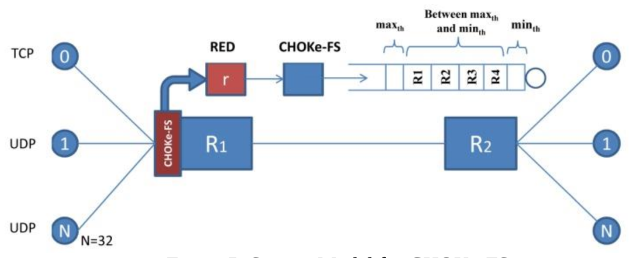

Figure 5: System Model for CHOKe-FS

In our simulations, we use 1 UDP flow and 31 TCP flows and the link between two getaways (R1 and R2) is used as the bottleneck link in this scenario as it shown in Figure 5. Packets size for all flows is 1000 bytes and the buffer size in the simulation is 500 packets. The values of min threshold and max threshold are 20% and 80% of the queue buffer size respectively.

A. Simulation Environment

A general purpose programming language, C++ has been used to implement RED, CHOKe, CHOKe-D and CHOKe-FS and Poisson distribution is used for packet generation purpose. The parameters' values in Table 1 are used to validate the results of proposed simulation with [12]. Statistical composition of the system is a Packet.

Table 1
Simulation Parameter

	Parameter	Value
Simulation	Duration	100s
	Replication	500
Topology	Queue Type	RED, CHOKe, CHOKeFS, FIFO
	Buffer Size	500 pakets
	RRT	1ms
	Bottleneck-Link Capacity	2500 packets
	Max Number of sources	32 sources
	Bottleneck link connectivity	R1 to R2
Trafific	Packet size	1000 B
	UDP load	2500
	TCP Characteristic	N=31
	UDP Characteristic	1

B. Performance Metrics

To evaluate the performance any active queue management schemes different performance metrics have to be used and for this purpose we will use two performance metrics which are Fairness and Goodput. The bottleneck link in the network is represented by the link between Router 1 and Router2. The simulation simulates the network with 1Mbps link capacity shared by1 UDP flow and 31 TCP flows.

a. Fairness

Fairness is considered as a main aim of any active queue management scheme. In network it used to define whether applications or protocols are using shared network resources in a fair manner. Jain's Fairness Index is used to measure the fairness of CHOKe-FS in the network. The following equation is used to determine Jain's Fairness Index.

$$\mathrm{J}(x_1, x_2, \ldots\ldots, x_n) \quad = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \qquad (1)$$

b. Goodput

Goodput is the measurement of the overall performance of the network. It is defined as the total bandwidth received by user after excluding the duplicate packets. If the queue length in the routers is not stable, i.e., it fluctuates a lot, then the duration of delay between the packets will not be uniform, which will result in a high jitter.

V. Simulation Result

In this section the simulator has been evaluated and validated by using the parameters listed in Table 1 and it is based on the previous work [12], [10] which aimed to avoid congestion. The main target of this evaluation is to prove the ability of developed GPL simulator of AQM mechanisms to avoid congestion and improve the overall performance. The performance of four stated AQMs schemes are evaluated by using GPL and network topology shown in Figure 5.

A. Fairness

Figure 6 shows the result comparison of Jain's Fairness Index for the stated AQM mechanisms, it proves that GPL proposed simulation is successfully implemented. The differences amongst four algorithms are clear as can be seen from the Table 2.

Table 2
Fairness Results

AQM	JFI GPL Value
RED	0.028152
CHOKe	0.850430
CHOKeD	0.854013
CHOKe-FS	0.936365

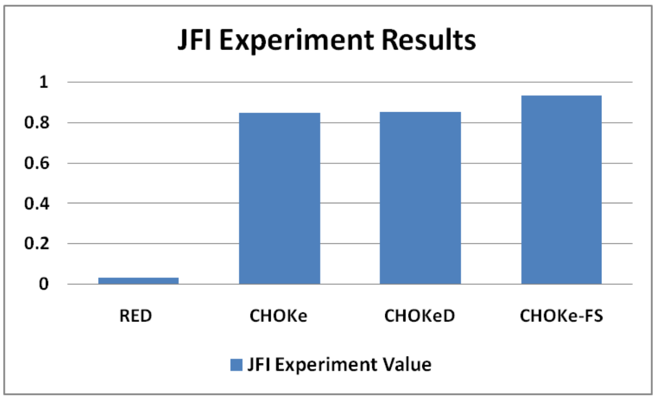

Figure 6: Fairness Results

Table 2 and Figure 6 show the simulation result of four active queue management schemes. These results are conducted by DES of stated AQMs mechanisms. As it's shown in the result, RED is completely fails to offer the fairness for different flows in the shared link while CHOKe and CHOKeD are providing a very good fairness and CHOKe-FS is the best technique to be used for this purpose.

B. Goodput

Figure 7 shows the result comparison of overall packet arrival for the stated AQM mechanisms, it proves that GPL proposed simulation is successfully implemented.

Table 3
Goodput Results

AQM	JFI GPL Value
RED	0.216
CHOKe	0.564
CHOKeD	0.583
CHOKe-FS	0.723

Goodput of RED, CHOKe, CHOKe-D and CHOKe-FS is calculated and presented in Table 3. The results gained from DES simulations for those four AQMs schemes proof that CHOKe-FS can provide a very high amount of Goodput with very high level of fairness. In addition, CHOKe and CHOKeD also proved a very close result with CHOKe-FS. At the low level of RED comes as with very low amount of Goodput and very low level of fairness because an unresponsive flow starved the bottleneck of the link and its high level of droping packets.

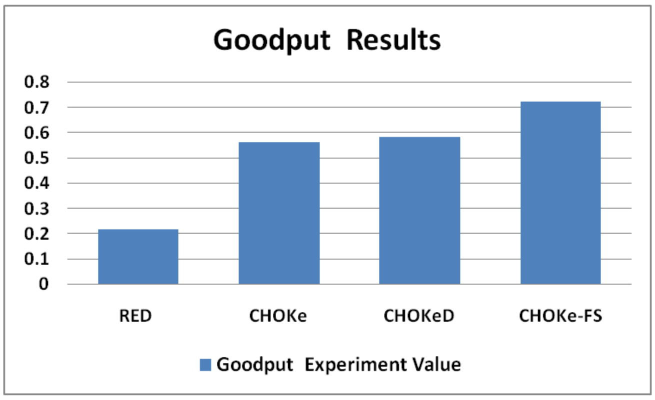

Figure 7: Goodput Results

VI. CONCLUSION

This study has significantly developed a Discrete Event Simulation (DES) using General Purpose Programming Language (GPL) for CHOKe-FS active queue management for different flows in the same link to share bandwidth in a fair manner. The proposed simulator has an easy and stand alone simulation configuration for stated AQMs. To verify and validate the developed simulator, an extensive number of experiments have been done. Our GPL simulator has control the congestion by distributing the queue capacity between flows in a fair way and has control the behavior of unresponsive flow by minimizing its flow on the link. CHOKe, CHOKeD and CHOKe-FS identify and penalize non-responsive flows and maintain fairness in the shared link among different traffic flows, while RED does not maintain fairness nor penalize non-responsive. CHOKe-FS, CHOKeD and CHOKe are providing high amount of Goodput with a high level of fairness for different flows on the same link. They can protect responsive flows from unresponsive flows. RED uses early detect to drop or mark the packet which leads to fail to protect responsive flows or provide a good amount of Goodput.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to UPM for the steady support on my work, for endurance, motivation, and enormous expertise, and to my lecturer Prof. Dr. Shamala A/p K Subramaniam for her motivation and enormous advices.

REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking (ToN), vol. 1, no. 4, pp. 397–413, 1993.

[2] R. Pan, B. Prabhakar, and K. Psounis, "Choke-a stateless active queue management scheme for approximating fair bandwidth allocation," in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2. IEEE, 2000, pp. 942–951.

[3] S. E. Dominic and G. R. Krishna, "Choke: A stateless queue for securing flows," (IJCSIT) International Journal of Computer Science and Information Technologies, vol. 6, no. 2, pp. 1457–1459, 2015.

[4] P. Chhabra, S. Chuig, A. Goel, A. John, A. Kumar, H. Saran, and R. Shorey, "Xchoke: Malicious source control for congestion avoidance at internet gateways," in Network Protocols, 2002. Proceedings. 10th IEEE International Conference on. IEEE, 2002, pp. 186–187.

[5] A. Eshete and Y. Jiang, "Protection from unresponsive flows with geometric choke," in Computers and Communications (ISCC), 2012 IEEE Symposium on. IEEE, 2012, pp. 000 339–000 344.

[6] S. Wen, Y. Fang, and H. Sun, "Chokew: bandwidth differentiation and tcp protection in core networks," in Military Communications Conference, 2005. MILCOM 2005. IEEE. IEEE, 2005, pp.1456–1462.

[7] L. Lu, H. Du, and R. P. Liu, "Choker: A novel aqm algorithm with proportional bandwidth allocation and tcp protection," IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pp. 637–644, 2014.

[8] H. Du, Y. Xiao, and K. Kim, "Mchokem algorithm with assured bandwidth allocation in diffserv networks," Journal of Systems Engineering and Electronics, vol. 21, no. 4, pp. 531–536, 2010.

[9] Z. Hussain, G. Abbas, and U. Raza, "Choke with recent drop history," in Frontiers of Information Technology (FIT), 2015 13th International Conference on. IEEE, 2015, pp. 160–165.

[10] S. Manzoor, G. Abbas, and M. Hussain, "Choked: Fair active queue management," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. IEEE, 2015, pp.512–516.

[11] M. Casoni, C. A. Grazia, M. Klapez, and N. Patriciello, "Qrm: A queue rate management for fairness and tcp flooding protection in mission-critical networks," Computer Networks, vol. 93, pp. 54–65, 2015.

[12] U. Raza, G. Abbas, and Z. Hussain, "Choke-fs: Choke with fair bandwidth share," in Information and Communication Technologies (ICICT), 2015 International Conference on. IEEE, 2015, pp. 1–5.