IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-11, Nov 2018 E-ISSN: 2347-2693

A Router based Hybrid Approach for Congestion Control in High speed Wired Networks

V. Kushwaha[1*], R. Gupta[2]

[1]Dept. of Computer Science, Banaras Hindu University, Varanasi, India
[2]School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India

[*]*Corresponding Author: vandanakus@bhu.ac.in*

Available online at: www.ijcseonline.org

Accepted: 18/Nov/2018, Published: 30/Nov/2018

***Abstract*—** Efficient management of congestion problem in High speed wired networks is a challenging issue due to higher bandwidth, large queuing delay, high burstiness and heterogeneous traffic flows. In this paper a hybrid router based mechanism New-Fair-Queuing-CoDel (nfqCoDel) has been proposed by incorporating the functionality of both, a packet scheduler and active queue manager in a single module called 'Fair Active Queue Manager (FAQM)'. nfqCoDel actively manage the router queue length with an objective to control the excessive delay experienced by source host and maintain the weighted fairness among different hosts. Thus nfqCoDel is able to solve the buffer-bloat issue while providing weighted fairness among different sources. Simulation results, implemented in NS-2, prove that 'nfqCoDel' performs better than other AQMs by providing maximum and stable throughput, stable average queue length, controlled delay at router buffer and weighted fairness among competing traffic sources.

***Keywords*—** High Speed Networks, Congestion Control, Fair Queuing Approach, Active Queue Management, Buffer-Bloat.

I. INTRODUCTION

The number of Internet users has drastically increased over the past few decades. This significant increase in Internet traffic results in network congestion at router queue. In addition to that, over the past few years, as cost of Random Access Memory (RAM) has decreased drastically, people started using high capacity RAM in routers that result in having large buffer area in routers. Using large buffers at router leads two serious issues: first one is persistently full buffer problem, commonly known as Buffer-Bloat [1] and the second one is unpredictable behaviour of TCP congestion control. Buffer-bloat imparts noticeable latency in the network, resulting in reduction in Quality of Service which clearly affects the consumers at Internet edge. Uncoordinated and unmanaged buffers often results in unpredictable behaviour of networks. Applications like Telnet, Web browsing suffers huge latency. One of the major objectives of congestion control is to reduce the packet drop at router. One obvious solution is to having large size buffer area at router end to minimize packet drop. However, packet drop event is essential for well behaved functioning of source based congestion control policy like TCP.TCP relies on timely congestion notification to adjust its transmission rate to the available bandwidth [2]. Buffer-bloat means that new arriving packets are continued to be buffered, instead of dropped due to large buffer size. It causes the increased queue size at the bottlenecks. During the packet drop situation, more packets may be dropped which causes decrease in the transmission rates of TCP senders. In particular, if several TCP applications are transmitting over the same congestion point, all flows will observe drops at the same time. Therefore, all transmission rates would be reduced simultaneously. This phenomenon is called TCP global synchronization [3]. An effective solution to the above mentioned problems lies in queue management techniques which are responsible for monitoring and managing queue size before it increases or reduces by an alarming rate [4]. Random Early Detection (RED) [3] is the most widely implemented AQM in routers. The behaviour of RED is such that it works by setting four essential parameters, namely: minimum threshold, maximum threshold, queue weight factor and maximum drop probability. These parameters have to be precisely adjusted depending on network scenario, increasing the overhead on network administrator. Hence, an AQM like CoDel (Controlled Delay Management)[5] is recently being adopted which is parameter less, eliminating the need of parameter setup for network administrator. CoDel is one of the most simple and efficient AQM algorithm. It helps to solve Buffer-Bloat problem in wired scenarios. Although CoDel performs well in most of the situations, its link utilization degrades with longer RTT values (greater than 100ms) [4]. While a variant of CoDel naming sfqCoDel [6] achieves almost the maximum utilization with varying RTT values. sfqCoDel maintain the fairness among different flows if bandwidth requirement are

same for each sources, and all sources are having same packet sizes. In real scenario bandwidth requirement and packet sizes of different sources may vary and sfqCoDel becomes unable to maintain fairness among different flows in such situation.

We proposed a hybrid router based mechanism '*nfqCoDel*' by incorporating the functionality of both, a packet scheduler and active queue manager in a single module called 'Fair Active Queue Manager (FAQM)'. *nfqCoDel* actively manage the router queue length with an objective to control the excessive delay experienced by source host and to maintain the weighted fairness among different hosts. Thus *nfqCoDel* solve the buffer-bloat issue while providing weighted fairness among different sources. We have implemented '*nfqCoDel*' in ns-2 [7] and performed simulation under various simulation scenario. Simulation results prove that '*nfqCoDel*' performs better than other AQM by providing maximum and stable throughput, stable average queue length, controlled delay at router buffer and fairness among competing traffic sources. The proposed solution *nfqCoDel* manages the router queue in a heterogeneous environment i.e. the sources having diverse traffic characteristics and variable QoS requirement. The main contributions of *nfqCoDel* are as follows:

- Manage the router queue length actively to avoid congestion.
- Control the excessive delay faced by traffic sources in case of large buffer at router, and
- Provide a weighted fairness among different traffic sources having diverse QoS requirements.

In the present work we analyze the interaction patterns of some recently proposed AQMs like CoDel, sfqCoDel and nfqCoDel with various TCPs like HTCP[8], COMPOUND[9] and CUBIC[10], designed for high speed wired network. We have considered the following issues related with TCP-AQM interaction:

- Whether the AQM algorithms designed by considering non high speed TCP variants working at source end, work well with the high speed TCP variants?
- How effectively a particular AQM will interact with a High speed TCP variant in terms of various performance parameters.

Solution to above mentioned issues may depend on the particular high speed TCP, particular AQM algorithm and particular network scenario used. We have considered three well-known AQMs, three TCP variants and two congestion scenarios and performed the simulation in all possible combinations. In every case four performance measures were observed: the average queue size, the throughput, the fairness and the packet loss ratio.

The paper is organized as follows: In Section II of this paper, a brief review of previous hybrid Router based congestion control approaches, especially for high speed Internet, has been mentioned. The Section III explains the Network Model under consideration. Section IV explains Why AQM alone is not sufficient for Router based Congestion Control? The section V of this paper the proposed hybrid router based approach for congestion control: nfqCoDel is explained with an activity diagram and Algorithm. In section VI, we performed a simulation based experimental evaluation and analysis of the proposed method. Finally section VII concludes the paper.

II. RELATED WORK

High speed networks have characteristics of high bandwidth, long queuing delay, and high burstiness which make it difficult to address issues such as fairness, low queuing delay and high link utilization [11]. Current high speed networks carry heterogeneous TCP flows which make it even more challenging to address these issues. AQM schemes have been fairly successful in addressing either fairness issues or large queuing delay but not both at the same time. Thus in addition to active queue management at router buffer, implementation of some flow scheduling approach at router queue also became necessary to impose fairness among different flows sharing the router buffer. Various scheduling and queue management algorithms are implemented to avoid congestion. Fair queuing (FQ), which can be construed as a packet approximation of generalized processor sharing (GPS), is a scheduling algorithm used by network schedulers, to allow flows in network to fairly share the link [12]. Pan et al. [13] have presented a lightweight active queue management design called "PIE" (Proportional Integral controller Enhanced) that can effectively control the average queuing latency to a target value. Simulation results, theoretical analysis, and Linux testbed results have shown that PIE can ensure low latency and achieve high link utilization under various congestion situations. Their design does not require per-packet timestamps, sfqCoDel [6], the first hybrid approach, was proposed as a variation of CoDel which drops packets intelligently by proactively dropping the one which occupies more bandwidth comparing to the rest, thus ensuring fair consumption of bandwidth by each packet. The goal of fair queuing is to give each distinct user of the network, a fair share of available bandwidth. Instead of keeping track of all active flows and their share of bandwidth, flows are hashed into a number of buckets, each of which has its own queue. These queues are served in a round-robin fashion using DRR++ approach [14], when packets are dequeued. L. Xue et al. [11] have proposed a new AQM scheme called Approximated-Fair and Controlled-Delay (AFCD) queuing for high speed networks with

following design goals: approximated fairness, controlled low queuing delay, high link utilization and simple implementation. The design of AFCD utilizes a novel synergistic approach by forming an alliance between approximated fair queuing and controlled delay queuing. It uses very small amount of state information in sending rate estimation of flows and makes drop decision based on a target delay of individual flow. Through experimental evaluation [11] in a 10Gbps high speed networking environment, AFCD meets its design goals by maintaining approximated fair share of bandwidth among flows and ensuring a controlled very low queuing delay with comparable link utilization. Hong et al. [15] have explored fairness and application performance capabilities of five packet scheduling disciplines: FCFS drop tail, Adaptive RED, CoDel, PIE, and DRR. They focused on downstream queuing in a DOCSIS cable environment. They observed that CoDel and PIE are quite effective at maintaining the target queue delay and consequently better able to address bufferbloat than DT and ARED. CoDel and PIE are more sensitive to TCP/RTT unfairness than DT and ARED. They pointed out addressing bufferbloat at the expense of fairness might prove problematic in emerging converged broadcast networks that must deal with net neutrality issues. Al-Saadi et al. [16] have proposed a prototype "FlowQueue-PIE" (FQ-PIE) implementation that combines FQ-CoDel's Flow Queuing with PIE's individual queue management. It is a recent hybrid scheduler-AQM scheme implemented in the FreeBSD operating system. They experimentally compare their implementations against the current Linux CoDel, FQ-CoDel and PIE and found that FQ-PIE provides low queueing delay and relatively fair capacity sharing between competing flows. Ko et al. [17] have proposed FDAQM, a fairness-aware delay-controlled AQM to satisfy all three key network performance metrics: end-to-end throughput, throughput fairness, and delay, simultaneously in the 802.11sbased MRMC WMN. The proposed FD-AQM achieves the goal by adopting a per-queue drop interval which is recalibrated according to the average round-trip time of flows, and a drop decision mechanism which considers average throughput as well as target delay. Through simulations, they have shown that FD-AQM is superior to other AQM schemes in terms of throughput and fairness. Jiang et al. [18] have proposed an efficient active queue management algorithm with *C*ontrolling *F*airness and *D*elay (*CFD*). Different from most other schemes, CFD can achieve the low queuing delay and inter-flow fairness simultaneously without the queue isolation. Extensive simulation results also proved the effectiveness and feasibility of CFD. Menth et al. [19] have proposed a new AQM mechanism CP-AQM based on the idea of congestion policing. They evaluated its performance for various networking scenarios and transport protocols, and illustrated the impact of its parameters. They simulated average queue length and utilization in the presence of tail-drop and CP-AQM for various configurations, networking scenarios and transport protocols on a 10 Mb/s link. CP-AQM keeps the average queue length very short in case of persistent overload through non-responsive traffic. With reasonable configuration it achieves also short average queue lengths for TCP traffic (New Reno and Cubic) and 100% utilization if multiple flows are transmitted. Armitage et al. [20] have experimentally emulated a congested home broadband service using FIFO, CoDel, PIE and FQ-{CoDel,PIE} queue management schemes to explore interactions between elastic, TCP-based bulk data transfers and traffic typical of low-rate, UDP-based interactive applications. Their results demonstrate that single-queue AQM schemes, such as CoDel and PIE, create significantly increased levels of packet loss for low-rate UDP traffic during periods of competition with elastic, TCP-based bulk data transfers. However, they also observed that FlowQueue variants, such as FQ-CoDel and FQ-PIE, provide significant protection for interactive traffic flows, allowing for almost zero packet loss during periods of competition with otherwise bandwidth hungry TCP flows. Ye et al. [21] have proposed a general framework to combat bufferbloat in multi-bottleneck networks. They first conduct an equilibrium analysis for a general multi-bottleneck TCP/AQM system and develop an algorithm to compute the equilibrium point. They present a case study to analyze the stability of the recently proposed Controlled Delay (CoDel) in multi-bottleneck networks and devise Self-tuning CoDel to improve the system stability and performance using the proposed framework. Extensive simulation results shows that Self-tuning CoDel effectively stabilizes queueing delay in multi-bottleneck scenarios, and thus contributes to combating bufferbloat. Ganesh Babu et al. [22] have compared the two methodologies ECN and QBER for reducing the congestion. Sunitha et al. [23] have proposed a nature inspired optimal path finding algorithm to mitigate congestion in WSNs.

It can be observed that very few hybrid solutions are available to solve congestion control issue. We extend the above contributions further by proposing an efficient hybrid approach for router based congestion control, 'nfqCoDel' with an objective to make a balance among throughput, delay and fairness.

III. WHY AQM ALONE IS NOT SUFFICIENT FOR ROUTER BASED CONGESTION CONTROL?

Network congestion is a major issue in high speed wired networks having diverse nature of traffic sources. As the number of flows increases the resource share of each flow decreases at the bottle-neck link. Due to lack of resources, the network suffers congestion which causes packet drops and excessive delay. The objective of congestion control is to achieve efficiency i.e. maximum utilization, minimum queue size and minimum packet drops while giving fair access to all the sources.

There are mainly two approaches to congestion control – source based congestion control and router based congestion control [24]. In source based congestion control, senders detect the congestion and react to it accordingly. TCP is an important example of this approach. When a packet is dropped, the sender assumes that congestion has occurred and reduces the sending rate. When a packet is successfully transmitted, senders increase their rate. The other approach is router based congestion control. The idea behind this is that since routers have more information about the state of the network, they can be useful in detecting congestion and should take part in the decision of congestion control. Routers actually measure the congestion level by comparing input traffic to capacity and by looking at the queue size; thus, they can send feedback as soon as they notice that the queue length is growing. Routers could also be used to give priorities to some sources as compared to others. An important example of router congestion control is Active Queue Management. AQM refers to a class of algorithms designed to provide improved queuing mechanisms for routers. These schemes are called active because they dynamically send signal for congestion to sources even before the queue overflows; explicitly, by marking packets or implicitly, by dropping packets.

Although Active Queue Management can improve the QoS parameters by regulating the queue length and preventing the congestion, it is not able to provide strict QoS guarantee to diverse traffic sources [25]. Packet Scheduling has an important role in congestion control, thus a Packet Scheduler is also implemented in the Router. Packet Scheduling is primarily used to decide which packet to send next from the router's output port to provide per flow bandwidth guarantee. However it doesn't have any mechanism to control the size of the queue.

These two router mechanisms: packet scheduling and queue management are used individually either for managing the queues or for maintaining the per flow bandwidth requirement of different flows. Packet Scheduling and packet dropping decisions are not orthogonal to each other thus both should be implemented simultaneously at the router Buffer. Thus we need a mechanism which can facilitate queue management and fair queuing simultaneously for effective congestion control. Here we try to answer the question that, why any single approach is not sufficient to provide quality service and we advocate that why AQM and Packet Scheduling should be considered together as a combined approach. sfqCoDel is the one hybrid router based approach for congestion control.

sfqCoDel uses DRR++ [14] scheduling algorithm for differentiating between good queue (queue containing traffic of low bandwidth sessions) and bad queue (queue containing traffic of hog sessions).After that it apply the CoDel queue management on the selected good queue. sfqCoDel stochastically map the packets into the buckets. Although sfqCoDel works better in some scenario but it may have following limitations:

- Due to stochastic nature of hashing, multiple flows may end up being hashed into the same bucket.
- Hash collision can be minimized by increasing the number of hash bucket. However large number of buckets will result in difficulty in managing a large number of queues and memory overhead.
- sfqCoDel mixes packets from multiple flows, so there is a chance of unfairness when one delay-sensitive and one FTP flow map in the same bucket.
- sfqCoDel uses DRR++ scheduling approach for selecting the flow to apply the CoDel for queue management. DRR++ is a round robin based packet scheduling algorithm. There are several inherent limitations of round robin scheduling approach as mentioned by academicians and researchers [26]. The three major issues with round robin based scheduling approaches are: unfairness, poor delay and burstiness.
- The CoDel queue management approach used by sfqCoDel also has several limitations. These limitations are related with the parameters sojourn-time and target. Merely depend on sojourn-time is not sufficient in some cases as it may give inappropriate results. Similarly assuming uniform target of 5ms irrespective of the flow QoS requirement is not preferable.

The above limitations of sfqCoDel lead the requirement of further research to mitigate the observed challenges. In our proposed approach, we made an attempt to overcome some of the above mentioned limitations of sfqCoDel. The focus of our approach is to minimize the excessive delay of delay sensitive traffic with an objective to improve the fairness among traffic with less memory overhead.

IV. PROBLEM STATEMENT

"To Design an efficient *queue manager* and *scheduler* at router buffer to control the congestion, minimize the excessive delay of delay sensitive traffic classes, while maintaining the fairness among different traffic classes."

In this work, we propose a modified hybrid approach for queue management and packet scheduling to enhance the buffer management at the router queue and to improve the fairness among multiple flows sharing the single bottleneck while controlling the excessive delay with less memory and implementation overhead. The design objective is divided into two dimensional goals as follows:

- To improve the Active Queue management by using a modified buffer management policy.

- To improve the Packet Scheduling for maintaining the fairness among multiple flows.
- We describe the proposed approach in next section.

V. PROPOSED APPROACH FOR SOLVING BUFFER-BLOAT AND NETWORK CONGESTION PROBLEM

We propose an approach for effective queue management at router buffer and named it nfqCoDel. nfqCoDel is an hybrid router based approach for congestion control, which combines the features of a packet scheduler and an active queue manager with objective to solve bufferbloat issue at router buffer while maintaining the fairness among different flows. To achieve the design goals we propose a hybrid approach which incorporates the queue management and packet scheduling policies both working simultaneously at the router end. For the purpose of design and implementation of our approach, we consider the following router model:

A. Router's Model

Network communication equipments include a switch (router) component for switching communication packets between input ports and output ports. There are three main functional modules within input queued router architecture shown in Figure 1. These are as follows:

a. Packet Classifier
b. Flow Scheduler
c. Active Queue Manager

These three modules are working in sequential pipelining order. We have demonstrated the working of these three functional modules with the help of the fig. 2.We have mentioned the functionality of each module in brief as follows:

a. *Packet Classifier*

When a packet arrives at the router it is stored in the input queue of router buffer. There are multiple output queues and each belonging to a particular service class. The Packet classifier module classifies the packets available in input queue into the multiple output queues based upon the flow_Id field of each packet.

b. *Flow Scheduler*

The role of flow scheduler is to select a flow (queue) for active queue management to provide per-flow bandwidth guarantees. Scheduler decides the order in which flows are selected for applying queue management. Thus the main objective of flow scheduler is to decide the flow service order of a flow while maintaining the QoS requirement of different traffics.

c. *Active Queue Manager*

An active queue manager monitors the flow (queue) selected by flow scheduler proactively, for measuring the incipient congestion within the network. It manages the queue length and decide what action has to be taken if queue length exceeds to some decided threshold value. In case of congestion in network, it sends some signals explicitly or implicitly to the source ends, so that they can either minimize their transmission speed or stop their transmission for some time.

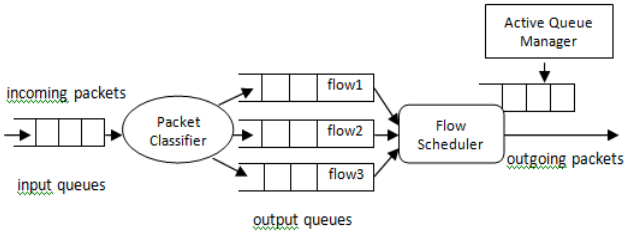

Figure. 1. Demonstration of 'Router Buffer Management' in proposed approach

Thus all the three components plays an important role in router based congestion control .A better result may be obtained if these functionalities are implemented in an efficient manner. We propose here, a possible modification in router mechanism, in order to enhance the queue management and flow scheduling algorithms working at the router. Descriptions of these modifications are given below.

B. The Proposed nfqCoDel AQM

nfqCoDel maintains multiple queues (q_1 q_2 ..... q_n) in router buffer and each queue q_i is associated with a weight parameter w_i. Let there are two queues in router buffer q_i and q_j with corresponding weight w_i and w_j. The parameter weight represent the relative priority of respective queue and (w_i>w_j) implies that queue 'q_i' is having higher priority as compared to queue q_j, that is traffic flow mapped in queue q_i will get preferential treatment over the traffic mapped in queue q_j. There may be many traffic sources and each traffic source forms a flow having distinct flow-Id f_i. In our proposed approach, a queue is reserved for a specific traffic flow so every queue contains traffic of specific flow with assigned flow-id. Packets arriving from a particular traffic source will map into their corresponding router queue based on its flow_id f_i. The nfqCoDel algorithm consists of two logical parts: the *scheduler* who selects the queue from which packet will be dequeued, and the *active queue manager* which works on each of the queues. Following are the description of each logical modules of nfqCoDel in detail:

a. Flow Scheduler
b. Active Queue Manager

a. Flow Scheduler :

The main objective of flow scheduler module is to select a queue (flow) for queue management while maintaining a

weighted fairness among different traffic flows. nfqCoDel uses a proposed fair queuing mechanism: 'NFQ' as flow scheduler.

b. Active Queue Manager

Active queue manager module performs queue management on the flow selected by queue scheduler. In our algorithm, we named this module mCoDel (Modified CoDel). Each queue is set up to have a separate set of mCoDel state variables.

c. nfqCoDel Parameters :

This section explains the parameters used in nfqCoDel as follows:

interval The _interval_ parameter has the same semantics as CoDel and is used to ensure that the measured minimum delay does not become too stale.

target The _target_ parameter has the same semantics as CoDel. It is the acceptable minimum standing/persistent queue delay for each nfqCoDel Queue.

new_interval_ and new_target_ It has been observed that different traffic flows are having different QoS requirements in terms of throughput and delay. Therefore it is not appropriate to consider a fixed value for interval_ and target_ parameters. We suggest redefining these parameters and making it adaptive to the different types of traffics so that it can maintain QoS requirements accordingly. It would be better to redefine these parameters, which is adaptive in nature. The proposed interval_ and target_ parameters are as follows:

new_interval_= interval_ + (1/flow[i].wi)

*new_target_= target_+ (0.0075) * flow[i].wi*

C. Activity diagram of nfqCoDel AQM

This section describes the operation of the nfqCoDel scheduler and active queue manager (mCoDel). nfqCoDel as a router mechanism consists of two functional modules one for enqueue and the other for dequeue operations. Figure 2 exhibits the activity diagram of nfqCoDel, which clearly depict the relationship between enqueue and dequeue functions.The mentioned steps in activity diagrams are described in following subsections:

1) Enqueue Function

The packet enqueue mechanism consists of three stages:

- Classification of packets into respective sub-queues.
- Time-stamping packet arrival time for each incoming packet.
- Optionally drop a packet when the total number of enqueued packets exceeds the given threshold.

When a packet is enqueued, it is first classified based on its header information and placed into the appropriate sub-queue. By default, this is done by checking the fid fileld in packet header, and then map the packet in a queue reserved for flow_Id=fid. Once the packet has been successfully placed into respective sub-queue, it is handed over to the mCoDel algorithm for time-stamping. mCoDel records the enqueue time of every packet as a timestamp in packet header. It is then added to the tail of the selected queue, and the queue's byte count is updated by the packet size.

2) Dequeue Function

The major portion of nfqCoDel is cantered at packet dequeue time. Dequeue function consists of two steps: selecting a queue from which a packet can be dequeued and finally dequeue the packet according to proposed criteria: mCoDel.

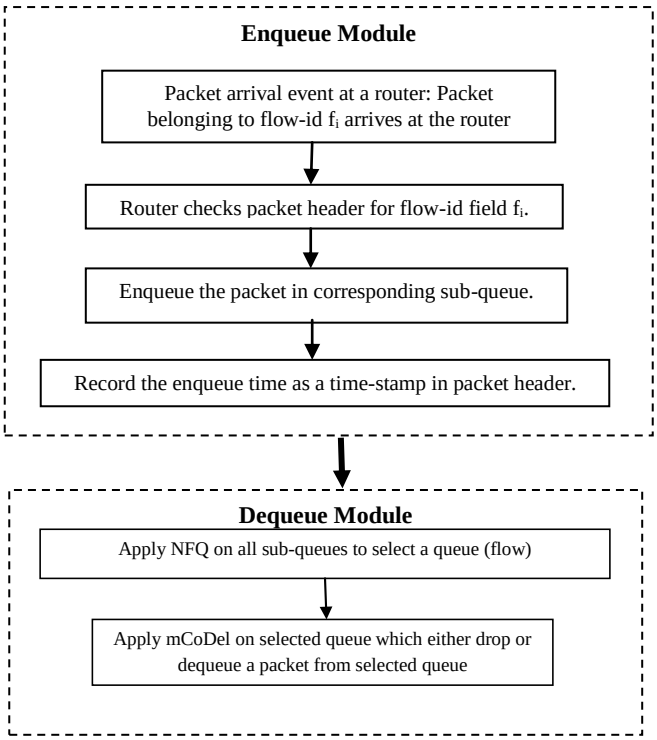

Figure 2. Activity Diagram for nfqCoDel Algorithm

Step 1: In its first step, the dequeue function consists of following sub steps:

i. The scheduler first calculates the time-stamp

q_i->*timestamp* for each nonempty sub-queue q_i with weight w_i by considering the packet at head (HOL) of corresponding sub-queue, using the formula:

$$q_i\text{->}timestamp = pkt\_size\ (HOL\ packet\ at\ q_i)/w_i$$

ii. Find the minimum timestamp TS_{min} among all non empty sub queue as:

$$TS_{min} = MIN(q_i\text{->}timestamp)$$

iii. For each non empty sub queue :

a. Flow q_i having timestamp q_i->*timestamp greater than* TS_{min} update the timestamp by using the formula:

$$q_i\text{->}timestamp = q_i\text{->}timestamp - TS_{min}$$

b. A Flow q_i having timestamp q_i->*timestamp less than or equal to* TS_{min} will be selected in round robin manner, to apply mCoDel in next step.

Step 2: In its second step, the dequeue function apply the mCoDel algorithm on queue q_i , selected in previous step. mCoDel calculate the sojourn time of HOL packet of selected flow, and compare it with the target_, if sojourn time exceed target_ for interval_ amount of duration, it may discard one or more packets from the head of that queue, otherwise return the packet that should be dequeued.

VI. EXPERIMENTAL EVALUATION

In this section, we evaluate nfqCoDel carefully, and compare nfqCoDel to other related AQM schemes such as RED, CoDel, and sfqCoDel. We evaluate the interaction pattern of 'nfqCoDel' with various high speed TCPs carefully. We conduct various experiments to simulate various scenarios in high speed networks.

A. Simulation Setup

The network simulator ns-2.35 [7] is used to conduct a comparative analysis among RED, nfqCoDel, sfqCoDel and CoDel AQM mechanisms. A single duplex bottleneck topology is used for all simulations. Simulation parameters are depicted in Table 1. The bottleneck bandwidth is set to 622Mbps and bottleneck round trip delay set to 48ms. Non bottleneck bandwidth of 1Gbps with round trip delay set to 1ms. Bottleneck buffer size is set to 8xBDP (bandwidth-delay product).Two types of traffic included are TCP and UDP. Based on recommended values [5], the values of interval and target queue delay for CoDel are set to 100ms and 5ms respectively.

Table 1. Simulation parameters description

Network type	High speed Wide Area Network
Simulation time	100 sec.
Link bandwidth for incoming traffic	622 Mbps
Bottleneck Link bandwidth	1 Gbps
RTT delay	100 ms
Queue buffer size	8*bdp (bandwidth delay product)
TCP traffic description	Packet size= 1000 bytes, 500 bytes
	Traffic type= FTP
UDP traffic description	Packet size= 1000 bytes
	Traffic type= CBR with rate 2Mbps

The proposed approach is an attempt to fulfil the diverse demand of different traffic flows sharing a common bottleneck link. There are different possible simulations scenarios by considering different combinations of traffic sources with diverse QoS requirements. We have performed various experiments by considering following simulation scenarios:

1) All flows are TCP with equal QoS requirement

Under this scenario all traffic flows are assumed to be TCP type. We have performed the simulation by considering that all TCPs are having equal QoS requirements in terms of throughput and delay. All flows are assigned equal weight parameter.

2) All flows are TCP with different QoS requirement

In this scenario all traffic flows are assumed to be TCP type. We have performed the simulation by considering that Sources are having different QoS requirements in terms of throughput and delay. Different flows are assigned weight parameter based upon its priority level; a flow having higher weight represents its weighted share in available network resources.

3) Mixed flows(TCP+UDP)

Under this scenario traffic flows are assumed to be of mixed type i.e both TCP and UDP. We have performed the simulation for testing the unfairness issue between TCP and UDP traffic. All the simulations are performed by considering equal weights of different mixed flows.

4) All flows are TCP having variable packet sizes

Here we considered that all the flows are TCP type but having different packet sizes. The main motivation behind this simulation scenario is to compare the performance of different AQM algorithms and the proposed algorithm in terms of fairness.

B. Performance Metrics

The major performance parameters considered for analysis include: link utilization of bottleneck link, queue size at the congested router, packet drop rate and fairness. A high speed TCP variant used is CUBIC TCP for all the simulations.

C. Result and Analysis

In this section we explain the result and analysis of each scenario as follows:

1) All flows are TCP with equal QoS requirements

We considered three pairs of TCP flows sharing the common medium and performed the simulation by considering various AQM at bottleneck router, and found the following results: Figure 3 exhibits that all AQM methods are having a stable queue length. sfqCoDel is having lowest mean queue length while nfqCoDel is quite similar to RED in managing queue length.

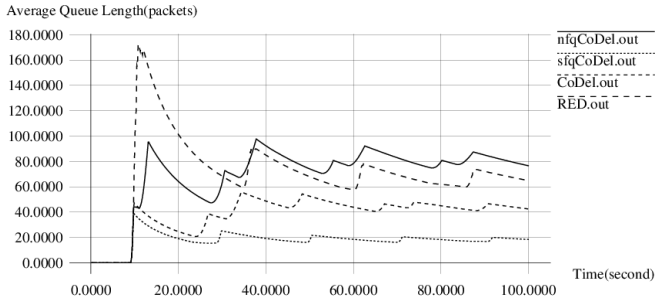

Figure 3. Average queue length of different AQM

Figure 4(a) to Figure 4(e) represents the throughput and fairness characteristics of RED, CoDel, sfqCoDel and nfqCoDel respectively. From these figures it is clear that RED exhibit unstable throughput because of global synchronization issue, still there is fairness among different sources. CoDel is an improvement over RED, as it solves global synchronization problem that results in increased throughput. But there is a prevalent unfairness among different sources; means CoDel is unable to provide fairness. sfqCoDel solves the unfairness issue of CoDel, but still there are instability in throughput as there are more oscillations in throughput graph. The proposed approach, nfqCoDel is also able to solve the unfairness issue of CoDel, just like sfqCoDel, along with that it is able to provide a more stable throughput as compared to sfqCoDel.

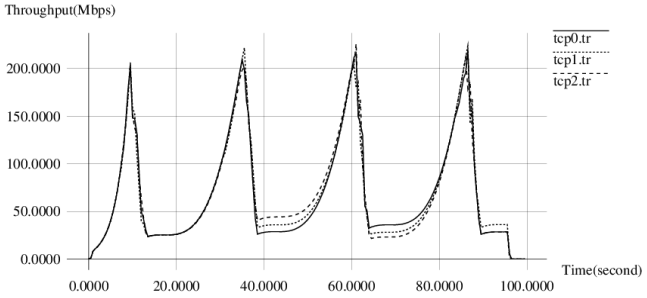

Figure 4(a). Throughput of TCPs under RED AQM

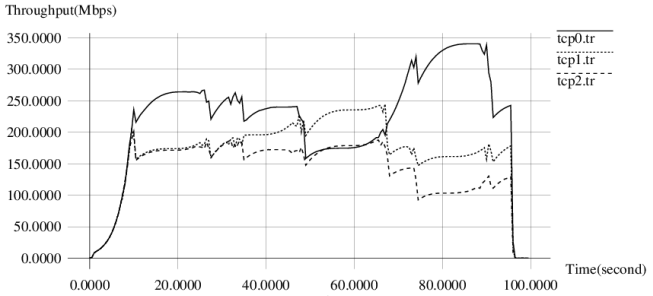

Figure 4(b). Throughput of TCPs under CoDel AQM

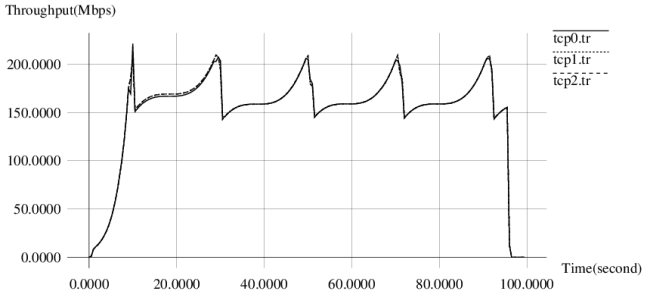

Figure 4(c). Throughput of TCPs under sfqCoDel AQM

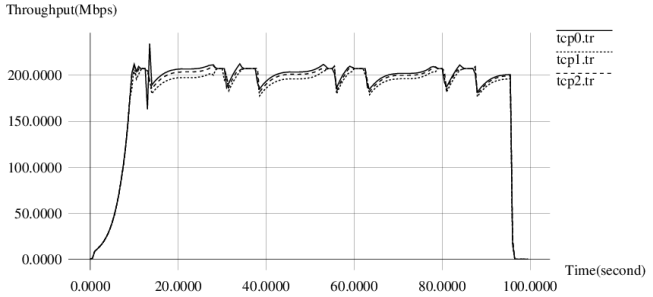

Figure 4(d). Throughput of TCPs under nfqCoDel AQM

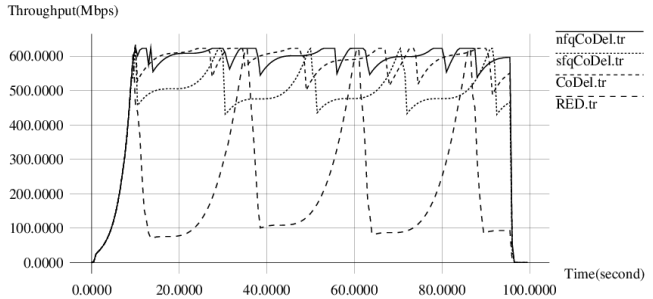

Figure 4(e). Total Throughput of TCP flows under different AQMs

Table 2: Performance parameters for TCP

AQM →	RED	CoDel	sfqCoDel	nfqCoDel
Througput	224.661	550.52	472.167	559.46
Link utilization	36.11%	88.50%	75.91%	89.94%
Fairness	0.9999	0.9657	0.9999	0.9998
Loss Rate % (packet count)	0.7731 (19894)	0.000527 (33)	0.00135 (73)	0.00037 (24)
Delay	0.0527	0.0507	0.0504	0.0512

Table 2. Further conclude the overall performance of RED, CoDel, sfqCoDel and nfqCoDel in terms of different performance criterion. It can be observed from above table that RED is having poor performance in terms of throughput, link utilization and packet loss rate due to self-synchronization problem. As far as fairness and delay is concerned they are better but there is less significance of these two parameters if RED is not able to perform better in terms of throughput, link utilization and packet loss rate. CoDel performs quite better than RED in terms of various performance criterions except the unfairness issue. sfqCoDel is an improvement over CoDel to solve the unfairness among different TCP flows by providing the best possible fairness, but on the cost of decreased throughput and link utilization. nfqCoDel performance is better in terms of various performance parameters as compared to sfqCoDel.

2) All flows are TCP with different QoS requirement

In this simulation scenario we have considered three TCP flows: TCP1, TCP2 and TCP3, with different QoS requirements in terms of throughput and delay. These flows are assigned different weight in proportion of their QoS requirements. Here we assumed the weight ratio 1:2:4 for the flows TCP1, TCP2 and TCP3 respectively, all are sending packets with equal speed of 1Gbps. Our major objective of this scenario is to provide weighted performance goals to different TCPs according to their weights. We have performed this simulation by applying two variations of nfqCoDel as:

nfqCoDel with fixed target_ and interval_ parameters

First we assumed a fixed value for parameters interval_ and target_ for different TCP flows, as assumed in CoDel and sfqCoDel. The simulation outcomes are displayed in table 3(a).It can be observed that it is not possible to provide throughput to three flows in proportion of their weight. As third flow TCP3 is getting fewer throughputs.

- *nfqCoDel with adaptive target_ and interval_ parameters*

Next we assumed adaptive value for parameters interval_ and target_ for different TCP flows, as proposed in previous section. The simulation outcomes are displayed in table 3(b).

Table 3(a). Performance with fixed parameters

Flow ID	Flow weight	Throughput	Delay
TCP1	1	106.91	0.0854
TCP2	2	207.72	0.0865
TCP3	4	264.52	0.0504
Total Throughput/Delay		579.15	0.0741
Link utilization		93.11%	
Loss rate		1035 packets	

Table 3(b). Performance with adaptive parameters

Flow ID	Flow weight	Throughput	Delay
TCP1	1	86.51	0.110
TCP2	2	166.96	0.094
TCP3	4	325.71	0.069
Total Throughput/Delay		579.17	.0915
Link utilization		93.125%	
Loss rate		918 packets	

These results clearly depict that three TCP flows are getting throughput in proportion of their weight, along with that the flows are experiencing delay in inverse proportion of their weights, as the flow having higher weight is having minimal delay as compared to flow having lower weight.

Thus we can conclude that, use of adaptive interval_ and target_ parameters are beneficial toward achieving the objective of weighted fairness among different TCP connections, which was not possible to achieve by using CoDel or sfqCoDel as an AQM at router buffer.

3) Mixed flows(TCP+UDP)

In these scenario two flows, one TCP Cubic and one UDP are simulated simultaneously. The UDP flow is a non-responsive flow sending at a speed of 500Mbps. Figure 5(a) represents the throughput of TCP and UDP flows in presence of different AQM at router. RED and CoDel are not able to solve the unfairness issue of TCP in presence of UDP flow. UDP is nonresponsive against the congestion signal sent by the router and destination host, thus it grab all the available bandwidth by sending packets in its usual speed, on the other hand TCP respond with congestion signal by minimizing its transmission speed. sfqCoDel tries to solve this unfairness issue between TCP and UDP, as it improves the fairness between both the flows. The proposed approach nfqCoDel is a further improvement over sfqCoDel as it tries to minimize the unfairness between TCP and UDP.

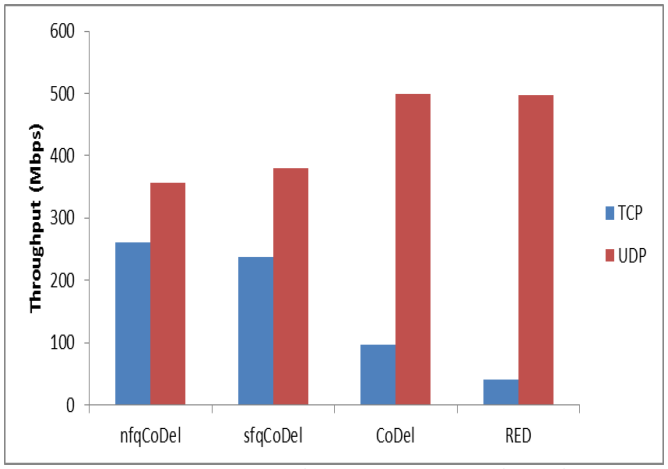

Figure 5(a). Throughput performance in presence of UDP flows

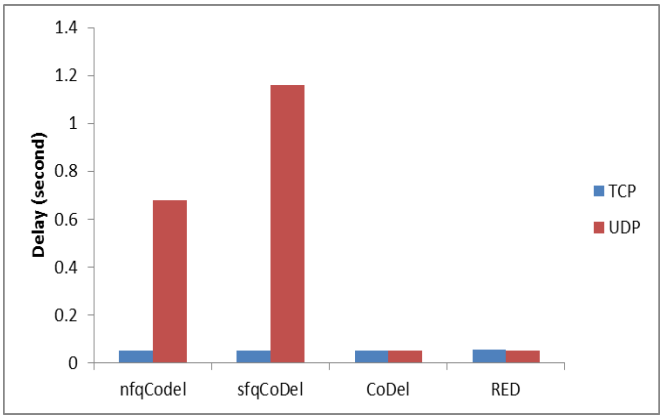

Figure 5(b). Delay performance in presence of UDP flows

Figure 5(b) represents the queuing delay characteristics of TCP and UDP flows in presence of different AQM at router. RED and CoDel are still able to control the queuing delay of TCP and UDP flows to a minimum level, on the other hand sfqCoDel shows a poor delay performance for UDP flows, but able to control the delay of TCP flows. Thus sfqCoDel is able to solve the unfairness issue between TCP and UDP on the cost of worst queuing delay of UDP flow. Our proposed approach nfqCoDel is able to minimize the queuing delay of UDP flow while still maintaining a minimal delay for TCP flows.

Table 4(a) and Table 4(b) further conclude the overall performance of RED, CoDel, sfqCoDel and nfqCoDel in terms of throughput, fairness and queuing delay. From these tables it can be inferred that the proposed method nfqCoDel is able to cope with unfairness issue between TCP and nonresponsive UDP flows, in a better manner, while maintaining a minimum delay level for TCP and a controlled delay for UDP flows.

Table 4(a).Throughput/Fairness in presence of UDP

AQM→	nfqCoDel	sfqCoDel	CoDel	RED
TCP	260.318	237.81	97.06	41.24
UDP	356.07	379.36	499.72	497.71
Fairness	0.976	0.95	0.687	0.582

Table 4(b).Delay in presence of UDP

AQM→	nfqCoDel	sfqCoDel	CoDel	RED
TCP	0.0525	0.0504	0.0507	0.0536
UDP	0.6777	1.16	0.0506	0.0509
Total delay	0.365	0.605	0.0506	0.0523

4) All flows are TCP having variable packet sizes

We considered this simulation scenario to study the unfairness introduced among different TCP flows due to variable packet length. Here we simulated two TCP flows tcp0 and tcp1 having different packet length 1000 byte and 500 byte respectively. We found the following observations:
Figure 6(a) to Figure 6(d) represents the throughput and fairness characteristics of RED, CoDel, sfqCoDel and nfqCoDel respectively when two TCP sources tcp0 and tcp1 are having different packet sizes. RED exhibits unstable throughput because of global synchronization issue, there is unfairness among different sources. Both CoDel and sfqCodel solves global synchronization problem that results in increased throughput. But there is a prevalent unfairness among different sources; means CoDel and sfqCoDel are unable to provide fairness between two flows having different packet sizes. The proposed approach, nfqCoDel is able to solve the unfairness issue due to variable packet lenght.

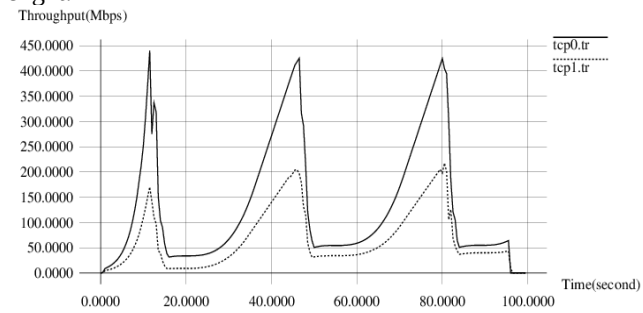

Figure 6(a). Throughput of TCPs under RED AQM

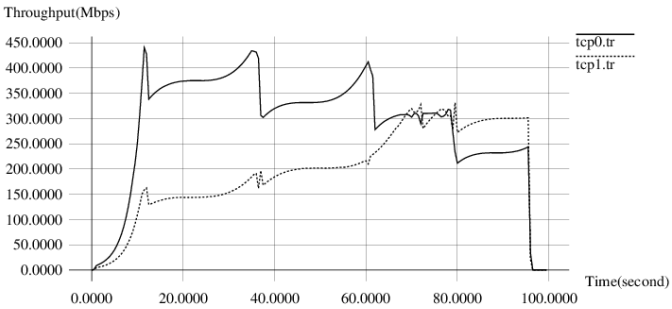

Figure 6(b). Throughput of TCPs under CoDel AQM

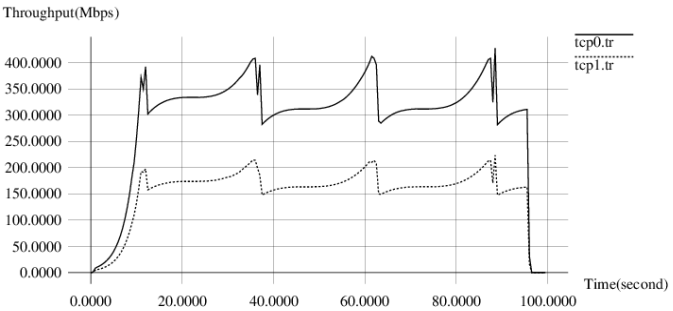

Figure 6(c). Throughput of TCPs under sfqCoDel AQM

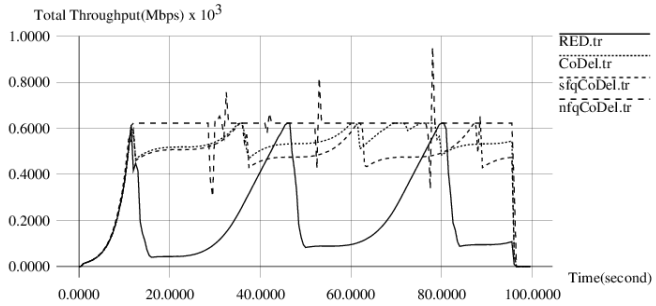

Figure 6(d). Throughput of TCPs under nfqCoDel AQM

Figure 6(e) represents the performance of diiferent AQM with respect to total throughput. We observed that the proposed approach nfqCoDel performs better as compared to other approaches. nfqCoDel is able to provide maximum and stable total throughput.

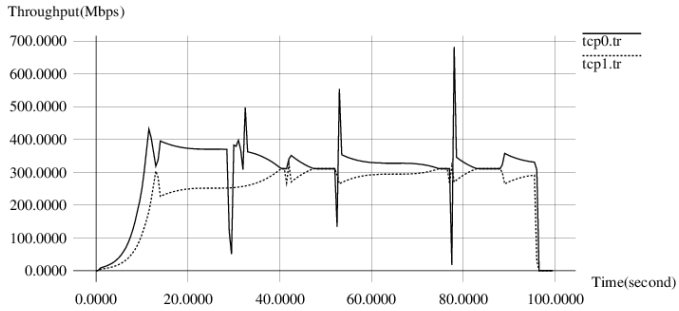

Figure 6(e). Total Throughput of TCP flows under different AQMs

Table 5: Performance of AQM approaches (variable packet length)

AQM →	RED	CoDel	sfqCoDel	nfqCoDel
Throughput	214.96	500.55	462.34	**559.46**
Link utilization	34.56%	80.47%	74.33%	**89.94%**
Fairness	0.8989	0.9637	0.9104	**0.9998**
Loss Rate % (packet count)	0.59009 (15224)	0.00031 (19)	0.00058 (32)	0.00037 (24)
Delay for flow 1	0.05218	0.05054	0.05038	**0.42979**
Delay for flow 2	0.05179	0.05052	0.05037	0.05072
Average Delay	0.05198	0.05053	0.05038	0.24025

Table 5. Further conclude the overall performance of RED, CoDel, sfqCoDel and nfqCoDel in terms of different performance criterion while considering variable packet length. It can be observed from above table that RED is having poor performance in terms of throughput, link utilization and packet loss rate due to self-synchronization problem. As far as fairness and delay is concerned they are better but there is less significance of these two parameters if RED is not able to perform better in terms of throughput, link utilization and packet loss rate. CoDel performs quite better than RED in terms of various performances. sfqCoDel is better than RED but its performance deteriorates than CoDel with respect to throughput, link utilization and fairness, when traffic sources are using packets of different sizes . The TCP flow tcp0 grabs more bandwidth as it is having larger packet size as compared to second flow tcp1 which leads to unfairness between two flows. The proposed approach nfqCoDel performs better in terms of various performance parameters as compared to sfqCoDel. nfqCoDel is able to provide highest throughput, link utilization and fairness in spite of traffic sources are having different packet lengths. Loss rate of nfqCoDel is also comparable to sfqCoDel. The average end-to-end delay of proposed approach nfqCoDel is heighest among all the AQMs; this is just because of first TCP flow which is having larger packet length. The packet from first flow tcp0 is bigger in size will wait more in corresponding sub-queue to maintain the fairness with the second flow having small packet size which leads more delay for the first flow.

VII. CONCLUSION

We proposed nfqCoDel, an efficient approach for queue management in high speed wired network with an objective to provide weighted fairness and low queuing delay in a controlled manner. nfqCoDel uses a hybrid approach of flow scheduling and queue management at router buffer to achieve the mentioned goals. We evaluated the performance of nfqCoDel in a 1Gbps high speed networking environment using network simulator ns-2 by considering various simulation scenarios. Simulation results exhibit that overall performance of nfqCoDel is better among its peers in various scenarios. In terms of fairness, nfqCoDel performs as well as sfqCoDel. nfqCoDel is capable to provide weighted throughput and fairness, by providing weights to different traffic flow. In terms of delay, nfqCoDel is able to control queuing delay to be low, which is comparable with CoDel and sfqCoDel. nfqCoDel gets a higher throughput and low loss rate performance as compared to other AQM schemes.

REFERENCES

[1] J. Gettys, K. Nichols, "Bufferbloat: dark buffers in the internet". Communications of the ACM, Vol.55 (1), pp: 57-65, 2012.

[2] H. To, T M Thi, W Hwang, "Cascade Probability Control to Mitigate Bufferbloat under Multiple Real-World TCP Stacks" in Special Issue of Control Problem of Nonlinear Systems with Applications, Mathematical Problems in Engineering, vol.2015, 2015.

[3] S. Floyd, V. Jacobson, "Random early detection gateways for congestion avoidance". IEEE/ACM Transactions on Networking, Vol.1 (4), pp: 397–413, 1993.

[4] V. P. Rao, Mohit P. Tahiliani, Udaya Kumar K. Shenoy, "Analysis of sfqCoDel for Active Queue Management", Applications of Digital Information and Web Technologies (ICADIWT), 2014.

[5] K. Nichols, V. Jacobson, "Controlling queue delay". Communications of the ACM, Vol.55 (7), pp: 42-50, 2012.

[6] T. Hoeiland-Joergensen et al., "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm" RFC 8290, January 2018.

[7] Ns-2 Network Simulator. (n. d.). Retrieved from:

⟨http://www.isi.edu/nsnam/ns/⟩.
[8] D. Leith, R. Shorten, "H-TCP: TCP for high-speed and long distance networks". In: Proceedings of the PFLDnet, 2004.
[9] K. Tan, J. Song, "Compound TCP: a scalable and TCP-friendly congestion control for high-speed networks". In Proc. 4th International Workshop on Protocols for FAST Long-Distance Networks, 2006.
[10] I. Rhee, L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant". SIGOPS Operating System Review, Vol.42 (5), pp: 64–74, 2008.
[11] L. Xue et al., "Afcd: An approximated-fair and controlled-delay queuing for high speed networks". In proceeding of 22nd International Conference on Computer Communications and Networks (ICCCN), pp: 1-7. IEEE, 2013.
[12] C. Vijith C and M. Azath, "Survey on CHOKe AQM Family", International Journal of Computer Science International Journal of Computer Sciences and Engineering, Volume-2, Issue-11, pp- 81-85, 2014.
[13] R. Pan et al., "PIE: a lightweight control scheme to address the bufferbloat problem", in Proc.of the IEEE International Conference on High Performance Switching and Routing (HPSR), Jul. 2013.
[14] M. H. MacGregor, W. Shi, "Deficits for Bursty Latency critical Flows: DRR++", Proc. IEEE ICON '00, Singapore, pp: 287-293, 2000.
[15] G. Hong, J. Martin, J. M. Westall, "On fairness and application performance of active queue management in broadband cable networks". Computer Network. 91, 390–406, 2015.
[16] R. Al-Saadi, G. Armitage, "Dummynet AQM v0.2 – CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 160418A, 18 April 2016.
[17] N. S. Ko, M. H. Kim, H. S. Park, "FD-AQM: Fairness-Aware Delay-Controlled Active Queue Management in 802.11s-Based Multi-Radio Multi-Channel Wireless Mesh Networks", IEEE Communications Letters, Vol. 19, No. 5, pp: 839-842, 2015.
[18] X. Jiang, G. Jin, "CFD: an efficient active queue management algorithm with CFD", Electronics Letters IET, Vol. 52, Issue. 24, pp: 2015-2017, 2016.
[19] M. Menth, S. Veith, "Active Queue Management Based on Congestion Policing (CP-AQM)". In Proceeding of International Conference on Measurement, Modelling and Evaluation of Computing Systems. LNCS, Vol. 10740, pp: 173-187, 2018.
[20] G. Armitage, R. Collom, "Benefits of FlowQueue-based Active Queue Management for Interactive Online Games", In Proceeding of 26th International Conference on Computer Communications and Networks (ICCCN 2017), Vancouver, BC, Canada, 2017.
[21] J. Ye, K. C. Leung and V. O. K. Li, S. H. Low,"Combating Bufferbloat in Multi-Bottleneck Networks: Equilibrium, Stability, and Algorithms", Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 2018-001, 2018.
[22] V. Ganesh Babu and K.Saraswathi, "Packet Mark Methodology for Reduce the Congestion", International Journal of Scientific Research in Computer Science and Engineering, Vol. 1, pp: 39-42, 2013.
[23] G.P. Sunitha , B.P. Vijay Kumar , S.M. Dilip Kumar, "A Nature Inspired Optimal Path Finding Algorithm to Mitigate Congestion in WSNs", International Journal of Scientific Research in Network Security and Communication, Volume-6, Issue-3, June 2018.
[24] V. Kushwaha , R. Gupta, "Congestion control for high-speed wired network: a systematic literature review". Journal of Network and Computer Applications, Vol.45, pp: 62–78, 2014.
[25] S. Ryu, C. Rump, C. Qiao, "Advances in active queue management (AQM) based TCP congestion control". Telecommunication Systems, Vol. 25(3–4), pp: 317–351, 2004.
[26] S. Ramabhadran,J. Pasquale, "The Stratified Round Robin Scheduler: Design, Analysis, and Implementation", IEEE/ACM Transactions on Networking, Vol. 14(6), pp: 1362-1373, 2006.

Authors Profile

Vandana Kushwaha is working as an Assistant Professor in Department of Computer Science, Institute of Science, Banaras Hindu University, Varanasi (India). She has 10 years of teaching experience of UG and PG students. She has done her Ph.D. in Computer Science at the Department of Computer Science, Institute of Science, Banaras Hindu University, Varanasi (India).Her research interests include High Speed Network, Wireless Sensor Network, Network algorithms and Congestion control protocols. She has 6 papers in international Journals and one paper in international conference. She has 1 book chapter (from CRC Press) in her credit.

Ratneshwer is working as an Assistant Professor in School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi,, India. He has done his Ph.D. In Component Based Software Engineering at Department of Computer Science and Engineering, Indian Institute of Technology, Banaras Hindu University (IIT-BHU), Varanasi (India). He has 11 years of teaching experience of UG and PG students. He is currently working on Computer networks and software engineering. He has 22 papers in international journals and 16 papers in international/national conference proceedings. He has 1 book chapter (from IGI Global Publisher) and one monologue (from LAP Germany) in his credit.

