Analysis of SfqCoDel to Reduce the Congestion and Delay Due To Excessive Network Traffic

Sneha Y, Vinita Singh, Zaiba H Jamadar, Toshitha K Uttaiah, Dr. Nalini Niranjan

Abstract— **With the increase in network traffic, a better and fair queue management is necessary to control dropping of packets inefficiently. This includes a better buffer and queue management which can be achieved by using the AQM technique. The various variants of AQM (Active Queue Management) are existent which can be improvised according to the scenario at hand to control the ever increasing traffic problems. The AQM technique which has been in use is the CoDel [1] algorithm which is a parameter-less and dynamic AQM variant. As a variant of this, the SfqCoDel [1] (Stochastic Fair Queue) has been proposed to overcome the backdrops of CoDel [1] and to drop the packets in a fair manner. The SfqCoDel [1] drops the packets in a network by comparing the larger bandwidth utilized by the packets queued in the network unlike CoDel [1] which drops the packets by calculating a particular timestamp. The analysis and comparison of CoDel [1] and SfqCoDel [1] shows that SfqCoDel [1] is better than CoDel [1] in terms of error rate, throughput etc.**

I. Introduction

The internet has grown over the years as the number of people adapting to the growing network scenario have also increased. This has led to increase in the problems regarding buffer size and inadequate queue length to accommodate the growing communication in the network. The Active Queue Management is a technique which addresses the BufferBloat [2] problem and even reduces the latency suffered due to network congestion. BufferBloat [2] problem refers to the full buffer problem which leads to network congestion and delay. The variant of AQM being CoDel [1] has been implemented in various platforms as it handles the queue in an efficient manner. It considers the timestamp attached to the header of the packet. The proposed system consists of the SfqCoDel that considers the bandwidth to bring about fair queue management. The comparison of the existing and proposed algorithms leads to a conclusion that the proposed algorithm i.e., the SfqCoDel is better.

Manuscript received March 21, 2016

Sneha Y, Final Year Student, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore
Vinita Singh, Final Year Student, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore
Zaiba H Jamadar, Final Year Student, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore
Toshitha K Uttaiah, Final Year Student, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore
Dr. Nalini Niranjan, Professor, Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bangalore

II. WHY CHOOSE NS-2

With the rise in new ideas and technology computer networks are subjected to diverse changes. This involves bringing about changes to the different protocols that already exist into newer ones. Due to these changes it gives rise to various complexities. Also with these different protocols and various other necessities it is hard to deploy them easily and is also very costly. The solution to all this comes with the feature called as network simulation. Network simulation is communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities using mathematical formulas, or actually capturing and playing back observations from a production network. A network simulator is software that predicts the behavior of a computer network. There are various types of network simulators that are available. They are ns-1, ns-2, ns-3. For the above proposed solution for generating SfqCoDel method the programming concept of Mat lab can also be used.

NS-1 was the first simulator that was developed. This is now neither developed nor is maintained. NS-2 is the version that we use for generating the solution. NS-2 uses a combination of both OTcl and C++ programming languages. Each of these alone has disadvantages but when put together does the magic. For efficiency reason, NS-2 [3] separates control path implementations from the data path implementation.

On the other hand we have mat lab which is it allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python. Mat lab is an interpreted language. The disadvantage with this form of programming language is that here is its execution speed. With an interpreted language, the computer running the program has to analyse and interpret the code which is done by the interpreter before it can be executed, resulting in slower processing performance. Thus looking at these advantages of NS-2 we can conclude that NS-2 is the best option that we could use.

III. EXISTING SYSTEM: CODEL

In the living scenario, we have control-delay (CoDel) which is a simple yet efficient AQM (active queue management) algorithm. CoDel algorithm is dynamic (parameter-less), controls delay and adapts according to situation. CoDel's algorithm is free of the queue metrics. Rather than measuring queue size in bytes or packets, CoDel uses the packet-source-journey time through the queue. Actual delay experienced by each packet is independent of link rate, boosts the performance of buffer, and is directly related to the user-friendly performance.

IV. PROPOSED SYSTEM: SFQCODEL

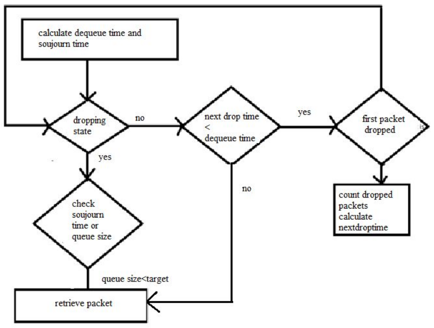
SfqCoDel (Stochastic Fair Queuing Controlled Delay) is queuing technique that collaborates fairness with the CoDel AQM scheme. SfqCoDel uses a stochastic model to distinguish incoming packets into different flows and is used to provide a equal share of the bandwidth to all the flows along the queue.

The SfqCoDel algorithm can be divided into two logical events: the scheduler selects which queue to drop a packet from, and the control-delay (CoDel) AQM which works on each of the queues. On dropping a packet, SfqCoDel selects a queue from which to dequeue by a round-robin-scheduling scheme, in which each queue is allowed to dequeue up to a flexible quantum of bytes for each cycle.

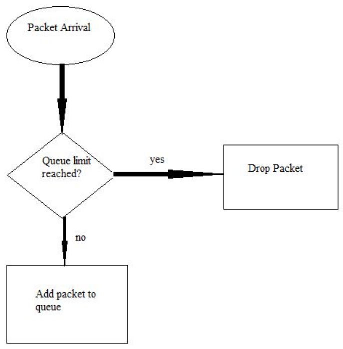
SfqCoDel's (round-robin scheduling) scheduler is byte-based, employing a definite and defined round-robin mechanism between queues. This works by keeping track of the current byte defined by each queue. This definite amount is initialised to the flexible quantum; each time a queue gets a drop opportunity, it gets to drop packets, decreasing the defined amount by the packet size for each packet, until the quantum runs into the negative, at which point it is increased by one quantum, and the dequeue opportunity ends.

V. FLOWCHART FOR CODEL

1. On arrival of packet (enqueue packet)


calculate dequeue time and soujourn time
dropping state
no
next drop time < dequeue time
yes
first packet dropped
yes
no
check soujourn time or queue size
count dropped packets calculate nextdroptime
queue size<target
retrieve packet

2. Departure of packet (Dequeue Packet)



VI. ALGORITHM FOR EXISTING AND PROPOSED SYTEMS

```
On arrival of every packet:
if current_queue_size < queue_limit then
    Enqueue the packet
    Attach a timestamp in packet header
end
else
    Drop the Packet
end
On departure of every packet:
dequeue_time = timestamp for dequeue time
soujourn_time = dequeue_time – enqueue_time
if inside the dropping state then
    if soujourn_time < target or
    current_queue_size < MTU then
        Do not drop packets
        Come out of dropping state
    end
    else
        while dequeue_time ≥ next_drop_time do
            Drop the packet
            count = count + 1
            next_drop_time + = interval /$\sqrt{count}$
        end
    end
end
else if outside dropping state and first packet is
being dropped then
    Enter the dropping state
end
```

The above algorithm depicts the working of the existing system: CoDel. The CoDel exists in two states: a dropping state and a non-dropping state. As shown by the algorithm, when a packet enters the network the current queue size is checked with the queue limit to accommodate the packet that has arrived. If the queue limit has not been reached then the packet is enqueued and the timestamp is attached to the header of the packet. If the queue limit has been reached then the packet is not added to the queue.

When the packet is exiting the queue it is checked for the soujourn time by the formula given and compared with target or current queue size is less than MTU then do not drop the packet otherwise the packet is calculated for the next drop time and it is dropped after incrementing the count. If the scenario has not yet entered the dropping state, then it enters the dropping state and the count is incremented by one.

```
On the advent of a packet:
if queue_limit has not been reached
    Hash the flow of packets to the respective bucket
end
else
    Drop the packet
end
On egress of the packet:
Egress_time=timestamp of packet exit
Journey_time=total time spent in queue
 if inside dropping state then
    if journey_time<target or
        current_queue_size<MTU then
            Do not drop packets
            Come out of dropping state
    end
 else
```

```
while egress_time>target do
    Compare bandwidth occupied by each packet
    Drop the packet occupying largest bandwidth
    Count = count+1
/* dropping of packets takes place in round robin fashion* /
            end
          end
      end
else if outside dropping state and first packet being dropped
then
          Enter dropping state
          Set count=1
end
```

The above algorithm depicts the working of the proposed system: SfqCoDel algorithm. On the arrival of a packet the algorithm checks whether the queue limit has been reached. If the condition hasn't been satisfies then the packet is added to the respective bucket by using the hash function to the packet flow.

If the queue limit has been reached then drop the packet. On the departure of the packet, the egress time is taken from the packet header and the journey time is calculated using the formula. Journey time is the difference between packet advent time and egress time. When the scenario is inside the dropping state then it checks if the journey time is less than the target time or the current queue size is less than the MTU then the scenario comes out of the dropping state and the packet is not dropped. If it is not satisfied then the egress time is checked and if it is greater than the target time then another condition regarding the bandwidth is checked. The packet with the largest bandwidth in the network is dropped and the count is incremented by one. The packets are dropped in a round robin fashion. In the case where the scenario has still not entered the dropping state and it is dropping the first packet then the scenario enters the dropping state and count is set to one.

VII. COMPARISON BETWEEN EXISTING AND PROPOSED SYSTEM

CoDel has some limitations that can be overcome by SfqCoDel[1] which stands to be our proposed system for fair queue management in excessive network traffic and bottleneck situations. CoDel has been tried with big routers and with more number of simultaneous flows and there, while SfqCoDel does well in such situations.

In SfqCoDel, each bucket contains a CoDel managed queue instead of a FIFO queue. This queue that is being employed in SfqCoDel is fair in nature because it drops those packets only which consume a larger bandwidth and more time. SfqCoDel prevents starvation of packets which can be transferred in less time irrespective of the time they arrived while this is not the case with CoDel. According to the simulation results, SfqCoDel has a better link utilization and less drop rate than CoDel.

VIII. COMPARATIVE ANALYSIS

The following graphs depict the analysis done on both the algorithms regarding bit error rate, throughput as well as packet delivery ratio. We can see the red line indicates existing system (CoDel) and green line indicates the proposed system (SfqCoDel). The x-axis and y-axis differ for each of the graphs. The results regarding the observations from the xgraph give a clear idea about why SfqCoDel is said to be a better and improvised variant of the CoDel algorithm for Active Queue Management technique. The results are recorded as follows with the xgraph comparative analysis:

\

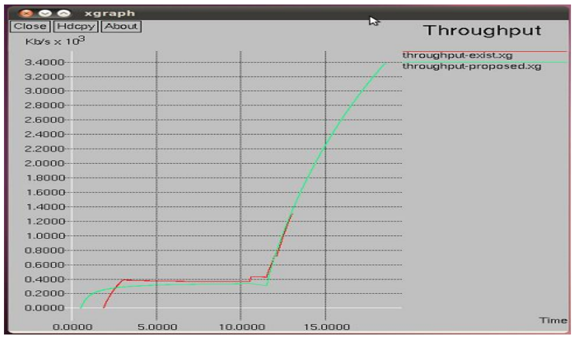


Fig 1: Throughput Analysis

- It is the rate of production or the rate at which something can be processed. When used in the context of communication networks, such as Ethernet or packet radio or network throughput is the rate of successful message delivery over a communication channel [8].
- The Fig 1 shows that the throughput of the proposed system is comparatively greater than the existing system.

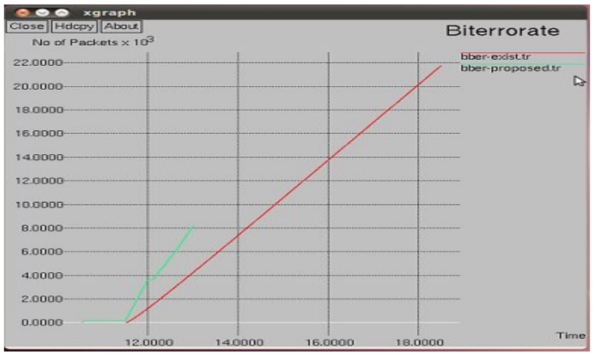


Fig 2: Bit Error Rate Analysis

- The bit error rate (BER) is the number of bit errors per unit time. The bit error ratio is the number of bit errors divided by the total number of transferred bits during a studied time interval. BER is expressed as a percentage [8].
- The above xgraph Fig 2 depicts that the existing system has a greater bit error rate than the proposed system.

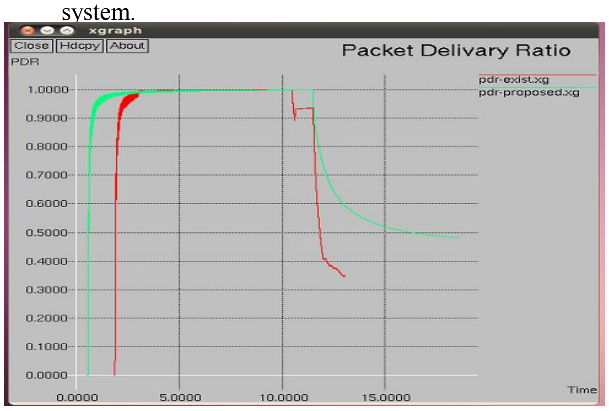


Fig 3: Packet Delivery Ratio

- The ratio of the number of packet delivered to the packets sent. This illustrates the level of delivered data to the destination [8].
- The xgraph depicts that the packet delivery ratio is much more stable, constant and better in the proposed system rather than the existing which seems unstable, fluctuating and lagging at many points.
- Through thorough analysis of all the graphs we come to a conclusion that SfqCoDel is better than the CoDel when implemented to control the network congestion due to bottleneck and BufferBloat [2] problems as explained above in the paper.

IX. CALCULATIVE ANALYSIS

The calculations made from the graph depicted above, throughput of CoDel is better by 2% than SfqCoDel at 3ms as SfqCoDel starts off with a low throughput. As we can see at 12ms and 13ms SfqCoDel proves to be 2% and 4% better than CoDel respectively. CoDel starts dropping Packets at 14th millisecond while SfqCoDel continues to deliver packets till the 18th millisecond.
Similar analysis and results have been concluded considering the bit error rate as well as the packet delivery ratio metrics. From this we can conclude that SfqCoDel is a better performer as a network scheduling algorithm than CoDel.

X. FUTURE SCOPE

Due to the extreme usage of internet from the past few decades and also our extreme dependence on internet for everything, we can see that our systems have slowed down to a large extent. This is due to early clogging of buffer resulting in packets being dropped. This situation is called buffering (Buffering in internet terms means when the page we search is loading for a long time). As a solution to the problem of BufferBloat, CoDel algorithm was used. SfqCoDel is a variant of CoDel drops packets smartly.
Future researches in this domain can be implemented on a higher and upcoming platform such as NS3. NS3 can help in keeping a track of each dropped packet with its details. The buckets at each node can have multiple queues which will be a collaboration of good and bad queues. Hashing can be implemented as well as depicted in the packet header for each packet in the queue.

CONCLUSION

Through time the computer technology will keep changing bringing about problems like latency, traffic and reduced throughput all of which could result in unnecessary dropping of useful packets. With the existing system CoDel a solution is provided however it still leads to dropping of all the packets when the traffic sets in. The proposed solution depicted in the paper efficiently controls this problem by using the concept of bandwidth.
This paper shows the concepts of CoDel and how the limitations of it overcome by an appropriate variant called SfqCoDel. The appropriate flow charts and algorithms have been implemented to show the working of the proposed system. Also the comparisons between the two systems are plotted using an X-graph. Here fixed parameters are chosen for comparison of the two with which we can also prove that SfqCoDel is better.

REFERENCES

1. Analysis of SfqCoDel for Active Queue Management; Preethi Rao V. NMAM Institute of Technology Nitte, Mohit P. Tahiliani National Institute of Technology Karnataka Surathkal, Udaya Kumar K. Shenoy NMAM Institute of Technology Nitte, Karnataka.
2. http://www.bufferbloat.net/projects/ CoDel/wiki
3. https://tools.ietf.org/html/draft-nichols-tsvwg-CoDel-02
4. https://lwn.net/Articles/496509/
5. http://queue.acm.org/detail.cfm?id= 2209336
6. http://www.cablelabs.com/wpcontent/uploads/2014/05/ PreliminaryStudyOfCoDelAQM_DOCSIS-Network.pdf
7. http://www.net.in.tum.de/fileadmin/ TUM/NET/NET-2013-08-1/NET-2013-08-1_08.pdf
8. https://en.wikipedia.org/wiki/ Network_simulation
9. http://www.cse.wustl.edu/~jain/cse5 67-08/ftp/simtools/
10. http://www.cse.wustl.edu/~jain/cse 567-08/ftp/simtools/index.html#5
11. http://www.answers.com/Q/What_are_disadvantages_of_ Matlab