Multilevel feedback queueing and resource utilization for solving bufferbloat

1. Introduction

Bufferbloat is a software issue with networking equipment that causes spikes in Internet connection's latency when a device on the network uploads or downloads files It happens when data packets are temporarily stored in large buffers before being transmitted, causing delays in their delivery. While buffering is essential for maintaining a stable connection (it helps smooth out variations in throughput, ensuring continuous data flow even when network conditions fluctuate), excessive buffering can result in poor network performance A large body of literature studied the bufferbloat problem and proposed solutions from different perspectives, including buffer sizing [1], TCP congestion control [2], and queue management [3]. SQM(Smart Queue Management) helps to manage buffer sizes dynamically to prevent excessive queuing and to address the bufferbloat . Some popular SQM algorithms include: CoDel (Controlled Delay): CoDel aims to keep the queue delay low by dropping packets when necessary. It prevents bufferbloat by ensuring that packets are not unnecessarily held in queues[4], FQ-CoDel (Fair Queue Controlled Delay): combines fair queuing with CoDel to provide better performance for interactive applications [5]. SFQ (Stochastic Fair Queuing): SFQ assigns packets to queues based on hashing, ensuring fair distribution of bandwidth , unfortunately SFQ is not able to manage the resources properly and suffer from bufferbloat[6].

a) Our contribution

In this paper, we aim to develop an easy-to-implement algorithm for solving the bufferbloat problem that overcomes the resource management issue. We introduce MLFQ that categorised the data packets into different levels or queues based on their priority or other characteristics . Each queue is assigned a different allocation of network bandwidth , and data packets move between queues based on their network behaviour . The goal is to balance fairness and efficiency in network resource allocation and data transmission.

2. Related Work

Over the past few decades, a significant body of research has been conducted on bufferbloat and some of the results are as follows : NASA report says , SFQ is an efficient queueing discipline for providing equal access to the available bandwidth . The isolation of the streams helps to ensure that no stream receives more than its fair share and that each stream degrades gracefully as more streams are added. SFQ also seems to possess very good scaling properties; but more work needs to be done to verify this. In particular, the choice of hash function and seed perturbation technique needs further investigation. The current choices may prove inadequate in a more stressful environment[6]

The Overhead of being fair , an research article says that the fairness of the SFQ is greatly affected by the number of queues to which flows are hashed [7] . One of the most important drawbacks of this method is unfair behavior with the flows colliding with other flows. Thus, as the name reveals, fair is guaranteed as stochastically. [8] It is suitable for use in high speed computer networks that covers a wide range of CPU, memory and fairness trade-offs. It offers elegant degradation under overload and sudden failure. [9]

3. Our approach/Methodology

Firstly we had taken up the references of the parameters used by the other researchers to showcase the performance of the SFQ ,so that we can prove the SFQ as a culprit of delay generation . Table to demonstrate SFQ's performance on the basis of NASA report :

Parameter	SFQ	FIFO
1)Fair utilization of available resources	Better	
2)Starvation prevention	Better	
3)Graceful degradation under overload conditions		Better
4)Resource usage		Better

With the help of the reports we can say that SFQ (Stochastic Fair Queuing) neither able to perform in the overload conditions nor able to manage resources properly .So for that , we introduce our alternative i.e MLFQ(Multilevel feedback queue) as a solution as it havespace complexity O(n) where n is the total number of packets & SFQ have space complexity O(n+m) where n is the total number of flows and m is the total number of packets .

MLFQ Algorithm :

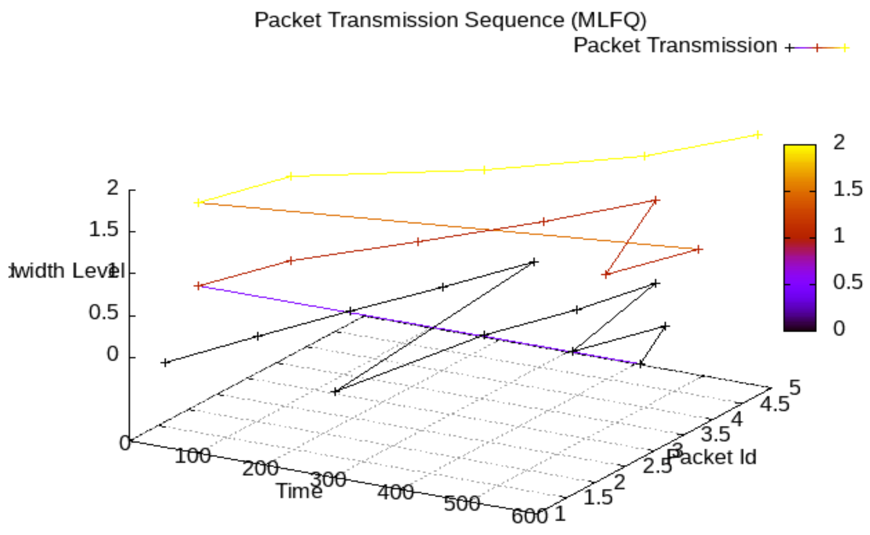
```
   MLFQ(network_packets, time_quantum_levels):
   Initialize empty queues for each feedback level
   Initialize a queue to store packets waiting to be processed for packet
   In the network packet :
   Add packet to the queue based on its initial priority for each
   For time quantum level in time quantum levels :
   while the queue is not empty:
    packet = dequeue packet from the queue
    remaining_size = packet.size
    while remaining_size > 0:
    if packet.size <= bandwidth of current level:
    Transmit packet
    remaining_size = 0
    else:
    Transmit packet with available bandwidth
  remaining_size -= transmitted size
  If remaining_size > 0:
  Reduce priority level of packet
  Enqueue packet into the next lower priority level queue
```

4. Result analysis & Discussion

Comparison table

Algorithm	Time Complexity	Space Complexity
SFQ	O(n)	O(n+m)
MLFQ	O(S)	O(n)

Where for SFQ n is number of flows and m is total number of packets.
& for MFQ S is sum of the packet sizes and n is the total number of packets **.**
From the above comparison table, it is clearly shown that MLFQ performs better in terms of space complexity**.**



INTERPRETATION

Round Robin (RR) vs. Multi-Level Feedback Queue (MLFQ)

1. Round Robin (RR):

- Performance: RR schedules packets in a cyclic manner without considering packet size or arrival time. This approach can lead to inefficiencies, especially with varying packet sizes and arrival times.
- Results: Transmission times varied, with larger packets potentially causing delays when processed in lower bandwidth levels.

2. Multi-Level Feedback Queue (MLFQ):

- Performance: MLFQ adapts dynamically by prioritizing packets based on size and arrival time, adjusting bandwidth allocation across different levels.
- Results: MLFQ showed reduced latency and improved throughput, efficiently handling diverse packet sizes and arrival times.

Hence, MLFQ outperforms RR by providing better performance in terms of latency and
bandwidth utilization. It is more effective for networks with varied packet sizes and arrival
rates, while RR may suffice for simpler scenarios.

5. Conclusion

We developed an alternative of SFQ i.e MLFQ (Multilevel feedback queue) so that the resource management can be done properly and the problem bufferbloat can be address properly .

6. References

[1]Buffer Sizing Experiments at Facebook (stanford.edu)
[2] Impact of TCP congestion control on bufferbloat in cellular networks | IEEE Conference Publication | IEEE Xplore
[3] Adams: Active queue management: A survey - Google Scholar
[4] Towards a better understanding and analysis of controlled delay (CoDel) algorithm by using fluid modelling - Patil - 2019 - IET Networks - Wiley Online Library

[5] (PDF) Analysing the Latency of Sparse Flows in the FQ-CoDel Queue Management Algorithm (researchgate.net)

[6] 19970014215.pdf (nasa.gov)

[7](PDF) The Overhead of Being Fair (researchgate.net)

[8] https://www.lifesciencesite.com/lsj/life1001/059_12183life1001/059_12183life1001_372_374.pdf

[9] https://www.ijcsit.com/docs/Volume%203/Vol3Issue2/ijcsit20120302108.pdf