COL334 Assignment 3

Kavya Chopra (2021CS10081), Manpreet Singh(2021CS10070)

November 1, 2023

1 Algorithm

We have two implementations,

1.a Round Robin

We use a round robin approach, where we maintain a set of packets which have not yet been received, and request packets by iterating over this set. In this scenario, the packets received from the server may not be in the exact order as we are requesting. Since the server has a variable rate of filling the tokens, we constantly need to re-estimate the rate at which we're sending requests. We did this by considering the number of responses in the previous 10 requests, if we're missing replies, then we increase the time interval between two requests otherwise we can reduce this time interval. This is a more "continuous" implementation, as we're tuning the time between each request as compared to times between a burst of requests.

1.b Bursty approach

To estimate the rate at which the server can send data packets(the rate of filling tokens), we can keep track of how many requests are being skipped over a 10 packet burst. If an unusually large number of requests are being skipped, we can infer that we are getting squished and we should slow down the rate of sending requests(essentially increase the time between two consecutive bursts) as well as decrease the burst size, which is also essentially decreasing the rate of sending requests, otherwise we can increase the rate of sending requests. We also fine tune the send time between sending these bursts accordingly.

1.c Metrics

We were able to get the entire file from Vayu server in about 21.6s, and with a penalty of about 54 using the packet burst approach. We don't get squished even once.
Using the round robin approach we are able to get the file from the server in about 22.67s with a penalty of 39. We don't get squished even once.

2 Graphs

2.a Round Robin

With the round robin approach we get the following graph

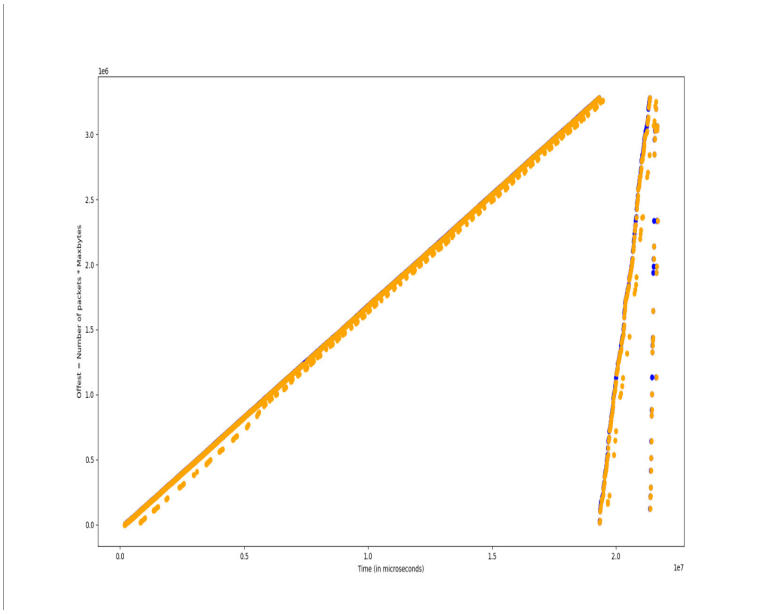

Figure 1: Round Robin Graph: Offset vs Time

We get different straight lines, because some requests are skipped by the server in the first pass, which are then received in the future passes. Here's a zoomed version of how the requests and responses look like

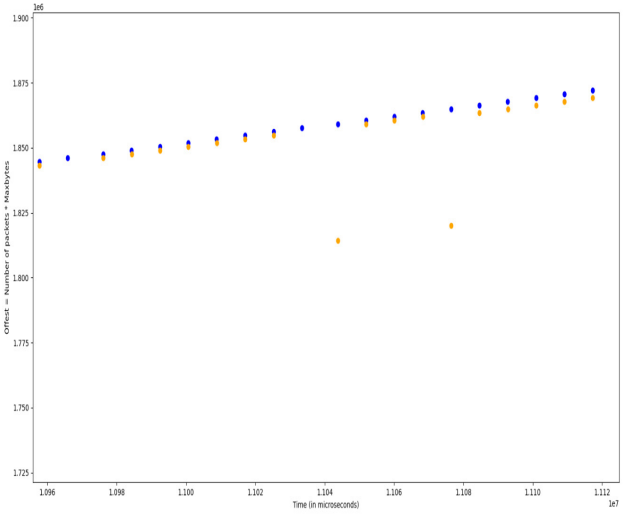

Figure 2: Round Robin Graph: Zoomed version

2.b Bursty

With the bursty approach, we get

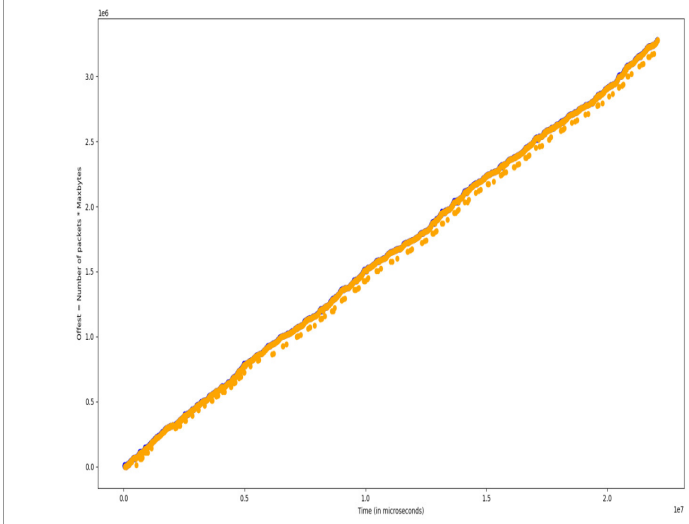

Figure 3: Offset with time

The send requests are blue and the receive requests are orange, since we're sending requests in bursts, but receiving in a staged manner, the blue dots are on the vertical line and the yellow ones form a straight line.

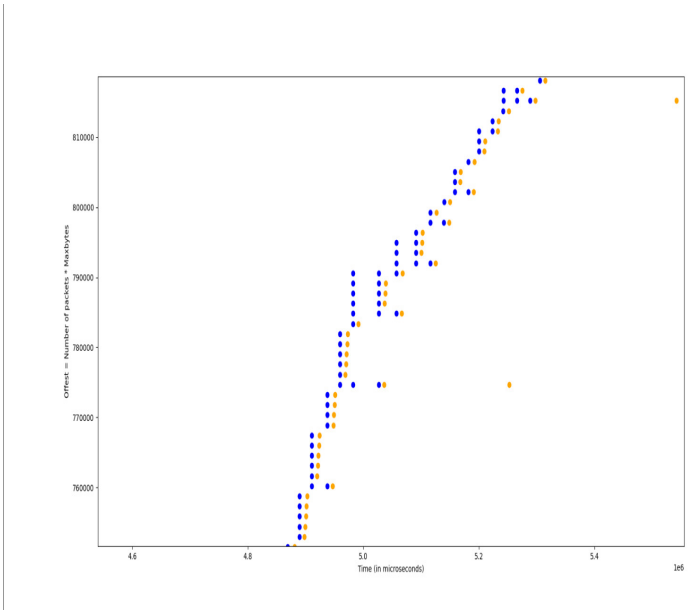

Figure 4: Offset with time for burst approach

This is a further zoomed in view of the same graph, again we send the requests all at once (in bursts) and receive the server replies in a staged manner.

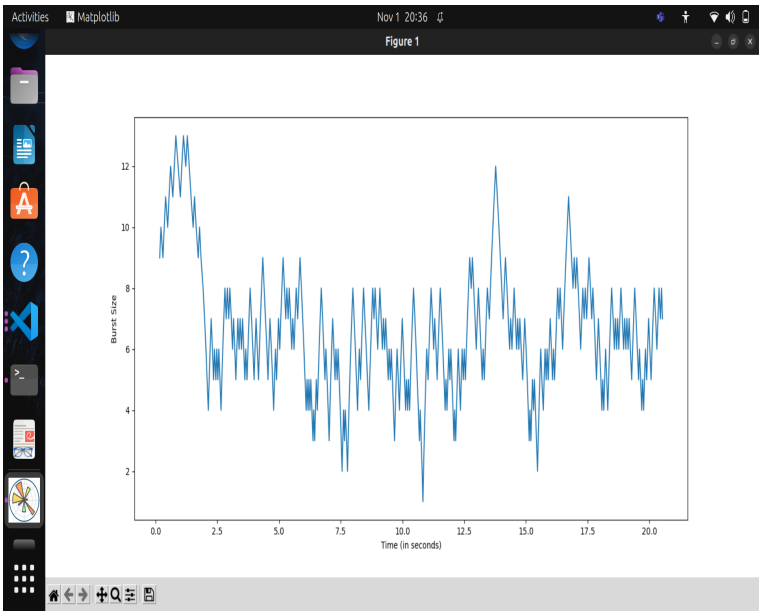

Figure 5: Burst size variation with time

This shows the variation of burst size with time as the server changes its rate, when more requests are dropped, we decrease the burst size to a minimum of one, otherwise increase it to a maximum of 15. We can see our implementation adjusts burst sizes dynamically due to the high variance across the graph.