Computer Networks Lab Report

Md Talha Yaseen Khan 111701019

Lab-5 (FIFO queuing and Token bucket)

1. Token bucket

- ### Code Overview
  - Function **void shape(int capacity, double rate)** takes two argument, size of bucket and rate at which tokens are falling into bucket.
  - I initialized a variable *filledBucket* with 0 (means bucket is filled initially) of data type double to keep track of how much bucket is filled.
  - Then I checked for packet-1. If length of packet-1 is greater than capacity of bucket then transmission time would be equal to *(Arrivial time + (length-capacity)/rate)*, if length is smaller than capacity then *transmission time = arrival time* and *(capacity – length)* amount of bucket wil be remain filled.
  - Then by using loop, I'm checking for remaining 99 packets.
  - If capacity of bucket is zero then time taken to transmit would be *length/rate* and *transmission time = max{tTime(previous packet), aTime(current packet)} + time taken to transmit*.
  - And if capacity isn't zero then bucket will be filled with tokens when there is time gap between arrival time of current packets and transmission time of previous packet. (Maximum value of filledBucket = capacity).
  - If bucket is filled more than length then no time will be taken to transmit. So *transmission time = arrival time* and *filledBucket = filledBucket – length.*
  - If bucket is totally filled and length of packet is also greater than capacity of bucket and previous packet transmission time is less

than current arrival time then *Transmission time = arrival time + (length – capacity)/rate.*

- **Command to compile and run are:**

  ```
  gcc shape.c -o shape
  ./testShape.sh
  ```

2. FIFO queuing

• Code Overview

- Function `void fifo(int capacity, double rate)` takes two argument, buffer size of queue and rate at which data is pushed to the output line of queue.
- In FIFO queuing some packets may be dropped and not been sent. This can happen when queue is full or packet length is greater than capacity of queue. And that's why I am not fixed the length of tTime array. Whenever packet is being send I am allocating the memory for tTime array.
- I initialize n as 0 and emptySpace as size of queue, where n denotes how much packets are being send and emptySpace keeps track of emptySpace in queue.
- Them I am checking for first packet, if length of first packet is less than size of queue then first packet will be sent and *remaining empty space = capacity – length of first packet.* Otherwise packet will not be sent.
- Then by using I'm checking for remaining 99 packets.
- Now, I'm checking for emptySpace and if emptySpace is less than length of packet and if n > 0, then reallocating the space for tTime and *if emptySpace = capacity* then *transmission time = arrivals time + length/rate* otherwise *transmission time = previous packet tTime + length/rate* and incrementing the n.
- If this the first packet which is satisfying the criteria of being send then *transmission time = arrivals time + length/rate* and *emptySpace is decremented by length of that packet.*
- **Command to compile and run are:**

  ```
  gcc fifo.c -o fifo
  ./testFifo.sh
  ```

3. Combination of Token Bucket and FIFO Queue

- **Code Overview**
  - Assigned globally a maxErr = 10^{-7} , which is the maximum error that can be there in x.
  - There are three functions here.
    - **`double modulus(double a)`**
      - This function returns the absolute value of arguments.
    - **`void shape(int capacity, double rate)`**
      - This is the same function which I implement in first question for Token bucket.
    - **`bool fifo(int capacity, double rate)`**
      - This is the same function which I implement in second question for FIFO queue.
      - The only difference is, this function returns boolean (true or false).
      - When all the packets is sent successfully (no packet dropped) then this function returns **true**.
      - When some packets is dropped then this function returns **false**.
  - In the main function I'm using Binary Search algorithm to find the value of largest x in minimum iteration.
  - Performing Binary Search from start = 0.1 to finish = 100.0, and initializing x = 0.1. If no packet has dropped and error is less than 10^{-7} then returning that x value.
  - If no packet has dropped and error is greater than 10^{-7} then to minimize the error assigning x = mid and start = x.
  - If some packet has dropped then searching x in between start to mid.
  - Largest value of x I got is **1.956529.**
  - **Command to compile and run are:**

```
gcc 3.c -o 3
./3 < arrivals.txt
```