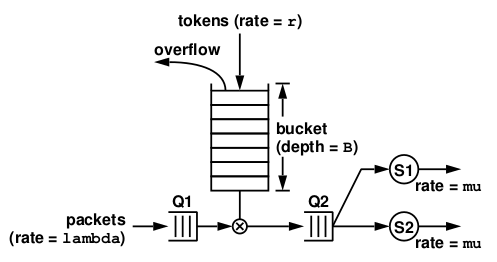
In this I will emulate/simulate a traffic shaper which transmits packets controlled by a token bucket filter depicted below using multi-threading within a single process.

 **igure 1** above depicts the **system** you are required to emulate. The **token bucket** has a capacity (bucket depth) of **B** tokens. Tokens arrive into the token bucket according to an unusual arrival process where the inter-arrival time between two consecutive tokens is **1/r**. We will call **r** the **token arrival rate** (although technically speaking, it's not exactly the token arrival rate; please understand that this is **quite different** from saying that the tokens arrive at a constant rate of **r**). Extra tokens (overflow) would simply disappear if the token bucket is full. A token bucket, together with its control mechanism, is referred to as a **token bucket filter**.

Packets arrive at the token bucket filter according to an unusual arrival process where the inter-arrival time between two consecutive packets is **1/lambda**. We will call **lambda** the **packet arrival rate** (although technically speaking, it's not exactly the packet arrival rate; please understand that this is **quite different** from saying that the packets arrive at a constant rate of**lambda**). Each packet requires **P** tokens in order for it to be eligiable for transmission. (Packets that are eligiable for transmission are queued at the **Q2** facility.) When a packet arrives, if **Q1** is not empty, it will just get queued onto the **Q1** facility. Otherwise, it will check if the token bucket has **P** or more tokens in it. If the token bucket has **P** or more tokens in it, **P** tokens will be removed from the token bucket and the packet will join the **Q2** facility (technically speaking, you are **required** to first add the packet to **Q1** and timestamp the packet, remove the **P** tokents from the token bucket and the packet from **Q1** and timestamp the packet, before moving the packet into **Q2**). If the token bucket does not have enough tokens, the packet gets queued into the**Q1** facility. (Please note that, ***[BC: next 3 words added 2/7/2016]*** **in this case**, you do **not** have to check if there is enough tokens in the bucket so you can move the packet at the head of **Q1**into **Q2** and you need to understand why you do **not** need to perform such a check.) Finally, if the number of tokens required by a packet is larget than the bucket depth, the packet must be**dropped** (otherwise, it will block all other packets that follow it).

The transmission facility (denoted as **S1** and **S2** in the above figure and they are referred to as the "**servers**") serves packets in **Q2** in the first-come-first-served order and at a service rate of **mu** per second. When a server becomes available, it will dequeue the first packet from **Q2** and start transmitting the packet. When a packet has received **1/mu** seconds of service, it leaves the system. You are required to keep the servers **as busy as possible**.

When a token arrives at the **token bucket**, it will add a token into the **token bucket**. If the bucket is already full, the token will be lost. It will then check to see if **Q1** is empty. If **Q1** is not empty, it will see if there is enough tokens to make the packet at the head of **Q1** be eligible for transmission (packets in **Q1** in also served in the first-come-first-served order). If it does, it will remove the corresponding number of tokens from the token bucket, remove that packet from **Q1** and move it into **Q2**, and **wake up** the servers in case they are sleeping. It will then check the packet that is now at the head of **Q1** to see if it's also eligible for transmission, and so on.

Technically speaking, the "servers" are not part of the "token bucket filter". Nevertheless, it's part of this assignment to emulation the severs because the servers are considered part of the "system" to be emulated.

Our system can run in only one of two modes.

|  |  |  |
| --- | --- | --- |
| **Deterministic** | : | In this mode, all inter-arrival times are equal to **1/lambda** seconds, all packets require exactly **P** tokens, and all service times are equal to **1/mu** seconds. If**1/lambda** is greater than 10 seconds, please use an inter-arrival time of 10 seconds. If **1/mu** is greater than 10 seconds, please use an service time of 10 seconds. |
|  | | |
| **Trace-driven** | : | In this mode, the emulation will be driven using a trace file. Each line in the trace file specifies the **inter-arrival time** of a packet, the **number of tokens** it need in order for it to be eligible for transmission, and its **service time**. (Please note that in this mode, it's perfectly fine if an inter-arrival time or a service time is greater than 10 seconds.) |

The  job is to emulate the packet and token arrivals, the operation of the token bucket filter, the first-come-first-served queues **Q1** and **Q2**, and servers **S1** and **S2**. A trace of the emulation for every important event occurred in the emulation is also produced.