|  |  |  |
| --- | --- | --- |
| **Assignment Description** | | |
| In this assignment, you will emulate/simulate a **traffic shaper** who transmits packets controlled by a **token bucket filter** depicted below using multi-threading within a single process.     |  | | --- | | **Figure 1**: A system with a token bucket filter. |   **Figure 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 at a **constant rate** of **r** tokens per second. Extra tokens (overflow) would simply disappear if the token bucket is full. A token bucket, together with its control mechanism, is referred as a **token bucket filter**.  Packets arrive at the token bucket filter at a rate of **lambda** packets per second (i.e., packets have an inter-arrival time of **1/lambda**) and each packet requires **P** tokens in order for it to be eligible for transmission. (Packets that are eligible 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** tokens 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. You should also check if there is enough tokens in the bucket so you can move the packet at the head of **Q1** into **Q2**.  The transmission facility **S** serves packets in **Q2** at a service rate of **mu** per second. When a packet has received **1/mu** seconds of service, it leaves our system.  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 transmissions. 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 server (by broadcasting the corresponding condition). Technically speaking, the "server" is not part of the token bucket filter. Nevertheless, it's part of this assignment to emulation the sever as well.  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, we will drive the emulation using a [tracefile](http://merlot.usc.edu/cs402-s14/projects/warmup2/" \l "tracespec). 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 eligiable for transmission, and its **service time**. |   Your 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 server **S**. You also must produce a trace of your emulation for every important event occurred in your emulation. Please see [more details](http://merlot.usc.edu/cs402-s14/projects/warmup2/#exec) below for the requirements.  You must use:   * one thread for packet arrival * one thread for token arrival * one thread for server   You must **not** use one thread for each packet.  In addition, you must use at least one mutex to protect **Q1**, **Q2**, and the token bucket.  Finally, **Q1** and **Q2** must have infinite capacity (i.e., you should use **My420List** from [warmup assignment #1](http://merlot.usc.edu/cs402-s14/projects/warmup1) to implement them and **not** use arrays).  We will **not** go over the [lecture slides for this assignment](http://merlot.usc.edu/cs402-s14/lectures.html#warmup2) in class. Although it's important that you are familiar with it. Please read it over. If you have questions, please e-mail the **instructor**. | | |
|  |  |  |
| **Compiling** | | |
| Please use a Makefile so that when the grader simply enters:  make warmup2  an executable named **warmup2** is created. Please make sure that your submission conforms to [other general compilation requirements](http://merlot.usc.edu/cs402-s14/makefile.html#requirements) and [README requirements](http://merlot.usc.edu/cs402-s14/makefile.html#readme). | | |
|  |  |  |
| **Commandline** | | |
| The command line syntax for **warmup2** is as follows:  **warmup2** [**-lambda** lambda] [**-mu** mu] \  [**-r** r] [**-B** B] [**-P** P] [**-n** num] \  [**-t** tsfile]  Square bracketed items are optional. You must follow the UNIX convention that **commandline options** can come in any order.  The lambda, mu, r, B, and P parameters all have obvious meanings. The **-n** option specifies the total number of packets to arrive. If the **-t** option is specified, tsfile is a [trace specification file](http://merlot.usc.edu/cs402-s14/projects/warmup2/#tracespec) that you should use to drive your emulation. In this case, you should ignore the **-lambda**, **-mu**, **-P**, and **-num** commandline options and run your emulation in the [trace-driven mode](http://merlot.usc.edu/cs402-s14/projects/warmup2/#tracedriven). You may assume that tsfile conforms to the [tracefile format specification](http://merlot.usc.edu/cs402-s14/projects/warmup2/" \l "tracespec). (This means that if you detect an error in this file, you may simply print an error message and call exit(). There is no need to perform error recovery.) If the **-t** option is not used, you should run your emulation in the [deterministic mode](http://merlot.usc.edu/cs402-s14/projects/warmup2/#det).  The default value (i.e., if it's not specified in a commandline option) for lambda is 0.5 (packets per second), the default value for mu is 0.35 (packets per second), the default value for r is 1.5 (tokens per second), the default value for B is 10 (tokens), the default value for P is 3 (tokens), and the default value for num is 20 (packets).B, P, and num must be positive integers with a maximum value of 2147483647 (0x7fffffff). lambda, mu, and r must be positive real numbers. If **1/r** is greater than 10 seconds, please use an inter-token-arrival time of 10 seconds. | | |