**Simulation Project - 1st Assignment**

*Please note that any exchange of code with another student is discouraged as it constitutes cheating*. *We will run Moss to verify that there has not been code sharing.*

**1. Project description**

The simulation project is motivated from a feedback-based scheme for downloading video on demand and broadcasting live programs, whereby a client controls the download speed based on how fast or how slowly it receives the packets from the video server.

We will assume that the video server can transmit at various speeds. The client observes its queue where all the video packets await for processing and displaying on the client’s screen. If the queue size drops below a threshold *TL*, the client instructs the server to increase its transmission speed. If this persists, it will keep increasing the server’s speed until it reaches its highest speed. If on the other hand, the queue is over a threshold *TH*, it will instruct the server to decrease its transmission rate. If this persists, it will keep decreasing the server’s speed until it reaches its lowest speed.

The objective of the simulation project is to calculate the two thresholds of the queue length that will cause the client to request a change in the server’s transmission speed.

Figure 1: The system under study

We will make the following assumptions:

* We will assume that the server can transmit at two different speeds, high and low.
* The client is represented by a single server queue, as shown in figure 1. The infinite server queue is used to model the network delay, that is, the time it takes for a packet to travel from the video server to the client.
* We will assume that the client makes a decision about the transmission speed each time a packet arrives at its queue. This decision is communicated to the video server instantaneously which acts accordingly.
* If the number of packets waiting in the client queue is less or equal than a threshold *TL*, then it instructs the video server to switch from the low transmission rate to the high transmission rate. No action is taken if the server is already transmitting at the high rate. If the number of packets waiting in the client queue is greater or equal to *TH*, then it instructs the video server to switch from the high transmission rate to the low transmission rate. No action is taken if the server is already transmitting at the low rate.
* All packets have the same length, i.e., 1500 bytes.

**2. The hand simulation**

We will start with a hand simulation in order to understand how the event clocks work and how the simulation advances from event to event. In subsequent assignments, we will add more complexity. We will make the following assumptions:

* The two thresholds are the same, i.e., *TL = TH=T*, and *T*=2. Switch to high rate if the number in the queue upon arrival of a packet is less or equal to *T*. Switch to the low rate if the number in the queue upon arrival of a packet is greater than *T*.
* High transmission rate: A packet is transmitted every 1 unit of time.
* Low transmission rate: A packet is transmitted every 2 units of time.
* The time a packet spends in the infinite server queue will vary so that to induce different delays to different packets. This time cannot be less than the total propagation delay, which we will assume that is equal to the 2 units of time. (We will make up some numbers in the hand simulation.)
* The service time of the packet queue at the client is the time to process the packets so that their content can be displayed on the screen, and it is equal to 1 unit of time.

*State of the system*

The first task in developing a simulation is to identify the state of the system. This is consists of a set of variables that completely describes the system at any time *t*. In our case, the state of the system is described by following variables:

* Server: transmission speed, departure time of next packet.
* Infinite server queue: Number of packets in the queue and the time at which each packet will depart.
* Client queue: Number of packets waiting in the queue, time the packet in service will complete its service.

*Events*

The second task is to identify the various events whose occurrence change the state of the system. These are:

1. Departure of a packet from the server
2. Departure of a packet from the infinite server queue
3. Departure of a packet from the client queue.

*Actions*

For each event, we have to identify the action that needs to be taken. Often the occurrence of an event may cause new events to occur.

1. Departure of a packet from the server

a The packet joins the infinite server queue. Decide how long it would stay in the infinite server queue.

b. Calculate when the next packet will depart from the server. For this you will have to determine whether the server has to change the transmission speed based on a flag set by the client.

2. Departure of a packet from the infinite server queue

a. The packet joins the client queue, and if its server is busy, then no further action is required.

b. If its server is idle, then the packet goes into service. Determine when the service will be completed.

c. Check the occupancy level of the client queue (includes the packet that has just arrived) and accordingly set the flag to up (increase speed) or down (decrease speed).

3. Departure of a packet from the client queue

a. The packet departs from the system. If no packet is waiting, then the server become idle. No further action is taken.

b. Otherwise, the next packet in line goes into service. Determine when the service will be completed.

*Clocks*

Each event is associated with a clock that shows the time in the future that will occur. Let the clocks be:

* CL-p: departure of a packet from the server
* CL-isq: departure of a packet from the infinite server queue
* CL-cq: departure of a packet from the client queue

Also, it is useful to maintain a clock that shows the current time of the simulation. This is known as the master clock, MCL.

*Main logic*

The main logic is actually quite simple. Once all the actions associated with an event have been taken care of, search the event clocks and locate the event with the smallest clock value. This is the next event that will occur. Execute the logic associated with the event. Then locate the next event, and so on.

*Initial conditions*

These condition describe the initial state of the system at time zero. We will assume that at time zero, the server starts transmitting a packet at the high transmission speed.. The infinite server queue and the client’s packet queue are empty. As will be seen later on, the initial conditions do not have an impact on the steady-state analysis of the system.

*Hand simulation*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Actions** | | **Server** | | **Infinite server q.** | **Client queue** | | |
| **MCL** | **Event no** | **Trans.**  **speeed** | **CLp** | **CL-isq for each**  **waiting packet (delay)** | **CL-cq** | **Νο queue** | **flag** |
| 0 |  | H | 1 |  |  |  |  |
| 1 | 1 | H | 2 | 3 (2) |  |  |  |
| 2 | 1 | H | 3 | 3 (2), 4(2) |  |  |  |
| 3 | 1 | H | 4 | 3(2), 4(2), 6(3) |  |  |  |
| 3 | 2 | H | 4 | 4(2), 6(3) | 4 | 1 | up |
| 4 | 1 | H | 5 | 4(2), 6(3), 8(4) | 4 | 1 |  |
| 4 | 2 | H | 5 | 6(3), 8(4) | 4 | 2 | up |
| 4 | 3 | H | 5 | 6(3), 8(4) | 5 | 1 |  |
| 5 | 1 | H | 6 | 6(3), 8(4),7(2) | 5 | 1 |  |
| 5 | 3 | H | 6 | 6(3), 8(4),7(2) | - | 0 |  |
| 6 | 1 | H | 7 | 6(3), 8(4),7(2),8(2) | - | 0 |  |
| 6 | 2 | H | 7 | 8(4),7(2),8(2) | 7 | 1 | up |
| 7 | 1 | H | 8 | 8(4),7(2),8(2),10(3) | 7 | 1 |  |
| 7 | 2 | H | 8 | 8(4),8(2),10(3) | 7 | 2 | up |
| 7 | 3 | H | 8 | 8(4),8(2),10(3) | 8 | 1 |  |
| 8 | 1 | H | 9 | 8(4),8(2),10(3),12(4) | 8 | 1 |  |
| 8 | 2 | H | 9 | 10(3),12(4) | 8 | 3 | down |
| 8 | 3 | H | 9 | 10(3),12(4) | 9 | 2 |  |
| 9 | 1 | L | 11 | 10(3),12(4),11(2) | 9 | 2 |  |
| 9 | 3 | L | 11 | 10(3),12(4),11(2) | 10 | 1 |  |
| 10 | 2 | L | 11 | 12(4),11(2) | 10 | 2 | up |
| 10 | 3 | L | 11 | 12(4),11(2) | 11 | 1 |  |

**3. Assignment 1**

Carry out a new hand simulation using the above table. Use the same assumptions as above, except for the delay times in the infinite server queue. For these times use the following sequence: 2,2,4,3,2,2,5,4,3,2,6,5,4,3,2,5,4,3,2,4,3,2, 2,2. Stop the simulation when you use all the delay numbers.