Simulation of M/M/2/2+5 Queueing System

Jiaqi Yan, Ping Liu
CS555 Project
supervised by Edward Chlebus

November 22, 2015

Abstract

In this project, we simulate and analyze the M/M/2/2+5 queueing system. To generate generate packet arriving time and processing time, we use Python's built-in *random* module, which is validated carefully. Then we apply the **Welch graphical procedure** to eliminate the warm-up period in the simulation. With the stationary region, we then analyze the system's properties such as blocking probability and mean number of packet in the system. The 90% confidence interval for these properties are also given.

Contents

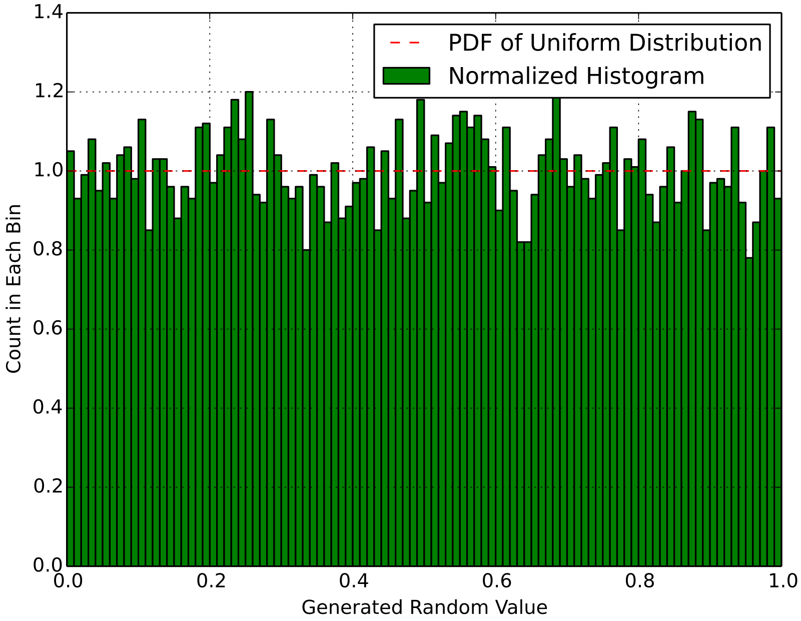

Figure 1: Histogram of Uniform Random Values Generated by Numpy

1 Discrete Event Simulation

We designed and implemented our own simple simulator for M/M/2/2+5 queueing system. The simulation workflow is shown in Algorithm 1.

2 Random Generator

2.1 Validation

In M/M/2/2+5 queueing system, the number of packet arriving in a fixed interval follows **Poisson Distribution** and the service time for each packet follows **Exponential Distribution**. These input data is generated by *Numpy*'s *random* module. Before running the simulator, it is important to test the wellness of this random generator.

We evaluate Numpy's random generator in two ways. First we generate random values follows uniform distribution and then plot their histogram. As shown in Figure 1, the number of random values falling into each interval is close to each other. This means that the generated values are very close to uniformly distributed. Besides, we compare the normalized histogram to the 'best fit' curve of both uniform distribution and normal distribution in Figure 1 and 2. From both figures we can say that the random generator generated random values of user-specified distribution.

To generate different random value sequences, we feed the random generator a seed, which is the integral value of system current time. That is the number of

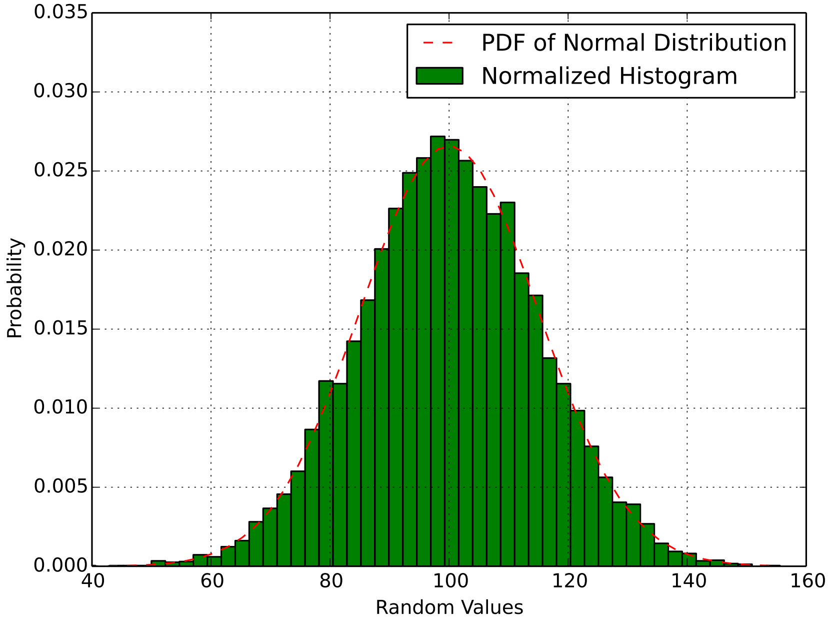

Figure 2: Comparison to Best Fit Curve for Normal Distribution

seconds since **Unix Epoch**. We generated two sequences of 1000000 numbers each and use **t-test** to test statistically are they significantly different from each other.

3 Eliminate Warm-up Period

4 System Properties at Stationary State

4.1 Blocking Probability

4.2 Mean Spending Time

4.3 Mean Number of Packet

Algorithm 1 Core of Discrete Event Simulation

```
function SIMULATION_CORE($arrive\_time\_seq$, $serve\_time\_seq$)
    let $arrive\_time\_seq$ denote the arriving time of each packet
    let $serve\_time\_seq$ denote the service time of each packet
    let $N$ denote the number of total packets
    while $pkt\_served + pkt\_dropped < N$ do
        if $pkt\_seen < N$ then                ▷ Insert new arrival event to event list
            $ts \leftarrow arrive\_time\_seq[pkt\_seen]$
            Create new arrival event $evt$ with time stamp $ts$ and id $pkt\_seen$
            Insert $evt$ to $event\_list$
        end if
        Sort $event\_list$ based on events' time stamp
        Pop up the next event $evt_x$ we need to handle in $event\_list$
        $clock \leftarrow evt_x.time\_stamp$
        if $evt_x$ is departure event then
            if queue buffer is empty then
                Set the status of the server $evt_x$ is leaving to $idle$
            else
                $- - waiting$
            end if
            $+ + pkt\_served$
            $evt\_x.exit_time \leftarrow clock$
            $spending\_time \leftarrow evt\_x.exit\_time - evt\_x.enter\_time$
            Record the spending time of packet of event $evt_x$
        end if
        if $evt_x$ is arrival event then
            if queue buffer is full then
                $+ + pkt\_dropped$
            else
                if There is available server then
                    $id \leftarrow evt_x.pkt\_id$
                    $ts \leftarrow clock + serve\_time\_seq[id]$
                    Choose an available server $s$
                    Mark $s$ as $busy$
                else
                    $(ts, s) \leftarrow$SCHEDULE_DEPARTURE( )
                    $+ + waiting$
                end if
                Create new departure event $evt$ with time stamp $ts$
                $evt.enter\_time \leftarrow clock$
                $evt.depart\_srv \leftarrow s$
                Insert $evt$ to $event\_list$
            end if
        end if
    end while
end function
```