COMPENG 4DK4 Lab 5 Report

Aaron Pinto
pintoa9

Raeed Hassan
hassam41

November 30, 2022

Random Number Generator Seeds

For the experiments in this lab, we used the same set of 8 random number seeds for all experiments. Experiment 1 instructs us to include runs with our *McMaster Student ID numbers* as our seeds. We used our *McMaster IDs* and shifted them by one digit at a time to create 4 different seeds from each our IDs, for a total of 8 different seeds. All the random number generator seeds can be seen in Table 1. In the C code used for the experiments, leading zeroes are removed.

Table 1: Random Number Generator Seeds

400188200	400190637
001882004	001906374
018820040	019063740
188200400	190637400

Experiment 1

The simulation was created using the `simlib` library and existing functions from Lab 4. The Lab 4 Experiment 6 code featuring two channels (a reservation channel and data transmission channel) was re-used with the reservation channel being changed to feature a regular transmission channel instead of an S-ALOHA channel, and the use of multiple stations/channels removed. Two packet arrival events were created, and initially ran in the main function to schedule the first job arrivals. The packet arrival event function for $M = G$ is shown in Listing 1. The jobs are then placed into the station queue, and transmission on the station begins if the job is the only one in the queue. The end of transmission event is shown in Listing 2, where jobs that finished transmission from the station are removed from the station queue, and put into the server queue for execution. Server execution occurs the same way transmission at the base station occurs. The end of server execution event is shown in Listing 3.

The simulation was run with $U_b = 10U_g = 10, U_g = 1, J_g = J_b = 1$. To determine the maximum arrival rate, λ^*, that can be supported for $d_{max} = 20$, the simulation was ran with increasing values of the arrival packet arrival rate λ. The mean delay versus λ for both devices was plotted in Figure 1. For $d_{max} = 20$ and $U_b = 10, U_g = J_g = J_b = 1$, the maximum arrival rate, λ^*, is slightly above 0.06 (the mean delay for device B at $\lambda = 0.06$ is 19.9).

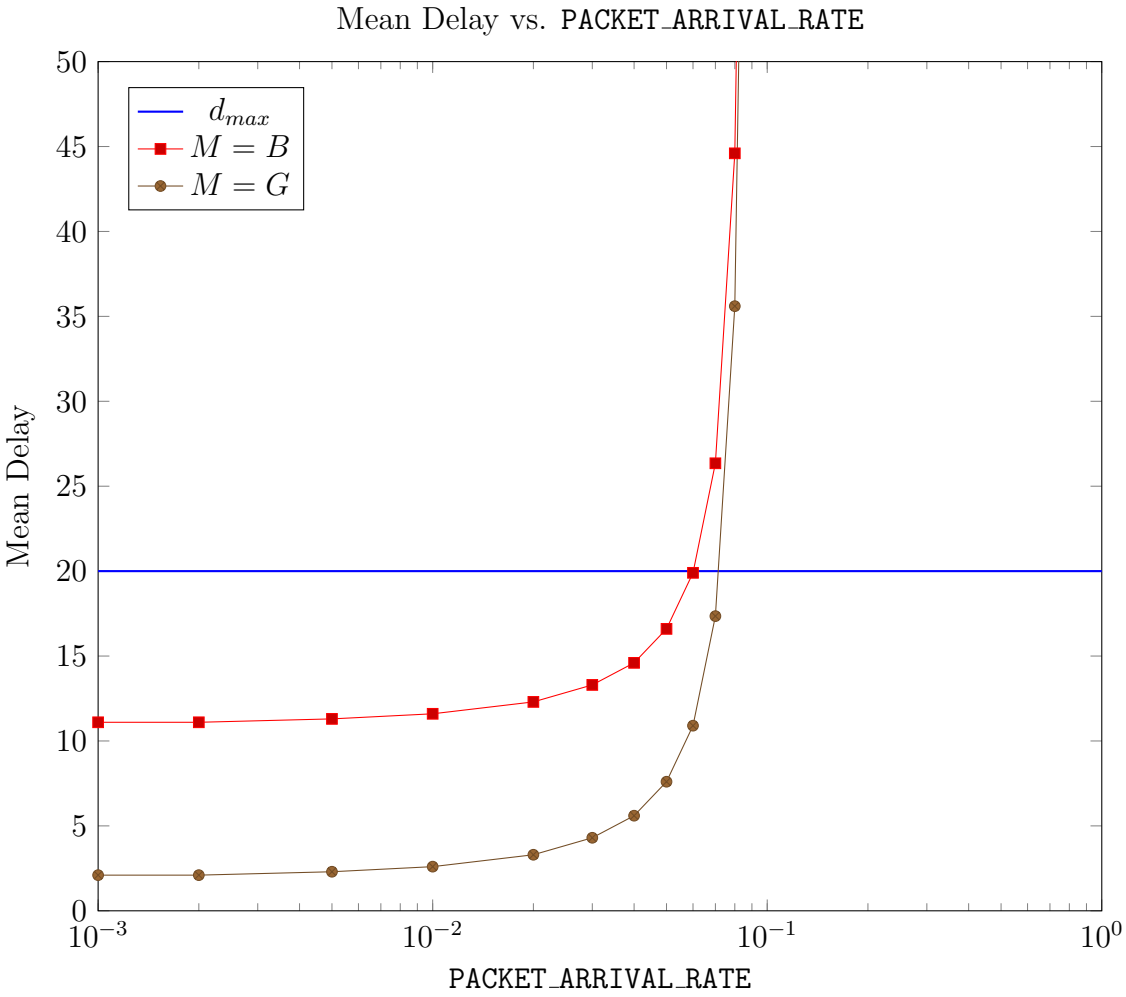

Figure 1: Experiment 1

Listing 1: Arrival Event

```
1  void
2  packet_arrival_event_G(Simulation_Run_Ptr simulation_run, void*
       dummy_ptr)
3  {
4      Packet_Ptr new_packet;
5      Time now;
6      Simulation_Run_Data_Ptr data;
7
8      now = simulation_run_get_time(simulation_run);
9
10     data = (Simulation_Run_Data_Ptr) simulation_run_data(
           simulation_run);
11     data->arrival_count++;
12
13     new_packet = (Packet_Ptr) xmalloc(sizeof(Packet));
14     new_packet->arrive_time = now;
15     new_packet->service_time = MEAN_STATION_SERVICE_TIME;
16     new_packet->status = WAITING;
17     new_packet->device_id = DEVICE_G;
18
19     /* Put the packet in the buffer at station. */
20     fifoqueue_put(data->station_queue, (void *) new_packet);
21
22     /* If this is the only packet at the station, transmit it).
23     It stays in the queue either way. */
24     if(fifoqueue_size(data->station_queue) == 1) {
25     /* Transmit the packet. */
26     schedule_transmission_start_event(simulation_run, now, (void
           *) new_packet);
27     }
28
29     /* Schedule the next packet arrival. */
30     schedule_packet_arrival_event_G(simulation_run,
31         now + exponential_generator((double) 1/PACKET_ARRIVAL_RATE
               ));
32 }
```

Listing 2: Station Transmission Event

```
1  void transmission_end_event(Simulation_Run_Ptr simulation_run,
       void *packet) {
2      Packet_Ptr this_packet;
3      Time now;
4      Simulation_Run_Data_Ptr data;
```

```
5
6       data = (Simulation_Run_Data_Ptr) simulation_run_data(
            simulation_run);
7
8       now = simulation_run_get_time(simulation_run);
9
10      this_packet = (Packet_Ptr) packet;
11
12      TRACE(printf("Success.\n"););
13
14      output_blip_to_screen(simulation_run);
15
16      /* Remove job from station queue and put into server queue */
17      fifoqueue_get(data->station_queue);
18      fifoqueue_put(data->server_queue, (void *) this_packet);
19
20      if(fifoqueue_size(data->server_queue) == 1) {
21      /* Execute job on server. */
22          schedule_data_transmission_start_event(simulation_run, now
                , fifoqueue_get(data->server_queue));
23      }
24
25      /* Transmit next job in station queue if queue contains job */
26      if(fifoqueue_size(data->station_queue) > 0) {
27          schedule_transmission_start_event(simulation_run, now,
                fifoqueue_see_front(data->station_queue));
28      }
29  }
```

Listing 3: Server Execution Event

```
1   void data_transmission_end_event(Simulation_Run_Ptr simulation_run
        , void *packet) {
2       Packet_Ptr this_packet, next_packet;
3       Time now;
4       Simulation_Run_Data_Ptr data;
5
6       data = (Simulation_Run_Data_Ptr) simulation_run_data(
            simulation_run);
7
8       now = simulation_run_get_time(simulation_run);
9
10      this_packet = (Packet_Ptr) packet;
11
12      TRACE(printf("Success.\n"););
```

```
13
14      /* Collect statistics. */
15      data->number_of_packets_processed++;
16
17      data->accumulated_delay += now - this_packet->arrive_time;
18      if (this_packet->device_id) {
19          data->b_delay += now - this_packet->arrive_time;
20          data->b_processed++;
21      } else {
22          data->g_delay += now - this_packet->arrive_time;
23          data->g_processed++;
24      }
25
26
27      output_blip_to_screen(simulation_run);
28
29      /* This packet is done. */
30      free((void *) packet);
31
32      /* See if there is another packet in the server queue. If so,
            begin execution. */
33      if (fifoqueue_size(data->server_queue) > 0) {
34          next_packet = fifoqueue_get(data->server_queue);
35
36          schedule_data_transmission_start_event(simulation_run, now
                , (void *) next_packet);
37      }
38  }
```