COMPENG 4DK4 Lab 4 Report

Aaron Pinto
pintoa9

Raeed Hassan
hassam41

November 14, 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 2

The mean delay versus arrival rate curves with `NUMBER_OF_STATIONS` set to 10 can be seen in Figure 1. The mean delay versus arrival rate curves with `NUMBER_OF_STATIONS` set to 5 can be seen in Figure 2. Both curves show an exponential increase in the mean delay up to a certain arrival rate, before quickly shooting up as the number of collisions rapidly increased. As we increase the `MEAN_BACKOFF_DURATION` we observe that the mean delay increases, however the mean delay also remains stable until a larger packet arrival rate before experiencing the rapid exponential growth.

Experiment 3

To modify the simulation to use a binary exponential backoff, the backoff duration during packet transmission was modified to use the backoff duration in Listing 1. As the backoff duration is no longer dependent on a `MEAN_BACKOFF_DURATION` anymore, there is only one mean delay curve versus arrival rate curve for any value of `NUMBER_OF_STATIONS`. The plot of the mean delay versus arrival rate for 5 and 10 stations can be seen in Figure 3. The curves display very similar behaviour, with the mean delay with 10 stations being larger as the packet arrival rate increases.

Listing 1: Binary Exponential Backoff

```
1 backoff_duration = uniform_generator() * pow(2.0, this_packet->
    collision_count);
```

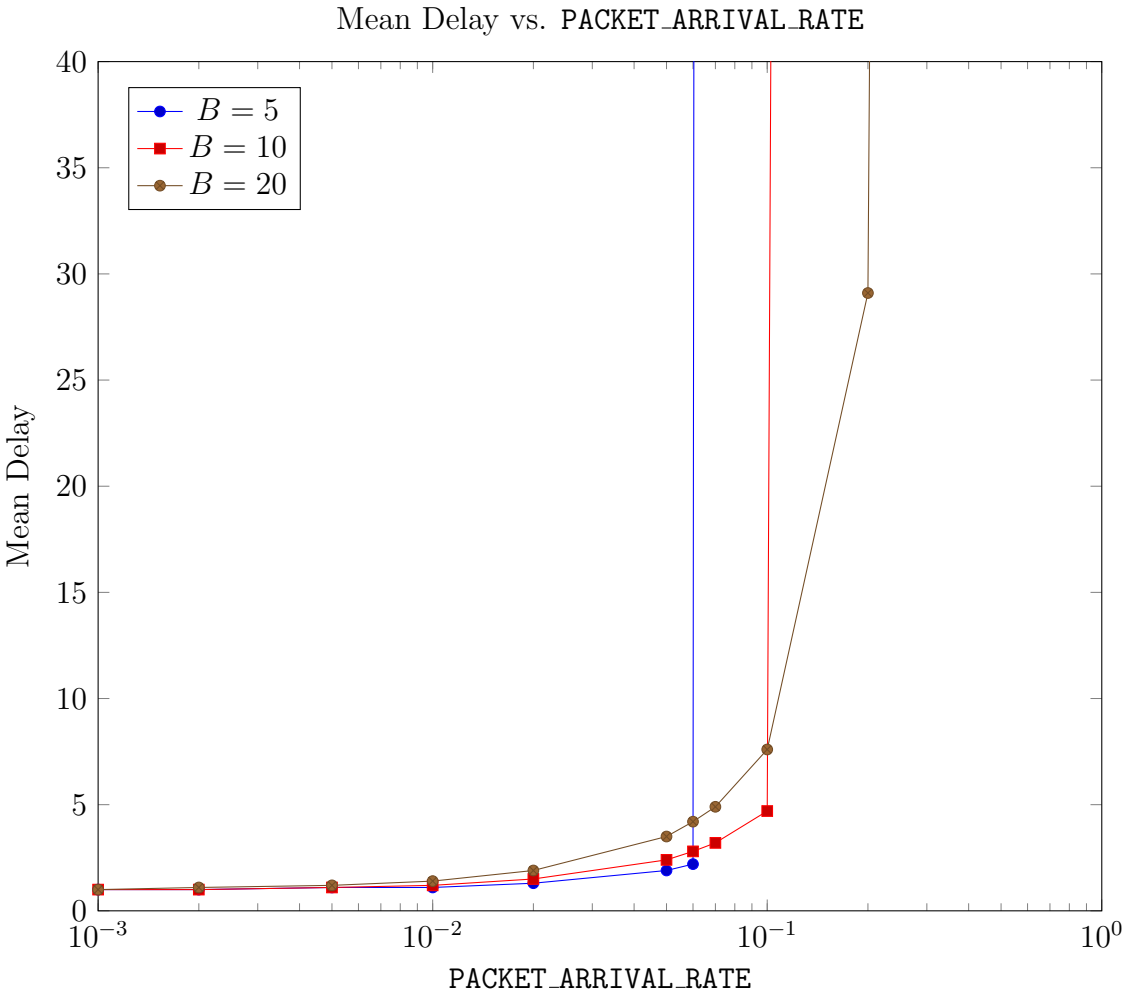

Figure 1: Experiment 2: Mean Delay vs. Packet Arrival Rate with 10 Stations

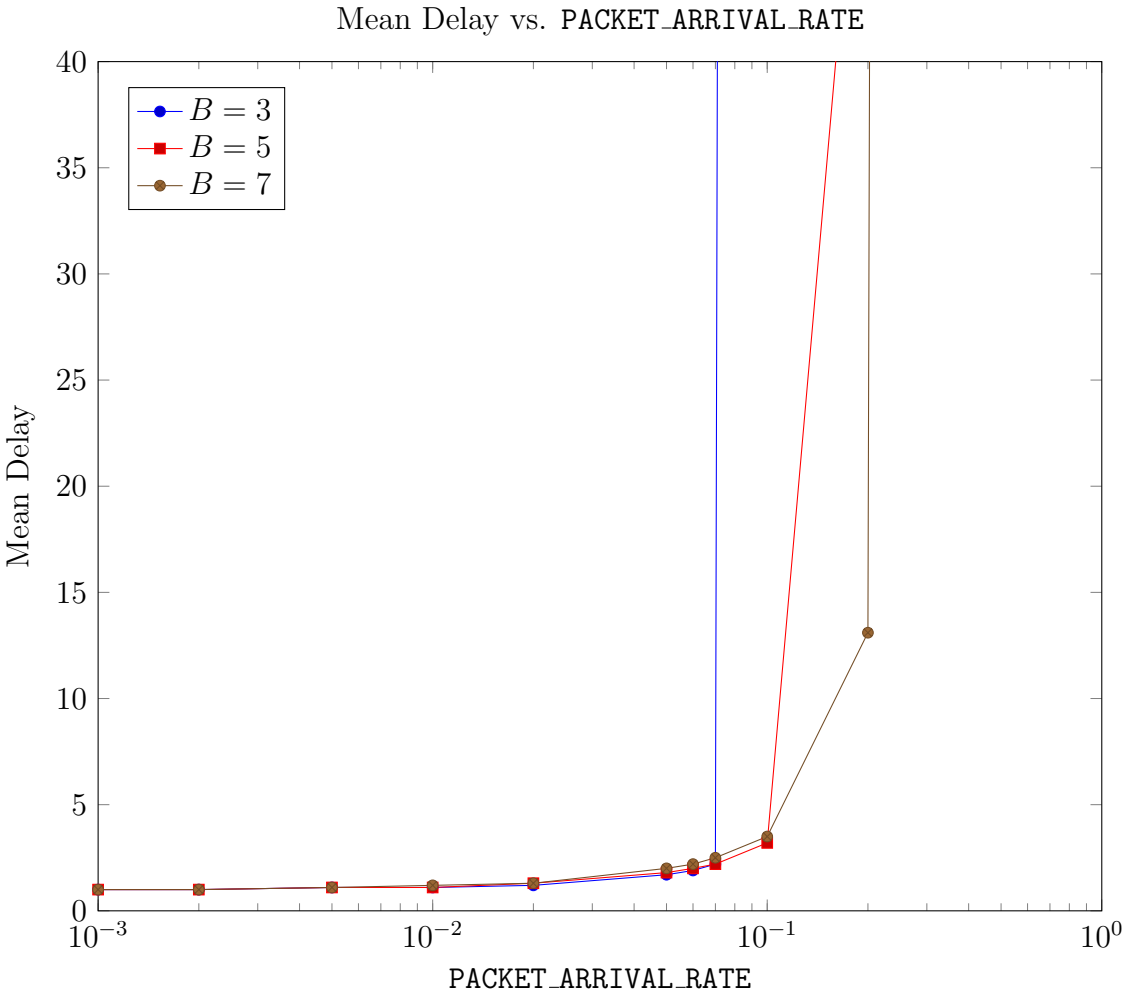

Figure 2: Experiment 2: Mean Delay vs. Packet Arrival Rate with 5 Stations

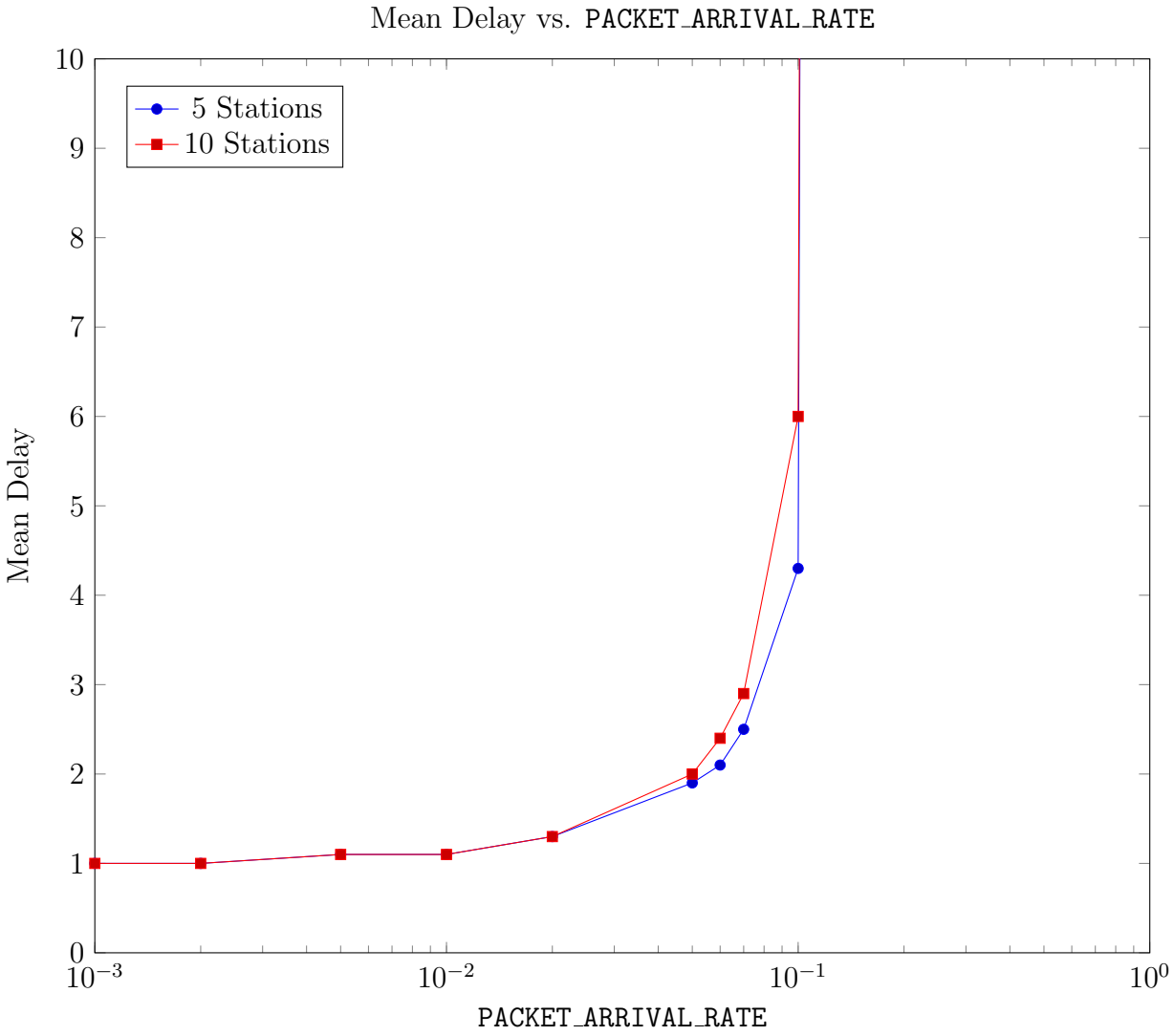

Figure 3: Experiment 3: Mean Delay vs. Packet Arrival Rate

Experiment 4

To modify the simulation from Part 3 so that one particular station always re-transmits an un-successful packet in the very next slot, the packet transmission code was modified so it would schedule the start transmission event immediately if the station ID matched the one that was set to persist, and backoff like in Part 3 otherwise. The listing for the modified code can be seen in Listing 2.

Listing 2: Binary Exponential Backoff with Single Persisting Channel

```
1 if (this_packet->station_id == 0) {
2     schedule_transmission_start_event(simulation_run, now, (void
          *) this_packet);
3 } else {
4     backoff_duration = uniform_generator() * pow(2.0, this_packet
          ->collision_count);
5     schedule_transmission_start_event(simulation_run, now +
          backoff_duration, (void *) this_packet);
6 }
```

The mean delay versus arrival rate curves for the persisting station, and the remaining channels can be seen in Figure 4. The behaviour for the other stations as the total number of stations match the behaviour seen in Part 3, however the mean delay for these stations at each packet arrival rate is slightly higher than the ones seen in Part 3. For the station that will always persist, the mean delay is significantly lower than the other stations, and will hit a peak mean delay before beginning to plateau instead of experiencing exponential growth in the mean delay like the other stations.

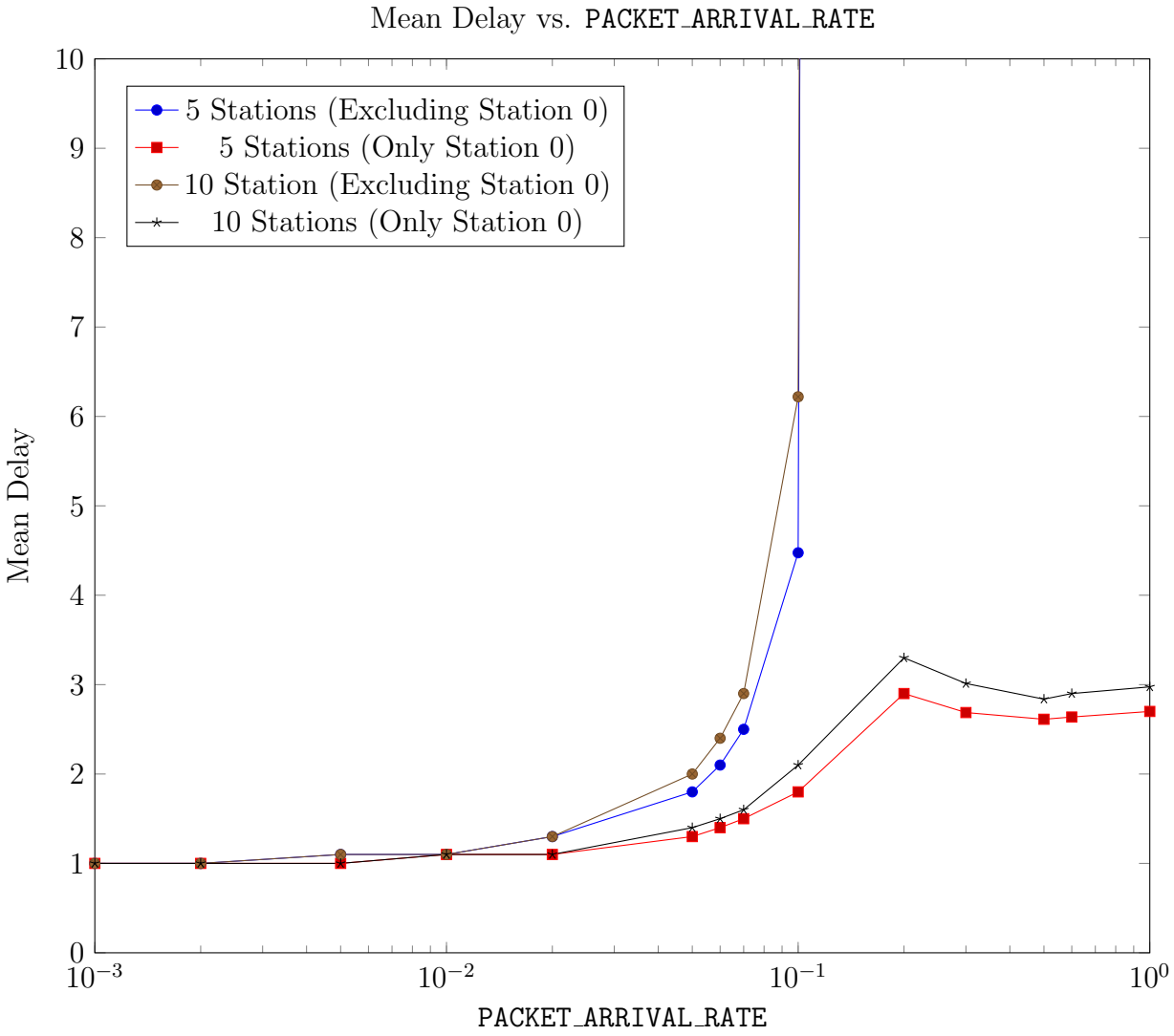

Figure 4: Experiment 4: Mean Delay vs. Packet Arrival Rate

Experiment 5

To implement the Slotted-ALOHA protocol, the packet arrival, packet transmission start and packet transmission backoff events were modified to only begin at ϵ seconds after or before time slot boundaries, and the binary exponential backoff from Part 3 was modified to equal a multiple of the slot duration. The modifications for the packet events can be seen in Listings 3–5. The simulations were performed with $\epsilon = 0.01$ with a slot duration of 1.02 seconds.

Listing 3: Packet Arrival

```
1 next_slot = SLOT_DURATION * ceil(simulation_run_get_time(
    simulation_run) / SLOT_DURATION) + EPSILON;sion_start_event(
    simulation_run, now + backoff_duration, (void *) this_packet);
```

Listing 4: Packet Transmission Start

```
1 /* Schedule the end of packet transmission event. */
2 schedule_transmission_end_event(simulation_run, SLOT_DURATION *
    ceil((simulation_run_get_time(simulation_run) + this_packet->
    service_time) / SLOT_DURATION) - EPSILON, (void *) this_packet)
    ;
```

Listing 5: Packet Transmission Backoff

```
1 backoff_duration = SLOT_DURATION * ceil(uniform_generator()*pow
    (2.0, this_packet->collision_count));
```

The plot of the mean delay versus arrival rate for 5 and 10 stations can be seen in Figure 5. The curves display very similar behaviour, with the mean delay with 10 stations being larger as the packet arrival rate increases. Compared to the performance of the system in Part 3, as the arrival rate increases the performances begin to converge, but the initial mean delay begins at a mean delay of 1.5 instead of 1 due the overhead of ϵ. To negate this effect, we tried smaller values of ϵ but found that very small values of ϵ were susceptible to causing a large number of collisions ultimately increasing the mean delay significantly.

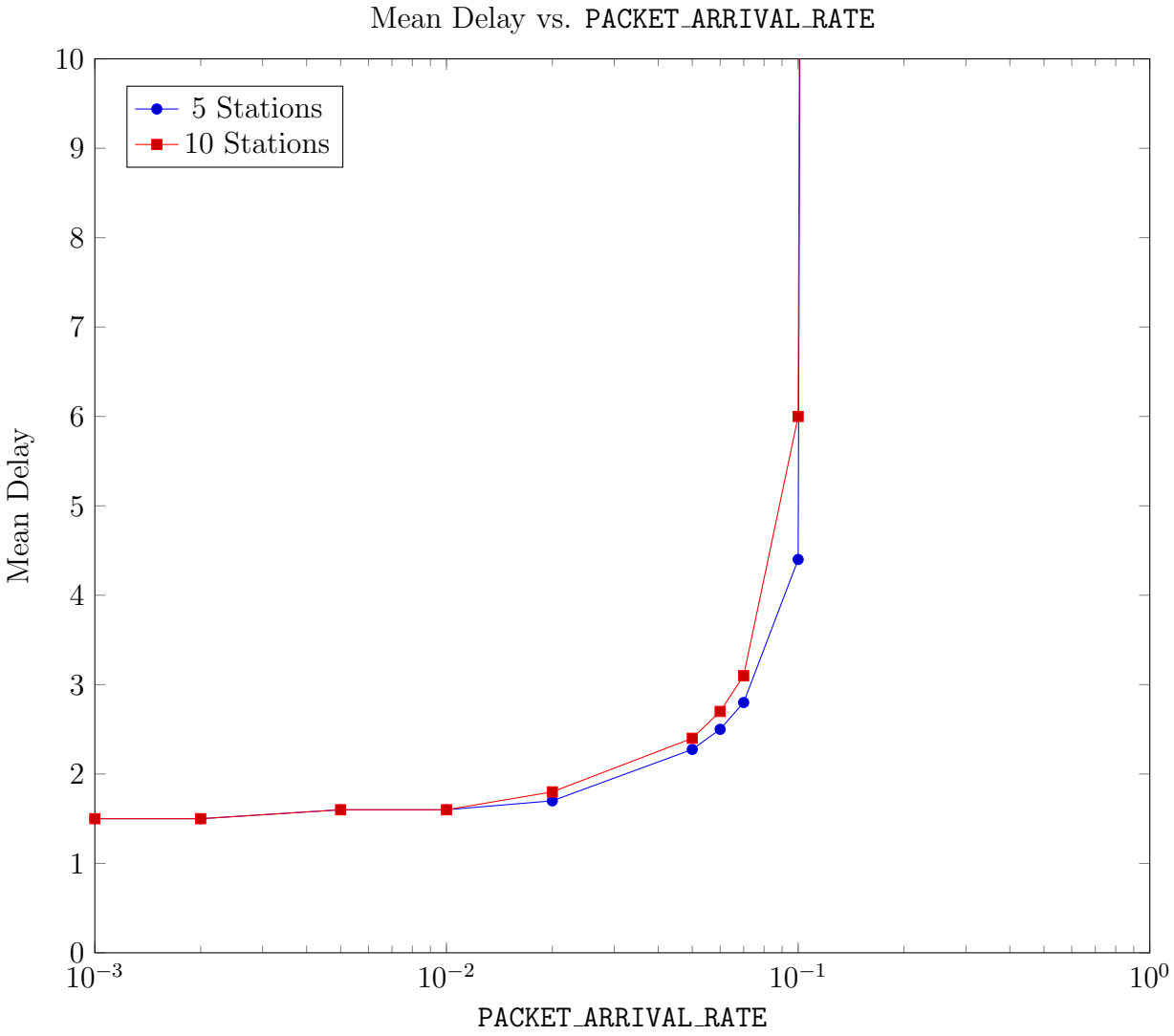

Figure 5: Experiment 5: Mean Delay vs. Packet Arrival Rate

Experiment 6

To simulate this simulation, we re-used the simulation from Part 5 as the S-ALOHA reservation channel, and created an additional channel for data transmission. When transmission successfully ends on the reservation channel, we schedule a start event for data packet transmission, which is nearly identical to the existing packet transmission function `schedule_transmission_start_event`. This eventually leads to an end transmission event for data packets, similar to the transmission functions already provided. The data packet end transmission event is shown in Listing 6.

Listing 6: Data Packet Transmission End Event

```
1  void data_transmission_end_event(Simulation_Run_Ptr simulation_run
       , void *packet) {
2      Packet_Ptr this_packet, next_packet;
3      Buffer_Ptr data_buffer;
4      Time now;
5      Simulation_Run_Data_Ptr data;
6      Channel_Ptr channel;
7
8      data = (Simulation_Run_Data_Ptr) simulation_run_data(
           simulation_run);
9      channel = data->data_channel;
10
11     now = simulation_run_get_time(simulation_run);
12
13     this_packet = (Packet_Ptr) packet;
14     data_buffer = data->data_channel_queue;
15
16     /* This station has stopped transmitting. */
17     decrement_transmitting_stn_count(channel);
18     set_channel_state(channel, IDLE);
19
20     TRACE(printf("Success.\n"););
21
22     /* Collect statistics. */
23     data->number_of_packets_processed++;
24
25     (data->stations + this_packet->station_id)->packet_count++;
26     (data->stations + this_packet->station_id)->accumulated_delay
           += now - this_packet->arrive_time;
27
28     data->number_of_collisions += this_packet->collision_count;
29     data->accumulated_delay += now - this_packet->arrive_time;
30     output_blip_to_screen(simulation_run);
31
```

```
32      /* This packet is done. */
33      free((void *) packet);
34
35      /* See if there is another packet in the data channel queue.
           If so, enable it for transmission. We will transmit
           immediately. */
36      if (fifoqueue_size(data_buffer) > 0) {
37          next_packet = fifoqueue_get(data_buffer);
38
39          schedule_data_transmission_start_event(simulation_run, now
               , (void *) next_packet);
40      }
41  }
```

The mean delay versus number of stations curve can be seen in Figure 6. The mean data packet duration, X, was set to 1 second, the packet arrival rate was set to 0.05, and the slot duration, X_r, was set to 0.1 seconds. The number of stations has no effect on the mean delay of the data packets in the system.

The mean delay versus mean data packet duration curve can be seen in Figure 7. The number of stations was set to 10 stations, the packet arrival rate was set to 0.05, and the slot duration, X_r, was set to 0.1 seconds. The mean delay experiences exponential increase as the mean data packet duration increases.

The mean delay versus packet arrival rate curve can be seen in Figure 8. The number of stations was set to 10 stations, the mean data packet duration, X, was set to 1 second, and the packet arrival rate was set to 0.05. The mean delay experiences exponential increase as the slot duration, X_r, increases.

The mean delay versus packet arrival rate curve can be seen in Figure 9. The number of stations was set to 10 stations, the mean data packet duration, X, was set to 1 second, and the slot duration, X_r, was set to 0.1 seconds. The mean delay experiences exponential increase as the mean data packet duration increases.

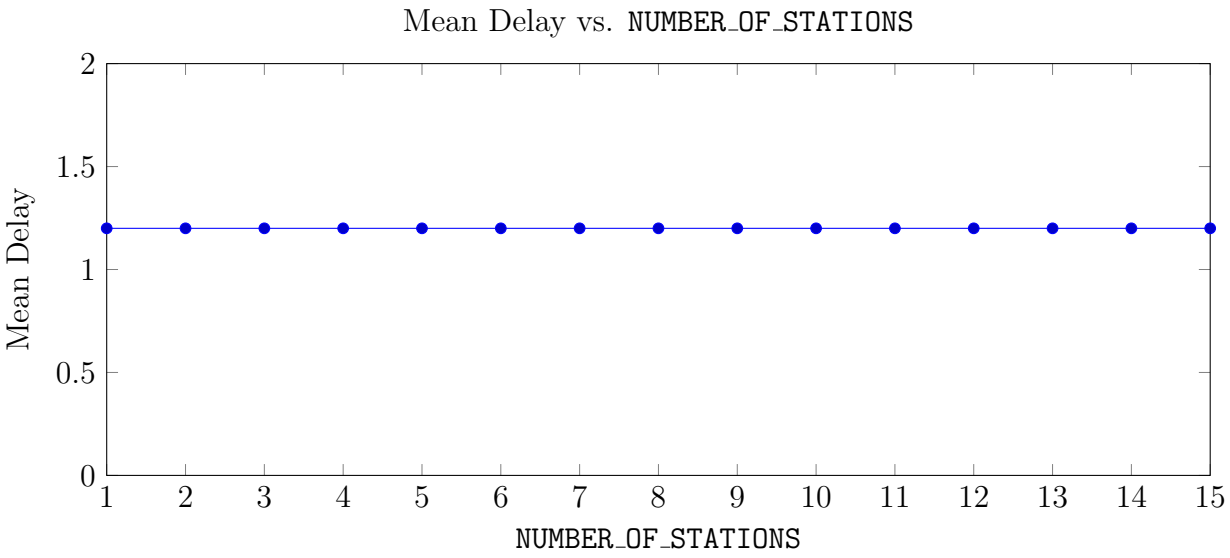

Figure 6: Experiment 6: Mean Delay vs. Number of Stations

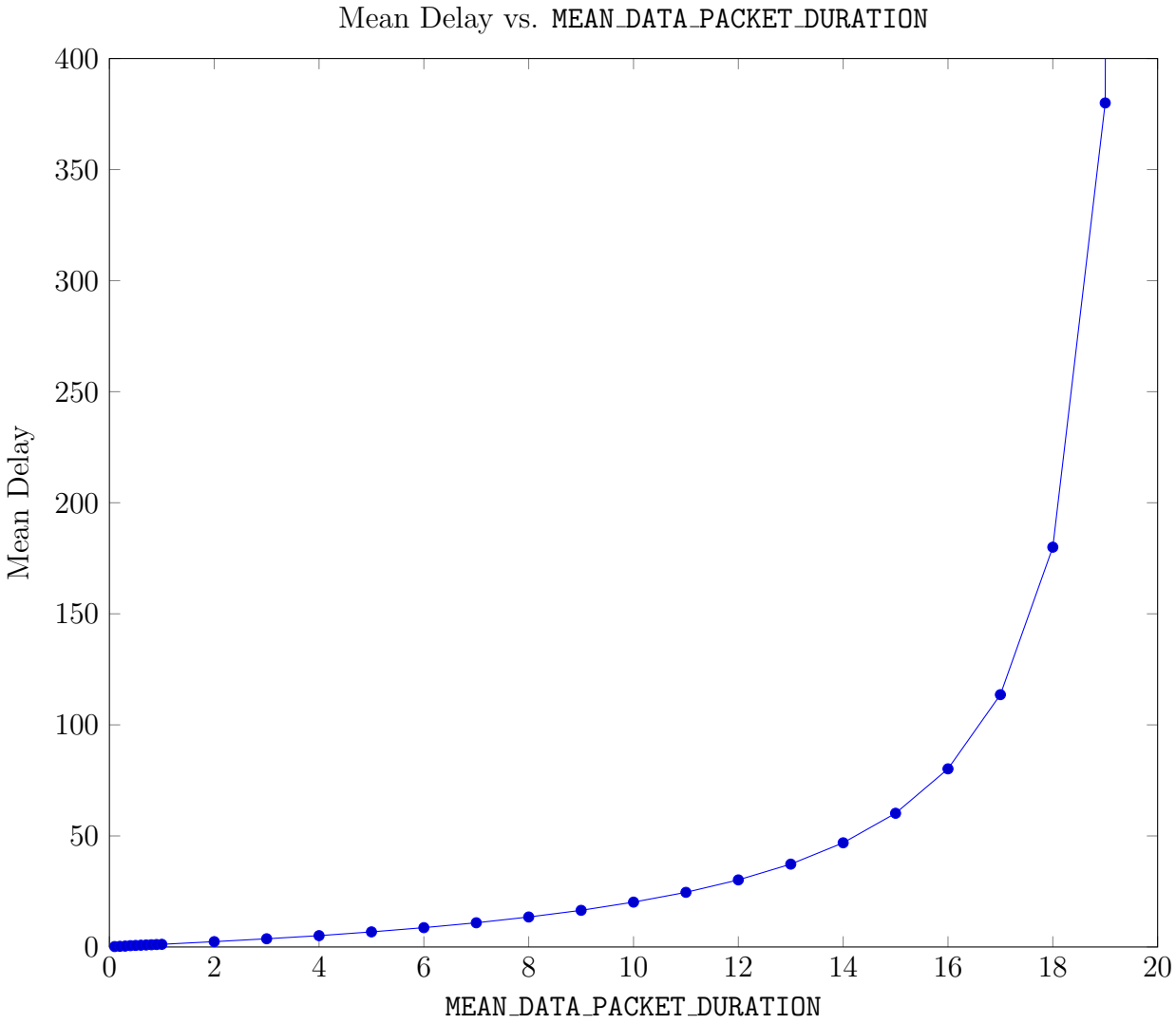

Figure 7: Experiment 6: Mean Delay vs. Mean Data Packet Duration, X

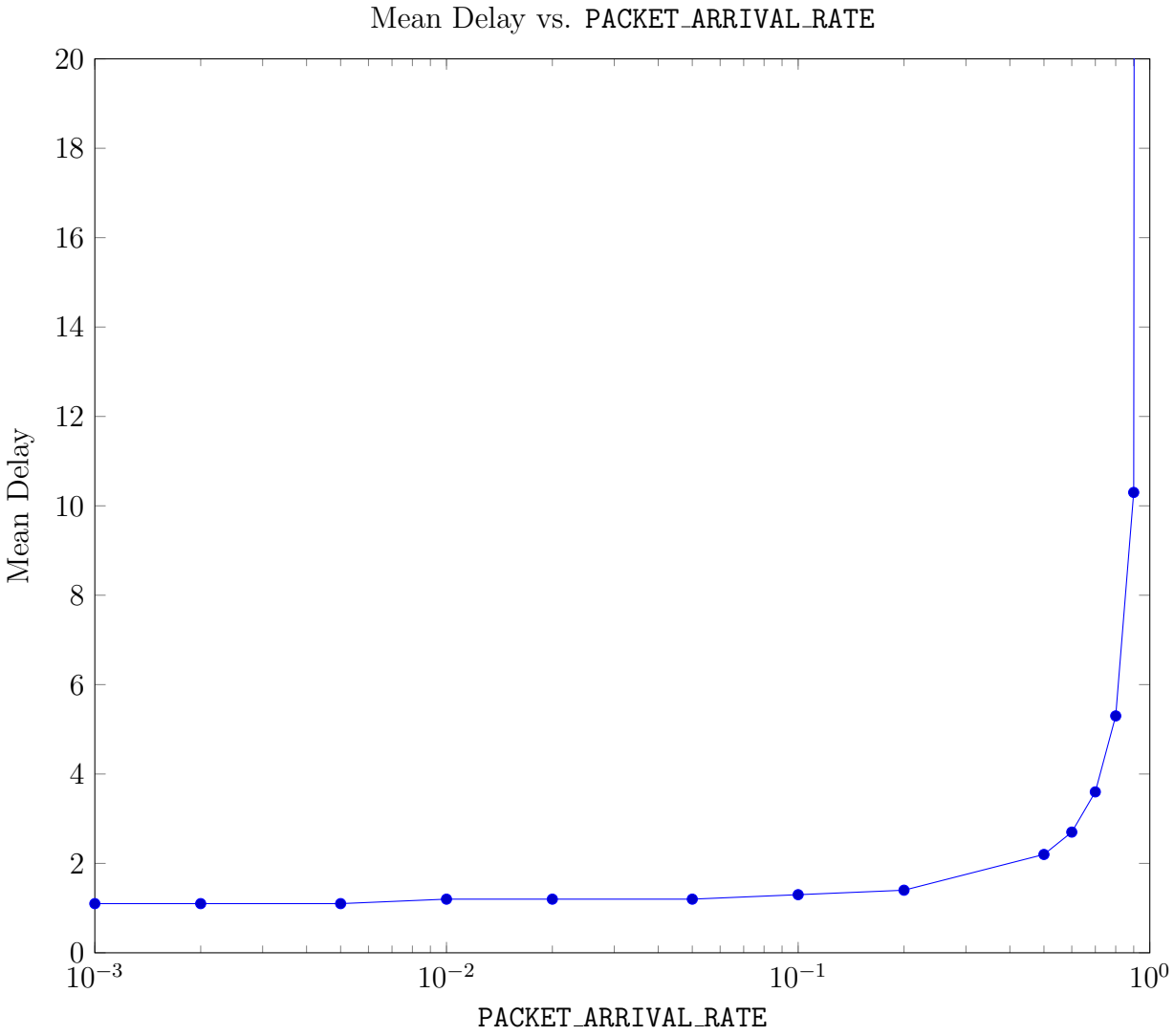

Figure 8: Experiment 6: Mean Delay vs. Packet Arrival Rate

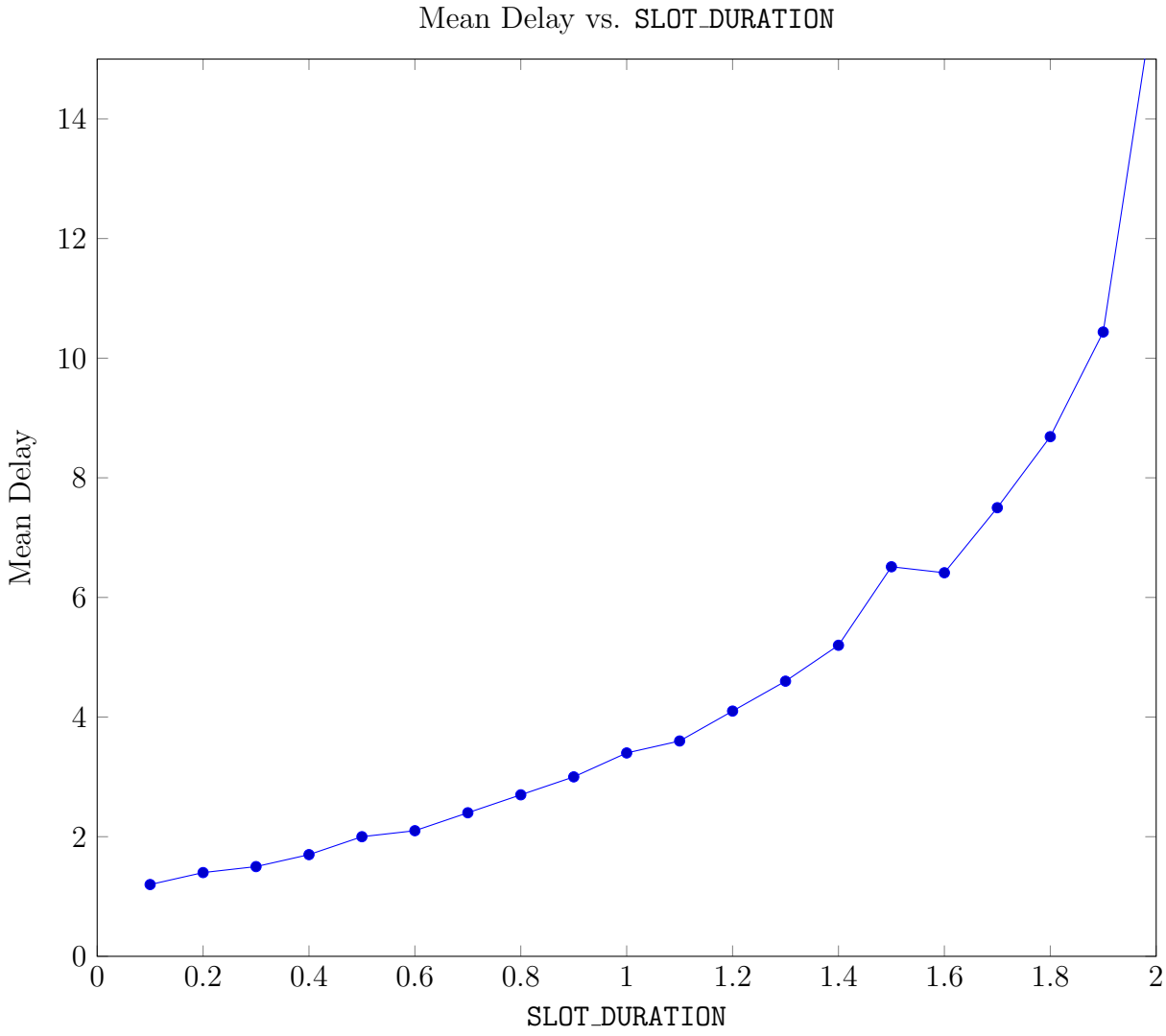

Figure 9: Experiment 6: Mean Delay vs. Slot Duration, X_r