COMPENG 4DK4 Lab 2 Report

Aaron Pinto
pintoa9

Raeed Hassan
hassam41

October 6, 2022

Random Number Generator Seeds

For the experiments in this lab, we used the same set of 18 random number seeds for all experiments. Experiment 2 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 9 different seeds from each our IDs, for a total of 18 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.

400188200	400190637
001882004	001906374
018820040	019063740
188200400	190637400
882004001	906374001
820040018	063740019
200400188	637400190
004001882	374001906
040018820	740019063

Table 1: Random Number Generator Seeds

Experiment 2

A plot of the mean delay vs. packet arrival rate is shown in Figure 1. At low packet arrival rate values, we see the mean delay approaches 0.5 msec. The mean delay axis intercept at these low packet arrival rates equal to the packet length divided by the link bit rate, in this experiment this is 500 bits divided by 1000 bits per msec, or 0.5 msec. Similar to Lab 1, we can observe that the mean delay begins to increase exponentially from this mean delay axis intercept value as we begin to increase the packet arrival rate.

Experiment 3

The code was modified to add a check to see if the delay for any packet is greater than 20 msec. The function used to check this is shown in Listing 1. The modified code was run with different values of `PACKET_ARRIVAL_RATE` λ to determine the maximum value of λ where the average probability was less than 2%. The maximum value determined as 402 packets per second, with the probability of a packet's delay exceeds 20 msec beginning to exceed 2% at 403 packets per second.

Listing 1: Modifications to Experiment 3 Code

```
void check_delay(Simulation_Run_Ptr simulation_run, double arr[])
    {
  Simulation_Run_Data_Ptr data;

```

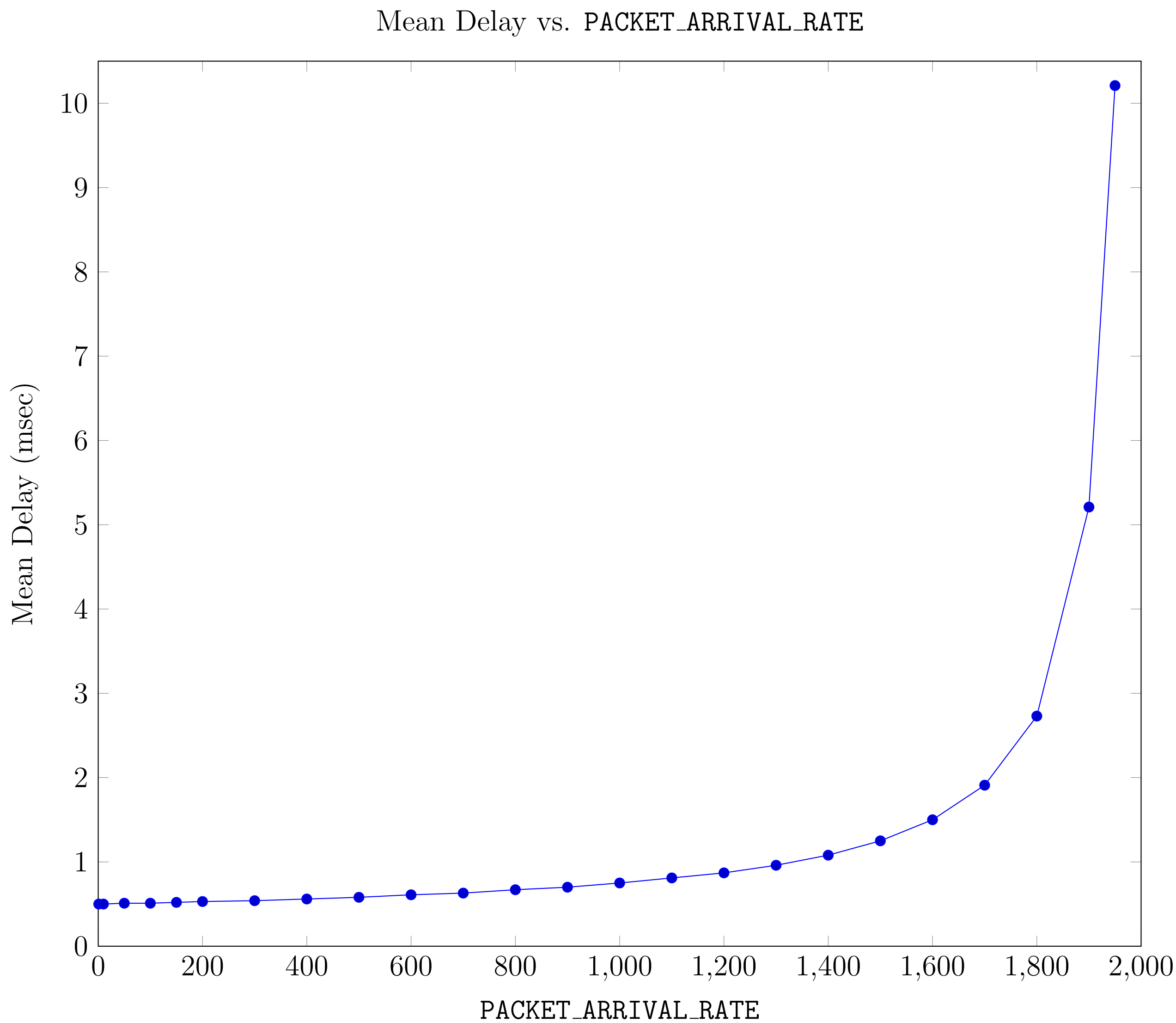

Figure 1: Experiment 2: Mean Delay vs. Packet Arrival Rate

```
    data = (Simulation_Run_Data_Ptr) simulation_run_data(
        simulation_run);

    // Get current packet and time
    double a = 1e3 * data->accumulated_delay;
    int b = data->number_of_packets_processed;

    // Check if num packets has changed
    if (arr[1] + 1 == b) {
        // Check if change in time is greater than 20 (packet
            delay > 20 msec)
        if (a > arr[2] + 20) {
            arr[0]++;
        }

        // Update with new packet and time
        arr[1] = b;
        arr[2] = a;
    }
}
```

Experiment 4

To achieve this M/D/2 queueing system, the `packet_arrival_event` in `packet_arrival.c` was modified with the changes in Listing 2 which adds an additional link. A plot of the mean delay vs. packet arrival rate is shown in Figure 2. At low packet arrival rate values, we see the mean delay approaches 1.0 msec. The mean delay axis intercept at these low packet arrival rates equal to the packet length divided by the link bit rate, in this experiment this is 500 bits divided by 500 bits per msec, or 1.0 msec. If we compare it to experiment 2, we can see that adding the 2nd link decreases the rate of mean delay growth as the packet arrival rate values increase. Although, similar to Lab 1, we can observe that the mean delay growth is still exponential as we begin to increase the packet arrival rate.

Listing 2: Modifications to Experiment 4 Code

```
if (server_state(data->link1) == BUSY) {
    if (server_state(data->link2) == BUSY) {
        fifoqueue_put(data->buffer, (void *) new_packet);
    } else {
        start_transmission_on_link(simulation_run, new_packet,
            data->link2);
    }
} else {
    start_transmission_on_link(simulation_run, new_packet, data->
        link1);
```

```
}
```

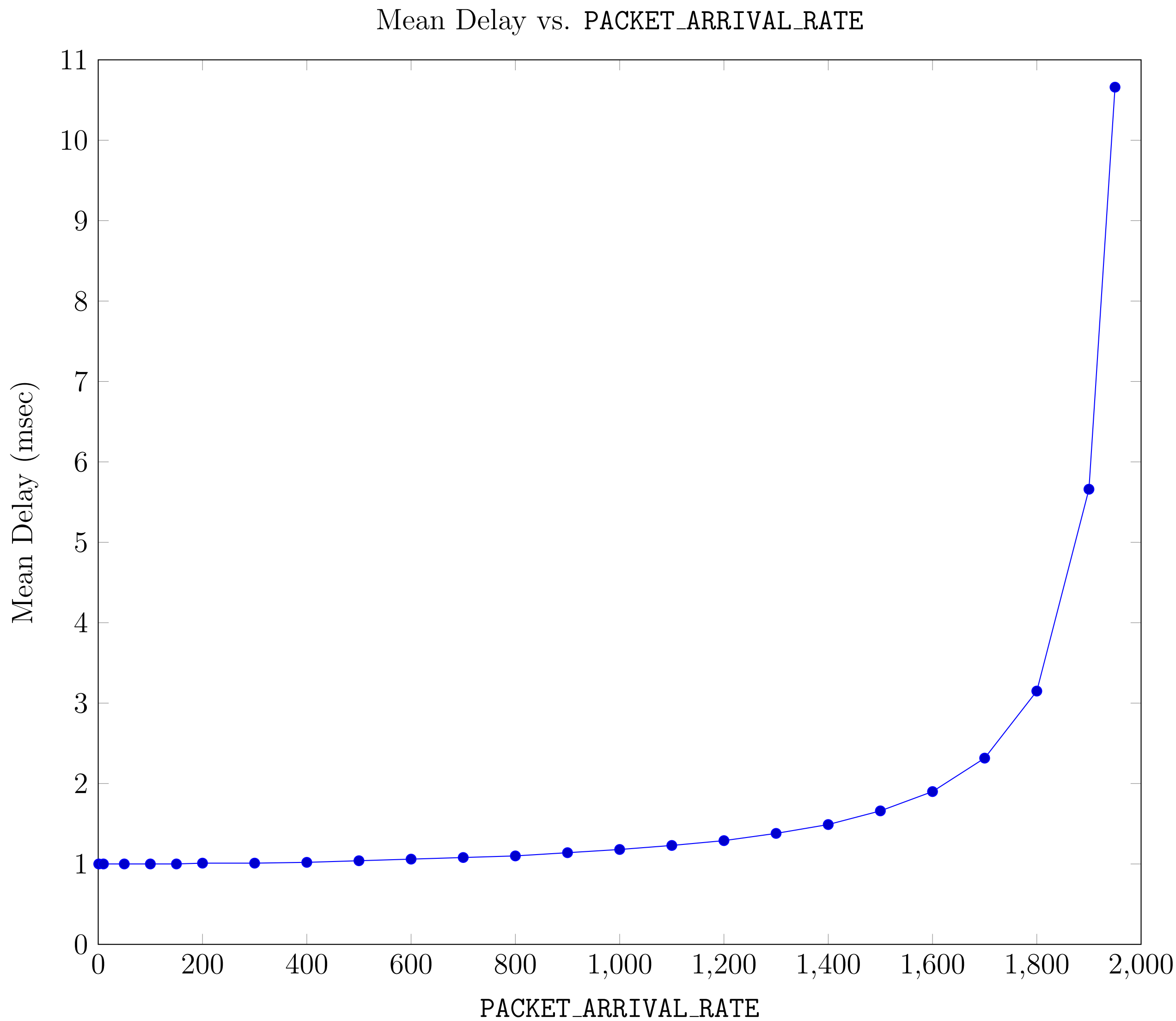

Figure 2: Experiment 4: Mean Delay vs. Packet Arrival Rate

Experiment 5

Experiment 6

The packet length of each G.711 encoded voice packet was determined to be 1792 bits per packet, or 224 bytes per packet. As the voice packets arrive every 20 ms, each voice packet must contain 20 ms of the voice payload. Therefore, each packet contains a 62 byte header and 160 bytes of the voice payload (20 ms at 64 Kbps = 1280 bits or 160 bytes).

The simulation was modified to add this additional voice packet. A new arrival event for the voice stream was created, with the code shown in Listing 3. Additional changes were made to the program to log this additional data (separating the delay and number of packets processed for data packets and voice packets).

A graph comparing the voice stream delay and data packet delay is shown in Figure 3. Both plots show an exponential curve similar to what is seen in Experiment 2. For the data packet delay, the exponential curve is expected as voice stream packets will transmitted quickly (relative to the transmission of the data packet, for which the mean transmission time is over 20 times larger than the voice stream packet transmission time) if they are in queue, meaning the data packet mean delay is relatively unaffected by this and will mostly exhibit the exponential behaviour expected for the Poisson distributed arrivals. For the data stream packets, the exponential curve is expected as these packets arrive slower than the mean service time of the data packets leading to these packets being stuck in queue longer as the packet arrival rate decreases. Therefore the mean delay curve for the voice stream packets follows the same curve as the data packet delay curve, but is about 40 msec below the packet delay curve.

Listing 3: Modifications to Experiment 6 Code

```
void voice_packet_arrival_event(Simulation_Run_Ptr simulation_run, void *ptr) {
        Simulation_Run_Data_Ptr data;
        Packet_Ptr new_packet;

        data = (Simulation_Run_Data_Ptr) simulation_run_data(simulation_run);
        data->arrival_count++;

        new_packet = (Packet_Ptr) xmalloc(sizeof(Packet));
        new_packet->arrive_time = simulation_run_get_time(simulation_run);
        new_packet->service_time = get_voice_packet_transmission_time();
        new_packet->status = WAITING;
        new_packet->packet_type = 1;

        /*
        * Start transmission if the data link is free. Otherwise put the packet into
        * the buffer.
        */

        if (server_state(data->link) == BUSY) {
                fifoqueue_put(data->buffer, (void *) new_packet);
        } else {
                start_transmission_on_link(simulation_run, new_packet, data->link);
```

```
        }

        /*
        * Schedule the next packet arrival. Independent, exponentially distributed
        * interarrival times gives us Poisson process arrivals.
        */

        schedule_voice_arrival_event(simulation_run,
           simulation_run_get_time(simulation_run) + 0.2);
}
```

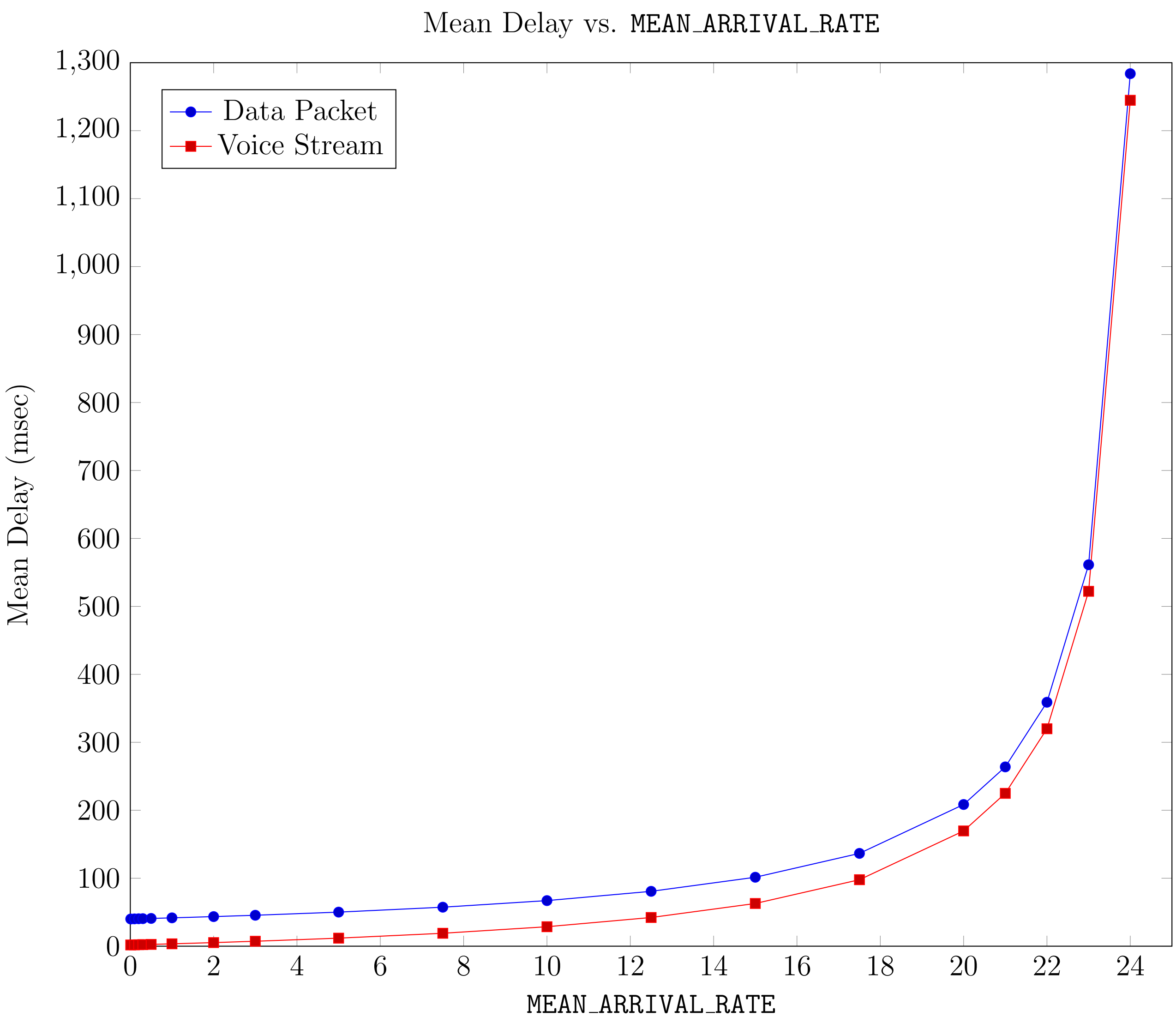

Figure 3: Experiment 6: Mean Delay vs. Packet Arrival Rate

Experiment 7