Sarah Anderson & Denise Linn

Dr. Walt Ligon

ECE 2230: Computer Systems Engineering

December 2, 2019

Airport Simulation Part 2

For this part of the project we added simulations queues to Part 1 of the Airport Project. We also made a few changes to part one of the project such as setting the global time in the event_cause() function instead of updating the time in main. We added a size argument to the event_init function to determine the size of the priority queue. The final change we made was to move all of the existing time functions and add any needed ones to time.c. When this project is finished it should show passengers advancing through each stage of the airport. The passengers should enter queues when they reach the airline desk, the ID check, the security check, and the train terminal. They will be removed from the queue when they reach the front of the line. All of the queues will be very similar except the scan queue which will be implemented using four different queues instead of one queue.

In order to add the simulations queues we needed to add the queue functions to main.c. For example, in the AirlineQ case of the switch statement we create the first queue by calling the queue_init function. Then we check to see if the queue is full using the queue_full() function. If the queue is not full, which will always be the case since we are using the linked list implementation, the passenger is added to the queue. Next, we check to see if the passenger that was just added is the only passenger in line. If they are then they advance to the front of the line immediately. In order to simulate this the event type is changed to EV_AIRLINE, which is the next stage the passenger will get to when they leave the event queue. We also set the event time to time_airline() which is the amount of time the passenger will spend at the desk of the airline. Finally, we schedule this event.

If the passenger is not the only person in the line, then they are inserted into the queue and their event is freed. When the passengers are waiting in line they are not a part of an event. When the person in front of the passenger reaches the front of the line they will trigger the EV_AIRLINE event for the passenger.

When a passenger reaches the front of the line they are removed from the queue. If there is someone behind them a new event will be created for that passenger. The event type is AIRLINE and the event time is time_airline(). Then the newly created event is scheduled allowing the person who is now at the front of the line to eventually exit the queue. For the passenger that was talking to the airline representative, their event type is changed to the next stage of the airport, EV_IDQ. Their event time is set to time_idq and the event is scheduled.

This process repeated for the ID queue and the train queue. The scan queue is implemented using 4 different queues. In order to do this 4 scan queues are initialized in an array. The main change to the queue process when there are multiple queues comes when adding passengers to the queue. Using if statements, we find the scan queue with the smallest number of passengers and add the new passenger to that queue. If all of the queues are the same size then the passenger will be added to the first queue. Once the passenger has been added to queue it is dealt with in much the same way as the single queues. If it is the first passenger it will immediately advance to the front of the line and if not the person in front of them will trigger their next event.

At the end of the code we free all of the queues that we initialized in the function as well as all of the events we malloced in the switch statements. For this program also we made a makefile, so to compile the program you type “make” and to run the program you type “make run”.