Question 13.2

In this problem you can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute (i.e., mean interarrival rate ≣1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate ≣2 = 0.75 minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute). Use the Arena software (PC users) or Python with SimPy (PC or Mac users) to build a simulation of the system, and then vary the number of ID/boarding-pass checkers and personal-check queues to determine how many are needed to keep average wait times below 15 minutes. [If you're using SimPy, or if you have access to a non-student version of Arena, you can use λ1 = 50 to simulate a busier airport.]

To simulate an airport security checkpoint and determine how many servers are needed to keep average wait times under 15 minutes.

The simulation:

Passengers arrive to the ID check queue based on a Poisson distribution with average 5 per minute. The ID check servers process passengers at a rate of 0.75 minutes on average. After ID check, passengers go to the shortest of multiple security check queues. Security scanning takes a random uniform time of 0.5 to 1 minute. The simulation tests different numbers of ID and security servers. It runs multiple times and calculates the average wait time. What was achieved:

I built a SimPy model to simulate the passenger arrival and two queues. The code loops through different combinations of server counts. It prints out the average wait time for each scenario. The optimal number of servers to keep wait times under 15 minutes was determined. In summary, a discrete event simulation modeled the airport queues and tested different resource levels to see how it impacted average passenger wait times. The goal was to find the right amount of servers to keep delays acceptable.

In [23]:
```
# Import necessary modules
import simpy  # SimPy simulation library
import random  # Python's built-in random module

# Set constants for the simulation
numCheckers = 35 # smallest number of boarding-pass checkers
```

```
numScanners = 35 # smallest number of scanners

arrRate = 50 # arrival rate (passengers per minute)
#checkRate is set to 0.75, which means that, on average, it takes 0.75 minut
checkRate = 0.75 # boarding-pass check rate (minutes per passenger)
#minScan = 0.5: This line sets minScan to 0.5. This value represents the min
minScan = 0.5 # scanner minimum time for uniform distribution
#maxScan = 1.0: This line sets maxScan to 1.0. This value represents the max
maxScan = 1.0 # scanner maximum time for uniform distribution
#runTime = 720: This line sets runTime to 720. This value represents the tot
runTime = 720 # run time (minutes) per simulation
#replications = 100: This line sets replications to 100. This value represen
replications = 100 # number of replications

# Initialize global variables to store average times for different scenarios
#avgCheckTime, for example, is a 2D array where the first dimension represen
avgCheckTime = [[0.0 for i in range(6)] for j in range(6)] # average boardin
#Similarly, avgScanTime, avgWaitTime, and avgSystemTime follow the same stru
avgScanTime = [[0.0 for i in range(6)] for j in range(6)] # average scan tim
avgWaitTime = [[0.0 for i in range(6)] for j in range(6)] # average total wa
avgSystemTime = [[0.0 for i in range(6)] for j in range(6)] # average total

# Create a simulation model using SimPy
class System(object):
    def __init__(self,env,i,j):
        self.env = env
        self.checker = simpy.Resource(env,i+numCheckers) # define number of
        self.scanner = [] # define a set of scanners with 1 each; needed bec
        for i in range(j+numScanners):
            self.scanner.append(simpy.Resource(env,1))

    # define boarding-pass check time (exponential)
    def check(self,passenger):
        # For some reason in python, expovariate actually uses 1 over the me
        yield self.env.timeout(random.expovariate(1.0/checkRate))

    # define scan time (uniform)
    def scan(self,passenger):
        yield self.env.timeout(random.uniform(minScan,maxScan))

# Passenger process through system
#env: This parameter represents the simulation environment created using Sim
#name: This parameter represents the name or identifier of the passenger bei
#s: This parameter represents an instance of the System class. The System cl
#i: This parameter represents the number of boarding-pass checkers to be use
#j: This parameter represents the number of scanners to be used in the curre
#pnum: This parameter likely represents a unique identifier or index for the

def passenger(env,name,s,i,j,pnum):

    # access global variables to be able to modify them
    global checkWait
    global scanWait
```

```
    global sysTime
    global totThrough

    timeArrive = env.now # Record the arrival time of the passenger


    # print('%s arrives at time %.2f' % (name,timeArrive))
    # Boarding-pass check process
    with s.checker.request() as request:
        # print('check queue length = %d' % len(s.checker.queue))
        yield request # request a checker
        tIn = env.now # note when passenger starts being checked
        yield env.process(s.check(name)) # call check process
        tOut = env.now # note when passenger ends being checked
        checkTime[pnum] = (tOut - tIn) # calculate total time for passenger

    # Find the shortest scanner queue (note: scanners are numbered 0 through
    minq = 0
    for k in range(1,j+numScanners):
        if (len(s.scanner[k].queue) < len(s.scanner[minq].queue)):
            minq = k

    # print('scanner queue %d lengths = %d' % (minq,len(s.scanner[minq].queu

    # Go through scanner queue
    with s.scanner[minq].request() as request: # use scanner number minq (th
          yield request # request the scanner
          tIn = env.now # note when passenger starts being scanned
          yield env.process(s.scan(name)) # call scan process
          tOut = env.now # note when passenger ends being scanned
          scanTime[pnum] = (tOut - tIn) # calculate total time for passenger

    timeLeave = env.now # note time passenger finishes
    sysTime[pnum] = (timeLeave - timeArrive) # calculate total time in syste
    totThrough += 1 # count another passenger who got through the system


# Passenger arrival process

def setup(env,i,j):
    k = 0
    s = System(env,i,j)
    while True: # keep doing it (until simulation ends)
        yield env.timeout(random.expovariate(arrRate)) # find tieme until ne
        k += 1 # count one more passenger

        # send the passenger through its process
        env.process(passenger(env,'Passenger %d' % k,s,i,j,k)) # name the pa


# Run the simulation for different numbers of checkers and scanners
for i in range(6): # number of boarding-pass checkers
    for j in range(6): # number of scanners
```

```
        # for each replication
        for k in range(replications):

            # choose random seed
            random.seed(k)

            # create environment
            env = simpy.Environment()

            # initialize global variables
            totThrough = 0
            checkTime = [0.0] * int(arrRate*runTime*1.5)
            scanTime = [0.0] * int(arrRate*runTime*1.5)
            sysTime = [0.0] * int(arrRate*runTime*1.5)

            # run the simulation
            env.process(setup(env,i,j)) # start passenger arrival process
            env.run(until=runTime) # run for runTime simulated minutes

            # Calculate average times for this replication

            # print('%d : Replication %d times %.2f %.2f %.2f' % (totThrough

            avgSystemTime[i][j] += (sum(sysTime[1:totThrough]) / totThrough)
            avgCheckTime[i][j] += (sum(checkTime[1:totThrough]) / totThrough
            avgScanTime[i][j] += (sum(scanTime[1:totThrough]) / totThrough)

        avgWaitTime[i][j] = (avgSystemTime[i][j] - avgCheckTime[i][j] - avgS


        # Calculate overall averages across all replications

        avgSystemTime[i][j] /= replications
        avgCheckTime[i][j] /= replications
        avgScanTime[i][j] /= replications
        avgWaitTime[i][j] /= replications

        print('----- %d -- %d -----' % (i+35,j+35))
        print('Average system time = %.2f' % avgSystemTime[i][j])
        print('Average check time = %.2f' % avgCheckTime[i][j])
        print('Average scan time = %.2f' % avgScanTime[i][j])
```

```
----- 35 -- 35 -----
Average system time = 27.92
Average check time = 0.75
Average scan time = 0.75
----- 35 -- 36 -----
Average system time = 25.95
Average check time = 0.75
Average scan time = 0.75
----- 35 -- 37 -----
Average system time = 25.97
```

```
Average check time = 0.75
Average scan time = 0.75
----- 35 -- 38 -----
Average system time = 26.21
Average check time = 0.75
Average scan time = 0.75
----- 35 -- 39 -----
Average system time = 26.19
Average check time = 0.75
Average scan time = 0.75
----- 35 -- 40 -----
Average system time = 26.11
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 35 -----
Average system time = 26.60
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 36 -----
Average system time = 18.12
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 37 -----
Average system time = 17.34
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 38 -----
Average system time = 16.76
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 39 -----
Average system time = 16.75
Average check time = 0.75
Average scan time = 0.75
----- 36 -- 40 -----
Average system time = 16.65
Average check time = 0.75
Average scan time = 0.75
----- 37 -- 35 -----
Average system time = 26.08
Average check time = 0.75
Average scan time = 0.75
----- 37 -- 36 -----
Average system time = 16.78
Average check time = 0.75
Average scan time = 0.75
----- 37 -- 37 -----
Average system time = 9.53
Average check time = 0.75
Average scan time = 0.75
----- 37 -- 38 -----
Average system time = 8.15
Average check time = 0.75
```

```
Average scan time = 0.75
----- 37 -- 39 -----
Average system time = 7.93
Average check time = 0.75
Average scan time = 0.75
----- 37 -- 40 -----
Average system time = 8.01
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 35 -----
Average system time = 26.11
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 36 -----
Average system time = 16.56
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 37 -----
Average system time = 7.84
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 38 -----
Average system time = 3.82
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 39 -----
Average system time = 3.47
Average check time = 0.75
Average scan time = 0.75
----- 38 -- 40 -----
Average system time = 3.41
Average check time = 0.75
Average scan time = 0.75
----- 39 -- 35 -----
Average system time = 26.07
Average check time = 0.75
Average scan time = 0.75
----- 39 -- 36 -----
Average system time = 16.62
Average check time = 0.75
Average scan time = 0.75
----- 39 -- 37 -----
Average system time = 7.64
Average check time = 0.75
Average scan time = 0.75
----- 39 -- 38 -----
Average system time = 3.04
Average check time = 0.75
Average scan time = 0.75
----- 39 -- 39 -----
Average system time = 2.65
Average check time = 0.75
Average scan time = 0.75
```

```
----- 39 -- 40 -----
Average system time = 2.54
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 35 -----
Average system time = 25.79
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 36 -----
Average system time = 16.75
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 37 -----
Average system time = 7.49
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 38 -----
Average system time = 2.97
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 39 -----
Average system time = 2.47
Average check time = 0.75
Average scan time = 0.75
----- 40 -- 40 -----
Average system time = 2.38
Average check time = 0.75
Average scan time = 0.75
```

Using SimPy to analyze an airport security checkpoint. The goal was to determine the number of boarding pass checkers and security scanners needed to keep average passenger wait times under 15 minutes.

I modeled the system as passengers arriving according to a Poisson process to a boarding pass check queue. The checkers had exponential service times. After boarding pass check, passengers went to the shortest of multiple security queues with uniform scanner service times.

I implemented the model, defining classes for the system and passenger processes. Key parameters like arrival rate, number of servers, and service time distributions were set up as constants.

Please not that I used $\lambda 1 = 50$ to simulate a busier airport.

The simulation looped through different combinations of boarding pass checkers (35-40) and security scanners (35-40). For each combination, 100 replications were run to account for randomness. Data was collected on average time in system, at check, and at scanner for each passenger.

The output shows the average times across the 100 replications for each checker/scanner combination. The results clearly demonstrate that 35 checkers and 35 scanners are insufficient, with average system times exceeding 25 minutes.

Increasing to 36 checkers reduces system time to around 17 minutes regardless of scanners. Further increasing checkers lowers times, with 37 checkers and 38 scanners reducing average system time to 3.82 minutes.

So in summary, to meet the goal of less than 15 minute average system time, the model estimated that approximately 37 boarding pass checkers and 38 security scanners are needed for the simulated passenger arrival rate. The simulation allowed efficiently identifying a resource level combination that kept delays acceptable.

In [ ]:

```

```