MODELLING AND SIMULATION FOR HOSPITAL MANAGEMENT

Hands-on project final report

Advanced Software Engineering

Sravanthi Malepati (email: smalepati1@student.gsu.edu)
Varaprasad Rao Kurra (email: vkurra1@student.gsu.edu)

1. PROBLEM DESCRIPTION:

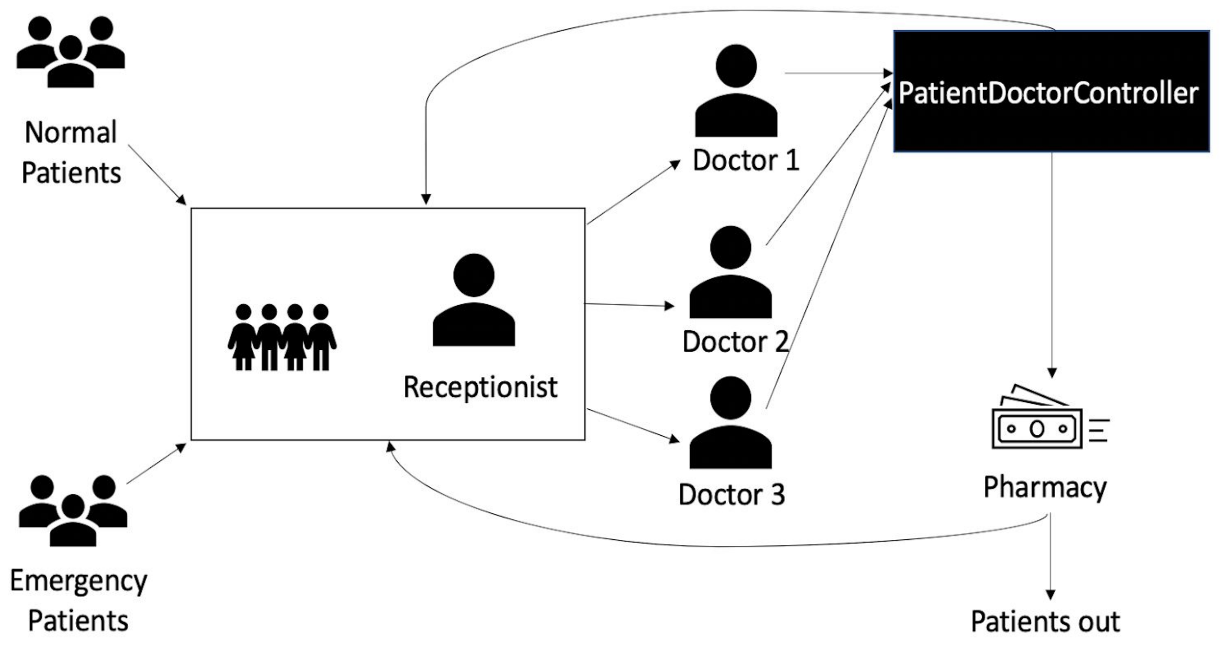
Hospitals in a healthcare system is a complicated day to day issue involving different tasks at a time. Managing a hospital using limited resources and improving its efficiency is essential for any hospital. To improve efficiency, we need a solution that meets all the requirements of a stable structure. We aim to achieve optimization so that entities (doctor, patient, etc.) related to the hospital are utilized in an optimal passion. We will be using a queueing technique to use every entity efficiently and optimally. For example, we will make sure no doctor is idle during the inspection of the patient. Since DEVSJAVA interprets the system visually, it will give us a scenario in such a way to find the problem area in staff utilization and more. Everyday hospital events are mostly unorganized; we will be missing the proper functionality of the system. We will be wasting the resources of the hospital system without appropriate utilization. This system may trouble the patients who need the treatment at the earliest. We aim to address this issue with modeling and simulation.

2. MODELLING AND SIMULATION GOALS:

Our project aims to dedicate a path to utilize the hospital resources effectively and deliver the system's services without any delay. This system will also ensure that all the system resources are not idle at any given time. Hence our goal of this project is to face challenges efficiently with limited resources. For example, how will the simulated system respond when one of its resources is busy, and how effective is this situation handled.

3. MODEL DESIGN AND DESCRIPTION:

We aim to handle the dynamic flow of patients by utilizing the doctors in treating the patients. So we simulate the entire flow of normal patients and emergency patients by consulting the available doctors with a receptionist component. Here PatientGenerator randomly generates the patients that are either normal patients or emergency patients. All the patients will be kept in a queue at the Receptionist, giving priority to the emergency patients. Depending upon the availability of doctors, the patients will be assigned to each doctor, giving their first preference to emergency patients. Here Patient-Doctor Controller components manage and coordinate the flow of patients by connecting patients with doctors. Once the patients are treated, doctors will be free, and the status will be sent to the Receptionist by Patient-Doctor Controller. And treated patients will go to the pharmacy for the medication payment and then check out from the hospital. The receptionist will have the details of medication for all the patients who got treated. Overall system design is represented as follows that includes all the hospital entities and the flow of action between them.



Overview of the System

4. DESIGN:

4.1. COMPONENTS:

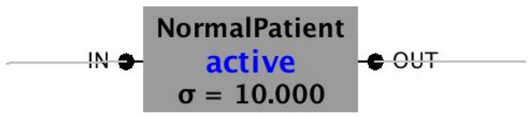
Considering the flow of actions in the system, we have different components to generate patients and monitor and track the actions with the given frequency in the Healthcare systems like PatientGenerator, Receptionist, and PatientDoctorController. Now, let us look into every component of the system in detail.

PATIENTGENERATOR

PatientGenerator is one of the key components to generate patients randomly, which are either normal patients or emergency patients. Initially, the PatientGenerator component has an active time of 10, so randomly normal patients with an active time of 10 and emergency patients with an active time of 20 are generated. For every patient-generated, the number of normal patients and the number of emergency patients will be incremented by one. The output of the patientgenerator is patients, which act as imports to the Receptionist component.

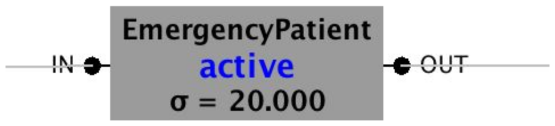
- NORMAL PATIENT

Normal Patients are the patients who visit the Healthcare system regularly on a weekly or monthly basis. So, we can expect more number of normal patients. After each iteration, the patient-generator randomly generates a normal patient, and the count will be incremented by one, and it is sent to the receptionist. The initial state of normal patients is active, and the frequency is 10. The output of normal patients is wired to the receptionist.



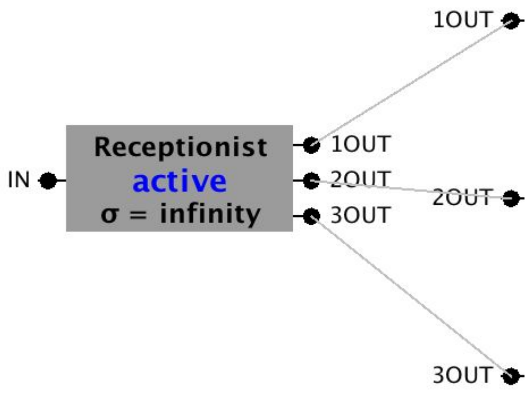
- EMERGENCY PATIENT

Emergency Patients are the patients who need immediate attention or treatment by a doctor in the Healthcare system. We can expect a small number of emergency patients, and these emergency patients should be given the highest priority. After each iteration, the patient-generator randomly generates an emergency patient, and the count will be incremented by one, and it is sent to the receptionist. The initial state of emergency patients is active, and the frequency is 20. The output of emergency patients is wired to the receptionist.



RECEPTIONIST

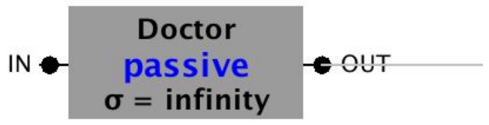
Any patient who enters the Healthcare system meets the receptionist. In our system, both the normal patients and emergency patients are the inputs to the receptionist component. At the receptionist, we have maintained a data structure to hold the patient inflow and control the flow depending upon the availability of doctors. If any emergency patient comes in, then the first preference will be given to emergency patients considering the situation of patients. So, in the waiting queue, if the arrived patient is an emergency patient, then emergency patients will be kept in the first position ignoring normal patients. The receptionist component plays an important role in tracking the patient's information and assigning them to doctors. The initial state of the receptionist is always active because of the flow of patients. The output of the receptionist component is wired to doctors.



After consulting the doctors, the receptionist component receives information about the patient's medication, and it also tracks the information about the doctor who treated a particular patient.

AVAILABLEDOCTORS

As the static parameter, we have used three doctors in our simulation model as an optimal number of doctors. The input of Doctor components is the patients that are from the receptionist. Firstly the receptionist records the incoming patients and depending on the availability of doctors, and patients will be assigned. Here, if the state of doctors is passive, it means a patient is assigned to that particular doctor. And the state of every doctor changes upon assigning the patients changes to active, adding up the frequency. And the state of doctors after treating the patient changes to passive. Each doctor is given a random amount of time to treat the patient, depending upon the patient's condition. The output of all the doctors is wired to the patient-doctor controller component.



- DOCTORONE

The input of Doctorone is the patient that will be sent from the receptionist. Once the patient meets doctorone, the state of doctorone changes from passive to active, adding up the current patient's frequency.

- DOCTORTWO

The input of Doctortwo is the patient that will be sent from the receptionist. Once the patient meets Doctortwo, the state of doctortwo changes from passive to active, adding up the current patient's frequency.

- DOCTORTHREE

The input of Doctorthree is the patient that will be sent from the receptionist. Once the patient meets Doctortwo, the state of Doctorthree changes from passive to active, adding up the current patient's frequency.

If all the doctors are busy with patients, i.e., the state of doctors is passive, then the receptionist component maintains a list of waiting patients; once any of the doctors are free, then emergency patients will be assigned to available doctors giving first preference. If there are no emergency patients, then normal patients waiting on the list will be assigned to the doctors.

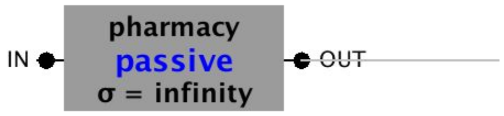
PATIENTDOCTORCONTROLLER

A PatientDoctorController component keeps track of all the doctors treating the patients. The input of PatientDoctorController is the patients after being treated by the doctors. After successful treatment, the patients will be processed towards the PatientDoctorController component. The output of the PatientDoctorController component is wired to the receptionist and the pharmacy. This component sends an acknowledgment to the Receptionist component and the Pharmacy component. Thus the Receptionist can remove the patient from the list. And the pharmacy components process the payment for the medication that is advised to that particular patient. Here, the PatientDoctorController is the key coordinator in tracking the status of the patients and the Doctors.

PatientDoctorController
passive
σ = infinity
IN
OUT

PHARMACY

One of the key components in this simulation system is the pharmacy. The Pharmacy component receives the input from the PatientDoctorController. After the treatment, the patient is sent to the pharmacy to get his/her prescription. This component will make sure that there are no long-standing queues in the pharmacy. At the pharmacy component, the payment for medication for each patient takes place. The output of the pharmacy component is wired to the receptionist component, where the medication details of each patient will be tracked and also to the whole system.



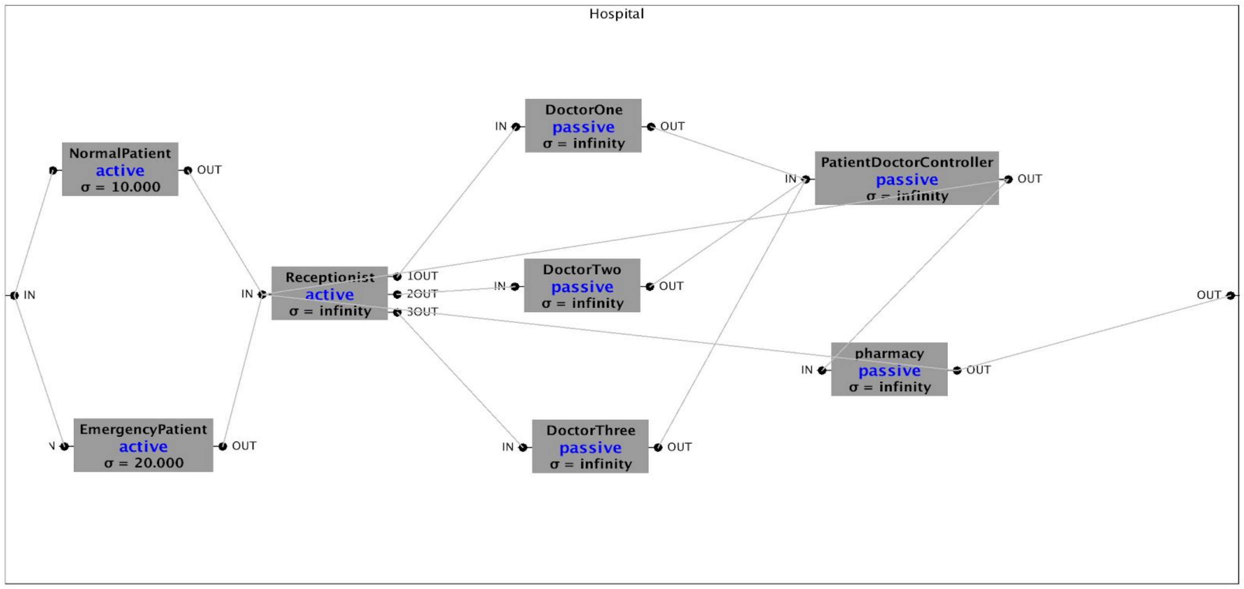
HOSPITAL SYSTEM

Altogether, all these components described above complete the Hospital System. Now let us look at the bigger picture of the system. We have two types of patients coming into the hospital. The first one being NormalPatient, someone who consults for daily checkups, annual physical, etc. The other is EmergencyPatient, who has to be treated at the earliest.

Initially, the receptionist records the incoming patients. If the patient is a normal patient, he/she is added to the queue; if the patient is an emergency patient, he/she will be added to the first of the list. We used the linked list data structure to do this for us. Whenever we have a doctor available, the patient preferably in the queue is allocated a doctor to treat.

After successfully allocating patients to the doctors, the receptionist component keeps all the incoming patients in a queue depending upon the patient's state. When the doctors are available, the receptionist forwards that patient to that particular doctor. HospitalSystem, which we designed, helps to keep track of the incoming and outgoing patients.

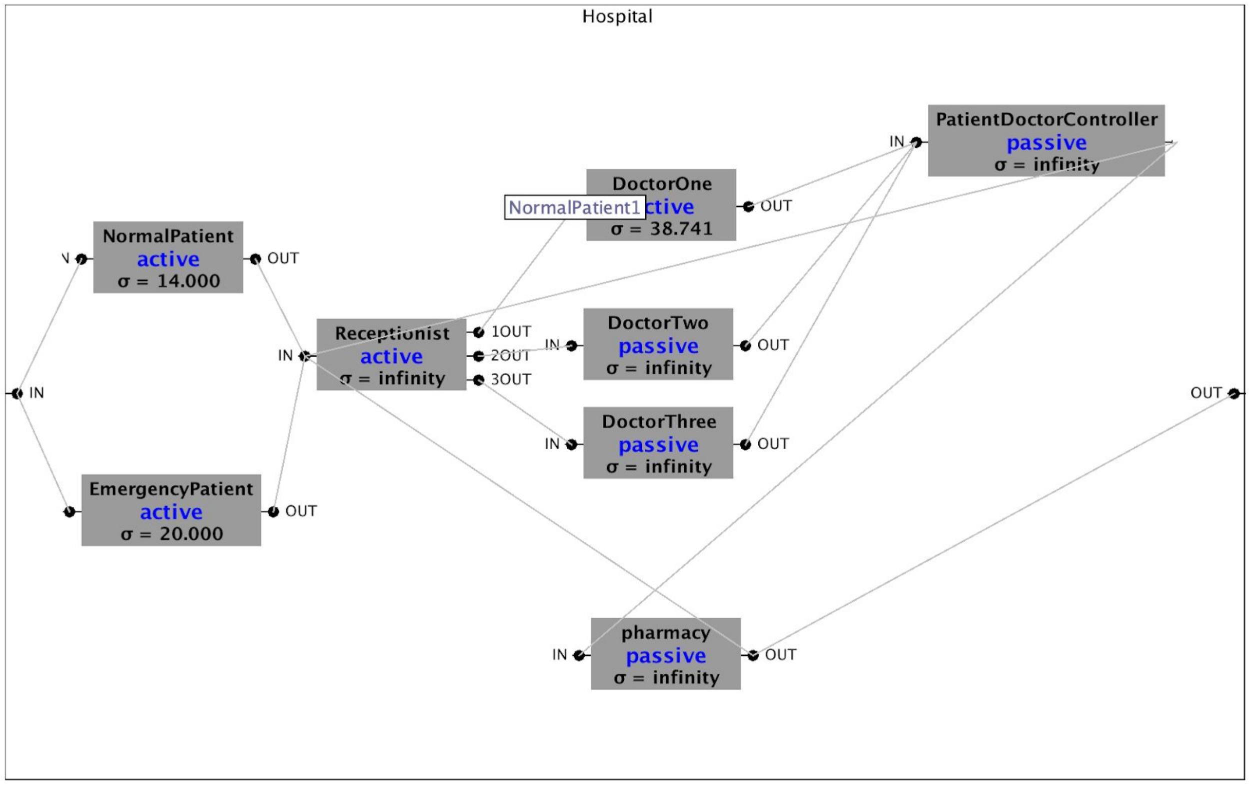
The whole system that we designed using modeling and simulation looks as shown in the below diagram.

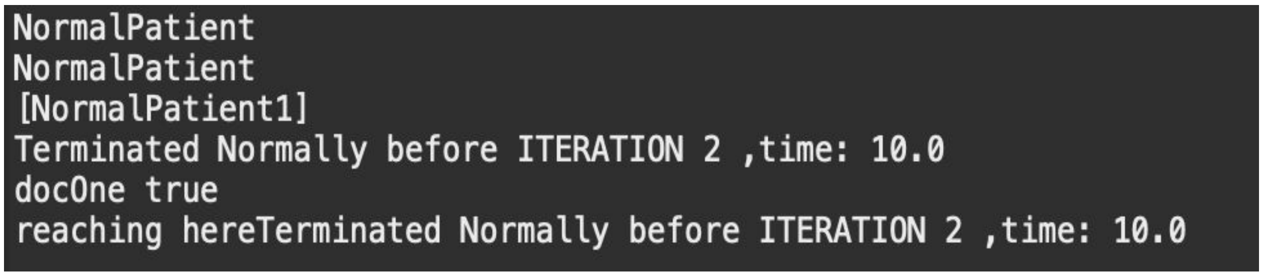


Modeling and Simulation can help us to visualize and draft an optimal solution for any given problem. And the workflow of our model after generating the patients looks as shown in the below screenshot.

Case 1:

Here a normalpatient is generated, and it is assigned to the available doctor, i.e., DoctorOne.

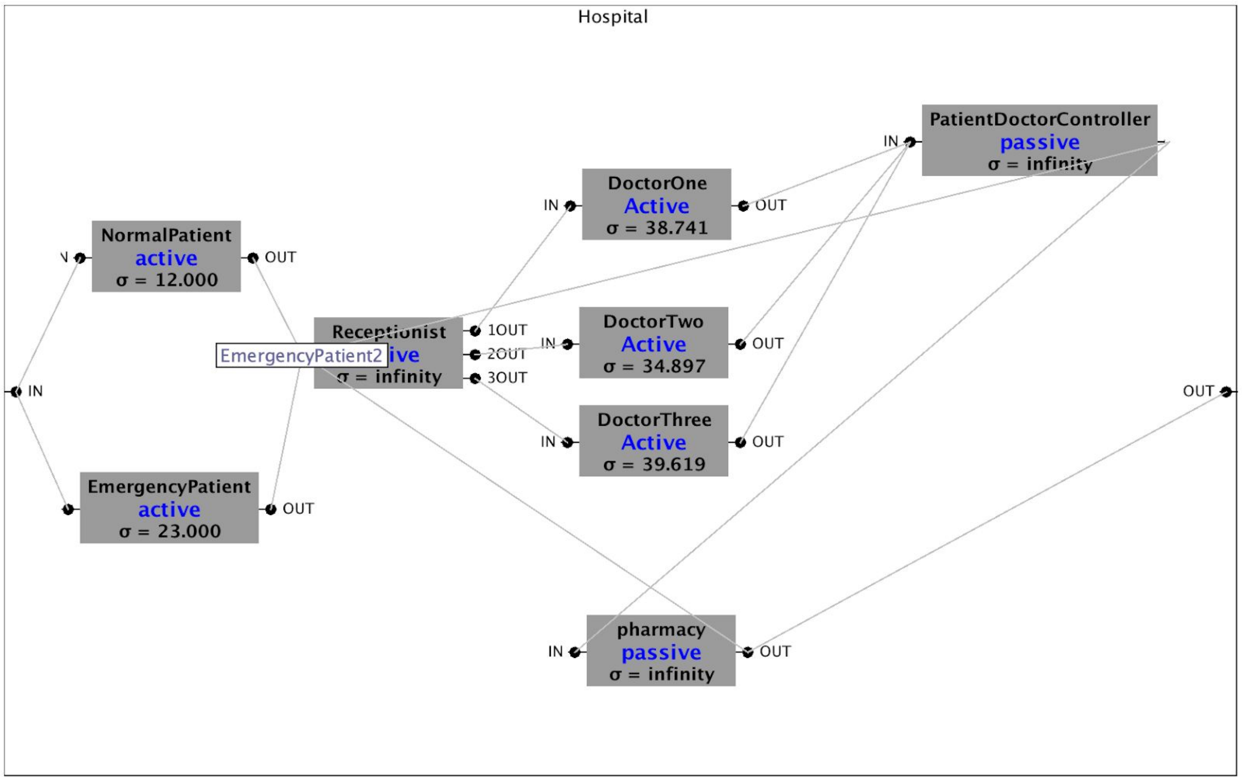


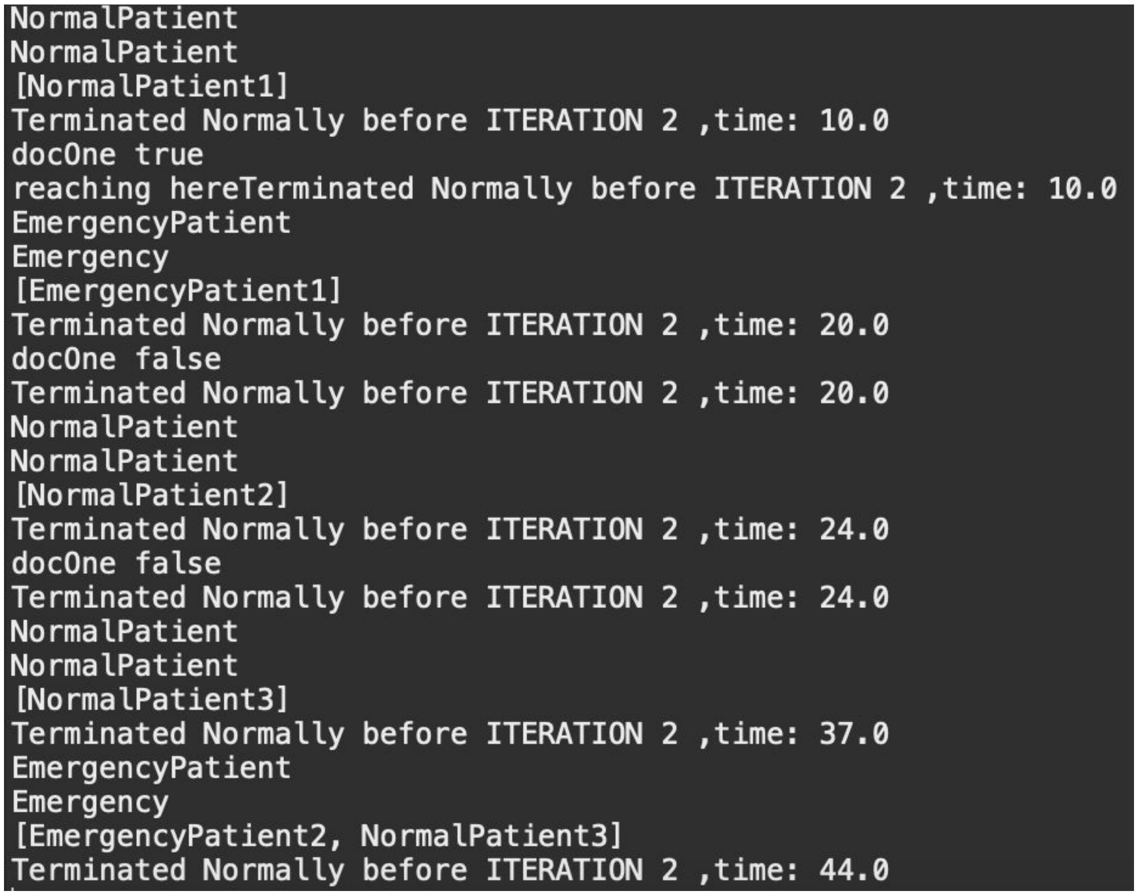


```
NormalPatient
NormalPatient
 [NormalPatient1]
Terminated Normally before ITERATION 2 ,time: 10.0
docOne true
reaching hereTerminated Normally before ITERATION 2 ,time: 10.0
```

Case 2:

All three doctors are active and no doctor is free, and the normalpatient and emergencypatient at the receptionist are waiting to consult the doctor. A list is maintained at the receptionist for the waiting patients. At the receptionist component, the emergencypatient will be kept first in the list. The below screenshots show the result.

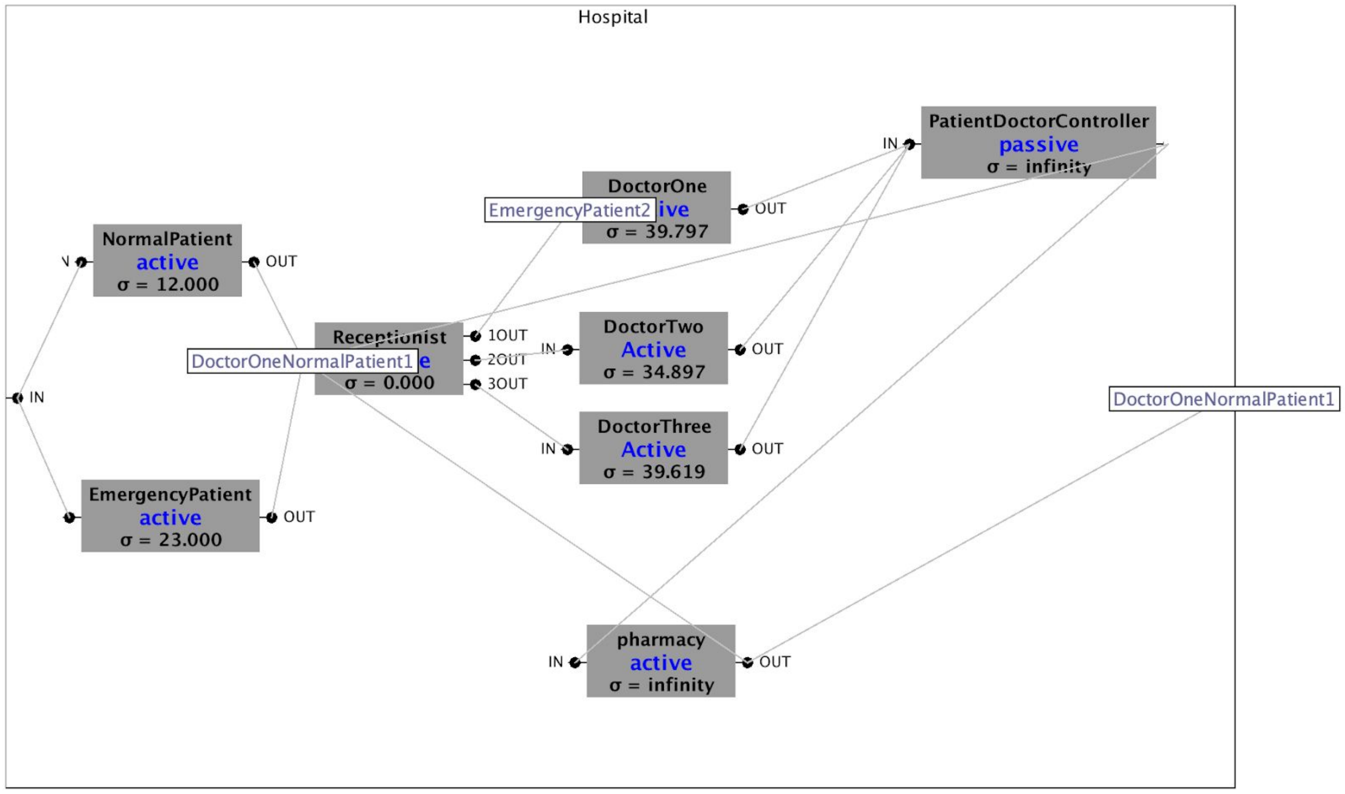


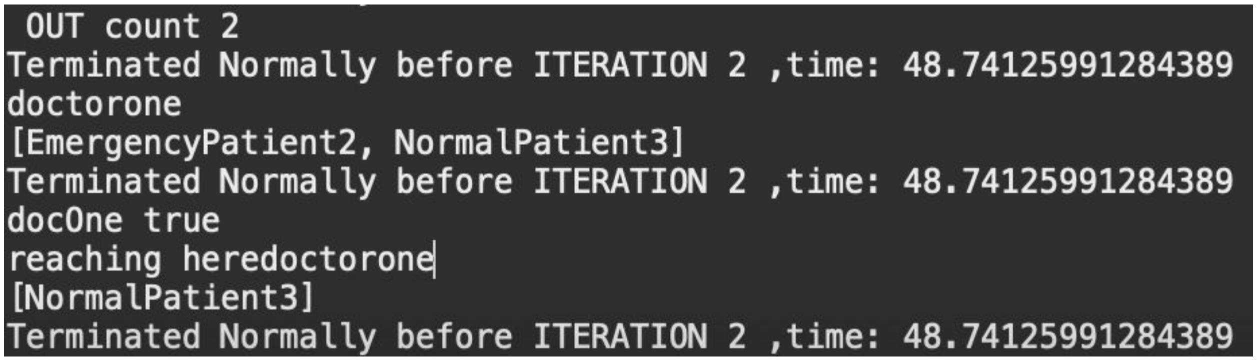


```
NormalPatient
NormalPatient
[NormalPatient1]
Terminated Normally before ITERATION 2 ,time: 10.0
docOne true
reaching hereTerminated Normally before ITERATION 2 ,time: 10.0
EmergencyPatient
Emergency
[EmergencyPatient1]
Terminated Normally before ITERATION 2 ,time: 20.0
docOne false
Terminated Normally before ITERATION 2 ,time: 20.0
NormalPatient
NormalPatient
[NormalPatient2]
Terminated Normally before ITERATION 2 ,time: 24.0
docOne false
Terminated Normally before ITERATION 2 ,time: 24.0
NormalPatient
NormalPatient
[NormalPatient3]
Terminated Normally before ITERATION 2 ,time: 37.0
EmergencyPatient
Emergency
[EmergencyPatient2, NormalPatient3]
Terminated Normally before ITERATION 2 ,time: 44.0
```

Case 3:

The final output of normalpatient after getting treated is shown in the below screenshot, and emergencypatient in the waitlist will be assigned to the available doctor, i.e., doctorone.




```
 OUT count 2
Terminated Normally before ITERATION 2 ,time: 48.74125991284389
doctorone
[EmergencyPatient2, NormalPatient3]
Terminated Normally before ITERATION 2 ,time: 48.74125991284389
docOne true
reaching heredoctorone
[NormalPatient3]
Terminated Normally before ITERATION 2 ,time: 48.74125991284389
```

5. SIMULATION RESULTS

Our simulation results focus on the average time required to treat the patient based on the condition, i.e., Normal or Emergency.

The maximum number of patients that are assigned to the doctors is 3.

Experiment 1: Let us calculate the average time required to treat a NormalPatient. We will take five normal patients and analyze the results of patient treatment on average.

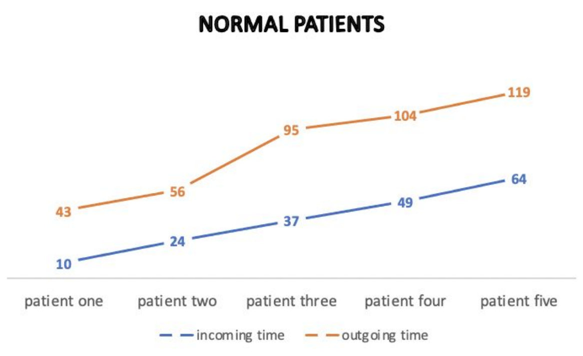
	Incoming Time	Outgoing Time
Patient One	10	43
Patient Two	24	56
Patient Three	37	95
Patient Four	49	104
Patient Five	64	119

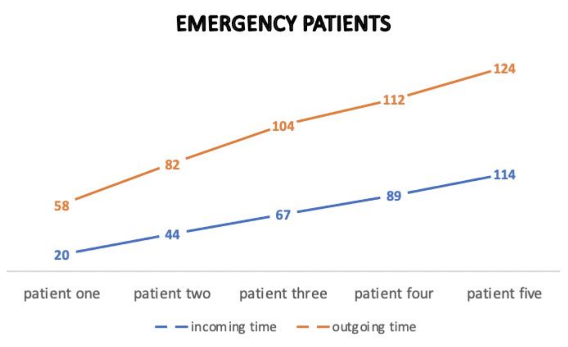
The average Time taken to treat five normal patients is 44.6 Time Units.

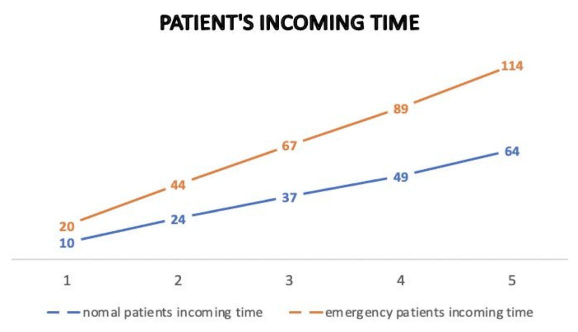
Experiment 2: Let us calculate the average time required to treat an EmergencyPatient. We will take five emergency patients and analyze the results of patient treatment on average.

	Incoming Time	Outgoing Time
Emergency Patient One	20	58
Emergency Patient Two	44	82
Emergency Patient Three	67	104
Emergency Patient Four	89	112
Emergency Patient Five	114	124

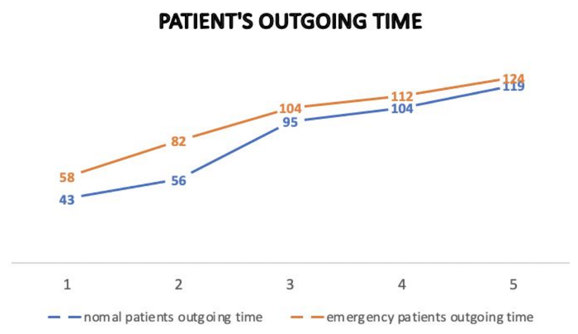
The average Time taken to treat 5 Emergency Patients is 29.5 Time Units.







PATIENT'S INCOMING TIME
114
89
67
44
20
64
49
37
24
10
1
2
3
4
5
nomal patients incoming time
emergency patients incoming time



PATIENT'S OUTGOING TIME
124
119
112
104
104
95
82
58
56
43
1
2
3
4
5
nomal patients outgoing time
emergency patients outgoing time