Hands-on Project Final Report

AirStrip Management System Using DEVS JAVA

CSC 8350: Advance Software engineering
Department of Computer Science, Georgia State university
Submitted
By

Yasaswini Kandru(002568411)
Naveen Syamala(002605464)

Problem Statement:

The airstrip Management system is a complex system that manages the flights to land or take-off. We have landing and take-off requests from domestic and international flights. It is important to manage the airstrip and allocate the airstrip to domestic and international flights. Only one flight has to use the airstrip at a time. If one flight is using the airstrip then all other requests are queued. Once the airstrip is free the first element in the queue has to use the airstrip, if there are no requests in the queue then the airstrip is in the passive state.

Modeling and Simulation Goals:

Our goal is to manage the airstrip for both domestic and international flights through a systemic flow. We will generate the requests from domestic and international generators. We will couple the domestic landing and domestic take-off generators to the domestic airstrip generator. Coming to international flights we will couple the international landing and the domestic take-off generators to the international airstrip generator, both the domestic airstrip and international airstrip generators will queue the requests and process the request and send them to the airstrip system. In the end, we will calculate the clock values.

Model and Design:

The requests from the domestic land generator and take-off generator are coupled to the domestic airstrip controller and the requests from the international land generator and take-off generator are coupled to the international airstrip controller, both the domestic airstrip controller and the international airstrip controller are coupled to the airstrip controller. Here the priority is given to the international flights, if the domestic flight requests to land or take-off and at the same time international flight requests to land or take-off then international flight gets the first priority to use the airstrip, at this scenario the domestic flight goes into a queue.

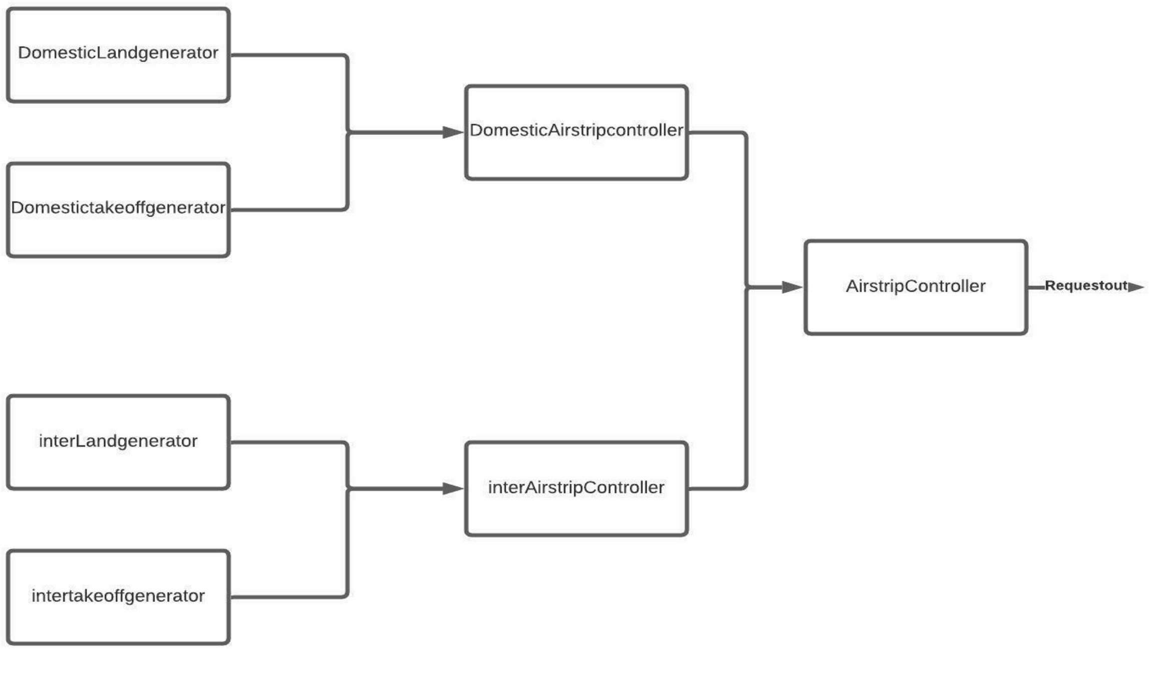


Fig 1. Overview of the airstrip management system

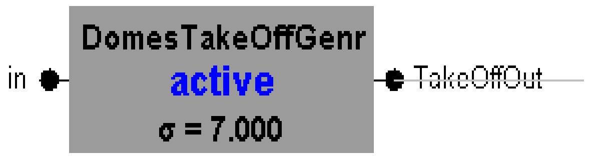
Our assumptions are

- Only one flight has to use the airstrip at a time.
- If international flight requests to land or take-off then all other requests are queued.
- Once the airstrip is free then the first element in the queue gets to use the airstrip.
- If there are no requests in the queue then the airstrip is in the passive state.

Design Components:

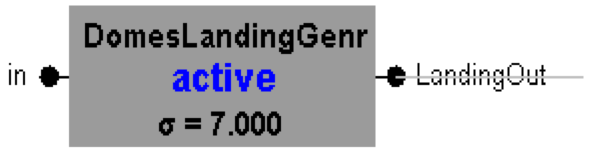
DomesticTake-offGenerator

The DomesticTake-off generator has one input port, in and one output port, TakeOffOut. Initially it is in passive state. When take-off is requested then the DomesticTake-of generator will generate the request then the state will change from passive to active state. A new request is randomly generated by the DomesticTake-off generator and it maintains an active state for that frequency.



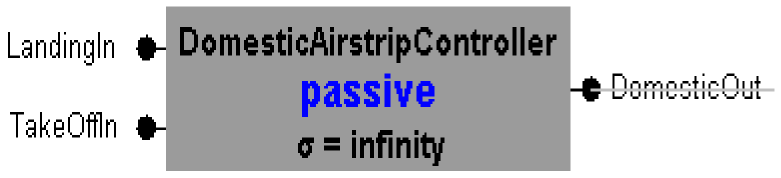
DomesticLandingGenerator

The DomesticLanding generator has one input port, in and one output port, LandingOut.The DomesticLandingGenerator initially is in passive state.When a request is generated then the state changes from passive state to active state. A new request is randomly generated by the DomesticLandingGenerator and will be in active state for that frequency.



DomesticAirstripController

The DomesticAirstripController will manage the queues from the Domestic landing generator and the domestic take-off generator, DomesticAirstripController will send the first element in the queue to use the airstrip.



One of the input port, LandingIn is coupled with LandingOut port of Domestic Landing Generator. Other input port is coupled with TakeOffOut port of Domestic TakeOff Generator.Below is the code snippet for the corresponding code.

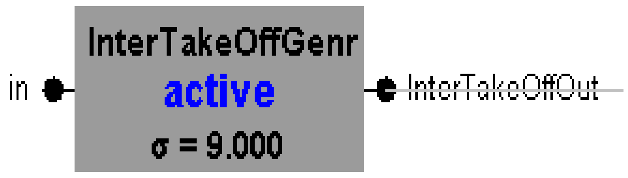
```
//domestic
 ViewableAtomic d_landing = new LandingGenr("DomesticLandingGenr",6);
 ViewableAtomic d_takeoff = new TakeOffGenr("DomesticTakeoffGenr",2);
 ViewableAtomic domestic = new DomesticAirstripController("DomesticAirstripController");

 add(d_landing);
 add(d_takeoff);
 add(domestic);

 addCoupling(d_landing,"LandingOut",domestic,"LandingIn");
 addCoupling(d_takeoff,"TakeOffOut",domestic,"TakeOffIn");
```

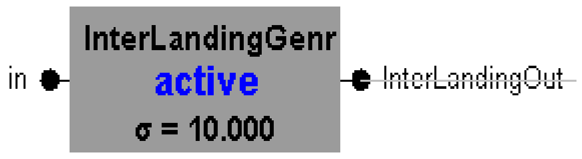
InternationalTakeOffGenerator

The InternationalTakeOffGenerator initially it is in passive state. It has one input port, in and one output port, InterTakeOffOut. When take-off is requested then the InternationalTake-offGenerator will generate the request then the state will changes from passive to active state. A new request is randomly generated by the InternationalTake-offGenerator and it maintains an active state for that frequency.



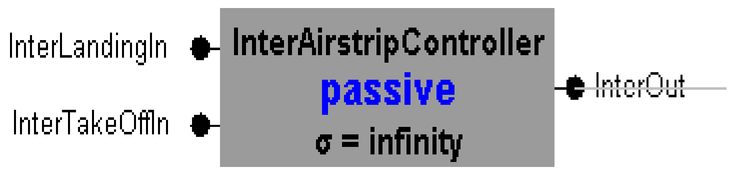
InternationalLandingGenerator

The InternationalLandingGenerator initially is in a passive state, when a request is generated when the state changes from passive state to active state. A new request is randomly generated by the InternationalLandingGenerator and will be inactive state for that frequency. The InternationalLandingGenerator has one input port, in and one output port, InterLandingOut.



InternationalAirstripController

The InternationalAirstripController will manage the queues from the international landing generator and the international take-off generator, InternationalAirstripController will send the first element in the queue to use the airstrip.



One of the input port, InterLandingIn is coupled with InterLandingOut port of International Landing Generator. Other input port is coupled with InterTakeOffOut port of International TakeOff Generator. Below is the code snippet for the corresponding code.

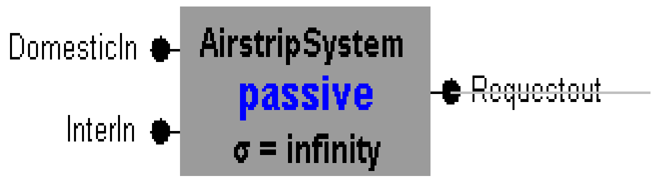
```
//International
 ViewableAtomic i_landing = new InterLandingGenr("InterLandingGenr",4);
 ViewableAtomic i_takeoff = new InterTakeOffGenr("InterTakeoffGenr",3);
 ViewableAtomic inter = new InterAirstripController("InterAirstripController");

 add(i_landing);
 add(i_takeoff);
 add(inter);

 addCoupling(i_landing,"InterLandingOut",inter,"InterLandingIn");
 addCoupling(i_takeoff,"InterTakeOffOut",inter,"InterTakeOffIn");
```

AirstripSystem

Once the airstrip is free the first element in the queue gets to use the airstrip. International flights have the highest priority to use the airstrip, if the airstrip gets the request from domestic and international flights the airstrip gives highest priority to the international flights.



One of the input ports, DomesticIn of this model is coupled with DomesticOut of Domestic Controller. And the other port, InterIn is coupled with InterOut of International Controller. Also, the output port of Airstrip is connected to the final output. Corresponding code can be seen below.

```
    //Airstrip
    ViewableAtomic airstrip = new AirstripSystem("AirstripSystem");

    add(airstrip);

    addCoupling(domestic,"DomesticOut",airstrip,"DomesticIn");
    addCoupling(inter,"InterOut",airstrip,"InterIn");
    addCoupling(airstrip,"Requestout",this,"out");
```

Simulation model:

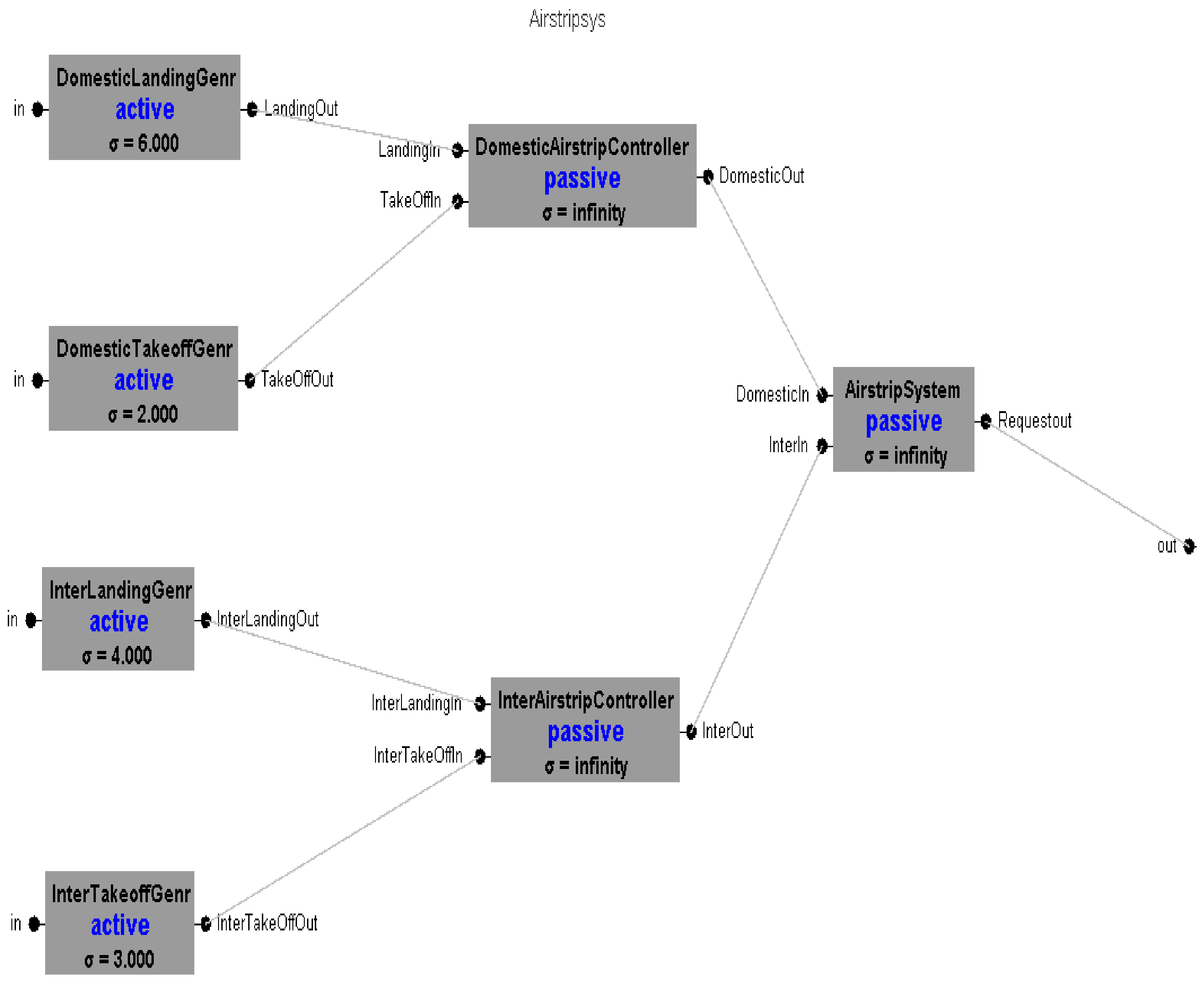


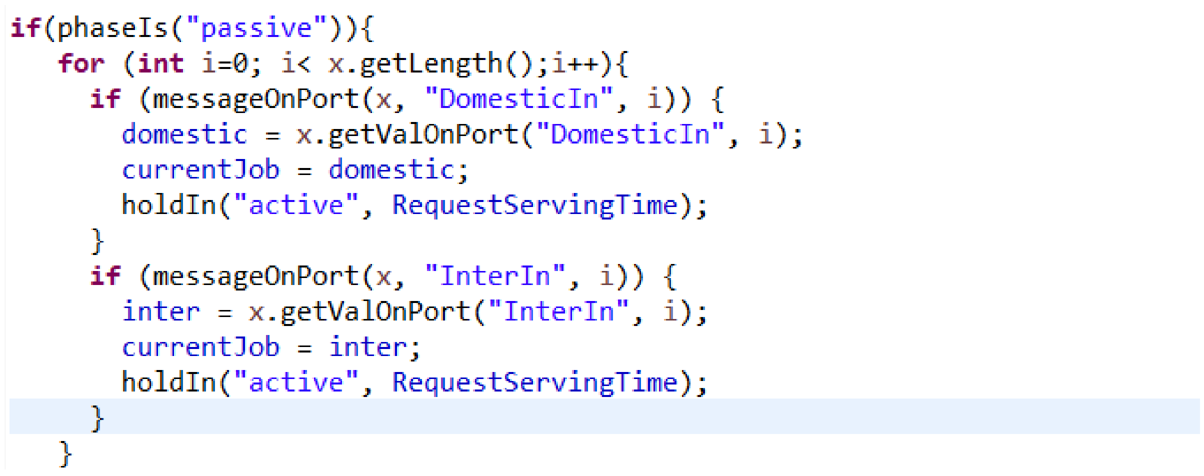
Fig 2. Simulation model of Airstrip Management

Results:

The Airstrip management system simulation model, only one flight can use the airstrip at a time. When one flight gets to use the airstrip then all the flights are queued once the airstrip is free the first element in the queue gets to use the airstrip.If there are no requests in the queue then the airstrip will be in a passive state.

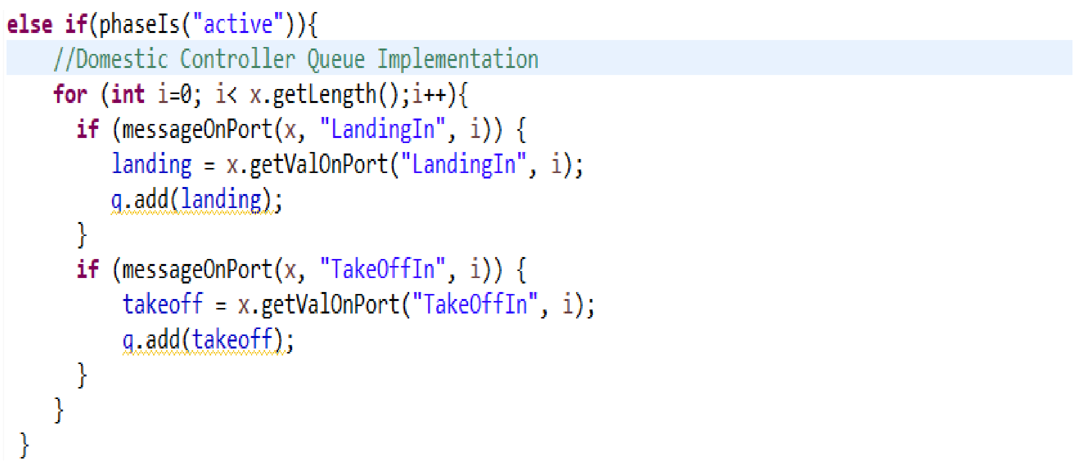
We have employed queue data structure to process the multiple requests generated. Queue employes first in first out scenario and hence the requests that reaches the queue will be processed first, followed by the second element and so on. We have deployed three different queues, one to handle the requests from domestic flights, one for international flight and one for airstrip allocation.

The below is the code snippet how airstrip model functions when in active state.



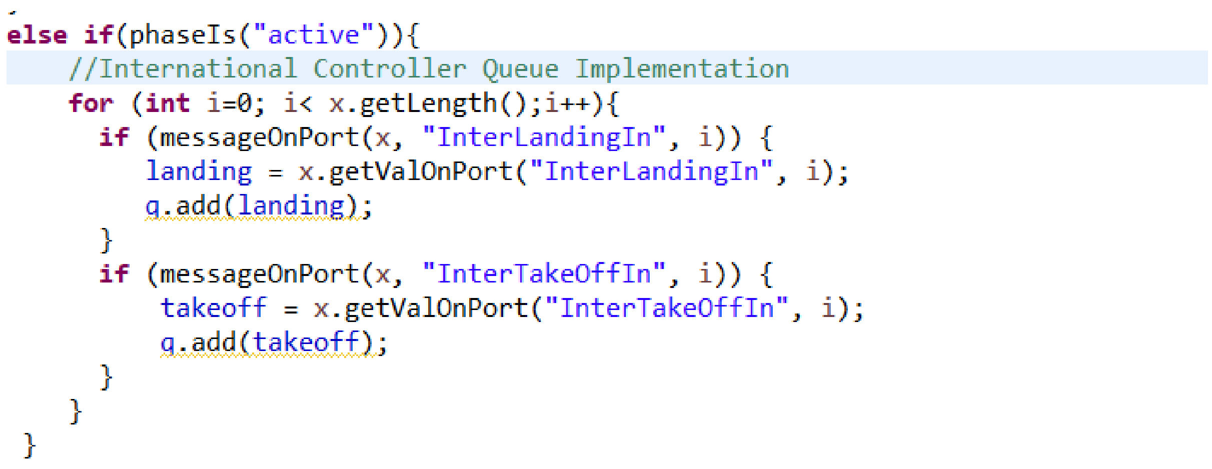
```
if(phaseIs("passive")){
   for (int i=0; i< x.getLength();i++){
     if (messageOnPort(x, "DomesticIn", i)) {
       domestic = x.getValOnPort("DomesticIn", i);
       currentJob = domestic;
       holdIn("active", RequestServingTime);
     }
     if (messageOnPort(x, "InterIn", i)) {
       inter = x.getValOnPort("InterIn", i);
       currentJob = inter;
       holdIn("active", RequestServingTime);
     }
   }
```

The below is the code snippet for domestic flights controller queue.



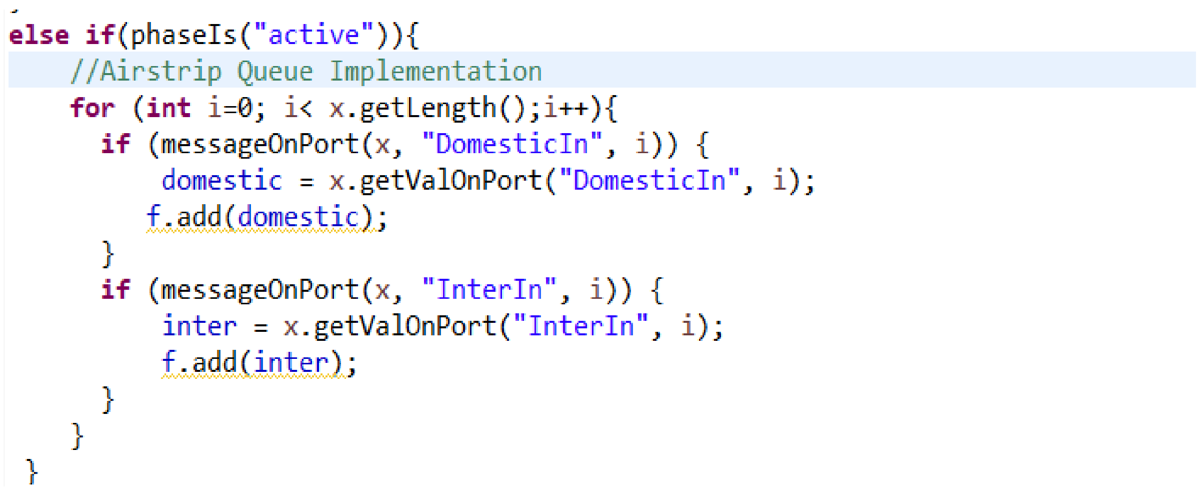
```
else if(phaseIs("active")){
   //Domestic Controller Queue Implementation
   for (int i=0; i< x.getLength();i++){
     if (messageOnPort(x, "LandingIn", i)) {
        landing = x.getValOnPort("LandingIn", i);
        q.add(landing);
     }
     if (messageOnPort(x, "TakeOffIn", i)) {
         takeoff = x.getValOnPort("TakeOffIn", i);
         q.add(takeoff);
     }
   }
 }
```

The below is the code snippet for International flights controller queue.

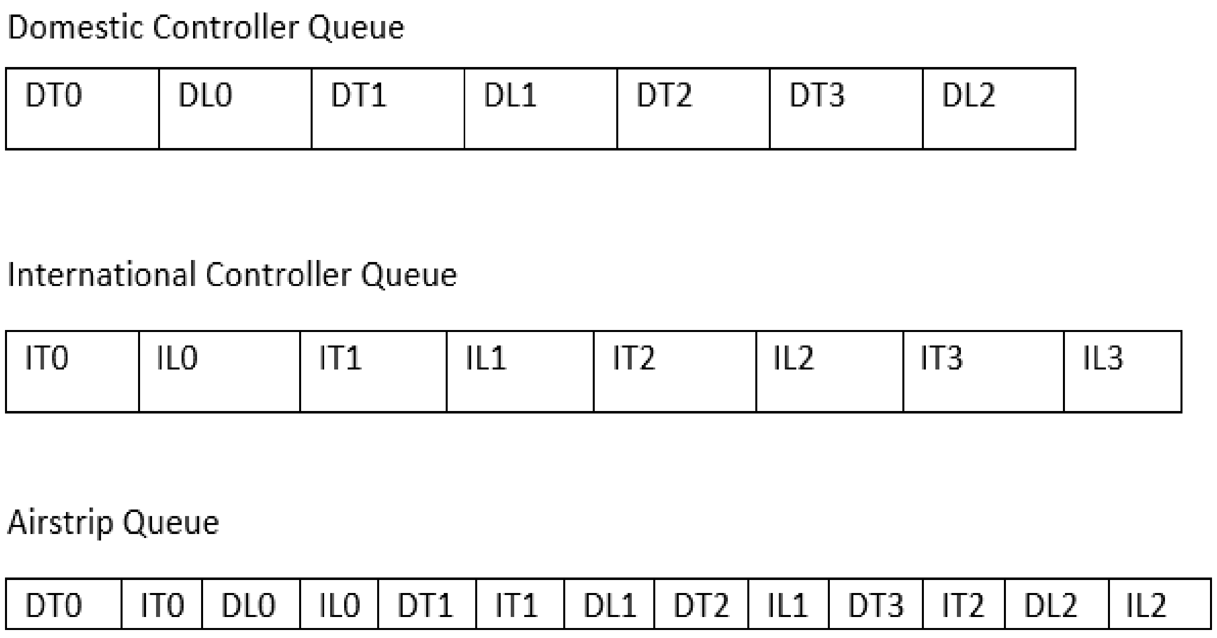


```
else if(phaseIs("active")){
    //International Controller Queue Implementation
    for (int i=0; i< x.getLength();i++){
      if (messageOnPort(x, "InterLandingIn", i)) {
         landing = x.getValOnPort("InterLandingIn", i);
         q.add(landing);
      }
      if (messageOnPort(x, "InterTakeOffIn", i)) {
          takeoff = x.getValOnPort("InterTakeOffIn", i);
          q.add(takeoff);
      }
    }
 }
```

The below is the code snippet for Airstrip queue.



```
else if(phaseIs("active")){
    //Airstrip Queue Implementation
    for (int i=0; i< x.getLength();i++){
      if (messageOnPort(x, "DomesticIn", i)) {
          domestic = x.getValOnPort("DomesticIn", i);
         f.add(domestic);
      }
      if (messageOnPort(x, "InterIn", i)) {
          inter = x.getValOnPort("InterIn", i);
          f.add(inter);
      }
    }
 }
```



Domestic Controller Queue

DT0	DL0	DT1	DL1	DT2	DT3	DL2

International Controller Queue

IT0	IL0	IT1	IL1	IT2	IL2	IT3	IL3

Airstrip Queue

DT0	IT0	DL0	IL0	DT1	IT1	DL1	DT2	IL1	DT3	IT2	DL2	IL2

Fig 2. Requests handled by different queues.

The above image represents the order of requests in Domestic flight requests queue and international flight requests queue. As we can observe, the DomesTakeOff0 is the first request in the queue, followed by DomesLanding0, DomesTakeOff1, DomesLanding1etc. Similarly, International flight requests queue is occupied by InterTakeOff0, InterLanding0, InterTakeOff1, InterLanding1 etc. In the airstrip controller queue, we can see the order of domestic, international flights take off and landing requests.

Analysis:

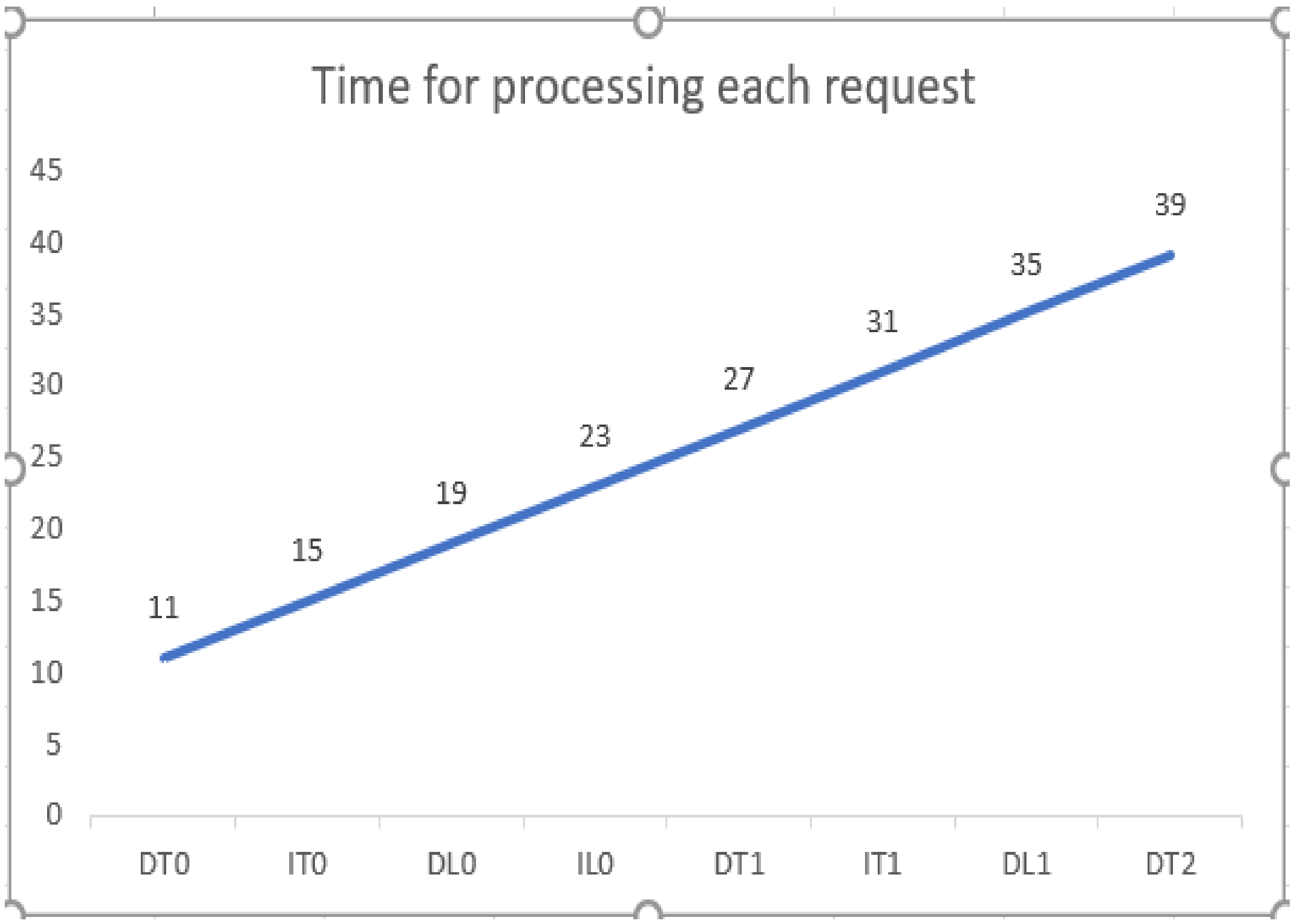


Fig 3. Time taken to process each request

The above graph depicts the processing time for landing, takeoff requests generated by both domestic and international flights. All the different types of request have different processing times and they are processed according to the queue status. Priority is given to the first elements in the queue. It takes around 11 units of time to complete the first request i.e DomesticTakeOff0. When the model is busy processing DomesticTakeOff0, the requests are queued and given access when the airstrip is passive. Since the rest of elements are already queued it, time taken for completing the following requests will be the sum of time taken by first request plus the count of queue multiplied by four. Count is queue is the number of waiting requests when the request has reached airstrip queue.

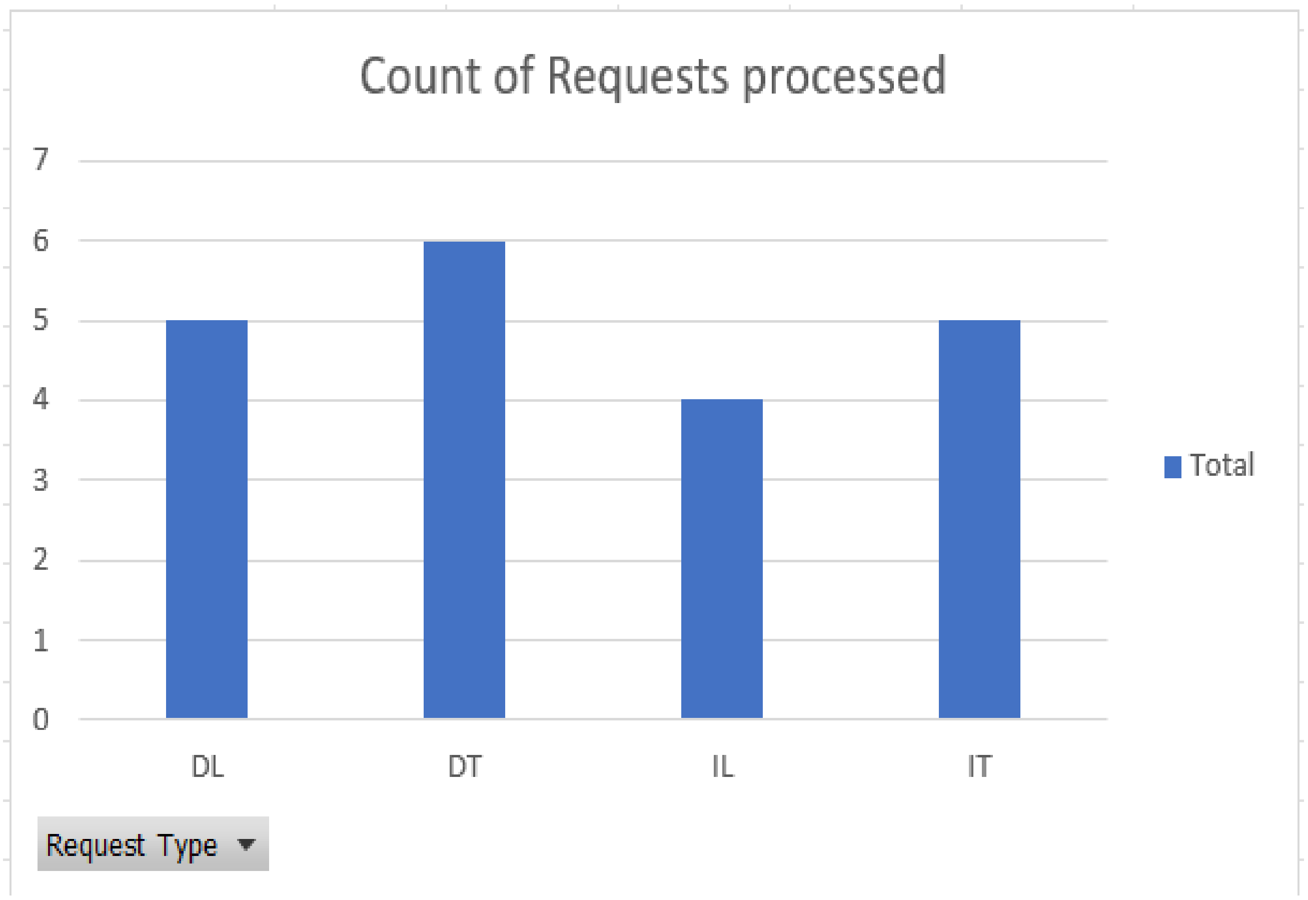


Fig 4. The number of requests processed for each type

The above graph shows the total number of requests processed for each type by the time clock reaches 103 units. The simulation model has processed five landing requests, six takeoff requests of domestic flights and four landing requests, five takeoff requests of international flights.

Conclusion:

This Airstrip management system will allocate airstrip for domestic and international flights, the airstrip will give the highest priority to the international flights when both domestic and international flights come to use the airstrip, we have calculated the time for each process of each request and we plotted a graph.

In the future scope, we are going to add charted and military flights to the simulation model and give the priority to both international and military flights.