

ATC System Simulation

Ahmed A. H. Alkurdi

College of Computer Science and Information Technology, Nawroz University, Kurdistan Region of Iraq.

ABSTRACT

Nowadays, Simulation has seen a major role in system development. Real world programs and events are simulated using computer software for the purpose of reducing risk, failure, testing and education. Simulation is essential to real world programs like Air Traffic Control system. Simulating such a system is vital as its failure can be catastrophic and disastrous. In this article a study of the simulation of Air traffic control system using Gridsim toolkit is presented. Air traffic control is a system that manages airplane movement on ground and in the sky. It is responsible for all the components of an airport. Gridsim toolkit is a simulation tool used for resource modelling and application scheduling in parallel and distributed computing systems. In this paper, Gridsim toolkit is used to map the Air traffic control components in real time to simulate the system work where different example of producing the system are presented.

KEYWORDS : Gridsim toolkit, Simulation, Resource Modelling, Application Scheduling, Air Traffic Control.

1. INTRODUCTION

Air traffic control (ATC) is a service that facilitates the safe and orderly movement of aircraft on the ground and in the air by receiving and processing data from radar and devices that monitor weather conditions and by maintaining radio contact with pilots. Pilots are obliged by flight rules. Flight rules are designed to maintain order, if everyone who piloted an airplane flew without regard for others that would result in chaos. The rules, actually Federal Laws, are called the Federal Aviation Regulations (FAR). The Federal Aviation Regulations are basis for air traffic control operating rules; specialists at air traffic control work in teams at various Federal Aviation Administration facilities carefully monitor air traffic in airspace during both day and night. Air Traffic Controllers provide many services to pilots for example, traffic advisories and routing to avoid bad weather conditions and ensure efficiency. Due to the large amount of flights, controllers must carefully integrate flight plans that might be during the same time at the same airport using identical

routes (Conrad 2010). In this paper, Procedures that occur during flights from take-off to landing will be discussed going through departure, en-route and descend, All the transferred information and data being generated from pre-flight to landing and all other processes that occur during flights will be discussed, focusing on the role that ATC systems have during these processes.

2. Air Traffic Control System

The air traffic control system, which is run by the Federal Aviation Administration (FAA), has been designed around these airspace divisions. The air traffic control system divisions are : (Freudenrich, 2001, 2001).

- **Air Traffic Control System Command Centre (ATCSCC).**
- **Air route Traffic Control Centres (ARTCC).**
- **Terminal Radar Approach Control (TRACON).**
- **Air Traffic Control Tower (ATCT).**
- **Flight Service Station (FSS).**

2.1 Air Traffic Control System Command Centre (ATCSCC).

Air Traffic Control System Command Centre (ATCSCC) manages and regulates the air traffic in case of traffic congestions, bad weather, equipment and runway shutdown or any other conditions that can impact or place stress. The Traffic Management Specialists at the ATCSCC can modify traffic demands by taking any actions in order to remain within the system. The ATCSCC is cooperating with the airline personnel, Traffic Management Specialists (TMS) and Air Traffic

Academic Journal of Nawroz University
(AJNU) Volume 7, No 3 (2018).

Received 10 April 2018;

Regular research paper : Published 20 July 2018

Corresponding author's e-mail : ahmad.ala89@gmail.com

Copyright ©2018 Ahmed A. H. Alkurdi.

This is an open access article distributed under the
Creative Commons Attribution License.

Controllers (ATC) at affected facilities and designed to satisfy the requirements and demands of the public travelers. Due to the increase in air traffic service demands, capacity and safety requirements must be balanced by The National Air Traffic Management System. Thus National Air Traffic Management System must be automated. Air traffic control and traffic management performance will be enhanced by automation to confront the ever increasing traffic loads. National Airspace System uses Air traffic management to reduce delay time and congestion while increasing throughput (Conrad 2010).

2.2 Air Route Traffic Control Centres (ARTCC).

Air route traffic control centre (ARTCC) monitors airplanes that are en-route from TRACON. ARTCCs, commonly known as "Centres", provide Air Traffic Service to airplanes in controlled airspace operating on Instrument Flight Rules (IFR) flight plans, and mainly while the flight is en-route. Air traffic controllers monitor and control all kinds of aircrafts like private single engine, army jets, commercial airlines and commuter airlines (Conrad 2010).

The air route control centre contains controller to aid in fulfilling its functions :

- Radar Associated Controller.
- Radar Controller.
- Radar Hand-off Controller

Flight progress strips are used to point the location of aircrafts while en-route, flight progress stripes are pieces of paper that include information from the flight plan such as information about the pilot, flight type, destination, aircraft type etc. these stripes must be printed 20 minutes before an aircraft enters a centre's space, these strips are then sent to the Radar associated controller which insures that all information necessary to communicate with the aircraft are included, afterwards the Radar controller ensures that communication takes place between ground controllers and the aircraft. The radar controller separates aircrafts and directs them into safe flight distance. Aircrafts should be 5 miles separated laterally or longitudinally by standard. The radar controller and the associated controller both provide services to ensure the coordination between sectors. The radar hand-off controller is used as an extra eye when air traffic becomes heavy. It aids the radar team in maintaining safe distance between aircrafts and coordinate between other controllers when needed (Conrad 2010). The air route traffic control centre maintains communication with all aircrafts until aircrafts are about 240km away from their final destination, at this time it will instruct the pilot to descend where handling is turned to the TRACON controller (Conrad 2010).

2.3 Terminal Radar Approach Control (TRACON).

TRACONs are facilities that contain radar operations from which controllers can control and direct airplanes during departure, descent and approach phases of flight. One TRACON can manage handling the air traffic for several airports in its surrounding area. The TRACON wide airspace depends on its location. The TRACON controller manages the process of directing aircraft that are transitioning from the en route phases through the approach phase into a destination airport located within the TRACON's airspace. More than one controller position is available in the TRACON (Conrad 2010). These positions are listed depending on its designated :

- High altitude descent controller
- Low altitude descent controller
- Approach controller
- Feeder controller

As the airspace of a TRACON varies in size depending on its location and the size and number of airports it serves, so more than a TRACON controller is needed through the two phases of descent. In general, it has about a 40-50 mile radius. Using radar and two-way radio, TRACON controllers give pilots instructions regarding heading, speed, and rate of ascent or descent within the TRACON area. Usually, a series of TRACON controllers direct an approaching aircraft through two stages of descent. The high altitude controller is to hand off the aircraft to the low altitude controller, who then passes the aircraft to the approach controller. Then the approach controller merges the many descending aircraft flying toward the same destination airport into one line of air traffic with consideration to the safe separation space required. When the aircraft is within the airport's air space an electronic transfer will occur, typically about five miles away. The transfer starts from approach control that hand off the control to the feeder controller. When reaching the airport space the control will be transferred to a local controller in the airport ATCT. At that moment the aircraft will be controlled by one of the controllers in TRACON (Conrad 2010). The TRACON also controls departing aircraft that are handed off from the local controller in the ATCT and are transitioning from the departure phase of flight beyond about five miles of the airport to the en route phase, when the flight is handed off to the ARTCC at about 40-50 miles from the airport (Wu and Zhou, 2012).

2.4 Air Traffic Control Tower (ATCT).

This is the famous tower that is there in every airport, it has an all-round window top that makes every part of the airport visible, this control helps coordinate landing, take-off, aircraft ground movement and in-range air-traffic. The air traffic control tower has four main classifications (Conrad 2010) :

- Flight Data Controller.

- Clearance Delivery Controller.
- Ground Controller.
- Local Controller.

a. Flight Data Controller controls and coordinates close-range air traffic through electronic flight data display “user request evaluation tool (URET)”, it also retrieves and passes on IFR departure clearance, functions the flight data processing equipment, relays weather information and controls automatic terminal information service (ATIS) which in turn monitors weather and makes a record every hour or less at bad weather conditions (Conrad 2010).

b. Clearance Delivery Controller obtains and relays departure clearances, departure clearances contain many information such as airplane identification, clearance limit, departure process, flight route, altitude, etc. Clearance delivery controller also insures that the route for the flight indicated is the same as the route approved for the flight request (Conrad 2010).

c. Ground Controller manages and coordinates ground movement of airplanes taxiing or vehicles on taxiways and inactive runways. It issues clearances to aircrafts to use runways after it checks with the Local controller, it prevents unauthorized use of active runways, runway incursions, which imposes a big threat to aircraft safety. The ground controller is held responsible for the protection of critical areas as well. Critical areas become larger with bad weather conditions and include localizer, glide slope and precision approach (Conrad 2010).

d. Local Controller coordinates and controls the separation and sequencing of aircrafts on departure and arrival and issues clearances for aircrafts landing or departing, it also relays IFR clearances and taxiing procedure. One other responsibility of the flight service station is providing assistance to flights flying thorough their section (Conrad 2010).

2.5 Flight Service Station (FSS).

The flight service station is responsible for offering valued information and service to private aircraft pilots, it cooperates with the automated flight service station to provide these services and both are FAA air traffic facilities. These facilities also offer pilots en-route radio communication and Visual Flight Rules (VFR) search and rescue services. The flight service station also come to aid aircrafts in emergencies or when their lost as sometimes pilots lose track of their route or become disoriented or sometimes aircrafts need to perform an emergency landing which wasn't scheduled, so the FSS provides the pilots with the required assistance thought the emergency services at that zone (Conrad 2010). ATC clearances are also relayed by the flight service station, Notices to Airmen (NOTAM) recordings are provided by FSS, it also provides National Airspace System (NAS) information and weather broadcasts aviation

(Conrad 2010). Flight Advisory Services are offered by the FSS along with weather data information and airport advisory generation; it also provides Virtual Flight Rules, Instrument Flight Rules (IFR) and monitors Navigational Aid System (NAVAIDS) (Conrad 2010). Nowadays technologies have improved and renovated flight service stations by making Automated Flight Service Stations available, these new stations are fitted with modern technology to offer more efficient assistance to specialists in providing Air Traffic Services (Conrad 2010).

3. Flight Profile

Every flying aircraft follows a flight pattern that begins at pre-flight and ends landing. This pattern is called a flight profile. Typical a flight profile involves seven phases. ATC facilities, along with their controllers, monitor these flight phases. An ATC controller follows precise instructions and measures to direct a flight through designated routes. Safety and efficiency of the flight are ensured though ATC controllers which use special equipment and decision support tools. During these flight phases communication is maintained between the aircraft, pilot and air traffic controller. Their interactions are described as follows (Wu and Zhou, 2012).

a. Preflight : In this phase latest weather information are passed to the pilot, then the pilot files a flight plan with air traffic control which might include airline name, flight number, airplane type and information about the equipment unit, the flight plan also contains information about the route specified to the destination, speed and altitude of the flight. At this point the tower performs some routine procedures, first the Air Traffic Controller checks the weather and the flight plan, then its faced with three options, accept the flight plan, reject it or produce a different flight plan. If the ATC approves the flight plan it generates a communication code and a tracking number from the communication and tracking files, afterwards these information are all printed into “flight slips” along with the weather report, ground traffic and air traffic, these slips are forwarded to the controller to issue clearance (Freudenrich, 2001). before takeoff, flight routine checks are performed by the pilot, afterwards the pilot drives the aircraft away from the terminal's gate, and clearance is issues by the controller to take the plane onto the specified runway (Wu and Zhou, 2012).

b. Takeoff : This phase mainly is about the permission being generated by the local control and passed to the pilot in which he will power up the airplane for take-off roll (Wu and Zhou, 2012). The sky, airfield and tunway are monitored by the local controller in the tower though surface radar, the radar is responsible for maintaining safe distance between airplanes. Then final clearance is issued for take-off by the controller and a

new radio frequency is provided for the departure controller. After the pilot is cleared to fly, he builds up speed down the runway and takes the plane into air. When the plane leaves up, it's handed electronically to the departure controller by the local controller (Freudenrich, 2001).

c. Departure : after take-off, the pilot must change radio frequencies to communicate with the Departure control as instructed, the Departure Control in the TRACON send new flight instruction to the pilot. The instructions will show the pilot a pre-determined, ideal routing in which the aircraft shall follow to its route. More altitude and routing clearance are issued for the pilot. The controller monitors the airplane and its path on the radarscope. The departure controller transfers the aircrafts onwards to the next airspace controller as it reaches the edge of the TRACON airspace (Wu and Zhou, 2012).

d. En-route : In this phase the pilot is advised about the altitude and direction of the flight, and the radio frequency in which he should tune into by the ARTCC which controls this phase from the TRACON. This phase may take hours or minutes depending on the distance between origin and destination (Wu and Zhou, 2012).

e. Decent : When the aircrafts approaches its destination, the pilot is advised to tune into a different radio frequency and contact Approach Control for instructions. The pilot must descend and change flight

heading. Afterwards maneuvers begin towards the destination airport (Wu and Zhou, 2012).

f. Approach : The Approach Controller at the TRACON sends the pilot an approach clearance to the destination airport. Then the pilot is instructed to line in the aircraft with other flights landing in the same airport. A specific procedure is provided to the pilot so that he could get in line for landing. Afterwards the pilot is advised to tune into a different frequency to communicate with the Local Controller to request landing clearance. The Tower receives the airplane electronically from the TRACON (Wu and Zhou, 2012).

g. Landing : After the Local Controller takes over, it monitors the runway and the sky to insure safety and generate an all clear to land signal to the pilot. After landing safely on the ground, the plane is handed over to the Ground Control. The Ground Controller provides directions to the pilot through taxiways to the final destination gate at the terminal (Wu and Zhou, 2012).

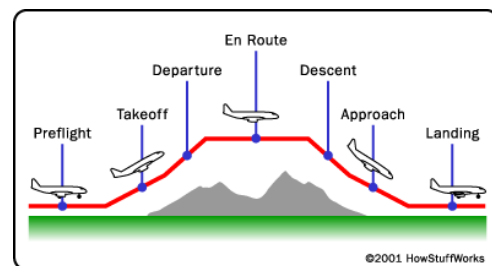


Figure (1) : Flight Phases

4. Data Flow Diagram

Below is a data flow diagram of the proposed ATC system.

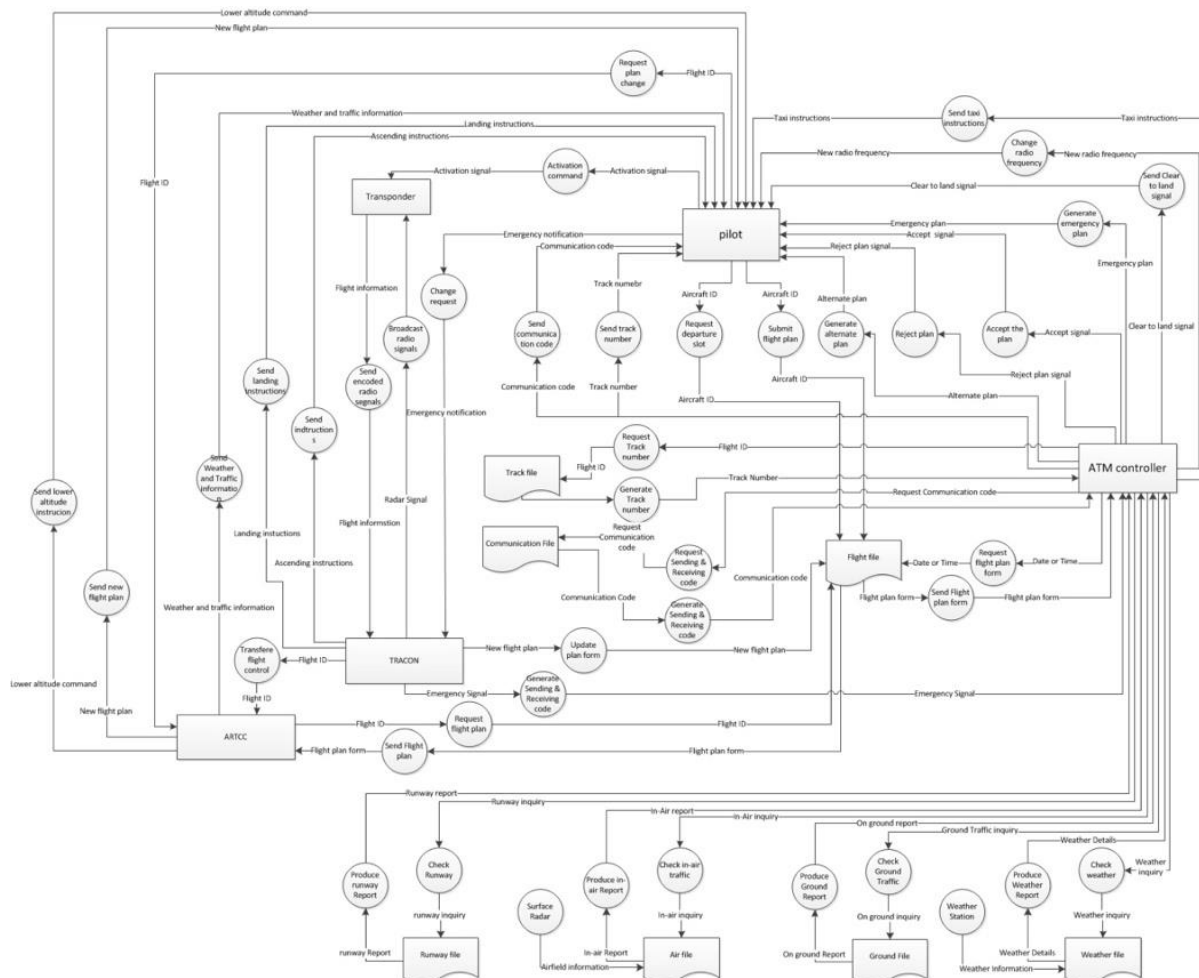


Figure (2) : ATC Data Flow Diagram

5. ATC Simulation Using GridSim

Gridsim provides a set of entities to support the simulation of a single processor or a multi-processor, diverse resource that might be a time shared or a space shared system. It supports simulation of geographically distributed system by setting the systems clock to different time zones. Gridsim contains entities that allow us to simulate network communication between resources. Multi-threaded entities are created by Gridsim during simulation, these entities are parallel entities that run in their own threads (Bedwal, Tayal, & Batra, 2014).

The entities provided by GridSim are as follows (Buyya and Murshed, 2002) :

- a. User Entity** : Grid users are represented by instances of the user entity which has a number of properties assigned to it, for example, type of job, scheduling optimization and activity rate.
- b. Broker Entity** : the broker is represented by the broker entity, it is responsible for scheduling user jobs,

and each user is connected to an instance of the broker entity.

- c. Resource Entity** : Grid resources are represented by instances of the resource entity, resources are differentiated from each other their characteristics, e.g. Number of processors, cost and speed of processing.

- d. Grid Information Service** : it is responsible for resource registration and tracking of available resources in the grid.

- e. Input and Output** : it represents the communication between gridsim entities though their I/O channels.

- f. Processing Elements** : PEs represent single instances of a CPU as a single processor.

- g. Gridlet** : it represents the jobs in a gridsim environment along with their information.

- h. GridStatistics** : responsible for recording statistical data.

- i. Shutdown Entity** : responsible for closing previously opened entities.

- j. Report Writer Entity** : it generates reports from the

GridsimStatistics entity.

Scheduling Policies

GridSim implements scheduling policies to accommodate user need, there are several scheduling policies e.g. time-shared or space-shared, moreover, users can create a custom scheduling policy that satisfies their specific needs (Sulistio, Cibej, Venugopal, Robic, & Buyya, 2002).

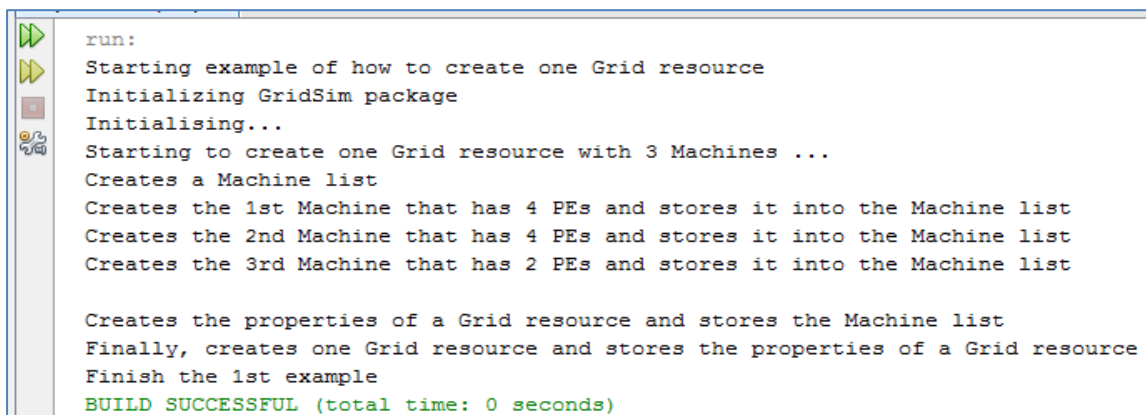
a. Time-shared Scheduling policy : when using time-shared scheduling, gridlets are assigned to PEs in such a manner that all gridlets get an equal share of the PE, this is done by identifying a time interval in which gridlets use PEs concurrently until their execution is finished (Sulistio, Cibej, Venugopal, Robic, & Buyya, 2002). Time-shared scheduling policy usually uses the Round-Robin Job scheduling policy, it is usually implemented on grid resources that have one PE or a shared memory multiprocessor (Buyya and Murshed, 2002).

b. Space-shared Scheduling Policy : this scheduling policy assigns gridlets to dedicated PEs where the gridlets are executed, space-shared scheduling assigns a single gridlet to dedicated PE where it is executed and all other gridlets that arrive will be queued until the

current gridlet using the PE is finished (Sulistio, Cibej, Venugopal, Robic, & Buyya, 2002). Space-shared scheduling policy is implemented on a First-Come First-Serve (FCFS) or a Shortest Job First (FJF) scheduling policy, this scheduling policy is usually used with grid resources that have distributed memory multiprocessing system, in some cases high-end SMP systems use space-shared scheduling policy (Buyya and Murshed, 2002).

5.1 Example 1

Example 1 demonstrates the process required to create one grid resource with the machines. The first two machines have 4 processing elements while the third one has two processing elements (PE). To create a grid resource, the Gridsim package must first be initialized, afterwards a machine list to hold the machines must be created, and then the machines will be created, each with a unique ID and number of PEs, each PE has a specific million instructions per second (MIPS) related to it, the next step is to create a resource characteristics object which holds the properties of the grid resource, such as Architecture, Operating System, List of Machines, Allocation policy, Time zone and price, Finally the grid resource object can be created.



```
run:
Starting example of how to create one Grid resource
Initializing GridSim package
Initialising...
Starting to create one Grid resource with 3 Machines ...
Creates a Machine list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list

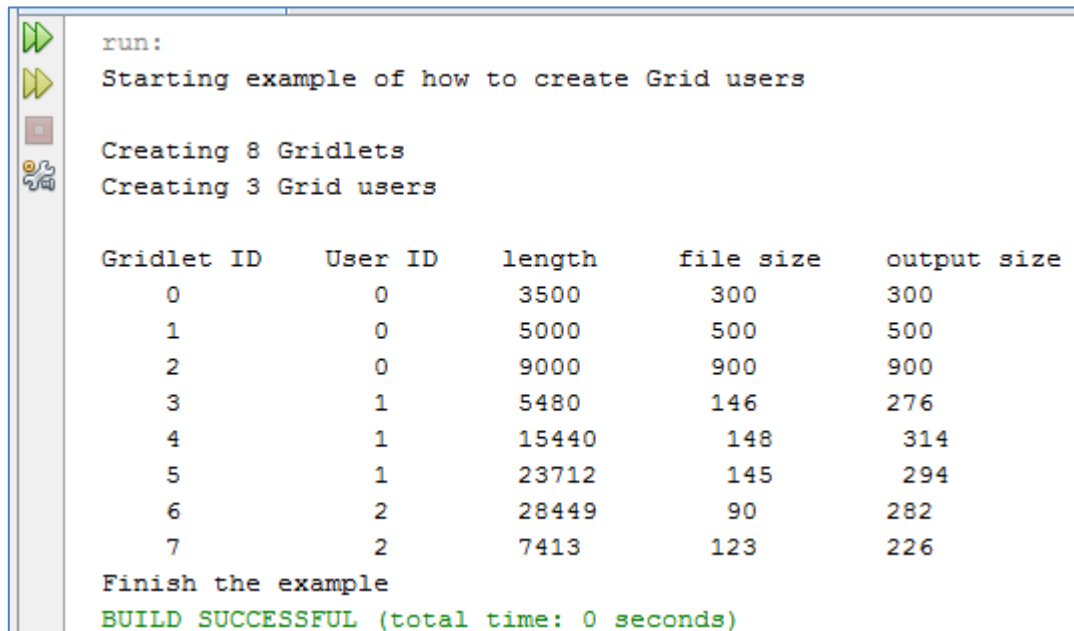
Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
Finish the 1st example
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure (3) : Example 1 output

5.2 Example 2

This example shows how to create grid users and gridlets, gridlets are assigned to grid users, they represent the instructions that a user wants to run. To create gridlets, first a list to hold the gridlets must be created, in this example three gridlets are created

manually while five others are created using GridSimRandom function. Gridlets have unique ID, Job length, File size and output size. Creating Grid Users involves creating a user list to store all the user information. Then gridlets can be assigned to grid users.



```
run:
Starting example of how to create Grid users

Creating 8 Gridlets
Creating 3 Grid users

Gridlet ID    User ID    length    file size    output size
0             0          3500      300          300
1             0          5000      500          500
2             0          9000      900          900
3             1          5480      146          276
4             1          15440     148          314
5             1          23712     145          294
6             2          28449     90           282
7             2          7413      123          226

Finish the example
BUILD SUCCESSFUL (total time: 0 seconds)
```

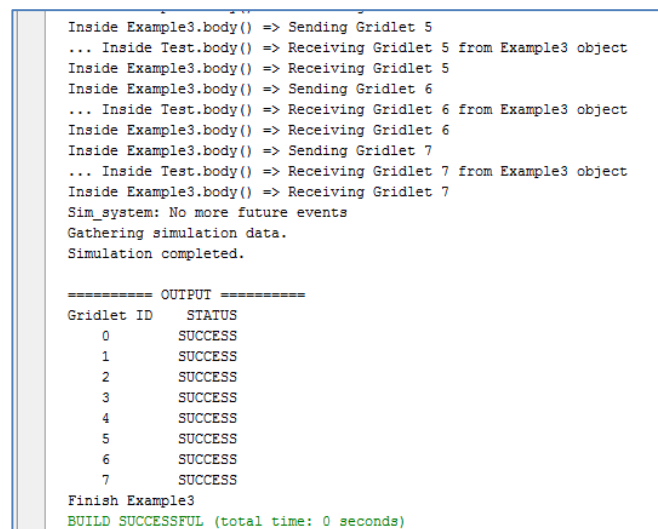
Figure (4) : Example 2 output

5.3 Example 3

The main objective of this example is to create gridlets and send them to other gridsim entities.

First it is necessary to initialize the gridsim package to create gridsim entities, then a list of gridlets must be created to be sent to the gridsim entity, then an object can be created (Example3 object) that has three parameters name, speed and list of gridlets, then simulation can be started and the gridlets can be sent to

another gridlist object (Test object) where they are received and stored using the GridsimTag.Schedule_now function to avoid scheduling issues. In the Test object input and output entities are established and the object listens to the event of simulation and waits to receive gridlets from the other entity (Example3 Object) one at a time, afterwards it sends the gridlets back to example3 object.



```
Inside Example3.body() => Sending Gridlet 5
... Inside Test.body() => Receiving Gridlet 5 from Example3 object
Inside Example3.body() => Receiving Gridlet 5
Inside Example3.body() => Sending Gridlet 6
... Inside Test.body() => Receiving Gridlet 6 from Example3 object
Inside Example3.body() => Receiving Gridlet 6
Inside Example3.body() => Sending Gridlet 7
... Inside Test.body() => Receiving Gridlet 7 from Example3 object
Inside Example3.body() => Receiving Gridlet 7
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

===== OUTPUT =====
Gridlet ID    STATUS
0             SUCCESS
1             SUCCESS
2             SUCCESS
3             SUCCESS
4             SUCCESS
5             SUCCESS
6             SUCCESS
7             SUCCESS

Finish Example3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure (5) : Example 3 output

5.4 Example 4

In this example gridlets are created and sent to a grid resource. At the beginning the gridsim package must be initialized and a grid resource object must be created, then an object is created (Example4 object) with a specific ID and speed and the simulation is started. Three machines are created each with a number of PEs for the grid resource, then a grid user is created that has 8 gridlets, each gridlet has a unique ID. Since Gridsim environment uses Multi-threaded environment the request acquiring the characteristics of grid resource

might arrive before the grid resource has been registered with the Grid Information Service (GIS) thus the request must be sent after the grid resource is registered. After the characteristics have been retrieved the event is recorded into a text file, all recordings are for statistical purposes, afterwards gridlets are sent to a grid resource entity and the event is recorded, then after the gridlet is sent back from the resource entity, this event is recorded again, the retrieved gridlets are all stored into a new gridletlist object, finally all entities are shut down including grid statistic entity.

```

Creating a grid user entity with name = Example4, and id = 9
Creating 8 Gridlets
Starting GridSim version 5.0
Entities started.
Received ResourceCharacteristics from Resource_0, with id = 5
Sending Gridlet_0 to Resource_0 with id = 5
Receiving Gridlet 0
Sending Gridlet_1 to Resource_0 with id = 5
Receiving Gridlet 1
Sending Gridlet_2 to Resource_0 with id = 5
Receiving Gridlet 2
Sending Gridlet_3 to Resource_0 with id = 5
Receiving Gridlet 3
Sending Gridlet_4 to Resource_0 with id = 5
Receiving Gridlet 4
Sending Gridlet_5 to Resource_0 with id = 5
Receiving Gridlet 5
Sending Gridlet_6 to Resource_0 with id = 5
Receiving Gridlet 6
Sending Gridlet_7 to Resource_0 with id = 5
Receiving Gridlet 7
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

===== OUTPUT =====
Gridlet ID   STATUS   Resource ID   Cost
0           SUCCESS    5          27.851468885941646
1           SUCCESS    5          39.78779840848807
2           SUCCESS    5          71.6180371352786
3           SUCCESS    5          12.260185935286579
4           SUCCESS    5          32.160803970344375
5           SUCCESS    5          26.989166622345806
6           SUCCESS    5          16.721823905496365
7           SUCCESS    5          20.974565689486667

Finish Example4
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure (6) : Example 4 output

5.5 Example 5

Example 5 demonstrates the creation of a grid user with one or more gridlets and sending them to a grid resource entity for processing. First of all the gridsim package is initialized then three grid resource objects are created by defining a machinelist object and storing machines in the list with their properties, and defining ResourceCharacteristics object to store grid resource properties. Then an object is created (Example5 object) which involves creating 8 gridlets and their properties

are set, afterwards the simulation is started. The gridlets are then assigned to grid resources, a waiting interval is essential for the grid resources to get registered with the GIS, the following events are recorded into a text file for statistical purposes, then grid resource characteristics are requested and gridlets are sent to their specific grid resources, afterwards the gridlets are sent back from the resource entity and are stored in a new gridlist object, finally all entities are shut down.

```

Waiting to get list of resources ...
Waiting to get list of resources ...
Received ResourceCharacteristics from Resource_2, with id = 13
Received ResourceCharacteristics from Resource_0, with id = 5
Received ResourceCharacteristics from Resource_1, with id = 9
Sending Gridlet_0 to Resource_0 with id = 5
Receiving Gridlet 0
Sending Gridlet_1 to Resource_2 with id = 13
Receiving Gridlet 1
Sending Gridlet_2 to Resource_0 with id = 5
Receiving Gridlet 2
Sending Gridlet_3 to Resource_0 with id = 5
Receiving Gridlet 3
Sending Gridlet_4 to Resource_0 with id = 5
Receiving Gridlet 4
Sending Gridlet_5 to Resource_1 with id = 9
Receiving Gridlet 5
Sending Gridlet_6 to Resource_1 with id = 9
Receiving Gridlet 6
Sending Gridlet_7 to Resource_1 with id = 9
Receiving Gridlet 7
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

===== OUTPUT =====
Gridlet ID   STATUS   Resource ID   Cost
0           SUCCESS      5   27.851458885941646
1           SUCCESS     13   39.78779840848807
2           SUCCESS      5   71.6180371352786
3           SUCCESS      5   12.260185925286578
4           SUCCESS      5   32.160803970344375
5           SUCCESS      9   26.989166622345806
6           SUCCESS      9   16.721823905496365
7           SUCCESS      9   20.974565689486667
Finish Example5

```

Figure (7) : Example 5 output

5.6 Example 6

Example 6 shows how gridlets can be submitted to grid resource entities using GridSim package.

First gridsim package is initialized and three grid resource objects are created each with 3 machines, machines have have specific id,number of PEs and MIPs, that are stored in the machinelist of the specific grid resource, then three grid users are created with

specific ID, Baud_rate and number of resources, afterwards the simulation is started. After the resources finish registering with GIS gridlets are allocated to grid resources randomly, the grid resource will modify the status of the gridlet to "Success" and all the actions are recorded into stat.txt file and the log is printed at the end.

```

User_1:Sending Gridlet_2 to Resource_0 with id = 5
User_2:Receiving Gridlet 1
User_2:Sending Gridlet_2 to Resource_1 with id = 9
User_0:Receiving Gridlet 2
User_0:**** Exiting body()
User_1:Receiving Gridlet 2
User_1:Sending Gridlet_3 to Resource_0 with id = 5
User_1:Receiving Gridlet 3
User_1:**** Exiting body()
User_2:Receiving Gridlet 2
User_2:Sending Gridlet_3 to Resource_1 with id = 9
User_2:Receiving Gridlet 3
User_2:**** Exiting body()
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

===== OUIPUT for User_0 =====
Gridlet ID   STATUS   Resource ID   Cost
0            SUCCESS    5            27.851458885941646
1            SUCCESS    5            39.78779840848807
2            SUCCESS    9            71.6180371352786

===== OUIPUT for User_1 =====
Gridlet ID   STATUS   Resource ID   Cost
0            SUCCESS    5            27.851458885941668
1            SUCCESS    13           39.78779840848807
2            SUCCESS    5            71.6180371352786
3            SUCCESS    5            3.0

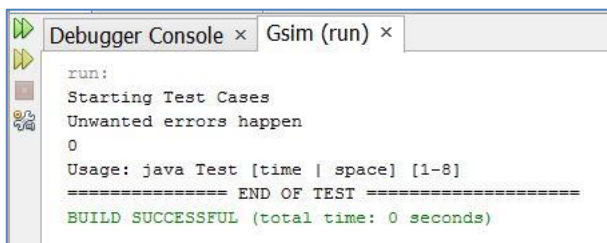
===== OUIPUT for User_2 =====
Gridlet ID   STATUS   Resource ID   Cost
0            SUCCESS    9            27.851458885941668
1            SUCCESS    5            39.78779840848807
2            SUCCESS    9            71.6180371352786
3            SUCCESS    9            21.94554472011822
Finish Example6
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure (8) : Example 6 output

5.7 Example 7

Example 7 demonstrates how gridlets can be submitted, canceled, paused, resumed or moved on grid resources or from one grid resource to another. This examples provides a set of test classes that represent cases in which gridlets can be sent for different actions e.g. test case 6 pauses a gridlet, move it and waits until it finishes execution. The first thing is to start the gridsim package and create 3 grid resources each with one machine that has 3 PEs, then three users are created each with 4 gridlets and simulation is started. At this point different gridlets are send to different test cases where the pass though different actions. The actions that occur on a specific gridlet are printed.



```

run:
Starting Test Cases
Unwanted errors happen
0
Usage: java Test [time | space] [1-8]
===== END OF TEST =====
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure (9) : Example 7 output

5.8 Example 8

Example 8 show how custom allocation policies are created.

The process begins with initializing the gridsim package, creating one grid resource and one user with four gridlets and starting the simulation. A delay is necessary for gridlets to be registered with GIS and then the available resources are aquired, then gridlets are sent to grid resources, here gridlets with even numbers require acknowledgement while gridlets with odd numbers don't. Here the grid resource must extend the AllocationPolicy class and implement 5 methods cancel, resume, pause, move and status. This class receives the gridlets, changes their status to success and send them back to parent class.

```

Time below denotes the simulation time.
Time (sec)      Description Gridlet #1
-----
0.00  Creates Gridlet ID #1
0.00  Assigns the Gridlet to User_0 (ID #9)
132.20 Allocates this Gridlet to GridResource_0 (ID #6) with cost = $3.0/sec
132.20 Sets Gridlet status from Created to Success

Gridlet #1, length = 600.0, finished so far = 0.0
=====

Time below denotes the simulation time.
Time (sec)      Description Gridlet #2
-----
0.00  Creates Gridlet ID #2
0.00  Assigns the Gridlet to User_0 (ID #9)
148.20 Allocates this Gridlet to GridResource_0 (ID #6) with cost = $3.0/sec
148.20 Sets Gridlet status from Created to Success

Gridlet #2, length = 200.0, finished so far = 0.0
=====

Time below denotes the simulation time.
Time (sec)      Description Gridlet #3
-----
0.00  Creates Gridlet ID #3
0.00  Assigns the Gridlet to User_0 (ID #9)
228.84 Allocates this Gridlet to GridResource_0 (ID #6) with cost = $3.0/sec
228.84 Sets Gridlet status from Created to Success

Gridlet #3, length = 300.0, finished so far = 0.0
=====

Finish Example8

```

Figure (10) : Example 8 output

5.9 Example 9

Example 9 demonstrates the steps required to create a custom grid resource and grid information service entity. First gridsim package is initialized without a GIS, then a new GIS entity is created where GridSim.setGIS(gis) is called on the GIS entity, afterwards a grid user and a grid resource are created, then simulation begins. To create our own GIS, a class that inherits from gridsim.GridInformationService class is created, then processOtherEvent() method is overridden so that new tags can be processed, moreover the connection from and to GIS must be done by I/O ports

Creating our own grid resource entity involves creating a class that registers new tags to GIS entity and receiving new tags from sender and printing a receive confirmation, to create a new grid resource first the parent constructors are called and the new grid resource is created with nam, baud_rate, ResourceCharacteristics, ResourceCalendar and ARPolicy parameters, I/O ports should be used for communication with Grid resource. In this example hello is registered and tags are tested to the custom GIS from the custom grid resource.

```

Output - Gsim (run) X
run:
Starting Example9
Initializing GridSim package
Initialising...
Creates one Grid resource with name = Resource_0
Creating a grid user entity with name = User_0, and id = 8
Starting GridSim version 5.0
Entities started.
Resource_0.registerOtherEntity(): register HELLO tag to NewGIS at time 0.0
Resource_0.registerOtherEntity(): register TEST tag to NewGIS at time 0.0
NewGIS: Received HELLO tag from Resource_0 at time 1.92
NewGIS: Received TEST tag from Resource_0 at time 2.88
User_0:Received ResourceCharacteristics from Resource_0, with id = 4
User_0: Sending TEST tag to Resource_0 at time 8.96
User_0: Sending HELLO tag to Resource_0 at time 8.96
Resource_0: received TEST tag from User_0 at time 9.9200000000000002
Resource_0: received HELLO tag from User_0 at time 10.8800000000000003
User_0:*** Exiting body()
NewGIS: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.
Finish Example9
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure (11) : Example 9 output

5.10 Example 10

Example 10 shows how to use basic advanced reservation functionalities like create, commit and status.

Primarily, gridsim package is initialized, then the current time is set by creating an instance of calendar and setting its value from calendar.getInstance() method. It's important that the start and end time must never be less than the initialization time. Afterwards five grid resources are created with specific parameters of name, total PE, number of machines, time zone and MIPs rating, the most important thing about creating these resources is setting their allocation policy to ADVANCED_RESERVATION, after that five users are created with specific parameters as well, the users that have an even number will have a time zone of GMT+8 while the ones with odd have a time zone of GMT-3, and then simulation is started.

Some reservation functionalities are explored in this example, such as :

- Request a new advance reservation
- Request a new immediate reservation
- Current timer as starting time
- Commit an accepted reservation
- Check reservation status

Here a list is created to hold the resources that support AR and their respective name and total PEs, afterwards the reservations are sent to grid resources and the results are acquired.

```

=====
User_0: query result = AR_STATUS_NOT_STARTED
=====
User_4: query result = AR_STATUS_NOT_STARTED
=====
User_2: Exiting body() with number of failed reservation is 0
User_0: Exiting body() with number of failed reservation is 0
User_4: Exiting body() with number of failed reservation is 0
User_1: Exiting body() with number of failed reservation is 0
User_3: Exiting body() with number of failed reservation is 0
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

===== OUTPUT for User_0 =====
Gridlet ID   STATUS   Resource ID   Cost
0            Success   21           1838.44800000000185
2            Success   21           940.48799999999976
3            Success   21           8138.06399999999984
1            Success   12           10836.0

```

Figure (12) : Example 10 output

6. Simulation scenario of ATC in GridSim

A scenario of ATC system in Gridsim can be simulated, to simulate such a system the gridsim package must first be initialized and a grid resource must be created that would resemble the whole flight simulation system, then seven machines must be created each to represent a phase of the flight, these machine and their entities will produce the machine object which are saved in a machine list, to explain the scenario the operations will be discussed phase by phase until the end. Phase one is the preflight, machine1 is responsible for this phase. Machine1 is implemented by creating 2 users and 6 PEs,

this procedure starts when user1 (Pilot) submits the flight plan to PE1, PE1 then request weather report from PE2, ground traffic report from PE3 and air traffic report from PE4, after PE1 receives all the request information and the flight plan it makes the decision, then PE1 request communication code, tracking number from PE5, PE6 respectively and send all information to user2 (ATC), if decision is accept user2 issues clearance to pull from gate to runway to user1 and send communication code and tracking number to user1. If PE1 rejects the plan then it generates an emergency plan and carries out the rest of the procedures with the new plan. Phase two is take-off which is represented in machine2, machine2 contains of 2 users and 4 PEs, machine2 receives information(flight plan) from machine1, then PE1 is activated, PE1 request ground report from PE2, air traffic report from PE3 and runway report from PE4, PE1 then makes a decision based on the reports and forwards the decision to user1 (ATC) were, if there is no complication, user1 send clear to take-off signal to user2(Pilot). Phase three is departure, here machine3 takes over from machine2, machine3 is made up of 2 users and 1 PE, PE1 receives the information(flight plan) from the previous phase and checks the flight plan, then it generates a new radio frequency and forwards it to user1(departure controller) which in turn it forward it to user2(pilot). Phase four is en-route, machine4 is responsible here which has 2 users and 1 PE, it receives information(flight plan) from machine3 and activates PE1, PE1 checks flight plan and provides direction and altitude information to user1(ARTCC) which provides altitude and directions to user2(pilot). Phase five is descend, this is represented in machine5 which consists of 2 users and 1 PE, PE1 receives information(flight plan) from previous phase and checks the flight plan, then it generates new altitude and direction and forwards them to user1(Approach controller), user1 then send the information to user2(pilot). Phase six is approach, machine6 takes over here, 2 users and 1 PE are created for this phase, machine6 receives information(flight plan) from machine5 and sends them to PE1, PE1 then checks the information and produces a clearance to approach signal to user1(approach controller) were in turn it send a clear to approach signal to user2(pilot) Phase seven is landing, here machine7 takes over, machine7 has 4 PEs and 3 users, first PE1 investigates the flight plan and request air traffic report from PE2 and runway report from PE3, then based on the information it makes a decision and send it to user1(local controller), if there is no problem, user1 send land signal to user2(pilot), after landing user3(ground controller) takes over, user3 request ground report from PE4 and based on the report sends directions to user2.

All implementation of the procedures and activities of PEs are saved in the body().

Figure (13) : below is a Proposed GUI for the system.

7. Conclusion

Simulation gives us the ability to test real world events and programs without taking risks associated with failure, simulation may have many purposes e.g. performance optimization, safety, testing and education. ATC is a very critical system that provides safety and orderly movement for air traffic around the world. ATC process may oppose great threat to life if they were to be applied without simulation. In this paper, the simulation of ATC in Gridsim Toolkit was studied. Gridsim toolkit provides a comprehensive set of entities and services to aid us in the process of simulation for a given program. In the beginning, the ATC system was discussed with its divisions and all phases a flight takes from source to destination. Then a Data Flow Diagram design for the whole system was presented, also the simulation scenario of ATC system in Gridsim was discussed along with implementation and a GUI of the scenario.

References

1. Bedwal, T., Tayal, R., & Batra, A. (2014). Grid Computing and GridSim Toolkit : An overview. *International Journal of Computer Applications*.
2. Buyya, R., & Murshed, M. (2002). Gridsim : A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation : practice and experience*, 1175-1220.
3. Conrad, L. (2010, April). Air Traffic Management System. Retrieved March 2014, from NASA : [http : //archive.is/5VROb](http://archive.is/5VROb)
4. Freudenrich, 2001, C. (2001, June 12). How Air Traffic Control Works. Retrieved March 2014, from Science : [http : //science.howstuffworks.com/transport/flight/modern/air-traffic-control.htm](http://science.howstuffworks.com/transport/flight/modern/air-traffic-control.htm)
5. Sulistio, A., Cibej, U., Venugopal, S., Robic, B., & Buyya, R. (2002). A toolkit for modelling and simulating Data Grids : An extension to GridSim. *CONCURRENCY AND COMPUTATION : PRACTICE AND EXPERIENCE*, 1591-1609.
6. Wu, H., & Zhou, M. (2012). Modeling Next Generation Air Traffic Control System with Petri Nets. *International Journal of Intelligent Control and Systems*, 53-68.