A discrete-event simulation model of air traffic flow is developed to take advantage of the computational efficiency afforded by event-based simulation. The model is based on a set of reusable library block components that may be used to build models of varying degrees of complexity. The building blocks of discrete-event simulation are described. To illustrate the modeling technique for air traffic flow management, both small-scale and large-scale models of air traffic flow are developed. Example results are presented to show how event-based simulation may be used to perform air traffic flow management research.

I. Introduction

Air Traffic Flow Management (TFM) involves the modeling, analysis, simulation, and control of air traffic flows in the national airspace system (NAS). With the understanding that safety is the highest priority of air traffic management, the objective of TFM is to determine the most efficient air traffic flow control procedures. As the amount of traffic in the NAS has increased, the system has become brittle; storms that shut down jet routes or runways in one region may have significant adverse effects on flights in other parts of the NAS. It is often difficult to recover from these events. Researchers have been investigating ways to improve the robustness and efficiency of air traffic flow control. Research into new TFM techniques is currently a major component of the FAA’s Next Generation Air Transportation System (NextGen).

About ten years ago, researchers began investigating aggregate or Eulerian flow models of air traffic. These studies were inspired by the use of Eulerian models in fluid flow networks and their application to automobile traffic modeling, analysis, and simulation. Menon was among the first to develop a linearized Eulerian flow model of air traffic.\(^1\) A similar aggregate flow model was concurrently developed by Sridhar et al. at NASA Ames Research Center.\(^2\) These flow models enabled researchers to model complex traffic flow patterns using low-order linear state space models with possibly time-varying coefficients. Instead of simulating the detailed trajectories and interactions of tens of thousands of aircraft (which would consume large amounts of computing resources), aggregate flow models reduced the number of model states to just the number of airspace areas involved in the simulation. For investigating Air Route Traffic Control Center (ARTCC) capacities, the number of states is reduced to around 25 for the entire continental United States.

Once aggregate flow models were established, researchers began to use them to perform computationally intensive optimization studies and Monte Carlo simulations. Grabbe et al. used an aggregate model to investigate optimal constraint recovery strategies.\(^3\) Their work highlighted the utility of using computationally efficient aggregate flow models for optimization studies. Multidisciplinary optimization is another area in which efficient network flow models could be an enabling technology.\(^4\)

Discrete-event simulation (DES) provides an alternate way to achieve computational efficiency and to enable larger scale simulations.\(^5\) DES has become an increasingly common tool for analyzing systems that are characterized by discrete and asynchronous events. The air traffic system is a network and therefore may be modeled, analyzed, and simulated using DES techniques. By using DES, researchers can quickly build complex simulation models of air traffic scenarios. Instead of spending time investigating how to develop elaborate system models, they can concentrate their efforts on studying the behavior of these systems. DES can also be applied to generate realistic traffic scenarios for use in other research. Computational benefits are realized in DES by eliminating the need to integrate either differential or difference equations over a large time span with small time steps. As with aggregate flow models, the number of states in a discrete-event simulation is proportional to the number of airspace regions and airports in the simulation. In contrast with aggregate flow models, DES maintains information about individual

---

1 Engineering Consultant, Consulting Services Group, 3 Apple Hill Drive, Associate Fellow AIAA.
2 Support Engineer, Engineering Development Group, 3 Apple Hill Drive.
3 Technical Marketing Engineer, Product Marketing Group, 3 Apple Hill Drive.
flights as they traverse the simulation. This enables researchers to more easily compute properties for individual aircraft than in the case of aggregate flow models. Modeling individual aircraft comes with additional computational and storage size requirements, but the additional required resources scale linearly, rather than exponentially, with the number of aircraft in the simulation.

In this paper, network models of the NAS are developed to illustrate key concepts and features of DES. The models shown here have been developed using MATLAB®, Simulink® and SimEvents® software products, though the illustrated concepts may be implemented in other software environments. The basic modeling concepts are described first, followed by a discussion on the scope of the example models that are developed. The next section describes the handful of modeling components that may be used to build up complete simulations of the NAS. Both small-scale and large-scale models are then described, and example simulation results are presented.

II. Discrete-Event System Modeling

DES can be a much more computationally efficient means of simulating systems that are better described as a sequence of events rather than as a detailed time based integration of either continuous or discrete states. Instead of having to update states at regular time increments, DES can skip over potentially large time steps because the behavior of the system does not change in between events. This section presents some of the basic concepts of DES. A more complete treatment of DES may be found in the literature.

A. Entities, Attributes, and Events

DES deals with the flow of entities through a simulated network based on the attributes of each entity and any events that occur. In the case of air traffic flow, an aircraft may be modeled as an entity, which might have the following types of attributes:

- Flight identifier
- Departure airport information
- Destination airport information
- Climb-out parameters (for example, average climb-out time for each flight)
- Descent time
- Departure time
- Cruise speed
- Trajectory plan (more detailed than a flight plan)

Events are instantaneous transitions that produce a change in a state variable or an output, or that trigger other events. Example events in an air traffic flow simulation might be:

- Aircraft departs from a particular runway at a particular airport
- Aircraft transitions from one region of airspace to another
- Aircraft enters a holding queue because of downstream congestion

B. Servers and Queues

Servers are operators that act upon entities for some amount of time (Fig. 1). The time may be constant for any entity that is served, or the time may be dynamic based upon either an input signal or an entity attribute. In the case of an air traffic model, a runway could be modeled as a server with the service time either set to a constant or varied depending upon the type of aircraft landing and the aircraft’s destination terminal gate.

There are three main types of servers: the single-Server, the N-Server, and the infinite-Server. The single-Server can process only one entity at a time. That entity will be processed for the amount of time specified by the Service Time parameter after which another entity may enter the server. The N-Server can process up to N entities in parallel, and the infinite-Server can process an infinite number of entities in parallel. In the air traffic example, if an airport had three runways, then the runway model would use an N-server with the number of servers set to three.
A queue may be thought of as a container that can be filled until it reaches capacity and emptied if the exit to the queue is open (not blocked). The queue capacity is generally selected as a parameter (Fig. 2). In the air traffic example, a runway may have a departure queue that is blocked whenever the runway is occupied by either a departing or arriving aircraft. When the runway is occupied, any new aircraft requesting departure from that runway will be blocked and will have to enter a departure queue. Once the runway is clear, the departure queue can release an aircraft to depart from the selected runway. The three Queue blocks available in the SimEvents software product are First-in-First-Out (FIFO), Last-in-First-Out (LIFO), or Priority. The Priority Queue assigns output priorities to entities based on the value of a selected attribute.

By combining servers and queues, more interesting units can be created. For example, an airport would be modeled by one or more servers to represent one or more runways. The airport would also have two queues: a holding queue for arriving aircraft and a runway queue for departing aircraft.

C. Signal Routing: Enable Gates and Switches

Enable Gates are used to enable the flow of entities when the enable input is high (greater than zero) or to block the flow of entities when the enable input is low (zero) (Fig. 3). For example, an Enable Gate could be used in an airport subsystem to control the flow of departure aircraft to the runway. This flow would normally be enabled, but
when the number of aircraft queued up for the runway exceeds a preset threshold, the gate would be disabled to block departing aircraft from the runway.

D. Network Topologies

Different network topologies may be modeled using DES, including Ring, Mesh, Star, Fully-Connected, Line, Tree, and Bus Network topologies (Fig. 5). Each topology has different strengths and weaknesses, which may make it more or less suitable for modeling a particular network. For example, a Mesh Network might be a more physically representative network model of air traffic flow since it can model geographic proximity directly. However, a Star Network might be a better choice for producing large-scale simulations. The example models developed in this paper use a Star Network topology because of its relative computational efficiency when scaled up to large numbers of airspace areas and because it simplifies network connections within the model. A complete discussion of the utility of each of these network topologies is beyond the scope of this paper and will not be included.
III. Discrete-Event Simulation of the NAS

TFM research investigations tend to focus on either the analysis of recorded or generated data or on the synthesis of new flow control procedures based on recorded or generated data. Analysis investigations are used to better understand how high-delay situations evolve. For example, heavy thunderstorm activity might close down a few key jet routes and airports during a time of high-volume traffic. This in turn causes delays to ripple through the NAS. Being able to quickly simulate this situation by replaying data through DES may provide valuable insight into key bottlenecks in the system so that new airspace configurations or ground delay procedures may be devised. Synthesis investigations might involve parametric studies (Monte Carlo simulation or randomized optimization) to minimize delay by optimizing arrival rates, departure rates, or sector boundaries. If Monte Carlo simulations are required to achieve statistical significance in the results, the NAS simulations must be run many times and therefore must be computationally efficient.

A. Air Traffic Flow Modeling Requirements

For analysis or synthesis studies, the requirements of a DES of the NAS are similar. The most important metric to compute in a simulation of the NAS is flight delay. Delay must be computed both at the unit level (for example, runway, airport, or sector) and at the system level (total delay in the NAS). Other metrics that might be of interest include flight distances, fuel economy, number of passengers serviced, and the number of canceled flights, among others.

To support parametric optimization studies or Monte Carlo simulations, the DES model should be parameterized so that a single structural model can run many different scenarios by changing parameter values. Examples of parameters that are of interest for TFM simulations are the capacities of each airspace area or airport, the number of runways at each airport, and the size of holding pattern queues. These parameters are straightforward to implement in DES. Aside from computational efficiency, ease of development and scalability are important considerations for TFM simulations.

B. Modeling Components

A number of modeling components have been created and can be combined and interconnected to form complete network models of traffic flow. For air traffic flow modeling, components are needed for preprocessing scenarios, generating aircraft entities, modeling airspace volumes (for example, ARTCC or sectors), and modeling sources and sinks (airports). Each of these components is described in this section.
1. Aircraft Generator Component

The Aircraft Generator component is responsible for parsing all of the preprocessing information in the simulation to initiate aircraft flights at the correct times and to route the aircraft to the correct nodes (airports, sectors, or Centers) in the network. In the example shown in Fig. 6, the **AllDepartures** block creates aircraft according to the scheduled departure times determined in the preprocessing phase and starts a timer for each aircraft as it is created so that total flight time may be calculated. The **Get Attribute**, **Discrete Event Subsystem**, **Single Server**, and **Set Attribute1** blocks determine initial flight properties for aircraft based on the index of the current aircraft. The **OperatorFunction** block performs initial routing of the aircraft to the correct departure airport and routes the aircraft out of the Generator component to the Operator component.

There are three parameter inputs to the Aircraft Generator Block. The first is the **Departures** Structure, which is a MATLAB structure variable. It contains seven elements that describe basic aircraft properties (Table 1). Each of the elements is a 1 x N row vector, where N is the number of aircraft in the simulation.

<table>
<thead>
<tr>
<th>Departure Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FID</td>
<td>Flight Identification number</td>
</tr>
<tr>
<td>Index</td>
<td>Integer index of each aircraft relative to one another</td>
</tr>
<tr>
<td>DepartureAirport</td>
<td>Integer index of the departure airport for each aircraft</td>
</tr>
<tr>
<td>DestinationAirport</td>
<td>Integer index of the destination airport for each aircraft</td>
</tr>
<tr>
<td>ClimbOutTime</td>
<td>Amount of time required for each aircraft to climb to top of ascent</td>
</tr>
<tr>
<td>DescentTime</td>
<td>Amount of time required for each aircraft to descend to the runway</td>
</tr>
<tr>
<td>Schedule</td>
<td>This is the time at which the aircraft entities are generated</td>
</tr>
</tbody>
</table>

The **AllPaths** and **AllLegs** parameters of the Aircraft Generator Block are both N x M arrays, where N is again the number of aircraft in the simulation and M is the maximum number of Centers (or airspace regions) that an aircraft can travel through. The **AllPaths** array contains the route information for each aircraft as a sequence of Center numbers that the aircraft will travel through. The **AllLegs** array contains the transit time for each aircraft for each of the Centers that the aircraft will travel through.
2. Airport Component

The Airport component is comprised of lower-level queues, gates, servers, and switches (Fig. 7). These components determine whether an aircraft entity is taking off or landing and then route the aircraft accordingly. The queuing blocks enable the Airport component to model delays when downstream components are blocking due to over-capacity situations.

The following parameters have been included in the airport model to enable their values to be modified for simulation:

- **Name**: Airport Identifier
- **# Planes to force ground hold**: Maximum number of aircraft allowed in both the landing and departure queues before the queues block further aircraft from entering.
- **Number of runways**
- **Runway time**: Runway mean occupancy time
- **Circling capacity**: Maximum queue length for aircraft placed in arrival queue holding patterns

The TrafficLogic subsystem within the Airport subsystem determines whether an aircraft is departing from or arriving at the airport and then routes the aircraft accordingly. This subsystem also implements both the runway (departure) queue and the circling (arrival) queue. Another subsystem within the TrafficLogic block disables departure traffic when the combined number of aircraft in both the departure queue and the arrival queue exceeds a threshold value.

The OUT1 branch from the Output Switch block routes the departing aircraft to the Operator subsystem after determining the next Center to route the aircraft to and then holds the aircraft in the ClimbOut server for an amount of time set by the aircraft ClimbOutTime attribute. The OUT2 branch from the Output Switch block routes the
landing aircraft to be removed from the simulation. The simulation time for each aircraft is read and recorded along with the aircraft ID.

The AirportOut output from the Airport block is used to route statistics out of the Airport subsystem for display in root-level scopes.

3. Center (airspace region) Component

A Center may be considered as a model of a region of airspace such as an ARTCC, a sector, or an abstract airspace region. The Center model (Fig. 8) is comprised of a tunable N-server and a few logging and scope blocks. The Center capacity is a tunable parameter that allows users to change the number of aircraft that can reside in the Center at any given time by changing the number of servers in the Tunable_NServer block. The Operator Function determines the downstream routing for each aircraft entity as it passes through the block.

Figure 8. Center model component
4. **Operator Component**

The Operator component is the router of the Star Network topology chosen for this network design (Fig. 9). It determines where to route each aircraft based upon its current attributes. The Operator does not contain any memory components; it is merely a routing mechanism. As a result, aircraft entities never spend any time inside the Operator itself. If downstream Centers or Airports are blocked due to over-capacity situations, then the Operator component propagates the delay upstream and won’t allow aircraft entities to enter the Operator until the downstream blocking condition is resolved.

The dialog box of the Operator subsystem provides a mechanism for the user to automatically generate the correct number of inputs and outputs for the Operator. The user can specify the number of Centers and the number of Airports that must connect to the Operator. A callback function processes this information and automatically generates the blocks and connections within the Operator block. This is an especially useful capability for large-scale cases in which many centers and airports are simulated.

C. **Input and Output Data Processing**

The preprocessing functions enable users to create a simulation scenario by defining attributes about each aircraft, airport, and trajectory in the simulation. The preprocessing required to generate simulation data may be categorized into one of the following functional groups: Initialization, Aircraft and Flight Data Generation, Model Creation, and Post-Simulation Analysis.

The Initialization sets up the computational environment in preparation for simulation and sets parameters that define the scope of the simulation, including the size of the airspace being investigated, the number of airports, the number of flights, and the types of routes being flown (for example, Great-Circle or Rhumb Line). Aircraft and
Flight Data Generation involves the actual generation of flight routes for each aircraft. If a source of flight plan data is to be used instead of generated route data, this function could instead simply read and process the recorded data in preparation for simulation. The Model Creation may either be done programmatically or graphically by adding the desired component subsystems and connecting them together. Smaller scale simulations would typically be created graphically while larger scale simulations would usually be generated programmatically. Finally, the Post-Simulation Analysis functions display the output data from the simulation. Typical output displays are delay histograms and graphical animations that display the number of aircraft in different regions of airspace via a color-coded map.

IV. Small-Scale Model

A small-scale model has been created to assist with integrated testing of the air traffic discrete-event simulation components. Using this model, simple scenarios may be quickly generated and the results analyzed to confirm that the simulation results agree with expectations.

A scenario with four Centers, three airports, and two flights has been created (Fig. 10). The Centers represent four quadrants on a Cartesian grid and three of the Centers have airports associated with them. Center 1 has the Green airport, Center 2 has the Blue airport, and Center 3 has the Red airport. Flight 1 is scheduled to depart from the Green airport, fly through Center 1, fly through Center 3, and then land at the Red airport. A short time after Flight 1 takes off, Flight 2 departs from the Red airport headed for the Blue airport in Center 2. The flight route is from Center 3 to Center 4 and then to Center 2 where Flight 2 then lands at the Blue airport. This scenario creates a situation in which both aircraft reside in Center 3 for a short amount of time. The block diagram corresponding to this situation is shown in Fig. 11.

![Figure 10. A simple air traffic flow scenario](image-url)
In the first simulation, Case 1, the capacity of each center is set to a number greater than two so that both aircraft are permitted to occupy Center 3 at the same time. Plots of the number of aircraft in each center show how the aircraft traverse through the Centers and how both aircraft occupy Center 3 for a short time (Fig. 12).

Figure 11. Block Diagram of small scale scenario

Figure 12. Center aircraft counts for Case 1
In Case 2, the capacity of Center 3 is set to one so that both aircraft will no longer be allowed to occupy Center 3 at the same time. Since Flight 2 enters Center 3 first, this will block Flight 1 from entering until Flight 2 has left Center 3 and entered Center 4. This causes Flight 1 to be delayed by around 40 minutes compared to Case 1 (Fig. 13).

![Figure 13. Center aircraft counts for Case 2](image)

V. Large-Scale Model

The same components that were used for the small-scale model may be used to create much larger scale models of more realistic air traffic flow scenarios. For example, simulating most of a day’s worth of Class-A air traffic over the continental United States requires 20 ARTCC Centers, approximately 200 airports, and thousands of flights. Although the computational run time for models of this size is not significant (on the order of a few seconds on a ThinkPad T60 laptop with 2GB RAM), the compile time—the time required to compile a model into a parameterized executable process—begins to grow significantly as the number of functional blocks and connections grows. For the example with 20 Centers, 200 airports and thousands of flights, the compile time is measured in hours. Once the model is compiled, however, the run time is measured in seconds. After compiling a model once, a user can run many parallel simulations very rapidly. This powerful concept may be used to generate a statistically significant set of results in a short time.

A full-scale model of traffic flow across all ARTCC in the continental United States has been created. The true geographic boundaries of each ARTCC are used and a representative subset of 78 actual airports has been included in the simulation. Because of the large scale, developing such a model manually by connecting components in a graphical user interface can be burdensome. Simulink® and similar software tools provide application programming interfaces (APIs) that can be used to programmatically generate large-scale models. In this case, the large-scale model has been created automatically based on a few definition files. The block diagram of the large-scale model is too large to be displayed in this paper, but the map shown in Fig. 14 was generated from the actual ARTCC boundary data used for the simulation.
A. Monte Carlo Simulation Example

A Monte Carlo simulation has been run to determine the effect on system delay of reducing the capacity of the Kansas City ARTCC (ZKC). This Center is located in the middle of the United States and its capacity is presumed to affect traffic between many different city pairs. Two-thousand randomly-generated flights have been run 20 times in this case with ZKC capacity values ranging from 25 to 6. Reduced capacity could arise from bad weather, RADAR outages, or air traffic control personnel limitations. As is recommended, this model was compiled into a parameterized executable process to eliminate the need to recompile the model for each run of the simulation.

Since the results of these simulations are based upon randomly generated traffic data, they are not valid in an absolute sense, but the relative trends of the results are of interest. With a capacity limit of 25 aircraft on the ZKC Center, little or no delay is experienced. As the capacity limit is reduced, there is a dramatic rise in the amount of delay in the system (Fig. 15).

Figure 14. Large-scale model of air traffic over the continental United States.
VI. Conclusion

This paper illustrates the application of discrete-event simulation to efficiently model air traffic flow at a scale that is useful for Monte Carlo simulation and randomized optimization studies. A small-scale example was developed to illustrate the DES modeling features and a large-scale example was used to show how a simulation of the entire NAS can be easily developed from the same basic modeling components. Results of a Monte Carlo simulation with the 20 real ARTCC Centers, 78 airports, and 2000 flights were presented. The simulation investigated the effects of reducing the capacity of the Kansas ARTCC and showed how reduced capacity of this key Center can dramatically increase delay across the entire system.

VII. Acknowledgments

The authors would like to thank Dr. Gano Chatterji of the NASA Ames Research Center for providing information on the mapping of ARTCCs.

Figure 15. Results of large-scale Monte Carlo simulation showing average delay per flight as a function of ZKC capacity
VIII. References


