Design and Implementation of a Sector-Based Airspace Model for the MIT Extensible Air Network Simulation

by

Colin J. Whittaker

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author . . . [illegible] . . .
Department of Electrical Engineering and Computer Science
May 19, 2006

Certified by . . . [illegible] . . .
John-Paul Barrington Clarke
Principal Research Scientist
Thesis Supervisor

Accepted by . . . [illegible] . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Design and Implementation of a Sector-Based Airspace Model for the MIT Extensible Air Network Simulation

by

Colin J. Whittaker

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2006, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The MIT Extensible Air Network Simulation (MEANS) is a tool that has been designed to assist airline schedulers and air traffic managers in predicting flight delays for given air traffic scenarios. One aspect of the simulation, the determination of flight times, has received criticism from the MEANS users as being too simplistic for their needs. Currently, MEANS predicts flight times based on a historical distribution of observed flight times between city pairs. This system ignores the effects of flight level winds and airspace congestion, two major determiners of flight time.

The replacement flight time model presented divides the airspace into discrete sectors based on existing divisions in air traffic control. Each sector has its own wind conditions and capacity limitations which affect passing flights. Results show that, after some calibration, the new flight time model produces accurate flight times when the airspace is divided into ARTCC domains and does not introduce additional errors into other parts of the simulation. Additionally, test scenarios show that the new system is capable of modeling airspace capacity events, such as a radar failure. Comparative results reveal that the old, distribution model produces surprisingly accurate flight times for typical wind conditions and airspace utilization.

Thesis Supervisor: John-Paul Barrington Clarke
Title: Principal Research Scientist

Acknowledgments

A number of people have contributed to this work, and I would like to acknowledge them here:

Professor J.P. Clarke for welcoming an outsider into the aerospace world.

Terran Melconian for developing MEANS and its related tools, keeping our servers running, finding my bugs, and much more.

Jonathan Histon for settling the debate on how many planes can fit in the sky at one time.

Bob Hoffman of Metron Aviation for coming through with the sector boundary data.

Elizabeth Bly, Robin Riedel, and everyone else for building MEANS into what it is today.

Emily Egan for everything.

And my family for being there for me.

Contents

1 Introduction and Motivation **13**

1.1 Structure of the NAS 14

1.1.1 The Role of Sectors in the NAS 15

1.1.2 The Role of Flight Plans in the NAS 16

1.2 Structure of MEANS 16

1.2.1 Gate Module 19

1.2.2 Tower Module 19

1.2.3 En Route/Airspace Module 20

1.3 Additional Desired Properties for MEANS 20

2 Design of the Sector-Based Airspace Module **23**

2.1 Design Overview 23

2.1.1 The Legacy Airspace Specification 24

2.1.2 Required Changes to the Specification 25

2.2 The Sector-Based Airspace 26

2.2.1 Sectors 27

2.2.2 Flight Plans 28

2.3 Sector Design 29

2.3.1 Capacity and Scheduling 29

2.3.2 Sector Transitions 31

2.3.3 Variations of Flight Acceptance Rules 32

2.4 Argument for Miles-In-Trail Simulation 33

3 Implementation Details and Flight Plan Generation **37**

3.1 Airspace Implementation 37

3.2 Sector Implementation 39

3.2.1 Sector Data Structures 42

3.2.2 Requirements and Invariants 44

3.2.3 Schedule Operations and Correctness 45

3.3 Flight Plan Generation 49

4 Results and Validation **53**

4.1 Experimental Setup 54

4.2 Results 55

4.2.1 Actual Delay Statistics 55

4.2.2 Historical Distribution Implementation 56

4.2.3 Sector-Based Implementation: Center Granularity 59

4.2.4 Sector-Based Implementation: Sector Granularity 62

4.2.5 Case Study of New York Area 65

4.3 Analysis 70

4.3.1 Flight Duration Correlation 71

4.3.2 Acceleration Considerations 74

4.3.3 Additional Overload Slots 78

5 Conclusions and Future Work **79**

5.1 Future Work 80

5.1.1 Acceleration Compensation Calibration 80

5.1.2 Aircraft Cruising Speeds 81

5.1.3 Improved Flight Plans 81

5.1.4 Improved Flight Level Wind Data 81

5.1.5 New Airspace Induced Ground Hold Rules 81

List of Figures

1-1 ARTCC regions 14

1-2 The relationship of component modules in MEANS 17

3-1 Calculation of sector crossing time 41

4-1 Actual flight delays 55

4-2 Flight delays predicted by the historical distribution implementation . 56

4-3 Deviations from actual delays for the historical distibution implementation 57

4-4 Deviations from actual flight times for the historical distribution implementation 58

4-5 Flight delays predicted by the sector-based implementation at center granularity 59

4-6 Deviations from actual delays for the sector-based implementation at center granularity 60

4-7 Deviations from actual flight times for the sector-based implementation at center granularity 61

4-8 Flight delays predicted by the sector-based implementation at sector granularity 63

4-9 Deviations from actual delays for the sector-based implementation at sector granularity 63

4-10 Deviations from actual flight times for the sector-based implementation at sector granularity 64

4-11 Deviations from actual delays in the New York area for the historical distribution implementation 65
4-12 Deviations from actual delays in the New York area for the sector-based implementation at center granularity 66
4-13 Deviations from actual delays in the New York area for the sector-based implementation at sector granularity 67
4-14 Deviations from actual delays in the New York area with failures for the sector-based implementation at centers granularity 68
4-15 Flight duration correlation for the historical distribution implementation 72
4-16 Flight duration correlation for the sector-based implementation at center granularity 72
4-17 Flight duration correlation for the sector-based implementation at sector granularity 73
4-18 Flight delays predicted by the sector-based implementation at center granularity with modified flight plans 75
4-19 Deviations from actual flight times for the sector-based implementation at center granularity with modified flight plans 75
4-20 Flight delays predicted by the sector-based implementation at sector granularity with modified flight plans 76
4-21 Deviations from actual flight times for the sector-based implementation at sector granularity with modified flight plans 77

List of Tables

4.1 Statistics for figure 4-1 55

4.2 Statistics for figure 4-2 56

4.3 Statistics for figure 4-3 57

4.4 Statistics for figure 4-4 58

4.5 Statistics for figure 4-5 60

4.6 Statistics for figure 4-6 60

4.7 Statistics for figure 4-7 61

4.8 Statistics for figure 4-8 63

4.9 Statistics for figure 4-9 64

4.10 Statistics for figure 4-10 64

4.11 Statistics for figure 4-11 66

4.12 Statistics for figure 4-12 66

4.13 Statistics for figure 4-13 67

4.14 Statistics for figure 4-14 69

4.15 Statistics for figure 4-18 75

4.16 Statistics for figure 4-19 76

4.17 Statistics for figure 4-20 76

4.18 Statistics for figure 4-21 77

4.19 New statistics for figure 4-9 with additional extra slots. Compare to table 4.9 78

Chapter 1

Introduction and Motivation

The MIT Extensible Air Network Simulation, or MEANS, is a modular simulation tool for researchers to investigate various strategies in air traffic management and airline operations. MEANS provides a high-level simulation of the United States National Airspace System (NAS), where each flight is treated individually. However, MEANS does not attempt to calculate exact flight trajectories or simulate complex flight rules. Instead, MEANS generates realistic flight and passenger delay profiles for the scenario that is input. The complexity of the simulation is minimized, and MEANS executes very quickly, normally simulating a day's worth of air traffic in under five minutes. A researcher can realistically run hundreds of randomized simulations to reduce the impact of errors introduced by simplification on the resulting operational data. This data can then be used in financial models of airline operations to determine the potential cost of the schedule or strategy under investigation. [14]

MEANS provides users with a degree of flexibility through its modular design. Each aspect of the simulation has a number of different interchangeable implementations. The implementations differ on the amount of detail simulated and the data required for execution. Users can choose the implementation that to best suits their needs. If none of the available implementations suit the user's needs, they may extend MEANS with a custom implementation of their own. For example, Elizabeth Bly S.M. 2005 built a custom control tower implementation to optimize takeoff and landing queue ordering. [3]

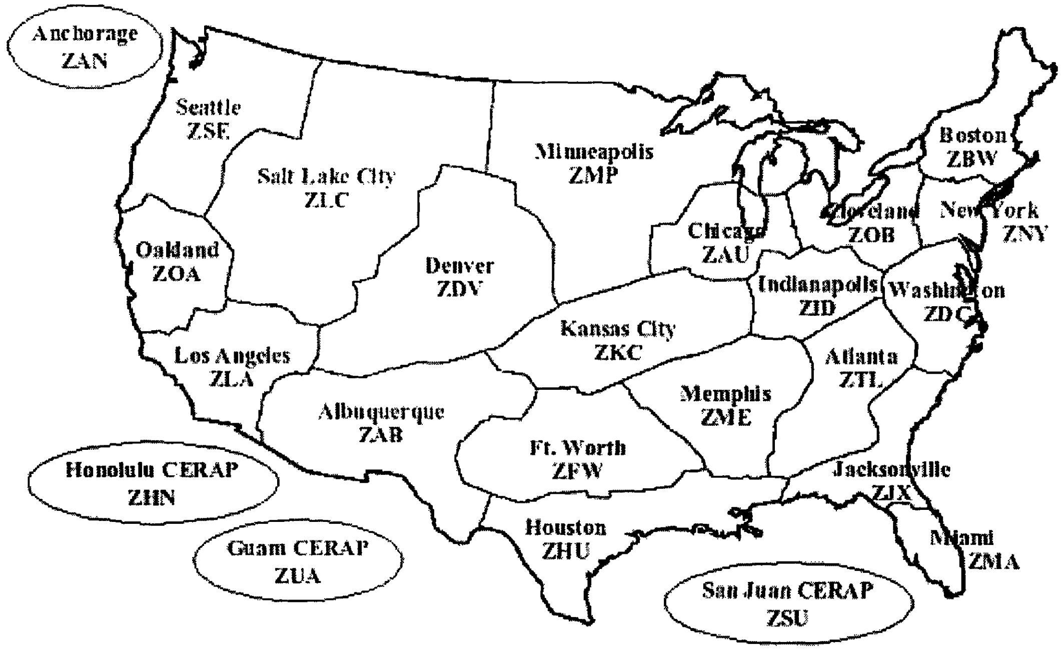


Figure 1-1: ARTCC regions

One increasingly important air traffic management issue that MEANS could help study is the effects of heavy en route congestion as more and more aircraft try to fly through the same airspace at the same time. While en route congestion has not severely impacted air traffic management up to now, the increasing flight load on the NAS is quickly approaching the capacity limits of the current control systems. With a new, detailed model of the NAS, MEANS could help to better understand the effects of airspace congestion and to devise air traffic flow management and control strategies. However, the model must be simple enough to rapidly test a wide array of possible new control strategies. MEANS could prove to be a valuable tool in the effort to redesign the NAS to handle a greater flight load.

1.1 Structure of the NAS

In the United States, air traffic control is provided by a collection of radar centers of various types. At airports, tower radar systems and Terminal Radar Approach CONtrol (TRACON) provide radar support to departing and arriving flights. Airport control towers typically determine which flights may take off and land as well as handling aircraft movements on the ground. TRACONs handle flights on approach and departing as they transition between the airport and cruising altitudes. TRACONs also set holding patterns for aircraft waiting to land. [4] In MEANS, the tower module performs the duties of the TRACONs and the airport control towers, although the tower module does not handle aircraft movement on the ground (see section 1.2). The Air Route Traffic Control Centers (ARTCC) manage en route flights, ensuring safe aircraft passage through their regions of the national airspace. Approximately 25 ARTCCs provide complete en route coverage for the United States, as shown in figure 1-1 [7]. The airspace/en route module in MEANS simulates the national airspace as a whole, including all of the ARTCCs.

1.1.1 The Role of Sectors in the NAS

Each ARTCC is divided into a number of control sectors. Each sector is typically handled by a single air traffic controller. A sector can only have as many planes in it as the controller can simultaneously handle, as additional aircraft in the sector increase the likelihood of a air safety violation. The flight capacity of a sector is tracked as a Monitor Alert Parameter (MAP). The MAP value for a sector is determined as a function of the average crossing time for flights passing through the sector. Possible values of the MAP value range from five for average crossing times of three minutes or less to eighteen for average crossing times of twelve minutes or more. MAP values can be raised or lowered by up to three at the discretion of the on-duty controllers. When the number of flights in a sector approaches the MAP value, the sector is flagged with a yellow alert, and then a red alert if conditions worsen. Under red alert conditions, flights may be prohibited from entering the sector to avoid unsafe airspace congestion. The maximum capacity of a sector may be temporarily reduced in response to bad weather or equipment malfunctions, as a controller needs to handle each flight more carefully. [8]

When the number of planes in a sector approaches the maximum capacity of that sector, air traffic controllers have several available options to reduce congestion. Controllers can reroute flights to less crowded neighboring sectors if congestion is localized. In many situations, controllers enforce a miles-in-trail restriction, forcing flights traveling along the same flight paths to slow down and follow the flight in front of them at a greater distance than normal. While this strategy prevents crowding from reaching dangerous levels, it can cause congestion delays to propagate as the miles-in-trail restriction affects sectors feeding the overcrowded sector. [7]

1.1.2 The Role of Flight Plans in the NAS

When flying at cruising altitude, flights do not typically fly the most direct path to their destination. Instead, they must follow a flight plan. Flight plans consist of series of Air Traffic Service (ATS) routes connected with waypoints. An ATS route is a "route designed for the management of air traffic operations or for the provision of air traffic services." [9] Routes consist of a series of straight segments connected at navigational markers. Sector geometries match the ATS routes in many places, allowing some sectors to almost exclusively handle traffic along a certain route. Normally, a flight plan consists of an alternating sequence of ATS routes and waypoints, with the waypoints indicating the points at which the plan switches routes. Waypoint-waypoint and route-route transitions do occur. If two waypoints are close together, no specific connecting route is necessary. If two routes only intersect at one point, specifying a crossover waypoint would be redundant.

The route and waypoint system of flight planning was developed prior to widespread global position systems and can be implemented using beacons and aeronautical landmarks. When all aircraft come equipped with GPS gear, the FAA may convert the current flight planning system into a "free flight" system, allowing for arbitrary waypoints. If this change occurs, the arrangement of the sectors will require adjustment to better handle the new flight trajectories.

1.2 Structure of MEANS

The driver of the simulation is an event queue. Rather than simulating real time flight movements, the event queue allows MEANS to only consider a flight when an event

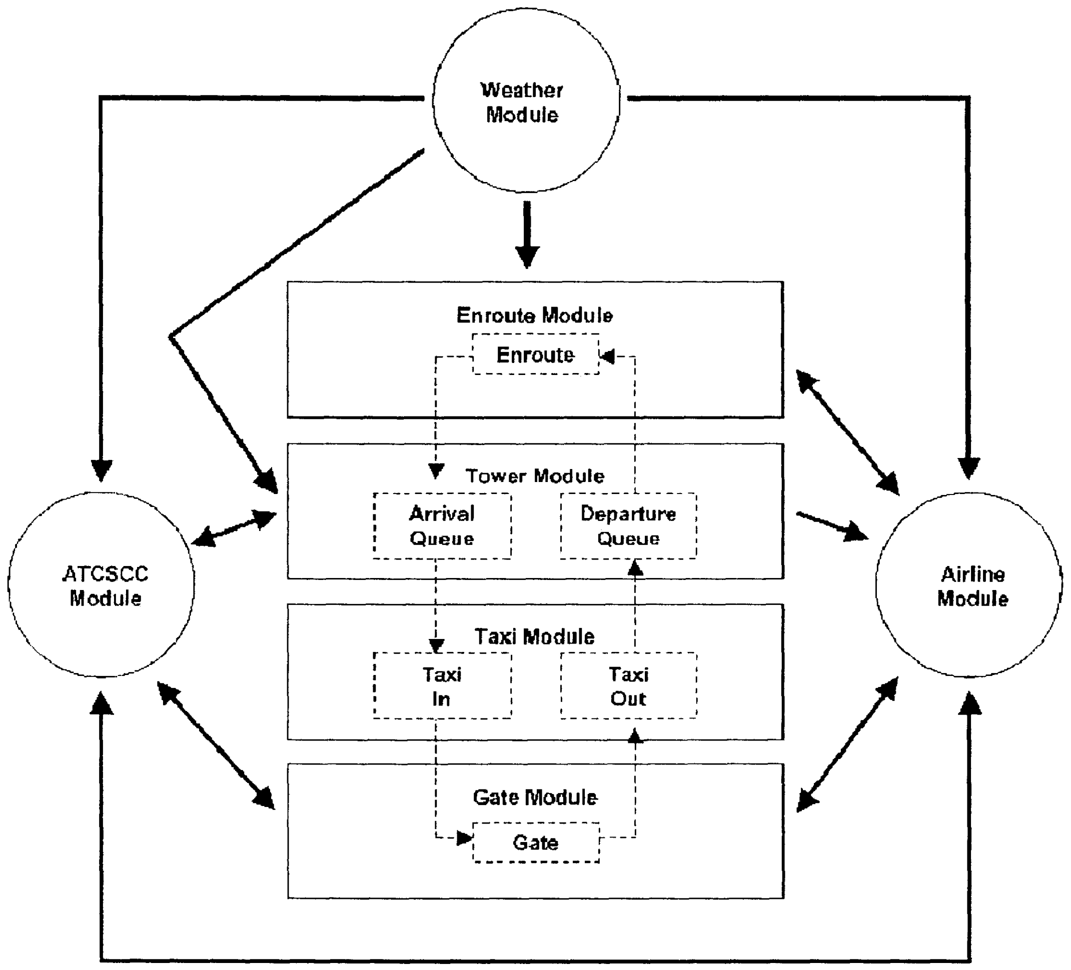


Figure 1-2: The relationship of component modules in MEANS

involving that flight reaches the front of the queue. These events include all major state changes, such as pushing back from a gate or arriving at a destination airport's arrival queue. The event queue is sorted according to simulated time of occurrence, so earlier events are processed first. The time of the event at the head of the queue is treated as the current time for the simulation. Time is not allowed to go backwards, so all events added to the queue must have associated times later than the current time. Initially, the event queue is populated with the flight schedule to be simulated. Processing these events generates additional events which propagate the simulation.

The MEANS module structure is shown in figure 1-2. Each module simulates a different part of the air transportation network. The modules at the center of the diagram represent simulated phases of a commercial flight. Flights start in the gate module. Once ready to depart, the flight moves from the gate module to the taxi module, to simulate the plane taxiing to its runway. Next, the tower module determines when the flight can take off, considering local weather and airport utilization levels. Once the flight is airborne, the en route, or airspace, module takes control and determines how long the flight takes to fly to its destination. When the flight arrives at its destination, the en route module passes the flight to the arrival queue of the destination control tower module. The tower module decides when this plane can land, again taking local weather conditions and current airport congestion into account. Next, the taxi module moves the aircraft from the runway to its assigned gate. Once back at the gate, the gate module determines how long before the plane is ready to depart on its next leg, including refueling and maintenance time.

Three of the modules simulate peripheral parts of the flight process. The weather module provides weather and weather prediction information to the other modules. This data allows for variable simulated flight rules, ground delay programs, and alternate runway utilization. The airline module responds to disruptions in the simulated flight schedule. For instance, if bad weather forces a ground delay program, airlines may cancel some flights to reduce the delays experienced by other flights. The Air Traffic Control System Command Center (ATCSCC) module is responsible for setting ground delay programs in response to inclement airport weather. If flights are expected to be unable to land at an airport because of weather-induced capacity constraints, the ATCSCC prevents the aircraft destined for the affected airport from taking off. The delay program holds the aircraft on the ground until they can be assigned a landing slot at their destination. This system is designed to prevent airborne aircraft from cluttering the airspace around congested airports and wasting fuel.

A summary of the major modules and implementations follows.

1.2.1 Gate Module

The gate module is responsible for determining how long a flight stays at the airport gate before it can push back and taxi to its runway. The exact time is determined principally by the aircraft's weight class. Mechanical failures and related delays are simulated stochastically. Additionally, the gate module responds to requests for ground delay program from the ATCSCC. If the ATCSCC anticipates that the destination airport of a flight will be congested, it will hold the aircraft on the ground. Several different ground delay program implementations are available in MEANS. The simplest never requires a ground delay. More complete implementations declare ground delay programs automatically in response to airport conditions or as specified by an input file. [4]

1.2.2 Tower Module

The tower module is responsible for determining which aircraft are permitted to takeoff and land. Flights that have taxied out to the runway wait in a departure queue until the tower gives them permission to leave. Similarly, flights that have completed their en route travel wait in the tower's arrival queue until the tower allows them to land. The rates at which the tower allows aircraft to takeoff and land are generally limited, though a trivial implementation of the tower module does provide unlimited tower capacity. One implementation specifies exact arrival and departure rates for each airport based on an input file. More advanced implementations determine the arrival and departure rates from a pareto frontier. An experimental implementation of the tower dynamically reorders arriving and departing flights to minimize the inter-flight delays.

1.2.3 En Route/Airspace Module

The en route module, also called the airspace module, determines how long the flight takes from the time it takes off until it reaches its destination airport. This time does not include circling the airport or landing; it only includes the time until the flight reaches the arrival queue for the tower at the destination. The most basic implementation of the airspace module calculates the en route time as the scheduled flight duration minus thirty minutes for airport circling and landing. A more refined implementation uses a historical distribution of flight times between airport pairs to estimate the flight time. For airport pairs where no historical data is available, the en route time is function of the scheduled flight time and the distance between the two airports.

1.3 Additional Desired Properties for MEANS

Many of the modules in MEANS provide implementations that closely model real world phenomena. These implementations attempt to capture not just a general statistical distribution but the actual physical process of the real world behavior. By comparison, the airspace implementations is quite limited. In fact, MEANS is incapable of simulating certain scenarios because the airspace module cannot provide the required level of detail. Specifically, all current airspace module implementations do not consider air traffic congestion when determining flight times. Without any congestion effects, MEANS cannot reasonably model scenarios involving airspace sector capacity reductions due to inclement weather. This also means that MEANS could not currently be used in efforts to redesign the NAS to minimize the impact of an increased en route flight load. Also, the current implementations do not provide any mechanism for modeling high altitude winds, which can dramatically impact flight times. Furthermore, several of MEANS's partners from the airline industry have expressed interest in extending MEANS to simulate the European airspace. Without additional fidelity, MEANS cannot be expanded to support the European airspace system, which has far more complicated flight rules than the United States.

The goal of this thesis is to design and implement an improved version of the airspace module. The new airspace model must at least account for congestion effects, individual sector capacity changes, and high altitude winds. With these additional features, MEANS will be capable of simulating the European airspace. Several existing simulations, such as NASA's FACET [6] already provide such detail by precisely calculating all aircraft trajectories as flights respond to airspace conditions while flying to their destinations. However, the new model cannot attempt to precisely calculate individual flight trajectories. Doing so would lengthen the execution times of MEANS unacceptably. Instead, the new model must strike a balance between precision and simplicity, simulating the required effects with as little computing effort as possible. A successful implementation should attempt to model all of the required features, not severely impact execution time, and improve on the per-flight accuracy of the en route travel times from the previous implementations.

Chapter 2

Design of the Sector-Based Airspace Module

2.1 Design Overview

The new model simulates the en route airspace as a collection of interconnected sectors. Each sector has a single set of local conditions and a set maximum capacity. All winds and capacity restraints apply to the sector as a whole. Flights traverse these sectors as they follow their provided flight plans. These flight plans consist of a series of sector crossings, or a set heading and distance required to cross the sector. This sector crossing information is translated into a sector crossing time based on a formula including the aircraft's airspeed, the required crossing distance and heading, and the current sector wind speed and heading. After a flight completes its crossing of a particular sector, that sector hands it off to the next sector. When the flight reaches the end of its flight plan, the final sector hands the flight off to the destination airport's arrival queue.

This model has been chosen because it mimics all the components of the real world airspace required to meet our simulation goals without introducing unnecessary complexity. Most of the desired simulation detail is derived from the sector conditions as it applies to individual flights. The behavior of the sectors is designed to match the behavior of real sectors as closely as possible with regard to their interactions with passing flights. By simulating sectors as discrete units rather than continuous spaces, the new system captures the sector-level effects without incurring the cost of modeling complex internal sector behaviors.

2.1.1 The Legacy Airspace Specification

Before attempting to outline the design for the new airspace model and implementation, the exact programmatic specification of the airspace module in MEANS must be made clear. All airspace implementations inherit two functions from the abstract base class AirspaceBase that defines the airspace interface. The functions are predict() and accept(). The function predict() takes as parameters a flight and a takeoff time and returns a nonbinding estimate of the time that the flight will reach its destination airport. The function accept() is called by the tower module to move a flight from the departure queue to the airspace, more or less a "take off" function. The accept() function takes the same parameters as predict(), a flight and a time of takeoff, which is presumably the current time in the simulation. The accept() fucntion does not return any value to the tower. After the tower calls accept(), it is the airspace's responsibility to set up any necessary events to cause the flight to reach the arrival queue of its destination airport after an appropriate flight time. The current implementations of accept() create an event at the expected arrival time for the flight to trigger the appropriate handoff function.

One important feature of the accept() function is that it does not provide any mechanism to reject incoming flights. Even if the flight is malformed, possibly with an invalid destination airport, the airspace cannot legally notify the caller of accept(). The only possible action for the airspace in such a situation is to crash MEANS, hopefully with a useful error message to the operator. Dropping the offending flight violates the expectations of the module and can lead to undefined behavior elsewhere in the program.

2.1.2 Required Changes to the Specification

In order to properly simulate a capacity constrained airspace, the airspace must have the option of preventing flights from taking off. If the sector into which the flight takes off is overcrowded, the flight must not take off, as doing so would force the sector to exceed its maximum capacity. Unfortunately, the current airspace module specification does not provide any mechanism allowing the airspace to reject flights or notify towers that a flight will be rejected. Without this capacity, the simulation goals are not satisfiable. In order to achieve the stated goals, the airspace specification must be changed. The selected changes should minimize the impact of the modification on the rest of the system to reduce the volume of code refactoring needed to implement the changes.

The specification was modified in two ways to satisfy these requirements. First, a new method was added to airspace interface, check(). The check() method takes the same arguments as predict() and accept(), a flight and a takeoff time. The function check() returns the number of seconds the flight must wait after its desired takeoff time before the airspace will permit it to take off. If the airspace would permit the flight to take off right away, check() will return zero. The check() return value is generally a nonbinding estimate. However, check()'s return value is binding in one special case. If check() returns zero when the provided takeoff time is the current simulation time, then the airspace guarantees that the it will accept the flight on a call to accept() without any additional delay. The second change modifies the calling requirements of the accept() method. The old specification permitted any call to accept(). The new specification requires that for a given call of accept(), a matching call to check() with the same parameters returns zero at the time of the call to accept(). This now allows the airspace module to prevent flights from taking off by returning nonzero values for check().

Adding the new specification for the existing implementations is not difficult. All old, unconstrained airspace implementations may safely return zero for all calls to check(), as these implementations never have any reason to reject a flight. All tower module implementations need modifications to force them to call check() before they attempt to pass flights to the airspace. If the call to check() is nonzero, the tower skips that flight and allows other flights to take off instead.

Unfortunately, the most advanced implementations of the tower module, those with dynamic reordering of flights waiting to take off, are not compatible with the new specification. Imposing additional delays on the flights in the takeoff queue interferes with the dynamic reordering logic. While the reordering logic can be modified to consider airspace imposed takeoff delays, doing so is beyond the scope of this thesis. Consequently, the sector-based airspace implementation needs to include error checking code to detect if it is operating alongside a noncompliant tower implementation, so that MEANS can gracefully abort with an error message. The older airspace implementations are still compatible with these tower implementations despite the interface change, since they never impose any takeoff restrictions.

2.2 The Sector-Based Airspace

At the core of the new implementation is the class inheriting the airspace module base class, called AirspaceDetailed. The predict() function determines the arrival time at the destination landing queue by querying each sector along the flight's plan and summing up the expected crossing time for each sector. The check() function determines the airspace induced delay by summing up the delay imposed by each sector along the flight path. This value is intended to underestimate the actual delay, since other flights may induce additional delays between the call to check() and the flight's takeoff time. The accept() method first enters the new flight in the module's flight tracking data structures, and then passes the flight to the first sector in its flight plan.

Besides being responsible for providing the external interface for the module, AirspaceDetailed coordinates the various sectors in the simulation with each other and all en route flights. AirspaceDetailed keeps pointers to all of the individual sector objects (see section 3.2 for more information on sectors,) in a collection associated with the name of the sector for easy lookup. Additionally, AirspaceDetailed maintains a catalog of all flight plans used in the current flight schedule. The current design allows only one flight plan per origin-destination city pair, but this could be modified to permit multiple possible flight plans in the future. As flights follow their flight plans, AirspaceDetailed tracks their position in the airspace by associating each flight with a marker indicating which flight plan segment the flight is in. When a flight moves to a new sector, takes off, or lands, AirspaceDetailed updates these markers to accurately reflect the progress of the flight.

From a flight's perspective, the tower first checks with the airspace to make sure that the flight is allowed to take off. If it is, the tower invokes AirspaceDetailed's accept() method, causing the airspace module to initialize the appropriate tracking markers and insert the flight into the first sector on its flight plan. When the sector receives the flight, it schedules a callback in the event queue to trigger the transition of the flight to the next sector. At the time of this callback, the sector checks to see if the next sector on the flight's plan will accept it. If the next sector is open, then the flight makes the transition, all necessary AirspaceDetailed state is updated, and the process repeats in the next sector. If the next sector does not accept the flight, then the callback is rescheduled for a later time. The sector reattempts the transition at that time. The last sector on the flight plan transitions the flight into the arrival queue of the destination airport instead of a subsequent sector. Once the flight has arrived at its destination, AirspaceDetailed clears all remaining state referring to the completed flight, both to free up memory and to ensure that any subsequent legs of the flight are handled correctly.

2.2.1 Sectors

Besides handling flights that are in the process of crossing the sector, a sector must maintain additional state to ensure that the airspace as a whole runs smoothly. Sectors must record expected flight crossings, scheduled by flights at takeoff. This information is required for the sector to accurately predict how much delay a future flight will encounter when it attempts to enter the sector. The sum of these delays is the value returned by AirspaceDetailed's check() function, which is the first guard against airspace overcrowding. Since flights may be delayed en route, all data regarding expected flight crossings may change, and the structures in which the data is kept must be capable of dynamically modifying the information. Additionally, sectors must track their simulated conditions, including maximum capacity and flight level winds, both of which are used in determining sector crossing times for passing flights.

2.2.2 Flight Plans

Associated with each origin-destination city pair is a flight plan. Each flight plan consists of a series of segments, each segment listing a sector to cross along with the heading and distance of the crossing. With this information, combined with knowledge of aircraft cruising speeds and flight level winds, the sectors can approximate the unconstrained crossing time of the flight (see section 3.1 for more information.) AirspaceDetailed maintains a listing of all flight plans. Separately, AirspaceDetailed maintains pointers for each active flight into the flight plan structure indicating that flight's progress. A flight's pointer should point to a segment relating to the sector it is currently in.

Currently, each city pair has only one flight plan associated with it, and every flight making that trip uses that plan. Additionally, the flight plan system does not allow for rerouting en route flights. Without keeping real time positions of every flight, it is not possible to properly calculate a revised flight plan with correct sector crossing distances for arbitrary rerouting. Since such a large amount of resources would be required to simulate this relatively rare feature, explicit rerouting is left out of the new airspace module's design. The effects of rerouting should be captured by other features of the system, principally by time variable sector capacities along with capacity induced delays.

2.3 Sector Design

While the AirspaceDetailed class is responsible for most of the coordination between sectors and for implementing the airspace interface, the individual sectors handle most of the computational work. The sectors are principally tasked with accepting passing flights and ensuring their correct delivery to the next sector on their flight plans or to their destination airport. In order to fulfill that role, sectors must carry out a number of processes. First, sectors must be capable of handling incoming flights and signaling whether or not they accepted presented flights. Next, the sectors must calculate the crossing time for traversing flights. Using these crossing time, the sectors must then schedule the flights' exits by inserting the appropriate callbacks into the event queue. When the callbacks are triggered, the sectors must then hand the flights off to either the next sectors on their flight plans or their destination airport. If the hand off to another sector fails, then the sector must reschedule the callback for a time when the handoff is expected to succeed. In order to facilitate the process, the receiving sector should provide an indication as to when it will accept a previously rejected flight.

In addition to moving flights across the airspace, sectors play an important role in coordinating flights throughout the system. In particular, sectors must provide information to AirspaceDetailed regarding potential crossing times and delays to entry in response to calls to predict() and check(). In order for the produced information to be meaningful, the sectors must maintain data on future predicted flight crossings. A schedule entry for each crossing flight should be created when that flight takes off, modified if the flight encounters delays, and cleared when the flight has completed its crossing. Consequently, sectors must be capable of performing all calculations conjecturally, using sector conditions and flight occupancy statistics that will not be in effect until later, if ever.

2.3.1 Capacity and Scheduling

The primary data structure in the sector tracks the current and scheduled occupancy of the sector. This data structure must be capable of supporting all of the roles the sectors need to fill, as described above. Specifically, the data structure must properly account for all flights that are currently in the sector. Without this, the sector would have no way of knowing whether or not it had reached its maximum capacity. The callbacks that handle the actual mechanics of the flight crossing do not present the sectors with enough information to track sector occupancy. To provide this information, the data structure needs to handle adding entering flights and removing exiting flights. If a flight's exit is delayed, the sector's data structure must not prematurely remove the flight. Furthermore, the data structure should track the expected exit times for all flights in the sector so that it can produce accurate occupancy reports for future times.

Additionally, the occupancy data structure must catalog scheduled flights that have not yet reached the sector. At takeoff, each flight schedules crossings for each sector for the appropriate time interval. The structure must be capable of processing this information to give correct estimates of future sector occupancy. Furthermore, the structure may be required to reschedule scheduled crossings if flights encounter unexpected delays prior to reaching the sector. Combined with the records of the flights currently in the sector, the data structure should provide a reasonable, accurate prediction of sector occupancy for the near future. Flights that have not yet taken off are not included in the structure's listing. Consequently, predicted occupancies will not be meaningful for times more than a flight's duration in the future. Fortunately, this sector occupancy information is used for flight planning and should not need occupancy estimates any farther in the future than that. In particular, the information is used to calculating delays for flights waiting to takeoff. Prior to takeoff, the tower must perform a check() call on the flight to ensure that it will not encounter any delays en route to its destination. In order to do this, each sector must know its expected occupancy for the estimated crossing time so that it can report whether or not it can handle the flight. As specified, the data structure should be capable of providing this information.

In general, the occupancy data structure will support two basic operations, updates to the schedule information and queries about occupancy at a given time. In order to support efficient updates, the structure should not require extensive examination or correction of stored records. Ideally, only the records being changed would be modified, and these records could be looked up efficiently. Since each record would catalog either a flight entering or leaving the sector, each record could contain a field for the change in occupancy of the sector at that time. To change the state of the structure, no records other than those directly modified would need correction. However, calculating the scheduled occupancy of the sector at a given time would require a complete traversal of the data structure to sum up all of the individual changes up to that time. On the other hand, to support efficient occupancy queries, the structure should provide lookups of the occupancy itself. This could be implemented by associating each record with the total number of aircraft in or predicted to be in the sector at the time of the record. Of course, any modifications to the structure would require a complete structure traversal to correct all totals to reflect any changes. Since both of these operations are common in sector operations, neither design option is clearly favored. The selected implementation of the structure merges the two possibilities to provide reasonable performance for all standard operations.

2.3.2 Sector Transitions

Once a flight's callback is triggered, the sector must first determine where the flight is headed next. AirspaceDetailed tracks this information and can easily provide it upon request. If there is not a following sector, then the flight has arrived at its destination. The sector transfers the flight to its destination's arrival queue and notifies AirspaceDetailed that the flight has left the airspace. If there is another sector on the flight's plan, then the sector attempts to pass the flight to the next sector. If the next sector accepts the flight, the new sector notifies AirspaceDetailed that the flight has advanced a segment on its flight plan and integrates the flight into its internal structures. All the previous sector has to do is to remove the passed flight from its data structures. If the next sector is full, it rejects the flight and indicates a delay interval after which it should accept the flight. This interval is calculated by examining the occupancy data structure and finding the next time that the total occupancy of the sector drops below the maximum capacity. The sector holding the flight must reschedule the transition for the flight, so that the transition will eventually succeed, and update any internal state referring to the flight's exit time. Additionally, the sector should reschedule the crossings for every following sector, so that the scheduled crossings reflect the flight's deviation from its original crossing schedule.

2.3.3 Variations of Flight Acceptance Rules

In most cases, the rules determining whether or not a sector accepts a presented flight are quite simple. The sector accepts flights so long as the total number of flights in the sector does not reach the sector's maximum capacity. Two special cases are required for reasonable operation of the new airspace. First, flights taking off must always be accepted by the first sector on their flight plan. The airspace interface does not permit a flight to be rejected from the airspace entirely, so the first sector must accept the flight even if it causes that sector to exceed its capacity. Tower modules must still call AirspaceDetailed's check() method before attempting to force aircraft into the airspace, so this rule change should not cause sectors to exceed their capacities in practice. This change is only required to guarantee that flights do not get lost during the handoff between the tower and the airspace modules.

The second special case involves sectors at their maximum capacity. Since sectors at their capacity reject incoming flights, deadlock could arise if full sectors require each other to accept flights. If each full sector requires another full sector to accept a flight before it will accept flights, then no progress is possible, and the simulation will not terminate. In order to prevent this problem in most cases, flights originating from full sectors are subject to a different set of acceptance rules. In particular, the receiving sector must accept the flight so long as it does not exceed the sector's capacity by more than one. This rule gives full sectors special access to an occupancy slot unavailable to unconstrained sectors. Since this extra slot is only used to clear potentially deadlocking situations, it does not routinely cause sectors to exceed their normal capacity limits.

The provision of a single extra slot prevents deadlock in all but very rare circumstances. In order for the new system to deadlock, a collection of interdependent sectors must all exceed their capacity by one. This would nullify the specially provided extra capacity slot. Prior to such a situation, all of the interdependent sectors except one would exceed their capacity by one. In this state, the number of available extra slots across collection sectors is one. In order to transition from this prior state to the deadlocking state, the remaining slot must be occupied by a flight entering the collection. The deadlocking flight cannot come from another sector in the collection, as the originating sector would open up a slot in the process. Also, the deadlocking flight cannot come from an outside sector that has not reached its capacity, as its flights are subject to the original acceptance rules. The only possible origin of the deadlocking flight is a temporarily full sector outside of the interdependent collection. The offending sector could be full from a spike in otherwise unconstrained traffic and could force the deadlocking flight into the collection using the special acceptance rule. Since the original acceptance rules rarely resulted in deadlock, and the new rules only fail under specific and unlikely circumstances, this solution is expected to functionally eliminate deadlock from the system.

2.4 Argument for Miles-In-Trail Simulation

In order for the proposed airspace design to be useful, it must simulate real world air traffic effects. While the individual sectors handle unimpeded travel times appropriately, it is not clear that a purely capacity limited sector structure can adequately simulate miles-in-trail restrictions. By applying Little's Law from processor architecture theory [11] to the aircraft in a sector, it can be shown that a capacity limited system reasonably approximates a miles-in-trail limitation. Little's Law states that the average number of instructions in a processor is the product of the throughput of the processor, in instructions per second, and the latency of each instruction. Rearranging terms, the law also states that the throughput of a processor is the average number of instructions "in flight" divided by the latency of each instruction. Increasing the number of simultaneous instructions or decreasing the latency increases the throughput of the processor. Applying this to flights in a sector, the throughput of a sector, in flights per unit time, is the average occupancy of the sector divided by the sector crossing time of a flight. While these flights do not necessarily follow the exact same path, all existing flight paths can be collapsed into a single path for the purpose of this argument. One mapping to the single path case is to assign sector occupancy slots to the different flight paths in a round robin fashion, thereby interleaving the different paths into one.

Little's Law as applied to sector throughput and the single path assumption can be used to demonstrate that a miles-in-trail restriction is identical to a capacity constrained system in terms of sector throughput. First, consider a miles-in-trail restricted sector. The number of flights the sector can handle simultaneously is the sector crossing distance divided by the miles-in-trail distance, since the sector only allows one plane every miles-in-trail miles. The latency is the average crossing time for traversing flights. Notice that the sector throughput decreases as the miles-in-trail distance increases, as the sector can hold fewer flights at the same time. Now, consider a purely capacity constrained sector. The number of flight the constrained sector can handle simultaneously is simply its maximum capacity. The latency is again the average crossing time. By decreasing the maximum capacity of the sector, the maximum throughput of the sector is reduced just the same as if the miles-in-trail distance had been increased. From a sector throughput point of view, there is no difference between increasing a miles-in-trail restriction and decreasing a maximum capacity. While the inter-flight times are not guaranteed to be the same, the ability to simulate the throughput effect a miles-in-trail restriction for a sector is sufficient for MEANS.

Since miles-in-trail restrictions propagate in the real world as congestion spreads, the capacity constrained system should also propagate any throughput reductions that would normally arise. This is indeed the case. When a sector reaches its maximum capacity, additional flights into the constrained sector must wait until a slot opens up. By waiting, these flights experience a greater latency in sectors neighboring the constrained sector than if they had been allowed to travel without the transition delay. From the above argument, increasing the latency of the flights reduces the maximum throughput of the adjoining sector. This reduction in throughput correctly simulates the reduction in throughput from the propagation of a mile-in-trail restriction into adjacent sectors. Furthermore, if the congestion does not clear, the sectors neighboring the original overcrowded sector reach their maximum capacities. In this case, the increased latencies and corresponding decrease in throughput spread to additional sectors, simulating the continuing propagation of a miles-in-trail restriction.

Chapter 3

Implementation Details and Flight Plan Generation

3.1 Airspace Implementation

The implementation of the airspace interface, AirspaceDetailed, is responsible for coordinating between sectors, tracking the progress of each flight, and providing the airspace interface. In order to maintain precise control over the sectors, AirspaceDetailed creates all of the sectors in the system and stores references to them in a map associating each sector's name with its pointer. This structure holds the primary copies of the pointers to the sectors. Other copies of the pointers can exist, but they cannot be used to destroy the sector objects. Only when this data structure is deconstructed are the individual sectors deleted. While this structure retains the primary copies of each sector, it is only used to help create the flight plans in the system. When the flight plan configuration file is loaded into AirspaceDetailed at startup, the file refers to each sector by name. The sector name map converts the name in the file to a copy of the object reference for the named sector. The sectors are not actually named in the flight plans that AirspaceDetailed stores, (refer to section 2.2.2 for more information on flight plans.) All sector access in the simulation after startup is done through the copies of sector references in the flight plans. By performing the string name to reference conversion at startup, the bulk of the simulation can avoid a potentially expensive map lookup for each sector access.

The flight plans are stored as vectors, with each flight plan vector identified in a map by its origin and destination airports. As mentioned before, there is only one flight plan per origin-destination pair. The data structure could be modified in the future to support multiple flight plans by having each origin-destination pair map to a vector of flight plan vectors. The flight plan associated with each flight is simply the flight plan listed in the map with the appropriate origin and destination. For flights in the air, AirspaceDetailed associates the aircraft with an index into the flight plan vector to track the progress of the flight. Flights that have not taken off and flights that have already landed do not have associated indices. While the index is not publicly visible, sectors can retrieve the current flight plan segment and the next flight plans segment, which is null if the aircraft is on its last segment. Additionally, sectors can request that AirspaceDetailed advance an aircraft to its next segment in response to a sector transition. If the aircraft was previously on the ground, AirspaceDetailed creates an entry for the newly airborne flight and marks that the flight is in its first flight plan segment. If the aircraft was in the last segment of its flight plan, AirspaceDetailed clears the entry for the flight to indicate that the flight is now on the ground.

In addition to tracking the current location of each flight, the position information associated with each flight is needed to properly reschedule sector crossings for flights that have been delayed en route. When a flight is delayed, AircraftDetailed must notify each sector after the flight's current segment to reschedule the expected crossing. Initial scheduling does not require the position information, as it always contacts each sector except the first sector in the flight plan.

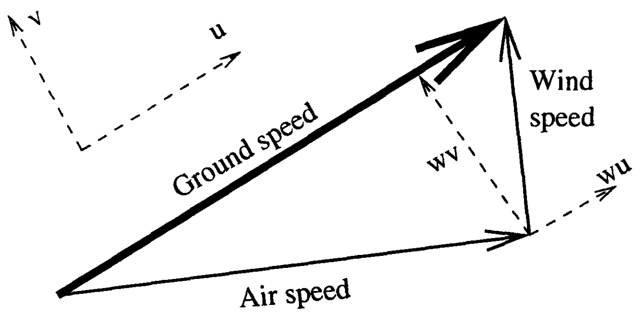
To support the new airspace interface, AirspaceDetailed must provide a predict() method, a check() method, and an accept() method. The function predict() calculates the expected arrival time for a flight at its destination airport given a flight and its takeoff time. AirspaceDetailed calculates this value by querying the sectors associated with each segment on the flight's plan in order. Each sector responds with the expected total crossing time for the sector, the sum any entrance delay from overcrowding and the expected flight time for the crossing. The sum of these total crossing times is the total expected duration of the flight, so the expected arrival time is the sum of the crossing times and the takeoff time. Intermediate sums are used to determine the time the flight arrives at each sector on its flight path, so that the sectors can determine the appropriate conditions when calculating the crossing times. The function check() returns the total delay expected for a given flight taking off at the provided time. AirspaceDetailed calculates this value similarly to the calculations in predict(). Intermediate flight time sums are again used to determine the entry time for each of the sectors along the flight plan. However, instead of returning the arrival time, check() returns the sum of the delays. The calculation of the total crossing time for each sector returns the entry delay time in addition to the total crossing time. If the value returned by check() is zero, then no delays are expected for the flight as it follows its flight plan.

Calling accept() triggers AirspaceDetailed to enter the requested flight into the airspace. AirspaceDetailed first calls check() to ensure that the flight is permitted to take off. Next, AirspaceDetailed forces the first sector on the flight's flight plan to accept the flight. Because the airspace interface does not allow the sector to reject the flight, capacity limits for the first sector are ignored. The prior call to check() guarantees that any capacity limits are enforced. In the process of entering the flight into the first sector, the sector calls into AirspaceDetailed to initialize all appropriate tracking state for the flight. Finally, AirspaceDetailed schedules sector crossings for each subsequent sector on the flight's path. The initial sector already contains the flight, so scheduling the flight is unnecessary. When each sector schedules the flight, it returns the expected exit time for the crossing, which is then used for the entrance time for the following sector.

3.2 Sector Implementation

To support AirspaceDetailed, sectors must be able to appropriately enter and exit each flight passing through, allow for flights to be scheduled and rescheduled, and provide estimated crossing times for anticipated sector crossings. Entering flights can presented to the sector in three ways. First, flights taking off must be accepted. Second, flights entering from an overcrowded sector are subject to lenient entrance rules, allowed to exceed the sector's capacity by one. Third, flights entering from unconstrained sectors are subject to normal entrance rules and cannot cause the sector to exceed the sector's maximum capacity. Once a flight has been accepted, the sector treats flights the same in all three cases. First, a call is made to AirspaceDetailed to increment the flight's location index, then sector checks that it is the current sector according to AirspaceDetailed, ensuring the flight is in the correct sector. Next, the sector calculates the flight time for the crossing according to its current wind conditions, the heading and distance of the crossing, and the aircraft's speed. Since a detailed matching of aircraft tail numbers to cruising speeds was not available, all aircraft speeds are approximated as 450 knots, which is a reasonable approximation of the cruising speed for most airliners [12]. Future versions can provide better airspeed estimates for better precision. The exact calculation, summarized in figure 3-1, assumes the aircraft is tracking towards its destination, meaning that the aircraft angles itself to maintain the appropriate ground heading despite any wind. The effective ground speed is $w_u + \sqrt{|n|^2 - w_v^2}$. w_u is the portion of wind velocity in the direction the aircraft is traveling. The square root is the portion of the aircraft speed in the traveling direction, derived via the Pythagorean Theorem from the aircraft's airspeed, $|n|$, and the portion of the wind velocity orthogonal to the flight direction, labeled w_v. w_u and w_v are calculated by first subtracting out the flight heading from the wind heading before deriving the vector components of the wind. Finally, all necessary sector state is updated to reflect the new flight and its anticipated exit time.

A callback is entered into MEANS's event queue for the calculated exit time. When the callback is triggered, the sector attempts to transition the flight into the next segment of its flight plan. If AirspaceDetailed reports that the aircraft has arrived at its destination, then the sector hands the flight off to the destination arrival queue, notifies AirspaceDetailed that the flight has landed, and logs the transition in the



Crossing time = Crossing distance / |Ground speed|

Figure 3-1: Calculation of sector crossing time

internal state. If the aircraft is headed into another sector, then if the next sector accepts the flight, then the current sector only logs the flight transition internally, as all additional computational work is handled by the next sector. If the next sector rejects the flight, the next sector gives an estimate for how long the flight will have to wait before being allowed entry. The current sector reschedules the exit of the flight for after the delay, notifies AirspaceDetailed to reschedule all subsequent sector crossings to reflect the added delay, and enters another callback into the event queue for the new exit time.

Sectors handle scheduling and rescheduling crossings in much the same way. The only difference is that when rescheduling a sector crossing, the sector first clears the existing schedule entry. In either case, the expected entry time is provided to the scheduling or rescheduling function, which then calculates the flight time required to cross the sector. The expected exit time is simply the anticipated entry time plus the required transit time. Any delays the flight encounters if it is denied immediate entrance into the next sector are not incorporated into the expected exit time.

To provide estimated crossing times to AirspaceDetailed, the sectors must sum the flight time and any entrance delay due to sector capacity constraints. The flight time calculations are described above. The entry delay is determined by examining the sector's schedule from the requested entry time forward to find the earliest time at which the sector would accept an additional flight. The entry delay itself is used by AirspaceDetailed in the check() calculations and is optionally returned by reference to the caller along with the total estimated crossing time.

3.2.1 Sector Data Structures

Sectors have three data structures to hold all the state required to perform these operations. The first structure holds a listing of all the flights currently in the sector. The second holds a listing of all the flights that are scheduled to be in the sector. The third structure maintains a schedule for the occupancy of the sector. In all three structures, flights are identified by the airline and tail number of the aircraft, and by the segment index of the sector crossing in the flight's plan. The segment index allows for flights to pass through the same sector more than once. Without a segment index reference, it would not be possible to differentiate the separate crossings.

The schedule structure associates times with sector events, such as flight entrances or exits. Each event contains the identifier of the associated flight, the change in the occupancy of the sector in response to the event, and the total number of scheduled aircraft remaining in the sector after the event occurs. The data is stored in a multimap, since several flight events can occur at the same time. A multimap maps keys to values. In this case, the keys are the times and the values are the events. Multiple events may be associated with a single time. The multimap stores the entries in key sorted order, so the structure is incrementally searchable, meaning that given one entry, the next entry in time and the previous entry in time are easily obtained. This is important, since a lookup for a given time will only yield one of the events occurring at the listed time. The rest must be obtained through incremental search. Also, incremental search is required for schedule scanning operations. One example of this is the determination of the next time after a given time that the occupancy of sector is below its capacity, and a flight denied entrance may be allowed to enter. A multimap is not capable of efficient indexed lookup. In general, the entire structure must be analyzed to find the nth entry. Instead, the structure allows for efficient lookups of the events given a time of occurrence. The total number of aircraft in the sector is the sum of all of the changes in sector occupancy for all of the events up to and including the event holding the total. While the storage of the change in occupancy may seem redundant, storing the changes per event allows for more efficient correction of the schedule structure in response to schedule changes.

Additionally, a special scheduled flight event exists at the sector's notion of the current time that represents the entrance of all flights currently in the sector. This block has a special identifier and has its change in occupancy set to the number of aircraft currently in the sector, possibly zero. The use of this special block is preferred to individual markers for flight entrances. Not only does it reduce the number of entries in the system, the use of the special block also guarantees that the schedule always has at least one entry, which reduces the number of computational edge cases.

The structure containing a sector's current flights maps flights to exit times. The callback to exit the flight from the sector is scheduled for the time associated with the flight in this structure. If a flight fails to exit the sector at its scheduled time, then the exit time associated with the flight is updated to reflect the delay. The exit time refers to the flight's exit event as stored in the schedule structure, so any changes to exit time in one structure must be mirrored in the other. When a flight exits a sector, its entry in this data structure is removed. Also, the number of entries in this structure is the current occupancy of the sector, as used by the sector to decide whether or not to accept an additional flight.

The structure containing scheduled flights is very similar to the structure of current flights. It maps each scheduled flight to an anticipated entry time and an expected exit time. These times are not binding, but they do reference specific entries in the schedule structure. Consequently, if a flight is rescheduled, both this structure and the schedule must be corrected. While the times associated with the scheduled flight are tentative, the flight crossing itself is expected to occur. Flights in the list of scheduled flights must at some point enter the sector, as the schedule entry is only cleared when the flight enters the sector. In the event that en route flight rerouting is added in the future, this structure will have to be capable of removing rerouted flights.

3.2.2 Requirements and Invariants

Given the complexity of the sector's data structures and the level of optimization employed in their implementation, it is vital to first establish the logical invariants for the structures that guarantee correctness. The structure containing the set of current flights must always contain all flights in the sector and no more. In order for this requirement to be satisfied, this data structure must be empty at both the beginning and end of the simulation. At both these times, all sectors are guaranteed to be empty, since the simulation starts with no flights in transit and only terminates once all active flights have arrived at their destinations. This implies that the number of flights entering a sector of the course of the simulation must equal the number of flights exiting the sector.

The structure containing the scheduled flights must contain only flights that will at some future point enter the sector. Therefore, flights that are already in the sector and flights that will not pass through the sector should never appear in the structure of scheduled flights. Additionally, the correctness of the structure requires that only flights that have taken off appear in the scheduled flight structures. This is necessary because the identification of the flights does not guarantee correct differentiation between different legs of the same flight with the same aircraft. Also, it follows from the previous requirements that the scheduled flights structure must also be empty at the beginning and termination of the simulation.

Because of the independent but related fields in each schedule event, great care is needed to preserve the integrity of the schedule data structure. Most importantly, every listing of the total number of the aircraft in the sector after a schedule event must equal the sum of all of the changes in sector occupancy listed in the schedule up to and including that event. Also, each flight in the schedule has an entry event, either as a scheduled entry or as part of the special current flights event, and an exit event, since it must not remain in the sector indefinitely. Since each entry event has a corresponding exit event, the listed occupancy after the last event in the schedule must be zero. The special current flights event, marking the collected entry of all flights currently in the sector, must increase the occupancy by the number of flights in the sector as listed in the current flights data structure. Also, the current flights event must be listed in the schedule under the sector's notion of the current time of the simulation. If the sector's current time changes, then the special event must be moved. Additionally, besides the special current flights event, each scheduled event must relate to a flight in either the current flights or the scheduled flights structures. These structures contain the times at which events relating to particular flights occur. Without the times, the entire schedule would need to be searched to find events relating to a particular flight. Similarly, for each flight in each of these structures, the schedule may only contain events relating to the flight at times these structures associated with the flight and must contain an event at each such time. Finally, the schedule should never list the occupancy of the sector as less than zero. Since schedule events occurring at the same time may get reordered during schedule maintenance, it is sufficient to require that the listed occupancy never drop below zero for the last event in the structure occurring at any given time.

3.2.3 Schedule Operations and Correctness

While the operation and correctness of the current flights structure and the schedule flights structure are fairly straightforward, the schedule structures operation requires further elaboration. Each schedule operation is detailed and proved to maintain the required invariants of the schedule structure provided only that the invariants were maintained at the beginning of the operation. This, combined with a proof that the schedule initializes into a valid state is sufficient to conclude by induction that the schedule is correct for any reachable state.

One key insight into the schedule operations enumerated below is that each operation results in a zero net change in the schedule's final occupancy. For each flight added to the schedule, one is correspondingly taken away. If this were violated, then the occupancy of the sector listed in the final event in the schedule would not continue to be zero as required. Furthermore, after each of these zero net change operations, only events occurring between the first and last schedule modification of the operation require corrections to their total occupancy data. All events before the first schedule modification do not require correction, since no events before them have been modified, leaving the running sum unchanged. Also, events after the last schedule modification do not require correction, as the sum of the modifications to the sector's occupancy prior to the event is zero. This means that in order to correct the occupancy totals to match the updated events, only events between the earliest and latest change require correction. This optimization bounds the number of corrections require to the largest possible span of a modification, which is considerably less than the total number of events in the schedule. In order to use this optimization, each operation must be demonstrated to result in a zero net change. This is required anyway to preserve the total occupancy of zero for the end of the schedule. In each of the following cases, if the net change is shown to be zero, then the optimized correction algorithm runs on the modified portion of the schedule at the conclusion of the operation.

Accepting a Flight

First, the schedule may process the entry of an unscheduled flight into the sector. This would occur if the flight took off into the sector without first crossing any other sector. In response to this action, the flight is added to the structure of current flights with an associated exit time calculated appropriately. An event is entered into the schedule with a change of negative one at the exit time to represent the flight's predicted exit, as mirrored in the current flights structure. Finally, the special event containing the current flights increments its change by one to reflect the additional flight, and reinserts itself into the schedule at the time the flight entered the sector, and the sector's current time is updated to reflect the change.

The net change in scheduled occupancy for this operation is zero, since the special event had its listed change incremented and the new exit event decrements the total. Therefore, the correction algorithm described above may run correctly, fixing all inconsistencies in the scheduled occupancies, and the final scheduled occupancy of sector remains zero. Also, this operation will not cause the sector to have any periods of negative occupancy, since the entry must occur before the new exit event. In between, all scheduled occupancies will be incremented. Therefore, so long as the schedule was in a legal state prior to the event, processing the entry of an unscheduled flight does not invalidate the schedule.

Processing scheduled flights entering a sector works in much the same way. The only difference is that prior to the operations described for the unscheduled case, the sector must clear all data relating to the scheduled flight. This includes removing both the scheduled entrance and exit from the schedule and removing the flight from the scheduled flights structure. By removing these entries at the same time, the scheduled flights structure and the schedule remain synchronized as required. The remainder of the operation is identical to the previous case.

Since both the scheduled entrance and exit are removed, the net impact on sector occupancy is still zero. The sector occupancy cannot be driven below zero in this case. Removing the events regarding the scheduled crossing removes a temporary increment to the sector's scheduled occupancy and does not introduce additional negative occupancy changes. Again, this operation is correct and will result in valid schedule state provided the state was valid at the start of the operation.

Exiting a Flight

The cases where a flight exits into another sector and exits to its destination airport are identical from the perspective of the current sector's schedule. First, the schedule event corresponding to the flight's exit and the entry relating to the flight in the structure of current flights are deleted. Also, the special event has its change value decremented to reflect the departed flight and is reinserted into the schedule at the time of the flight's exit.

Again, the requirement of a zero net change is satisfied, as is the required matching between the schedule and the current flights structure. While the removal of the flight cannot bring the scheduled occupancy of the sector below zero, moving the special event with the current flights could. That does not occur in this case, since for every increment stored in the special event, a corresponding decrement representing the exit of that flight exists at some later scheduled time. The reinsertion of the special event cannot move it later than any remaining exit events, since doing so would mean that the exit event's callback did not occur. Therefore, exiting a flight is correct provided that the schedule was correct before the operation.

In the event that a flight attempts to exit the sector and fails due to overcrowding in the next sector, relatively little work is required to update the schedule. The current flights structure changes the exit time associated with the flight to the new exit time, and the schedule similarly moves the flight's exit event to the new time, preserving the correlation between the structures. The special event with the current flights is not moved.

Since no events were added or removed, only moved, the schedule's final occupancy is unaffected and still zero. Since the exit time is pushed back, the only difference in sector occupancy is noticed between the old and the new exit times as an increment. Therefore, the scheduled sector occupancy can never fall below zero, and the operation is valid so long as the schedule was in a correct state before the attempted exit.

Scheduling and Rescheduling a Flight

After takeoff, each anticipated flight crossing is inserted into the schedule. This is accomplished by creating a scheduled flight entry for the flight along with anticipated entry and exit times, and inserting into the schedule events corresponding to these events at the appropriate times.

Since one flight entry and one flight exit are inserted into the schedule, the net change is zero, as required. Since the scheduled time of the extra flight entry precedes the schedule time of the flight exit, the scheduled occupancy of the sector cannot fall below zero because of this operation. The schedule event containing the current flights is not moved or modified. Therefore, the scheduling operation results in a correct schedule state assuming a previously correct schedule state.

Rescheduling a flight crossing is very similar to scheduling a flight crossing from the perspective of the schedule structure. First, the old schedule entry is cleared, just like the schedule entry was cleared during a scheduled flight entrance operation. The same procedure and reasoning applies here, so this portion of the operation has been shown to be correct. Next, the new schedule entries are added to reflect the new anticipated crossing times. This operation is identical to that described and proved correct in the previous case. Therefore, just like the preceding cases, the rescheduling operation is correct provided only that the schedule structure was in a valid state prior to the operation.

Start-up Correctness

Finally, to conclude by induction that the schedule remains valid in any reachable state, the initial state of the schedule structure must be valid. At simulation startup, both the current flights structure and the scheduled flights structure are empty. The schedule has a single entry, the special event for current flights listed at the simulation start time holding no flights. The schedule correctly does not have any events relating to nonexistent flights, and the number of flights in the current flights structure equals the value in the special event at zero. The only total occupancy figure, the one belonging to the special event, is correct at zero. Furthermore, since the special event is also the last event in the schedule, the schedule also correctly concludes with a total scheduled occupancy of zero. Therefore, the schedule structure is in a valid state upon initialization. Thus, the proof by induction that any reachable state of the schedule structure is valid is complete.

3.3 Flight Plan Generation

While most of the data required to operate MEANS is readily available, public listings of flight plans are inadequate for the purposes of the simulation. Available listings do not cover all origin-destination city pairs and are highly irregular in format. In order to test the new airspace system, approximate flight plans are needed. The plans need to be generated using only a list of all city pairs required and a description of all of the sectors under consideration. The sectors are described as lists of geographic coordinates detailing the outlines of the sectors.

To develop the flight plan, first all geographic coordinates in terms of latitude and longitude are converted into nautical miles north and east of Topeka, Kansas, using the degrees to nautical miles conversions found at Topeka, Kansas. Topeka was selected because it is roughly in the center of the continental United States. The overall flight path generated is a straight line in the new coordinate space from the origin to the destination. This path is tested every one hundred nautical miles to determine which sector that point of the path is in. If a sector transition is detected, the transition point is refined to within ten nautical miles. This error distance is traversed by modern airliners in under two minutes, so it will not severely impact the results of the simulation. The flight plan generated includes each sector crossing with its calculated distance at the heading of the overall flight path. All distances are scaled so that the total distance of the flight plan equals the great circle distance between the origin and destination airports.

To simplify the determination of sector membership of a given point, each sector is collapsed into a single point at the average of all of the points making up the perimeter of the sector. A point is classified as in a sector if it is closer to that sector's representative point than it is to the center points of any other sector. While this method may seem to oversimplify the problem, it produces reasonable results. Most of the sectors are shaped such that they are reasonably approximated in a closest point algorithm. Also, most sectors have elaborate border geometry, meaning that a precise calculation of sector membership would be prohibitively computationally intensive. Furthermore, since the flight path itself is an approximation and sector transitions are only accurate within ten nautical miles, the added degree of precision may not even result in noticeably different flight plans.

The generated flight plans are intended to be temporary approximations of the actual flight plans. Hopefully in the future, the MEANS project will receive or compile enough flight plan data that these approximations are no longer needed. Since the generated flight plans must be used for the time being, it is important to pinpoint their deficiencies. Besides only being accurate to within ten nautical miles, the flight plans also rely on the straight line flight path approximation and simplified sector membership calculations. Combined, these approximations mean that the sectors crossed in the flight plan may not precisely match the sectors actually crossed on a real flight plan. The sector listings should be close enough for testing purposes. Also, the total distance for the flight plan, set to be the great circle distance between the departure and arrival airports, is an underestimate. Real flights have to position themselves for landing approaches and must fly set departure patterns. Consequently, both the flight path distance and corresponding flight time of the plans generated will be somewhat less than their real values. Furthermore, this method of generating flight plans will not work well for areas outside of the United States. For European flights, at a minimum Topeka must be replaced as the origin city. For worldwide flights, great circle routes would invalidate the assumed straight line routes, rendering this technique of flight plan approximation useless.

Chapter 4

Results and Validation

In order to determine the usefulness of both the old historical distribution system and the new sector based system, two analysis metrics are used. First, the distribution of all simulated flight delays is compared with the distribution of all actual flight delays. The closer the simulated distribution matches the actual distribution with regard to shape and key parameters such as mean and standard deviation, the better the simulated results. Second, for each flight, the simulated delay is compared with the actual delay. In a perfect simulation capturing all of the factors present in the real world, all of the delays will be the same. Naturally, no simulation is perfect, and some variation in the per flight delays will appear. Charting the difference between the actual delay and the simulated delay shows the degree to which the simulated delay matches reality. Better simulations yield charts centered close to zero with as little variance as possible. Both of these measure the error of MEANS as a whole rather than specifically the error of the en route portion. Consequently, the historical distribution airspace module is used as a control. The system presented in this thesis is evaluated based on the changes seen in the overall results obtained by replacing the old airspace implementation with the new one. Additional metrics are introduced as needed.

4.1 Experimental Setup

The day of March 2, 2005 serves as the basis for the experiments to determine the merits of the new airspace system. The quality of the data available to the MEANS project is best for this month. The exact day selected is the first Wednesday of the month. The schedule of flights to simulate is generated from Bureau of Transportation Statistics' On-Time Performance (BTS OTP) data, containing all flights for the major carriers in the United States [2]. To compensate for additional flights belonging to smaller carriers and to private owners, additional padding flights are introduced to present realistic flight loads to airport control towers. The volume of these padding flights is derived from Aircraft Situation Display to Industry (ASDI) data, which consist of actual arrival and departure counts for fifteen minute intervals at major airports [1]. Airport area weather and weather predictions used in the simulation are simple replays of actual historical conditions and predictions. The distribution of flight times in between city pairs observed during 2005 provide the basis for the historical distribution airspace implementation.

Additional data is needed to operate the new sector based airspace implementation. First, flight level wind readings come from the Radiosonde Observation (RAOB) database [10]. This database does not provide enough readings to provide a separate reading for each sector. Instead, one reading is selected per ARTCC, and that reading is applied to each of the sectors in that center. Sector and ARTCC boundaries provided by the FAA provide a basis for the flight plan generation program detailed in section 3.3. Flight plans are generated for both sector crossings and center crossings to allow for testing at different granularities. A flight plan is generated for each city pair flown in the selected flight schedule. The maximum capacity for all sectors is set at fifteen, which is roughly the average for sector capacities as determined by their MAP values. The exact capacities for all sectors is not publicly available. For centers, the maximum capacity is the sum of all of the capacities of their component sectors.

4.2 Results

4.2.1 Actual Delay Statistics

The overall delay distribution for the flights reported in the BTS OTP data, shown in figure 4-1, shows a roughly normal distribution with a noticeable rightward skew. The distribution is centered around zero, meaning that on average flights arrived roughly on time during this day. No flights arrived more than an hour early, and only a small fraction of the flights arrived more than two hours late. This is understandable, since it is highly unlikely that flights will arrive substantially early, and except under severe circumstances, flights rarely arrive extremely late.

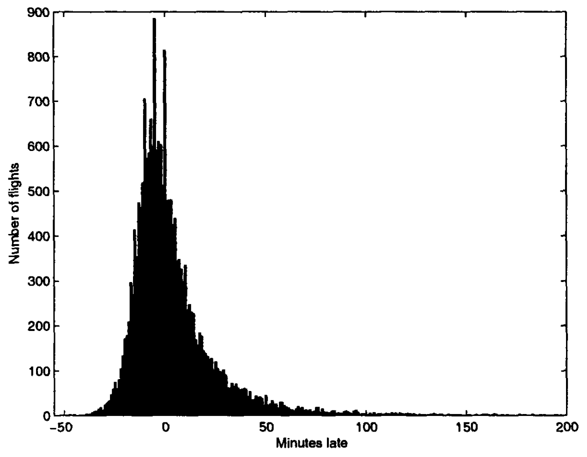


Figure 4-1: Actual flight delays

Table 4.1: Statistics for figure 4-1

n	19916
mean	4.0
median	-1
std dev	24.6
range	-52 – 586
# > 200	30
# < -55	0

4.2.2 Historical Distribution Implementation

The MEANS experiment to generate this data took three minutes to complete. The distribution of delays produced by MEANS using the older, historical distribution airspace implementation is shown in figure 4-2. While the general shape of the distribution is roughly correct, the large secondary spike is unusual. Additionally, the per-flight differences from the actual delays in figure 4-3 are relatively small, forming a roughly normal curve around zero minutes of deviation. Besides the spike, these figures show that the old system produced better results than originally thought. However, both distributions are centered about twelve minutes too early relative to the actual delays. Since these figures show the delay on arrival, it is not clear which component of MEANS is responsible for the systematic early arrival of the flights.

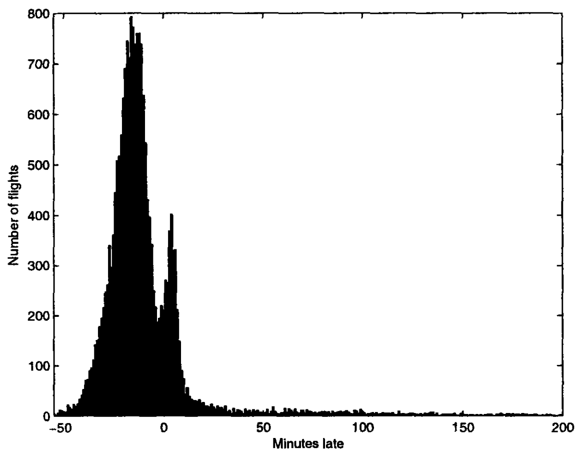


Figure 4-2: Flight delays predicted by the historical distribution implementation

Table 4.2: Statistics for figure 4-2

n	20072
mean	-8.3
median	-14
std dev	30.6
range	-104 – 583
# > 200	68
# < -55	30

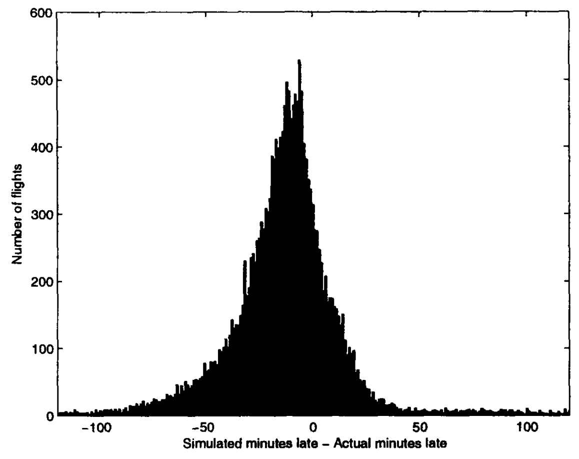


Figure 4-3: Deviations from actual delays for the historical distibution implementation

Table 4.3: Statistics for figure 4-3

n	19794
mean	-12.4
median	-12
std dev	36.6
range	-602 – 558
# > 120	214
# < -120	131

The difference in flight time between the actual flights and the simulated flights, removing all other aspects of the simulation, is shown in figure 4-4. Since the actual flight times are recorded from wheels up time to wheels down time, the simulated time includes waiting in the arrival queue to land in addition to the en route time. While the figure shows a slight secondary spike, the distribution is very tight and centered almost exactly at zero. This suggests that the airspace module is not to blame for the early arrivals of the simulated planes. Also, this data suggests that the older airspace module is significantly better than previously thought, introducing very little error into the simulation. The secondary spike is still an issue. It most likely arises from a secondary flight time calculation method employed when a historical distribution is not available. If no historical distribution is available, the flight time is calculated as a function of the distance between the two airports and the scheduled duration of the flight.

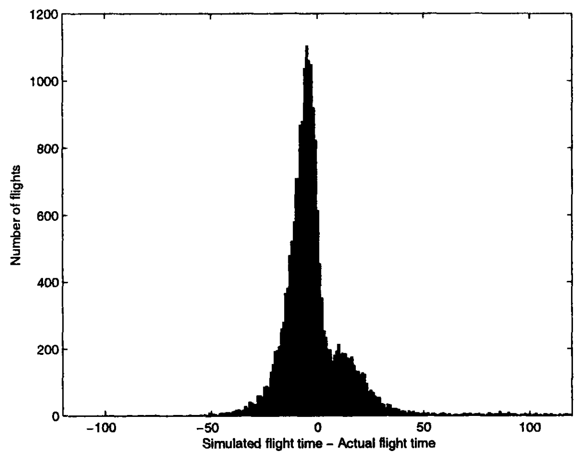


Figure 4-4: Deviations from actual flight times for the historical distribution implementation

Table 4.4: Statistics for figure 4-4

n	19794
mean	-1.4
median	-4
std dev	19.2
range	-92 – 190
# > 120	149
# < -120	0

4.2.3 Sector-Based Implementation: Center Granularity

In this experiment, the new airspace module is configured to simulate air traffic at the ARTCC level. This simulation took only fifteen seconds longer than the original simulation, an eight percent increase, suggesting a minimal runtime impact of the new system. Figures 4-5 and 4-6 correspond to the first two figures from the previous section. While the shape of the overall distribution more closely matches the actual distribution, both figures are shifted another eleven minutes too early. As shown in figure 4-7, despite having an exceptionally tight distribution of flight time deviations, the additional eleven minutes is indeed a product of the new airspace implementation. While the new system appears promising at this point, the fact that it has introduced a systematic error of eleven minutes is worrisome.

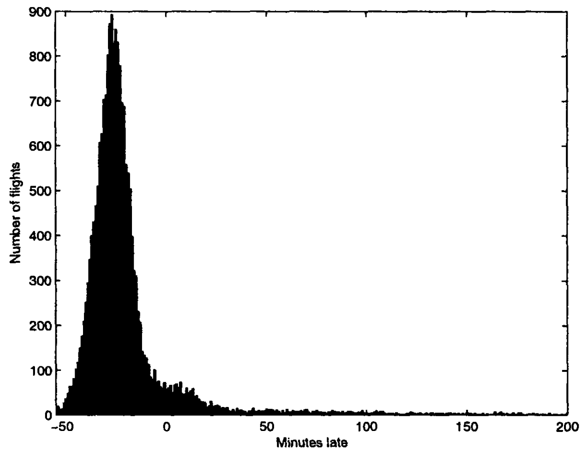


Figure 4-5: Flight delays predicted by the sector-based implementation at center granularity

Table 4.5: Statistics for figure 4-5

n	20072
mean	-18.9
median	-25
std dev	30.6
range	-118 – 574
# > 200	45
# < -55	29

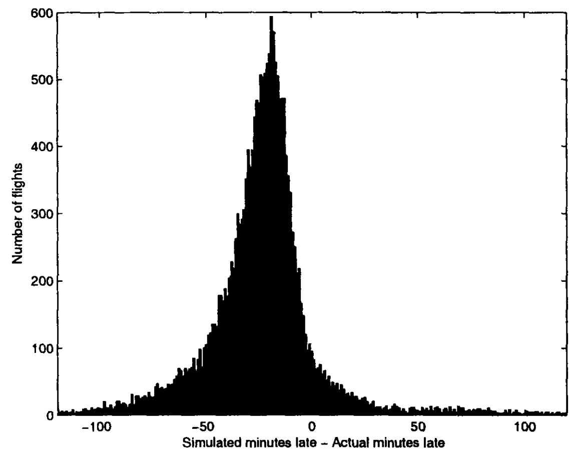


Figure 4-6: Deviations from actual delays for the sector-based implementation at center granularity

Table 4.6: Statistics for figure 4-6

n	19794
mean	-23.0
median	-23
std dev	36.0
range	-616 – 530
# > 120	195
# < -120	170

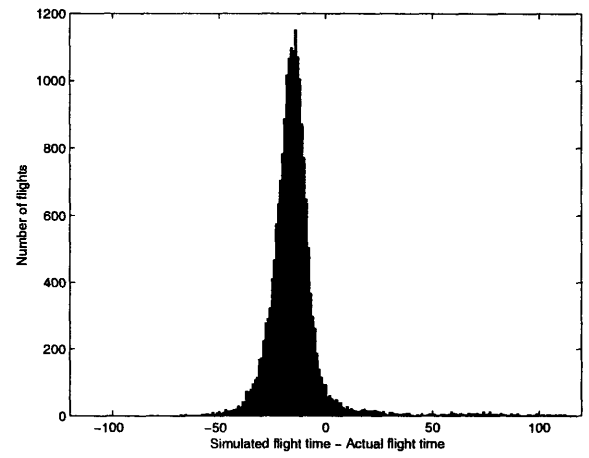


Figure 4-7: Deviations from actual flight times for the sector-based implementation at center granularity

Table 4.7: Statistics for figure 4-7

n	19794
mean	-13.8
median	-16
std dev	18.5
range	-82 - 205
# > 120	89
# < -120	0

4.2.4 Sector-Based Implementation: Sector Granularity

For this experiment, MEANS is configured to simulate individual sectors. The runtime of the experiment was five and a half minutes, two and a half minutes longer than the same experiment with the old implementation. Since there are roughly fifteen sectors per ARTCC, the new airspace module must coordinate fifteen times more components. The increase in runtime is ten times larger than the increase for the center level experiment, which indicates a roughly linear growth in the runtime of the simulation with additional sectors.

As shown in figures 4-8 and 4-9, this experiment produces the best shaped overall delay curve, and a more symmetrical per-flight delay difference than the same experiment at center granularity. The differences in flight times shown in figure 4-10 are almost as good as the previous experiment. The flights are still arriving eleven minutes too early, suggesting that the problem observed in the previous case still applies when addition detail is added. However, the means and standard deviations of the first two figures suggests a serious problem. Figures 4-8 and 4-9 have extreme tails to the right, containing roughly fifteen percent of the total flights between 200 and 1000 minutes late. Since this tail is not nearly as noticeable in figure 4-10, the simulated flight time cannot be entirely responsible for the problem. Instead, the problem must relate to aircraft not being able to take off due to excessive airspace induced ground holds. Since the capacity of the airspace over each airport has been reduced to fifteen by switching to sector level simulation, passing air traffic could easily oversaturate the sector, prevent aircraft from departing normally. While the system avoids deadlock in the air, flights waiting to enter the airspace may be held indefinitely, since they do not have access to the deadlock preventing additional sector occupancy slots.

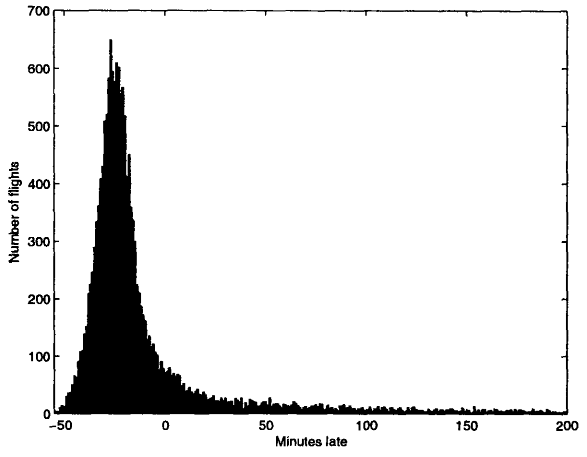


Figure 4-8: Flight delays predicted by the sector-based implementation at sector granularity

Table 4.8: Statistics for figure 4-8

n	20072
mean	85.5
median	-19
std dev	218.7
range	-118 – 1423
# > 200	3627
# < -55	15

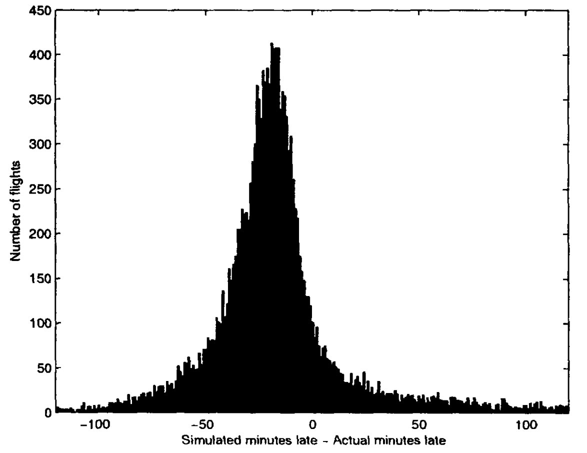


Figure 4-9: Deviations from actual delays for the sector-based implementation at sector granularity

Table 4.9: Statistics for figure 4-9

n	19794
mean	81.0
median	-15
std dev	218.8
range	-531 - 1132
# > 120	3968
# < -120	133

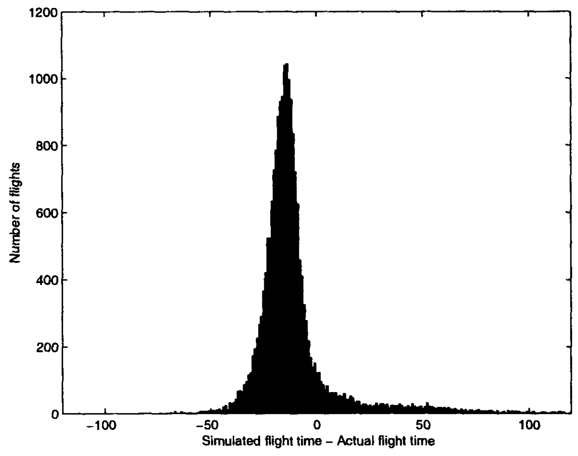


Figure 4-10: Deviations from actual flight times for the sector-based implementation at sector granularity

Table 4.10: Statistics for figure 4-10

n	19794
mean	-10.0
median	-14
std dev	24.0
range	-83 - 487
# > 120	118
# < -120	0

4.2.5 Case Study of New York Area

In addition to accurate overall simulation, MEANS should also simulate effects within an individual market. To test this, figures 4-11, 4-12 and 4-13 were generated using only flights arriving at or departing from LGA, JFK, and EWR, all New York City area airports. Looking first at figure 4-11, generated by the historical distribution implementation, while the general distribution is no worse than in the comparable system-wide case in figure 4-3, the distribution has shifted an additional ten to fifteen minutes too early. Both examples of the new system in figures 4-12 and 4-13 are shifted eleven minutes earlier than the historical airspace model, just as in the overall case. Since the relationship between the different airspace implementations remains the same, the overall shift to the left is most likely an artifact of another module of MEANS behaving poorly for the New York area airports. However, this data does not suggest that either the old or new airspace models handles individual markets any differently than it handles the entire airspace.

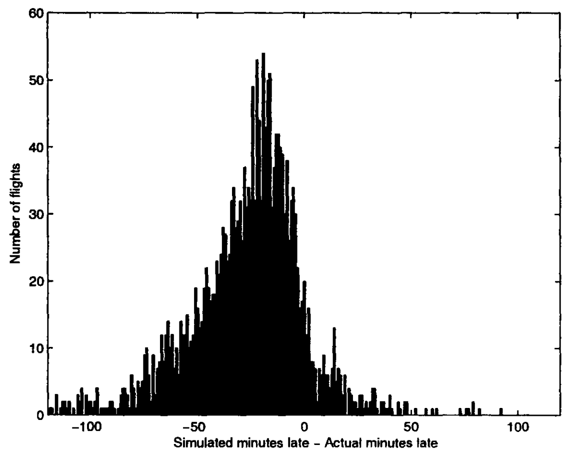


Figure 4-11: Deviations from actual delays in the New York area for the historical distribution implementation

Table 4.11: Statistics for figure 4-11

n	2112
mean	-27.8
median	-23
std dev	35.4
range	-515 – 558
# > 120	4
# < -120	26

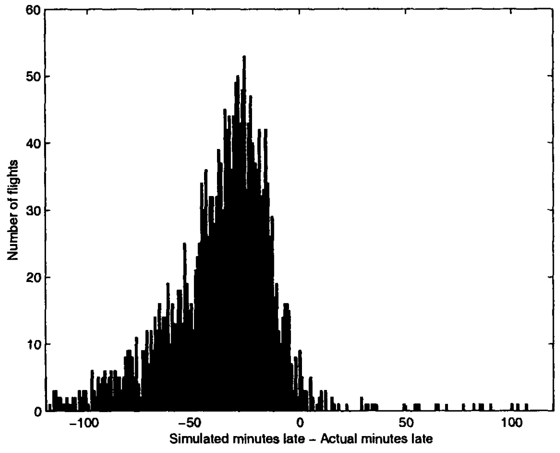


Figure 4-12: Deviations from actual delays in the New York area for the sector-based implementation at center granularity

Table 4.12: Statistics for figure 4-12

n	2112
mean	-39.0
median	-34
std dev	35.3
range	-531 – 530
# > 120	4
# < -120	40

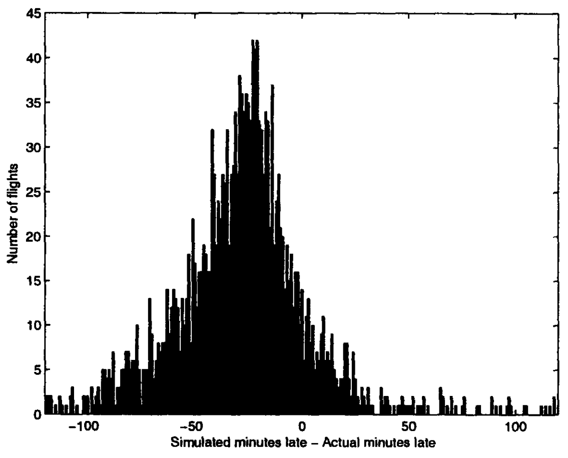


Figure 4-13: Deviations from actual delays in the New York area for the sector-based implementation at sector granularity

Table 4.13: Statistics for figure 4-13

n	2112
mean	15.5
median	-25
std dev	156.5
range	-531 - 833
# > 120	196
# < -120	36

Introducing a Failure

One of the primary goals of the new airspace implementation is to simulate severe local failures. A special experiment reducing the New York ARTCC capacity to ten percent of its normal value between noon and 6PM tests this capability. This reduction is an artificial creation, but it could have come about in response to a failed radar system or severe weather in the area. The simulated impact on the New York area flights is shown in figure 4-14, generated by the new sector-based implementation operating at center level granularity. While actual data is not available for this hypothetical case, the figure clearly shows that the capacity reduction severely impacted delays for the flights operating in that area. These delays arise from aircraft being unable to take off into the reduced and overburdened airspace as well as inbound flights having trouble reaching their destinations. Additional testing on documented events is required to fine tune this failure model, but this data shows that the basic mechanism works as intended.

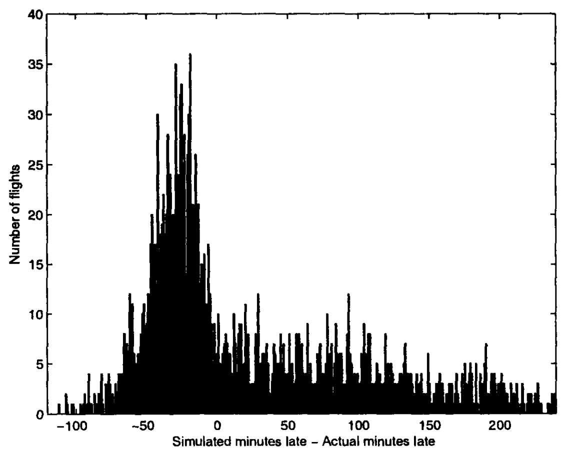


Figure 4-14: Deviations from actual delays in the New York area with failures for the sector-based implementation at centers granularity

Table 4.14: Statistics for figure 4-14

n	2112
mean	30.8
median	-9
std dev	93.0
range	-531 – 529
# > 120	364
# < -120	15

4.3 Analysis

These tests confirm that the new airspace implementation works correctly and generates flight times that approximate reality, despite some problems. A secondary discovery in the results is that the older, historical distribution airspace implementation produces better flight times than originally thought. This difference between the simulated and actual flight times from figure 4-4 are quite small, and the distribution of errors is correctly centered at zero. Previously, no information regarding the accuracy of the historical distribution model was available. However, the historical distribution gives weight to flight times under normal wind and airspace load conditions. The day chosen for the experiment did not have unusual wind patterns or major capacity constraints, so the flight times produced by the historical distribution are as good as they can get. Under adverse conditions, like unusual winds or major capacity disruptions like the one explored in section 4.2.5, the historical model would perform less well. Another observation regarding the results is that the other modules in MEANS appear to be responsible for simulated flights arriving roughly twelve minutes too early. This bias seriously hampers MEANS usefulness in predicting delays and late arrivals. As shown in section 4.2.5, this bias is not uniform across the system. One possible source of this bias is the taxiing module, which computes optimal taxi times, assuming no restrictions in aircraft maneuvering. It is possible that these simulated taxi times are consistently too short. While discovering and repairing this bias is important to the MEANS project, doing so is out of the scope of this thesis.

While the preliminary results indicate that the new airspace implementation works correctly, the results bring forward two questions about the new system's usefulness. First and most critically, the new system predicts flight times that are almost uniformly eleven minutes too short. The consistent bias in the flight times presents a real hurdle to the viability of the new implementation. Without finding the cause of the bias and repairing it, the new system may only be useful to simulate unusual cases rather than serving as a complete replacement to the old system. The second issue with the new system regards the extremely large rightward tail on the delays predicted at the sector granularity. Since the flight times generated by the new system at the sector granularity do not exhibit as extreme a rightward tail as the overall delays, the problem seems to be with the ground holds associated with airspace overcrowding. This effect will require additional investigation before the new system should be used for per sector simulation.

4.3.1 Flight Duration Correlation

One possible source of the bias in flight times predicted by the new implementation is that the airspeeds are too high or the flight paths are too short. The airspeeds are not precisely matched to the individual aircraft, so the airspeeds could be wrong. Also, the simplified flight paths used to generate the flight plans may cut portions of the flight, reducing the overall flight mileage and incorrectly lowering the flight times. In both of these cases, the magnitude of the error would be correlated to the duration of the flight. To see if this is the case, consider the relationship between the actual duration of the flight and the simulated duration of the flight for the different implementations and granularities, as shown in figures 4-15, 4-16, and 4-17. The line actual duration = simulated duration is charted for reference. None of the three graphs show increasing variance with duration. In fact, there appears to be a slight decrease in variance with flight duration, particularly the negative variance responsible for the bias. The only noteworthy element of these plots is that figure 4-17 shows that the sector granularity experiment of the new airspace system produced some overly long outliers for longer flights. This suggests that whatever issue is affecting take off delays may be impacting flight times as well. In any case, incorrect airspeeds or flight path lengths do not appear to be the cause of the bias.

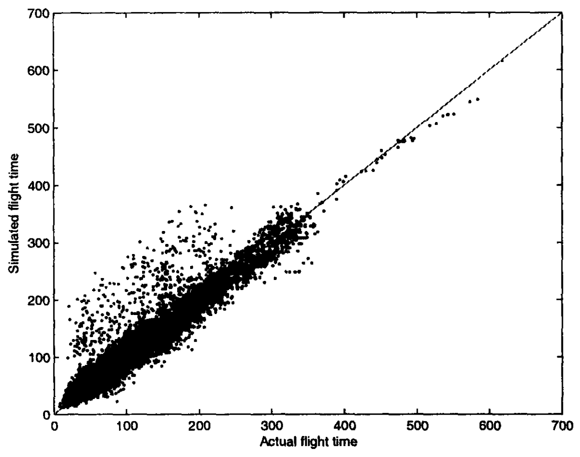


Figure 4-15: Flight duration correlation for the historical distribution implementation

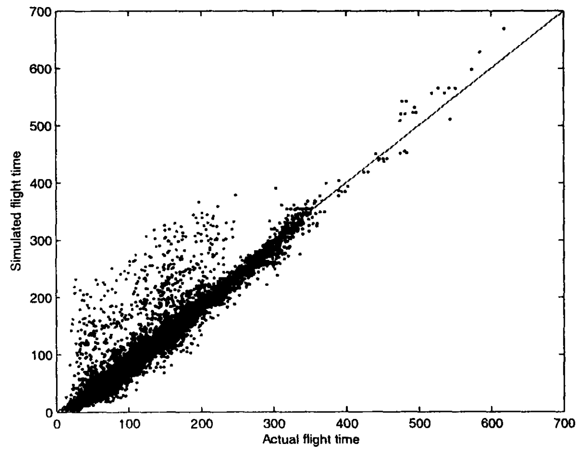


Figure 4-16: Flight duration correlation for the sector-based implementation at center granularity

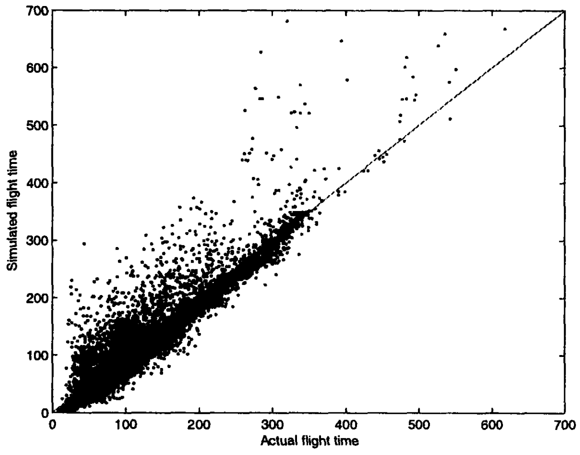


Figure 4-17: Flight duration correlation for the sector-based implementation at sector granularity

4.3.2 Acceleration Considerations

The approximate flight trajectories should capture the entire flight distance. However, they do not factor in the effect of acceleration during the ascent to flight level after takeoff and the deceleration during the descent from flight level before landing. Since the new implementation assumes constant airspeed, the simulated aircraft flies too fast for the beginning and ending of its journey. Assuming a constant acceleration from 200 knots to 450 knots over the first twenty minutes of the flight, the flight will travel roughly 108 nautical miles during that time. If the aircraft were traveling that distance at a constant 450 knots, it would take only fourteen minutes. If the six minute difference applies at both the beginning and ending of the flight, this could explain the entire eleven minute bias.

To test the impact of ignored accelerations, the experiments at both the center and sector granularity for the sector-based implementation are repeated with an additional 100nm flight segment inserted at the beginning of the flight plan. This segment crosses an artificial sector with infinite capacity. At 450 knots, the flights take about thirteen minutes to cross, which should negate the bias seen earlier. Additionally, the added sector at takeoff may compensate for the unusual behavior of the airspace induced ground holds in the sector granularity experiment.

The new delay distributions are shown in figures 4-18 and 4-20, and the new flight time deviations are shown in figures 4-19 and 4-21. In all cases, the airspace induced bias is almost completely eliminated. At the center level, this experiment produced the best delay distribution and the smallest deviation from the actual flight times. At the sector level, the same improvements apply to the bulk of the distributions, but the troublesome rightward tail remains. Clearly, adding compensating flight segments has solved the issue of bias. Unfortunately, this technique did not solve the rightward tail problem in the sector granularity experiment's delays.

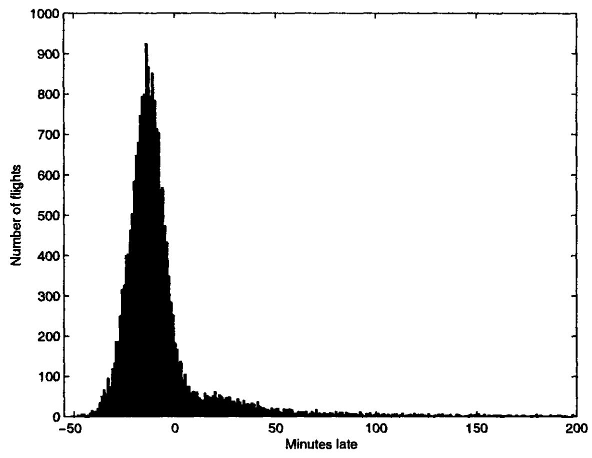


Figure 4-18: Flight delays predicted by the sector-based implementation at center granularity with modified flight plans

Table 4.15: Statistics for figure 4-18

n	20070
mean	-5.1
median	-12
std dev	32.0
range	-105 – 611
# > 200	76
# < -55	1

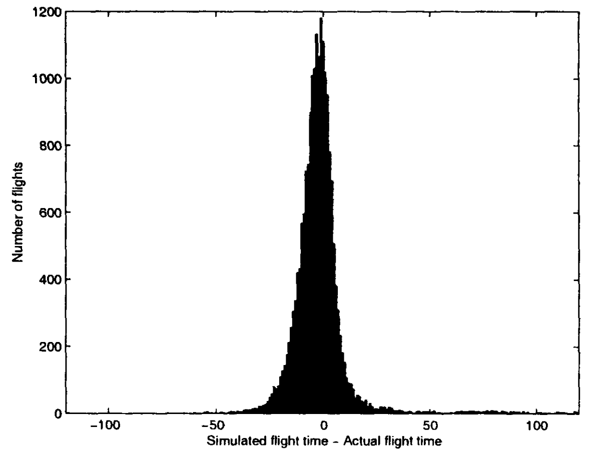


Figure 4-19: Deviations from actual flight times for the sector-based implementation at center granularity with modified flight plans

Table 4.16: Statistics for figure 4-19

n	19792
mean	-1.0
median	-2
std dev	17.3
range	-73 -- 219
# > 120	92
# < -120	0

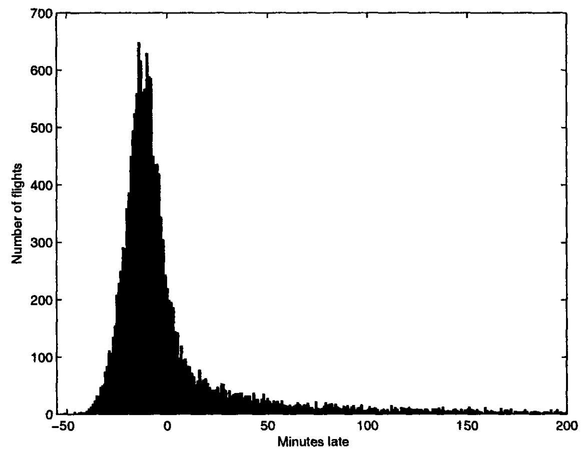


Figure 4-20: Flight delays predicted by the sector-based implementation at sector granularity with modified flight plans

Table 4.17: Statistics for figure 4-20

n	20070
mean	101.0
median	-5
std dev	221.6
range	-105 - 1372
# > 200	3724
# < -55	1

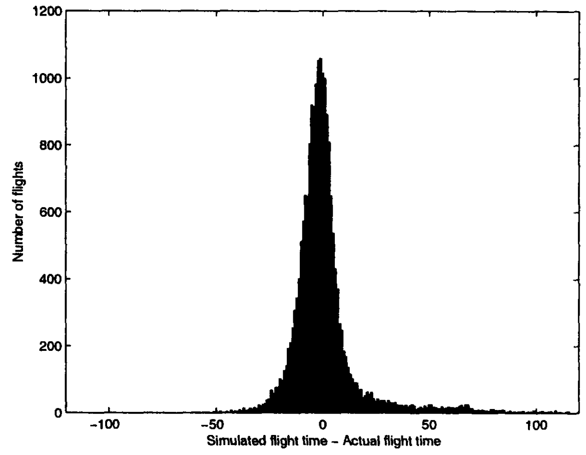


Figure 4-21: Deviations from actual flight times for the sector-based implementation at sector granularity with modified flight plans

Table 4.18: Statistics for figure 4-21

n	19792
mean	1.7
median	-1
std dev	20.4
range	-74 - 361
# > 120	96
# < -120	0

4.3.3 Additional Overload Slots

One additional possibility for repairing the sector based experiments rightward tail is to make the flight acceptance rules for crowded sectors to even more lenient. In practice, air traffic controllers have the option to raise a sector's capacity by up to three [8]. It follows that raising the number of flights over the capacity of a sector to admit from adjacent crowded sectors to three might solve some of the rightward tail problems seen in the delay graphs. Table 4.19 represents the statistic generated from repeating the experiment in section 4.2.4 with this modification. The overall graphs did not have a noticeable change. The statistics indicate that while this did in fact reduce the tail's extent and magnitude, it only reduced them by a small amount. The crucial problem of airspace induced ground holds remains unsolved.

Table 4.19: New statistics for figure 4-9 with additional extra slots. Compare to table 4.9

n	19794
mean	73.2
median	-15
std dev	206.6
range	-531 – 1037
# > 120	3819
# < -120	134

Chapter 5

Conclusions and Future Work

The results indicate that the new sector-based system of modeling individual ARTCCs and employing compensating flight segments is the best airspace implementation to date. It surpasses the older system based on historical flight time distributions in terms of delay distribution shape and deviations from actual flight times. Additionally, the new system is capable of modeling unusual flight level wind conditions and airspace capacity modifications, which the distribution based system could not model. Despite the numerous approximations and assumptions in the new model required for testing, it is ready to replace the existing implementation.

Despite showing some promise, the new implementation does not seem useable when modeling individual sectors. In all cases, a large tail to the right of the simulated delay curve accounting for approximately fifteen percent of the data points renders the results generated practically useless. The source of the tail appears to be unnecessary airspace induced ground holds in response to overcrowding in en route sectors. Ignoring the tail, the sector level simulations produced the most symmetric deviations from actual flight times and produced an overall delay distribution that most closely matched the actual distribution. While not ready for active use currently, sector level simulation using the new airspace implementation will be the method of choice if the tail can be removed.

One possible reason for the different behavior of the new implementation at different granularities is that the larger area of the centers mitigates some of the effects of the approximations in the system. At the center level, small scale differences between the simulated flight path and the actual flight path are hidden by the relatively large ARTCC areas and capacities. These differences could include approximation errors of the flight plan generator and small scale reroutings in response to local crowding or severe weather. The net effect of these differences on the center crossings would not be large and would typically not modify the order of center areas crossed. At the more fine grained sector level, small modifications to the predicted flight plans often result in crossing other sectors than those in the simulation. Inaccurate flight plans could artificially congest some sectors and form bottlenecks. This would result in the delay tails observed.

Additionally, the analysis required for understanding the behavior of the presented implementation provides insight into the other MEANS components. This research demonstrates the accuracy of the original distribution based airspace. In fact, early analysis revealed a serious bug in the script responsible for generating the required distributions. Without this analysis, this defect might have gone undetected. Now, the historical distribution implementation is thoroughly understood and known to be mostly defect free. Also, analysis into the overall flight delays of MEANS revealed a twelve minute bias in some module or modules other than the airspace module. Once the source of this bias is found and fixed, MEANS will produce flight delays that almost exactly match the actual values.

5.1 Future Work

5.1.1 Acceleration Compensation Calibration

While the compensating flight segment works perfectly in the presented experiments, more research is needed to precisely quantify and understand the effects and parameters of this extra segment. At the very least, the compensating segment should be broken in half, putting one half before the first actual segment and the other half after the last. This more accurately represents the additional flight time arising from takeoff and landing accelerations. However, the current technique of putting the entire compensating distance at the beginning of the flight appears to work accpetably.

5.1.2 Aircraft Cruising Speeds

For the purposes of this thesis, all aircraft cruising speeds are assumed to be 450 knots. This is not the case in the real world. Accurately mapping aircraft to cruising speeds would reduce the remaining errors in the predicted flight times.

5.1.3 Improved Flight Plans

The flight plans used in this thesis are approximations. Accurate information on sector crossings and distances would also reduce the errors in predicted flight times. Additionally, accurate sector crossing information might reduce incorrect localized overcrowding for the sector level simulation, possibly reducing the observed delay tail. At the center level, this is less of an issue because the large overall capacity of the ARTCC can mitigate the effects of local overcrowding.

5.1.4 Improved Flight Level Wind Data

In all sector level experiments, all sectors belonging to the same center have the same wind conditions simply because the available data is too sparse to provide readings for each sector. Finding a better source of flight level wind data than the RAOB database could reduce the error in the flight times even further.

5.1.5 New Airspace Induced Ground Hold Rules

As the severe tail in the delays predicted by the sector level simulation indicate, the rules governing ground holds dictated by the airspace need to be reevaluated. The current system of holding flights until they encounter no expected airspace delay is clearly too constraining. Both the flight entrance rules and the computation of check() appear to limit aircraft movement more than necessary. Discovering and implementing a new rule set should eliminate the delay tail and allow the more precise sector level simulation to be used confidently. For center level simulation, overcrowding is less common since the larger total capacity of the centers minimizes the impact of localized crowding. Consequently, ground holds far are less common under the current rule set, so the impact on the center level simulation of a new rule set is minimal.

Bibliography

[1] Aircraft Situation Display to Industry (ASDI). http://www.fly.faa.gov/ASDI/asdi.html.

[2] Airline On-Time Statistics. http://www.bts.gov/programs/airline_information/airline_ontime_statistics/.

[3] Elizabeth Bly. Effects of Reduced IFR Arrival-Arrival Wake Vortex Separation Minima and Improved Runway Operations Sequencing on Flight Delay. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2005.

[4] John-Paul Clarke, Terran Melconian, Elizabeth Bly, and Fabio Rabbani. MEANS - MIT Extensible Air Network Simulation. Submitted for publication, 2006.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, second edition, 2001.

[6] Jonas Dino. Future ATM Concepts Evaluation Tool (FACET). http://www.nasa.gov/centers/ames/research/lifeonearth/lifeonearth-facet.html, March 2006.

[7] Michelle Eshow. Airspace, Procedures, and Flight Plans. http://sdg.lcs.mit.edu/workshop/atc_overview.PDF, September 2006. Session 1 of the Redesigning Air Traffic Control workshop.

[8] Section 7. Monitor Alert Parameter. http://www.faa.gov/Atpubs/FAC/Ch17/s1707.html, February 2006. Part of Chapter 17 of FAA Order 7210.3, Facility Operation and Administration.

[9] Chapter 20. AIR NAVIGATIONAL ROUTES. http://www.faa.gov/atpubs/AIR/air2001.html, February 2006. Part of FAA Order 7400.2F, Procedures for Handling Airspace Matters.

[10] Mark Govett. Radiosonde Database Access. http://raob.fsl.noaa.gov/.

[11] John L. Hennessey and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, Boston, Massachusetts, third edition, 2003.

[12] John Lundgren. Airliners.net: Aircraft info and history section. http://www.airliners.net/info/.

[13] Terran Melconian. Effects of Increased Nonstop Routing on Airline Cost and Profit Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, September 2001.

[14] MIT Extensible Air Network Simulation. http://means.mit.edu/means/.

[15] National Geospatial-Intelligence Agency. Digital Aeronautical Flight Information File. https://164.214.2.62/products/digitalaero/index.cfm, January 2006.

[16] Ed Williams. Aviation Formulary V1.42. http://williams.best.vwh.net/avform.htm, July 2004.