DECENTRALIZED AIRCRAFT LANDING SCHEDULING AT SINGLE RUNWAY NON-CONTROLLED AIRPORTS

A Dissertation

by

YUANYUAN DING

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2007

Major Subject: Aerospace Engineering

DECENTRALIZED AIRCRAFT LANDING SCHEDULING AT SINGLE RUNWAY NON-CONTROLLED AIRPORTS

A Dissertation

by

YUANYUAN DING

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John Valasek
Committee Members,	John H. Painter
	Thomas Strganac
	Thomas R. Ioerger
Head of Department,	Helen Reed

May 2007

Major Subject: Aerospace Engineering

ABSTRACT

Decentralized Aircraft Landing Scheduling at

Single Runway Non-Controlled Airports. (May 2007)

Yuanyuan Ding, B.S., Beijing University of Aeronautics and Astronautics;

M.S., Beijing University of Aeronautics and Astronautics

Chair of Advisory Committee: Dr. John Valasek

The existing air transportation system is approaching a bottleneck because its dominant hub-and-spoke model results in a concentration of a large percentage of the air traffic at a few hub airports. Advanced technologies are greatly needed to enhance the transportation capabilities of the small airports in the U.S.A., and distribute the high volume of air traffic at the hub airports to those small airports, which are mostly non-controlled airports. Currently, two major focus areas of research are being pursued to achieve this objective. One focus concentrates on the development of tools to improve operations in the current Air Traffic Management system. A more long-term research effort focuses on the development of decentralized Air Traffic Management techniques.

This dissertation takes the latter approach and seeks to analyze the degree of decentralization for scheduling aircraft landings in the dynamic operational environment at single runway non-controlled airports. Moreover, it explores the feasibility and capability of scheduling aircraft landings within uninterrupted free-flight environment in which there is no existence of Air Traffic Control (ATC). First, it addresses the approach of developing static optimization algorithms for scheduling aircraft landings and, thus, analyzes the capability of automated aircraft landing scheduling at single runway non-controlled airports. Then, it provides detailed description of the implementation of a distributed Air Traffic Management (ATM) system that achieves

decentralized aircraft landing scheduling with acceptable performance whereas a solution to the distributed coordination issues is presented. Finally real-time Monte Carlo flight simulations of multi-aircraft landing scenarios are conducted to evaluate the static and dynamic performance of the aircraft landing scheduling algorithms and operation concepts introduced.

Results presented in the dissertation demonstrate that decentralized aircraft landing scheduling at single runway non-controlled airports can be achieved. It is shown from the flight simulations that reasonable performance of decentralized aircraft landing scheduling is achieved with successful integration of publisher/subscriber communication scheme and aircraft landing scheduling model. The extension from the non-controlled airport application to controlled airport case is expected with suitable amendment, where the reliance on centralized air traffic management can be reduced gradually in favor of a decentralized management to provide more airspace capacity, flight flexibility, and increase operation robustness.

ACKNOWLEDGMENTS

The writing of a dissertation can be a lonely and exhausting experience especially when you are working at the same time, yet it is obviously not possible without the personal and practical support of numerous people.

I would like to acknowledge many people for helping me during my doctoral work. I would especially like to thank my advisor, Prof. John Valasek, for his generous time and commitment. His kindness helped me get through the culture barrier I encountered the first couple of years after I arrived in the United States. Throughout my doctoral work he not only encouraged me to develop independent thinking and research skills, but also continually stimulated my analytical thinking and greatly assisted me with scientific writing.

I am also very grateful for having an exceptional doctoral committee and wish to thank Prof. John Painter, Prof. Thomas Strganac, and Prof. Thomas Ioerger for their continual support and encouragement. I owe a special note of gratitude to my colleagues at the Flight Simulation Lab for assisting me with collecting flight simulation data.

Special thanks go to RTI International, as part of the North Carolina and Upper Great Plains Small Aircraft Transportation System Laboratory. I gratefully acknowledge its financial support of my research.

I am extremely grateful for the assistance, generosity, and advice I received from my colleagues at S-TEC Corporation. Special thank goes to Elizabeth Brandi, my manager at S-TEC Corporation, for her generous support of my research.

I extend many thanks to my friends, especially Jimmy Rong, Song Deng, Xuyang Ding, Yuchun Yuan, and Wayne Frantz. I would like to thank Karen Knabe, Graduate Program Administrative Assistant of Department of Aerospace Engineering, for her help with all my paperwork throughout my doctoral program at TAMU.

Finally, I would like to thank my family. My mother was a constant source of support and happy spirit, and my father extended his passion in aerospace engineering to me. I am grateful to my sister and her family for their encouragement and enthusiasm. I am especially grateful to my wife for she has always been there for me, especial her mental support when I encountered huge difficulty in continuing my doctoral program one and half years ago. I would not have completed my doctoral program without her patience, persistence, and spiritual support.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

When the brothers Orville and Wilbur Wright performed the very first power-driven flight of a heavier-than-air machine in the world back on December 17th, 1903, they would have never imagined the tremendous growth of air traffic from thereon. The number of air travelers surged in particular due to the introduction of jet airliners in the late 1950s and the resulting jet-age in the 1960s. The data recording of the Air Transport Association of America (ATA) started with about 6,000 domestic air travelers back in 1926 and has its peak with 610, 600, 000 in 2000. A history of air traffic control from its beginning can be found in [1].

The demand for air travel will continue to increase over the next few decades. It has been forecast that by 2008 the number of passengers will increase 43 percent and an additional 2,500 planes will be needed to accommodate them. Under the current system, the additional traffic would cause a 250 percent rise in delays. These numbers were projected before the tragic events of September 11th, 2001 when air travel encountered a slump that year. It has been predicted that the airline industry will recover from the aftermath and air travel will increase again.

Meanwhile, it is becoming apparent that the existing air transport system is approaching a bottleneck. In the United States, flight delays and cancellations are a familiar part of air travel and the Air Transport Association claims that delays cost the U.S. airlines billions of dollars per year [2]. This is certainly not only a problem of the continental United States, but can be observed worldwide. In particular in Europe the airspace is already so congested that delays occur on a regular basis. The same problem will arise for Asia where the economic growth will spark air traffic in the near future as well.

This dissertation follows the style of *Journal of Guidance, Control and Dynamics*.

The problem mainly results from the dominant hub-and-spoke model that results in a concentration of a large percentage of the air traffic at a few airports. Although great efforts have been taken to improve the current Air Traffic Management (ATM) system and construct new runways, it appears that the continued projection of growth in aviation, as well as airline and civil aviation economics, shows that those airports will be inadequate to address the need for increased capacity [3].

The current hub-and-spoke system limits travelers to the air carrier's schedule and the inefficiencies of connections at distant hubs. Many regional trips can be driven in the same time or even less and for a lower cost than it takes to fly commercially when considering connection times and the times it takes to travel to and from the airport. At the same time, the "value of time" is becoming an important concern that places new demands on current transportation systems. More flexibility and efficiency for door-to-door travel will be needed. Travel to a small or metro-satellite airport has a great potential of being a big market. The migration of people away from urban and suburban centers requires greater access to transportation from more widespread locations throughout the country. Unfortunately, at these smaller airports, viable, cost competitive air transportation is not currently available.

As described above, increasing capacity alone does not appear to provide a long-term solution to the problem of delay, or satisfy the demand for more direct flights. Many people, including licensed pilots, are surprised to learn that there are about 5,400 existing public-use-landing facilities in the current National Airspace System (NAS). However, scheduled air carriers serve only about 660 of these facilities. Moreover, the Federal Aviation Administration (FAA) estimates 98 percent of the U.S. population lives within 20 miles of at least one of these public-use airports [4]. Most of these under utilized public airports do not lie in the existing Air Traffic Control (ATC) radar coverage, nor do they have a control tower. However, they still have instrument approach procedures with which Instrument Flight Rules (IFR) traffic can be operated.

So why not take advantage of those small airports to promote more evenly distributed air traffic and reduce congestion at large hub airports? In addition, this will unburden the stressed-out business traveler living in a near-by community.

A number of candidate technologies have emerged that are aimed at distributing the heavy air traffic at the hub airports. Small Aircraft Transportation System (SATS), the program organized by the National Aeronautics and Space Administration (NASA), has shown great potential. A new aviation system based on SATS technologies would enhance the current transportation capabilities of the nation's small airports, and thus provide some relief to hub airports congestion, particularly in high-density corridors for point-to-point travel [4].

One of the key issues that the SATS research program focuses on achieving is High-Volume Operations (HVO) at airports without control towers or terminal radar facilities, i.e., non-controlled airports [5]. As stated earlier, most of non-controlled airports have instrument approach procedures with which Instrument Flight Rules (IFR) traffic can be operated. However, these airports do not lie in the existing Air Traffic Control (ATC) radar coverage, nor are they equipped with a control tower. In the current ATC system, controllers take responsibility for maintaining safe separations among all aircraft, and sequencing all arriving aircraft into a certain landing order. At the terminal area of non-controlled airports, controllers are no longer in control, and therefore either an air traffic control automation system or a decentralized ATM system can be applied. Regardless of which system is chosen, it is necessary to analyze both options from the technological and operational views.

A. Air Traffic Control Automation System

In general, air traffic control automation consists of two basic functionalities: trajectory analysis and aircraft scheduling. Trajectory analysis provides flight path predictions, and automated aircraft scheduling takes advantage of accurate aircraft trajectories to produce efficient landing sequences [6].

The aircraft landing scheduling problem is concerned with determining landing times on a runway for a sequence of aircraft, such that each aircraft lands within its predetermined landing time window, while satisfying separation criterion between aircraft. Upon entering into the terminal area of an airport, an aircraft is assigned a landing time and a runway. The landing time must lie within a predetermined time window, bounded by an earliest time and a latest time. The aircraft can land at the earliest time if it flies at its maximum airspeed, while it will land at the latest time if it flies at its most fuel-efficient airspeed while also holding for the maximum allowable time [7].

Each aircraft produces its preferred landing time if it flies at its most economical, preferred speed, the cruise speed. If the aircraft is required to slow down, hold, or speed up for separation assurance or other incidental reasons, extra cost will be incurred. In general, this cost will grow as the difference between the assigned landing time and the preferred landing time increases. Figure 1 shows an example of the variation in cost for an aircraft within its landing time window.

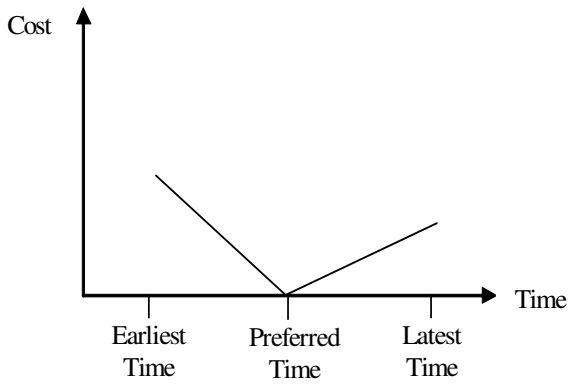

Figure 1: Variation in Cost for an Aircraft within its Landing Time Window I

Another issue is the separation criterion assurance. It is well known that the FAA regulates a certain separation requirement among flights in en-route airspace. Similarly, the landing time of an aircraft and its successive aircraft must be greater than a specified minimum, referred to as

landing separation time. The landing separation time depends on the type of the aircraft, due to aerodynamic considerations. It is straightforward to understand that the landing separation time between large commercial flights is usually greater than the separation time between small General Aviation (GA) flights, as commercial aircraft generate greater wake turbulence than GA aircraft.

In the past decade, researchers at NASA Ames Research Center have been developing an ATC automation tool called Center-TRACON Automation System, or CTAS [8]. CTAS can assist air traffic controllers in both en-route and terminal areas by providing computer-generated flight trajectories. In particular, the Final Approach Spacing Tool (FAST), a component of CTAS, can assist TRACON controllers to efficiently schedule arriving aircraft [9, 10]. However, it is likely to be very costly to install CTAS at non-controlled airports as CTAS is a set of tools designed to help air traffic controllers manage the increasingly complex air traffic flows at large airports. The SATS research program aims to develop a simpler ATC automation tool for the purpose of easier integration with the current National Airspace System (NAS). According to the SATS concept, pilots are required to take responsibility for self-separation within the terminal area of the non-controlled airports. Meanwhile, aircraft scheduling will become the key operational issue at non-controlled airports as the operation volume keeps increasing in the near future. Therefore, the ATC automation system of the non-controlled airport mainly tries to improve the performance of aircraft scheduling.

B. Decentralized Air Traffic Management System

The ATC automation system takes over the sequencing responsibilities of human air traffic controllers. However, installation and maintenance of a ground-based automated system at non-controlled airports can be very costly, even for a simple SATS-based one. More importantly, it still works as a centralized control that places limits on airspace capacity and flight flexibility and provides insufficient operation robustness since it will disorder the landing sequence when it

becomes disfunctional. An alternative is to reduce the reliance on centralized air traffic management in favor of decentralized management. However, the real question is how decentralized should the non-controlled airports be, or more simply how far we can extend the free-flight concept in the case of the non-controlled airports? Can we achieve the decentralized mode for the ATM system of non-controlled airports such that the on-board aircraft landing scheduling tool takes over the sequencing/scheduling job?

The on-board aircraft landing scheduling tool represents uninterrupted free-flight to the threshold, where flight crews take over all of the responsibilities that the current controllers have, and pilots are required to use the on-board scheduling tool to provide sequencing advisories dynamically at the terminal area of the non-controlled airports. Pilots might use the on-board scheduling tool to reschedule themselves constantly during the whole landing operation since a change may occur in the dynamic operational environment, such as the appearance of a new aircraft. This can be referred as the dynamic, or online, Aircraft Landing Problem with respect to the static Aircraft Landing Problem where the static operational environment was assumed. The ground-based aircraft landing scheduling automation tool only makes the scheduling decision once during the whole landing operation at a certain scheduling point with complete knowledge of the set of aircraft that are going to be sequenced. This can be regarded as a static problem since it represents the initial (static) situation.

C. Research Objectives and Contributions

This research aims to analyze the feasibility and capability of the decentralized aircraft landing scheduling operations at non-controlled airports. In general, research on aircraft scheduling can be roughly divided into two areas. One area determines efficient scheduling algorithms, and the other studies performance potentials and overall strategies of automated aircraft scheduling [6]. This research is of the latter.

Firstly, this research seeks to develop static optimization algorithms for aircraft landing scheduling, and analyze the capability of automated aircraft landing scheduling on single runway at non-controlled airports. This is done by establishing an aircraft scheduling model for non-controlled airports, and then developing different scheduling algorithms which are evaluated for capability analysis via flight simulation.

Secondly, this research seeks to analyze the degree of decentralization for aircraft landing scheduling in the dynamic operational environment at non-controlled airports. This is done by firstly developing an aircraft agent with distributed coordination functions, followed by the methodologies description of how to integrate the aircraft landing scheduling model into the flight deck on-board system of an aircraft agent. Then, uninterrupted free-flight aircraft landing operation at non-controlled airports, represented by the on-board aircraft landing scheduling tool, is examined in which the ground-based automated system serves as the baseline system for comparison. Two coordination models, with focus on event-based coordination model, are then developed and related distributed coordination issues are addressed.

The decentralized aircraft landing scheduling problem has not been clearly stated in the literature by far prior to the advent of this research. It is an important problem deserving of closer attention since it provides a clear approach to the distributed air traffic management system. This research is concerned with the non-controlled airport case since current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the analysis of the distributed air traffic management system and the free-flight concept. Its application is expected to extend to controlled airports with suitable amendment, where the reliance on centralized air traffic management can be reduced in favor of a decentralized management to provide more airspace capacity, flight flexibility, and increase operation robustness.

D. Structure of the Dissertation

The following is an outline of the content of the dissertation:

In chapter II, an overview of the current Air Traffic Management system, including operational components and concepts, is provided first. Then the concept of free flight is introduced. Finally, general information of the SATS program and non-controlled airport feature are reviewed.

In chapter III, an approach of developing static optimization algorithms for aircraft landing scheduling at non-controlled airport is given. Modeling and implementation of an Air Traffic Control automation system, which handles the static case for aircraft landing scheduling at non-controlled airport, are addressed. The Air Traffic Control automation system, automatic but still "centralized-like", serves as the baseline system for comparison with decentralized aircraft landing scheduling addressed in chapter IV.

Chapter IV discusses the approach of solving the problem of decentralized aircraft landing scheduling in the dynamic operational environment at non-controlled airport. Modeling and implementation of a decentralized Air Traffic Management system at non-controlled airport, which handles the dynamic aircraft landing, are addressed.

Chapter V presents the overall flight simulation architecture first. Then the numerical solution methodologies for the implementation of the models that were derived in chapter III and IV are provided, followed by the discussion of the numerical results.

In chapter VI, it presents conclusions that could be drawn from the research that was presented in this dissertation. It then gives recommendations on issue for future research and development.

CHAPTER II

UNDERSTANDING THE AIR TRAFFIC MANAGEMENT SYSTEM – PRESENT AND FUTURE

The research addressed in this dissertation aims to analyze the feasibility and capability of the decentralized aircraft landing scheduling operations at non-controlled airports. It is concerned with the non-controlled airport case since current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the analysis of the decentralized Air Traffic Management (ATM) system and the free-flight concept. It is author's believe that an overview of the current ATM system, free flight concept, and non-controlled airport features, which describes the operational environment for this research, is a necessity.

A. Air Traffic Management System

The Air Traffic Management provides services that are in place to assure safe and controlled air travel. In the current ATM system, there are different control stations an aircraft will go through on its way in the National Airspace System (NAS). All these control stations provide services that direct aircraft on the ground and in the air. The primary task of the service is to separate certain aircraft — to prevent them from coming too close to each other horizontally or vertically. Secondary tasks include ensuring orderly and expeditious flow of traffic, such as aircraft landing scheduling, and providing information to pilots, such as weather and navigation information.

1. Air Route Traffic Control Center

Air Route Traffic Control Centers are established primarily to provide air traffic service to aircraft operating on IFR flight plans within controlled airspace, and principally during the en route phase of flight [11]. Each center is responsible for many thousands of square miles of airspace (known as a Flight Information Region) and for the airports within that airspace. Centers control Instrument Flight Rules (IFR) aircraft from the time the aircraft departs an airport or leaves the terminal area's airspace or until the aircraft approaches the airspace controlled by a terminal area or if the airport does not have terminal area control, until the aircraft lands. Centers may also "pick up" aircraft that are airborne and integrate them into the IFR system. These aircraft must, however, remain Visual Flight Rules (VFR) until the Center provides a clearance.

2. Clearance Delivery Control

Clearance delivery is the position that coordinates with the Air Route Traffic Control Center to obtain releases for aircraft. Under normal conditions, this is more or less automatic. When weather or extremely high demand for a certain airport become a factor, there may be ground "stops", the airport may go "IFR", or re-routes to ensure the system does not get overloaded. The primary responsibility of the clearance delivery control is to ensure that the aircraft have the proper route and release time. This information is also coordinated with the Air Route Traffic Control Center and the ground controller in order to ensure the aircraft reaches the runway in time to meet the release time.

3. Ground Control

Ground Control is responsible for the airport "maneuvering" areas, or areas not released to the airlines or other users. This generally includes all taxiways, holding areas, and some transitional intersections where aircraft arrive having vacated the runway and departure gates. Exact areas and control responsibilities are clearly defined in local documents and agreements at each airport. Any aircraft, vehicle, or person walking or working in these areas is required to have clearance from the ground controller.

4. Tower Control

Tower control is responsible for the active runway surfaces. Local control clears aircraft for take off or landing and ensures the runway is clear for these aircraft. To accomplish this, local control controllers are normally given 2 to 5 nautical miles (4 to 9 km) of airspace around the airport, allowing them to give the clearances necessary for airport safety. If the local controller detects any unsafe condition, a landing aircraft will be told to "go around" and will be re-sequenced into the landing pattern by the approach or terminal area controller. Within the tower, a highly disciplined communications process between tower and ground control is an absolute necessity. Ground control must request and gain approval from tower control to cross any runway with any aircraft or vehicle. Likewise, tower control must ensure ground control is aware of any operations that impact the taxiways and must work with the approach radar controllers to ensure "holes" or "gaps" in the arrival traffic are created (where necessary) to allow taxiing traffic to cross runways and to allow departures aircraft to take off.

5. Approach and Terminal Control

Many airports have a radar control facility that is associated with the airport. In most countries, this is referred to as *Approach* or *Terminal Control*; in the U.S., it is often still referred to as a TRACON or **T**erminal **R**adar **A**pproach **CON**trol facility. While every airport varies, terminal controllers usually handle traffic in a 30 to 50 nautical mile (56 to 187 km) radius from the airport and from the surface up to 10,000 feet. Terminal control is responsible for providing all Air Traffic Control (ATC) services within their airspace. Traffic flow is broadly divided into departures, arrivals, overflights, and VFR aircraft. As aircraft move in and out of the terminal airspace, they are handed off to the next appropriate control facility (a control tower, an en-route control facility, or a bordering terminal or approach control). Terminal control is responsible for ensuring that aircraft are at an appropriate altitude when they are handed off, and that aircraft arrive at a suitable rate for landing.

B. Free Flight Concept

Free flight is a developing ATC method that uses no centralized control (e.g. air traffic controllers). Instead, parts of airspace are reserved dynamically and automatically in a decentralized way using computer communication to ensure the required flight operations. It proposed the ATC schemes under which airline companies and pilots would be allowed greater flexibility in choosing paths from one airport to another. Current air traffic control operations channel air traffic along a modest number of fixed routes. Fixed routes minimize the potential for conflict, but produce flight plans that do not minimize fuel usage or flight time. In the highly competitive air carrier environment, airlines are anxious to reduce their fuel costs and increase aircraft utilization. At the extreme, free flight would abolish fixed routes, leaving the flight path completely up to the flight crew (or airline). The abandonment of fixed routes and the introduction of free flight is projected to reduce airline operating cost by allowing airlines or crew to select more fuel-efficient paths with reduced flight times.

One of the principal concerns in the introduction of free flight is the possibility of adverse changes in controller workload. As a result of the fixed routes of the current ATC system there is substantial repeatability from one day to the next. With practice a controller learns the patterns of traffic. This consistency facilitates the cognitive processing required for information acquisition, decision making, and response planning. The fixed route structure further reduces controller workload by limiting the number of crossing locations within a sector where aircraft would be most likely to violate separation. In short, the current fixed route system constrains the opportunities for conflicts, while providing cognitive support to controllers in detecting them.

This predictable structure will change under free flight. For TRACON controllers, the number of paths could approach the number of different airports feeding aircraft to a particular TRACON. For en-route controllers the situation under free flight could be even more complex, since traffic originating from several airports could be crossing a sector en-route to several distinct destination airports. Variations in wind and weather will cause further variations in paths on a daily or even

hourly basis. Thus, where controllers currently have a stable set of routes, free flight will greatly increase path diversity and reduce path predictability.

Free flight would be an empty solution if the cost in fuel savings to the airlines was negated by a significant increase in the cost of air traffic control. Yet any future air traffic control system clearly must maintain the current high level of safety. The challenge for free flight is how to achieve a very high level of safety while avoiding significant increases in staffing levels [12].

C. Small Aircraft Transportation System Project

The Small Aircraft Transportation System (SATS), a program organized by the National Aeronautics and Space Administration (NASA), is a revolutionary project designed to reduce overpopulation of the U.S.'s airport hubs, while increasing traffic at secondary and tertiary airports with advanced small planes, giving greater abilities to smaller communities. Being that ninety-eight percent of Americans live in short distance of a public-use airport, SATS' premise incorporates 5,400 under-utilized airports across the country into a vital tool helping alleviate the current overcrowding situation.

SATS program expected to create a new system of innovative technology in smaller aircraft to be implemented at secondary and tertiary airports, mostly are non-radar and non-towered airports, referred as non-controlled airports. The model for the future will have complete reliance on computer guidance so trips to a relatively close-distanced lake or coastline will be casual, convenient, and easy. Instead of investing millions of dollars per airport to expand and update its technology, the equipment and resources of the largest airports will be put into the plane itself. The abilities of the new planes is not suggesting the elimination of towers altogether, but a technological benefit in many places where the only thing available is a strip of asphalt. Major airports will be operating at normal capacity with their aircraft flying at high altitudes, along with small airports being utilized, flying their craft at lower altitudes, in essence, a complex highway system for the skies and for America.

The SATS research program invested in four operating capabilities: 1) high-volume operations at non-controlled airports; 2) lower adverse weather landing minimums at minimally-equipped landing facilities; 3) integration of SATS into a higher en-route capacity air traffic control system with complex flows and slower aircraft; 4) improved single-pilot ability to function competently in complex airspace in an evolving NAS [5].

D. Non-controlled Airport Features

Non-controlled airports are always surrounded by uncontrolled airspace (class G airspace) with a ceiling of 700 feet or 1200 feet Above Ground Level (AGL). U.S. airspace classification is shown in Figure 2. Within the class G airspace, it is the pilot's responsibility for traffic avoidance, even for Instrument Flight Rules (IFR) traffic. Therefore, current air traffic operations in instrument meteorological conditions (IMC) at non-controlled airports are constrained by the "one-in/one-out" procedure, to ensure safety of the aircraft flying approaches within uncontrolled airspace. In other words, when the airspace around non-controlled airports is occupied by one aircraft flying either an arrival or departure, additional requests for operations at the airport are postponed until the current operation is completed. We can easily tell that capacity at these airports is severely constrained by the "one-in/one-out" paradigm since one operation can take over 15 minutes to complete [13].

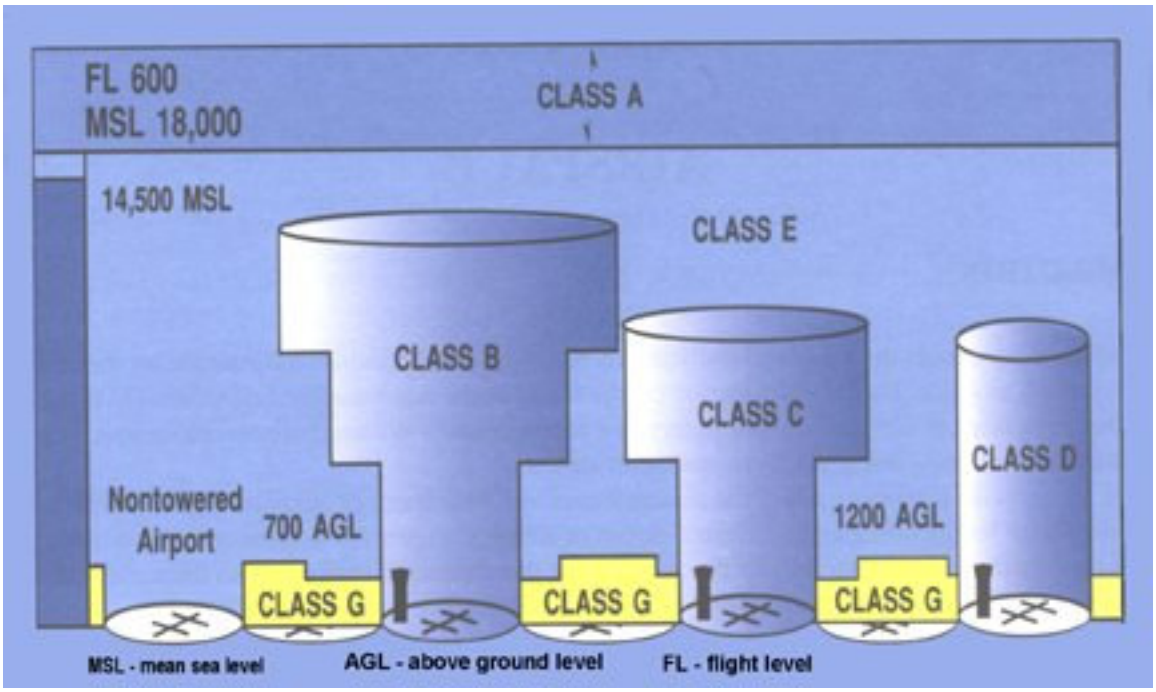

Figure 2: U.S. Airspace Classification

Several research projects researched this paradigm to enable multiple operations simultaneously within the airspace around non-controlled airports. Two of them address implementing multiple operations at Multi-Layer Air Traffic Space (MATS) and Self-Controlled Area (SCA) respectively, which SCA is defined in the SATS project introduced earlier [5, 14]. Each attempts to define a new airspace infrastructure around non-controlled airports, and then develops suitable procedures accordingly.

E. Conclusions

In this chapter, we firstly reviewed the functionalities of the operational components in the current ATM system. Then we gave a general description of the Free Flight concept and the SATS program. They both ultimately seek to achieve effective and efficient flight operations in the current NAS, with the former focusing on en-route flight operations, and the latter concentrating on terminal area operations at non-controlled airports. At this stage the potential

impact of Free flight on the operations of the national airspace system is still disputed, and demand measurement for SATS shows that NASA could possibly introduce an idea to the public that would never be used. However, it can be expected that the future air traffic management system will manage flight operations in a way that lies somewhere between the two extremes, fully centralized and uninterrupted free-flight, possibly moving gradually from centralized to more free-flight, as the concept of Highway-in-the-Sky emerges.

As stated previously, current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the implementation of uninterrupted free-flight concept. In the research addressed in this dissertation, the on-board aircraft landing scheduling tool developed represents uninterrupted free-flight to the threshold since there is no ground-based automated system to enforce any centralized control and flight crews take over all of the responsibilities that the current controllers have. Aircraft at the terminal area of non-controlled airports are placed in a complete decentralized environment, and it is author's believe that the research addressed in this dissertation will contribute to the future ATM revolution since the research will give a clear view of where to establish the line of free-flight concept application for the controlled airport case.

CHAPTER III

AIRCRAFT LANDING SCHEDULING OPTIMIZATION FOR SINGLE RUNWAY NON-CONTROLLED AIRPORTS: STATIC CASE

A. Introduction

In chapter II we gave an overview of the operational components and concepts involved in the aircraft landing operation at terminal area of non-controlled airports. The ultimate goal of the research addressed in this dissertation is to analyze the degree of decentralization for aircraft landing scheduling in the dynamic operational environment at non-controlled airports, and thus explore the feasibility and capability of aircraft landing scheduling within uninterrupted free-flight environment in which there is no existence of Air Traffic Control (ATC). Before we do that, we need to solve the problem of how to effectively and efficiently schedule aircraft landing at non-controlled airports. In this chapter we address the approach of developing static optimization algorithms for aircraft landing scheduling, and thus analyze the capability of automated aircraft landing scheduling at single runway non-controlled airports. This is done by establishing an aircraft landing scheduling model non-controlled airports, and then developing different scheduling algorithms that are evaluated for capability analysis via flight simulation.

The aircraft landing scheduling model and correlated scheduling algorithms are implemented as an air traffic control automation system that takes over the sequencing responsibilities of human air traffic controllers. From the technological view, the air traffic control automation system still utilizes centralized control that places limits on airspace capacity and flight flexibility and provides insufficient operation robustness since it will disorder the landing sequence when it malfunctions. From the operational view, the air traffic control automation system only deals with the static case for aircraft landing scheduling, where the air traffic control automation system only makes the scheduling decision once during the whole landing operation at a certain scheduling point with complete knowledge of the set of aircraft that are going to be sequenced.

Scheduling decision will not be updated dynamically after it is initially determined. However, the resolution for these issues, the distributed Air Traffic Management (ATM) system that handles decentralized aircraft landing scheduling dynamically, is beyond the scope of this chapter and will be addressed in chapter IV.

In this section, previous work that has been reported in the literature on the problem of aircraft landing scheduling is reviewed. It should be stressed that general scheduling problems are the wider problems that occur in the management of air traffic and they are usually examined from the operations research viewpoint. General scheduling problems are most often described using a three-field classification, i.e., machine environment/job characteristics/optimality criterion. The machine environment is always characterized by a string of two parameters that describe the relationship among the operating machines, such as if the operating machines are identical or not. The job characteristics are specified by a set of parameters that describe the relations between jobs, such as preemption and precedence relations. The optimality criterion term then gives the objective function. Some scheduling problems can be solved efficiently by reducing them to well-known combinatorial optimization problems, such as the linear programming problem, maximum flow problem, or transportation problem. Others can be solved by using standard techniques such as dynamic programming and branch-and-bound methods.

Ref. [15-22] give some examples that show how some general scheduling problems are solved by applying different approaches. Most of them deal with the scheduling problem with the three-field form of (single or identical parallel machine)/ (sequence-dependent jobs)/(minimize the total completion of time). Whereas Ref. [15-19] simplify it to some conventional combinatorial optimization problem (e.g., Ref. [17] transforms it to a linear programming problem and Ref. [15] transforms it to an integer programming problem). Ref [20-22] solves the scheduling problem using dynamic programming or branch-and-bound approaches.

As stated in previous section of this chapter, the aircraft landing scheduling problem is usually considered as an application in the field of operation research. There are two basic typical problem statements to describe the aircraft landing scheduling problem: linear/integer programming problem and the job shop scheduling problem. The Aircraft landing scheduling problem is described as an linear/integer programming problem by putting the separation requirements of all pairs of aircraft and landing window constraints of each aircraft into the standard linear/integer programming form. On the other hand, the problem of scheduling aircraft landings on one or more runways is the problem that is similar to a machine job scheduling problem with sequence-dependent processing times and with earliness and tardiness penalties.

1. Job Shop Scheduling Approach

An instance of a typical job shop scheduling problem consists of a set of n jobs and m machines. Each job consists of a chain of operation, and each operation needs to be processed during an uninterrupted period of time on a single machine for its entire duration. The objective is to find a schedule that minimizes the overall completion time of all the operations. The aircraft landing scheduling problem can be modeled as a job shop scheduling problem when runways represents machines, and aircraft landing operations represent jobs. Branch-and-bound algorithm and tree search algorithm are commonly applied to solve the job shop scheduling problem. Ref. [23] discusses both the static and dynamic aircraft landing problems and present a heuristic algorithm for the dynamic aircraft landing problem based upon a technique they refer to as *constrained position shifting*. This involves finding the best possible positions for the aircraft in the landing queue subject to the constraint that no aircraft can be moved more than a pre-specified number of positions away from the position it had in the landing queue based on First-come-first-serve. However, the separation constraint only applies to successive aircraft landings.

Ref. [24] aims to optimally land a set of aircraft on one or several runways in such a way that separation criteria between all pairs of aircraft (not just successive ones) are satisfied. It presents a specialized *simplex algorithm* which evaluates the landing times very rapidly, based on some partial ordering information. This method is then used in a problem space search heuristic as well as a *branch-and-bound method* for both single and multiple runway problems.

Ref. [25] presents a mixed-integer zero-one formulation of the aircraft landing scheduling problem together with a *tree search algorithm* based upon a Lagrangean lower bound, a lower bound derived from scheduling theory and a heuristic procedure.

Ref. [26] incorporates *constrained position shifting* within *dynamic programming recursion* (with successive separation) and considers the single runway static problem. It views the aircraft landing problem as comprising groups of identical aircraft waiting to land.

Ref. [27] presents a *depth-first tree search algorithm* based on enumerating all possible aircraft sequences. Branches in the tree were discarded when the cost of a partially constructed sequence exceeds the best-known feasible solution (*branch-and-bound method*).

2. Combinatorial Optimization Approach

Aircraft landing scheduling problem can also be solved efficiently by reducing them to well-known combinatorial optimization problems, such as linear programs, maximum flow problems, or transportation problems. Ref. [7] and Ref. [28] present a mixed-integer zero-one formulation for both the static and dynamic aircraft landing problems. They strengthen the linear programming relaxations of these formulations by introducing additional constraints. The problem is solved optimally using a *linear programming-based tree search algorithm.* Ref. 26 uses a similar approach and considers the problem of assigning priorities to aircraft waiting to land from a queuing theory viewpoint.

3. Knowledge-Based/Fuzzy Reasoning Approach

Final Approach Spacing Tool (FAST), a component of CTAS, has been developed to assist TRACON controllers to efficiently schedule arriving aircraft. Papers on FAST provide the

knowledge-based approach for the aircraft landing scheduling problem [29-33]. They propose a fuzzy reasoning-based method for both aircraft sequencing in the terminal area and runway assignment. The scheduling system sequences and assigns landing times to arrival aircraft by utilizing continuous updates of aircraft radar data and controller inputs. The scheduling algorithm contains a knowledge base which was refined during several thousand hours of controller-in-the-loop real-time simulations. The knowledge base then applies fuzzy reasoning to evaluate propositions that consider both performance criteria and workload criteria, such as delay reduction and conflict avoidance.

It should be noted that the vast majority of non-controlled airports only have instrument approaches designed for the primary runway. Even for those airfields that do not have ATC service but do have multiple runways, in bad weather there is almost always only one runway that instrument approaches are flown to. Since the number of non-controlled airports which are capable of multiple runway operations is estimated to be only about 1%, only the single runway case is considered here.

This chapter is organized as follows. First, the basic problem of aircraft landing scheduling is described. Then an aircraft landing scheduling model of the static case for single runways at non-controlled airports is established. Finally, methodologies and capabilities of developing and implementing different scheduling algorithms are presented.

B. Aircraft Landing Scheduling Problem

The aircraft landing scheduling problem is concerned with determining landing times on a runway for a sequence of aircraft, such that each aircraft lands within its predetermined landing time window, while satisfying separation criterion between aircraft. Upon entering into the terminal area of an airport, an aircraft is assigned a landing time and a runway. The landing time must lie within a predetermined time window, bounded by an earliest time and a latest time. The aircraft can land at the earliest time if it flies at its maximum airspeed, while it will land at the

latest time if it flies at its most fuel-efficient airspeed while also holding for the maximum allowable time [7].

Each aircraft produces its preferred landing time if it flies at its most economical, preferred speed, the cruise speed. If the aircraft is required to slow down, hold, or speed up for separation assurance or other incidental reasons, extra cost will be incurred. In general, this cost will grow as the difference between the assigned landing time and the preferred landing time increases. Figure 3 shows an example of the variation in cost for an aircraft within its landing time window.

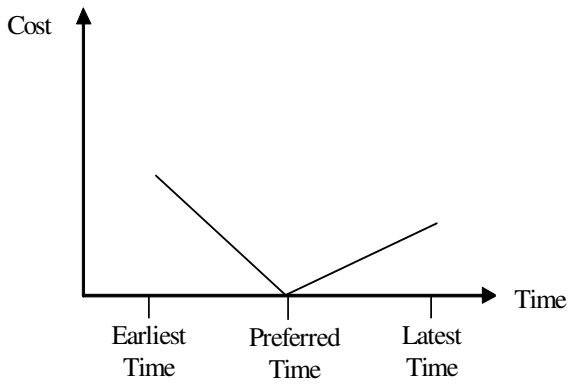

Figure 3: Variation in Cost for an Aircraft within its Landing Time Window II

Another issue is the separation criterion assurance. It is well known that the FAA regulates a certain separation requirement among flights in en-route airspace. Similarly, the landing time of an aircraft and its successive aircraft must be greater than a specified minimum, referred to as landing separation time. The landing separation time depends on the type of the aircraft, due to aerodynamic considerations. It is straightforward to understand that the landing separation time between large commercial flights is usually greater than the separation time between small General Aviation (GA) flights, as commercial aircraft generate greater wake turbulence than GA aircraft.

Research on aircraft scheduling can be divided into two areas. One area determines efficient scheduling algorithms, and the other studies performance potentials and overall strategies of automated aircraft scheduling [6]. The aircraft landing scheduling problem is usually considered

as an application in the field of operations research, and two approaches are often taken: linear/integer programming (LP), and job shop scheduling. The aircraft landing scheduling problem is described as an LP problem by representing the separation requirements of all pairs of aircraft and the landing window constraints of each aircraft in the standard linear/integer programming form [7, 26]. The basic mathematical description of LP is introduced in a later section. The problem of scheduling aircraft landings on one or more runways is a problem that is also similar to a machine job scheduling problem with sequence-dependent processing times, and earliness and tardiness penalties. A typical job shop scheduling problem consists of a set of n jobs and m machines, in which the objective is to find a schedule that minimizes the overall completion time of all the operations. The aircraft landing scheduling problem can be modeled as a job shop scheduling problem when runways represents machines, and aircraft landing operations represent jobs [23, 25]. The basic mathematical description of job shop scheduling algorithms is introduced in a later section, and both the LP and job shop scheduling approaches are used in this research.

C. Aircraft Landing Scheduling Model for Single Runway Non-Controlled Airports

1. Notations

Constants used to describe the aircraft landing scheduling model are defined as follows:

E_i = earliest landing time for aircraft i (i = 1,… N)

f_i = penalty cost per unit of time if aircraft i lands before the preferred landing time P_i (i = 1,… N)

g_i = penalty cost per unit of time if aircraft i lands after the preferred landing time P_i (i = 1,… N)

L_i = latest landing time for aircraft i (i = 1,… N)

N = number of aircraft

P_i = preferred landing time for aircraft i (i = 1,… N)

S_{ij} = separation time requirement between aircraft i and j, where aircraft i lands before j ($i = 1,\ldots N, j = 1,\ldots N, i \neq j$)

The landing time window of aircraft i is therefore denoted as [Ei, Li], where Ei ≤ Pi ≤ Li.

Variables used to describe the aircraft landing scheduling model are defined as follows:

a_i = time aircraft i lands after the preferred landing time P_i ($i = 1,\ldots N$)

b_i = time aircraft i lands before the preferred landing time P_i ($i = 1,\ldots N$)

x_i = landing time for aircraft i

δ_{ij} = 1 if aircraft i lands before aircraft j, and 0 otherwise ($i = 1,\ldots N, j = 1,\ldots N, i \neq j$)

2. Constraints

Figure 4 is a graphical representation of the constraints developed below. It shows the overlapping landing time windows of aircraft i and j. The first set of constraints are:

$$E_i \leq x_i \leq L_i, \qquad i = 1,\ldots N \tag{1}$$

which ensures that each aircraft must land within its predetermined landing time window. Now, considering pairs of aircraft (i, j) provides another constraint:

$$\delta_{ij} + \delta_{ji} = 1, \qquad i = 1,\ldots N; j = 1,\ldots N; i < j \tag{2}$$

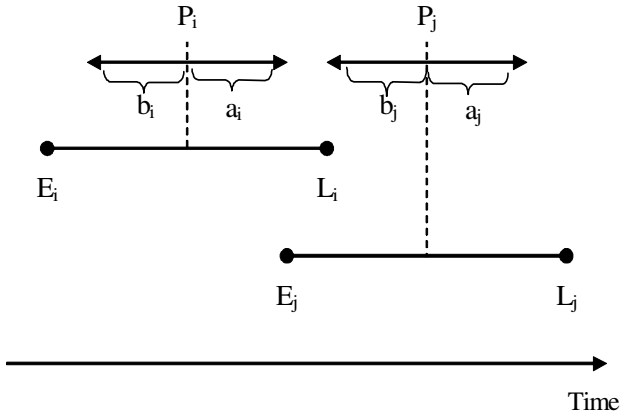

Figure 4: Example of Overlapping Landing Time Windows

In words, either aircraft i must land before aircraft j ($\delta_{ij} = 1$) or aircraft j must land before aircraft i ($\delta_{ji} = 1$). The constraints described above mainly deal with the order of pairs of aircraft (i, j).

However, even if the landing order of pairs of aircraft (i, j) is known, it does not necessarily imply the separation constraints are automatically satisfied. For example, if the landing time window for two aircraft i and j are (10, 20) and (30, 40) respectively, and the separation time S_{ij} =15, the separation constraint is not automatically satisfied; there exist landing times for i and j that violate the separation constraint. Hence, the separation constraint is necessarily defined as:

$$x_i + S_{ij}\,\delta_{ij} - (L_i - E_j)\,\delta_{ji} \le x_j$$

$$i = 1,\ldots N;\ j = 1,\ldots N;\ i \ne j \qquad (3)$$

Two cases are considered here. First, if $\delta_{ij} = 1$, then i lands before j and thus $\delta_{ji} = 0$ from Eq.(2). Therefore, inequality (3) becomes $x_j \ge x_i + S_{ij}$, ensuring that the separation requirement is satisfied. Second, if $\delta_{ij} = 0$, then j lands before i and thus $\delta_{ji} = 1$ from Eq.(2). Therefore, inequality (3) becomes $x_i - x_j \le L_i - E_j$, and it can be deduced easily starting from the inequality (1). Finally, the constraints which relate a_i, b_i and x_i variables are:

$$\text{Max}\,(0, P_i - x_i) \le b_i \le P_i - E_i, \qquad i = 1,\ldots N \qquad (4)$$

$$\text{Max}\,(0, x_i - P_i) \le a_i \le L_i - P_i, \qquad i = 1,\ldots N \qquad (5)$$

$$x_i = P_i - b_i + a_i, \qquad i = 1,\ldots N \qquad (6)$$

Inequality (4) ensures that b_i is at least as big as zero, and the time difference between P_i and x_i, and at most the time difference between P_i and E_i (see Figure 4). Inequality (5) shows the similar meaning for a_i. Equation (6) relates the a_i, b_i and x_i variables to the preferred landing time P_i. It should be noted that for the i^{th} aircraft in a specified scenario, at least one of the values among a_i and b_i should be equal to zero.

It is worthy mentioning that the constraints described above can be considered as a non-controlled airport application customization of the constraint set developed in Ref. 14, and the author does not claim any credit of original contribution of developing the constraint set. The main purpose of constraint customization is to formulate the aircraft landing scheduling model, and thus to develop the air traffic control automation system that will be served as the baseline

system for comparison with the distributed air traffic management system developed in chapter IV.

3. Problem Specific Features of Non-Controlled Airports

Non-controlled airports are always surrounded by uncontrolled airspace (class G airspace) with a ceiling of 700 feet or 1100 feet Above Ground Level (AGL). Within the class G airspace, it is the pilot's responsibility for traffic avoidance, even for Instrument Flight Rules (IFR) traffic. Therefore, current air traffic operations in instrument meteorological conditions (IMC) at non-controlled airports are constrained by the "one-in/one-out" procedure, to ensure safety of the aircraft flying approaches within uncontrolled airspace. In other words, when the airspace around non-controlled airports is occupied by one aircraft flying either an arrival or departure, additional requests for operations at the airport are postponed until the current operation is completed. We can easily tell that capacity at these airports is severely constrained by the "one-in/one-out" paradigm since one operation can take over 15 minutes to complete [13].

Several research projects are currently researching this current paradigm to enable multiple operations simultaneously within the airspace around non-controlled airports. Two of them address implementing multiple operations at Multi-Layer Air Traffic Space (MATS) and Self-Controlled Area (SCA) respectively [5, 14]. Each attempts to define a new airspace infrastructure around non-controlled airports, and then develops suitable procedures accordingly. Since we will use Small Aircraft Transportation System (SATS) SCA High Volume Operation (HVO) scenarios to analyze our aircraft landing scheduling algorithms in the following sections of the paper, a brief description of the SATS SCA HVO program is provided. SCA is a block of airspace established around non-controlled airports where the responsibility for safe aircraft separation lies with the pilot. The SCA HVO program enables multiple operations within an SCA by having the aircraft hold in stacks at Initial Approach Fixes, and then follow specified procedures (either vertical entry or lateral entry) to enter the SCA and complete approaches. This is shown in Figure 5.

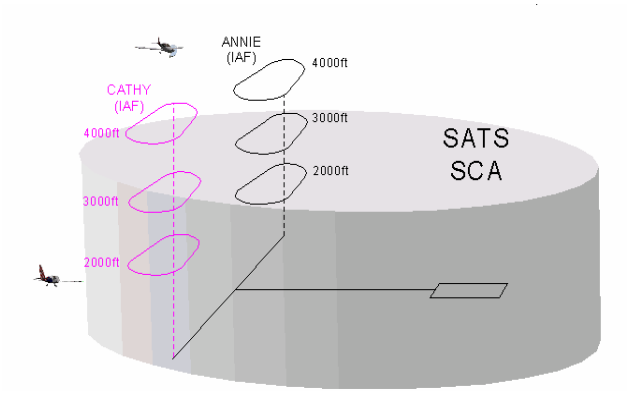

Figure 5: SATS Self-Controlled Area High-Volume Operations Concept

It should be noted that either vertical or lateral entry landing operation can be divided into a sequence of flight segments. Aircraft at one flight segment is required to determine its action towards the subsequent flight segment depending on its availability. For example, an aircraft is at 4000ft AGL above the SCA and it plans to fly a vertical entry procedure, the subsequent flight segment is the holding pattern at 3000ft AGL (the ceiling of the SCA). Before it approaches to the holding pattern at 3000ft AGL, it will have to check its availability by determining if it is occupied by another aircraft or not. It will not be allowed to descend to the 3000ft AGL holding pattern until it is clear. Figure 6 shows the decision tree along the flight path for an aircraft in a SCA HVO scenario.

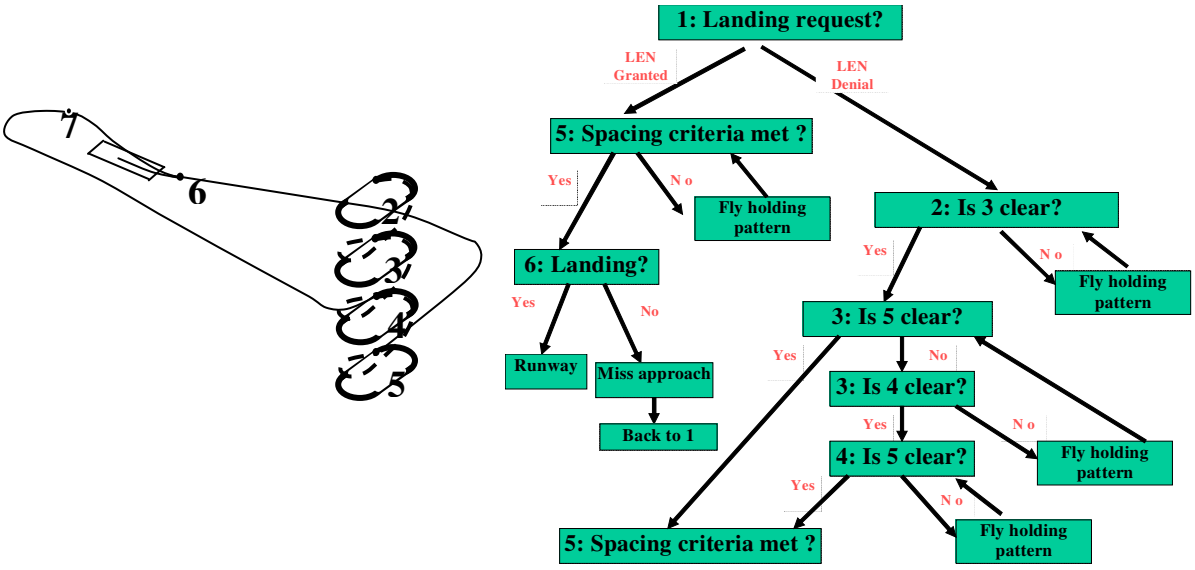

Figure 6: Decision Tree along Flight Path for SCA HVO Procedure

4. Objective Functions and Performance Metrics

In the field of aircraft scheduling, the choice of objective function has usually caused the most discussion. Different users can always provide convincing arguments for their choices of objective function. Two different objective functions are defined based on the special features of the SCA HVO described above, and are used as two of the three performance metrics used in the numerical examples. A third performance metric which is not an objective function is also introduced.

1) Minimize Total Cost of Deviation (TCD) from the preferred landing time. This performance metric indirectly measures the sequencing/scheduling algorithm efficiency with regard to the complete flight path. TCD is the sum of weighted Dynamic Time of Arrival (DTA) deviation of all the aircraft in a scenario. Deviation time is calculated as the difference between DTA and the preferred landing time. Preferred landing time is calculated as the flight time from the scenario entry point to the runway, assuming the constant airspeed of the holding speed. The penalty cost depends on the aircraft type.

$$Minimize \sum_{i=1}^{N} (f_i b_i + g_i a_i) \tag{7}$$

2) Minimize Total Holding Time (THT). This performance metric indirectly measures the fuel and efficiency affected by the sequencing/scheduling algorithm, with special consideration of the SATS HVO scenario at non-controlled airports. It is the sum of delay time of all the aircraft in a scenario during the flight segment of three SATS holding patterns (2000ft, 3000 ft, and 4000 feet respectively). The holding time for one aircraft is the time from when it arrives at the IAF until it can begin its approach; the total holding time is the sum of the holding times for all aircraft in the scenario. In the context of SCA HVO procedures total holding time measures the total "wasted time" of all aircraft while flying holding patterns above or within the SCA and is thus a suitable metric for evaluating the efficiency of the aircraft landing scheduling algorithms for SCA HVO. Note that the total holding time cannot

be explicitly defined using the constants and variables given in the previous sections of the paper; it can only be calculated while running real-time flight simulation.

3) Total Delay Time of Feeder Route. This performance metric directly measures the delay time with regard to the flight segment of the feeder route. It is not an objective function, but is introduced here as a supplementary metric to evaluate flight efficiency, since the delay encountered from the scenario entry point to the IAF is distinguishable from the holding delay measured in Total Holding Time. It is the sum of the delay time of all aircraft in a scenario during the feeder route flight segment (from the scenario entry point to IAF), and is a result of implementing the prescribed sequence.

Note that similar aircraft landing scheduling problems with objective functions that are linear functions of time deviation from preferred landing time can be solved without any substantial change to the current formulation, simply by introducing new penalty cost parameters. However, objective functions that are nonlinear functions of time deviation from preferred landing time are beyond the scope of the present research.

The complete aircraft landing scheduling model of the single runway case for non-controlled airports is now established—minimize either total cost of deviation or total holding time subject to equations and inequalities (1) to (7).

5. Scheduling Point

The scheduler makes its scheduling decision at a certain scheduling point, ideally an actual point in time or space. For example, it can be the origination airport from which an aircraft departs. It can also be the boundary of Center airspace, and thus depends on the definition of the airspace infrastructure around the non-controlled airports. For instance, if the airspace infrastructure defined in Ref [14] is applied, the boundary of the terminal area is approximately 50 nautical miles away from the airports. In this research, since we use the SATS HVO scenarios to conduct scheduling algorithm analysis, the scheduling point is chosen as the time when the first

aircraft in each scenario reaches the waypoint that is 20 nautical miles to its assigned Initial Approach Fix.

D. Aircraft Landing Scheduling Algorithms for Single Runway Non-Controlled Airports

1. First-Come-First-Serve Scheduling Algorithm

The first-come-first-serve scheduling algorithm is the baseline algorithm for comparing with the optimal scheduling algorithm. There is no specified objective function in the first-come-first-serve scheduling model. Instead, the first-come-first-serve paradigm is applied as long as the constraints described in (1) to (6) are satisfied. The first aircraft in the scenario who hits the scheduling point gets the first slot in the sequence, and the aircraft that is nearest to the scheduling point (according to the time-based linear projection) gets the second slot in the sequence, and so on. It should be mentioned that procedure-based projection is used to calculate the time from the scheduling point until the aircraft lands, since the aircraft has to fly specified procedures defined in SCA HVO to complete an approach. For instance, an aircraft has to fly the holding pattern in stacks at Initial Approach Fixes before it enters into the SCA vertically.

2. Optimal Scheduling Algorithm

The optimal aircraft landing scheduling problem can be stated from different viewpoints and then solved using different algorithms. Therefore, a variety of algorithms has been developed in the literature on the problem of scheduling aircraft landing optimally. In this research, it is solved as a linear programming problem and as a job shop scheduling problem.

1) Linear Programming Problem

In mathematics, linear programming problems are optimization problems in which the objective function and the constraints are all linear. It is usually in the following form:

$$Minimize\ z(x) = c_1 x_1 + ... + c_n x_n \tag{8}$$

$$subject\ to$$

$$a_{11} x_1 + ... + a_{1n} x_n \geq b_1$$

$$\vdots$$

$$a_{1m} x_1 + ... + a_{mn} x_n \geq b_m$$

$$x_i \geq 0\ \ for\ i = 1, ..., n. \tag{9}$$

Or in the vector form:

$$Minimize\ Z(X) = C^T X \tag{10}$$

$$subject\ to\ \ AX \geq B \tag{11}$$

where *Z(X)* defines the objective function and $AX \geq B$ defines the constraints set.

A vector X = (x_1, …, x_n) satisfying the inequality set (11) is called a feasible solution. The linear programming problem is to find a feasible solution that minimizes (10). The most popular method for solving linear programming problem is the simplex algorithm [34]. It is an iterative procedure that finds an optimal solution or detects infeasibility after a finite number of steps. Although it should be noted that the number of iteration steps might be exponential, the simplex algorithm is very efficient in practice. Figure 7 shows the flow chart of the simplex algorithm.

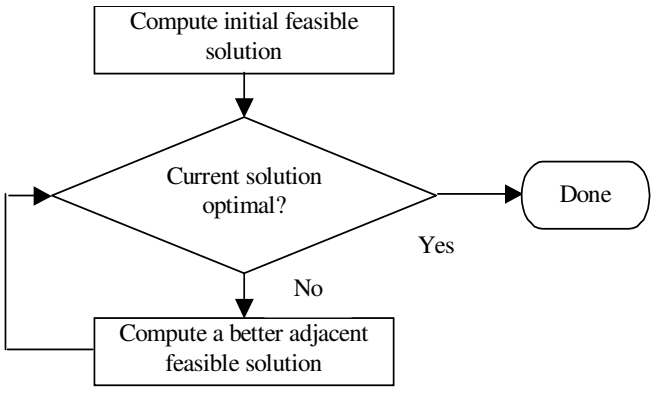

Figure 7: Flow Chart of Simplex Algorithm

It can be seen from Figure 7 that the simplex algorithm is similar to a tree search algorithm based on enumerating the possible solution. The current solution being searched is discarded when the cost exceeds the best-known feasible solution. The aircraft landing scheduling model developed in the previous section can be formulated as an LP problem.

2) Job Shop Scheduling Problem

The aircraft landing scheduling problem can be posed as a job shop scheduling problem in which the single runway represents the single machine, and the aircraft represents jobs. The earliest time E_i associated with each aircraft (job) is the release time (sometimes called the ready time) of the job. The processing time of a particular job (aircraft) on the single machine (runway) is then dependent upon all the other jobs that will follow it on the same machine. This is because the processing time for a particular job on a particular machine must be sufficient to ensure that the following jobs (aircraft) are not started before the appropriate separation time (e.g., S_{ij} in the scheduling model) has elapsed. In other words, the problem of scheduling aircraft landing can be viewed as a job shop scheduling problem with release time E_i and sequence-dependent processing times S_{ij}. In particular, the scheduling model described earlier can be solved as a job shop scheduling problem in the form of:

$$1/\ E_i\ /\text{seq-dep}/\sum w_i c_i \tag{12}$$

Where Ei indicates the release time (the earliest landing time), seq-dep includes all of the separation time requirement Sij, $\sum$wici represents the objective function defined in Eqs. (7), and there is a single runway. Several algorithms are available to solve the job shop scheduling problem, e.g., branch-and-bound algorithm and tree search algorithm.

E. Conclusions

In this chapter an aircraft landing scheduling model and static optimization scheduling algorithms using linear programming and job shop solutions were developed and implemented as an air traffic control automation system for automated aircraft landing scheduling at single

runway, non-controlled airports. Performance of the optimization algorithm was compared to a first-come-first-served scheduling algorithm in terms of total cost of deviation, total holding time, and total delay time of feeder route.

CHAPTER IV

DECENTRALIZED AIRCRAFT LANDING SCHEDULING AT SINGLE RUNWAY NON-CONTROLLED AIRPORTS: DYNAMIC CASE

A. Introduction

In chapter III we examined the capability of automated optimized aircraft landing scheduling at single runway non-controlled airports. An aircraft landing scheduling model and several scheduling algorithms for single runway non-controlled airports were developed and implemented as an air traffic control automation system that takes over the sequencing responsibilities of human air traffic controllers [35]. However, the ultimate objective for this research is to analyze the feasibility and capability of decentralized aircraft landing scheduling in the dynamic operational environment at non-controlled airports in which there is no existence of Air Traffic Control (ATC). This chapter will then attempt to provide detailed description of the implementation of a distributed Air Traffic Management (ATM) system that achieves decentralized aircraft landing scheduling with acceptable performance, and to address all correlated issues.

From the technological view, the air traffic control automation system addressed in chapter III utilizes centralized control as opposed to the decentralized control applied in the distributed ATM system. A significant research issue is the level to which the non-controlled airports should be decentralized. The air traffic control automation system does not provide sufficient operational robustness since it can disorder the landing sequence if it malfunctions. Distributed ATM system, however, is to reduce the reliance on centralized air traffic management in favor of decentralized management. The real question is how decentralized should the non-controlled airports be, or more simply how far can we extend the free-flight concept in the case of the non-controlled airports? Can we achieve the decentralized mode for the ATM system of non-controlled airports such that flight crews even take over the scheduling responsibilities with the assistance of on-

board aircraft landing scheduling tool, which represents the ultimate uninterrupted free-flight scenario? This chapter seeks to address the technology challenges raised by these questions.

From the operational view, the air traffic control automation system addressed in chapter III only deals with the static case for aircraft landing scheduling, where the air traffic control automation system only makes the scheduling decision once during the whole landing operation at a certain scheduling point with complete knowledge of the set of aircraft that are going to be sequenced. Once initially determined, the scheduling decision is not updated dynamically. It is therefore not a total solution, since it still places limits on airspace capacity and flight flexibility. The distributed ATM system, on the other hand, performs the aircraft landing scheduling dynamically since flight crews can use the on-board scheduling tool to reschedule themselves constantly during the whole landing operation since a change may occur in the dynamic operational environment, such as the appearance of a new aircraft. What is not fully understood is whether or not flight crews can adequately assume the scheduling responsibilities with the assistance of on-board aircraft landing scheduling tool in such a decentralized ATM system mode. Decentralized aircraft scheduling is most often addressed in the literature by work on Distributed Air/Ground-Traffic Management (DAG-TM) and free-flight concepts. From the operational view, these works can be divided into four categories: on-board decision support tools, small airport automation systems, air-ground integration/simulations, and coordination. Each of these categories is described below.

1. On-Board Decision Support Tool

The free-flight concept puts more responsibilities on the pilot. In the free-flight environment, pilots are required to ensure their own separation in the en-route airspace, and even in the high-density terminal area. It becomes vital to improve the on-board system to assist pilots with more accurate and efficient flight operations advisories.

One potential solution is presented in [36], which is a sub-project of the DAG-TM program conducted by NASA Ames Research Center. An airborne tool called the Advanced Terminal

Area Approach Spacing (ATAAS) tool is based on the concept of an aircraft maintaining a time-based, rather than distance-based, spacing interval from the preceding aircraft. The ATAAS tool uses Automatic Dependent Surveillance-Broadcast (ADS-B) aircraft state data along with final approach speeds and wind data to compute speed commands for the ATAAS equipped aircraft to maintain the required time interval behind the other aircraft.

Ref. [37-39] address a more advanced on-board guidance system called Airborne Merging and Spacing for Terminal Arrivals (AMSTAR). Ref. [37] presents the general information, Ref. [38] introduces the speed control law, and Ref. [39] describes the simulation environment. It is actually the direct descendant of the ATAAS concept and implementation. Whereas ATAAS was intended for use only when the lead and following aircraft were in-trail, AMSTAR would permit time-based spacing between any two aircraft headed for the same runway, even if they were not yet physically in-trail.

It is important to note all existing on-board decision support tools describe in the literate review above are focused on separation assurance and self-spacing, and leave the responsibilities of aircraft landing scheduling to ground-based air traffic control. The DAG-TM concepts envision a possible way to manage arrivals such that the flight deck is responsible for landing scheduling in the uninterrupted free-flight environment, but the feasibility and capability analysis of an on-board landing scheduling tool has not been explored. The research described in this chapter seeks to fill this technical gap and to address the technological and operational challenges brought by the introductory of on-board landing scheduling tool.

2. Small Airport Automation

Several government-sponsored projects targeted at developing small airport automation include:

- Small Aircraft Transportation System (SATS)
- Smart Airport Automation System (SAASY)

Ref. [40] gives the overall introduction of the Smart Airport Automation System project. SAASY was a NASA sponsored project in the early twenty-first century to provide Visual Flight Rules (VFR) to a flight airport, sequencing, runway, and conflict advisories for GA pilots operating at airports that do not have operating air traffic control towers. The main motivation for SAASY is to provide the mechanism to increase VFR safety**.** The sequence advisory provides conflict-free arrival and departure sequence assignments to enhance traffic safety. An initial aircraft sequence is generated based on a first-come-first-serve algorithm that uses the time-at-runway threshold generated by the trajectory predictor. The final sequence is modified using a set of priority and right-of-way rules to better emulate the pilot's decision process when approaching the airport.

Ref. [41] presents the general information of the SATS project. SATS was conducted through a public-private partnership including NASA, the FAA, and the National Consortium for Aviation Mobility SATSLabs. Within the Self-Controlled Area, properly-equipped SATS aircraft equipped with Automatic Dependent Surveillance-Broadcasting (ADS-B) surveillance, two-way data links, automation-pilot data link communications interfaces, and cockpit displays of traffic information provide self-separation. The sequencing of arrivals is handled by the Airport Management Module (AMM), a small airport system that monitors local traffic. The SATS small airport AMM system focuses on providing IFR capacity and not VFR safety.

Ref. [42] explores past and ongoing research focused on the development of automation to support safer and more efficient aircraft operations at the small airports. The current and expected future operational problems for aircraft utilization of these airports are reviewed first.

Then, the topology of small airport system concepts in general is reviewed, and finally the major issues that drive the system design and operational usage are discussed.

Both SATS and SAASY projects seek to improve the operation effectiveness and efficiency of the under-utilized public-use airports, focusing on IFR capacity and VFR safety respectively. However, the first-come-first-serve sequencing algorithm proposed in SAASY project still places capacity limitations, and the priority rules introduced only apply to VFR traffic. SATS project proposes a new set of operation concepts that break the "one-in-one-out" paradigm and thus provide the capability of multiple operations at a time at the terminal area of small airports. The new set of operation concepts, combined with the on-board SATS Conflict Detection & Avoidance (CD&A) algorithm, provides the separation assurance but leaves the sequencing responsibility to AMM, a ground-based airport automation system which still works as a centralized control. It is therefore concluded that the previous research work on small airport automation still leaves the capability analysis of scheduling IFR traffic in a complete distributed environment as an open discussion.

3. Air-Ground Integration/Simulation

How to integrate either the autonomous airborne components or small airport automation systems into the current NAS is another major question related to the research issues addressed in this chapter. Ref. [43-45]attempt to answer that question from their own perspectives. Ref. [43] describes the procedures of how to evaluate the technical and operational feasibilities of the autonomous airborne components of DAG-TM, based on the free-flight concept. It includes an overview of research approaches, the airborne technologies under development, and a summary of experimental investigations and findings to date. In Ref. [44], results and observations from integrated air ground simulations that were conducted over the last few years at NASA Ames Research Center are reviewed. Ref. [45] addresses a mixed-fidelity simulation environment for a human-in-the-loop study of DAG-TM concepts. Decision support tools for flight crews and air traffic service providers are accessible at the respective operator stations in the simulation

infrastructure. Many operator positions, either operated with human-in-the-loop or autonomously run with agent support, are provided to facilitate large-scale experiments supporting high numbers of pilot, air traffic controller, and air traffic service providers. This simulation environment provides a highly realistic and flexible test bed to gain a solid understanding of interactions in the very complex distributed air traffic environment.

The scheduling/sequencing function component in the Air-Ground integration simulation described above are still simulated as a centralized control component, either implemented as a subset of the ground-based airport automation system agent or directly included human air traffic controller in the loop. Simulation environment that includes disturbed scheduling/sequencing function component has not been developed.

4. Coordination

The design of coordination mechanisms for multi-agent system has proven to be a difficult problem. In the last decade a variety of such mechanisms over a wide range of task domains has been studied. Although the literature highlights some elegant solutions, they are generally domain-specific and provide only indirect insight into important questions. For instance, how appropriate is a given coordination mechanism is for a particular domain? What performance characteristics can be expected, and how is it related to other coordination mechanisms? How can it be modified to improve system performance [46]. A general solution to the distributed coordination problem for multi-agent system is beyond the scope of this research, since the objective here is to propose and develop a coordination mechanism that fits a specific domain: distributed aircraft landing scheduling at non-controlled airports. Two candidate approaches, direct coordination model and event-based model, are favorable. Direct coordination models used in distributed applications are usually the ones in which the agent's interactions involve explicit task-directed communications or negotiations about global resource usage or task assignments in order to achieve coordinated behavior [46]. In event-based coordination models, each agent operates under local control. The system-level coordinated behavior arises from

agent-agent interactions by generating events, and reacting to the particular events of interest, without an explicit notion of task-directed communication or negotiation. Event-based coordination in a distributed system is dominated by client/server platform relying on synchronous request/reply. However, this architecture is not well suited to implement information-driven applications like air traffic control, news delivery, and stock quoting due to the inherent mismatch between the demands of these applications and the characteristics of those platforms. In contrast to that, publish/subscribe (pub/sub) directly reflects the intrinsic behavior of information-driven applications because communication here is indirect and initiated by producers of information: producers publish notifications and these are delivered to subscribed consumers by the help of a notification service that decouples the producers and the consumers [47]. Though pub/sub is not a recent achievement [48, 49], its use in large-scale, wide-area communication has become a hot research topic only in the last a few years, making pub/sub move from a simple application of multicast to a communication paradigm in its own right. This happened because the anonymous, loosely coupled communication scheme, which is proper of the pub/sub paradigm, fits well to the highly dynamic nature of large-scale environments. Therefore, publish/subscribe should be the first choice for implementing such applications.

All of the previous work reported in the literature addresses on-board decision support tools for airborne separation and final approach spacing at controlled airports, and all of the landing scheduling tools are ground-based. In the domain of aircraft landing scheduling, landing operations at terminal area of the airports are still under centralized-control implemented by ground-based airport automation system or human air traffic controller. A systematic approach to achieve scheduling multiple landing operations at a time in a decentralized dynamic operational environment has not been explored. Firstly the infrastructure of the decentralized operational environment, which includes the functional components and operation concepts, needs to be established. Secondly, decentralized ATM mode certainly introduces more dynamics to the operational environment due to the increasing interaction among aircraft, and issues of handling

dynamic situation such as the appearance of a new aircraft need to be addressed. Finally, a communication model needs to be developed to ensure robust and effective interaction among aircraft during decentralized aircraft landing scheduling operations. The contributions of the research addressed in this chapter lie in the development of an event-based coordination pub/sub model in the domain of aircraft scheduling. An original approach is used here to solve the decentralized aircraft landing scheduling problem, and the event-based coordination model is more robust than the direct coordination model. Most importantly, the mathematical description of the publish/subscribe rules for the event-based coordination model in this research provides a better way than plain text description, which usually is the way most of the ATM papers use on introductory of a new operation concept or improvement to the current NAS. To assist with achieving a global solution of aircraft landing scheduling when several aircraft are involved in a distributed environment, the event-based distributed coordination model is integrated with an on-board system. Called the On-Board Aircraft Landing Scheduling Tool, it provides the pilot with the capability to self-schedule in the dynamic operational environment of the terminal area of non-controlled airports. The aircraft agent developed in this chapter is designed and implemented for FAR 23 Class GA aircraft under Free Flight conditions. Although the target operational environment investigated in this work is non-controlled airports, with suitable modification the approach is potentially flexible enough to extend to controlled airports too, where decentralized management can provide more airspace capacity, flight flexibility, and increased operational robustness.

This chapter seeks to analyze the degree of decentralization for aircraft landing scheduling in the dynamic operational environment at non-controlled airports, and thus explore the feasibility and capability of aircraft landing scheduling within uninterrupted free-flight environment in which there is no existence of ATC. This chapter is organized as follows. Initially, an aircraft agent is developed for the agent-based flight simulation, followed by the methodologies description of how to integrate the aircraft landing scheduling model into the flight deck on-board

system of an aircraft agent. Then, uninterrupted free-flight aircraft landing operation at non-controlled airports, represented by the on-board aircraft landing scheduling tool, is examined in which the ground-based automated air traffic control system serves as the baseline system for comparison. Finally, two coordination models, with focus on event-based coordination model, are developed, and related distributed coordination issues are addressed.

B. Decentralized Aircraft Landing Scheduling at Non-Controlled Airports

Decentralized aircraft landing scheduling problem has not been clearly stated in the literature by far prior to the advent of this research. However, it is an important problem deserving of closer attention since it provides a clear approach to the distributed air traffic management system. As stated earlier, the purpose of this research is to analyze the degree of decentralization for aircraft landing scheduling in the dynamic operational environment at non-controlled airports. It starts with the non-control airport case since current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the analysis of the distributed air traffic management system and the free-flight concept.

Similar as stated in chapter III, only the single runway case is considered since the vast majority of non-controlled airports only have instrument approaches designed for the primary runway. Even for those airfields that do not have ATC service but do have multiple runways, estimated to be only about 1%, in bad weather there is almost always only one runway that instrument approaches are flown to.

1. Integration of Aircraft Agent and Aircraft Landing Scheduling Tool

In this section the aircraft agent with distributed coordination function and the aircraft landing scheduling model developed in chapter III are briefly reviewed, followed by the methodologies description of how to integrate the aircraft landing scheduling model into the flight deck on-board

system of an aircraft agent. The on-board aircraft landing scheduling tool provides the capability of decentralized aircraft landing operation.

1) Aircraft Agent

The aircraft agent developed in this research is designed and implemented for FAR 23 Class GA aircraft under Free Flight conditions. Most FAR 23 Class GA aircraft in use today, however, are not equipped with the standard navigation and communication devices commonly found on commercial air transportations. As the concept of decentralized aircraft landing scheduling introduced in this research is designed for a future implementation of Free Flight, it is anticipated that this advanced equipment will be available on GA aircraft at that time. Therefore, in this research it then assumes that the aircraft agents implement real-time flight operations with the aid of the following on-board devices: ADS-B devices, so that the aircraft has the access to current traffic information; Flight Management System (FMS), so that the aircraft can utilize the traffic information from ADS-B devices for Conflict Detection & Resolution (CD&R) and landing scheduling; and a communication device that gives the aircraft the capability to communicate with other aircraft. The CD&R algorithm developed in this research is a modified version of the original Small Aircraft Transportation System (SATS) CD&R algorithm, where modifications are made as per the review comments of the CD&R modeling methods currently in use or under operational evaluation in [50]. It should be noted that the CD&R algorithm not only maintains a distance-based separation in the en-route phase of flight, but also enforces a time-based spacing in the final approach. It is assumed that each aircraft agent is equipped with such on-board CD&R module to plan maneuvers for an optimized and conflict-free trajectory. It is also assumed that the on-board CD&R module is sufficient to provide self-separation advisories for pilots since this research targets issues of decentralized aircraft landing operation only.

2) Aircraft Landing Scheduling Model and Algorithms

The aircraft landing scheduling problem is the problem of deciding a landing time on an appropriate runway for each aircraft for a given set of aircraft such that each aircraft lands within

a predetermined time window, and the separation criteria the aircraft landings are respected. In chapter III, the aircraft landing scheduling model at non-controlled airports is formulated as: *minimize* $Z(\underline{x})$ *subject to* $C(\underline{x})$. $Z(\underline{x})$ is the objective function, and two objective functions are defined: minimizing the total cost of deviation from the preferred landing time and minimizing the total holding time. $C(\underline{x})$ represents the constraints of the problem. The set of constraints defined in the model are given as:

$$E_i \le x_i \le L_i, \qquad i = 1,\ldots, N \tag{1}$$

$$\delta_{ij} + \delta_{ji} = 1, \qquad i = 1,\ldots, N; j = 1,\ldots N; i < j \tag{2}$$

$$x_i + S_{ij}\,\delta_{ij} - (L_i - E_j)\,\delta_{ji} \le x_j \qquad i = 1,\ldots, N; j = 1,\ldots, N; i \ne j \tag{3}$$

$$\text{Max}\,(0, P_i - x_i) \le b_i \le P_i - E_i, \qquad i = 1,\ldots, N \tag{4}$$

$$\text{Max}\,(0, x_i - P_i) \le a_i \le L_i - P_i, \qquad i = 1,\ldots, N \tag{5}$$

$$x_i = P_i - b_i + a_i, \qquad i = 1,\ldots, N \tag{6}$$

Inequality (1) ensures that each aircraft must land within its predetermined landing time window, where E_i and L_i represent the earliest and latest landing time of aircraft i, respectively. Equation (2) describes that either aircraft i must land before aircraft j ($\delta_{ij} = 1$) or aircraft j must land before aircraft i ($\delta_{ji} = 1$). Inequality (3) defines the separation constraint where S_{ij} represents the separation time requirement between aircraft i and j. The variables a_i and b_i describe how soon aircraft i lands after or before the preferred landing time P_i, respectively. Inequality (4) ensures that b_i is at least as big as the maximum of zero or the time difference between P_i and x_i, and at most the time difference between P_i and E_i. Inequality (5) shows the similar meaning for a_i. Equation (6) relates the a_i, b_i, and x_i variables to the preferred landing time P_i.

In chapter III, two scheduling approaches are developed and implemented to solve the aircraft landing scheduling problem described above: first-come-first-serve scheduling and optimal scheduling. The first-come-first-serve scheduling is the baseline approach for comparing with the optimal scheduling. There is no specified objective function in the first-come-first-serve scheduling model. Instead, the paradigm is applied as long as the constraints described in (1) to

(6) are satisfied. The optimal aircraft landing scheduling problem is solved in two different ways, one is solved as a linear programming problem using the simplex algorithm, and the other is solved as a job shop scheduling problem using branch-and-bound and tree-search algorithms. Three performance metrics, with the first two working as objective functions directly, are used for the optimal scheduling algorithm efficiency and effectiveness evaluation. The first performance metric, total cost of deviation from the preferred landing time, is the sum of weighted Dynamic Time of Arrival (DTA) deviation of all the aircraft in a scenario. This performance metric indirectly measures the scheduling algorithm efficiency with regard to the complete flight path. The second performance metric, total holding time, is the sum of delay time of all the aircraft in a scenario during the flight segment of three SATS holding patterns (2000ft, 3000 ft, and 4000 feet respectively). This performance metric indirectly measures the fuel and efficiency affected by the scheduling algorithm, with special consideration of the SATS High Volume Operation (HVO) scenario at non-controlled airports. The last performance metric, total delay time of feeder route, directly measures the delay time with regard to the flight segment of the feeder route. It is not an objective function, but introduced as a supplementary metric to evaluate flight efficiency, since the delay encountered from the scenario entry point to the IAF is distinguishable from the holding delay measured in total holding time.

3) On-board Aircraft Landing Scheduling Tool

In chapter III, the aircraft landing scheduling model and scheduling algorithms were implemented as a ground-based air traffic control automation system. Now we need to integrate the aircraft landing scheduling model into the on-board system of an aircraft agent for the sake of feasibility and capability analysis of decentralized aircraft landing operation. The integration of the scheduling model into on-board system alone, however, can only output a local scheduling decision based on its knowledge of a set of aircraft that are going to be sequenced. To assist with achieving a global solution of aircraft landing scheduling when several aircraft are involved in a distributed environment, a distributed coordination module is integrated with the on-board

system. It is called the On-Board Aircraft Landing Scheduling Tool, and it provides the pilot with the capability to self-schedule in the dynamic operational environment of the terminal area of non-controlled airports.

In [51], Harper, Mulgund, et.al. present a definition of degrees of pilot autonomy in Free Flight , as shown in Table 1.

Table 1. Degree of Pilot Autonomy[51]

Degree	Level of Autonomy
1	Standard ATC. Pilots act as instructed by ATC.
2	Pilot is free to search for and negotiate potential solutions with other pilots of level 2 or higher than and with ATC, and implement the resulting globally approved actions.
3	Pilot is free to search for and negotiate potential solutions with other pilots of level 2 or higher, and pose solutions and ATC for approval before implementation.

It is clearly seen that even for the highest level of pilot autonomy defined in [51], ATC is always included into the system and acts as a high-level supervisor or coordinator. As stated earlier in this chapter, the purpose of this research is to explore the feasibility and capability of uninterrupted free-flight environment in which there is no existence of ATC. In general, the abandonment of central control and stringent hierarchical data structures in favor of decentralized control strategies based on interactions, which require autonomous components, leads to solutions that are more flexible, more tolerant to perturbations, and thus are capable of supporting more emergences of new properties. In the specific domain of distributed aircraft landing scheduling at non-controlled airports addressed in this research, it seeks to utilize decentralized control strategies to achieve acceptable operational performance, especially for landing scheduling

operation, at non-controlled airports that there is no top level supervisor or central control so that all the aircraft involved are operated in a complete distributed environment. The Distributed Air/Ground-Traffic Management (DAG-TM) concepts envision a possible way to manage arrivals such that the flight deck is responsible for landing scheduling in the uninterrupted free-flight environment, but the capability analysis of an on-board landing scheduling tool has not been explored prior to the advent of this research. It can be expected that the future air traffic management system will manage aircraft landing in a way that lies somewhere between the two extremes, fully ground-based and uninterrupted free-flight, possibly moving gradually from ground-based to more free-flight. Figure 8 illustrates the situation. It shows that as more operational responsibilities are transferred from the Air Traffic Service Provider components to the Flight Crew components, in other words, the closer to uninterrupted free-flight operational environment, the Flight Crew has more planning capability and the CNS infrastructure burden decreases. However, the trade-offs are the increasing flight crew workloads and avionics equipage cost.

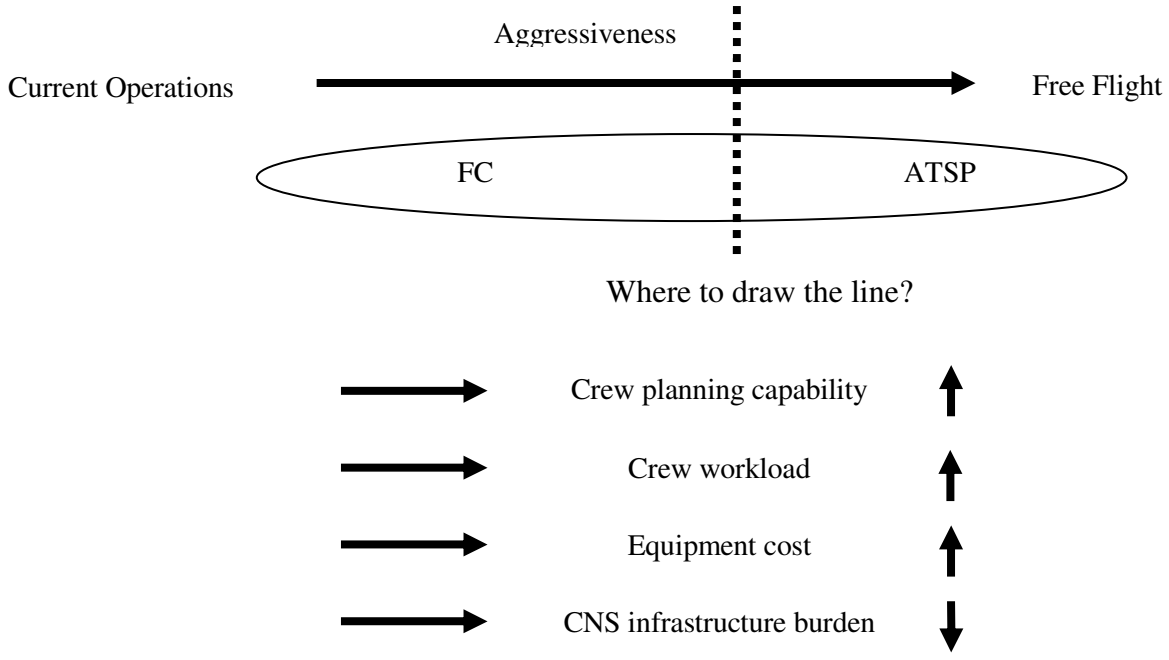

Figure 8: Operation Concept – from Ground-Based to Free-Flight

As stated previously, current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the implementation of uninterrupted free-flight concept. In this research, the on-board aircraft landing scheduling tool is developed, and it represents the ultimate uninterrupted free-flight scenario since there is no ground-based automated system to enforce any centralized control and flight crews take over all of the responsibilities that the current controllers have. Aircraft at the terminal area of non-controlled airports are placed in a complete distributed environment, and each pilot is required to use the on-board aircraft landing scheduling tool to provide sequencing advisories. It should be stressed that the air traffic control automation system discussed in chapter III, functioning as a ground-based aircraft landing scheduling tool, serves as baseline system for comparison. Although comparing the performance of these two options is not critical for the non-controlled airports case since we aim to achieve uninterrupted free-flight environment, it is important for the future extension to controlled airports since the approach will give a clear view of where to establish the line of free-flight concept application for the controlled airport case.

2. Distributed Coordination in the Dynamic Operational Environment

In this research, it is assumed that flight crews are required to use the on-board aircraft landing scheduling tool to provide scheduling advisories dynamically in the ultimate uninterrupted free-flight operational environment at the terminal area of non-controlled airports. It then presents an instance of a typical distributed coordination problem for a multi-agent system when all aircraft agents in a scenario are required to coordinate in order to achieve a scheduling decision after they utilize the on-board scheduling tool to reach their own respectively. The distributed coordination function module then becomes the key element of the on-board aircraft landing scheduling tool to achieve a global solution of aircraft landing scheduling when several aircraft are involved in the distributed environment. It is not exaggerating that the distributed coordination plays the most important role in the success of decentralized aircraft landing operation. This section will then

seek to address how the challenges brought by distributed coordination issues are resolved in this research, from the establishment of the mathematical model to the application implementation.

Two coordination models, direct coordination model and event-based model, are candidates for solving the problem of distributed aircraft landing scheduling at non-controlled airports.

1) Direct Coordination Model

In general, direct coordination models used in distributed applications are usually the ones in which the agent's interactions involve explicit task-directed communications or negotiations about global resource usage or task assignments in order to achieve coordinated behavior [46]. In this research, direct coordination model is implemented by proposing a means that group of agents can coordinate by communication with each other in a direct and explicit way, specifically, negotiation messages exchange among aircraft using on-board ADS-B equipment, where the approach detail is addressed in [52]. Originally in [52] it establishes a pair-wise argument-based negotiation approach that achieves collaboration among aircraft for searching multilateral acceptable solution in the Conflict Detection and Resolution domain. In this research, it modifies the negotiation algorithm to achieve a mutual acceptable solution in aircraft landing scheduling domain. Dead-lock in the negotiation is resolved by either introducing rejection action or applying wait-die scheme. The wait-die scheme uses a time stamp to label a proposal and takes a re-propose action when response is not received in a certain amount of time. Figure 9 shows the negotiation protocol.

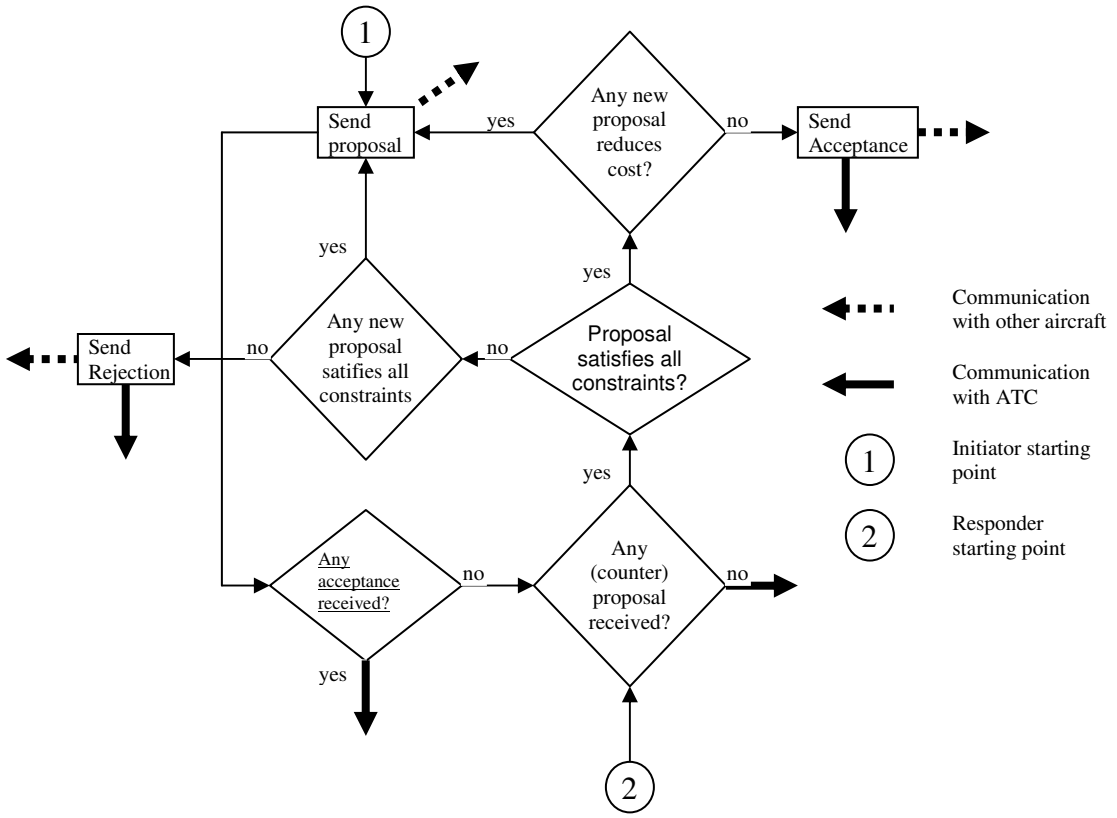

Figure 9: Pair-wise Argument-based Negotiation Protocol

At this time, it is still an open and energetically debated issue as to the relative merit of direct coordination model and event-based coordination model. For the specific case addressed in this research, it is expected that the explicit coordination with negotiation approach will achieve acceptable performance for distributed aircraft landing scheduling for simple air traffic scenarios (with less than 4 aircraft involved). However, the pair-wise argument-based nature of the negotiation algorithm will bring capacity and performance issues when resolving more advanced air traffic scenarios since it will take maximum C_N^2 negotiations to achieve a global solution if the scenario involves N aircraft. The complexity of the scenario will make the time it costs to reach the final global scheduling decision unbearable compared to the event-based coordination

approach. Sometimes even worse, it will not achieve a converged global solution due to the non-deterministic characteristic of the negotiation algorithm.

2) Event-Based Coordination Model

Event-based coordination model, on the other hand, is one in which each agent operates under local control and system-level coordinated behavior arises from agent-agent interactions by generating events and by reacting to events of interest, without an explicit notion of task-directed communication or negotiation. Event-based coordination in a distributed system is dominated by client/server platform relying on synchronous request/reply. However, this architecture is not well suited to this research due to the following reason:

a) In the client/server platform, clients and servers are coupled, i.e., sever needs to know the identification of the client who requests service so that it can deliver the service to the client. However, in the decentralized aircraft landing scheduling application address in this research, it is desired that the aircraft who generate the scheduling decision and the aircraft who follow it are decoupled, i.e., the aircraft following the scheduling decision only care if an active scheduling decision is generated or not, but independent from the identification of the aircraft who generate it.

b) The client/server platform heavily relies on synchronous request/replay, i.e., the client needs to be blocked waiting for its requesting service from the server in order to maintain client/server synchronization. It is not desirable for this research since the aircraft needs to perform concurrent flight operations while waiting for the scheduling decision from the other aircraft that generate them.

In contrast to client/server platform, pub/sub paradigm directly reflects the intrinsic behavior of information-driven applications because communication here is indirect and initiated by producers of information: producers publish notifications and these are delivered to subscribed consumers by the help of a notification service that decouples the producers and the consumers [47]. The loosely coupled communication scheme that introduced by the pub/sub paradigm fits

well to the highly dynamic nature of the operational environment of the decentralized aircraft landing operations addressed in this research. Furthermore, pub/sub paradigm represents a general-purpose solution for information dissemination that can fit the scenarios that require an asynchronous many-to-many communication, and it is exactly the desirable feature in the decentralized air traffic management system. Therefore, pub/sub communication paradigm is chosen to implement event-based coordination in this research.

In the pub/sub system established in the specific domain of aircraft landing scheduling addressed in this research, the *producers* are defined here as arbitration aircraft that are triggered by time/distance based events. They then initiate the scheduling decision process or aircraft state update broadcast. *Consumers* are defined as the submission aircraft that are notified by the arbitration aircraft with the scheduling decision or aircraft state update notifications. The submission aircraft then activate the corresponding event handler based on their "subscription" to the ADS-B message notifications. Each aircraft in a scenario can take on the role of an arbitration aircraft or a submission aircraft. Arbitration aircraft generate ADS-B message notifications during the scheduling or aircraft state update process, which is then "consumed" by submission aircraft. The main semantical characterization of this pub/sub system is in the way ADS-B message notifications flow from arbitration aircraft to submission aircraft: submission aircraft are not directly targeted from arbitration aircraft, but rather they are indirectly addressed according to the content of ADS-B message notifications. That is, a submission aircraft only expresses its interest by issuing subscriptions for specific ADS-B message notifications, independently from the arbitration aircraft that generate them, and then it is asynchronously notified for all ADS-B message notifications, submitted by any arbitration aircraft, that match their subscription. In the present context, asynchronous means that a submission aircraft does not have to be blocked waiting for notifications to arrive (such as in client/server model), but it can keep on performing concurrent flight operations.

In most of the pub/sub systems, a logical intermediary between publishers and subscribers, known as Notification Service (NS), is usually implemented to avoid each publisher to have to know all the subscription for each possible subscriber. Both publishers and subscribers communicate only with a single entity, the Notification Service. In this research, the "centralized-like" middleware is implemented as a distributed set of processes under the assumption that the on-board ADS-B equipment is capable of dispatching ADS-B message notifications effectively via reliable channels.

a. Elements of the Decentralized Aircraft Landing Scheduling Pub/Sub System

The decentralized aircraft landing scheduling pub/sub system is represented by a triple set <*A, P, S*> of processes (Figure 10). Sets are defined depending on the role of processes in the dynamic system at time instance *t*: $A(t) = \sum_{i=1}^{n} a_i(t)$ is a set of n processes, called *arbitration process* initiated by *arbitration aircraft*, which are producers of information; $S(t) = \sum_{i=1}^{m} s_i(t)$ *is* a set of m processes, called *submission process* generated by *submission aircraft*, which are

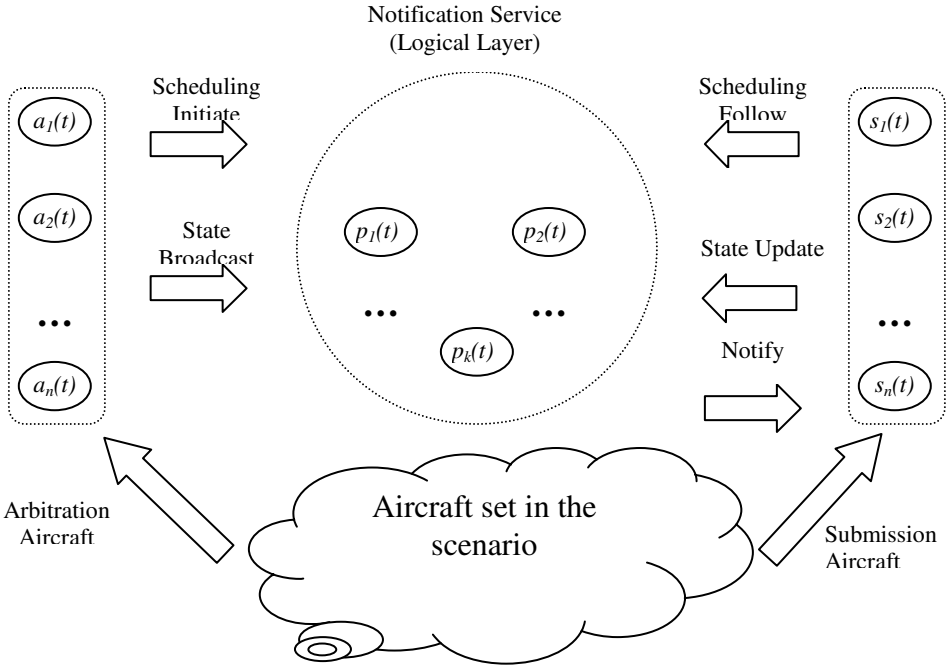

Figure 10: Decentralized Aircraft Landing Scheduling Pub/Sub System

consumers of information; $P(t) = \sum_{i=1}^{k} p_i(t)$ *is* a set of k processes, called *messenger process* worked as *centralized-like* middleware at logical layer, which are residents of the Notification Service.

It is assumed that arbitration and submission processes are decoupled, meaning a process in *A* cannot communicate directly with a process in *S* and vice versa at any time instance *t*. Decoupling is a desirable feature in this research since it isolates the distributed coordination process on making scheduling decision, which is the focus of this research, from the ADS-B communication issues such as addressing or synchronization. Processes in *A* and *S* can communicate with any process in *P*. Therefore, the set of *P* represents a logically centralized entity that allows the communication between *publishers* (or producer) and *subscribers* (or consumer), at the same time maintaining them decoupled. In this research, the physical implementation of the Notification Service is an on-board autonomous component on the top of integrated system of each aircraft agent.

b. Interaction between Process and the Notification Service

As stated earlier, there is no direct communication between processes in set *A* and *S*, and all communication has to pass through the Notification Service. The execution of the decentralized aircraft landing scheduling pub/sub system then comprises two categories of operations: *process-side operations*, started by time/distance based event trigger of aircraft agent; and *NS-side operations*, started by the NS. More specifically, any aircraft agent in the system can initiate an *arbitration process* (publish) or a *submission process* (subscribe), but it is the NS that plays the role of notifying a matching occurrence to interested subscribers. The following four types of operations are therefore defined accordingly:

a) *Pub(a_i):* operation of publishing process a_i in set *A*. a_i is initiated by an *arbitration aircraft* and it can be a *scheduling initiate process* or *state broadcast process*.

b) *Notify(p_i):* operation of issuing the notification of p_i.

c) *Sub(s_i):* operation of registration of a subscription s_i in set *S*. s_i is initiated by a *submission aircraft* and it can be a *scheduling follow process* or *state update process*, which are the counterparts of publications of *scheduling initiate process* and *state broadcast process*.

d) *Unsub(s_i):* operation of cancellation of a subscription s_i in set *S*.

Then the operations *Pub(a_i), Sub(s_i),* and *Unsub(s_i)* are issued by a process in set *A* and *S* respectively, and are sent to NS for execution. Operation *Notify(p_i),* on the other hand, is issued by the NS and send to the process for execution. Note that operation *Notify(p_i)* only occurs after the execution of its *Pub(a_i)* counterpart (i.e., p_i and a_i share the same content, hereafter denoted as *content(p_i) = content(a_i)).*

c. Communication Delay Consideration

There are two types of communication delay that need to be taken into account for the physical implementation of the pub/sub system:

a) Subscribe/unsubscribe delay: when a subscribe/unsubscribe operation occurs, the NS is not immediately aware of the event since the registration or cancellation of a subscription takes a certain amount of time to be stored into the NS. In this research, the major contribution of this communication delay comes from the internal database update of the NS. To simulate such communication delay, an acceptable time threshold is used. We denote such delay as T_{sub} for subscribe operation and as T_{unsub} for unsubscribe operation. Therefore if a subscribe operation is issued at time t_s then it takes effect at a time t such that $t_s \le t \le t_s + T_{sub}$. The same holds for unsubscribe operation, that is, an unsubscribe operation issued at time t_u takes effect at a time t' such that $t_u \le t' \le t_u + T_{unsub}$

b) Publication/notification delay: similar as subscribe/unsubscribe operations, it would take certain amount of time to complete publication or notification operation. In this

research, the major contribution to publication delay comes from ADS-B transmission delay, whereas the computation time of "matching interested subscribers" in the NS mainly contributes to the notification delay. Again, two time thresholds, denoted as T_{pub} and T_{not}, are used to simulate publication delay and notification delay respectively.

d. Computational Model

In this research, two subscription types are defined: *scheduling decision subscription*, denoted as θ_d, which is initiated by a *scheduling follow process* in set *S*; and *aircraft state update subscription*, denoted as θ_u, which is initiated by a *state update process* in set *S*. Similarly, two publication types are defined: *scheduling decision publication*, denoted as Ψ_d, which is initiated by a *scheduling initiate process* in set *A*; and *aircraft state update publication*, denoted as Ψ_u, which is initiated by a *state broadcast process* in set *A*. Assuming the issue of an operation *op* = {$Pub(a_i)$, $Notify(p_i)$, $Sub(s_i)$, $Unsub(s_i)$} at time instance *t* generates an event $e_i(op, t)$. Then the local history of any process *i* can be denoted as a sequence of events according to the time order of occurrence: $h_i = \{e_i(op, t_1), e_i(op, t_2), ..., e_i(op, t_m)\}$ where $t_1 < t_2 < ... < t_m$. The global history is then denoted as $H = \{h_1, h_2, ..., h_n\}$, a collection of local histories, one for each process.

The time interval of an active subscription θ can be represented by two successive events $e_i(Sub(s_i), t_s)$ and $e_i(Unsub(s_i), t_u)$, denoted as $T(\theta)$. Such subscription time interval then include all events $e_i(op, t)$ s.t. $t_s \le t \le t_u$. The time internal of an active publication Ψ, denoted as $T(\Psi)$ can be represented by two successive events $e_j(Pub(a_j), t_p)$ and $e_i(Notify(p_j), t_n)$ such that $t_p \le t_n \le t_p + T_{pub} + T_{not}$ and $content(a_j) = content(p_j)$.

Several publish/subscribe rules are then applied to the decentralized aircraft landing scheduling pub/sub system accordingly to some general properties of pub/sub system and some application-specific features in this research:

a) Legality rule (general): a subscriber cannot be notified for any information it is not subscribed. The mathematical formulation is presented as follows:

$$\forall e_i(Notify(p_i),t)\in H \Rightarrow e_i(Notify(p_i),t)\in T(\theta)\ \ s.t.\ content(\theta.s)=content(p_i) \qquad (13)$$

b) Validity rule (general): while Legality states that a notify event belongs to *H* only if it is included in a subscription time interval matching that event, we need a property that ensures the notify events are not invented by a process, but are always invoked after publish event. In its mathematical formulation, this Validity rule is presented as follows:

$$\forall e_i(Notify(p_i),t)\in H \Rightarrow \exists e_j(pub(a_j),t')\in H\ \ s.t.\ content(p_i)=content(a_j)\ and\ t'<t \qquad (14)$$

c) Liveness rule (general): it states exactly to which subscriber a publication is notified to, considering both subscribe/unsubscribe delay and publication/notification delay. The mathematical formulation is presented as follows:

$$\forall(e_i(Pub(a_i),t)\bigcap(T(\theta)\in H\ s.t.T(\theta)\supset[t,t+T_{pub}+T_{not}])\ s.t.content(a_i)=conent(\theta.s))$$
$$\Rightarrow\exists e_j(Notify(p_j),t')\in H\ \ s.t.\ content(p_j)=content(a_i)\ and\ t<t'<t+T_{pub}+T_{not} \qquad (15)$$

d) Uniqueness rule 1 (application specific): it poses the constraint that at most one *scheduling decision publication* can be active at a time, i.e., at most one aircraft can initiate a scheduling decision process at a time. The mathematical formulation is presented as follows:

$$\exists(e_i(Pub(a_i),t)\bigcap(T(\Psi_d)\in H\ s.t.T(\Psi_d)\supset[t,t+T_{pub}+T_{not}])$$
$$\Rightarrow\neg(\exists e_j(pub(a_j),t')\bigcap(T(\Psi'_d)\in H\ s.t.T(\Psi'_d)\supset[t,t+T_{pub}+T_{not}])\ and\ i\neq j) \qquad (16)$$

e) Uniqueness rule 2 (application specific): it poses the constraint that at most one *aircraft state update publication* per aircraft ID can be active at a time, i.e., an aircraft can at most initiate one ADS-B state update broadcast process at a time. Its mathematical formulation is presented as:

$$\exists(e_i(Pub(a_i),t)\bigcap(T(\Psi_u)\in H\ s.t.T(\Psi_u)\supset[t,t+T_{pub}+T_{not}])$$
$$\Rightarrow\neg(\exists e_j(pub(a_j),t^{'})\bigcap(T(\Psi^{'}_u)\in H\ s.t.T(\Psi^{'}_u)\supset[t,t+T_{pub}+T_{not}])$$
$$and\ content(a_i.AircraftID)=content(a_j.AircraftID)\ and\ i\neq j) \quad (17)$$

f) Completeness rule (application specific): it states the fact all of the aircraft in the scenario need to be notified whenever an aircraft initiates a scheduling decision process or ADS-B state update broadcast process. The complete set of aircraft ID in the scenario is denoted as Ω. It unifies the information space of all aircraft in the scenario:

$$\forall(e_i(Pub(a_i),t)\Rightarrow\forall j(\exists e_k(Notify(p_k),t^{'})\in H\bigcap j\in\Omega\ s.t.\ content(p_k.AircraftID)=j$$
$$and\ t<t^{'}<t+T_{pub}+T_{not}) \quad (18)$$

e. Implementation Methodology

The following assumptions are made for the implementation of the event-based coordination model of the decentralized aircraft landing scheduling system:

a) ADS-B device provides the sufficient updated knowledge of all aircraft to be sequenced, and is defined here as aircraft state information (altitude, longitude, latitude, airspeed, vertical speed, and heading), time stamp, flight path intent information (next two waypoints on the intended flight path).

b) Emergency mode is not considered, i.e., no aircraft has higher priority than any other one due to the fuel status and emergency priority.

The event-based coordination model is designed and implemented in the form of a *finite state machine* that consists of a variety of discrete legal states and the legal transitions between states during the scheduling process and ADS-B state update process. Figure 11 shows the Matlab Stateflow implementation of the event-based coordination for scheduling process.

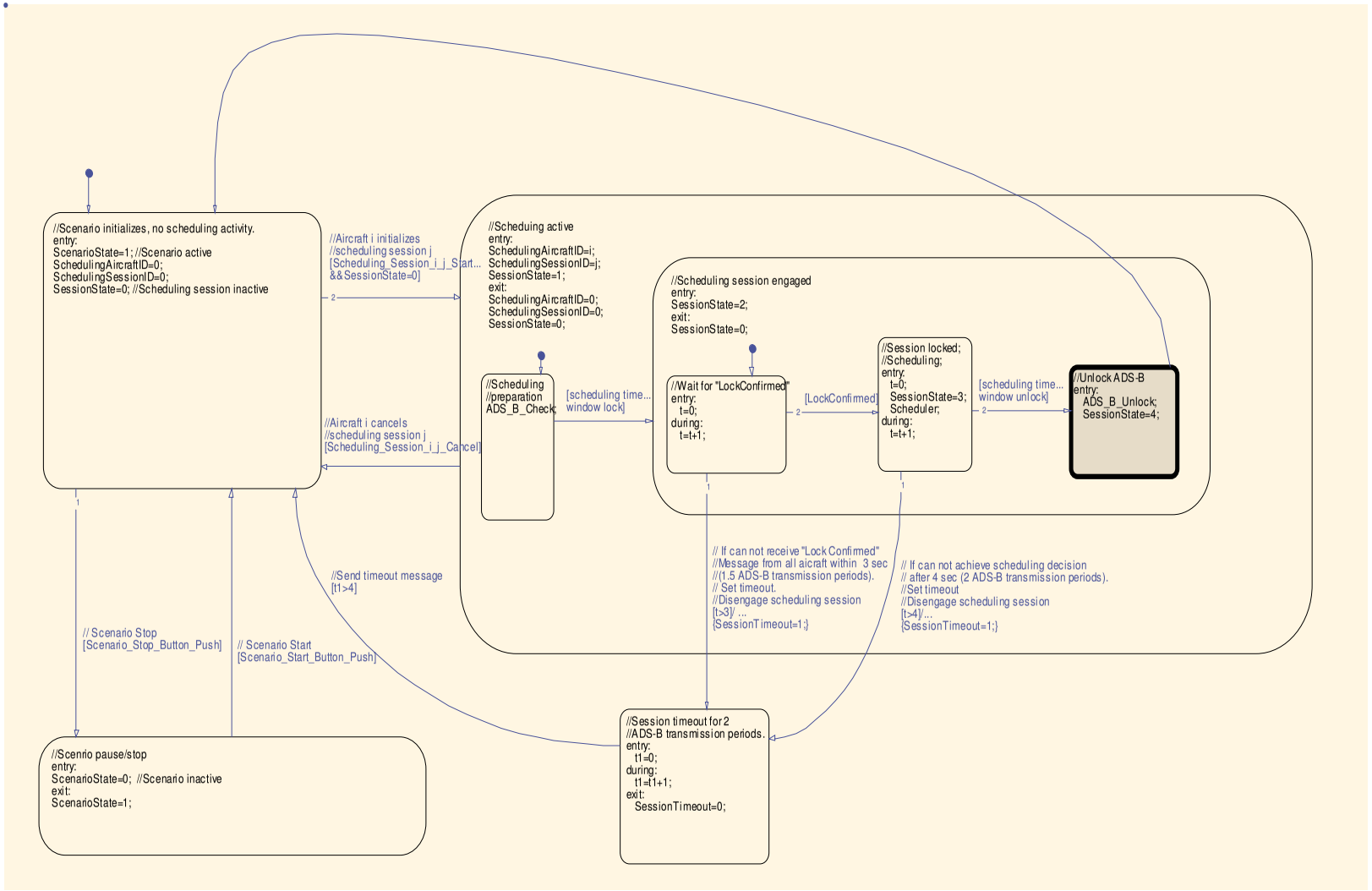

Figure 11: Event-based Coordination for Scheduling Process

Each aircraft in the scenario is equipped with an ADS-B transponder, and maintains a local database that stores a certain time range of ADS-B messages. An on-board scheduling tool using aircraft state information retrieved from the ADS-B message is used to make scheduling decisions. Details of the states defined in the event-based coordination for scheduling process and the transitions between states are described in Table 2, and details of states defined in the event-based coordination for ADS-B state update process and the transitions between states are described in Table 3.

Table 2. Event-based Coordination States/Transitions for Scheduling Process

State Description	Event Trigger	Actions
1. Scenario initialization	Scenario start command	Initialize all the aircraft agent in the scenario; create/update local ADS-B database for each aircraft agent. Move to state 2 under specified event trigger.
2. Scheduling session initialization	Aircraft i initializes a scheduling session when no other scheduling session active at the moment	Record the ID of aircraft i (arbitration aircraft); Record the scheduling session ID; Arbitration aircraft checks the time-stamp of all the other aircraft in the scenario; Set the scheduling session state to 1. Move to state 3 under specified event trigger.
3. Wait for "Lock Confirm"	Arbitration aircraft broadcasts a "scheduling time window lock" message	Notify all the other aircraft (we call them submission aircraft) in the scenario to lock their ADS-B message updates and to send back a "lock confirmed" message that includes the most recent ADS-B message; Set the scheduling session state to 2. Move to state 4 or 5 under specified event trigger.

Table 2 Continued

4. Scheduling session locked	Arbitration aircraft receives "Lock Confirm" message from all the other aircraft	Make scheduling decision; Set the scheduling session state to 3. Move to state 6.
5. Scheduling session timeout	Arbitration aircraft does not receive "Lock Confirm" message from all the other aircraft within 1.5 ADS-B transmission period, or can not achieve converged scheduling decision within 2 ADS-B transmission period.	Scheduling session timeout; Set SessionTimeout state to 1; Move to state 1 after 2 ADS-B transmission period timeout.
6. Unlock ADS-B	Arbitration aircraft broadcast a "scheduling time window unlock" message with the scheduling decision.	All submission aircraft unlock their ADS-B message updates and follow the scheduling decision made by the arbitration aircraft; Set the scheduling session to 4; Move back to state 1.

Table 3. Event-based Coordination States/Transitions for ADS-B State Update Process

State Description	Event Trigger	Actions
1. Scenario initialization	Scenario start command	Initialize all the aircraft agent in the scenario; create/update local ADS-B database for each aircraft agent. Move to state 2 under specified event trigger.

Table 3 Continued

2. ADS-B state update session initialization	Aircraft i initializes an ADS-B state update session when no other ADS-B state update session active issued by the same aircraft at the moment	Record the ID of aircraft i (arbitration aircraft); Record the scheduling session ID; Move to state 3 under specified event trigger.
3. ADS-B state broadcast	Arbitration aircraft broadcasts its updated ADS-B state message.	Notify all the other aircraft (submission aircraft) in the scenario to update their local database with the updated ADS-B state message from arbitration aircraft. Move to state 4.
4. Receive ADS-B state update notification	Submission aircraft receives ADS-B state update notification from the arbitration aircraft	Update local database with the updated ADS-B state message from arbitration aircraft. Move to state 1.

The following nominal scenario with 3 aircraft involved is provided for better understanding of how event-based coordination for scheduling process is achieved via pub/sub communication paradigm. Figures 12-15 show the scheduling initiation phase, scheduling following phase, dynamic re-scheduling phase, and unsubscribe scheduling phase of the nominal scenario.

- Phase 1 (Figure 12): Scenario starts at 3:00 PM. Two aircraft are approaching to their desired IAFs. Aircraft 1 is a Mooney 201, and its cruise speed is 180 knots and the distance to its target IAF (RAZVY) is 30 nm; aircraft 2 is a Piper Cub 80, and its cruise speed is 80 knots and distance to its target IAF (LOUIE) 35 nm. Aircraft 1 initiates a scheduling process by broadcasting a "scheduling time window locked" message. After it receives the "lock confirmed" message from aircraft 2, it makes the following scheduling decision using the on-board aircraft landing scheduling tool: Aircraft 1 gets the first spot in the landing sequence and its landing time is scheduled at 3:25 PM; Aircraft 2 gets the second sequencing spot and its landing time is scheduled at 3:47 PM. It then broadcasts the *scheduling decision publication*.

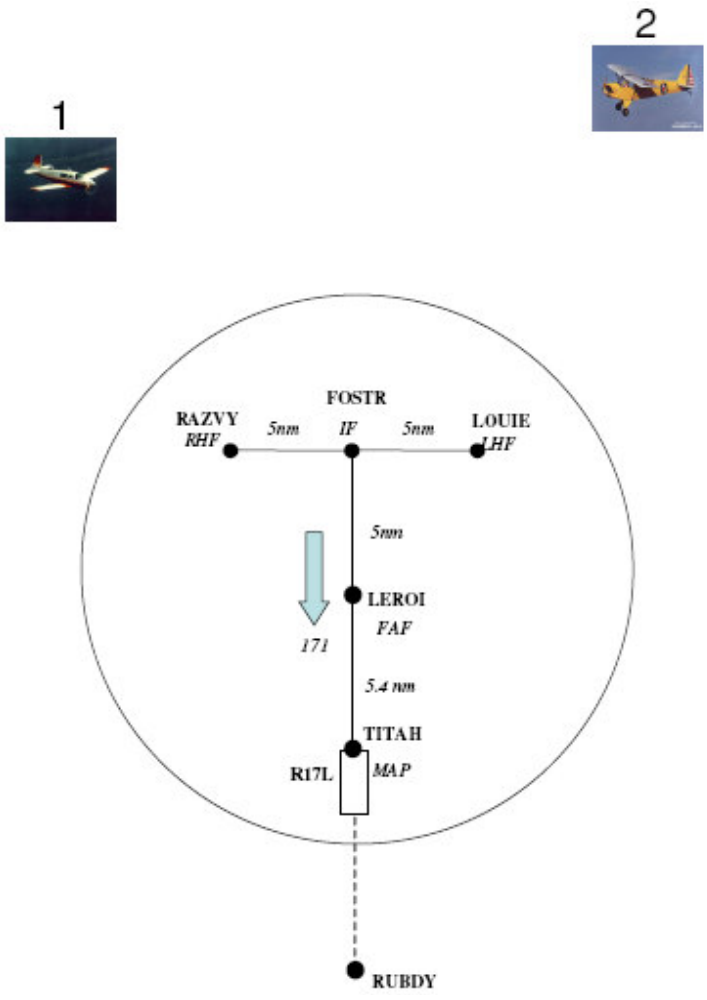

Figure 12: Nominal Scheduling Scenario Phase 1 - Scheduling Initiation

- Phase 2 (Figure 13): Aircraft 2 gets the notification with the scheduling decision since it is a subscriber to the *scheduling decision publication*. Both aircraft 1 and 2 follow the scheduling decision by planning the flight path that achieves the scheduled landing time. Aircraft 1 plans to arrive at its desired IAF (RAZVY) at 3:11 PM and starts the approach immediately whereas aircraft 2 plans to arrive at its desired IAF (LOUIE) at 3:28 PM starts the approach immediately. Separation requirement in the en-route phase of flight and spacing criteria within SCA are taken into consideration in the flight path planning using the on-board CD&R functionality.

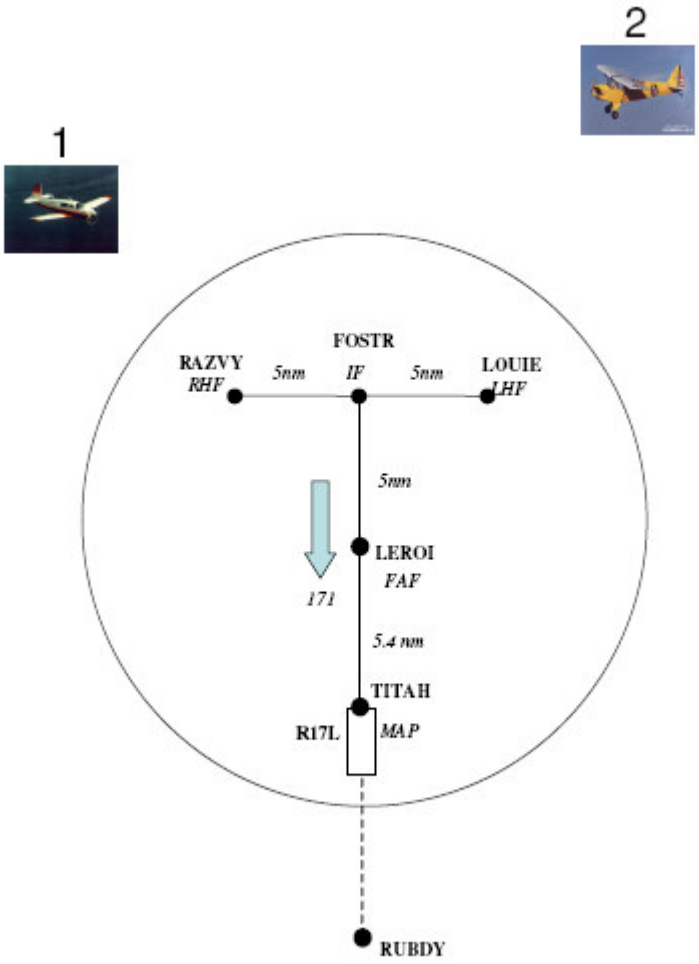

Figure 13: Nominal Scheduling Scenario Phase 2 - Scheduling Following

- Phase 3 (Figure 14): A Cessna 172 (aircraft 3 in the figure) with cruise speed 125 knots and distance to its target IAF (LOUIE) 37.5 nm enters into the scenario at 3:10 PM and it initiates a scheduling process immediately. It reaches the following scheduling decision that minimizes the global cost while maintaining the landing separation: Aircraft 1 keeps the first spot in the landing sequence and its scheduled landing time is unchanged. Aircraft 2 is re-scheduled to be the last one in the sequence and its scheduled landing time is change to 3:50 PM. Aircraft 3 gets the second spot and its landing time is scheduled at 3:42 PM. Aircraft 3 then broadcasts the new *scheduling decision publication*.

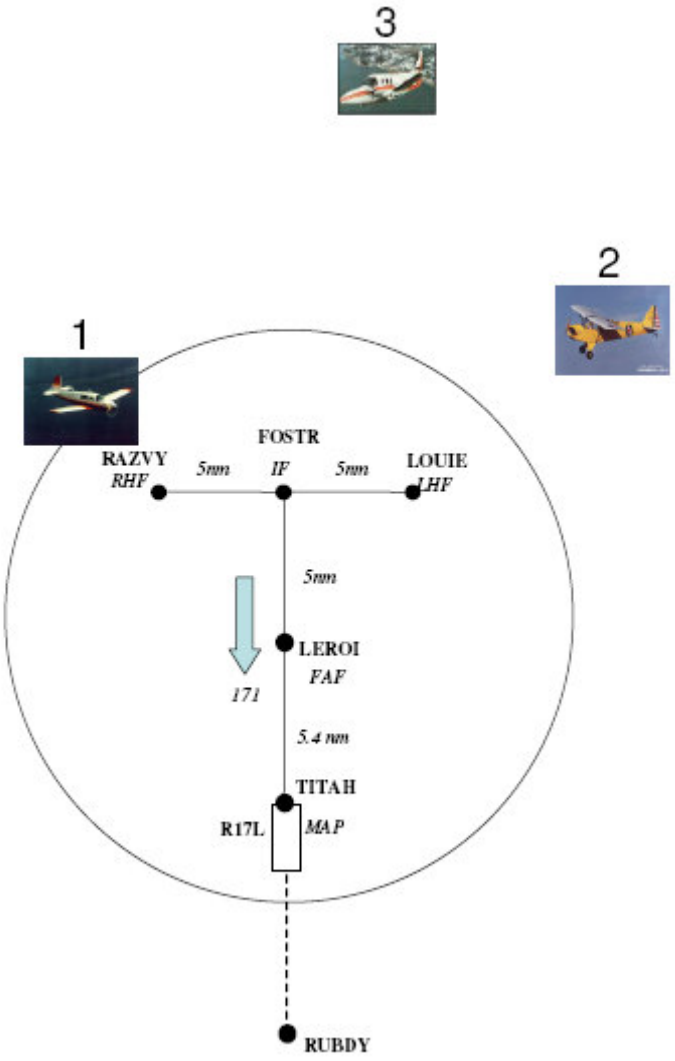

Figure 14: Nominal Scheduling Scenario Phase 3 - Dynamic Re-Scheduling

- Phase 4 (Figure 15): Aircraft and 2 get the notification with the new scheduling decision since they are both the subscriber to the *scheduling decision publication*. All three aircraft follow the new scheduling decision by re-planning the flight path that achieves the updated scheduled landing time. At 3:11 PM, aircraft 1 arrives at its target IAF (RAZVY). It then unsubscribes the *scheduling decision publication* before it enters into the SCA and starts the landing approach. It will not be notified with any further updated *scheduling decision publication.* However, it still receives the ADS-B state update from the other aircraft since it is a subscriber to the *aircraft state update publication.*

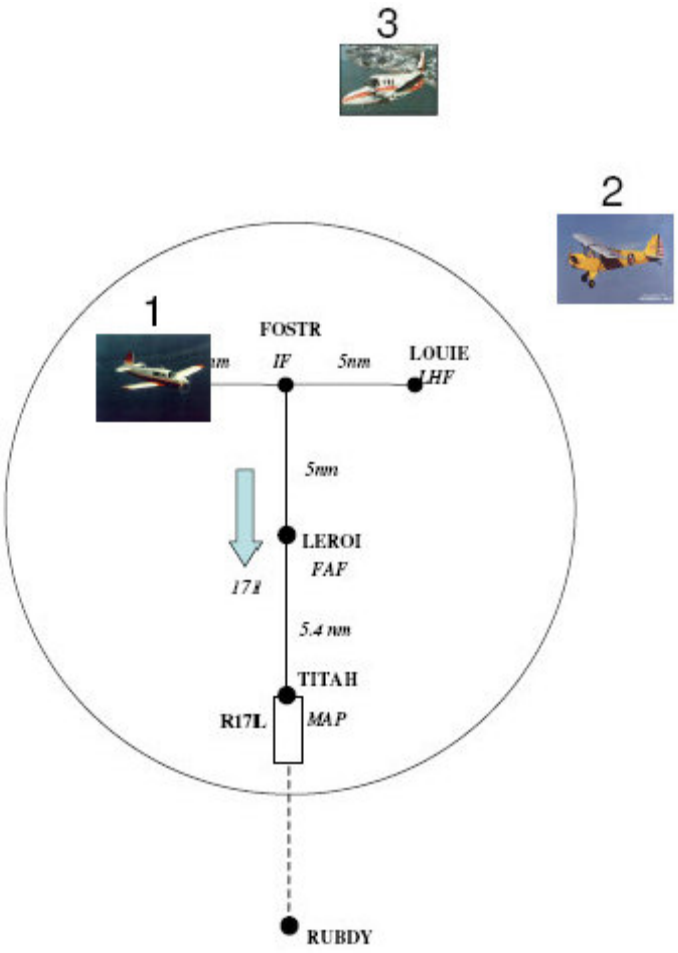

Figure 15: Nominal Scheduling Scenario Phase 4 – Unsubscribe Scheduling Decision Publication after Entering into SCA

The deterministic characteristic of the *state transition machine* implementation of the event-based coordination model for scheduling process assures that each aircraft in the distributed environment has identical information required for making a scheduling decision and will therefore generate an identical scheduling decision. However, further explorations are required for the following two boundary conditions:

a) More than one aircraft broadcast a *scheduling time window lock* message at the same time.

b) Aircraft is on the edge of updating an ADS-B message when receiving a *scheduling time window lock* message.

These two boundary conditions are extreme cases where they exceed the capability of the current computing power. Take condition 1 as an example, it will only be present when two aircraft broadcast the locking message exactly at the same time. Considering the current computing power can easily tell the ms time difference, its chance to happen in practice is rare. However, if it happens, i.e., two messages actually come in at the "same time" that the program can not tell the difference, the easiest practical solution is to reject both messages and let them re-broadcast. The systematic approach to address these issues is more like a computer science problem, which is beyond the scope of this research.

C. Special Considerations

There are several issues need to be addressed to complete the implementation of the pub/sub communication paradigm in decentralized aircraft landing scheduling application before we close this chapter.

1. Spacing Constraints within SCA

Earlier in this chapter it is mentioned that the on-board CD&R algorithm not only maintains a distance-based separation in the en-route phase of flight, but also enforces a time-based spacing in the final approach. This is achieved by taking separation requirement in the en-route phase of

flight and spacing criteria within SCA into account when the aircraft agent generates its flight path plan based on the current active scheduling decision. This is sufficient for the nominal scenarios when all the aircraft share the same target IAF, or time interval of two successive aircraft in the landing sequence with different target IAF arriving at IF (Intermediate Fix) meets the spacing criteria within SCA as per the scheduling decision. The nominal scenario showed earlier represents the case. However, applying landing separation constraint in scheduling decision-making alone is not sufficient for some nominal scenario since it only represents the accumulated separation at the runway threshold. Figure 16 shows an example.

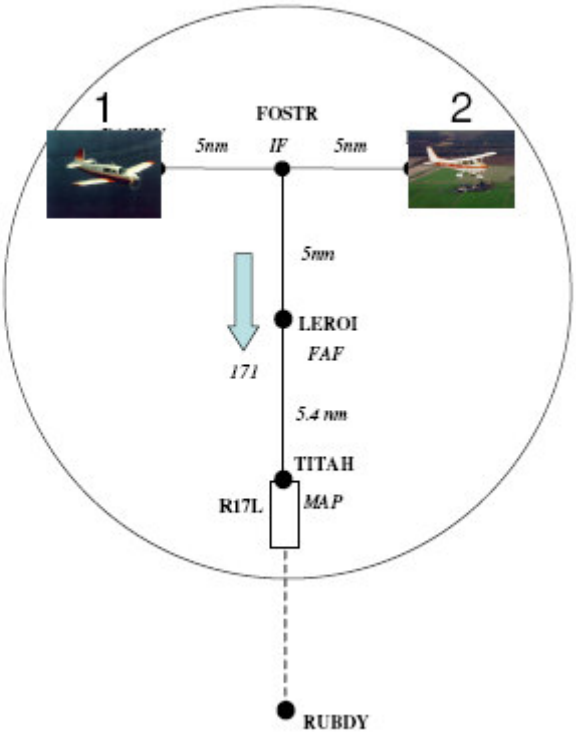

Figure 16: Nominal Scheduling Scenario – Insufficient Spacing Constraint

Supposing time stamp of the scenario is 3:00 PM. Aircraft 1 has holding speed of 65 knots and just arrives at IAF - RAZVY; aircraft 2 has holding speed of 80 knots and just arrives at the other IAF - LOUIE. Aircraft 1 initiates a scheduling process the scheduling decision is made as follows: Aircraft 2 lands first and its scheduled landing time is 3:12 PM; Aircraft 1 lands the

second and it is scheduled to land at 3:15 PM. They have 3 minutes landing separation since 2 minutes landing separation constraint is applied in the scheduling model. However, they both fly towards IF and at some point near to IF they will certainly violate the spacing criteria within SCA. Spacing constraints presented in [53] are then added to serve as additional constraints included in the aircraft landing scheduling model described in chapter III. Numerical result of the flight simulation exercise of the augmented scheduling model is presented in chapter V.

2. Dynamic Re-Scheduling Issues

Re-scheduling is triggered when some dynamic event happen in the operational environment, such as the appearance of a new aircraft or speed profile change due to deviation of the planned flight path. It is intentionally designed that aircraft that has entered into SCA will not be notified with any further updated *scheduling decision publication,* but still receives the ADS-B state update from the other aircraft. Scheduled landing time of all the aircraft within SCA are fixed and will not be affected by later scheduling decision update. The main purpose of this rule is to maintain the stable and orderly landing sequence within SCA due to the operation procedure constraints, such as a successive aircraft is not allowed to surpass any precedent aircraft within SCA. However, scheduling decision update needs to take those aircraft within SCA into account although the updated scheduling decision will not be delivered to those aircraft within SCA.

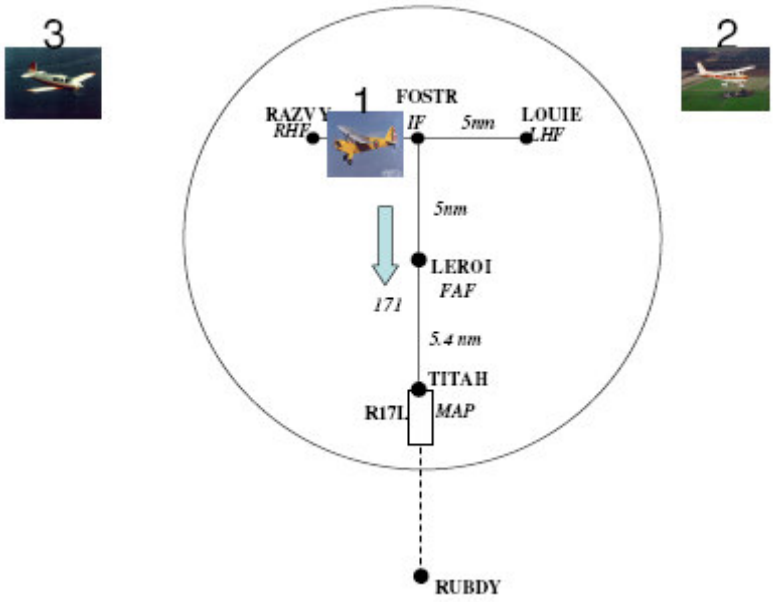

Figure 17: Nominal Scheduling Scenario – Dynamic Re-Scheduling Issue

Take the following scenario (Figure 17) as an example: Supposing time stamp of the scenario is 3:00 PM. Aircraft 1, a Piper Cub, with holding speed of 65 knots and approach speed of 50 knots, is flying towards IF. Its scheduled landing time is 3:14 PM and it will be fixed since aircraft 1 is within SCA already. Aircraft 2 is a Cessna 172, and it has holding speed of 100 knots and approach speed of 80 knots while being apart from its target IAF (LOUIE) 25 nm. Aircraft 3 is a Mooney 201 with holding speed of 110 knots and approach speed 80 knots while being 10 nm away from its target IAF (RAZVY). At the point aircraft 3 initiates a dynamic re-scheduling. If aircraft 1 is not taken into consideration when making updated scheduling decision, aircraft 3 will be scheduled first and its scheduled landing time is also 3:14 PM, which obviously violates the landing separation requirement. Therefore, aircraft within SCA need to be considered when dynamic re-scheduling occurs so that spacing criteria within SCA will be assured.

3. Pub/Sub System Optimality

Recall that we claimed earlier in chapter I that research on aircraft scheduling can be roughly divided into two areas. One area determines efficient scheduling algorithms, and the other studies

performance potentials and overall strategies of automated aircraft scheduling. This research is of the latter and its main focus is to establish a robust and operational effective communication paradigm that would well handle the information diffusion problem due to distributed coordination in the decentralized operational environment. Pub/sub communication paradigm is chosen not only because it provides a loosely coupled communication scheme that fits well to the highly dynamic nature of the operational environment of the decentralized aircraft landing operations, but it also represents a general-purpose solution for information dissemination that can fit the scenarios that require an asynchronous many-to-many communication, which is exactly the desirable feature in the decentralized air traffic management system. In general, optimality analysis of a pub/sub system involves with information routing optimization, which is beyond the scope of this research, and it is actually not necessary for this research since the pub/sub system established here has a very limited number of information routes. However, the simplex algorithm used in the on-board aircraft landing scheduling tool introduces some optimization concerns to the established pub/sub system in this research.

The simplex algorithm implemented in this research is as follows:

1) Rewrite the aircraft landing scheduling model into standard vector form of a linear programming problem (refer to equations 10 and 11 in chapter III).
2) Transfer the linear programming problem to be in augmented form:

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & I \end{bmatrix} \begin{bmatrix} Z \\ X \\ X_S \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \qquad X, X_S \geq 0 \tag{19}$$

Where X are the variables from the standard form, X_S are the introduced slack variables from the augmentation process, c contains the optimization coefficients, A and b describe the constraint set, and Z is the objective function to be maximized.

3) At any iteration of the simplex algorithm, the tableau will be of this form:

$$\begin{bmatrix} 1 & c_B^T B^{-1} A - c^T & c_B^T B^{-1} \\ 0 & B^{-1} A & B^{-1} \end{bmatrix} \begin{bmatrix} Z \\ X \\ X_S \end{bmatrix} = \begin{bmatrix} c_B^T B^{-1} b \\ B^{-1} b \end{bmatrix} \tag{20}$$

Where c_B is the coefficients of basic variables (non-zero values) in the c-matrix; and B is the columns of $[A \ I]$ corresponding to the basic variables.

4) Choose an initial basic feasible solution, if it is not an optimal solution, do the following:

- Determine direction of highest gradient: Choose the variable associated with the coefficient in $c_B B^{-1} A - c$ that has the highest negative magnitude. This basic variable, which we call the entering basic, will be increased to help maximize the objective function.
- Determine maximum step length: Use the $\begin{bmatrix} B^{-1} A & B^{-1} \end{bmatrix} \begin{bmatrix} X \\ X_S \end{bmatrix} = B^{-1} b$ sub-equation to determine which basic variable reaches zero first when the entering basic is increased. This variable, which we call the leaving basic then becomes non-basic (zero value). This operation only involves a single division for each basic variable, since the existing basic-variables occur diagonally in the tableau.
- Rewrite problem: Modify B and c_B to account for the new basic variables. This will automatically make the tableau diagonal for the existing and new basic variables.
- Check for improvement: Repeat procedure until no further improvement is possible, meaning all the coefficients of $c_B B^{-1} A - c$ are positive. Procedure is also terminated if all coefficients are zero, and the algorithm has walked in a circle and revisited a previous state.

It is worthy mentioning that the simplex algorithm implemented in this research is originated from Ref. [54]. The algorithm starts with an initial basic feasible solution and tests its optimality. If some optimality condition is verified, then the algorithm terminates. Otherwise, the algorithm

identifies an adjacent basic feasible solution, with a better objective value. The optimality of this new solution is tested again, and the entire scheme is repeated, until an optimal basic feasible solution is found. Since every time a new basic feasible solution is identified such that the objective value is improved, and the set of basic feasible solution is finite, it follows that the algorithm will terminate in a finite number of steps (iterations).

It is also interesting to examine the geometrical interpretation of the behavior of Simplex algorithm. Given the above description of the algorithm and the correspondence of basic feasible solution to extreme points, it follows that Simplex essentially starts from some initial extreme point, and follows a path along the edges of the feasible region towards an optimal extreme point, such that all the intermediate extreme points visited are improving (more accurately, not worsening) the objective function.

D. Conclusions

Addressed in this chapter are the approach and issues of implementing decentralized aircraft landing operations at the terminal area of non-controlled airports, and thus provides a clear approach to the distributed air traffic management system. An on-board aircraft landing scheduling tool, resulted from the integration of an aircraft landing scheduling model and distributed coordination function, was implemented to achieve dynamic self-scheduling in the ultimate uninterrupted free-flight operational environment. Distributed coordination issues, which pose most of the technology challenges to the decentralized aircraft landing operation, were addressed. Two coordination models, with focus on event-based coordination model, were discussed for comparison. Methodologies description of how to resolve challenges brought by distributed coordination issues using the event-based coordination model was provided, from the establishment of the mathematical model to the application implementation.

CHAPTER V

REAL-TIME SIMULATION METHODOLOGY AND NUMERICAL RESULT

Flight simulation methodologies are essential for evaluating the performance of the air traffic static aircraft landing scheduling algorithms used in the Air Traffic Control automation system, and the dynamic aircraft landing scheduling algorithms used in the distributed Air Traffic Management system, described in chapter III and IV respectively. In this chapter we first address the approach of determining customized simulation architecture suitable for this research, followed by how the final simulation architecture selection is implemented. Finally, numerical examples for static and dynamic cases respectively are presented for Monte Carlo, real-time flight simulation.

A. Real-Time Pilot-In-The-Loop Simulation

During the past eight years, extensive research work has been done in the Texas A&M University Flight Simulation Laboratory (FSL) on designing and developing intelligent cockpit systems and pilot decision-aiding tools for General Aviation (GA) aircraft using fix-based flight simulation validation and evaluation [55]. The Engineering Flight Simulator (EFS), and the Automated Safety and Training Avionics (ASTRA) combined provide a pilot-in-the-loop real-time simulation environment that can be adjusted to fit different simulation fidelity requirements.

1. Engineering Flight Simulator

The EFS is a real-time, nonlinear, six degree-of-freedom fixed base pilot-in-the-loop simulator powered by an SGI ONYX Reality II workstation with one R4400 processor chip and 256 MB RAM [56]. It contains a T-37 style cockpit with reconfigurable multifunction displays that can be rapidly modified and tailored to fit individual project needs for a wide range of general aviation, commercial, and military cockpit displays. The external environment is displayed on a three-panel projection surface that allows the pilot a field of view of 75 degrees vertically and 155

degrees horizontally [56]. The EFS is currently configured to simulate a Rockwell Commander C700, which is a light twin General Aviation (GA) aircraft. Figure 18 shows a pilot operating the EFS.

Figure 18: Cockpit and External Display of Real-Time Engineering Flight Simulator

2. Pilot Advisor and Training System

The Automated Safety and Training Avionics (ASTRA) is a computerized airborne expert system developed in a previous research program [57]. It is used to assess the pilot's flying performance, and issues recommendations for pilot actions in all flight phases from take off to landing. It infers the flight mode of an aircraft from sensed flight parameters using fuzzy logic methods. The pilot's flying performance is assessed based on the interpreted flight mode, an embedded knowledge base, and pilot inputs. Recommendations are then issued for pilot actions. Such a system improves safety by enhancing situational awareness, and reducing the cost and time required to achieve and maintain pilot proficiency. Figure 19 illustrates the modular design of ASTRA and the interfaces between the software components and the necessary supporting hardware.

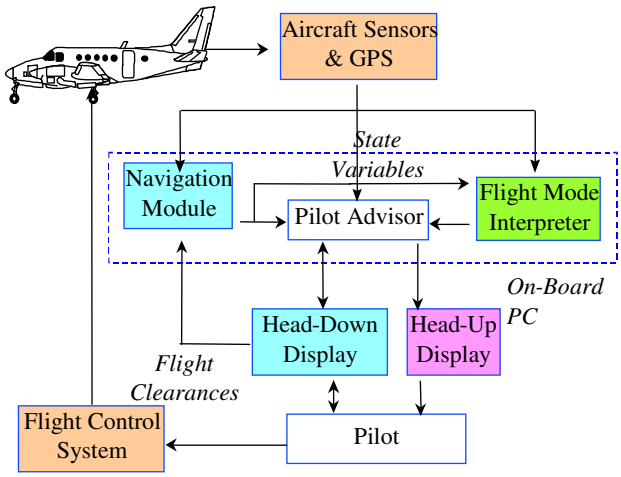

Figure 19: Automated Safety and Training Avionics Architecture

B. Fast-Time Multiple-Agent System Simulation

Multiple-agent system simulation is the candidate simulation architecture that achieves time and money cost effectiveness, especially when the operation concepts and algorithms are still under prototyping phase. In general, agent-based simulation model represents simulation architecture for modeling and simulation of complex systems consisting of entities of different behavior, and multiples of these different entities interact with each other in significant ways. Usually the most important feature of agent-based simulation model is the efficient integration of different entities interacting with each other. Different modeling and simulation approaches used must be integrated in a consistent manner so that the entire system can be simulated with an appropriate speed/accuracy tradeoff.

The main purpose of the research addressed in this paper is to explore the feasibility and capability of aircraft landing scheduling within uninterrupted dynamic free-flight environment at the terminal area of non-controlled airports. At the early stage of this research, pilot-in-the-loop flight simulation was original designed to achieve better simulation fidelity. However, considering the fact that the operation concepts introduced in this research was original and the feasibility analysis was more of concern than high fidelity simulation performance and human

factors at the time when it was under prototyping, multiple-agent system simulation emerged as a better candidate. As research approached in progress, the advantages of agent-based simulation took effect since it shortened the algorithms development and modification cycle considering the fact that it'd take huge amount of time and resource to conduct pilot training for the new operation concepts and thus execute pilot-in-the-loop simulation. It then came to the point that multiple-agent system simulation was chosen as the final simulation architecture. However, the simulation system in the FSL was not able to provide multiple-agent simulation environment that was required to validate the operation concepts and scheduling algorithms in this research. A multiple-agent simulation platform, named Air Traffic Information Management System (AIMS) was then developed to meet the simulation requirements for this research. In order to make AIMS an open simulation platform that satisfies the comprehensive simulation requirements for different kinds of Air Traffic Management (ATM) research projects, it was built as a pure agent-based and plug-in modular simulation system. Fig. 20 shows the hierarchy architecture of AIMS.

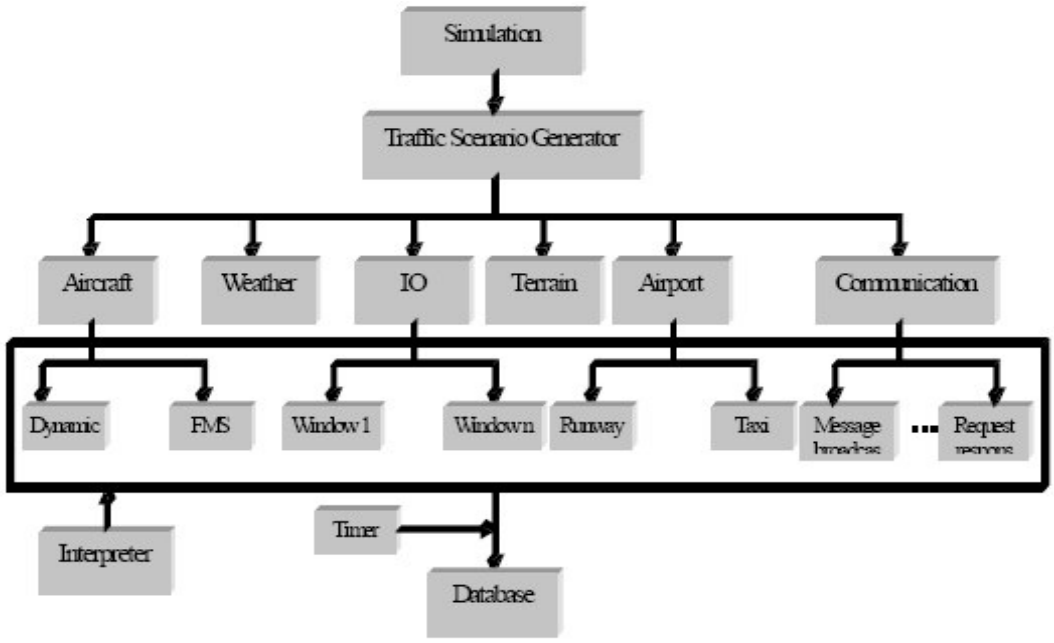

Figure 20: AIMS Hierarchy Architecture

AIMS is composed of four components:

1. Traffic Scenario Generator

The traffic scenario generator sets up the initial conditions and settings for each simulation, such as the initial configuration and conditions of each aircraft, the aircraft models, CD&R model, position, velocity, etc.; the airspace domain, either en-route airspace or terminal airspace; the weather and terrain configuration; and the simulation mode, either fast-time or real-time mode. Uncertainly factors are also introduced to provide the Monte Carlo simulation characteristics. The traffic scenario generator generates traffic scenarios based on either a combination of actual congested air traffic data, and simulated traffic data created by some known traffic distribution functions. Realistic traffic data can be obtained by using flight plans filed at the Air Route Traffic Control Center (ARTCC) host computer. Simulated traffic data is complemented to real traffic data to create some reasonable level of air traffic volume.

2. Intelligent Aircraft Agent

Each aircraft in AIMS is implemented as an intelligent aircraft agent. The aircraft agent developed in this research is designed and implemented for FAR 23 Class GA aircraft under Free Flight conditions. Most FAR 23 Class GA aircraft in use today, however, are not equipped with the standard navigation and communication devices commonly found on commercial air transportations. As the original operation concept introduced in this research is designed for a future implementation of Free Flight, it is anticipated that this advanced equipment will be available on GA aircraft at that time. Therefore, in this research it then assumes that the aircraft agents implement real-time flight operations with the aid of the following on-board devices: ADS-B devices, so that the aircraft has the access to current traffic information; Flight Management System (FMS), so that the aircraft can utilize the traffic information from ADS-B devices for Conflict Detection & Resolution (CD&R) and landing scheduling; and a communication device that gives the aircraft the capability to communicate with other aircraft. It should be noted that the CD&R algorithm not only maintains a distance-based separation in the

en-route phase of flight, but also enforces a time-based spacing in the final approach. It is assumed that each aircraft agent is equipped with such on-board CD&R module to plan maneuvers for an optimized and conflict-free trajectory. It is also assumed that the on-board CD&R module is sufficient to provide self-separation advisories for pilots since this research targets issues of aircraft landing scheduling operation only. Note that the ASTRA, the computerized airborne expert system introduced in section A, is integrated with aircraft agent to enhance the agent functionality. Figure 21 shows the architecture of an intelligent aircraft agent.

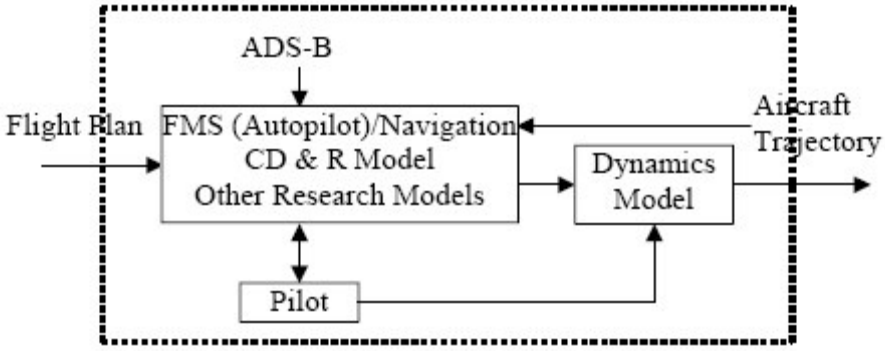

Figure 21. An Intelligent Aircraft Agent

3. Airport Model

Implementation of an airport model is divided into four steps. First, the terrain configuration of an airport and its surrounding area is generated based on its one-degree U.S. Geological Survey (USGS) Digital Elevation Models (DEM). Second, weather-constrained airspace in the terminal area of an airport is generated based on real-time weather conditions. Third, data transfer and data interpretation functions are developed to take responsibility for the interactive actions between the intelligent aircraft agent and the airport.

4. Weather Model

Real weather data can be obtained from resources on the Internet. This data is either recorded data in the form of historical databases, or actual real-time data, e.g., those provided by National Oceanic and Atmospheric Administration (NOAA) or National Weather Service (NWS). Since

most of the weather data is discrete data points, the weather model is established by applying interpolation/extrapolation methods and ruled-based model identification methods.

At the early stage of operation concept and algorithm prototyping and designing, fast-time multiple-agent system simulation was conducted in order to shorten the development cycle time. The most important advantage of the fast-time simulation is that it allows a variety of experimental conditions to be varied "Monte Carlo-style". Such simulations enable rapid, iterative concept refinement, and help focus subsequent real-time simulations by identifying test scenarios and experimental conditions likely to provide clear insights. In this research fast-time simulation studies were conducted by introducing a variety of distribution functions to simulate the uncertainty encountered in the real air traffic, and then evaluate the algorithm performance by verifying how often the converged solutions are achieved when uncertainty are introduced. It was also used to identify the boundary conditions of the scenario sets for normal operations, separated from abnormal operations that are not the concern of this research, in the subsequent real-time multiple-agent system simulation.

C. Real-Time Multiple-Agent System Simulation

As stated earlier, feasibility and capability analysis of the original operation concepts and scheduling algorithms are more of interest than high fidelity simulation performance and human factors in this research. Originally the operation concepts, scheduling algorithms, and multiple-agent system simulation system were implemented using development platform provided by Visual C++ Development Studio. It provided good fast-simulation performance with its strong support for event-driven mechanism, where fast development of Graphic User Interface (GUI) was achieved to generate initial conditions for user-customized scenarios. It was also suitable for the fast-time simulation due to the significant user-machine interaction during the simulation-run. Fast-time multiple-agent system simulation provided sufficient information for feasibility analysis at early stage of algorithm prototyping and designing, such as verifying if the scheduling

algorithms rendered the converged solutions. However, it reached its bottleneck when the research approached to the phase of capability analysis, where time-sensitive performance metrics were introduced to evaluate the effectiveness and efficiency of the operation concepts and scheduling algorithms. That was when an upgrade from fast-time simulation to real-time simulation emerged as a solution.

There are several commercial tools in the market supporting real-time simulation, such as Matlab by MathWorks and RT-LAB by Opal-RT. Considering the powerful feature of easy integration of user-customized code into Matlab Simulink toolbox which provides an interactive graphical environment for fast development, Matlab was selected for the real-time multiple-agent system simulation implementation. Four Matlab toolboxes were involved: Simulink, Stateflow, Real-Time Workshop, and Real-Time Windows Target.

1. Matlab Toolbox – Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports both linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two [58]. Simulink supports multiple rate system which is one of the essential features suitable for this research since the simulation system was designed to have different rates, such as sample rate for scheduling algorithm computation was expected to have higher rate than ADS-B transmission.

Simulink provides a GUI for building models as block diagrams, using click-and-drag mouse operations [58]. From the perspective of prototyping and designing, this is far more efficient that the other simulation packages that require the users to formulate differential equations and difference equations in a language or program. Simulink includes a comprehensive library of reusable block diagram for direct use such as sinks, sources, linear and nonlinear components, and connectors. Another essential feature for this research is Simulink provides users with the capabilities of customizing and creating user-defined blocks through its S-function mechanism. An S-function is a computer language description of a Simulink block, and it can be written in

Matlab, C, C++, Ada, or Fortran. Users can write a piece of computer program in their favorite language such as C++, and then convert it into S-function by following a set of simple rules. S-function is compiled as MEX-file which is one kind of the Matlab-defined executables that can be running in Matlab environment. S-functions use a special calling syntax that enables themselves to interact with Simulink equation solvers. The form of an S-function is very general and can accommodate continuous, discrete, and hybrid systems.

Simulink also provides the hierarchical architecture for model design so that users can build models using both top-down and bottom-up approaches [58]. A classic block diagram model of a dynamic system graphically usually consists of blocks and lines (signals). The history of these block diagram model is derived from engineering areas such as Feedback Control Theory and Signal Processing. The relationships between each elementary dynamic system in a block diagram are then illustrated by the use of signals connecting the blocks. Users can view the system at a high level, and then double-click blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact.

One last essential feature of Simulink provided for this research is its powerful model analysis tools include linearization and trimming tools, which can be accessed from the Matlab command line, plus the many tools in Matlab and its application toolboxes. For example, linearization tool was used to a huge amount in the aircraft agent design phase. And because MATLAB and Simulink are integrated, users can simulate, analyze, and revise the models in either environment at any point.

2. Matlab Toolbox – Stateflow

Stateflow is an interactive graphical design tool that works with Simulink to model and simulate event-driven systems, also called reactive systems. Event-driven systems transition from one operating mode to another in response to events and conditions [58]. These systems are often used to model logic for dynamically controlling a physical device. As addressed earlier in

chapter IV, the event-based coordination model used in the scheduling process was implemented using Stateflow.

Event-driven systems can be modeled as finite-state machines. A finite state machine is a representation of an event-driven (reactive) system. In an event-driven system, the system makes a transition from one state (mode) to another prescribed state, provided that the condition defining the change is true [58]. Traditionally, designers used truth tables to represent relationships among the inputs, outputs, and states of a finite state machine. The resulting table describes the logic necessary to control the behavior of the system under study. Another approach to designing event-driven systems is to model the behavior of the system by describing it in terms of transitions among states. The state that is active is determined based on the occurrence of events under certain conditions. State-transition diagrams and bubble diagrams are graphical representations based on this approach. A Stateflow diagram is a graphical representation of a finite state machine, where states and transitions form the basic building blocks of the system. Additionally, Stateflow enables the representation of hierarchy, parallelism, and history. Hierarchy enables the users to organize complex systems by defining a parent/offspring object structure. For example, users can organize states within other higher-level states. A system with parallelism can have two or more orthogonal states active at the same time. History provides the means to specify the destination state of a transition based on historical information. These characteristics enhance the usefulness of this approach and go beyond what state-transition diagrams and bubble diagrams provide.

One important feature of Stateflow that is critical for this research is its seamless integration with Simulink. Stateflow charts run as blocks in a Simulink model. The Stateflow block connects to other blocks in the Simulink model by input and output signals. Through these connections, Stateflow and Simulink share data and respond to events that are broadcast between model and chart. [58]. Because of this feature, users can develop their Stateflow chart before or after the Simulink model in which it will run. Stateflow comes with its own editor and debugger,

which allows users to simulate and test the chart logic before it is integrated with a Simulink model. Users can test a Stateflow chart independently of its parent model by attaching a Simulink Source block as an input and a Simulink Sink block as an output.

3. Matlab Toolbox – Real-Time Workshop

Real-Time Workshop is an extension of capabilities of Simulink and Matlab that automatically generates, packages, and compiles source code from Simulink models to create real-time software applications on a variety of systems. By providing a code generation environment for rapid prototyping and deployment, Real-Time Workshop is the foundation for production code generation capabilities [58]. Along with other tools and components from The MathWorks, Real-Time Workshop provides the following features:

- Automatic code generation tailored for a variety of target platforms.
- A rapid and direct path from system design to implementation.
- Seamless integration with Matlab and Simulink.
- A simple graphical user interface.
- An open architecture and extensible make process.

Real-Time Workshop generates optimized, portable, and customizable ANSI C or C++ code from Simulink models to create standalone implementations of models that operate in real-time and non-real-time in a variety of target environments. Generated code can run on PC hardware, microcontrollers on bare-board environments, and with commercial or proprietary real-time operating systems (RTOS) [58]. Real-Time Workshop is a key link in the set of system design tools providing a real-time development environment – a direct path from system design to hardware implementation. Users can streamline application development and reduce costs with Real-Time Workshop by testing design iterations with real-time hardware. Real-Time Workshop supports the execution of dynamic system models on hardware by automatically converting models to code and providing model debugging support. It is well suited for accelerating

simulations, rapid prototyping, turnkey solutions, and production of embedded real-time applications.

Using integrated makefile-based targeting support, Real-Time Workshop builds programs that can help speed up simulations, provide intellectual property protection, and run on a wide variety of real-time rapid prototyping or production targets. Simulink's external mode run-time monitor works seamlessly with real-time targets, providing an elegant signal monitoring and parameter tuning interface [58]. Usually users can start with modeling in Simulink, followed by an analysis of the simulations in Matlab. During the simulation process, users use the rapid simulation features of Real-Time Workshop to speed up simulations. After users are satisfied with the simulation results, they can use Real-Time Workshop in conjunction with a rapid prototyping target, such as Real-Time Windows Target or xPC Target. The rapid prototyping target is connected to the physical system. Users test and observe the system, using the Simulink model as the interface to the physical target. Once it is verified that the simulation is functioning properly, users use Real-Time Workshop to transform the model to C or C++ code. An extensible make process and download procedure creates an executable for the model and places it on the target system. Finally, using external mode, users can monitor and tune parameters in real-time as the model executes on the target environment. There are two broad classes of targets: rapid prototyping targets and embedded targets. Code generated for the rapid prototyping targets supports increased monitoring and tuning capabilities. Code generated for embedded targets is highly optimized and suitable for deployment in production systems, and can include application-specific entry points to monitor signals and tune parameters. It is worthy mentioning that embedded system target is not used in this research since the complete simulation system is PC-based.

4. Matlab Toolbox – Real-Time Windows Target

Real-Time Windows Target is a PC solution for prototyping and testing real-time systems. It is an environment where users can use a single PC as a host and target. In this environment users

use the desktop or laptop PC with Matlab, Simulink, and Stateflow to create models using Simulink blocks and Stateflow diagrams [58]. After creating a model and simulating it with Simulink in normal mode, users can generate executable code with Real-Time Workshop using the Real-Time Windows Target option. Then users can run their application in real time with Simulink external mode. Integration between Simulink external mode and the Real-Time Windows Target allows users to use the Simulink model as a graphical user interface for Signal visualization and parameter tuning.

Real-Time Windows Target uses a small real-time kernel to ensure that the real-time application runs in real time. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of time:

- Timer interrupt: The kernel intercepts the interrupt from the PC clock before the Windows operating system receives it. This blocks any calls to the Windows operating system. As a result, the kernel is able to give the real-time application the highest priority available. To achieve precise sampling, the kernel reprograms the PC clock to a higher frequency. Because the PC clock is also the primary source of time for the Windows operating system, the kernel sends a timer interrupt to the operating system at the original interrupt rate [58].
- Scheduler: The timer interrupt clocks a simple scheduler that runs the executable. The number of tasks is equal to the number of sampling periods in the model with multitasking mode. With single-tasking mode, there is only one task. The maximum number of tasks is 32, and faster tasks have higher priorities than slower tasks. For example, a faster task can interrupt a slower task. During execution, the executable stores data in buffers. Later, the data in these buffers is retrieved by the Scope block. The scheduling, data storing, data transferring, and running the executable all run at CPU ring zero [58].
- Communication with hardware: The kernel interfaces and communicates with I/O hardware using I/O driver blocks, and it checks for proper installation of the I/O board. If

the board has been properly installed, the drivers allow the real-time application to run. Drivers also run at CPU ring zero.

- Simulink external mode: Communication between Simulink and the real-time application is through the Simulink external mode interface module. This module talks directly to the real-time kernel, and is used to start the real-time application, change parameters, and retrieve scope data. Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, users can then edit the parameter values. Users must close the dialog box to have the changes take effect and allow Simulink to continue.
- Built-in C compiler: Real-Time Windows Target applications are compiled with the Open Watcom C/C++ compiler. Therefore, no third-party compilers are necessary.

5. Real-Time Multiple-Agent System Simulation Implementation

Up so far in this chapter, we've already addressed how we determine the multiple-agent system simulation is the most suitable simulation architecture for this research considering development cost, time and money resource. Description of the chosen toolset, Matlab, and features of its four toolboxes used in this research are also provided in previous sections. In this section we shall address how the multiple-agent system simulation architecture is implemented and integrated with the operation concepts and scheduling algorithms development process. It is shown in Figure 22.

Early in the design process, Simulink and Stateflow were used to help formulate objectives, problems, and constraints to create the initial design. Originally the operation concepts, scheduling algorithms, and multiple-agent system simulation system were implemented using development platform provided by Visual C++ Development Studio. They were converted into Simulink with some different means. Considering the simulation fidelity requirement of aircraft dynamics was not high, the intelligent aircraft agent was completely re-designed in Simulink taking advantage of its Aerospace Blockset which provided of a variety of reusable libraries and functions for easy low-middle fidelity aircraft dynamics implementation. Operation concepts that

include all of the process-related codes were re-designed using Stateflow, which actually turned out to be really straightforward due to the GUI feature of Stateflow. Scheduling algorithms required the best execution performance, and were thus converted to S-functions in Simulink with only interfaces to Simulink added while keeping the original C code of excellent execution performance inside S-function blocks. In the phase of problem formulation, system design, detailed design, and re-designing process, it was clearly noticed that Simulink and Stateflow provided the ability to simplify and accelerate most phases of these software development processes, and at the same time to eliminate repetitive and error-prone tasks. The complete simulation system design was finally implemented using built-in blocks from the Simulink and Stateflow libraries, incorporate specialized blocks from the other Matlab toolboxes such as Aerospace, Communications, and Signal Processing, and some customized S-function blocks converted from the original C codes developed in the Visual C++ Development Studio.

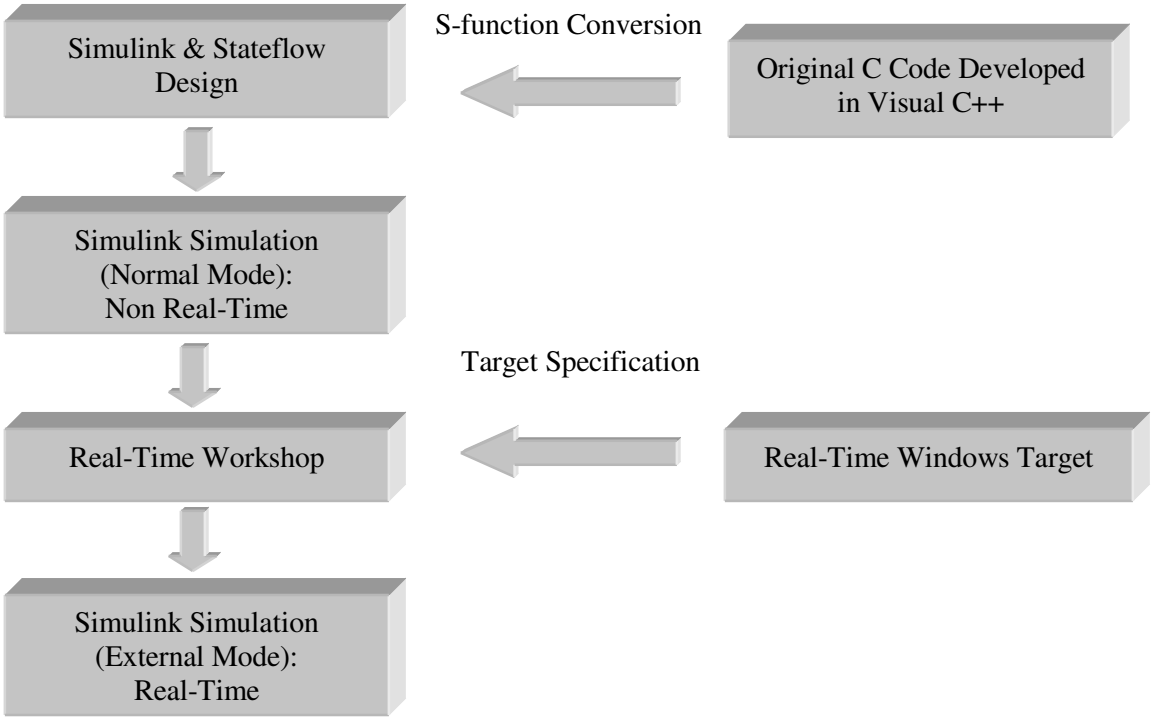

Figure 22: Real-Time Simulation System Implementation

Real-Time Workshop was used with target specified as Real-Time Windows Target to complete the design process. It closed the rapid prototyping loop by generating and optimizing code for given tasks and PC environments configured in the Real-Time Windows Target settings. Real-Time Workshop speeded up models by enabling high-speed simulations via Simulink Accelerator and by model referencing, which includes models in other models as blocks. Model's parameters were tuned using the Real-Time Workshop and Real-Time Windows Target for real-time Monte Carlo simulations.

D. Numerical Examples – Static Case

The objective is to perform Real-Time Multiple-Agent System Simulation, and compare the efficiency and effectiveness of the static aircraft landing scheduling algorithms used in the Air Traffic Control automation system addressed in chapter III. The aircraft landing scheduling model developed in chapter III is incorporated into Automated Safety and Training Avionics (ASTRA), a real-time computerized airborne expert system which is used here to generate scenarios with multiple intelligent aircraft agents [59]. Each agent is equipped with simulated Automatic Dependent Surveillance-Broadcast (ADS-B) devices so that they can collect information on the nearby traffic situation, and scheduling decisions from the scheduler. The agents also implement a modified SATS Conflict Detection & Resolution (CD&R) algorithm module to plan maneuvers for an optimized and conflict-free trajectory when following the scheduling decision issued by the scheduler to complete the landing. It should be noted that the CD&R algorithm not only maintains a distance-based separation in the en-route phase of flight, but also enforces a time-based spacing in the final approach. Complete knowledge of all aircraft to be sequenced is assumed, and is defined here as aircraft state information (altitude, longitude, latitude, airspeed, vertical speed, and heading), time stamp, flight path intent information (next two waypoints on the intended flight path), fuel status, and emergency priority.

The operational objective for all scenarios is to schedule the landing of between four and ten aircraft over a one hour time period at a non-controlled airport, using performance metrics of total cost of deviation, total holding time, and total delay time of feeder route. For the FCFS scheduling approach, the landing sequence is determined by the order in which each aircraft reaches the scheduling point. The scheduling solution values (performance metrics) are found by scheduling each aircraft at its preferred landing time, provided it is feasible. If it is not feasible, then each aircraft is scheduled as early as possible. For the optimal scheduling approach, both the landing sequence and scheduling solution values are solely determined by minimizing the corresponding objective function. The penalty cost fi and gi are set depending on the aircraft type. Four categories of test scenario are evaluated, involving 4, 6, 8, and 10 aircraft initially placed outside the SCA of TSTC Waco Regional Airport (KCNW), Waco, TX. The initial locations of each aircraft are determined randomly according to a normal distribution with a mean distance of 25 nautical miles to their assigned IAFs, and a standard deviation of 5 nautical miles. Each aircraft in the scenario has three speed profiles: cruise speed is the fastest and determines the earliest landing time; approach speed is the slowest and determines the latest landing time; and holding speed is used to determine the preferred landing time. Aircraft types used are heavy, medium, or light, with occurrence in any given scenario randomly generated with probabilities of 0.2, 0.4 and 0.4 respectively. The separation time requirements for different scenario setups are then set accordingly, e.g. the separation time requirement for a heavy-heavy case is five minutes, and two minutes for a light-light case. Figure 23 and Table 4 provide the graphical configuration and detailed information of a typical test scenario example respectively.

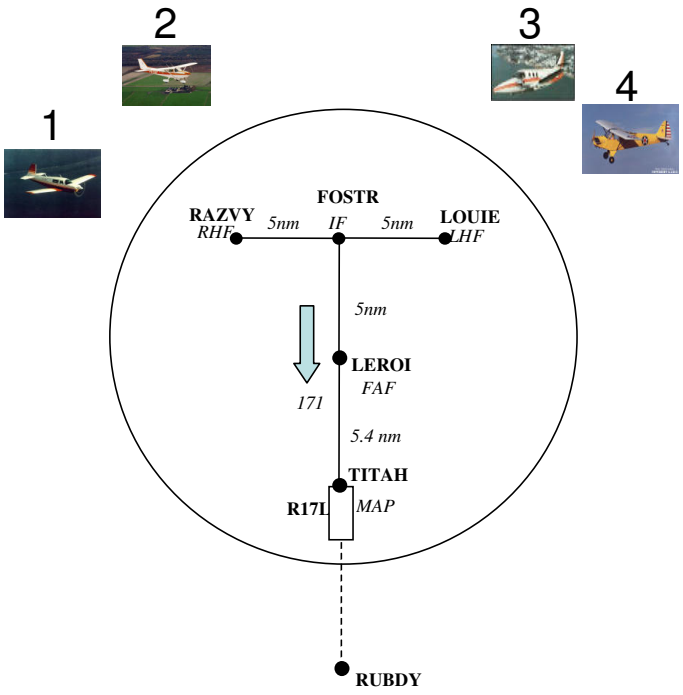

Figure 23: Generalized Approach Plate for a Typical Test Scenario

Table 4: Detailed Information of a Test Scenario

	1	2	3	4
Aircraft Type	Light	Medium	Heavy	Light
Flight ID	N5000L	N865CP	N700AE	N3998Z
Latitude	31.916546	32.116023	32.130356	32.016731
Longitude	-97.676885	-97.435805	-96.539107	-96.563069
Altitude	7000(FT)	5000(FT)	6000(FT)	4000(FT)
Heading (TRUE)	107(DEG)	144(DEG)	205(DEG)	225(DEG)
Planned IAF	RAZVY	RAZVY	LOUIE	LOUIE
Distance to IAF	25.9 (NM)	22.0 (NM)	29.0 (NM)	24.3 (NM)
Cruise Airspeed	180 (KNTS)	125 (KNTS)	150 (KNTS)	75 (KNTS)
Hold Airspeed	110 (KNTS)	100 (KNTS)	110 (KNTS)	65 (KNTS)
Approach Airspeed	80 (KNTS)	80 (KNTS)	90 (KNTS)	50 (KNTS)

A Monte Carlo approach was used totaling 960 runs, consisting of 40 individual scenarios for each of the four scenario classes evaluated (4, 6, 8, and 10 aircraft) run six times to get FCFS THT, optimal THT, FCFS TCD and total delay time of feeder route, and optimal TCD and total delay time of feeder route respectively. The 960 runs was found to be sufficient to highlight the desired trends, since the standard deviation for each performance metric was inversely proportional to the number of runs, and was decreasing by less than 5% per every ten runs thereafter.

Results for the four scenarios are presented in Table 5 and Figures 24-26. Note that all test scenarios produce converged solutions for both the FCFS scheduling and optimal scheduling algorithms. Compared with the FCFS scheduling, optimal scheduling is seen to significantly decrease the total cost of deviation by an average of 56.42%, decrease total holding time by an average of 52.16%, and decease total delay time of feeder route by 34.21%. It is noteworthy that test scenarios which involve several aircraft do not necessarily take more time to complete than those with fewer aircraft, despite the accumulation of landing operation time. Since the scheduling solution value is found by scheduling each aircraft at its preferred landing time (if it is feasible for the FCFS approach), it is possible for the FCFS scheduling to provide the optimal solution in situations where it obtains the optimal sequence. This results from the randomness introduced in generating the test scenarios.

Table 5. Numerical Results of Test Scenarios

Number of Aircraft in Scenario		4	6	8	10
Simulation Runs		40	40	40	40
FCFS solution TCD	Mean	13.95	19.64	30.17	36.52
	Standard Deviation	2.12	2.84	2.93	3.28
Optimal solution TCD	Mean	4.90	9.15	13.13	17.92
	Standard Deviation	0.54	1.09	1.18	1.99
FCFS solution THT (minutes)	Mean	21.11	30.73	36.04	48.55
	Standard Deviation	2.32	3.00	4.03	5.14
Optimal solution THT (minutes)	Mean	9.57	13.75	18.08	24.81
	Standard Deviation	0.79	1.57	1.95	2.26
FCFS solution $TDT_{feeder\ route}$ (minutes)	Mean	6.87	10.65	15.74	20.95
	Standard Deviation	1.03	1.43	1.92	2.03
Optimal solution $TDT_{feeder\ route}$ (minutes)	Mean	5.32	7.16	9.23	12.54
	Standard Deviation	0.47	0.69	1.22	1.45

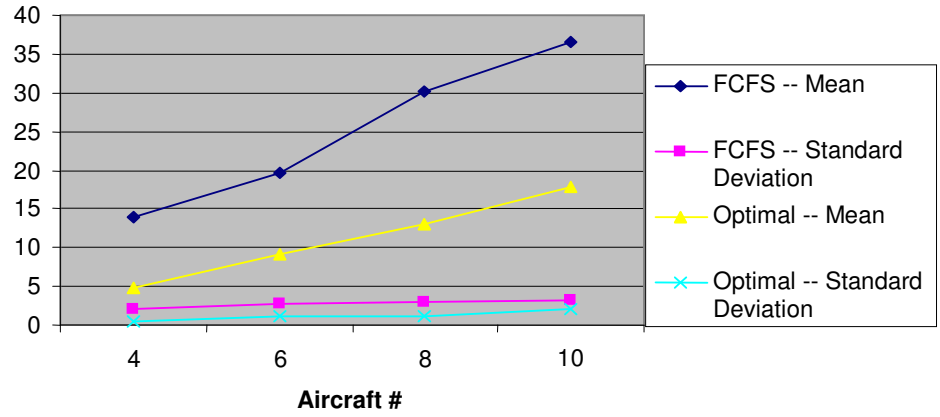

Figure 24: FCFS Vs. Optimal -- TCD

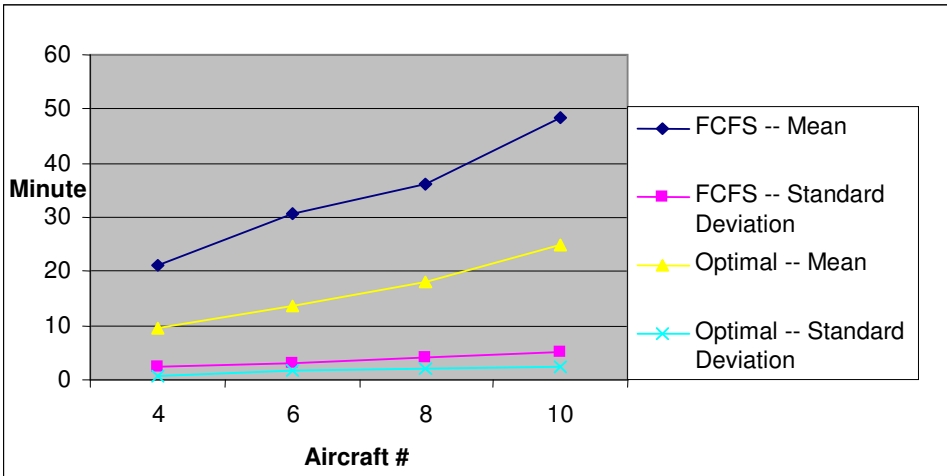

Figure 25: FCFS Vs. Optimal -- THT

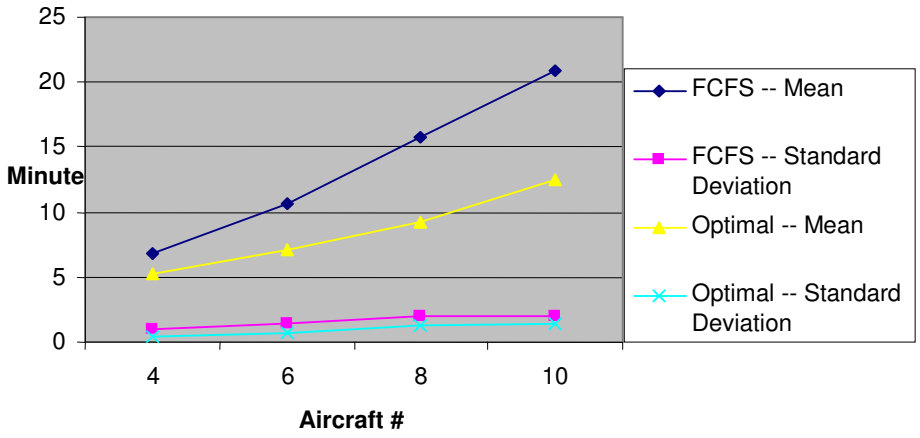

Figure 26: FCFS Vs. Optimal -- TDT (Feeder Route)

E. Numerical Examples – Dynamic Case

The objective of the numerical example is to evaluate the performance of the on-board aircraft landing scheduling tool used in the distributed Air Traffic Management system described in chapter IV. The scheduling model is integrated with the Automated Safety and Training Avionics (ASTRA), a real-time computerized airborne expert system and simulation environment that uses multiple intelligent aircraft agents [59]. Each agent is equipped with simulated ADS-B devices which provide traffic information and scheduling decisions from the scheduler (the arbitration aircraft for the on-board scheduling case and the automatic air traffic controller for the ground-based scheduling case). The agents also implement the modified SATS CD&R algorithm module to plan maneuvers for an optimized and conflict-free trajectory. It should be noted that the CD&R algorithm not only maintains a distance-based separation in the en-route phase of flight, but also enforces a time-based spacing in the final approach. Complete knowledge of all aircraft to be sequenced is assumed to be carried by the ADS-B messages and is defined here as aircraft state information (altitude, longitude, latitude, airspeed, vertical speed, and heading), time stamp, flight path intent information (next two waypoints on the intended flight path), fuel status, and emergency priority.

1. Scenario Design

The concept of the SATS scenario is used for the flight simulation since it breaks the current "one-in/one-out" procedure and enables multiple operations simultaneously at the terminal area of non-controlled airports. In SATS scenarios, a block of airspace named the Self-Controlled Area (SCA) is established around non-controlled airports. Multiple operations within an SCA can be achieved by having the aircraft hold in stacks at Initial Approach Fixes (IAFs) and then follow specified procedures (either a vertical entry or a lateral entry) to enter the SCA and complete approaches. This is shown in Figure 27.

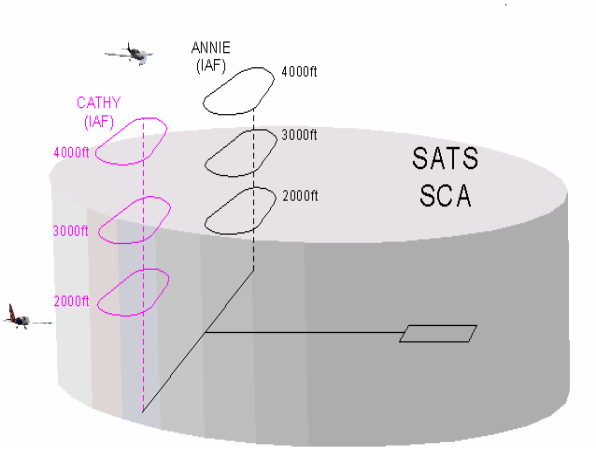

Figure 27: SATS Self-Controlled Area High-Volume Operations Concept II

It was concluded from the simulation result of the static case that the optimal scheduling algorithm can effectively enhance the operation efficiency compared with the first-come-first-serve scheduling algorithm, only the optimal scheduling algorithm is applied in each scenario.

Two categories of scenarios are designed to evaluate the static and dynamic response performance, respectively, where the ground-based scheduling tool serves as the baseline system for comparison. For static performance evaluation, basically for each scenario run both on-board and ground-based scheduling tool are triggered to apply optimal scheduling algorithm to compute the scheduling decisions when it reaches the scheduling point, and the scheduling decision

outputs from these two different resources are compared for conformance. A simple conformance ratio serves as the performance metric for the static performance evaluation.

For dynamic performance evaluation that considers the scheduling of aircraft landings in real-time, four sub-categories of test scenario consisting of 4, 6, 8, and 10 aircraft initially placed outside the SCA of TSTC Waco Regional Airport (KCNW), Waco, TX, are evaluated. Initial locations are determined randomly according to a Gaussian distribution with a mean distance of 25 nautical miles to the assigned IAF, and a standard deviation of five nautical miles. Aircraft types used are heavy, medium, or light, and appear randomly in any scenario with probabilities of 0.2, 0.4, and 0.4, respectively. Each aircraft has a cruise speed, approach speed, and holding speed. Separation time requirements for each scenario are set according to aircraft type, e.g. five minutes for a heavy-heavy case, and two minutes for a light-light case. Two dynamic cases are simulated to trigger the dynamic scheduling: new aircraft appearance, and operation environment condition changes that cause aircraft to drift from their planned flight path based on last scheduling decision and the resulting repositioning of needs to be considered. For simplification, the latter case is simulated simply by changing the aircraft speed profile. They are implemented by integrating a dynamic even trigger into the simulation system to trigger the desired dynamic case in real-time. Monte Carlo simulation was used on 40 individual scenarios for each of the four scenario classes. Three performance metrics stated earlier are used for the dynamic performance evaluation. To evaluate the impact of introducing spacing constraint into the scheduling model instead of on-board CD&R function (as proposed in chapter IV, section C-1), 10 additional scenarios for each of the four scenario classes were exercised where only TCD was evaluated for comparison.

2. Numerical Results

For static performance evaluation, the results show that the event-based coordination implementation can effectively implement the distributed decision-making on aircraft landing scheduling. All scenario runs output the same scheduling decisions from the on-board and

ground-based scheduling tool, and thus they gain a 100% conformance ratio. Observation of a computing time history also shows that the scheduling decision-making process execution time ratio of on-board scheduling tool and ground-based scheduling tool increases as number of aircraft in the scenario increases. This is a result from the extra processing steps taken in the on-board scheduling tool. When 10 aircraft are involved in the scenario and the ADS-B message transmission frequency is simulated at 10 Hz, the execution time of on-board scheduling (average around 0.12 Sec) exceeds the time period of ADS-B transmission cycle, which means that the scheduling decision is made on the out-of-dated aircraft state information since the new ADS-B message (represents the new aircraft state) is on the way of transmitting already. Scheduling decision conformance is still gained because of the ADS-B message lock mechanism. This issue can be directly solved by using a faster computer for the simulation. However, it should draw our concern that it somehow simulates some type of "delay time" (such as delay caused by pilot-machine interaction or ADS-B message transmission delay) if pilot-in-the-loop simulation is conducted.

For dynamic performance evaluation, results for the four scenario classes are presented in Table 6, Figure 28, and Figure 29, where the ground-based scheduling tool serves as baseline for comparison. All test scenarios produce converged solutions for on-board scheduling tool. Numbers of dynamic case used to trigger re-scheduling are 2, 3, 4, and 5 for test scenarios involved with 4, 6, 8, and 10 aircraft, respectively, compared to the ground-based scheduling values of all three performance metrics for on-board scheduling increase to some extent but without directly traceable statistical law. However, the ratio of mean value and standard deviation shows that the standard deviation increasing rate is obviously greater than the mean value increasing rate, comparing to the ground-based scheduling tool. These observations are resulted from the fact that the dynamic case (new aircraft entry or existing aircraft speed profile change) type and triggering time are both randomly selected in each scenario. Simulation results also show that on-board scheduling tool reaches its run-time bottleneck when 10 aircraft are

involved in the scenario and the ADS-B message transmission frequency is around 20 Hz. This results from the ADS-B message processing overload. Again, this issue can be directly solved for short-term by using a faster computer. For the long-run, ADS-B message complexity analysis and re-organization should be conducted for a better message parsing and processing performance.

Evaluation of impact of introducing spacing constraint in the scheduling model is shown in result Table 7. For most of the scenarios, the scheduling model that includes the spacing constraint (hereafter refer as augmented scheduling model) achieves slight better performance than the one without spacing constraint (hereafter refer as baseline scheduling model). Numerical results also show that the mean/standard deviation ratio of the augmented scheduling model is slightly smaller than the one of the baseline scheduling model for most of the scenarios. These are all resulted from the fact that the spacing constraint represents the separation requirement within the entire SCA whereas the landing separation constraint alone in the baseline scheduling model only represents accumulated separation at the runway threshold. The placement of spacing constraint in the scheduling model thus makes it more active and responsive to the dynamic re-scheduling events. However, there is no obvious difference between augmented scheduling model and baseline scheduling model since time-based spacing constraint is still enforced in baseline scheduling model via on-board CD&R functionality.

Table 6. Dynamic Performance Evaluation Numerical Results

Number of Aircraft in Scenario		4	6	8	10
Simulation Runs		40	40	40	40
Ground –Based TCD	Mean	4.90	9.15	13.13	17.92
	Standard Deviation	0.54	1.09	1.18	1.99
	Mean/Standard Deviation	9.07	8.39	11.12	9.00
On-Board TCD	Mean	6.11	11.92	17.67	24.28
	Standard Deviation	1.20	2.44	2.98	4.29
	Mean/Standard Deviation	5.09	4.88	5.92	5.66
Ground-Based THT (minutes)	Mean	9.57	13.75	18.08	24.81
	Standard Deviation	0.79	1.57	1.95	2.26
	Mean/Standard Deviation	12.11	8.76	9.27	10.98
On-Board THT (minutes)	Mean	13.55	18.66	23.41	32.84
	Standard Deviation	2.04	3.22	4.37	6.31
	Mean/Standard Deviation	6.64	5.80	5.36	5.20
Ground-Based	Mean	5.32	7.16	9.23	12.54

Table 6 Continued

	Standard Deviation	0.47	0.69	1.22	1.45
	Mean/Standard Deviation	11.32	10.38	7.57	8.65
	Mean	6.01	8.96	11.74	17.91
On-Board $TDT_{feeder\ route}$ (minutes)	Standard Deviation	1.02	1.39	2.46	2.95
	Mean/Standard Deviation	5.89	6.45	4.77	6.07

Table 7. Impact of Introducing Spacing Constraint into the Scheduling Model

Number of Aircraft in Scenario		4	6	8	10
Simulation Runs		10	10	10	10
On-Board TCD (spacing constraint is not included in the scheduling model)	Mean	4.92	9.23	12.68	18.56
	Standard Deviation	0.56	1.09	1.19	2.00
	Mean/Standard Deviation	8.79	8.47	10.66	9.28
On-Board TCD (spacing constraint is included in the scheduling model)	Mean	4.79	9.07	12.28	17.30
	Standard Deviation	0.54	1.11	1.16	2.12
	Mean/Standard Deviation	8.87	8.17	10.58	8.16

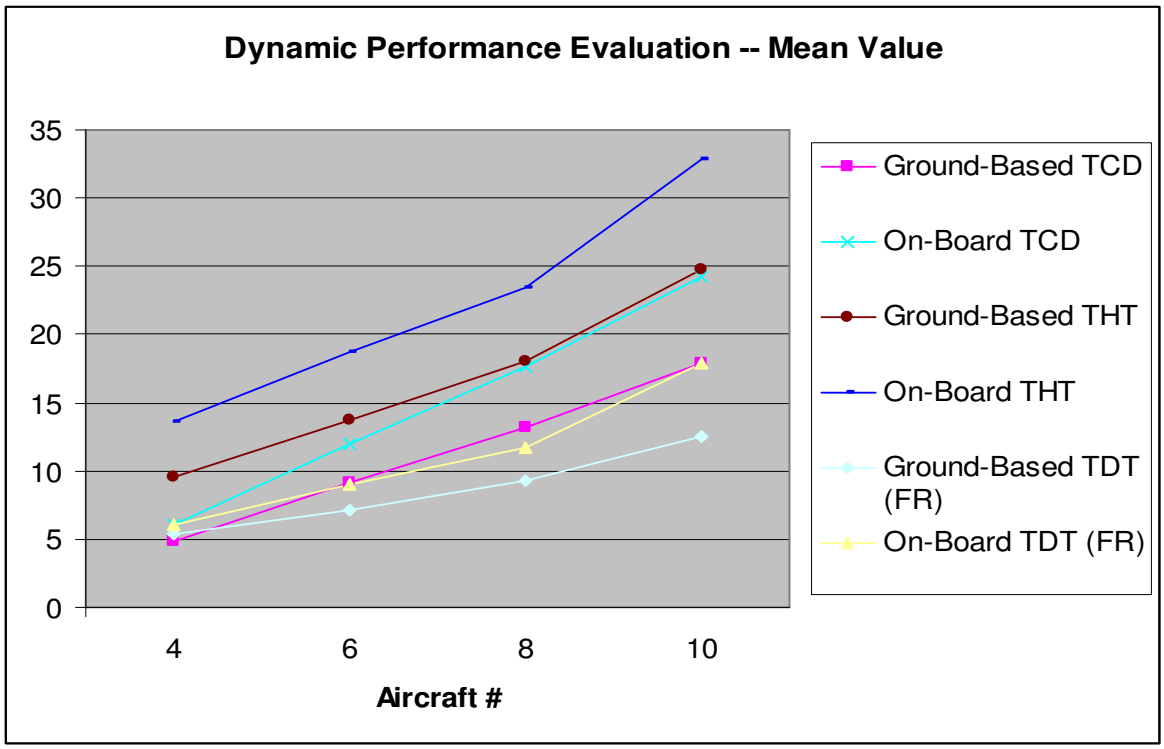

Figure 28: Dynamic Performance Evaluation – Mean Value

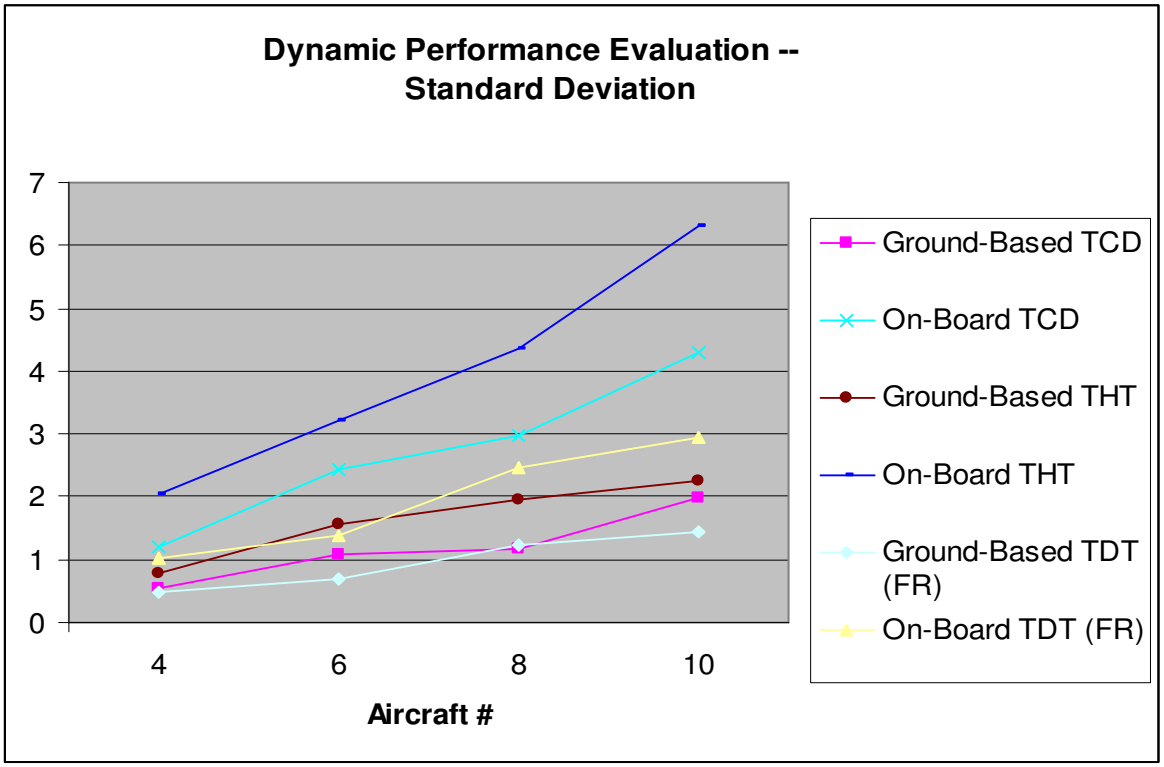

Figure 29: Dynamic Performance Evaluation – Standard Deviation

F. Conclusions

In this chapter we first reviewed several candidate simulation architectures for the performance evaluation of the operation concepts and scheduling algorithms introduced in this research. Real-time multiple-agent system simulation architecture was selected considering time and money resource, development cost, and requirements for simulation fidelity. Then the approach of implementing the lost-cost effective agent-based real-time simulation with Matlab toolset was provided. Finally, numeral results obtained from Real-time multiple-agent system simulation for static and dynamic cases were presented.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

It is becoming apparent that the existing air transport system is approaching a bottleneck. This mainly results from the dominant hub-and-spoke model that results in a concentration of a large percentage of the air traffic at a few hub airports. A promising solution is to distribute the congested air traffic at the hub airports to small airports where most of them do not have Air Traffic Control (ATC). Therefore, either an air traffic control automation system or a decentralized air traffic management system needs to be established to take over the aircraft landing sequencing and scheduling responsibilities. This dissertation aims to analyze the feasibility and capability of the decentralized aircraft landing scheduling operations at non-controlled airports. Firstly, this dissertation seeks to develop static optimization algorithms for aircraft landing scheduling, and analyze the capability of automated aircraft landing scheduling on single runways at non-controlled airports. Secondly, this dissertation seeks to analyze the degree of decentralization for aircraft landing scheduling in the dynamic operational environment at non-controlled airports. Finally, real-time multiple-agent system simulation is conducted to evaluate the performance of decentralized aircraft landing scheduling tool whereas air traffic control automation system serves as the baseline system for comparison.

A. Conclusions

In chapter II, we reviewed the functionalities of the operational components in the current ATM system. Then we gave a general description of the Free Flight concept and the SATS program. They both ultimately seek to achieve effective and efficient flight operations in the current NAS, with the former focusing on en-route flight operations, and the latter concentrating on terminal area operations at non-controlled airports.

In chapter III, an aircraft landing scheduling model and static optimization scheduling algorithms using linear programming and job shop solutions were developed and implemented as an air traffic control automation system for automated aircraft landing scheduling at single runway, non-controlled airports. Performance of the optimization algorithm was compared to a first-come-first-served scheduling algorithm in terms of total cost of deviation, total holding time, and total delay time of feeder route. Numerical results were presented in chapter V to support the development using four different multi-aircraft landing scenarios evaluated by Monte Carlo real-time flight simulation. Numerical results show that the optimal scheduling algorithm produced significant reductions in total cost of deviation (average of 56.42%), total holding time (average of 52.16%), and total delay time of feeder route (average of 34.21%) compared with first-come-first-serve scheduling. It was observed that in some scenarios, first-come-first-serve scheduling produced the optimal solution in situations where it happens to obtain the optimal sequence. Based on the reductions in total cost of deviation and total holding time for the test case ensemble considered, the optimal scheduling algorithm appears to be a promising candidate for enhancing the efficiency of aircraft landing operations at the terminal area of non-controlled airports.

In chapter IV, it addressed the approach and issues of implementing decentralized aircraft landing operations at the terminal area of non-controlled airports, and thus provided a clear approach to the distributed air traffic management system. An on-board aircraft landing scheduling tool, resulted from the integration of an aircraft landing scheduling model and distributed coordination function, was implemented to achieve dynamic self-scheduling in the ultimate uninterrupted free-flight operational environment. Distributed coordination issues, which posed most of the technology challenges to the decentralized aircraft landing operation, were addressed. Two coordination models, with focus on event-based coordination model, were discussed for comparison. Methodologies description of how to resolve challenges brought by distributed coordination issues using the event-based coordination model was provided, from the establishment of the mathematical model to the application implementation. Numerical results

were presented in chapter V to support the development using several different multi-aircraft landing scenarios evaluated by Monte Carlo real-time multiple-agent flight simulation. Numerical results showed that the event-based coordination implementation of the on-board aircraft landing scheduling tool achieved 100% scheduling decision conformance with ground-based scheduling tool regarding the static performance. The ADS-B message lock mechanism ensured the same static information was retrieved when both on-board and ground-based scheduling tool made scheduling decision at the specified scheduling points, and thus gained scheduling decision conformance even though the scheduling decision-making process execution time of on-board scheduling tool and ground-based scheduling tool were slightly different for some scenarios. Regarding the dynamic performance, numerical results showed that the on-board aircraft landing scheduling tool was able to provide the converged scheduling solutions dynamically in all nominal scenarios. However, it reached its run-time bottleneck in some abnormal scenarios such as when 10 aircraft were involved in the scenario and the ADS-B message transmission frequency is intentionally setup to abnormal 20 Hz. It was expected to be resolved after proper ADS-B message complexity analysis and re-organization were conducted for a better message parsing and processing performance.

In chapter V several candidate simulation architectures for the performance evaluation of the operation concepts and scheduling algorithms introduced in this research were reviewed. Real-time multiple-agent system simulation architecture was selected considering time and money resource, development cost, and requirements for simulation fidelity. Then the approach of implementing the lost-cost effective agent-based real-time simulation with Matlab toolset was provided. Finally, numeral results obtained from Real-time multiple-agent system simulation for static and dynamic cases were presented.

It is concluded from the numerical results presented in this dissertation that decentralized aircraft landing scheduling at non-controlled airports can be achieved with acceptable performance using the on-board aircraft landing scheduling tool.

B. Future Work

The decentralized aircraft landing scheduling problem has not been clearly stated in the literature by far prior to the advent of this dissertation. It is an important problem deserving of closer attention since it provides a clear approach to the distributed air traffic management system. Current operations at the terminal area of non-controlled airports have no centralized control, which exhibit an inherent property of distribution that gives a perfect environment for the implementation of uninterrupted free-flight concept. In the research addressed in this dissertation, the on-board aircraft landing scheduling tool developed represents uninterrupted free-flight to the threshold since there is no ground-based automated system to enforce any centralized control and flight crews take over all of the responsibilities that the current controllers have. Aircraft at the terminal area of non-controlled airports are placed in a complete decentralized environment, and it is author's believe that the research addressed in this dissertation will contribute to the future ATM revolution since the research will give a clear view of where to establish the line of free-flight concept application for the controlled airport case.

At this stage the potential impact of Free flight on the operations of the national airspace system is still disputed, and demand measurement for SATS shows that NASA could possibly introduce an idea to the public that would never be used. However, it can be expected that the future air traffic management system will manage flight operations in a way that lies somewhere between the two extremes, fully centralized and uninterrupted free-flight, possibly moving gradually from centralized to more free-flight, as the concept of Highway-in-the-Sky emerges. It's then authors' interest to extend the non-controlled airport application to controlled airport case with suitable amendment, where the reliance on centralized air traffic management can be reduced gradually in favor of a decentralized management to provide more airspace capacity, flight flexibility, and increase operation robustness. It has a promising future, at least the road worthy a try. While at this point the concept of Free Flight is sort of stuck at nowhere, the approach of using event-based coordination model to tackle the decentralized aircraft landing

scheduling problem at non-controlled airport might have a chance to “break the ice”, and bring the DAT-TM to the next level.

REFERENCES

[1]Nolan, S.M., Fundamentals of Air Traffic Control, Third ed, Wadsworth Publishing Company, Belmont, CA, 2003.

[2]Lucio, B., Paolo, D., and Amedo, R.O., Modeling and Simulation in Air Traffic Management, Springer, New York, 1997, p. 202.

[3]Conway, R.S., and Consiglio, M., "A Method of Separation Assurance for Instrument Flight Procedures at Non-radar Airports," AIAA-2002-4448, *AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, 2002.

[4]Croft, J., Small Aircrafts-To Be or Not To Be?, Aviation Week & Space Technology, Accessed 12, April, 2002, http://www.aviationnow.com/.

[5]2010 Concepts of Operations Document and Small Aircraft Transportation System: Systems Development and Evaluation Plan, NASA Langley Research Center, Hampton, VA, 2002.

[6]Chen, H., and Zhao, Y., "A New Queuing Model for Aircraft Landing Process," AIAA-1997-3737, *AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, 1997.

[7]Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., and Abramson, D., "Scheduling Aircraft Landings - the Static Case," *Transportation Science*, Vol. 34, No. 2, 2000, pp. 180-197.

[8]Denery, D.G., and Erzberger, H., The Center-TRACON Automation System: Simulation and Field Testing, NASA Ames Research Center, Moffett Field, CA, August, 1995.

[9]Davis, T.J., Erzberger, H., Green, S. M., and Nedell, W., "Design and Evaluation of Air Traffic Control Final Approach Spacing Tool," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 4, 1991, pp. 848-854.

[10]Davis, T.J., Erzeczowski, K. J., and Bergh, C., "The Final Approach Spacing Tool," *13th IFAC Symposium on Automatic Control in Aerospace*, Palo Alto, CA, 1994.

[11]Aeronautical Information Manual, FAA, Washington, D.C., 2006.

[12]Remington, R., and Johnston, C.J., Simulation Experiments Investigating Controller Workload Under Free-Flight Conditions, NASA Ames Research Center, Moffett Field, CA, June, 1995.

[13]Jones, K., Williams, D., Consiglio, M., Adams, C., and Abbott, T., IFR Operations at Non-towered, Non-radar Airports: Can We Do Better Than One-at-a-time?, NASA Langley Research Center, Hampton, VA, 2003.

[14]Ding, Y.Y., Rong, J., and Valasek, J., "Automation Capabilities Analysis Methodology for Non-Controlled Airports," AIAA-2003-5601, *AIAA Modeling, Simulation Technologies Conference*, Austin, TX, 2003.

[15]Drexl, A., and Jordon, C., "A Comparison of Constraint and Mixed-Integer Programming Solvers for Batch Sequencing with Sequence-Dependent Setups," *ORSA J. Comput.*, Vol. 7, 1995, pp. 160-165.

[16]Bianco, L., Dell'Olmo, P., and Giordani, S., "Minimizing Total Completion Time Subject to Release Dates and Sequence Dependent Processing Times, In Advances in Combinatorial Optimization," *Annals of Operation Research*, Vol. 86, 1999, pp. 393-415.

[17]Bianco, L., Ricciardelli, S., Rinaldi, G., and Sassano, A., "Scheduling Tasks with Sequence-Dependent Processing Times," *Naval Res. Logist*, Vol. 35, 1988, pp. 177-184.

[18]Guinet, A., "Scheduling Sequence-Dependent Jobs on Identical Parallel Machines to Minimize Completion Time Criteria," *Int. J. Product. Res.* , Vol. 31, No. 7, 1993, pp. 1579-1594.

[19]Pinedo, M., and Young, H.L., "Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup-times," *Eur.J.Oper.Res*, Vol. 100, No. 3, 1997, pp. 464-474.

[20]Low, C.Y., "Job Shop Scheduling Heuristics for Sequence Dependent Setups," *Comput. Industr. Eng.*, Vol. 29, No. 1-4, 1995, pp. 279-283.

[21]Psaraftis, H.N., "A Dynamic Programming Approach for Sequencing Groups of Identical Jobs," *Operations Research*, Vol. 28, No. 6, 1980, pp. 1347-1359.

[22]Thiele, O., and Brucker, P., "A Branch & Bound Method for the General-Shop Problem with Sequence Dependent Setup-times," *OR Spektrum*, Vol. 18, No. 3, 1996, pp. 145-161.

[23]Dear, R.G., and Sherif, Y.S., "The Dynamic Scheduling of Aircraft in High Density Terminal Areas," *Microelectronics and Reliability*, Vol. 29, No. 5, 1989, pp. 743-749.

[24]Ernst, A.T., Krishnamoorthy, M., and Storer, R.H., "Heuristic and Exact Algorithms for Scheduling Aircraft Landings," *Networks*, Vol. 34, No. 3, 1999, pp. 229-241.

[25]Bianco, L., Rinaldi, G., and Sassano, A., "A Combinatorial Optimization Approach to Aircraft Sequencing Problem," *Computer and Systems Science*, Vol. 38, 1987, pp. 324-339.

[26]Psaraftis, H.N., A Dynamic Programming Approach To The Aircraft Sequencing Problem, Flight Transportation Laboratory, MIT, Cambridge, MA, 1978.

[27]Brinton, C.R., "An Implicit Enumeration Algorithm for Arrival Aircraft Scheduling," *Proceedings of 11th IEEE/AIAA Digital Avionics Systems Conference*, Seattle, Washington, 1992, pp. 268-274.

[28]Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., and Abramson, D., "The Displacement Problem and Dynamically Scheduling Aircraft Landings," *Journal of the Operational Research Society*, Vol. 55, No. 1, 2004, pp. 54-64.

[29]Isaacson, D.R., Davis, T.J., and Robinson III, J.E., "Knowledge-Based Runway Assignment for Arrival Aircraft in the Terminal Area," *AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, 1997.

[30]Krzeczowski, K.J., Davis, T.J., Erzberger, Lev-Ram, I., and Bergh, C.P., "Knowledge-Based Scheduling of Arrival Aircraft in the Terminal Area," AIAA-95-3366, *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Baltimore, MD, 1995, pp. 1758-1768.

[31]Lee, K.K., and Davis, T.J., "The Development of the Final Approach Spacing Tool(FAST): A Cooperative Controller-Engineer Design Approach," *Journal of Control Engineering Practice*, Vol. 4, No. 8, 1996, pp. 1161-1168.

[32]Robinson III, J.E., and Isaacson, D.R., "A Concurrent Sequencing and Deconfliction Algorithm for Terminal Area Air Traffic Control," *AIAA Guidance, Navigation, and Control Conference*, Denver, CO, 2000.

[33]Robinson III, J.E., Davis, T.J., and Isaacson, D.R., "Fuzzy Reasoning-Based Sequencing of Arrival Aircraft in the Terminal Area," *AIAA Guidance, Navigation and Control Conference*, New Orleans, LA, 1997.

[34]Brucker, P., Scheduling Algorithms, Third ed, Springer, New York, 2001.

[35]Ding, Y.Y., and Valasek, J., "Aircraft Landing Scheduling Optimization for Single Runway Non-Controlled Airports---the Static Case," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 252-255.

[36]Sorensen, J.A., Detailed Description for CE-11 Terminal Arrival: Self Spacing for Merging and In-trail Separation, Technical Research in Advanced Air Transportation Technologies, NASA Ames Research Center, Moffett Field, CA, 2000.

[37]Barmore, B., and Abbott, T., "Airborne-managed Spacing in Multiple Arrival Streams," *24th International Congress of the Aeronautical Science*, Yokohama, Japan, 2004.

[38]Palmer, M.T., Barmore, B.E., and Abbott, T.S., "Integration of Paired-Dependent Speed Guidance into Current-Generation Glass-Cockpit Commercial Aircraft," *24th International Congress of the Aeronautical Science*, Yokohama, Japan, 2004.

[39]Bussink, F.J., Doble, N., Barmore, B., and Singer, S., "A Fast-Time Simulation Environment for Airborne Merging and Spacing Research," *23rd Digital Avionics Systems Conference*, Salt Lake City, UT, 2004.

[40]Clavier, O., Houck, S., Schleicher, D., and Davis, P. "The Smart Airport Automation System (SAASY)," *AIAA Guidance, Navigation, and Control Conference*, Austin, TX, 2003.

[41]Abbott, T., Jones, K.M., Consiglio, M., Williams, D.M., and Adams, C., Development of the SATS HVO Operational Concept: Nominal Operations, NASA Langley Research Center, Hampton, VA, 2003.

[42]Schleicher, D., Sorensen, J., and Peters, M., "The Past, Present, and Future of Small Airport Automation," AIAA-2003-6792, *AIAA 3rd Annual Aviation Technology, Integration, and Operations (ATIO) Forum*, Denver, CO, 2003.

[43]Ballin, M., Hoekstra, J., Wing, D., and Lohr, G. "NASA Langley and NLR Research of Distributed Air/Ground Traffic Management," AIAA-2002-5826, *AIAA Aircraft Technology, Integration, and Operations Conference*, Los Angeles, CA, 2002.

[44]Prevot, T., Palmer, E.A., Smith, N., and Callantine, T., "Future Air Traffic Management: A Perspective on Distributed Automation," *CSAPC '01 8th Conference on Cognitive Science Approaches to Process Control*, Munich, 2001.

[45]Prevot, T., Shelden, S., Palmer, E., Johnson, W., Battiste, V., Smith, N., Callantine, T., Lee, P.U., and Mercer, J. "Distributed Air/Ground Traffic Management Simulation: Results, Progress and Plans," AIAA-2003-5602, *AIAA Modeling and Simulation Technologies Conference*, Austin, TX, 2003.

[46]Gerkey, B., Jones, C., Shell, D., and Mataric, M.J., "Principled Approaches to the Design of Multi-Robot Systems," *Proceedings of Workshop on Networked Robotics, IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 71-80.

[47]Muhl, G., Large-Scale Content-Based Publish/Subscribe Systems, Ph.D. Dissertation, Darmstadt University of Technology, Darmstadt, Germany, 2002, p. 175.

[48]Birman, K.P., "The Process Group Approach to Reliable Distributed Computing," *Communication of the ACM*, Vol. 36, No. 12, 1993, pp. 37-53.

[49]Skeen, D., "An Information Bus Architecture for Large-Scale, Decision-Support Environments," *ACM SIGOPS Operating System Review*, Vol. 27, No. 5, 1992, pp. 58-58.

[50]Kuchar, J.K., and Yang, L.C., "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4, 2000, pp. 179-189.

[51]Harper, K., Mulfund, S., Guarino, S., Mehta, A., and Zacharias, G., "Air Traffic Controller Agent Model for Free Flight," AIAA-99-3987, *AIAA Guidance, Navigation and Control Conference*, Portland, OR, 1999.

[52]Rong, J., Geng, S., Valasek, J., and Ioerger, T., "Air Traffic Conflict Negotiation and Resolution Using An Onboard Multi-Agent System," DASC-345, *21st Digital Avionics Systems Conference (DASC) on Air Traffic Management for Commercial and Military Systems*, Irvine, CA, 2002.

[53]Helbing, K., Spaeth, T., and Valasek, J., "Improving Aircraft Sequencing and Separation at a Small Aircraft Transportation System Airport," *Journal of Aircraft*, Vol. 43, No. 6, 2006, pp. 1636-1642.

[54]Hillier, F.S., and Lieberman, G.J., Introduction to Operations Research, 8th ed, McGraw-Hill, Columbus, OH, 2001.

[55]Texas A&M University Flight Simulation Laboratory, College Station, TX.

[56]Engineering Flight Simulator System Description (Version 1.1), Aerospace Engineering Department, Texas A&M University, College Station, TX, 2000.

[57]Painter, J., Ward, D., Crump, J., Trang, J.A., Lee, K.A., et. al. "Decision Support for the General Aviation Pilot," *Proceedings of the 1997 IEEE International Conference on System, Man, and Cybernetics*, Orlando, FL, 1997, pp. 88-93.

[58]Online help document for Matlab Version 2006b, 2006, http://www.mathworks.com/.

[59]Rong, J., Spaeth, T., and Valasek, J., "Small Aircraft Pilot Assistant: Onboard Decision Support System for SATS Aircraft," AIAA-2005-7382, *AIAA 5th ATIO and 16th Lighter-than-Air and Balloon Systems Conferences*, Arlington, VA, 2005.

VITA

Name: Yuanyuan Ding

Address: Department of Aerospace Engineering, H.R. Bright Building, Rm. 701,

Ross Street - TAMU 3141, College Station TX 77843-3141

Email Address: dingyuanyuanpl@hotmail.com

Education: B.S., Manufacturing Engineering,
Beijing University of Aeronautics & Astronautics, Beijing, China, July 1998

M.S., Mechanical and Electronic Engineering,
Beijing University of Aeronautics & Astronautics, Beijing, China, May 2001

Ph.D., Aerospace Engineering,
Texas A&M University, College Station, Texas, May 2007

Professional: Senior Project Engineer

S-TEC Corporation, One S-TEC Way, Mineral Wells, TX 76067

(Phone) 817-215-7654, (Fax) 940-325-3904

<http://www.s-tec.com/>